УДК 519.16

Р.П. БАЗИЛЕВИЧ, Р.К. КУТЕЛЬМАХ

Національний університет "Львівська політехніка", Львів, Україна

АЛГОРИТМ ОПТИМІЗАЦІЇ РОЗВ'ЯЗКІВ ЗАДАЧІ КОМІВОЯЖЕРА У ЛОКАЛЬНІЙ ОБЛАСТІ

Запропоновано новий метод оптимізації розв'язків Евклідової задачі комівояжера, що може бути застосований для покращення розв'язку задачі, отриманого за допомогою декомпозиції чи будь-якого швидкого евристичного алгоритму. Як базову процедуру для розв'язання задачі комівояжера алгоритм використовує один із найефективніших сучасних підходів – алгоритм Кельда Гельсгауна, що є модифікацією алгоритму Ліна-Кернігана.

Ключові слова: *задача комівояжера, декомпозиція, транспортна задача, NP-складність, сканувальна область, оптимізація, алгоритм Ліна-Кернігана, алгоритм Гельсгауна.*

Вступ

Алгоритми, що пов'язані з транспортними задачами, мають велике прикладне застосування [1,2] – окрім транспортних систем їх застосовують у системах телекомунікацій, виробництві друкованих плат, лазерній нарізці пластмас і металів і т.д. З часом вимоги до швидкості обчислень та якості зростають а також зростають розмірності задач, що приводить до необхідності розробки спеціалізованих алгоритмів для розв'язування задач великих розмірностей.

Задача комівояжера є однією з базових транспортних задач. 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 [3].

Впродовж останніх років одержано нову версію алгоритму Ліна-Кернігана – алгоритм Ліна-Кернігана-Гельсгауна [6]. Він забезпечує оптимальний розв'язок 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 найточнішим евристичним алгоритмом[3,8]. Обчислювальна складність алгоритму - $O(n^{2,2})$.

В статті дано опис алгоритму оптимізації розв'язків задачі комівояжера, досліджено результати тестування задач розмірностями 1000 та 10000 точок.

1. Формулювання задачі

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 є симетричною, якщо $c_{i,j} = c_{j,i}$, $\forall i,j \in V$. Задачу вважають Евклідовою при умові, якщо $c_{i,j} + c_{j,k} \geq c_{i,k}$ $\forall i,j,k \in V$. Необхідно знайти 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:

$$L^*(S^*) = \Sigma ij\ l_{ij}^* \rightarrow \min\ \Sigma ij\ l_{ij}^*\ \ \forall\ l_{ij}^* \in l_{ij}', \qquad (1)$$

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

Запропоновано новий спосіб оптимізації маршруту для Евклідової задачі комівояжера.

2. Оптимізація маршруту за допомогою алгоритму геометричного сканування

Метод полягає в оптимізації декількох маршрутів одночасно – частин загального маршруту, що знаходятся геометрично близько одна до одної. Певним чином вибирається геометрична область довільної форми (квадратна, прямокутна, кругла та ін.), у якій розпізнаються окремі ділянки загального маршруту, та за допомогою класичного алгоритму розв'язання задачі комівояжера, розв'язується зада-

ча сумарної мінімізації маршрутів. Якщо сума довжин нових ділянок менша від суми попередніх, то старі ділянки маршрутів замінюються новими.

В якості вхідних даних для алгоритму подається загальний маршрут, що потрібно оптимізувати та додаткові параметри, а саме:

1. розмір геометричної області сканування (якщо область прямокутна, то ширина і висота, якщо кругла – то радіус і т.д.);
2. розмір області перетину (у відсотках) – розмір спільної геометричної області що належать двом сусіднім областям сканування;
3. базовий алгоритм.

Метод може бути застосовано для покращення будь-якого маршруту – початкового чи вже оптимізованого за допомогою іншого алгоритму. Пропонується повторне проходження алгоритму оптимізації (із зміненими параметрами, наприклад із зміненим розміром області сканування), де очікується ще деяке покращення якості маршруту.

Результати експериментів показують, що метод забезпечує високу якість оптимізації, особливо коли заданий достатньо великий розмір області сканування. У процесі роботи алгоритму мінімізуються сумарні довжини окремих частин маршруту – точки, що належали одним ділянкам маршруту, переходять до інших.

Розмір області перетину рекомендується задавати у межах 25-50%. При таких параметрах забезпечується детальне сканування усієї вхідної області при відносно немалих часових затратах.

Опишемо роботу алгоритму. Для усіх точок заданого вхідного маршруту обчислюється прямокутна область, у яку попадають усі точки. Далі згідно заданих параметрів визначається область сканування та встановлюється у верхню ліву позицію. Починається цикл роботи алгоритму із “розпізнаванням” та оптимізацією маршрутів у сканувальній області.

Частини маршруту, що перетинають область сканування, вважаються такими, що їй належать. Формується множина таких маршрутів у локальній області. Розв’язується задача сумарної мінімізації маршрутів локальної області. Якщо в результаті сумарна довжина всіх частин маршруту зменшується, попередні частини замінюються на нові. Область сканування переміщується згідно заданих параметрів і процес повторюється знову доти, поки не буде просканована всієї вхідної області точок. Схематично геометричне сканування показано на рисунку 1. Зображено поточний маршрут, прямокутні області сканування із частинами загального маршруту, що до них входять.

Розглянемо детальніше особливості запропонованого методу. Роботу одної ітерації алгоритму геометричного сканування можна розділити на такі етапи:

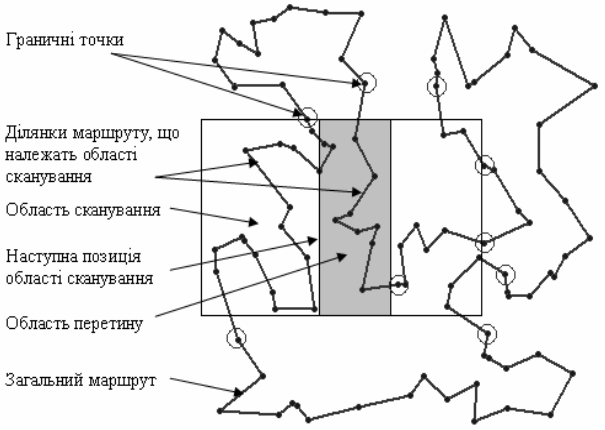

Рис. 1. Оптимізація маршруту геометричним скануванням

- встановлення позиції сканувальної області;
- ідентифікація точок, що належать сканувальній області;
- пошук серед точок області таких, що містять ребра перетину із сканувальною областю – пошук т.з. активних точок;
- ідентифікація маршрутів у сканувальній області (здійснюється на основі інформації про активні точки);
- сортування маршрутів у порядку їх проходження у загальному маршруті;
- встановлення граничних точок – точок поза межею області сканування;
- встановлення умовних ребер між парами граничних точок;
- розв’язання задачі сумарної мінімізації маршрутів;
- порівняння сумарної довжини отриманих маршрутів із попередньою та оновлення загального маршруту у випадку покращення.

Розглянемо усі етапи детальніше. Для прикладу візьмемо прямокутну область оптимізації. Нехай задано розміри області сканування: ширина - width та висота – height. Область перетину – 50%. Тоді на кожному кроці ітерації сканувальна область переміщуватиметься на width*0,5 точок вправо, а при досягненні крайнього горизонтального положення – перейде на height*0,5 точок вниз у позицію зліва. Для швидкої ідентифікації точок у елементарній області доцільно застосувати деякий швидкий алгоритм пошуку точок по заданих координатах області. Також для цієї ж мети ще перед початком циклу алгоритму може бути застосовано спеціальну попередню процедуру ідентифікації точок у областях сканування. Далі здійснюється пошук активних точок – розглядаються усі точки області і ті, що містять ребра перетину із областю – позначаються як активні. На основі інформації про активні точки здійснюється ідентифікація маршрутів у елементарній області.

Опишемо алгоритм ідентифікації маршрутів у області. Кожна точка загального маршруту містить інформацію про свою позицію у маршруті. Для пошуку маршруту в області, вибирається його активна точка. Далі вибирається наступна по загальному маршруту точка області і процес триває доти, поки не буде досягнуто іншої активної точки. Усі відвідані в цьому процесі точки складатимуть один маршрут, що належить елементарній області. Дві активні точки, що належать знайденому маршруту, встановлюються як звичайні і все повторюється. Процес триває доти, поки не залишиться активних точок у елементарній області. З наведеного випливає, що в кожній елементарній області завжди повинна бути парна кількість активних точок – на кожен маршрут припадає дві активні точки. Знайдені маршрути можуть бути вилучені, якщо вони складаються із замалої кількості точок – мінімальна кількість точок у ділянці маршруту може бути додатковим параметром алгоритму.

Наступний етап – сортування ідентифікованих маршрутів. Для сортування знайдених маршрутів шукається активна точка, що в загальному маршруті відвідується найшвидше з-поміж інших активних точок області – початкова точка оптимізації. Далі всі наступні маршрути області сортуються по порядку їх проходження основним маршрутом. Сортування маршрутів – дуже важливий етап. Усі ділянки обов'язково повинні бути відсортовані у правильній послідовності їх відвідування в головному маршруті – від цього залежить етап роботи алгоритму, де встановлюються умовні ребра між граничниими точками. Якщо умовні ребра будуть встановлені неправильно – можуть утворюватися внутрішні цикли. Граничними точками вибираються ті, що належать ребрам перетину загального маршруту із елементарною областю.

Встановлення умовних ребер між парами граничних точок є наступним етапом. Кожна ділянка містить дві граничні точки – початкову та кінцеву граничні точки. Кінцева гранична точка ділянки з'єднується умовним ребром із початковою граничною точкою наступної і так далі. Кінцева гранична точка останнього маршруту з'єднується із початковою граничною точкою першого маршруту. На рисунку 2 показано область сканування з ідентифікованими маршрутами, встановленими граничними точками та умовними ребрами.

Наступний етап – розв'язання задачі сумарної мінімізації маршрутів. Для розв'язання цієї задачі застосовується класичний базовий алгоритм розв'язання задачі комівояжера, проте з певною модифікацією. У якості вхідних даних для базового алгоритму є: множина усіх точок, що належить області сканування; усі граничні точки; усі встановлені умовні ребра між граничними точками.

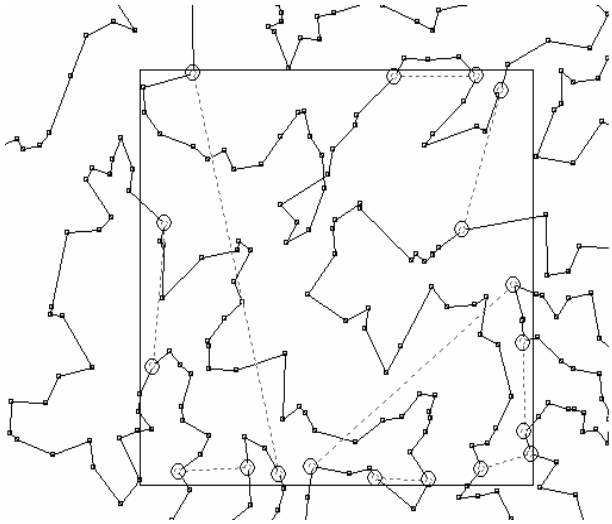

Рис. 2. Область сканування з ідетифікованими частинами маршруту, граничними точками та умовними ребрами

Результатом виконання базового алгоритму буде один маршрут між усіма точками (що належали раніше різним маршрутам). Варто зазначити, що у результуючому маршруті передані умовні ребра та граничні точки залишаться. І, після їх вилучення, отримаємо таку ж кількість маршрутів, що і була до того. На рисунку 3 показано область сканування після сумарної мінімізації маршрутів у ній.

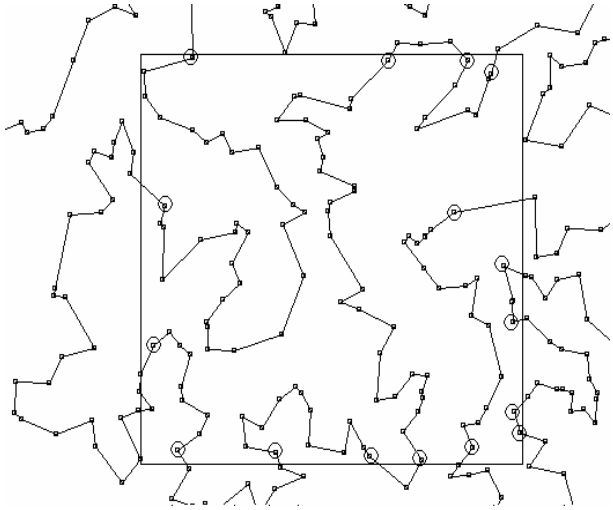

Рис. 3. Оптимізовані маршрути в елементарній області сканування

Основною перевагою описаного підходу є висока якість оптимізації маршруту. Даний підхід може бути застосований у різних типах транспортних задач. Обчислювальна складність залежить від обчислювальної складності базового алгоритму.

3. Аналіз одержаних результатів

Проведено тестування для задач із розмірностями 1000 та 10000 точок. Розподіл точок у тестах – довільний. Розмір областей сканування задавався однаковим по висоті та ширині від 100 до 500 пікселів при розмірі вхідної множини точок 1000x1000 пікселів. Маршрути для оптимізації були отримані відомим класичним алгоритмом *2-opt*, що забезпечує якість в межах 6-9% від оптимальної. У ролі базового використовувався алгоритм Гельсгауна.

Тести проводилися на ПК з процесором Intel Pentium IV з частотою 3 ГГц і 512 Мб ОЗП.

Досліджено як впливає розмір 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(70%), 300(30%), 300(50%), 300(70%), 400(30%), 400(50%), 400(70%), 500(30%), 500(50%), 500(70%). Таблиця 1 містить результати тестування задач розмірністю 1000 точок.

Таблиця 1

Результати тестування задач розмірністю 1000 точок

Розмір області сканування і перетину	Час виконання, сек.	Відсоток покращення маршруту
100(30%)	0,91	1,1%
100(50%)	1,48	1,17%
100(70%)	1,93	1,17%
200(30%)	1,97	2,06%
200(50%)	2,75	2,54%
200(70%)	3,35	2,75%
300(30%)	3,21	2,95%
300(50%)	3,98	3,43%
300(70%)	5,07	3,67%
400(30%)	8,2	3,8%
400(50%)	9,8	4,21%
400(70%)	11,2	4,32%
500(30%)	19,3	4,82%
500(50%)	25	5,45%
500(70%)	34	5,6%

При збільшенні розміру області оптимізації зростає якість маршруту. Для задачі розмірністю 1000 точок при розмірі області оптимізації 400x400 пікселів та області перетину 50% маршрут покращується на 4,21%. При цьому оптимізація триває близько 10 секунд.

Та ж тенденція спостерігається і для задач розмірністю 10000 точок. Таблиця 2 містить результати тестування.

Таблиця 2

Результати тестування задач розмірністю 10000 точок

Розмір області сканування і перетину	Час виконання, сек.	Відсоток покращення маршруту
100(30%)	10,57	1,17%
100(50%)	12,83	1,27%
100(70%)	20,1	1,35%
200(30%)	19,3	2,22%
200(50%)	23,7	2,43%
200(70%)	33,3	2,68%
300(30%)	29,1	2,78%
300(50%)	39,2	3,33%
300(70%)	55,7	3,72%
400(30%)	97	3,97%
400(50%)	148	4,33%
400(70%)	189	4,41%
500(30%)	207	4,63%
500(50%)	298	4,85%
500(70%)	357	5,02%

Висновки

Запропонований алгоритм оптимізації розв'язків задачі комівояжера демонструє покращення якості маршруту в межах 3-5 % (при достатньо великих розмірах області сканування та перетину) для розв'язків, одержаних за допомогою класичного алгоритму *2-opt*.

Зі збільшенням розміру області сканування час обчислень зростає пропорційно. Із різким збільшенням розмірності задачі не спостерігається різке збільшення часу обчислень. Якість, що забезпечує алгоритм, залишається сталою зі збільшенням розмірності задачі.

Література

1. Reinelt G. The Traveling Salesman: Computational Solutions for TSP Applications. Lecture Notes in Computer Science 840 / G. Reinelt. – Springer–Verlag, Berlin, 1994.

2. Reinelt G. Fast heuristics for large geometric traveling salesman problems / G. Reinelt // ORSA Journal on computing. - 1992. – Vol. 4. – P. 206-217.

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

4. Lin S. An effective heuristic algorithm for the Traveling salesman problern / S. Lin, B.W. Kernighan // Operations Research. - 1973. – Vol. 21. – P. 498–516.

5. Lin S. Computer solutions of the travelling salesman problem / S. Lin // Bell System Technical Journal. – 1965. – Vol. 44. – P. 2245–2269.

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

7. Reinelt G. TSPLIB – A traveling salesman problem library / G. Reinelt // ORSA Journal on Computing. – 1991. – Vol. 3. – P. 376-384.

8. Електронний ресурс: http://www.research.att.com/~dsj/chtsp.

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

10. Applegate D. Chained Lin–Kernighan for large traveling salesman problems / D. Applegate, W. Cook, A. Rohe // INFORMS J. Computing. – 2003. - Vol. 15. - No. 1. - P. 82-92.

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

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

13. Електронний ресурс: http://www.tsp.gatech.edu/concorde.html.

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

15. Laporte G. A Tabu Search using Genetic Diversification for the Clustered Traveling Salesman Problem / G. Laporte, J-Y. Potvin, F. Quilleret // Journal of Heuristics. – 1996. - Vol 2 (3). - P. 187-200.

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

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

18. Bazylevych R. Scanning-area algorithms for clustered TSP / R. Bazylevych, R Dupas, R Kutelmakh // Proceedings of International conference "Comp. science and Information Technologies" Lviv, Polytechnic University. – 2006. - P. 148-152.

Надійшла до редакції 14.01.2009

Рецензент: д-р техн. наук, проф. А.В. Скатков, Севастопольський національний технічний університет, Севастополь, Україна.

АЛГОРИТМ ОПТИМИЗАЦИИ РЕШЕНИЯ ЗАДАЧИ КОММИВОЯЖЕРА В ЛОКАЛЬНОЙ ОБЛАСТИ

Р.П. Базилевич, Р.К. Кутельмах

Предложен новый метод оптимизации решения Эвклидовой задачи коммивояжера, который может быть применен для улучшения решения задачи, полученного с помощью декомпозиции или какого-либо быстрого эвристического алгоритма. Как базовую процедуру для решения задачи коммивояжера алгоритм использует один из наиболее эффективных современных подходов – алгоритм Кельда Гельсгауна, который является модификацией алгоритма Лина-Кернигана.

Ключевые слова: задача коммивояжера, декомпозиция, транспортная задача, NP-сложность, область сканирования, оптимизация, алгоритм Лина-Кернигана, алгоритм Гельсгауна.

AN OPTIMIZATION ALGORITHM OF TRAVELING SALESMAN PROBLEM SOLVING IN LOCAL SPACE

R.P. Bazylevych, R.K.Kutelmah

The paper presents a new optimization method of Euclid's travelling salesman problem. The method can be used for improvement of problem solving that was obtained by using decomposition or other heuristic algorithm. The algorithm uses one of most effective modern approaches – algorithm of Helsgaun that is modification of algorithm of Lin-Kernighan such as a basic procedure for travelling salesman problem solving.

Key words: traveling salesman problem (TSP), decomposition, traffic problem, NP-complexity, scan space, optimization, algorithm of Lin-Kernighan, algorithm of Helsgaun.

Базилевич Роман Петрович – д-р техн. наук, професор, професор кафедри програмного забезпечення Інституту комп'ютерних наук та інформаційних технологій Національного університету Львівська політехніка, Львів, Україна, e-mail: rbaz@polynet.lviv.ua.

Кутельмах Роман Корнелійович – аспірант кафедри програмного забезпечення Інституту комп'ютерних наук та інформаційних технологій Національного університету Львівська політехніка, Львів, Україна, e-mail: rkutelmakh@polynet.lviv.ua.