Висновки

Провівши числові експерименти, можна зробити висновки:

- Використання пакета Comsol Multiphysic 3.4 для розв'язування задач деформування конструкцій складної геометрії вимагає детального дослідження числових розв'язків з метою перевірки їх достовірності. Це, своєю чергою, приводить до потреби в потужному комп'ютері.
- Числові розв'язки задач, отримані на недосконалій сітці і наведені на рис. 3, 7, 9а, свідчать про те, що до подібного деформування (фактично втрати стійкості) можуть приводити незначні дефекти геометрії конструкції, які з'являться у процесі виготовлення, експлуатації тощо.

1. Григоренко А.Я., Дыяк И.И., Макар В.М. Решение пространственной динамической задачи теории упругости для анизотропных тел // Прикл. механика. – 1998. – 43(44), №5. – С. 24–31. 2. Копитко М.Ф., Савула Я.Г. Алгоритмічний підхід до дослідження задач пружного деформування оболонок // Вісник Львів. ун-ту. Сер. мех.-мат. – 1997. – Вип. 46.– С. 10–16. 3. Лурье А.И. Теория упругости. – М.: Наука, 1970. – 939 с. 4. Меламед Л. Э. Femlab и ANSYS в расчетах гидродинамики атомных реакторов или научно-практический рассказ о том, как приспособить «тяжелые» пакеты для решения задач одного тяжелого класса // ExponentaPRO. – 2004. – №2. 5. Сергиенко И.В., Скопецкий В.В., Дейнека В.С. Математическое моделирование и исследование процесов в неоднородных средах. – К.: Наук. думка, 1991 – 432 с. 6. Тимошенко С.П., Дж. Гудьер Теория упругости. – М.:Наука, 1975. – 576 с. 7. Savula Y., Mang H., Dyyak I., Pauk N. Coupled boundary and finite element analysis of a special class of 2D problems of the theory of elasticity // Comput.&Struct.– 2000. – Vol. 75, No.2. – P. 157–165. 8. Zienkiewicz O.C. Finite element method. – London: McGraw-Hill, 1977. – 788 p.

УДК 519.16

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

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ІСНУЮЧИХ АЛГОРИТМІВ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ КОМІВОЯЖЕРА

Досліджено ефективність існуючих точних та евристичних алгоритмів розв'язання задачі комівояжера. Зроблено висновки щодо доцільності їх застосування при розв'язанні задач великих розмірностей, а також при декомпозиції.

Existing exact and heuristic algorithms' efficiency for solving Traveling Salesman Problem has been investigated. The conclusions were made of their application for solving large-scale problems as well as for using with decomposition.

Вступ

Задача комівояжера – одна з 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 – у 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 – Concorde TSP Solver [13]. Він забезпечив одержання оптимальних розв'язків для усіх тестів із бібліотеки TSPLIB, розмірністю включаючи 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).

Формулювання задачі

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

$$L^*(T^*) = \Sigma_{ij}\, l_{ij}^* \rightarrow \min \Sigma_{ij}\, l_{ij}^* \quad \forall\, l_{ij}^* \in l_{ij}',$$

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

Існує $N!$ різних варіантів маршрутів через задану множину точок. Задача є NP-складною, тому більша частина досліджень сконцентрована на отриманні розв'язків, близьких до оптимальних. Сьогодні існує багато методів розв'язування задачі комівояжера. Найпоширеніші з них описано нижче.

Існуючі методи розв'язування задачі комівояжера

Оптимальні розв'язки задачі комівояжера можуть бути отримані за допомогою методів лінійного програмування. Традиційно методи для розв'язання задачі комівояжера поділяються на:

1) алгоритми для знаходження точних маршрутів (потребують значних часових затрат і підходять лише для задач малих розмірностей);

2) евристичні алгоритми, що забезпечують якість маршруту, близьку до оптимальної, але не вважаються оптимальними.

Для оцінки якості розв'язку задачі комівояжера застосовується межа Хелда–Карпа, що ґрунтується на методах релаксації. При оцінці якості отриманого маршруту його довжину співвідносять із нею. Межа Хелда–Карпа є де-факто стандартом оцінювання якості отриманого маршруту.

Межа Хелда-Карпа

Ця величина, що відносно швидко обчислюється, має особливе значення при оцінці якості маршрутів для задач великих розмірностей, для яких оптимальний маршрут невідомий. Нижня межа Хелда–Карпа – це розв'язок релаксації лінійного програмування стандартного формулювання задачі комівояжера цілочисельного програмування. Методи лінійного програмування для оцінки межі Хелда–Карпа для задач з розмірністю понад декількох сотень точок не є ефективними. Ефективні реалізації лінійного програмування стають нездійсненними для задач розмірністю декілька тисяч точок. У зв'язку з цим Хелд та Карп пропонують ітеративний метод оцінювання [16, 17].

Ітеративний метод оцінки використовує релаксацію Лагранжа, що опрацьовує послідовність зв'язних графів, які в процесі роботи алгоритму все більш і більш стають близькими до маршрутів [18]. Ця техніка, що ґрунтується на ідеї 1-дерева, була вдало використана як основа для багатьох методів границь та меж для розв'язання задачі комівояжера [19]. Підхід також відомий в літературі як субградієнтна оптимізація [20]. Методи гілок та меж, що ґрунтуються на субградієнтній оптимізації, складаються з двох етапів. Перший етап – початкове сходження, обчислює нижню

межу для задачі комівояжера. Другий етап – загальне сходження, об'єднує процедуру гілок та меж з процесом релаксації щоб отримати розв'язки задачі комівояжера. Результат початкового сходження згодом став відомий у літературі як нижня межа Хелда–Карпа.

Герхард Рейнельт у своїй книзі [1] присвятив розділ для обчислення нижніх меж для задачі комівояжера. Його результати чітко показують, що релаксація Лагранжа, основана на 1-деревах, забезпечує найкращі результати зі всіх методів, що він досліджував.

Нижня межа Хелда–Карпа визначається релаксацією 1-дерева, де 1-дерево задачі з розмірністю N точок – це зв'язний граф з точками 1, 2,…, N, що складається з дерева з точок 2, 3,…, N та двох ребер, що зв'язані з точкою 1. Визначення нижньої межі Хелда–Карпа потребує знаходження послідовності 1-дерев, де мінімальне 1-дерево є мінімальне зв'язне дерево для точок 2, 3, …, N та двома найкоротшими ребрами до точки 1.

Маршрут – це 1-дерево, в якому кожна точка має степінь 2. З цього випливає, що якщо 1-дерево є маршрутом, то цей маршрут є оптимальним. Мінімальне 1-дерево показано на рис. 1.

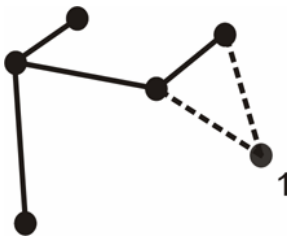

Рис. 1. Мінімальне 1-дерево

Процес релаксації Лагранжа ітеративно змінює степені кожної точки 1-дерева до величини 2. Іншими словами, алгоритм поступово перетворює мінімальні 1-дерева на маршрути.

Нижня межа Хелда–Карпа, яку можна відносно швидко і легко обчислити, використовуючи наведену техніку, має величезне практичне значення для оцінювання якості розв'язків задач великих розмірностей, де оптимум невідомий. Існує багато реальних задач з розмірністю від 10000 до 100000 точок (приклади таких задач з галузей кристалографії рентгенівських променів, виготовлення інтегрованих схем та свердління друкованих плат доступні в бібліотеці TSPLIB [7]).

Результат, отриманий Волсі [21] та пізніше Шмойсом та Вільямсоном [22], показує, що нижня межа Хелда–Карпа у найгіршому випадку ніколи не є нижчою від 2/3 довжини оптимального маршруту, коли забезпечується нерівність трикутника. Різні автори публікують результати обчислювальних експериментів, які демонструють, що різниця між нижньою межею та довжиною оптимального маршруту є значно меншою від 1% для задач розмірністю декілька сотень точок [23,24]. Роботи Джонсона та колег показують, що для евклідових задач з довільним розміщенням точок розмірністю декілька тисяч середня різниця між значеннями нижньої межі Хелда–Карпа та довжини оптимального маршруту становить 0,65 % [25, 26].

Точні алгоритми

Точні алгоритми для розв'язування задачі комівояжера забезпечують одержання оптимального розв'язку. Розрізняють три основні типи точних алгоритмів для розв'язання задачі:

1) метод гілок та меж, що може бути використаний для розв'язання задач розмірністю 50-100 точок;

2) прогресивні методи покращання, що використовують методи лінійного програмування; використовуються для задач розмірністю 120–200 точок;

3) сучасні реалізації методів гілок та меж та відсікаючих площин, що ґрунтуються на лінійному програмуванні; забезпечують оптимальні та ефективні розв'язки для задач розмірностями до 5000 точок, також цей метод у поєднанні з алгоритмом Ліна–Кернігана–Гельсгауна був застосований для знаходження оптимального маршруту для задачі розмірністю 85900 точок – найбільшої задачі, що розв'язана сьогодні оптимально.

Точний розв'язок для 15112 міст Німеччини був знайдений у 2001 році за допомогою методу відсікаючих площин, запропонованого Данцігом, Фалкерсоном та Джонсоном у 1954 році, що ґрунтується на методах лінійного програмування [27]. Обчислення проводилися на мережі зі 110 процесорів, розміщених в університетах Princeton University та Rice University. Загальний час обчислень був еквівалентним 22,6 рокам роботи процесора Alpha 500 MHz.

У травні 2004 року було розв'язано задачу комівояжера для 24978 міст Швеції: було знайдено маршрут довжиною приблизно 72500 кілометрів і було доведено, що не існує коротшого маршруту.

У березні 2005 року задачу комівояжера для 33810 точок, розміщених на друкованій платі, було розв'язано за допомогою програмного забезпечення Concorde [11, 13]: знайдено маршрут довжиною 66048945 одиниць та було доведено, що не існує коротшого маршруту. Обчислення зайняли приблизно 15,7 років процесорного часу. 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).

Наближені алгоритми

Сьогодні існує багато наближених (евристичних) алгоритмів розв'язання задачі. Наближені алгоритми забезпечують одержання розв'язку задачі, близького до оптимального, але не гарантують його оптимальності. Сучасні реалізації можуть з великою ймовірністю знайти розв'язок для задач великих розмірностей (мільйонів точок) у припустимих часових межах, який може бути всього на 2–3% довший від оптимального. Евристичні алгоритми залежно від особливостей своєї роботи поділяються на такі основні типи:

1) конструктивні алгоритми. Це найшвидші за часом обчислень методи, що забезпечують найгіршу якість розв'язків. За кожен крок роботи алгоритму до існуючої частини знайденого маршруту додається нове ребро. Найкращі конструктивні алгоритми забезпечують якість, на 10–15% гіршу від оптимального розв'язку;

2) алгоритми оптимізації маршруту. Методи, що поступово покращують початковий маршрут, отриманий за допомогою існуючого швидкого конструктивного алгоритму. Оптимізація відбувається завдяки зміні частин існуючого маршруту на коротші (локальній оптимізації). Забезпечують якість маршруту, на 3–10% гіршу від оптимального;

3) алгоритм Ліна–Кернігана та його модифікації. Це удосконалений алгоритм оптимізації маршруту, що забезпечує розв'язки, на 1–3% гірші від оптимального;

4) алгоритм Ліна–Кернігана–Гельсгауна. Модифікований алгоритм Ліна–Кернігана Келдом Гельсгауном, що забезпечує знаходження оптимальних розв'язків або дуже близьких до оптимальних (у межах до 1%);

5) генетичні алгоритми.

Далі описано особливості перелічених алгоритмів.

Конструктивні алгоритми

Конструктивні алгоритми належать до найпростіших методів знаходження розв'язків задачі комівояжера. За кожен крок виконання алгоритму до знайденої частини маршруту додається нове ребро. Алгоритм припиняє роботу, коли розв'язок знайдено і ніколи не намагається покращити його. До конструктивних алгоритмів належать:

1) алгоритм найближчого сусіда;

2) жадібний алгоритм;

3) алгоритм вставки.

Найприродніший евристичний алгоритм для розв'язання задачі комівояжера – алгоритм найближчого сусіда. Алгоритм починається в довільній точці та поступово відвідує кожну найближчу точку, що ще не була відвідана. Алгоритм завершується, коли відвідано всі точки. Остання точка з'єднується з першою. Обчислювальна складність алгоритму – $O(n^2)$.

АЛГОРИТМ НАЙБЛИЖЧОГО СУСІДА

ВХІДНІ ДАНІ: множина точок V розмірністю N.

ВИХІДНІ ДАНІ: маршрут T, що складається з послідовності відвідування точок множини V.

1) вибрати довільну точку v_1;
2) $T_1 = v_1$;
3) ДЛЯ i=2 ДО i=N ВИКОНАТИ
   a. ВИБРАТИ точку v_i, найближчу до точки T_{i-1};
   b. $T_i = v_i$;
4) $T_{N+1} = v_1$;
5) КІНЕЦЬ АЛГОРИТМУ.

Результатом виконання алгоритму найближчого сусіда є маршрут, приблизно на 25% довший від оптимального [3].

Жадібний алгоритм будує маршрут поступово, на кожному кроці додаючи найкоротше ребро, яке ще не належить маршруту. Ребро додається за умови, що воно не створить циклу всередині маршруту. Алгоритм завершується, коли додано N ребер або кожна точка має степінь 2. Кожне ребро може бути додане лише один раз. Обчислювальна складність алгоритму - $O(n^2 \log_2(n))$.

ЖАДІБНИЙ АЛГОРИТМ

ВХІДНІ ДАНІ: множина точок V розмірністю N, множина ребер E.

ВИХІДНІ ДАНІ: маршрут T, що складається з вибраних ребер множини E.

1) ВІДСОРТУВАТИ усі ребра множини E;
2) ВИБРАТИ найкоротше ребро e та ВИЛУЧИТИ його з множини E;
3) ЯКЩО додавання ребра e до множини T не створить циклу при кількості ребер менше N, ТО ДОДАТИ це ребро до множини T;
4) ЯКЩО у маршруті T не міститься N ребер, ТО ВЕРНУТИСЬ на КРОК 2.
5) КІНЕЦЬ АЛГОРИТМУ.

Згідно з проведеними експериментами [28], жадібний алгоритм забезпечує якість отриманого маршруту, приблизно на 20% гіршу від оптимального.

Алгоритм вставки також є простим методом швидкого знаходження наближених розв'язків задачі. На початку роботи алгоритму визначається маршрут, що складається з трьох точок і формує трикутник. На кожному наступному кроці до маршруту додається найближча точка, що не належить йому. Обчислювальна складність алгоритму – $O(n^2)$.

АЛГОРИТМ ВСТАВКИ

ВХІДНІ ДАНІ: множина точок V розмірністю N.

ВИХІДНІ ДАНІ: маршрут T, що складається з послідовності відвідування точок множини V:

1) СТВОРИТИ початковий маршрут з трьох точок v_1, v_2, v_3;
2) ДОДАТИ точки v_1, v_2, v_3 до множини T;
3) ВИЛУЧИТИ точки v_1, v_2, v_3 з множини V;
4) ДЛЯ i=4 ДО i=N ВИКОНАТИ
   a. ВИБРАТИ точку v_i, найближчу до маршруту T;
   b. ДОДАТИ точку v_i до множини T між двома найближчими до неї точками;
   c. ВИЛУЧИТИ точку v_i з множини V;
5) КІНЕЦЬ АЛГОРИТМУ.

Конструктивні алгоритми забезпечують найгіршу якість отриманих розв'язків задачі комівояжера при обчислювальній складності в середньому $O(n^2)$. Існують методи пришвидшення роботи описаних вище підходів. Зокрема завдяки методу, запропонованому Бентлі [29–32] – *k-d-*

деревам, що ґрунтується на побудові ієрархічної декомпозиції вхідної області точок на частини, можна пришвидшити роботу алгоритмів до часу, еквівалентному O(*nlog(n)*). Алгоритми локальної оптимізації, що описані в наступних підрозділах, використовують розв'язки, отримані за допомогою методів побудови маршруту як початкові.

Алгоритми оптимізації маршруту

Після того, як за допомогою певного алгоритму побудови маршруту було одержано деякий розв'язок, є можливість його покращити. Найпоширенішими методами покращання маршруту є алгоритми локальної оптимізації *2-opt* та *3-opt*.

У класичній реалізації алгоритм *2-opt* вилучає 2 ребра, що входять до маршруту і додає два нові так, щоб новий маршрут став коротшим. Існує лише один варіант заміни двох ребер на нові два, при якому новий маршрут залишиться маршрутом, а не утворяться 2 цикли. На рис. 2 показано заміну *2-opt*.

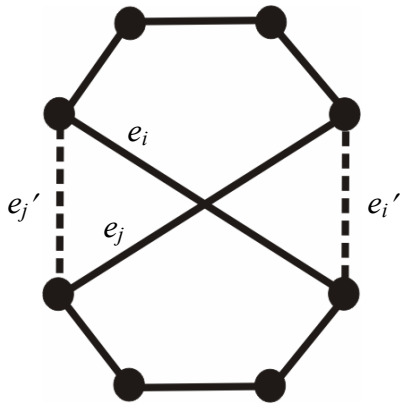

Рис. 2. Заміна 2-opt

Алгоритм триває доти, поки не залишиться можливих замін, що оптимізують маршрут. Одержаний маршрут називають 2-оптимальним.

АЛГОРИТМ 2-OPT

ВХІДНІ ДАНІ: множина точок V розмірністю N.

ВИХІДНІ ДАНІ: маршрут $T_{2\text{-}opt}$, що складається з послідовності відвідування точок множини V.

1) СТВОРИТИ початковий маршрут T за допомогою деякого алгоритму побудови маршруту;
2) ДЛЯ i=1 ДО i=N ВИКОНАТИ
   a. ДЛЯ j=1 ДО j=N ВИКОНАТИ
      i. ВИБРАТИ ребра e_i, e_j
      ii. ЯКЩО існують ребра e_i', e_j' такі, що забезпечують меншу довжину маршруту, ТО ЗАМІНЮЄМО в T ребра e_i, e_j на ребра e_i', e_j';
3) $T_{2\text{-}opt} = T$;
4) КІНЕЦЬ АЛГОРИТМУ.

Обчислювальна складність алгоритму - $O(n^2)$. За результатами досліджень [3], алгоритм *2-opt* забезпечує якість маршруту, на 4–7% гіршу від оптимальної.

Алгоритм *3-opt* працює так само, як і *2-opt*; відмінність полягає у тому, що замість заміни 2 ребер він замінює 3 ребра так, щоб маршрут став коротшим. Це означає, що існує 2 варіанти заміни 3 ребер. На рис. 3 показано два можливі варіанти заміни *3-opt*.

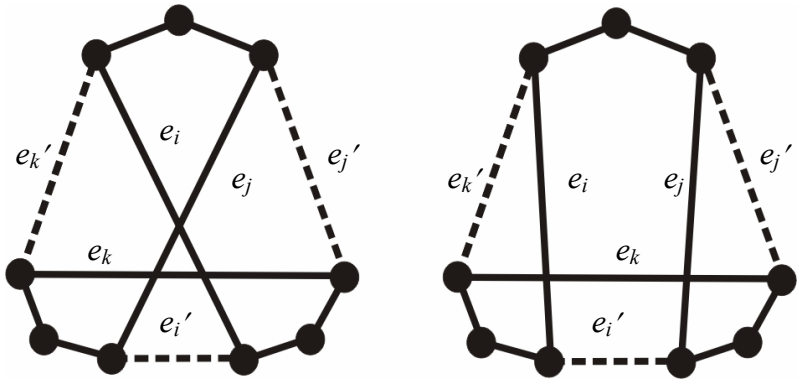

Рис. 3. Можливі варіанти заміни 3-opt

Заміна *3-opt* може розглядатися як 2 або 3 заміни *2-opt*. Алгоритм завершується, якщо немає більше таких замін, які б оптимізували маршрут. У результаті ми отримуємо маршрут, що називається 3-оптимальним. Якщо маршрут 3-оптимальний, то він також і 2-оптимальний [6].

АЛГОРИТМ 3-OPT

ВХІДНІ ДАНІ: множина точок V розмірністю N.

ВИХІДНІ ДАНІ: маршрут $T_{3\text{-}opt}$, що складається з послідовності відвідування точок множини V.

1) СТВОРИТИ початковий маршрут T за допомогою деякого алгоритму побудови маршруту;
2) ДЛЯ i=1 ДО i=N ВИКОНАТИ
    a. ДЛЯ j=1 ДО j=N ВИКОНАТИ
        i. ДЛЯ k=1 ДО k=N ВИКОНАТИ
            1. ВИБРАТИ ребра e_i, e_j, e_k
            2. ЯКЩО існують ребра e_i', e_j', e_k' такі, що забезпечують меншу довжину маршруту, ТО ЗАМІНЮЄМО в T ребра e_i, e_j, e_k на ребра e_i', e_j', e_k';
3) $T_{3\text{-}opt} = T$;
4) КІНЕЦЬ АЛГОРИТМУ.

Обчислювальна складність алгоритму - O(n^3). За результатами досліджень [3], алгоритм *3-opt* забезпечує якість маршруту на 2,5–4% гіршу від оптимального.

Для пришвидшення роботи алгоритмів застосовується обмеження для пошуку нових ребер. Для кожної вершини обчислюється m її найближчих сусідів. Це забезпечує пошук кожного нового ребра для точки не з-поміж усіх допустимих, а лише тих, що входять до списку сусідів. Тоді обчислювальна складність алгоритму *2-opt* становить O(mn), проте ми повинні знайти m найближчих сусідів для кожної точки. Це пришвидшення ліквідовує 2-оптимальність, але при достатньо великих m (m = 20) якість практично не втрачається [3]. При малих m забезпечується суттєвий виграш в часі, проте і значно гірша якість одержаного маршруту.

Існують алгоритми *4-opt*, *5-opt* і т.д., проте кожен з них потребує на порядок більше часу обчислень, забезпечуючи при цьому несуттєві покращення маршруту. Наступний підрозділ описує алгоритм Ліна–Кернігана – так званий змінний *k-opt*.

Версії алгоритму Ліна-Кернігана

Впродовж довгих років алгоритм Ліна–Кернігана [4] залишається одним з найкращих евристичних методів розв’язування задачі комівояжера [33]. Алгоритм забезпечує знаходження маршруту, в середньому не довшого на 2% від оптимального. Основний підхід Ліна–Кернігана ґрунтується на техніці локальної оптимізації *k-opt*, де k – змінне число, яке змінюється на кожному кроці виконання алгоритму. Це забезпечує одержання значно кращих розв’язків порівняно з

алгоритмами *2-opt* та *3-opt*, у яких число k є фіксованим. Проте алгоритм не є простим для реалізації, у сучасних версіях має багато параметрів. Існує багато покращень алгоритму для того, щоб збільшити його швидкодію. Обчислювальна складність алгоритму – $O(n^{2.2})$.

Алгоритм Ліна–Кернігана – алгоритм локальної оптимізації, що отримує початковий субоптимальний маршрут та оптимізовує його за допомогою замін ребер. Для задачі розмірністю N точок розв'язок не є довшим у $4\sqrt{N}$ разів від оптимального, а також ніколи не є гіршим за початковий маршрут [34]. На практиці алгоритм виконується швидко. Проте час виконання алгоритму збільшується для задач із кластерним розподілом точок [3, 33, 14].

Підхід запропонований у 1973 році Ліном та Керніганом. Більшість аспектів сучасної високоякісної реалізації було описано в їхній оригінальній праці [4]. Модифікації для розширення практичного застосування евристики для задач великих розмірностей – це використання структур даних з кращими характеристиками та додаткові елементи, наприклад, біти "не дивитись" (don't-look bits), запропоновані Бентлі – довели практичність свого застосування на експериментах [31, 32, 35].

Алгоритм Ліна–Кернігана передбачає зміни ребер, що послідовно перетворює початковий маршрут на інший. Зміни тривають доти, поки не буде можливих замін ребер, що забезпечать одержання коротшого маршруту. Дано початковий маршрут T. На кожному кроці виконання алгоритм намагається знайти дві множини ребер $X=\{x_1, x_2,\ldots, x_k\}$ та $Y=\{y_1, y_2,\ldots, y_k\}$ такі, що якщо ребра з множини X вилучити з маршруту, а натомість додати ребра з множини Y, то в результаті отримаємо кращий маршрут. Ця взаємна заміна ребер називається заміною *k-opt*. Множини X та Y створюються послідовно, елемент за елементом. Спочатку вони порожні. На кожному кроці i пара елементів x_i, y_i додається до відповідних множин. Для того, щоб досягти ефективного результату, до множин X та Y можуть входити тільки ті ребра, що відповідають необхідним критеріям:

1) критерій послідовної заміни: x_i та y_i , а також y_i та x_{i+1} повинні мати спільну точку; тобто якщо t_1 – одна з точок ребра x_1 , то для i>=1 маємо: $x_i = (t_{2i-1}, t_{2i})$, $y_i = (t_{2i}, t_{2i+1})$ та $x_{i+1} = (t_{2i+1}, t_{2i+2})$. Тобто послідовність $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4,\ldots, x_k, y_k)$ – це ланцюг суміжних ребер. Необхідною, але недостатньою умовою для того, щоб заміна ребер з множини X ребрами з множини Y сформувала новий маршрут є та, що ланцюг $(x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4,\ldots, x_k, y_k)$ повинен бути закритим, тобто $y_k = (t_{2k}, t_1)$;

2) критерій допустимості. Якщо $x_i = (t_{2i-1}, t_{2i})$, а t_{2i} з'єднана з точкою t_1 , то необхідно, щоб у результаті утворився маршрут. Цей критерій призначений для i >= 3 і гарантує можливість закриття маршруту. Критерій був включений до алгоритму для того, щоб зменшити час виконання та спростити кодування;

3) критерій позитивного приросту. Критерій вимагає того, що заміна ребер з множини X ребрами з множини Y скоротить маршрут. Якщо різниця в довжинах ребер між множинами X та Y – деяке число G, то необхідно, щоб $G > 0$. Це основне для ефективності алгоритму;

4) критерій диз'юнктивності. Умова критерію – множини X та Y не повинні перетинатися.

Нижче наведено роботу алгоритму Ліна–Кернігана.

<u>АЛГОРИТМ ЛІНА–КЕРНІГАНА</u>

ВХІДНІ ДАНІ: множина точок V розмірністю N.

ВИХІДНІ ДАНІ: маршрут T_{LK}, що складається з ребер, які формують маршрут.

1) СТВОРИТИ початковий маршрут T за допомогою деякого алгоритму побудови маршруту або створити випадковий маршрут;
2) $i = 1$. ВИБРАТИ точку t_1
3) ВИБРАТИ ребро $x_1 = (t_1, t_2) \in T$
4) ВИБРАТИ ребро $y_1 = (t_2, t_3) \notin T$ таке, що $G_1 > 0$. ЯКЩО не існує такої точки, ТО ПЕРЕЙТИ до кроку 12.
5) $i = i+1$
6) ВИБРАТИ ребро $x_i = (t_{2i-1}, t_{2i}) \in T$ таке, що

a. ЯКЩО точка t_{2i} з'єднана з точкою t_1 , ТО в результаті маємо маршрут T' ТА

b. $x_i \neq y_s$ для усіх $s < i$

ЯКЩО T' є кращим маршрутом за T ТО $T=T'$ ТА ПЕРЕЙТИ до кроку 2

7) ВИБРАТИ ребро $y_i = (t_{2i}, t_{2i+1}) \notin T$ таке, що

a. $G_i > 0$

b. $y_i \neq x_s$ для усіх $s < i$ ТА

c. існує x_{i+1}

ЯКЩО таке y_i існує, ТО ПЕРЕЙТИ до кроку 5

8) ЯКЩО існує альтернатива для y_2 ТО i=2 ТА ПЕРЕЙТИ до кроку 7

9) ЯКЩО існує альтернатива для x_2 ТО i=2 ТА ПЕРЕЙТИ до кроку 6

10) ЯКЩО існує альтернатива для y_1 ТО i=1 ТА ПЕРЕЙТИ до кроку 4

11) ЯКЩО існує альтернатива для x_1 ТО i=1 ТА ПЕРЕЙТИ до кроку 3

12) ЯКЩО існує альтернатива для t_1 ТО ПЕРЕЙТИ до кроку 2

13) $T_{LK} = T$

14) КІНЕЦЬ АЛГОРИТМУ

У роботах [3, 4, 6, 14, 33] детально описано усі характеристики алгоритму та специфіка його реалізації. Сьогодні існує багато різних модифікацій евристики Ліна–Кернігана – це ітеративний Лін-Керніган (Iterated Lin-Kernighan), ланцюговий Лін–Керніган (Chained Lin-Kernighan), багаторівневий Лін–Керніган (Multilevel Lin-Kernighan), Лін–Керніган з кластерною компенсацією (Lin-Kernighan with cluster compensation) тощо [3, 4, 6, 14, 33]. Усі модифікації розроблені для застосування переважно в задачах великих розмірностей чи в специфічних задачах сьогодення.

Особливою реалізацією алгоритму Ліна–Кернігана є його модифікація Келдом Гельсгауном. Основною модифікацією алгоритму є вибір кандидатів – ребер, що можуть входити до оптимального маршруту. Множина кандидатів формується за допомогою 1-дерева, де кожен кандидат має своє π-число. У процесі виконання алгоритму ребра вибираються із множини кандидатів, сформованих з 1-дерева.

Висновки

Точний алгоритм з пакета Concorde для розв'язання задачі комівояжера, що є сучасною реалізацією методу гілок та меж, забезпечує достатньо швидке одержання оптимального розв'язку задачі при розмірності не більше 1000 точок. Евристичні алгоритми Ліна–Кернігана та Ліна–Кернігана–Гельсгауна не гарантують оптимальності розв'язку. Якість, котру вони забезпечують, дуже близька до оптимальної. Алгоритм Ліна–Кернігана показує кращий час роботи порівняно з алгоритмом Ліна–Кернігана–Гельсгауна.

Проведені дослідження дали змогу оцінити ефективність використання існуючих алгоритмів як базових при декомпозиції задачі комівояжера великих розмірностей. Алгоритм Ліна–Кернігана–Гельсгауна доцільно використовувати при декомпозиції задачі комівояжера, коли основною метою є висока якість розв'язку. Алгоритм Ліна-Кернігана доцільно використовувати для розв'язування підзадач розмірністю декілька сотень точок, а також коли основною метою є малий час обчислень.

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 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. Held, M., and Karp, R. M. (1970), "The Traveling Salesman Problem and Minimum Spanning Trees", Operations Research. 18:1138–1162. 17. Held, M., and Karp, R. M. (1971), "The Traveling Salesman Problem and Minimum Spanning Trees: part II", Mathematical Programming 1:6–25. 18. C. L. Valenzuela and A. J. Jones, ''Estimating the Held–Karp lower bound for the geometric TSP,'' Manuscript dated 8 June 1995. 19. Balas, E., and Toth, P. (1985), "Branch and Bound Methods", in The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. E. L. Lawler, J. K Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys. Eds. John Wiley and Sons, New York. 20. Held, M., Wolfe, P., and Crowder, H. P. (1974), "Validation of subgradient optimization", Mathematical Programming 6:62–88. 21. Wolsey, L. (1980), "Heuristic analysis, linear programming, and branch and bound", Mathematical Programming Study 13:121–134. 22. Shmoys, D. B., and Williamson, D. P. (1990), "Analyzing the Held–Karp TSP Bound: A Monotonicity Property with Application", Information Processing Letters. 35:281–285. 23. Christofides, N. (1979), "The Traveling Salesman Problem", in Combinatorial Optimization. N. Christophides, A. Mingozzi, P. Toth and C. Sandi. Eds. John Wiley and Sons, New York. 24. Volgenant, T., and Jonker, R. (1982), "A branch and bound algorithm for the symmetric traveling salesman problem based on the 1–tree relaxation", European Journal of Operational Research. 9:83–89. 25. Johnson, David S. (1990), "Local Optimization and the Traveling Salesman Problem", Automata Languages and Programming: 17th International Colloquium Proceedings. 26. Johnson, D. S., McGeoch L. A., and Rothberg, E.E. (1996), "Asymptotic experimental analysis for the Held–Karp traveling salesman bound", Proceeding 1996 ACM–SIAM symposium on Discrete Algorithms. 27. G. B. Dantzig, R. Fulkerson, and S. M. Johnson, Solution of a large–scale traveling salesman problem, Operations Research 2 (1954), pp. 393–410. 28. Johnson, D. S., McGeoch L. A. The traveling salesman problem: A case study in local optimization. In E.H.L. Aaarts and J.K. Lenstra, editors, Local Search in Combinatorial Optimization, pp 215–310, John Wiley & Sons, New York, 1997. 29. J. L. Bentley, ''Multidimensional binary search trees used for associative search,'' Comm. ACM 18 (1975), 309–517. 30. J. L. Bentley, ''Experiments on traveling salesman heuristics,'' in Proc. 1st Ann. ACMSIAM Symp. on Discrete Algorithms, SIAM, Philadelphia, PA, 1990a, 91-99. 31. J. L. Bentley, ''K̃ d trees for semidynamic point sets,'' in Proc. 6th Ann. Symp. on Computational Geometry, ACM, New York, 1990b, 187–197. 32. J. L. Bentley, ''Fast algorithms for geometric traveling salesman problems,'' ORSA J. Comput. 4 (1992), 387–411. 33. D. S. Johnson. Local optimization and the traveling salesman problem. In ICALP '90. pages 446–461. Springer–Verlag, 1990. Proceedings of the lith Colloquium on Automata. Languages and Programming. 34. B. Chandra. H. Karioff. and C. Tovey. 'New results on the old k–opt algorithm for The TSP. In Proceedings of the Fifth ACM–SIAM Symposium on Discrete Algorithms. Pages 150–159, 1994. 35. L. Fredman, D. S. Johnson. L. A. McGeoch. and G. Ostheimer. Data Structures for Traveling salesmen. Journal of Algorithms. 18:432–479. 1995.