Simulated Annealing

und andere Suchmethoden

Carsten Franke

Züricher Hochschule für Angewandte Wissenschaften

09. Mai 2012

Lernziele

- Verschiedene heuristische Optimierungsverfahren vom Prinzip her verstehen
- Eine Implementierung ist dann in der Regel nicht schwer und sehr problemabhängig. Daher wird eine Implementierung hier nicht näher besprochen

Inhalt

Tabu Search

- basiert auf der systematischen Nutzung eines Gedächtnisses während der Suche
  - Kurzzeitgedächtnis beschränkt die Nachbarschaft $N(s)$ einer derzeitigen Lösung s auf die Untermenge $N'(s) \subseteq N(s)$
  - Langzeitgedächtnis erweitert $N(s)$ durch die Einbeziehung von weiteren Lösungen
- Tabu Search nutzt eine iterative, lokale Suche und bewegt sich stets zur besten Lösung in der Nachbarschaft, selbst wenn diese Lösung schlechter als die derzeitige Lösung ist (die beste Lösung der schlechteren).
- Schleifen werden vermieden, da die letzten Lösungen **tabu** sind.
- die tabu Liste hat eine bestimmte Länge, so dass die Lösungen nur für eine gewisse Anzahl Iterationen tabu sind.
- Simple tabu Suche = Suche nur mit Kurzzeitgedächtnis
- das Langzeitgedächtnis wird oft genutzt um die Häufigkeiten von Teillösungen zu identifizieren.

Tabu Search

Simple Tabu Search

$s \leftarrow$ Generiere initiale Lösung
initialisiere das Kurzzeitgedächtnis
$s_{best} \leftarrow s$
while Beendigungsbedingung nicht erfüllt **do**
  $A \leftarrow$ Generiere **mögliche** Lösungen im Umfeld(s)
  $s \leftarrow$ Wähle die beste Lösung aus A aus
  Update des Kurzzeitgedächntnisses
  if $(f(s) < f(s_{best}))$ **then**
    $s_{best} = s$
  end if
end while
return s_{best}

Geleitete Lokale Suche

- Eine Möglichkeit lokale Optima während der Suche zu vermeiden ist die Optimierungsfunktion dynamisch anzupassen.
- Zu diesem Zweck wird eine neue Funktion $h(s)$, $h : s \rightarrow \mathbb{R}$ einführt, die die ursprüngliche Funktion $f(\cdot)$ enthält und weitere Strafterme pn_i die mit den Funktionsteilen i assoziiert sind
- Die Funktion hat dann etwa folgendes Aussehen:
  $h(s) = f(s) + w \cdot \sum_{i=1}^{n} pn_i \cdot I_i(s)$, wobei w das Gewicht für die Strafterme beschreibt
- $I_i(s)$ ist eine Indikatorfunktion, die den Wert 1 annimmt, wenn die Lösung den Funktionteil beinhaltet und 0 sonst.

Indikatorfunktion

Beim TSP hat die Indikatorfunktion den Wert 1, wenn eine bestimmte Kante Element der Lösung ist und 0 sonst.

Geleitete Lokale Suche

- Wann immer die geleitete Lokale Suche in einem lokalen Optimum $\hat{s}$ gemäss $h(\cdot)$,,gefangen" ist, wird ein Nutzwert $u_i = \frac{I_i(\hat{s}) \cdot c_i}{1+pn_i}$ für jeden Teil berechnet, wobei c_i den jeweiligen Kostenanteil beschreibt
  - Bestandteile mit hohen Kosten haben hohen Einfluss
  - Um den Einfluss der besonders grossen Einflussfaktoren zu vermindern, werden die Strafterme für alle Terme mit hohem Nutzwert entsprechend angehoben
  - dann wird mit einer lokalen Suche ausgehend von $\hat{s}$ fortgefahren
  - während der lokalen Suche müssen dann die originale Optimierungsfunktion und die Funktion $h(s)$ ausgewertet werden. Die originale Optimierungsfunktion beschreibt die Qualität, die Funktion $h(s)$ die Richtung der Suche.

Geleitete Lokale Suche

Geleitete Lokale Suche

$s \leftarrow$ Generiere initiale Lösung
initialisiere die Strafterme
$s_{best} \leftarrow s$
while Beendigungsbedingung nicht erfüllt **do**
  $h \leftarrow$ berechne veränderte Funktion
  $\hat{s} \leftarrow$ lokale Suche $(\hat{s},h)$
  Update der Strafterme
  if $(f(s) < f(s_{best}))$ **then**
    $s_{best} = s$
  end if
end while
return s_{best}

Iterative Lokale Suche

- es wird erneut von einer initialen Startlösung ausgegangen
- eine lokale Suche wird durchgeführt bis ein lokales Optimum $\hat{s}$ gefunden ist
- die Lösung wird dann einfach zufällig in die Nachbarschaft zu s' bewegt (nicht unbedingt optimal)
- ausgehend von s' wird eine neue lokale Suche durchgeführt, die zum lokalen Optimum $\hat{s'}$ führt.
- ein Akzeptanzkriterium wird genutzt, um zu entscheiden, ob die Suche bei $\hat{s}$ oder $\hat{s'}$ forgesetzt wird

Iterative Lokale Suche

Iterative Lokale Suche

$s \leftarrow$ Generiere initiale Lösung
$\hat{s} \leftarrow$ lokale Suche (s)
$s_{best} \leftarrow \hat{s}$
while Beendigungsbedingung nicht erfüllt **do**
  $s' \leftarrow$ Variation $(\hat{s})$
  $\hat{s'} \leftarrow$ lokale Suche (s')
  if $(f(\hat{s'}) < f(s_{best}))$ **then**
    $s_{best} = \hat{s'}$
  end if
  $\hat{s} \leftarrow$ Akzeptanzkriterium$(\hat{s}, \hat{s'})$
end while
return s_{best}

Greedy zufällige adaptive Suche

- basiert auf der zufälligen Generierung einer Vielzahl von initialen Lösungen, von denen aus dann gesucht wird
- die verschiedenen Implementierungen unterscheiden sich in der Art, wie die initialen Lösungen generiert werden

Greedy zufällige adaptive Suche

Greedy zufällige adaptive Suche

while Beendigungsbedingung nicht erfüllt **do**
  $s \leftarrow$ generiere greedy zufällige adaptive Lösung
  $\hat{s} \leftarrow$ lokale Suche (s)
  if $(f(\hat{s}) < f(s_{best}))$ **then**
    $s_{best} = \hat{s}$
  end if
end while
return s_{best}

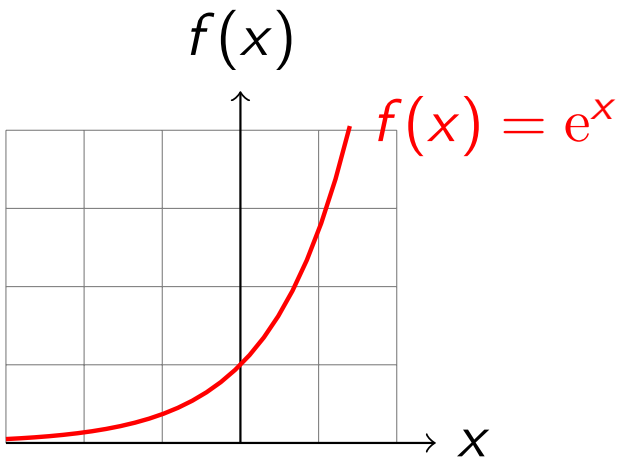
Simulated Annealing

- die Methode ist der Physik nachempfunden
- ein Metall wird erst geschmolzen und kühlt dann ganz langsam wieder ab
- im Endstadium ist die Struktur wieder sehr fest und energetisch optimal
- zwischendurch wird hin und wieder eine schlechtere Struktur akzeptiert (verhindert lokale Optima)

Simulated Annealing

transferiert diesen Prozess zur Verwendung für lokale Suchalgorithmen und kombinatorische Optimierung.

Metropolis Funktion



Metropolis Verteilungsfunktion

$$MV(E_{neu}, E_{alt}, T) = \begin{cases} 1, \text{wenn } E_{neu} < E_{alt} \\ e^{\left(-\frac{(E_{neu} - E_{alt})}{k_B \cdot T}\right)}, \text{sonst} \end{cases}$$

- E_{neu} : neuer Energiezustand
- E_{alt} : alter Energiezustand
- k_B : Boltzmann-Konstante
- T : Temperatur

Simulated Annealing

Simulated Annealing Algorithmus (für Minimierung)

$s \leftarrow$ Generiere initiale Lösung
$T \leftarrow$ initiale hohe Temperatur
$s_{best} \leftarrow s$
$n \leftarrow 0$
while äussere Schleifenbedingung nicht erfüllt **do**
  while innere Schleifenbedingung nicht erfüllt **do**
    $s' \leftarrow$ generiere Nachbar(s)
    $s \leftarrow s$ mit Wahrscheinlichkeit gemäss $MV(s, s', T)$
    if $(f(s) < f(s_{best}))$ **then**
      $s_{best} = s$
    end if
  end while
  T=neue Temperatur(T)
  $n \leftarrow n + 1$
end while
return s_{best}

Parameteranpassungen

Es müssen für den allgemeinen Algorithmus eine Vielzahl von Vereinbarungen getroffen werden.

- Funktion zur Erstellung der Initiallösung.
- Eine Funktion um die Nachbarschaft einer Lösung zu beschreiben.
- Prozess des Temperaturabkühlens $T_{neu} = T \cdot c$ mit $c \in]0, 1[$ (geometrisches Abkühlen)
- Anzahl der inneren Schleifendurchläufe (bei gleicher Temperatur)
- Anzahl der äusseren Schleifendurchläufe

Anwendungsbeispiel TSP

- initiale Tour: $T = (v_1, v_2, \cdots, v_m, v_1)$
- Ziel ist es eine bessere Tour ausgehend von T zu erstellen
- Verbesserungen könnten entstehen, indem einige Kanten durch andere ersetzt werden
- Prinzip entspricht der lokalen Suche.
- Unterschiede bestehen durch die unterschiedlichen Möglichkeiten benachbarte Lösungen zu definieren

TSP Verfahren

Verbesserungsverfahren für TSP = Kantenaustauschverfahren

k-OPT

- zentrale Idee: Ersetze $k > 1$ Kanten der aktuellen Tour T durch k **neue** Kanten. Die Mengen der entfernten und der hinzugefügten Kanten muss dabei nicht unbedingt disjunkt sein.

Definition (k-OPT Nachbarschaft)

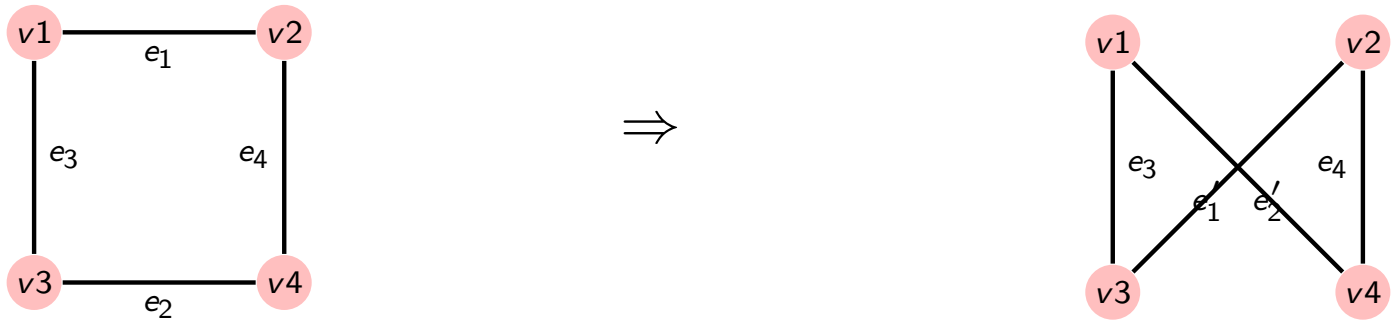
Die **k-Opt-Nachbarschaft** einer Tour T besteht aus allen Touren, die durch Entfernen von k Kanten aus T und anschließendes Hinzufügen von k Kanten erzeugt werden können.

Definition (k-Optimalität)

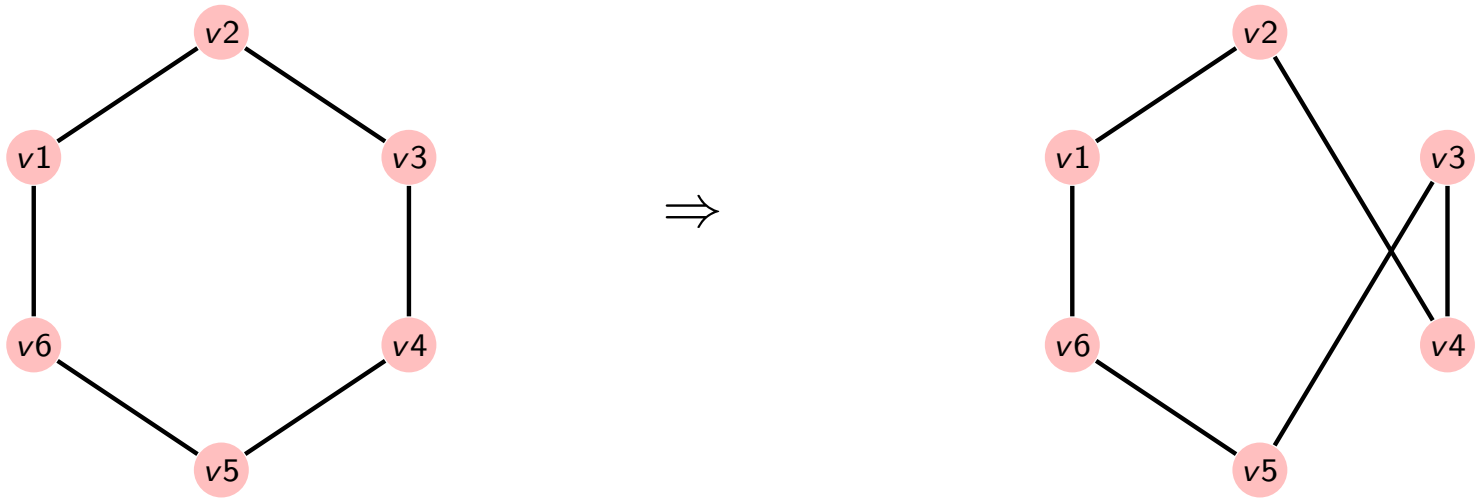
Eine Tour T heißt **k-optimal**, wenn sie innerhalb ihrer k-Opt-Nachbarschaft minimale Länge aufweist.

Beispiel 2-OPT

- das einfachste k-OPT Verfahren ist das 2-OPT Verfahren
- es werden zwei Kanten (e_1, e_2) entfernt
- dann gibt es nur **genau eine** Möglichkeit, diese beiden Kanten durch zwei neue Kanten (e_1', e_2') zu ersetzen:



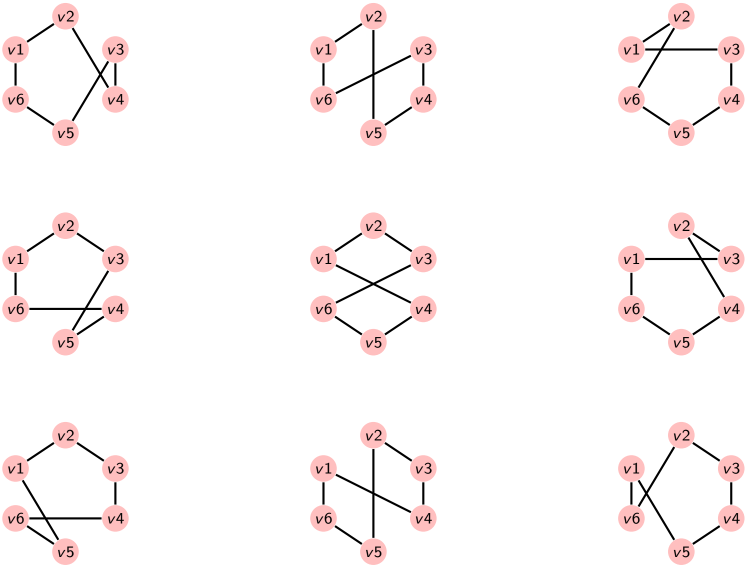
2-OPT für m=6 Städte



Übung

Bestimmen Sie alle möglichen 2-OPT Nachbarschaften für m=6 Städte.

2-OPT Nachbarschaften für m=6 Städte



Anzahl möglicher 2-OPT Tausche für m Städte

Übung

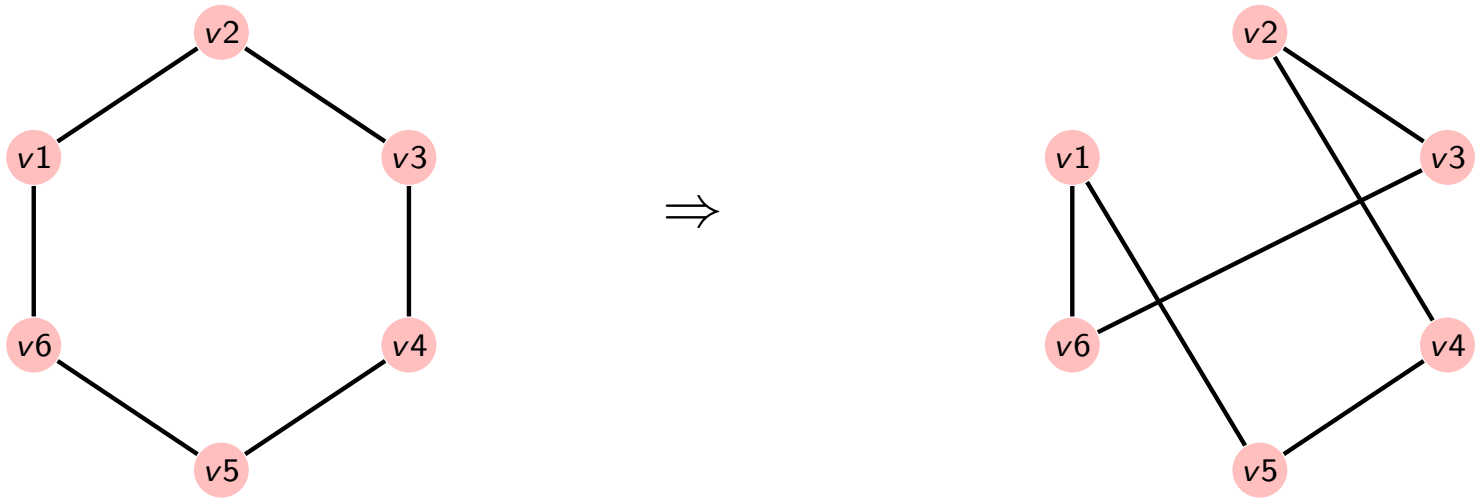
Bestimmen Sie die Anzahl möglicher 2-OPT Tausche für $m \geq 3$ Städte. Formulieren Sie die Lösung in Abhängigkeit von m.

Anzahl möglicher 2-OPT Tausche für m Städte

Lösung

Für $m \geq 3$ gibt es $\frac{m \cdot (m-3)}{2}$ mögliche 2-OPT Tausche.

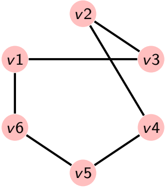
3-OPT für m=6 Städte

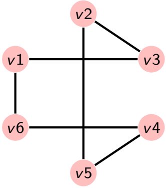


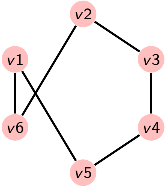
Übung

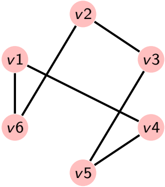
Bestimmen Sie alle übrigen 3-OPT Nachbarschaften für m=6 Städte für die gelöschten 3 Kanten.

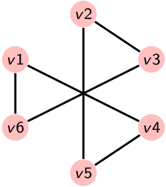
Übrige 3-OPT Nachbarschaften

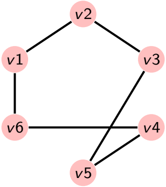












Anzahl k-OPT Schritte für allgemeines k

- häufig werden nur k=2 und k=3 OPT Verfahren implementiert
- Ursache: die Anzahl der Schritte wächst exponentiell mit k

Lemma (Obere Schranke für k-OPT Schritte)

Es sei $m > k > 3$. *Dann ist der Wert* $\binom{m}{k} \cdot (k-1)! \cdot 2^{k-1}$ *eine obere Schranke für die Anzahl der k-OPT Schritte.*

TSP mittels Simulated Annealing und k-OPT

- Startlösung: Nächster-Nachbar-Heuristik
- Nachbarschaft mittels 2-OPT (als ein Beispiel)
- Abbruchkriterien
  - Keine Verbesserung der TSP Lösung innerhalb von **x** Iterationen
  - Die Akzeptanzwarscheinlichkeit (mittels Metropolis-Funktion) unterschreitet den Wert **y**
  - Sinnvoll ist oft eine Kombination der beiden Kriterien

Lösung des TSP mittels Simulated Annealing

Implementieren Sie das TSP

- Initiallösung: Nächster-Nachbar-Lösung
- Verwendung der 2-OPT Methode
- zufällige Auswahl zweier Kanten für die 2-OPT Methode
- Wenn die Lösung direkt verbessert wurde, wird die Lösung übernommen. Ansonsten wird die Metropolis-Funktion genutzt.
- Abbruch wenn 200 Versuche zu keiner Verbesserung geführt haben.
- Temperaturverminderung von 300 Grad bis 20 Grad in 5 Grad Schritten
- Datei: Nutzen Sie die Datei Entfernungen_schweizer_Staedte.txt als Eingabe.

Vorlesungsplanung - Termine (Änderungen vorbehalten)

1. 21.02.2012: Einkriterielle Evolutionäre Optimierung I (CF)
2. 28.02.2012: Einkriterielle Evolutionäre Optimierung II (CF)
3. 06.03.2012: Test (1+2), Mehrkriterielle Evolutionäre Optimierung I (CF)
4. 13.03.2012: Statistische Lerntheorie I (JP)
5. 20.03.2012: Statistische Lerntheorie II (JP)
6. 27.03.2012: Test (4+5), Neuronale Netze (JP)
7. 10.04.2012: Mehrkriterielle Evolutionäre Optimierung II (CF)
8. 08.05.2012: Genetische Fuzzy Systeme (CF)
9. 09.05.2012: Test (3+7+8) Simulated Annealing, andere Suchmethoden (CF)
10. 15.05.2012: Meta-Heuristiken (ACO, PSO) (CF)
11. 22.05.2012: Support Vector Maschinen I (JP)
12. 29.05.2012: Support Vector Maschinen II (JP)
13. 05.06.2012: Test (6+7+12), Clustering (JP)
14. 12.06.2012: Lernen und Spieltheorie (JP)
15. 26.06.2012: 1. Termin mündliche Prüfungen
16. 03.07.2012: 2. Termin mündliche Prüfungen