Висновки

Розроблено алгоритм для кластеризації даних, що ґрунтуються на агломеративному ієрархічному підході. Реалізовано нову ідею для контролю алгоритмічної точності та складності за коефіцієнтом швидкості. Створено програмний пакет для кластеризації даних. Проведені експериментальні дослідження для великих вибірок даних підтверджують ефективність запропонованої ідеї: зменшення кількості об'єднань кластерів на рівнях дерева згортання без втрат алгоритмічної точності. Зменшення алгоритмічної складності дає змогу застосовувати алгоритм для великих груп вибірок, таких як візуальні образи, гени або тексти.

1. Andy M Yip, Chris Ding, Tony F.Chan. Dynamic Cluster Formation Using Level Set Methods. – IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.28, n. 6, pp.877-889, June, 2006. 2. Leo Grady, Eric L.Schwartz. Isoperimetric Graph partitioning for Image segmentation. – IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.28, n. 3, pp.469-475, March, 2006. 3. M. Pavan, M. Pelillo. Dominant sets and Pairwise Clustering. – IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.29, n. 1, pp.167-172, January, 2007. 4. C. Ding, X. He. Cluster Aggregate Inequality and Multilevel Hierarchical Clustering, Proc. 9th European Conf. Principles of Data Mining and Knowledge Discovery (2005). p. 71−83. 5. Мельник Р.А., Алексєєв О.А. Кластеризація ключів образів на основі декомпозиції їх множини // Відбір і обробка інформації. – 2006. – Вип. 24(100). – С.110–114. 6. Р. Мельник, Р. Тушницький. Каскадна декомпозиція множин великої розмірності при кластеризації ключів образів // "Комп'ютерні науки та інформаційні технології". – 2008. – № 604. – С. 249–254.

УДК 519.16

Р. Базилевич, Р. Кутельмах
Національний університет "Львівська політехніка",
кафедра програмного забезпечення

ОПТИМІЗАЦІЯ РОЗВ'ЯЗКІВ ЗАДАЧІ КОМІВОЯЖЕРА МЕТОДОМ ПОСЛІДОВНОГО СКАНУВАННЯ

Запропоновано новий метод оптимізації розв'язків задачі комівояжера. Метод може бути застосований для оптимізації початкового розв'язку задачі, отриманого за допомогою декомпозиції чи для покращення маршруту, отриманого будь-яким алгоритмом. Вхідними даними є маршрут, який необхідно покращити.

New approach for Traveling Salesman Problem(TSP) solutions optimization is proposed. Approach can be applied for initial solution optimization, calculated with the help of decomposition algorithm or for route optimization, calculated by any classic algorithm. Route to be improved is an input data for algorithm.

Вступ

Задача комівояжера – одна із основних задач комбінаторної оптимізації, що має широке прикладне застосування [1,2]. Існує небагато алгоритмів, що забезпечують одержання якісних розв'язків задачі комівояжера, особливо при малих часових затратах [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. Впродовж останніх років було запропоновано нову версію алгоритму Ліна–Кернігана – алгоритм Ліна–Кернігана–Гельсгауна [6], який забезпечує отримання 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 кращим 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]. Він забезпечив одержання оптимальних розв'язків для усіх тестів із бібліотеки TSPLIB, розмірністю 85900 точок. Розв'язання задачі з кількістю 85900 точок зайняло майже 136 років процесорного часу (тести проводились на кластері з ПК з процесорами Intel Xeon та AMD Opteron).

1. Формулювання задачі

За класичним формулюванням задача комівояжера представляється як граф $G=(V,E)$, де V – множина вершин, а E – множина його ребер. Вага (або довжина) 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 за умови, якщо $c_{i,j} + c_{j,k} \geq c_{i,k} \; \forall i,j,k \in V$. Необхідно знайти Гамільтонів цикл мінімальної ваги, тобто закритий цикл у графі, що містить 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:

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

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

Вважається, що існуючий маршрут одержано за допомогою декомпозиції чи деякого класичного алгоритму. Запропоновано новий метод оптимізації існуючого маршруту.

2. Оптимізація маршруту

Етап оптимізації забезпечує істотне покращання розв'язку, одержаного як результат деякого швидкого алгоритму. Із зростанням розміру області сканування (області оптимізації) покращується якість розв'язку, проте також збільшується і час обчислень. Результати оптимізації 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 чергове покращання маршруту. При кількості повторних проходжень більше чотирьох, покращання маршруту вже є незначним, або взагалі відсутнє.

2.1. Послідовна локальна оптимізація маршруту (метод послідовного сканування)

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

Як вхідні дані алгоритму подається маршрут, що потрібно оптимізувати, та додаткові параметри, що містять:

1. Розмір області оптимізації (scanning area size – SA_{size}) – кількість точок, що входять до однієї області оптимізації.
2. Розмір області перетину (overlapping area size – OA_{size}) – кількість спільних точок, що належать двом "сусіднім" областям оптимізації.
3. Базовий алгоритм – вибраний базовий алгоритм, застосований для оптимізації в заданій області сканування.

4. Кількість повторних проходжень – кількість повних циклів оптимізації для всього маршруту.

Процес оптимізації є таким. У вхідному маршруті вибирається його ділянка з кількістю точок SA_{size}. Перша та остання точки ділянки вважаються граничними. За допомогою певного вибраного базового алгоритму розв'язується незамкнена 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 покращання, маршрут замінюється на новий. Далі вибирається наступна ділянка з такою самою кількістю точок, але не з позиції, де закінчилася попередня ділянка, а відсунута на OA_{size} точок назад. Операція триває доти, поки не будуть розглянуті усі точки вхідного маршруту.

На рис. 1 схематично показано послідовну локальну оптимізацію маршруту.

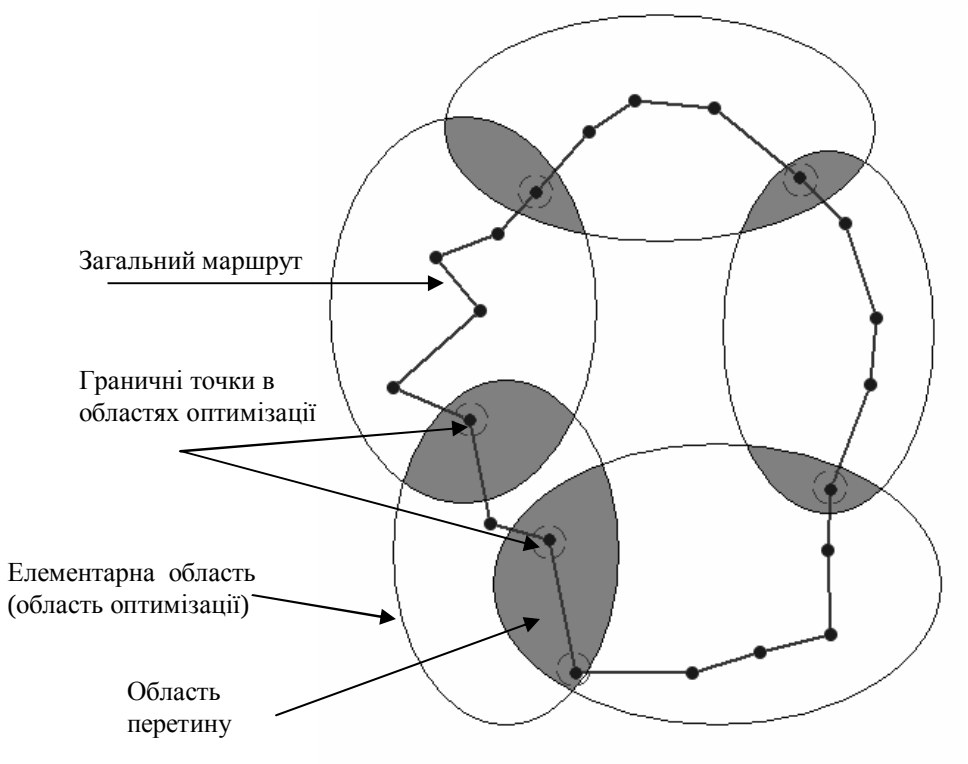

Рис. 1. Послідовна локальна оптимізація маршруту

Запропонований метод підходить для будь-якого маршруту, сформованого довільно. Найкращих результатів очікують при виборі алгоритму *Ліна–Кернігана–Гельсгауна* як базового. Проте можливе використання й інших алгоритмів, особливо для заданих малих за розміром 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, своєю чергою, покращує маршрут загалом.

Основною перевагою цього методу можна вважати простоту його реалізації та достатньо високу якість розв'язку. Основний недолік – алгоритм розглядає ділянки із точками, що розміщені одна за однією в існуючому маршруті. Можливий випадок, коли точки вхідного маршруту лежать близько одна до однієї, проте в реальному маршруті між ними є великий проміжок і вони належать різним областям оптимізації. Тоді алгоритм не розглядатиме їх як точки в одній ділянці і, відповідно, маршрут не буде покращано у цій локальній області. Цей 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 істотно збільшаться часові затрати. А в задачах великої розмірності дуже часто між близькими точками є великий проміжок у порядку

їх відвідування. На рис. 2 показано приклад такої ситуації. У обведеному колі показано точки, що знаходяться близько одна до однієї, проте належать різним областям оптимізації.

Розмір області перетину як параметр алгоритму рекомендовано задавати таким, щоб він становив 25–50% від розміру області сканування.

Після того, як оптимізацію виконано запропонованим алгоритмом, існують потенційно неоптимальні ділянки, де маршрути мають бути покращені (рис. 3).

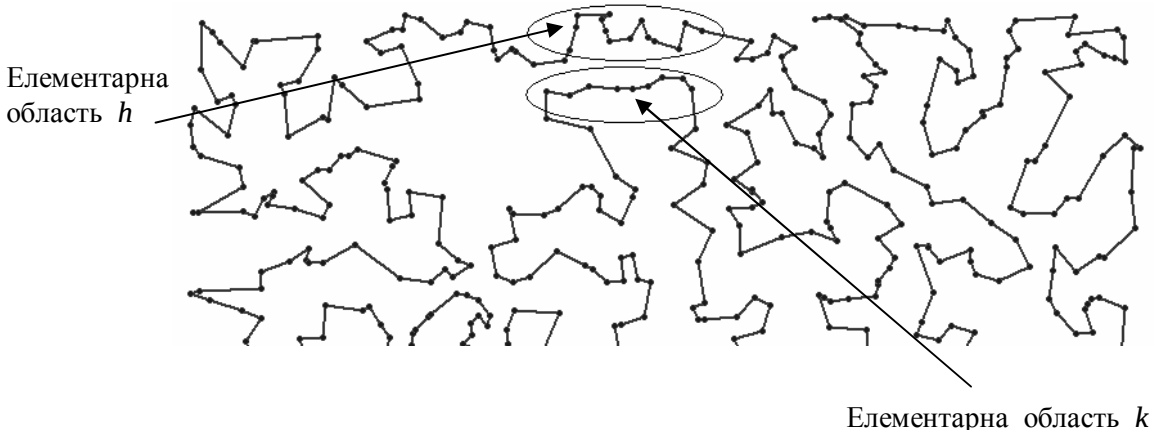

Рис. 2. Точки, що знаходяться близько одна до однієї, проте належать різним елементарним областям сканування

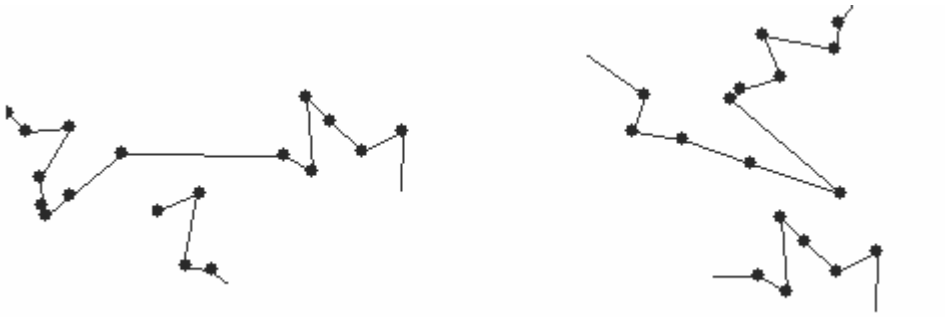

Рис. 3. Ділянки оптимізованого маршруту, що потребують покращання

Причина утворення таких ділянок: це існування точок, розміщених близько, проте в різних областях сканування. Є також можливість повторної оптимізації маршруту за допомогою того самого алгоритму, проте з іншими параметрами. У такому випадку очікується ще деяке покращання загального розв'язку.

3. Експерименти

Проведено тестування для задач із різними розмірностями – 1000, 2000, 5000 та 10000 точок. Розподіл точок у тестах – довільний. Розмір усіх підмножин був обмеженим за кількістю точок – до 250. Тести проводилися для порівняння часу та якості декомпозиційних алгоритмів порівняно із базовим без декомпозиції. Базовим було обрано алгоритм Ліна–Кернігана–Гельсгауна.

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 розмірністю 1000 точок наведені у табл. 1.

Таблиця 1

Результати тестування задач розмірністю 1000 точок

Розмірність – 1000		
	Час виконання, с	Довжина маршруту
1	2	3
Базовий алгоритм	21	0,00%
Декомпозиційний 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%

У табл.2 наведено результати тестування задач розмірністю 2000 точок.

Таблиця 2

Результати тестування задач розмірністю 2000 точок

Розмірність – 2000		
	Час виконання, с	Довжина маршруту
Базовий алгоритм	187	0,00%
Декомпозиційний 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%

Результати тестів задач розмірністю 5000 точок зібрані у табл. 3.

Таблиця 3

Результати тестування задач розмірністю 5000 точок

Розмірність – 5000		
	Час виконання, с	Довжина маршруту
1	2	3
Базовий алгоритм	3829	0,00%
Декомпозиційний 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%

У табл. 4 наведено результати тестування задач розмірністю 10000 точок.

Таблиця 4

Результати тестування задач розмірністю 10000 точок

Розмірність – 10000		
	Час виконання, с	Довжина маршруту
Базовий алгоритм	16646	0,00%
Декомпозиційний 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

Запропоновані алгоритми оптимізації розв'язків задачі комівояжера демонструють погіршення якості маршруту в межах 1–3 % (при достатньо великих розмірах області сканування та перетину) відносно базового алгоритму без декомпозиції (використовувався найкращий на сьогодні 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 різкого збільшення втрати якості при однакових параметрах алгоритму оптимізації.

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. Chvätal, and W. Cook. Findong tours in the TSP.1998. 12. D. Applegate, R.E. Bixby, V. Chvätal, and W. Cook. Findong cuts in the TSP.1995 13. http://www.tsp.gatech.edu/concorde.html 14. D. Neto. Efficient cluster compensation for Lin–Kernighan Heuristics. PhD thesis, Department of Computer Science, University of Toronto, 1999. 15. 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. 16. Базилевич Р.П., Кутельмах Р.К., Алгоритми динамічного формування моделі робочого поля для задачі комівояжера з кластерним розподілом точок // Вісник Нац. ун-ту "Львівська політехніка". – 2006. 17. Базилевич Р.П., Ремі Дюпа, Кутельмах Р.К., "Використання алгоритмів локальної оптимізації для розв'язування задачі комівояжера з кластерним розподілом точок" // Вісник Нац. ун-ту "Львівська політехніка". – 2006. 18. 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. 19. Held, M., and Karp, R. M. (1970), "The Traveling Salesman Problem and Minimum Spanning Trees", Operations Research. 18:1138–1162.