Technische Universität München
Zentrum Mathematik

Diskrete Optimierung: Fallstudien aus der Praxis

Barbara Wilhelm | Michael Ritter

Der Approximationsalgorithmus von Christofides

Im Folgenden finden Sie eine formale Beschreibung des gewählten Approximationsalgorithmus. Arbeiten Sie die Beschreibung durch, machen Sie sich Notizen. Versuchen Sie, die Schritte des Algorithmus und die Beweise mit Hilfe von kleinen Skizzen nachzuvollziehen. Überlegen Sie sich, wie Sie die Erklärung dieses Algorithmus gestalten (Struktur, wichtige Ideen, Resultate). Wenn Sie Fragen haben, wenden Sie sich an Ihre Betreuer.

Problem: Traveling Salesman
Input: Ein Graph $G = (V, E)$ mit einer Distanzfunktion $d : E \to \mathbb{Q}_{\geq 0}$.

Aufgabe: Finde eine Tour, die alle Knoten des Graphen G genau einmal besucht und deren Länge bezüglich d möglichst klein ist, oder stelle fest, dass keine solche Tour existiert.

Eine *Tour* in einem Graphen ist ein Kreis, der jeden Knoten des Graphen genau einmal enthält. Das Traveling Salesman-Problem fragt also nach einer möglichst kurzen „Rundreise“, die jeden Knoten genau einmal besucht. Wie Sie bereits wissen, gibt es für allgemeines TSP Nichtapproximierbarkeitsaussagen: Die Existenz einer Approximation mit konstantem Approximationsfaktor würde $\mathcal{P} = \mathcal{NP}$ implizieren. Für eine Reihe praktisch interessanter Probleme kann man aber eine eingeschränkte Version des TSP betrachten. Von besonderer Bedeutung ist hier das *metrische TSP*.

Problem: Metrisches TSP
Input: Ein *vollständiger* Graph $G = (V, E)$ mit einer Distanzfunktion $d : E \to \mathbb{Q}_{\geq 0}$, so dass die Dreiecksungleichung gilt, d. h., für alle $i, j, k \in V$ ist

$$d(\{i, j\}) \leq d(\{i, k\}) + d(\{k, j\}).$$

Aufgabe: Finde eine Tour, die alle Knoten des Graphen G genau einmal besucht und deren Länge bezüglich d möglichst klein ist.

Im metrischen TSP ist also einerseits der zu Grunde liegende Graph vollständig (und damit ist auch die Existenz einer TSP-Tour gesichert), andererseits gilt für die Distanzen eine Dreiecksungleichung. Für alle praktischen Probleme, bei denen die Entfernungen auf einer Metrik beruhen, ergibt sich also ein solches TSP. Es lässt sich zeigen, dass auch metrisches TSP ein $\mathcal{NP}$-schweres Problem ist, die Einschränkung verändert die Komplexität also nicht. Immerhin lassen sich mit Hilfe der Dreiecksungleichung aber effiziente Approximationsalgorithmen konstruieren und mit einem solchen wollen wir uns hier beschäftigen.

Der Christofides-Algorithmus arbeitet zunächst nicht mit einer TSP-Tour, sondern mit dem einfacheren Begriff der Eulertour.

Definition 1
Sei $G = (V, E)$ ein Graph. Eine *Eulertour* in G ist ein geschlossener Weg, der jede Kante von G genau einmal enthält (Knoten dürfen mehrfach benutzt werden). Diese Definition gilt ebenso für Graphen, die Multikanten enthalten (also mehrere parallele Kanten, die dann natürlich alle in der Eulertour genau einmal enthalten sein müssen und auch alle für die Länge einer solchen Tour mitgezählt werden).

Das folgende Resultat geht auf Leonhard Euler zurück (Sie haben bestimmt schon vom „Königsberger Brückenproblem“ gehört):

Satz 2
Ein Graph $G = (V, E)$ enthält genau dann eine Eulertour, wenn der Grad aller seiner Knoten gerade ist (ein solcher Graph heißt eulersch).

Beweis. Wir skizzieren kurz einen Algorithmus, der in einem eulerschen Graphen eine Eulertour konstruiert:

1. Starte mit einem leeren Kantenzug und einem beliebigen Knoten v mit $\deg(v) > 0$, dann ist $\deg(v) \geq 2$.
2. Wähle eine noch nicht benutzte, mit v inzidente Kante und füge sie zum Kantenzug hinzu. Für den Endknoten w dieser Kante gilt dann genau einer der folgenden Fälle:
   a) Der Knoten w ist noch zu mindestens einer weiteren, bisher nicht benutzten Kante inzident. In diesem Fall führe Schritt 2 mit w statt v durch.
   b) Es gibt keine mit w inzidente Kante, die noch nicht im Kantenzug verwendet wurde. Dann muss w aber der Startknoten des Prozesses gewesen sein, da von jedem anderen Knoten jeweils eine gerade Anzahl Kanten „verbraucht“ worden ist und alle Knoten geraden Grad haben. Es entsteht somit ein geschlossener Kantenzug (entweder ist das ein einzelner Kreis oder der Kantenzug zerfällt in mehrere kantendisjunkte Kreise).
3. Durch Wiederholen dieses Algorithmus erhält man eine Zerlegung des Graphen in kantendisjunkte Kreise, die alle Kanten des Graphen überdecken. Wir fügen diese Kreise nun zu einer Eulertour zusammen: Wähle einen Knoten, der in zwei Kreisen oder Subtouren enthalten ist, und füge die beiden Kreise an dem Knoten zu einer Tour zusammen. Wiederhole das so lange, bis alle Kreise und Subtouren zu einer Eulertour zusammengefügt sind. □

Die Approximationsalgorithmen für TSP, mit denen wir uns hier beschäftigen, basieren im Wesentlichen auf der Konstruktion eines spannenden eulerschen Subgraphen und einer Eulertour durch diesen Subgraphen. Aus solch einer Eulertour kann dann eine TSP-Tour für den ursprünglichen Graphen gewonnen werden, wie folgendes Lemma zeigt.

Lemma 3
Sei $G = (V, E)$ ein vollständiger Graph mit Distanzfunktion $d : E \to \mathbb{Q}_{\geq 0}$, welche die Dreiecksungleichung erfüllt. Sei weiter $T = (V, E')$ ein spannender Subgraph von G und $\tau = (v_1, e_1, v_2, e_2, \ldots, v_l, e_l, v_1)$ mit $v_i \in V$, $e_i \in E'$ eine Eulertour der Länge $d(\tau)$ durch T. Dann existiert eine TSP-Tour durch G, deren Länge höchstens $d(\tau)$ beträgt.

Beweis. Ist τ bereits eine TSP-Tour durch G, so ist nichts zu zeigen. Da T alle Knoten von G enthält, muss τ andernfalls mindestens einen Knoten mehrfach enthalten. Wir zeigen, dass sich in diesem Fall eine Eulertour durch einen anderen spannenden Subgraphen konstruieren lässt, die ebenfalls höchstens Länge $d(\tau)$ hat. Sei dazu $k \in \{1, \ldots, l\}$ so gewählt, dass $v_1, \ldots, v_{k-1}$ paarweise verschieden sind und $v_k \in \{v_1, \ldots, v_{k-1}\}$ (k ist also der erste doppelt besuchte Knoten). Definiere dann eine neue Tour

$$\tau' = \Big(v_1, e_1, \ldots, e_{k-2}, v_{k-1}, \{v_{k-1}, v_{k+1}\}, v_{k+1}, e_{k+1}, \ldots, v_l, e_l, v_1\Big).$$

Die Tour τ' entsteht also durch „Abkürzen“ aus τ, wobei der bereits besuchte Knoten v_k einfach übersprungen wird. Die notwendige Kante $\{v_{k-1}, v_{k+1}\}$ existiert wegen Vollständigkeit von G. Aufgrund der Dreiecksungleichung gilt

$$d(\tau') = d(\tau) - \Big[d(\{v_{k-1}, v_k\}) + d(\{v_k, v_{k+1}\})\Big] + d(\{v_{k-1}, v_{k+1}\}) \leq d(\tau),$$

die neue Tour ist also höchstens genauso lang wie die alte. Wiederholte Anwendung dieser Konstruktion erzeugt schließlich eine TSP-Tour mit der gewünschten Eigenschaft. □

Das Vorgehen für die Approximation ist nun wie folgt: Wähle zuerst einen spannenden Subgraphen von G und ergänze diesen wenn nötig so, dass ein eulerscher Graph entsteht. Konstruiere eine Eulertour durch den Graphen und gewinne durch Abkürzen daraus eine TSP-Tour. Die Approximationsgüte muss sich damit bereits aus der Eulertour ergeben (und damit letztlich aus dem gewählten spannenden Subgraphen bzw. dem daraus entstehenden erweiterten eulerschen Graphen).

Wir beginnen mit einer einfachen Konstruktion, die bereits eine Approximationsgüte von 2 liefert.

1. Starte mit einem vollständigen Graphen $G = (V, E)$ mit Distanzfunktion $d : E \to \mathbb{Q}_{\geq 0}$, die der Dreiecksungleichung genügt.
2. Bestimme einen minimalen Spannbaum T in G. Das Gewicht dieses Spannbaums sei $d(T)$.
3. Verdopple alle Kanten in T. Der so entstehende Multigraph T' besitzt Parallelkanten und ist eulersch (jeder Knoten hat geraden Grad). Die Distanzfunktion auf den neuen Kanten sei identisch zur Distanzfunktion auf den alten Kanten, das Gesamtgewicht von T' ist also $2d(T)$.
4. Bestimme eine Eulertour in T'. Diese hat Länge $2d(T)$. Durch Abkürzen entsteht daraus eine TSP-Tour τ mit $d(\tau) \leq 2d(T)$.

Satz 4
Die nach obigem Algorithmus bestimmte TSP-Tour τ erfüllt die Ungleichung

$$d(\tau) \leq 2d(\tau_{\text{opt}}),$$

wobei $d(\tau_{\text{opt}})$ die Länge einer optimalen TSP-Tour ist.

Beweis. Jede TSP-Tour durch G enthält sämtliche Knoten von G und genau $|V| = n$ Kanten. Sei τ_{opt} eine optimale TSP-Tour in G. Durch Entfernen einer beliebigen Kante aus τ_{opt} entsteht ein Spannbaum in G. Da der Algorithmus zunächst einen optimalen (d. h. minimales Gesamtgewicht) Spannbaum T bestimmt, gilt $d(\tau_{\text{opt}}) \geq d(T)$. Zusammen mit der Ungleichung $d(\tau) \leq 2d(T)$, die sich unmittelbar aus dem Algorithmus ergibt, folgt

$$d(\tau) \leq 2d(T) \leq 2d(\tau_{\text{opt}}),$$

also liefert der Algorithmus eine 2-Approximation. □

Wir kommen nun zum Christofides-Algorithmus. Der wesentliche Schwachpunkt der 2-Approximation besteht in der Tatsache, dass der berechnete Spannbaum T einfach verdoppelt wird, um einen eulerschen Graphen zu erhalten. Das macht zwar die Abschätzung einfach, erzeugt aber häufig unnötig viele Kanten, da manche Knoten des Spannbaums ja vielleicht bereits geraden Grad haben. Der Schlüssel zu einer besseren Methode liegt in folgendem Lemma.

Lemma 5
Sei $T = (V, E)$ ein Graph und $V_{ungerade} := \{v \in V : \deg(v)$ ungerade$\}$. Dann ist die Kardinalität $|V_{ungerade}|$ gerade, d. h. der Graph besitzt eine gerade Anzahl von Knoten mit ungeradem Grad.

Beweis. Sei $n = |V|$, $m = |E|$. Es gilt:

$$\sum_{v \in V} \deg(v) = 2m \quad \text{(gerade)}$$

Gäbe es eine ungerade Anzahl von Knoten in V, deren Grad ungerade ist, so enthielte die obige Summe eine ungerade Anzahl ungerader Summanden (und zusätzlich gerade Summanden), wäre also selbst ungerade, Widerspruch. □

Der Christofides-Algorithmus funktioniert wie folgt:

1. Starte mit einem vollständigen Graphen $G = (V, E)$ mit Distanzfunktion $d : E \to \mathbb{Q}_{\geq 0}$, welche die Dreiecksungleichung erfüllt.
2. Bestimme einen minimalen Spannbaum $T = (V, E_T)$ in G. Das Gewicht dieses Spannbaums sei $d(T)$. (Welche Algorithmen dazu kennen Sie? Welche Laufzeit haben diese Algorithmen?)
3. Bestimme ein perfektes Matching $M \subseteq E$ (warum gibt es das?) minimalen Gewichts in der Menge $\{v \in V : \deg_T(v)$ ungerade$\}$. Der Graph $T' = (V, E_T \cup M)$ ist dann ein spannender eulerscher Subgraph von G (jeder Knoten hat geraden Grad). Das Gesamtgewicht von T' ist $d(T') = \sum_{e \in E_T} d(e) + \sum_{e \in M} d(e)$.
4. Bestimme eine Eulertour in T'. Durch Abkürzen entsteht daraus eine TSP-Tour τ mit $d(\tau) \leq d(T')$.

Satz 6
Der Christofides-Algorithmus liefert eine $3/2$-Approximation für das metrische Traveling Salesman-Problem.

Beweis. Wir verwenden die Bezeichnung aus dem Christofides-Algorithmus oben. Sei τ_{opt} eine optimale TSP-Tour in G und τ die vom Christofides-Algorithmus gelieferte TSP-Tour. Da τ_{opt} nach Entfernen einer beliebigen Kante ein Spannbaum in G ist und da T ein minimaler Spannbaum in G ist, gilt

$$d(\tau_{\text{opt}}) \geq d(T).$$

Seien die Knoten $v_1, v_2, \ldots, v_n$ ohne Beschränkung der Allgemeinheit in der Reihenfolge nummeriert, in der sie von der optimalen Tour τ_{opt} besucht werden (bei beliebigem Anfang). Dann sind

$$M_1 := \{\{v_1, v_2\}, \{v_3, v_4\}, \ldots\}$$
$$M_2 := \{\{v_2, v_3\}, \{v_4, v_5\}, \ldots\}$$

Matchings in G. Sie enthalten entweder alle Knoten von G oder alle bis auf einen Knoten (falls die Knotenzahl von G ungerade ist). Die TSP-Tour τ_{opt} bildet einen Kreis in G, der abwechselnd Kanten aus M_1 und M_2 benutzt, es gilt somit

$$d(M_1) + d(M_2) \leq d(\tau_{\text{opt}}).$$

Da M (aus dem Algorithmus) ein Matching minimalen Gewichts auf einer geraden Teilmenge der Knoten von G darstellt, ist $d(M) \leq d(M_1)$ und $d(M) \leq d(M_2)$. Zusammen gilt also

$$2d(M) = d(M) + d(M) \leq d(M_1) + d(M_2) \leq d(\tau_{\text{opt}}).$$

Da die vom Christofides-Algorithmus berechnete Eulertour sich aus den Kanten von T und den Kanten von M zusammensetzt, und die berechnete TSP-Tour τ höchstens so lang ist wie diese Eulertour, folgt schließlich

$$d(\tau) \leq d(M) + d(T) \leq \frac{1}{2}d(\tau_{\text{opt}}) + d(\tau_{\text{opt}}) = \frac{3}{2}d(\tau_{\text{opt}}),$$

das beweist die behauptete Approximationsgüte. □