Оптимізація розв'язків задачі комівояжера методом послідовного сканування

Базилевич Роман Петрович, Кутельмах Роман Корнелійович

НУ "Львівська політехніка",

кафедра програмного забезпечення

Вступ

Задача комівояжера – одна і базових задач комбінаторної оптимізації, що має широке прикладне застосування [1,2].

Існує небагато алгоритмів, що забезпечують одержання якісних розв'язків задачі комівояжера [3]. Для розв'язування задачі комівояжера алгоритм Ліна-Кернігана є одним з найефективніших [4,5]. Його обчислювальна складність – $O(n^2)$. Одержані результати – в межах 1-3% від оптимального [3]. Впродовж останніх років було одержано нову версію алгоритму Ліна-Кернігана – алгоритм Ліна-Кернігана-Гельсгауна [6]. Він забезпечує знаходження оптимального розв'язку задачі для 7397 точок із бібліотеки тестів для транспортних задач – TSPLIB [7]. Як показали результати тестування існуючих методів розв'язування задачі комівояжера DIMACS TSP Challenge [3], він є найкращим евристичним алгоритмом[3,8]. Обчислювальна складність алгоритму - $O(n^{2,2})$. Групою вчених [9-12] розроблено пакет програмного забезпечення для точного розв'язування задачі комівояжера – Concodre TSP Solver [13].

1. Формулювання задачі

В класичному формулюванні задача комівояжера представляється як граф $G=(V,E)$, де V – множина вершин графа, а E – множина його ребер. Вага (або довжина) $c_{i,j}$ кожного ребра $e_{i,j} \in E$ вважається заданою. Задача вважається симетричною, якщо $c_{i,j} = c_{j,i}$, $\forall i,j \in V$. Крім того, задачу вважають Евклідовою при умові, якщо $c_{i,j} + c_{j,k} \geq c_{i,k}$ $\forall i,j,k \in V$. Необхідно знайти Гамільтонів цикл мінімальної ваги, де Гамільтонів цикл – це закритий цикл у графі, що включає усі вершини та передбачає відвідування кожної вершини лише один раз.

Розглядатиметься симетрична Евклідова задача комівояжера, де заданими вважають множину N з n точок ($|N|=n$), які описані їх координатами (x_i, y_i). Необхідно найти маршрут S^*, що проходить по одному разу через кожну точку, довжина якого $L^*(S^*)$ є мінімальною:

$$L^*(S^*) = \Sigma_{ij} l_{ij}^* \rightarrow min\ \Sigma_{ij} l_{ij}^* \quad \forall l_{ij}^* \in l_{ij}',$$

де l_{ij}' – деяка з допустимих за заданими обмеженнями ділянка між двома суміжними точками i та j виділеного маршруту.

Декомпозиційні алгоритми для розв'язування задачі комівояжера досліджувалися тривалий час. Інформація про існуючі декомпозиційні алгоритми, що забезпечують високу якість розв'язку, відсутня.

Запропоновані декомпозиційні алгоритми передбачають розбиття вхідної множини точок на підмножини ($N_0, N_1,..., N_k \in N$). Кожна підмножина $N_i \in N$ описується у вигляді однієї умовної точки. Відстань між підмножинами визначається як довжина найкоротшого ребра, що з'єднує дві підмножини, або як відстань між умовними центрами підмножин. Формування підмножин здійснюється за допомогою декомпозиційних алгоритмів, що повинно суттєво зменшити час обчислень. Задача з n точками замінюється задачею з k точками. Звичайно $k<<n$, що суттєво спрощує задачу. Деякою вибраною базовою процедурою (алгоритмом) P_0 розв'язується класична задача комівояжера між підмножинами, яка визначає порядок обходу підмножин (макромаршрут). Маючи порядок обходу, визначаються граничні точки у підмножинах – точки, що з'єднують підмножини у макромаршруті. Наступний етап – формування початкового розв'язку. Початковий розв'язок задачі складається з об'єднання часткових розв'язків задачі у кожній підмножині. Після того, як початковий розв'язок знайдено, настає етап оптимізації маршруту.

Запропоновано новий спосіб оптимізації існуючого маршруту. Розглянемо його детальніше.

2. Оптимізація маршруту

Етап оптимізації забезпечує суттєве покращення розв'язку, одержаного як результат деякого швидкого, проте не ефективного евристичного алгоритму. Оптимізація початкового розв'язку призначена для загального покращення маршруту шляхом покращення його на окремих ділянках. Запропонований метод оптимізації має наступні характеристики:

- елементарна область оптимізації (область сканування) – розмір ділянки маршруту, що оптимізується;
- область перетину – розмір області стику двох сусідніх елементарних областей;
- базовий алгоритм – вибраний базовий алгоритм, застосований для оптимізації в заданій області сканування;
- кількість повторних проходжень – кількість повних циклів оптимізації для цілого маршруту.

Із зростанням розміру області сканування покращується якість розв'язку, проте також збільшується і час обчислень. Результати оптимізації також залежать від обраного базового алгоритму. Рекомендується використовувати ефективний та високоякісний алгоритм –

Ліна-Кернігана чи Ліна-Кернігана-Гельсгауна. При виборі в якості базового алгоритму 2-opt чи 3-opt, якість отриманого маршруту буде гіршою. Також результати експериментів показують, що при застосуванні декількох повторних проходжень оптимізаційної процедури відбувається нове покращення маршруту. При кількості повторних проходжень більше чотирьох, покращення маршруту вже майже не відбувається.

2.1. Послідовна локальна оптимізація маршруту (метод послідовного сканування)

Запропонований метод для оптимізації розв'язку передбачає покращення маршруту на його ділянках, що вибираються послідовно вздовж існуючого маршруту. Розмір ділянки задається розміром елементарної області.

В якості вхідних даних алгоритму подається маршрут, що потрібно оптимізувати та додаткові параметри, що включають:

- розмір області оптимізації (scanning area size - *SAsize*) – кількість точок, що входять до однієї області оптимізації;
- розмір області перетину (overlapping area size - *OAsize*) – кількість спільних точок, що належать двом "сусіднім" областям оптимізації;
- базовий алгоритм;

Процес оптимізації відбувається наступним чином. У вхідному маршруті вибирається його ділянка з кількістю точок *SAsize*. Перша та остання точки ділянки вважаються граничними. За допомогою певного вибраного базового алгоритму, розв'язується незамкнута задача комівояжера для множини точок ділянки із заданим умовним ребром (що з'єднує його граничні точки). Довжина отриманого маршруту порівнюється із довжиною існуючого і, якщо є покращення, маршрут замінюється на новий. Далі вибирається наступна ділянка з такою ж кількістю точок, але не з позиції, де закінчилася попередня ділянка, а відсунута на *OAsize* точок назад. Операція триває доти, поки не буде розглянуто усіх точок вхідного маршруту.

На наступному рисунку схематично показано послідовну локальну оптимізацію маршруту (рис. 1).

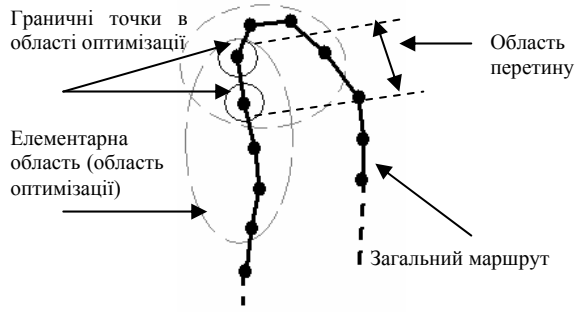

Рисунок 1: Послідовна локальна оптимізація маршруту.

Запропонований метод підходить для будь-якого вхідного маршруту, сформованого довільним чином. Найкращі результати очікуються при виборі алгоритму Ліна-Кернігана-Гельсгауна в якості базового. Проте можливе використання й інших алгоритмів, особливо для заданих малих по розміру областей сканування. Використання області перетину забезпечує вибір найкращого серед декількох маршрутів (іноді трьох чи чотирьох – якщо розмір області перетину досить великий) на вибраній ділянці. А це в свою чергу забезпечує кращі результати покращення маршруту загалом.

Основною перевагою даного методу можна вважати простоту його реалізації та достатньо високу якість розв'язку задачі. Основний недолік – алгоритм розглядає ділянки із точками, що розміщені одна за одною в існуючому маршруті. Можливий випадок, коли точки вхідного маршруту лежать близько одна до одної, проте в реальному маршруті між ними є великий проміжок і вони належать різним областям оптимізації. Тоді алгоритм не розгляне їх як точки у одній ділянці і, відповідно, маршрут не буде суттєво покращено у цій локальній області. Даний недолік особливо стосується задач великих розмірностей. Як би не зростала кількість точок, розмір елементарної області сканування не повинен зростати з такою ж пропорцією, бо тоді суттєво збільшаться часові затрати. А в задачах великої розмірності дуже часто між близькими точками є великий проміжок у порядку їх відвідування.

Розмір області перетину як параметр алгоритму рекомендовано задавати таким, щоб він становив 25-75% від розміру області сканування.

На наступному рисунку показано порівняння початкового маршруту та оптимізованого (рис. 2). Для задачі розмірністю 1000 точок заданий розмір області сканування 200 точок, а розмір області перетину - 50. Тобто, на кожному кроці виконання алгоритму, елементарна область сканування "просувається" на 150 точок. Зліва показано початковий маршрут, справа – оптимізований.

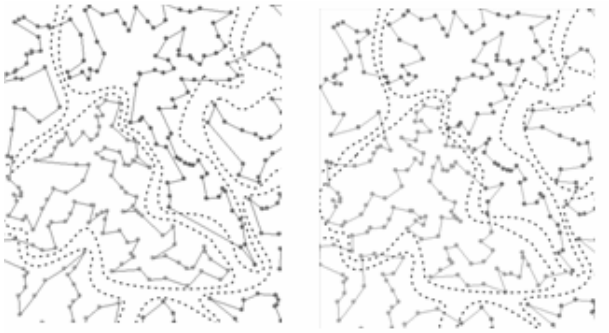

Рисунок 2: Порівняння початкового та оптимізованого маршрутів.

Після того як оптимізацію виконано, існують потенційно неоптимальні ділянки, де маршрути повинні бути покращені. Причина утворення таких ділянок описувалась вище – точки, розміщені близько, можуть не попадати в одну область сканування.

Після того, як маршрут покращено, є можливість його повторної оптимізації за допомогою того ж алгоритму, проте з іншими параметрами. В такому випадку очікується ще деяке покращення загального розв'язку.

3. Експерименти

Проведено тести для задач із різними розмірностями – 1000, 2000, 5000 та 10000 точок. Розподіл точок у тестах – довільний. Розмір усіх підмножин був обмеженим по кількості точок – до 250. Тести проводилися для порівняння часу та якості декомпозиційних алгоритмів у порівнянні із базовим без декомпозиції. У якості базового було обрано алгоритм Ліна-Кернігана-Гельсгауна.

Було досліджено як впливає розмір елементарної області сканування на якість оптимізації, а також як впливає розмір області перетину. Для задач усіх розмірностей задавалися наступні значення розміру області сканування і перетину (у дужках зазначено розмір області перетину): 100(30), 100(50), 100(70), 200(30), 200(50), 200(100), 300(30), 300(50), 300(100), 300(200), 400(30), 400(50), 400(100), 400(200), 400(300).

Результати тестів задач різних розмірностей в таблицях.

Таблиця 1: Результати тестування задач розмірністю 1000 точок

Розмірність – 1000		
	Час, с	Довжина
Базовий алгоритм	21	0,00%
Без оптимізації	10	3,31%
З оптимізацією 100(30)	13	2,69%
100(50)	13	2,57%
100(70)	16	2,36%
200(30)	16	2,02%
200(50)	17	2,50%
200(100)	21	1,07%
300(30)	25	1,09%
300(50)	22	0,82%
300(100)	25	0,95%
300(200)	36	0,85%
400(30)	27	1,47%
400(50)	34	1,43%
400(100)	34	0,44%
400(200)	35	0,53%
400(300)	57	0,25%

Таблиця 2: Результати тестування задач розмірністю 2000 точок

Розмірність – 2000		
	Час, с	Довжина
Базовий алгоритм	187	0,00%
Без оптимізації	21	3,67%
З оптимізацією 100(30)	26	3,20%
100(50)	28	3,19%
100(70)	33	3,12%
200(30)	43	2,70%
200(50)	35	2,99%
200(100)	37	2,70%
300(30)	42	2,48%
300(50)	50	2,29%
300(100)	61	2,29%
300(200)	77	1,84%
400(30)	52	2,46%
400(50)	59	2,06%
400(100)	72	1,92%
400(200)	89	0,99%
400(300)	127	1,25%

Таблиця 3: Результати тестування задач розмірністю 5000 точок

Розмірність – 5000		
	Час, с	Довжина
Базовий алгоритм	3829	0,00%
Без оптимізації	51	4,94%
З оптимізацією 100(30)	62	4,28%
100(50)	67	4,27%
100(70)	79	4,19%
200(30)	85	3,68%
200(50)	89	3,56%
200(100)	122	2,99%
300(30)	113	2,88%
300(50)	100	2,88%
300(100)	122	2,65%
300(200)	201	2,28%
400(30)	130	2,33%
400(50)	170	2,43%
400(100)	157	2,44%
400(200)	181	1,93%
400(300)	339	1,75%

Таблиця 4: Результати тестування задач розмірністю 10000 точок

Розмірність – 10000		
	Час, с	Довжина
Базовий алгоритм	16646	0,00%
Без оптимізації	104	5,22%
З оптимізацією 100(30)	129	4,72%
100(50)	137	4,62%
100(70)	157	4,53%
200(30)	175	4,21%

200(50)	168	4,14%
200(100)	200	3,71%
300(30)	223	3,57%
300(50)	227	3,45%
300(100)	245	3,34%
300(200)	381	2,96%
400(30)	301	3,26%
400(50)	283	3,13%
400(100)	330	3,04%
400(200)	377	2,88%
400(300)	655	2,63%

4. Висновки

Запропонований алгоритми оптимізації розв'язків задачі комівояжера демонструє погіршення якості маршруту в межах 1-3 % (при достатньо великих розмірах області сканування та перетину) по відношенню до базового алгоритму без декомпозиції (використовувався найкращий на даний час алгоритм, що забезпечує якість маршруту, в межах 1% від оптимального). При втраті якості на 2% спостерігається виграш у часі (у 30 разів для задачі розмірністю 10000 точок), що вказує на доцільність використання алгоритмів для задач великих розмірностей. Із різким збільшенням розмірності задачі не спостерігається різке збільшення втрати якості при однакових параметрах алгоритму оптимізації.

5. Список літератури

[1] Reinelt, Gerhard (1994), The Traveling Salesman: Computational Solutions for TSP Applications. Lecture Notes in Computer Science 840, Springer–Verlag, Berlin.

[2] Reinelt, Gerhard (1992), Fast heuristics for large geometric traveling salesman problems. ORSA Journal on computing, 4:206-217

[3] David S. Johnson and Lyle A. McGeoch. Experimental Analysis of Heuristics for the STSP. In Gutin and Punnen, editors, The Traveling Salesman Problem and its Variations. Kluwer Academic Publishers, 2002.

[4] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the Traveling salesman problern. Operations Research, 21:498–516. 1973.

[5] S. Lin. Computer solutions of the travelling salesman problem. Bell System Technical Journal 44, pages 2245–2269, 1965.

[6] K .Helsgaun, "An effective implementation of the Lin–Kernighan Traveling Salesman Heuristic", 2002.

[7] Reinelt, Gerhard(1991) TSPLIB – A traveling salesman problem library, ORSA Journal on Computing 3, 376–384.

[8] http://www.research.att.com/~dsj/chtsp/

[9] D. Applegate, R.E. Bixby, V. Chvátal, and W. Cook. On the solution of traveling salesman problems. Documenta Mathematica, Extra Volume ICM III:645–656, 1998.

[10] D. Applegate, W. Cook, A. Rohe. Chained Lin–Kernighan for large traveling salesman problems. INFORMS J. Computing, to appear.

[11] D. Applegate, R.E. Bixby, V. Chvátal, and W. Cook. Findong tours in the TSP.1998.

[12] D. Applegate, R.E. Bixby, V. Chvátal, and W. Cook. Findong cuts in the TSP.1995

[14] http://www.tsp.gatech.edu/concorde.html

[15] D. Neto. Efficient cluster compensation for Lin–Kernighan Heuristics. PhD thesis, Department of Computer Science, University of Toronto, 1999.

[16] G. Laporte, J-Y. Potvin, F. Quilleret, "A Tabu Search using Genetic Diversification for the Clustered Traveling Salesman Problem", Journal of Heuristics, Vol 2 (3), p. 187-200, 1996.

[17] Базилевич Р.П., Кутельмах Р.К., "Алгоритми динамічного формування моделі робочого поля для задачі комівояжера з кластерним розподілом точок", Вісник НУ "Львівська політехніка", Львів, 2006.

[18] Базилевич Р.П., Ремі Дюпа, Кутельмах Р.К., "Використання алгоритмів локальної оптимізації для розв'язування задачі комівояжера з кластерним розподілом точок", Вісник НУ "Львівська політехніка", Львів, 2006.

[19] Bazylevych R., Dupas R, Kutelmakh R., "Scanning-area algorithms for clustered TSP", Proceedings of International conference "Comp. science and Information Technologies", Lviv, Polytechnic University, 2006, pp. 148-152.

[20] Held, M., and Karp, R. M. (1970), "The Traveling Salesman Problem and Minimum Spanning Trees", Operations Research. 18:1138–1162.