Methoden und Anwendungen der Optimierung (MAO)

Kapitel 5: Metaheuristiken – Large Neighborhood Search

Univ.-Prof. Dr. Michael Schneider
Christian Schröder

Deutsche Post Chair – Optimization of Distribution Networks (DPO)
RWTH Aachen University

schroeder@dpo.rwth-aachen.de

WS 2017/18

Deutsche Post
Chair - Optimization of
Distribution Networks

RWTH AACHEN UNIVERSITY

Gesamtgliederung

Agenda

5 Metaheuristiken – Large Neighborhood Search
- Large Neighborhood Search
- Adaptive Large Neighborhood Search

Agenda

5 Metaheuristiken – Large Neighborhood Search
- Large Neighborhood Search
- Adaptive Large Neighborhood Search

Allgemeines Prinzip der Large Neighborhood Search

- Auch unter dem Namen Ruin and Recreate oder Fix and Optimize bekannt
- Idee: Graduelle Verbesserung einer Startlösung durch sukzessive Anwendung von Destroy- und Repair-Operatoren
  - Destroy-Operatoren entfernen einen Teil der aktuellen Lösung
  - Repair-Operatoren bauen entfernte Lösungsteile wieder auf
- wirkt den Nachteilen lokaler Suche entgegen: große Änderungen der Lösung mit Potential in weitere vielversprechende Regionen des Lösungsraums vorzudringen

Pseudocode Large Neighborhood Search

Algorithmus 1 : Large Neighborhood Search

```
// Input: Zulässige Lösung x
SETZE x^b = x;
repeat
    SETZE x^t = r(d(x))
    if akzeptiert (x^t, x) then
        SETZE x = x^t
    if c(x^t) < c(x^b) then
        SETZE x^b = x^t
until Abbruchkriterium erfüllt ist
return x^b
// Output: Beste gefundene Lösung
```

Entwurf von Destroy-Operatoren

- Zufall: verschiedene Teile der Lösung zerstören, Diversifikation

Entwurf von Destroy-Operatoren

- Zufall: verschiedene Teile der Lösung zerstören, Diversifikation
- Zerstörungsgrad
  - zu gering → keine Exploration des Lösungsraums
  - zu hoch → zeitaufwendig und führt möglicherweise zu schlechten Ergebnissen (insbesondere bei heuristischem Repair)
  - Trade-off zwischen Flexibilität/Diversifikation und Zeitaufwand
  - wird entweder nach jeder Iteration erhöht oder zufällig (aus gegebenem Intervall) gewählt

Entwurf von Repair-Operatoren

- Wahl zwischen optimierender oder heuristischer Methode
- Optimierende Methode ist zeitaufwendiger, führt aber in der Regel in wenigen Iterationen zu besseren Lösungen
- Heuristische Methode ist schneller und hat stärkere Diversifikationseffekte

Arten von Destroy-Operatoren

- Random Destroy: Entferne zufällig ausgewählte Elemente → Diversifikation

Arten von Destroy-Operatoren

- Random Destroy: Entferne zufällig ausgewählte Elemente → Diversifikation
- Worst oder Critical Destroy: Entferne Elemente, die hohe Kosten aufweisen oder die Lösungsstruktur beeinträchtigen → Intensivierung

Arten von Destroy-Operatoren

- Random Destroy: Entferne zufällig ausgewählte Elemente → Diversifikation
- Worst oder Critical Destroy: Entferne Elemente, die hohe Kosten aufweisen oder die Lösungsstruktur beeinträchtigen → Intensivierung
- Related Destroy: Entferne Elemente, die leicht auszutauschen sind (z. B. Vehicle Routing Problem: geographisch nah beieinander gelegene Kunden mit ähnlicher Nachfrage)

Arten von Destroy-Operatoren

- Random Destroy: Entferne zufällig ausgewählte Elemente → Diversifikation
- Worst oder Critical Destroy: Entferne Elemente, die hohe Kosten aufweisen oder die Lösungsstruktur beeinträchtigen → Intensivierung
- Related Destroy: Entferne Elemente, die leicht auszutauschen sind (z. B. Vehicle Routing Problem: geographisch nah beieinander gelegene Kunden mit ähnlicher Nachfrage)
  - Shaw Removal: Ähnlichkeitsmaß, das verschiedene Kriterien gewichtet

Arten von Destroy-Operatoren

- Random Destroy: Entferne zufällig ausgewählte Elemente → Diversifikation
- Worst oder Critical Destroy: Entferne Elemente, die hohe Kosten aufweisen oder die Lösungsstruktur beeinträchtigen → Intensivierung
- Related Destroy: Entferne Elemente, die leicht auszutauschen sind (z. B. Vehicle Routing Problem: geographisch nah beieinander gelegene Kunden mit ähnlicher Nachfrage)
  - Shaw Removal: Ähnlichkeitsmaß, das verschiedene Kriterien gewichtet
  - Cluster Removal: Verhindert Wiedereinfügen im selben Cluster

Arten von Destroy-Operatoren

- Random Destroy: Entferne zufällig ausgewählte Elemente → Diversifikation
- Worst oder Critical Destroy: Entferne Elemente, die hohe Kosten aufweisen oder die Lösungsstruktur beeinträchtigen → Intensivierung
- Related Destroy: Entferne Elemente, die leicht auszutauschen sind (z. B. Vehicle Routing Problem: geographisch nah beieinander gelegene Kunden mit ähnlicher Nachfrage)
  - Shaw Removal: Ähnlichkeitsmaß, das verschiedene Kriterien gewichtet
  - Cluster Removal: Verhindert Wiedereinfügen im selben Cluster
- Small Removal: Entferne Variablen mit geringen Koeffizienten in den Ressourcenrestriktionen (z. B. Kunden mit geringen Nachfragen) → leicht auszutauschen

Arten von Destroy-Operatoren

- Random Destroy: Entferne zufällig ausgewählte Elemente → Diversifikation
- Worst oder Critical Destroy: Entferne Elemente, die hohe Kosten aufweisen oder die Lösungsstruktur beeinträchtigen → Intensivierung
- Related Destroy: Entferne Elemente, die leicht auszutauschen sind (z. B. Vehicle Routing Problem: geographisch nah beieinander gelegene Kunden mit ähnlicher Nachfrage)
  - Shaw Removal: Ähnlichkeitsmaß, das verschiedene Kriterien gewichtet
  - Cluster Removal: Verhindert Wiedereinfügen im selben Cluster
- Small Removal: Entferne Variablen mit geringen Koeffizienten in den Ressourcenrestriktionen (z. B. Kunden mit geringen Nachfragen) → leicht auszutauschen
- History-Based Destroy: Entferne Elemente, die in der Vergangenheit Teil schlechter Lösungen waren

Arten von Repair-Operatoren

- Greedy-Heuristiken: iterativ bestmögliche Wahl, Problem: Wahlmöglichkeiten in späten Iterationen

Arten von Repair-Operatoren

- Greedy-Heuristiken: iterativ bestmögliche Wahl, Problem: Wahlmöglichkeiten in späten Iterationen
- Regret-Heuristiken: maximale Differenz zwischen bester Einfügemöglichkeit und k-bester

Arten von Repair-Operatoren

- Greedy-Heuristiken: iterativ bestmögliche Wahl, Problem: Wahlmöglichkeiten in späten Iterationen
- Regret-Heuristiken: maximale Differenz zwischen bester Einfügemöglichkeit und k-bester
- Relaxierte exakte Verfahren: Beschleunigung

Arten von Repair-Operatoren

- Greedy-Heuristiken: iterativ bestmögliche Wahl, Problem: Wahlmöglichkeiten in späten Iterationen
- Regret-Heuristiken: maximale Differenz zwischen bester Einfügemöglichkeit und k-bester
- Relaxierte exakte Verfahren: Beschleunigung
- Exakte Verfahren: schnell gute Lösungen, aber keine Diversifikation

Arten von Repair-Operatoren

- Greedy-Heuristiken: iterativ bestmögliche Wahl, Problem: Wahlmöglichkeiten in späten Iterationen
- Regret-Heuristiken: maximale Differenz zwischen bester Einfügemöglichkeit und k-bester
- Relaxierte exakte Verfahren: Beschleunigung
- Exakte Verfahren: schnell gute Lösungen, aber keine Diversifikation
- Lokale Suche

Agenda

5 Metaheuristiken – Large Neighborhood Search

- Large Neighborhood Search
- Adaptive Large Neighborhood Search

Allgemeines Prinzip der Adaptive Large Neighborhood Search

- Die Adaptive Large Neighborhood Search (ALNS) wurde erstmalig von Ropke und Pisinger (2005) vorgestellt
- ALNS erlaubt zur Intensivierung und Diversifizierung der Suche die Anwendung mehrerer Destroy- und Repair-Operatoren
- Operatoren werden Gewichte zugeordnet, die die Wahrscheinlichkeit bestimmen, mit der ein Operator genutzt wird
- Dynamische Gewichtsanpassung der Operatoren in jeder Iteration (oder nach einer festgelegten Anzahl an Iterationen) auf Basis der Performance in vorherigen Iterationen
- Unterschied zu LNS: ALNS kann Destroy- und Repair-Operatoren verwenden, die nur diversifizieren oder intensivieren und nur in wenigen Fällen geeignet sind

Allgemeines Prinzip der Adaptiven Large Neighborhood Search

Algorithmus 2 : Adaptive Large Neighborhood Search

```
// Input: Zulässige Lösung x
```
SETZE $x^b = x$; $\rho^- = (1, ..., 1)$; $\rho^+ = (1, ..., 1)$;
repeat
  WÄHLE Destroy- und Repair-Operatoren $d \in \Omega^-$ und $r \in \Omega^+$ unter Verwendung von ρ^- und ρ^+
  SETZE $x^t = r(d(x))$
  if *akzeptiert* (x^t, x) **then**
    SETZE $x = x^t$
  if $c(x^t) < c(x^b)$ **then**
    SETZE $x^b = x^t$
  AKTUALISIERE ρ^- und ρ^+
until *Abbruchkriterium erfüllt ist*
return x^b
```
// Output: Beste gefundene Lösung
```

Stärken der ALNS

- Bekannte Konstruktionsheuristiken oder exakte Verfahren → einfache Implementierung von Repair-Operatoren → ALNS als Vergleichsmethode
- Strukturiertes Durchsuchen eines großen Teils des Lösungsraums durch eine Reihe großer Nachbarschaften (seltenes Verfangen in lokalen Optima)
- Robuste Suche aufgrund des adaptiven Mechanismus
- Keine „entweder...oder“-Entscheidungen bei der Wahl der Nachbarschaften sondern „sowohl...als auch“-Entscheidungen
- Sehr gute Ergebnisse bei verschiedenen Transport- und Schedulingproblemen