Kap. 7.6 Verbesserungsheuristiken (letzte VO)

Professor Dr. Petra Mutzel
Lehrstuhl für Algorithm Engineering, LS11
Fakultät für Informatik, TU Dortmund

25. VO DAP2 SS 2009 23. Juli 2009

Überblick

- Verbesserungsheuristiken
  - Einfache lokale Suche
  - Simulated Annealing
  - Evolutionäre Algorithmen
  - Alternative Verfahren
- Klausur: noch etwas unklar?

Motivation

„Warum soll ich heute hier bleiben?"
interessant + höchst praxisrelevant!

„Was gibt es heute Besonderes?"
Ankündigung bzgl. Klausur!!!

Kap. 7.6 Verbesserungsheuristiken

- Verbesserungsheuristiken nehmen als Input eine zulässige Lösung für ein OP und versuchen diese durch „lokale" Änderungen zu verbessern.

7.6.1 Einfache lokale Suche

„Local Search"

- **Def.:** Die **Nachbarschaft** $N(x)$ (engl: *neighborhood*) einer Lösung x ist die Menge aller Lösungen, die von einer aktuellen Lösung x aus durch eine kleine Änderung – einen sogenannten Zug – erreichbar sind.

- Z.B. r-Nachbarschaft von x bei Rucksackproblem für festes r:
  Menge aller Lösungen, die sich von der Lösung x um maximal r Gegenstände unterscheiden.

Nachbarschaft einer Lösung

- Für Probleme, die sich durch binäre Vektoren kodieren lassen, ist der **Hammingabstand** ein sinnvoller Nachbarschaftsbegriff.
- Der **Hammingabstand** von $x,y \in \{0,1\}^n$ ist die Anzahl der i mit $x_i \neq y_i$ für $i=1,\ldots,n$.
- **Nachbarn** von x sind alle Punkte (entspricht Lösungen), die von x einen durch d beschränkten Hammingabstand haben.

Prinzip der einfachen lokalen Suche

Methode:

1. Sei x eine zulässige Lösung (Input)
2. **Repeat**
3. Wähle $y \in N(x)$
4. **If** y besser als x **then**
5. $x := y$ „y wird akzeptiert“
6. **until** Abbruchkriterium erfüllt

Kritisch ist Schritt 3

Auswahlstrategien für Schritt (3)

(1) **Random neighbor:** Es wird eine Nachbarlösung zufällig abgeleitet → schnell, aber y ist oft schlechter als x

(2) **Best improvement:** die beste Lösung aus $N(x)$ wird ausgewählt → oft langsam

(3) **First improvement:** $N(x)$ wird systematisch durchsucht, bis die erste Lösung besser als x gefunden wird oder alle Nachbarn betrachtet wurden.

Beste Strategie? → problemabhängig

aktive Forschung zu (2)

Abbruchkriterium

(1) Wenn in $N(x)$ keine Lösung existiert, die besser als die aktuelle Lösung ist, dann hat man ein lokales Optimum erreicht.

(2) Definition: x ist **lokales Optimum** wenn $f(y) \geq f(x)$ für alle $y \in N(x)$ bei Minimierungsaufgabe, und $f(y) \leq f(x)$ für alle $y \in N(x)$ für Maximierungsaufgabe.

(3) i.A. ist jedoch ein lokales Optimum nicht ein globales Optimum (das wir suchen)

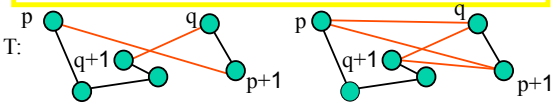
(4) Der Wert eines lokalen Optimum kann beliebig weit vom Wert eines globalen Optimum entfernt sein.

Beispiel: 2-OPT für TSP

(1) Wähle eine beliebige Anfangstour T. Die Knoten seien so nummeriert, wie sie in der Tour vorkommen: $T=\{(1,2),(3,4),(5,6),\ldots,(n,n+1)\}$ mit $n+1=1$.

(2) Für alle nicht-adjazenten Kantenpaare $(p,p+1),(q,q+1)$

- Falls $c_{p,p+1} + c_{q,q+1} > c_{pq} + c_{p+1,q+1}$ dann setze:
- $T := T \setminus \{(p,p+1),(q,q+1)\} \cup \{(p,q),(p+1,q+1)\}$
- Wiederhole (2)



Beispiel: r-OPT für TSP

(1) Wähle eine beliebige Anfangstour T.

(2) Sei Z die Menge aller r-elementigen Teilmengen von T

(3) Für alle $R \in Z$: setze $S := T \setminus R$ und konstruiere alle Touren, die S enthalten. Gibt es eine unter diesen, die besser als T ist, wird diese unser neues T

(4) Gehe zu (2)

Eine Tour, die durch einen r-Austausch nicht mehr verbessert werden kann, heißt **r-optimal.**

Diskussion

- Einfache lokale Suche ist in der Praxis sehr beliebt: deutliche Verbesserung innerhalb kurzer Zeit möglich.
- Für das TSP kommt 3-Opt meist sehr nahe an die optimale Lösung (ca. 3-4% nahe am Optimum). Jedoch erhält man bereits für $N \geq 200$ Städte sehr hohe Rechenzeiten, da der Aufwand pro Iteration bei $O(N^3)$ liegt.
- In der Praxis am besten: Lin-Kernighan Heuristik (1973): r variabel, meist $r \in \{2,3\}$: führt zu Lösungen, die 1-2% nahe am Optimum sind.

7.6.2 Simulated Annealing (SA)

- Problem bei einfacher lokaler Suche: man bleibt relativ schnell in lokalen Optima hängen, die relativ weit von der Optimallösung entfernt sind.
- Denn: es werden nur bessere Lösungen akzeptiert.
- Simulated Annealing ist eine allgemeine Erweiterung (**Meta-Heuristik**) der einfachen lokalen Suche.
- **Idee:** Akzeptiere manchmal (nicht allzu-oft) auch schlechtere Lösungen

Idee von Simulated Annealing

Kirkpatrick, Gelatt und Vecchi 1983
einfache Variante: Metropolis et al. 1953

- **Idee:** simuliert physikalischen Prozess, ein Metall in einen Zustand möglichst geringer innerer Energie zu bringen, in dem die Teilchen möglichst strukturiert angeordnet sind.
- Dabei wird das Metall erhitzt und sehr langsam abgekühlt.
- Die Teilchen verändern anfangs ihre Positionen und damit ihre Energieniveaus sehr stark, mit sinkender Temperatur aber geringer.

Simulated Annealing

(1) Sei x: zulässige Lösung (Input), t: Zeit; T: aktuelle Temperatur; T_0: InitTemperatur, Z: Zufallszahl $\in [0,1)$
(2) $t:=0$; $T:=T_0$
(3) **Repeat**
(4) Wähle $y \in N(x)$ zufällig
(5) **If** y besser als x **then**
(6) $x:=y$
(7) **else if** $Z < e^{-|f(y)-f(x)| / T}$ **then** $x:=y$
(8) $t:=t+1$; $T:=g(T,t)$
(9) **until** Abbruchkriterium erfüllt

nur geringfügig schlechtere Lösungen werden mit höherer Wahrscheinlichkeit akzeptiert als viel schlechtere

anfangs ist T höher, d.h. Akzeptanz für schlechtere Lösungen anfangs höher

Steuerung von T

- Die Akzeptanzwahrscheinlichkeit wird von Temperatur T und Fitnessdifferenz $f(y)-f(x)$ gesteuert.
- **Wichtig:** Geeignete Steuerung von T: anfangs T eher groß (d.h. Verschlecherung wird fast immer akzeptiert) und zuletzt T sehr klein (selten akzeptiert).
- $T=0$ entspricht der einfachen lokalen Suche, bei der immer nur bessere Lösungen akzeptiert werden.
- $g(T,t)$ ist Funktion, die von T und t abhängt.

Steuerung von T

- Das Cooling-Schema beschreibt, wie T in Abhängigkeit der Zeit t vermindert wird.
- **Geometrisches Cooling**: z.B. $g(T,t):=\alpha^{t-1}T_0$, $\alpha<1$ (z.B. 0,999)
- **Adaptives Cooling:** Der Anteil der Verbesserungen an den letzten erzeugten Lösungen wird gemessen und auf Grund dessen T stärker oder schwächer reduziert.

Abbruchkriterien

Einige Möglichkeiten:

- Ablauf einer bestimmten Zeit
- eine gefundene Lösung ist „gut genug“
- keine Verbesserung in den letzten k Iterationen

Bedingungen, damit SA gut funktioniert:

- **Lokalitätsbedingung: hohe Lokalität:** Eine Nachbarlösung muss aus ihrer Ausgangslösung durch eine „kleine“ Änderung abgeleitet werden können. Dieser kleine Unterschied muss i.A. auch eine kleine Änderung der Bewertung bewirken.
- **Erreichbarkeit eines Optimums:** Ausgehend von jeder möglichen Lösung muss eine optimale Lösung durch wiederholte Anwendung der Nachbarschaftsfunktion grundsätzlich mit einer Wahrscheinlichkeit >0 erreichbar sein.

7.6.3 Evolutionäre Algorithmen

- Unter dem Begriff **evolutionäre Algorithmen (EA)** werden eine Reihe von Meta-Heuristiken zusammengefasst, die Grundprinzipien der natürlichen Evolution in einfacher Weise nachahmen.
- Z.B. genetische Algorithmen, Evolutionsstrategien, Evolutionary Programming, Genetic Programming,…
- Unterschied zu Simulated Annealing: Es wird nicht mehr nur mit einer aktuellen Lösung gearbeitet, sondern mit einer Population (Menge) von Lösungen.

Evolutionäre Algorithmen

- Durch diese größere Vielfalt ist die Suche nach einer optimalen Lösung „breiter“ und robuster, d.h. die Chance lokalen Optima zu entkommen ist größer.

Schema zu EA

var Q_s: selektierte Eltern, Q_r: Zwischenlösungen
(1) P:= Menge von Ausgangslösungen
(2) bewerte(P)
(3) **Repeat**
(4) Q_s := Selektion(P)
(5) Q_r := Rekombination(Q_s)
(6) P := Mutation(Q_r)
(7) bewerte(P)
(8) **until** Abbruchkriterium erfüllt

Selektion in EA

- Die Selektion kopiert aus der aktuellen Population P Lösungen, die in den weiteren Schritten neue Nachkommen produzieren werden.
- Dabei werden grundsätzlich bessere Lösungen mit höherer Wahrscheinlichkeit gewählt.
- Schlechtere Lösungen haben aber i.A. auch Chancen selektiert zu werden.
- z.B. Tournament Selektion: Wähle k Lösungen aus P gleichverteilt zufällig aus (Mehrfachauswahl erlaubt). Die beste der k Lösungen ist die selektierte.

Rekombination in EA

- Generiere eine neue Lösung aus zwei selektierten Elternlösungen.
- Beachte die Lokalitätsbedingung!

- **Beispiel:** Edge-Recombination (ERX) bei TSP:
- **Ziel:** Aus den Kanten der Elterntouren T1 und T2 soll eine neue Tour erstellt werden, die möglichst nur aus Kanten der Eltern aufgebaut ist.
- Beginne bei v. Wiederhole: Wähle als nächstes zufällig einen unbesuchten Knoten w, der in T1 ∪ T2 adjazent zu v ist (falls existiert); Sonst wähle w zufällig aus unbesuchten Knoten; setze v:=w

Mutation in EA

- Mutation entspricht der Nachbarschaftsfunktion beim Simulated Annealing.
- Sie dient meist dazu, neue oder verlorengegangene Merkmale in die Population hineinzubringen.

Diskussion

- Evolutionäre Algorithmen werden oft in der Praxis eingesetzt
- Vorteil: größere Vielfalt, deswegen gut geeignet für Optimierung mit mehreren Zielfunktionen (multi-objective)
- Nachteil: im Vergleich zu Simulated Annealing relativ langsam

- LS11 und LS2 forschen u.a. auf diesem Gebiet und bieten viele LVAs diesbezüglich an
- auch an Local-Search-Verfahren wird noch sehr viel geforscht.

Alternative Heuristiken

- Es gibt noch einige andere Klassen von Meta-Heuristiken:
- Tabu-Suche: ähnlich wie einfache lokale Suche, bei der man auch schlechtere Lösungen akzeptiert; um „cycling“ zu verhindern merkt man sich Lösungen, bei denen man bereits war (tabu)
- Ameisen-Verfahren (Ant-Colony, ACO): Pheromone
- Sintflut-Verfahren: entspricht SA, aber „Temperatur fällt“ wird ersetzt durch „Wasser steigt“
- DNA-Computing: Reagenzglas

Weiterführende Literatur zu Optimierung

- Papadimitriou und Steiglitz: Combinatorial Optimization: Algorithms and Complexity, Prentice Hall, Inc., New Jersey 1982
- Cook, Cunningham, Pulleyblank und Schrijver: Combinatorial Optimization, John Wiley & Sons, Chichester 1998
- Aarts und Lenstra: Local Search in Combinatorial Optimization, John Wiley & Sons, Chichester 1997
- Michalewicz: Genetic Algorithms+Data Structures=Evolution Programs, Springer, 1996

Wichtigste Themen der VO

- Analyse: O/Θ/Ω-Notation
- ADT
- Sortieren: QuickS,MergeS, CountingS, RadixSort
- Heap+HeapSort+Priority Queues
- Suchen: Binary Search
- Binäre Suchbäume, AVL-Bäume, B-Bäume
- Skiplisten
- Hashing: Quadratic, Double (+Brent)
- Graphenalgorithmen: MST, Kürzeste Wege
- Optimierung: Greedy-Heuristiken, Approximations-algorithmen, B&B, DynProg, Local-Search ff

Klausur

- Klausur am 31. Juli 2009 um 10:15-11:45 Uhr
- Wo? Bitte beachten Sie die Hörsaaleinteilung im Web

- Den in der Vorlesung+Übung behandelten Stoff (außer Kap. 3.3 „Externe Sortierverfahren“)
- für die Juli-Klausur gilt: nicht Kapitel 7: Optimierung

- Alle Teilnehmer/innen müssen angemeldet sein (s. Informationen auf Web-Seite zu DAP2
- Fragen zur Zulassung bzw. Anmeldung beantwortet: dap2klausur2009@ls2.cs.tu-dortmund.de

VIEL ERFOLG!