Methoden und Anwendungen der Optimierung (MAO)

Kapitel 5: Metaheuristiken

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
- Einführung
- Iterierte Lokale Suche (ILS)
- Variable Neighborhood Descent/Search (VND/VNS)

Metaheuristiken

Ziele des Kapitels:

- wissen, was Metaheuristiken sind und wie man diese klassifizieren kann
- verstehen, wie man aus Lokaler Suche die Metaheuristiken ILS, VND und VNS aufbauen kann

Agenda

5 Metaheuristiken
- Einführung
- Iterierte Lokale Suche (ILS)
- Variable Neighborhood Descent/Search (VND/VNS)

Praktische Anforderungen an Planungsalgorithmen

- Viele Planungsprobleme in der Logistik und Produktion können als kombinatorische Optimierungsprobleme modelliert werden. Viele dieser Probleme sind NP-schwer.
  - viele Nebenbedingungen, evtl. mehrere Ziele
  - enorme Problemgrößen
  - strikte Antwortzeiten, z. B. bei dynamischer Planung

$\Rightarrow$ Metaheuristiken

(Meta)-heuristiken (Wiederholung)

Anwendungsbereich:

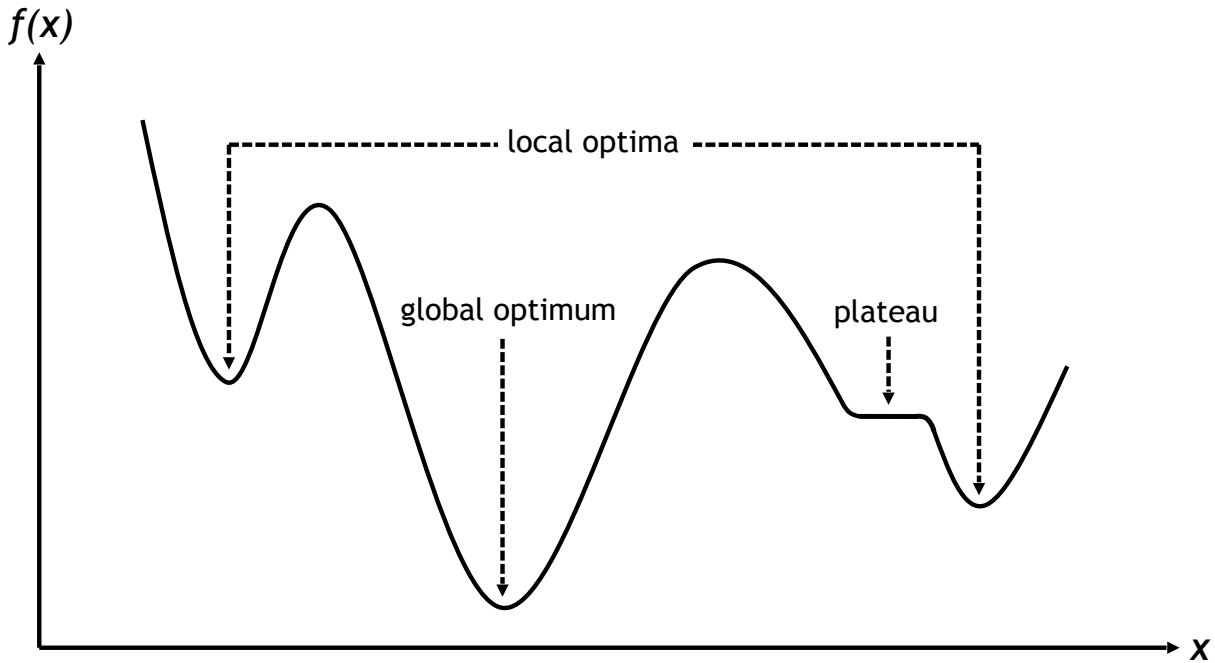
- Der Aufwand zur Problemlösung steigt sehr schnell an $\rightarrow$ exakte Verfahren nicht anwendbar
- Anforderungen an die Antwortzeit

Eigenschaften:

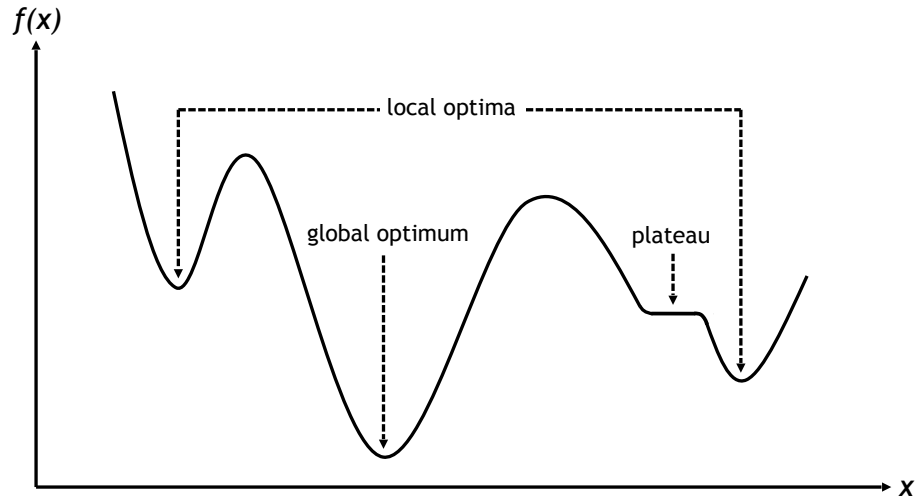
- keine Optimalitätsgarantie
- keine Aussage über Abweichung von optimaler Lösung (Approximationsalgorithmen)
- können *akzeptable* Lösungsgüte bei *vernünftigen* Laufzeiten bieten

Metaheuristiken als intelligente lokale Suche

- Einige Metaheuristiken zielen darauf ab, die Nachteile gieriger lokaler Suche zu überwinden
  - lokale Optima
  - Plateaus



Gegenmaßnahmen



- Gegenmaßnahmen?
  - Multi-Start Methoden
  - Gedächtnis (Tabusuche)
  - Randomisierung (Simulated Annealing)
  - Perturbation (Iterierte Lokale Suche, Variable Nachbarschaftssuche)

Definition Metaheuristik I

- A. Løkketangen/G. Laporte: a metaheuristic contains mechanisms that allow the search to get out of local optima.

  F. Glover, G. Kochenberger: Metaheuristics are „solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process capable of escaping from local optima and performing a robust search of a solution space.“

Definition Metaheuristik II

- Weitere Metaheuristiken imitieren Suchstrategien aus der Biologie und der Natur, z. B. Genetische Algorithmen, Ameisen-Algorithmen, ...
- E. Talbi: Metaheuristic search methods can be defined as upper level general methodologies (templates) that can be used as guiding strategies in designing underlying heuristics to solve specific optimization problems.
- I.H. Osman: A metaheuristic is formally defined as an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions.

Definition Metaheuristik III

P. Luke: Metaheuristics is a rather unfortunate[1] term often used to describe a major subfield, indeed the primary subfield, of stochastic optimization. Stochastic optimization is the general class of algorithms and techniques which employ some degree of randomness to find optimal (or as optimal as possible) solutions to hard problems. Metaheuristics are the most general of these kinds of algorithms, and are applied to a very wide range of problems.

1: Ordinarily I'd call the subfield stochastic optimization. But that's too general a term; it includes important algorithms like Markov Chain Monte Carlo (MCMC) or Gibbs Sampling, which are not in this category. Metaheuristics has lately been the term of use, but I think it's profoundly misleading and weird. When I hear metadiscussion I think: a discussion about discussions. Likewise when I hear metaheuristic I think: a heuristic about (or for) heuristics. That's not at all what these algorithms are about! Perhaps the lesser-used term black box optimization would be better, though it too comes with some additional baggage. Weak methods is also too broad a term: it doesn't imply stochasticity. Sometimes the term stochastic search is used: but I usually define search problems as all-or-nothing: either you find the solution or you don't. We're not doing search; we're doing optimization.

Defintion Metaheuristik IV

Metaheuristik (Definition)

T. Grünert, S. Irnich: Eine Metaheuristik ist ein übergeordneter Algorithmus, der die Lösungssuche eines oder mehrerer abhängiger Algorithmen steuert. Sie beruht auf einer Sammlung von (Meta-) Strategien, die unabhängig vom zugrunde liegenden Problem und den abhängig gesteuerten Algorithmen sind.

- Metaheuristiken sind Strategien, die den Suchprozess steuern. Diese kann man *unabhängig vom zugrunde liegenden Problem* in vielen Heuristiken anwenden.
- Durch die *Konkretisierung der Strategien* gelangt man zu einer Heuristik für ein spezifisches Optimierungsproblem.
- Insbesondere kann es eine große Anzahl von Heuristiken für ein Problem geben, die von derselben Metaheuristik abgeleitet wurden.

Definition Metaheuristik V

In diesem Sinne könnte man auch *Greedy-Algorithmen* mit

- Festlegung von abhängigen und unabhängigen Variablen
- Festlegung einer (Hilfs-)Bewertungsfunktion

und die *Lokale Suche* mit

- Festlegung einer Nachbarschaft
- Festlegung der Pivotisierungsregel
  (Ersten-/Besten-/k-Erstensuche)

als Metaheuristiken verstehen, auch wenn die Literatur dies in der Regel **nicht** tut.

Klassische Heuristiken vs. Metaheuristiken

Klassische Heuristiken:

- problemspezifisch und nutzen die Problemstruktur aus, d.h. sie sind entworfen für ein bestimmtes Problem und auch nur darauf anwendbar
- Entwurf qualitativ hochwertiger Heuristiken ist anspruchsvoll

Metaheuristiken:

- anwendbar auf eine große Auswahl an Optimierungsproblemen
- Anpassung, um ein bestimmtes Problem zu lösen, erfordert gewöhnlich deutlich weniger Aufwand als die Entwicklung einer spezialisierten Heuristik von Grund
- Aber: gute problemspezifische Heuristiken übertreffen leistungsmäßig oft Standard-Metaheuristiken $\Rightarrow$ qualitativ hochwertige Metaheuristiken beinhalten meist problemspezifisches Wissen

Geschichte der Metaheuristiken

Zur Historie: Der Begriff der „Metaheuristik“ wurde 1986 von Glover (1986) eingeführt, obwohl bereits in den sechziger Jahren Algorithmen entwickelt wurden, die man heute als Metaheuristiken bezeichnen würde.

- Meist dem Bereich Optimierung in der Informatik oder Mathematik zugeordnet. Von Bedeutung in zahlreichen Forschungs-„Communities“: KI insbesondere Computational Intelligence, Soft Computing und Operations Research ...
- Anwendungsfelder: Logistik und Produktion, Bioinformatik, Ingenieurswesen, Data Mining, Finanzwesen

Klassifikation von Metaheuristiken I

Klassifikation von Heuristiken und Metaheuristiken:

- *Startlösung und Stoppregel:*
  - **Konstruktions- oder Eröffnungsverfahren:** Startet ohne Lösung und stoppt nach Konstruktion einer Lösung (ant colony optimization, GRASP)
  - **Verbesserungsverfahren:** Startet mit einer vorgegebenen Lösung und versucht, diese zu verbessern. Stoppt, wenn keine lokale Verbesserung mehr möglich oder mittels künstlicher Stoppregel.
  - **Zusammengesetzte Verfahren:** Konstruktion und Verbesserung
- *Anzahl der Komponenten:* einfach vs. hybrid
- *Nachbarschaft:* ohne Nachbarschaftsbegriff, zufälliges Auswählen (=Sampling) oder explizites Durchsuchen von Nachbarschaften

Klassifikation von Metaheuristiken II

- *Zufall:* deterministisch vs. zufallsgesteuert
- *Gedächtnis:* mit vs. ohne Gedächtnis
- *natur-analog:* der Natur nachempfunden vs. künstlich
- *Anzahl Lösungen:* (gleichzeitige) Nutzung nur einer Lösung vs. Population von Lösungen (single-solution vs. population-based)

Arten von Metaheuristiken

Verbreitete Metaheuristiken sind:

- Iterierte Lokale Suche (iterated local search, ILS)
- Variable/r Nachbarschaftsabstieg/-suche
  (variable neighborhood descent/search, VND, VNS)
- Simulated Annealing (SA)
- Tabusuche (TS)
- Ruin-and-Recreate/Large Neighborhood Search (LNS)
- Lagrange-Heuristiken
- Genetische Algorithmen (GA) und Evolutionäre Strategien (ES):
  Evolutionary Computation (EC)
- Ameisenalgorithmen (ant systems, ant colony optimization)
- Very Large-Scale Neighborhood Search (VLSNS)
- Particle Swarm Optimization (PSO)
-

Wichtig!

- Erfolgreiche Metaheuristiken alternieren zwischen Intensivierung (Konzentration der Suche auf vielversprechende Bereiche des Suchraums) und Diversifikation (Erforschung neuer Bereiche im Suchraum).
- Es gibt nicht eine Metaheuristik, die für alle Probleme am besten geeignet ist.
- Der intelligente Einsatz von effizienten Datenstrukturen und Subalgorithmen ist von größter Bedeutung bei der Implementierung hochwertiger Metaheuristiken.

Charakteristika guter Metaheuristiken I

- Geschwindigkeit
  - hängt von der Planungsebene ab: strategisch vs. taktisch vs. operativ
  - Echtzeitumgebungen, dynamische Planung (neue Aufträge, neue Notfälle)
  - Interaktion

Charakteristika guter Metaheuristiken II

- Genauigkeit:
  - Abweichung von Zielfunktionswert der Optimallösung
  - Konsistenz: eine Heuristik, die in allen Fällen gut abschneidet, wird meist einer Heuristik vorgezogen, die in sehr vielen Fällen überlegen ist, aber in einigen Fällen sehr schlecht abschneidet. Ernsthaftes Problem: die zurückgegebenen Lösungen können durch genaues Hinsehen verbessert werden.
  - Gute Lösungen nach kurzer Zeit und eine stetige Verbesserung während des Laufs werden häufig einer einzigen finalen Antwort am Ende der (potentiell langen) Laufzeit vorgezogen.

Charakteristika guter Metaheuristiken III

- Einfachheit:
  - verständlich in Bezug auf die wesentlichen Ideen, in ausreichendem Detail beschrieben
  - robust, auch wenn nicht alle Details umgesetzt werden
  - vernünftige Anzahl von Parametern mit nachvollziehbarer Bezeichnung und Aussage. Anzahl kann durch reaktive Mechanismen oder Annahme eines festen Wertes reduziert werden

Charakteristika guter Metaheuristiken IV

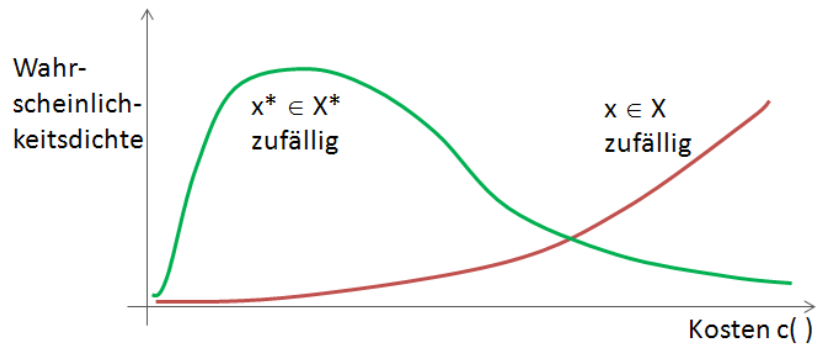
- Flexibilität der Metaheuristik nach Anpassung auf ein spezifisches Problem
  - Anpassung auf Probleme mit zusätzlichen Nebenbedingungen ohne starken Verlust an Lösungsqualität möglich.
  - Beispiel: Zielfunktion mit Strafkosten ermöglicht die Anwendung von einfachen Operatoren, die unzulässige Lösungen erzeugen.

Agenda

5 Metaheuristiken
- Einführung
- Iterierte Lokale Suche (ILS)
- Variable Neighborhood Descent/Search (VND/VNS)

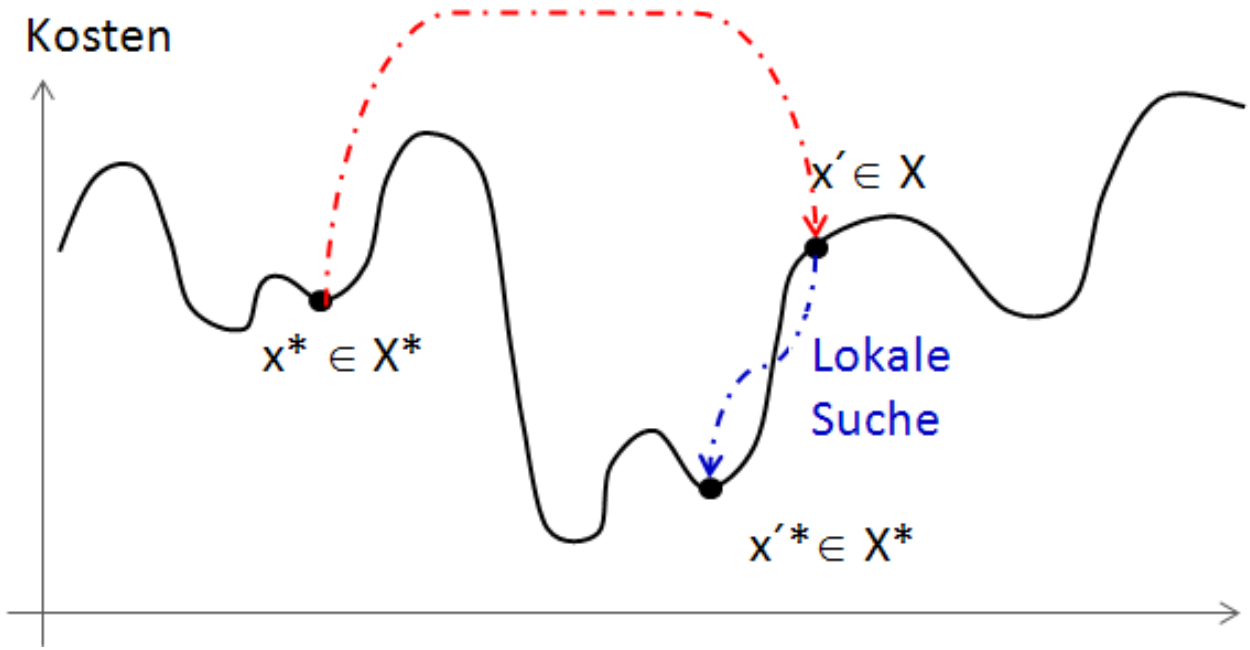
Grundidee Iterierte Lokale Suche (ILS) I

- Gegeben sei eine Instanz $P = (X, c)$ eines Minimierungsproblems Π und $\mathcal{N}$ Nachbarschaft. Die Menge $X^* \subset X$ sei die Menge der lokalen Minima.
- Idee von ILS: Suche auf der Menge der lokalen Minima X^*
- Begründung: Zufällig gewählte $x^* \in X^*$ sind im Mittel besser als zufällige $x \in X$



- Suchverlauf über lokale Optima:
  - Nachbarschaft im Raum lokaler Optima?
  - Realisierung durch Perturbation (Verwirbelung) der Lösung und Anwendung von lokaler Suche auf die verwirbelte Lösung

Grundidee Iterierte Lokale Suche (ILS) II



- x^* lokales Minimum
- $\rightsquigarrow x'$ im Allg. nicht lokal optimal
- $\rightsquigarrow x'^*$ lokales Minimum

Pseudocode Iterierte Lokale Suche (ILS) I

Algorithmus 1 : Iterierte Lokale Suche (ILS) für ein Minimierungsproblem

```
// Input: Startlösung x
SETZE x* := LokaleSuche(x)
repeat
    SETZE x' := Perturbation(x*, Historie)
    SETZE x'* := LokaleSuche(x')
    SETZE x* := AkzeptanzKriterium(x*, x'*, Historie)
until Terminierungsbedingung(Historie)
// Output: beste gefundene Lösung
```

Pseudocode Iterierte Lokale Suche (ILS) II

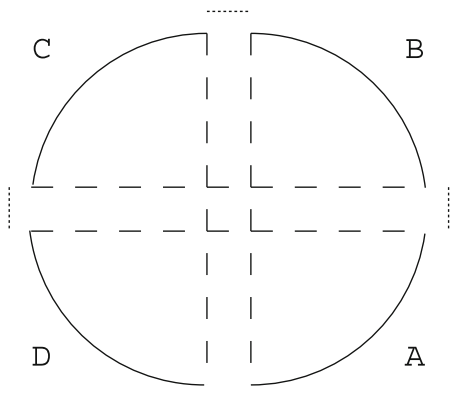
1. *LokaleSuche*(x) verbessert eine gegebene Lösung x mittels lokaler Suche und gibt ein entsprechendes lokales Minimum zurück
2. *Perturbation*(x, historie) verwirbelt eine Lösung (ggf. abhängig vom Verlauf der bisherigen Suche)
3. *AkzeptanzKriterium*(x, x', historie) akzeptiert – ggf. abhängig vom Verlauf der bisherigen Suche – eine neue Lösung x' oder gibt die (alte) Lösung x zurück

Beispiel: ILS für STSP I

Beispiel: Π =symmetrisches TSP.

Die heutzutage leistungsfähigsten Verbesserungsverfahren für das STSP beruhen auf dem ILS-Prinzip:

- *Lokale Suche* mittels *Lin-Kernighan-Suche* (einfacher zu implementieren, aber weniger effektiv sind 2-Opt und 3-Opt)
- *Perturbation* mittels *Double-Bridge*-Schritt: Gepunktete Kanten werden entfernt und durch gestrichelte Kanten ersetzt



Beispiel: ILS für STSP II

- Double-Bridge-Schritt:
  - Kanten zufällig gewählt (Gleichverteilung)
  - kann nicht direkt durch einen 2-Opt-Schritt rückgängig gemacht werden
  - empirisch als effektive Verwirbelung einer STSP-Tour identifiziert; effektiv unabhängig von der Größe der Instanz
- deterministische *Akzeptanz*: wähle *bessere der zwei Lösungen* x^* *und* x'^*

Weitere Informationen zu ILS:
`http://www.sls-book.net/Slides/sls-ils+vns.pdf`

Stärken ILS

- Gute Lösungsqualität mit einfacher Implementierung (zufällige Initiallösung, vorhandene lokale Suche, einfache zufällige Perturbation, Akzeptanz nach Lösungsgüte) $\rightarrow$ im Allgemeinen klar besser als zufällige Neustarts der lokalen Suche
- konzeptuell einfach und modularer Aufbau: vier Komponenten, deren Komplexität unabhängig voneinander schrittweise erhöht werden kann
- Geschwindigkeit: empirisch sind k lokale Suchen innerhalb ILS deutlich schneller als k lokale Suchen nach zufälligen Neustarts
- sehr gute Lösungsqualität für einige Optimierungsprobleme in den Bereichen Tourenplanung und Machine Scheduling

Design ILS I

- Initiallösung:
  - von Bedeutung, wenn schnell gute Lösungsqualität erwünscht
  - Einfluss nimmt mit Länge des Suchverlaufs ab $\rightarrow$ einfache Implementierung verwenden
- Perturbation: „A good perturbation transforms one excellent solution into an excellent starting point for a local search“
  - zu schwach: schnelle Rückkehr zu x^*, wenig Diversifikation
  - zu stark: entspricht einem zufälligen Neustart
  - Abstimmung Perturbation und lokale Suche (Güte, Geschwindigkeit)
  - sollte stochastisch sein, kann Gedächtnis zur Vermeidung von Zyklen benutzen
  - Stärke von Perturbation manchmal unabhängig von Instanzgröße (TSP), aber oft zunehmende Stärke notwendig: abhängig von Suchhistorie, vordefiniertes Schema (ähnlich VNS)

Design ILS II

- Akzeptanzkriterium:
  - Steuerung von Diversifikation und Intensivierung
  - oft basierend auf Simulated Annealing
- Lokale Suche:
  - gründlichere lokale Suche liefert meist bessere Ergebnisse: Tradeoff mit Laufzeit berücksichtigen
  - kurze Tabusuche-Läufe ebenfalls möglich

Design ILS III

- Zusammenfassung (Daumenregeln):
  - besser zahlreiche kleine Perturbationen akzeptieren als eine sehr große
  - Initiallösung weitgehend irrelevant
  - lokale Suche darf Perturbation nicht leicht rückgängig machen
  - Stärke von Perturbation abhängig davon wie stark sich die besten Lösungen in X^* häufen, d. h. wieviel gleiche Komponenten sie besitzen

Agenda

5 Metaheuristiken
- Einführung
- Iterierte Lokale Suche (ILS)
- Variable Neighborhood Descent/Search (VND/VNS)

Variable Neighborhood Descent (VND) I

Variable Neighborhood Descent (VND)= „variabler Nachbarschaftsabstieg“:
Deterministische Lokale Suche mit einer Folge von Nachbarschaften $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$. Es werden nur verbessernde Schritte durchgeführt und die Suche endet in einem lokalen Optimum bzgl. aller Nachbarschaften $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$.

Variable Neighborhood Descent (VND) II

Nachfolgend ist immer angenommen, dass $\mathcal{N}, \mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_{k_{max}}$ verschiedene Nachbarschaften für ein gegebenes kombinatorisches Optimierungsproblem Π sind.

Beispiel:

- Π=TSP mit Nachbarschaften 2-Opt, Relocation, Or-Opt, 3-Opt, Lin-Kernighan etc.

Merke: VND

VND bestimmt ein lokales Optimum bzgl. aller Nachbarschaften

Pseudocode VND I

Algorithmus 2 : VND (für ein Minimierungsproblem)

```
// Input: Zulässige Lösung x^0
SETZE t := 0 (Iterationszähler) und k := 1 (Nachbarschaftszähler)
repeat
    repeat
        Durchsuche Nachbarschaft N_k(x^t) nach (verbesserndem) Nachbarn
          x' ∈ N_k(x^t) mit c(x') < c(x^t)
        if verbessernder Nachbar x' gefunden then
            SETZE x^{t+1} := x'
            SETZE t := t + 1 und k := 1
    until keine Verbesserung mehr gefunden
    SETZE k := k + 1
until k > k_max
// Output: lokales Optimum x^t bzgl. aller Nachbarschaften
```

Pseudocode VND II

Bemerkungen:

- Im Spezialfall $k_{max} = 1$ erhält man die Lokale Suche mit der Nachbarschaft $\mathcal{N}_1$.
- Für Maximierungsprobleme ist jeweils „<“ durch „>“ zu ersetzen.
- Wie bei der Lokalen Suche kann auch hier die Suche nach einem verbessernden Nachbarn mit unterschiedlichen Strategien realisiert werden:
  - Bestensuche
  - Erstensuche
  - p-Erstensuche für $p > 1$

Pseudocode VND III

- Die Reihenfolge der Nachbarschaften sollte so gewählt werden, dass das Durchsuchen von $\mathcal{N}_k$ mit steigendem Index k aufwendiger wird. Beispielsweise durchsucht man beim TSP zuerst die 2-Opt-Nachbarschaft bis zum Erreichen eines lokalen Minimums und sucht dann für solche Minima nach verbessernden 3-Opt-Schritten usw.

Variable Neighborhood Search (VNS) I

Variable Neighborhood Search (VNS)= „variable Nachbarschaftssuche“:
Deterministische Wahl einer Nachbarschaft $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_k$ und zufällige Auswahl einer Nachbarlösung aus dieser gewählten Nachbarschaft. Nach einer Lokalen Suche mit einer unabhängigen Nachbarschaft $\mathcal{N}$ wird die Lösung

- entweder akzeptiert und damit zur aktuellen Lösung
- oder verworfen und der Prozess mit der aktuellen Lösung fortgeführt.

VNS II

VNS ist eine Metaheuristik, die

- das Verlassen lokaler Optima erlaubt
- deterministisch zwischen verschiedenen Nachbarschaften $\mathcal{N}_1, \ldots, \mathcal{N}_{k_{max}}$ wechselt
- zufallsgesteuert durch sog. „Schütteln“ (shaking) eine i.d.R. schlechtere Nachbarlösung als die aktuelle Lösung erzeugt; meistens wird diese Nachbarlösung mit einer anderen Nachbarschaft $\mathcal{N}$ lokal optimiert
- ein deterministisches Zielfunktion-basiertes Akzeptanzkriterium zur Annahme oder Ablehnung der Nachbarlösung benutzt
- eine künstliche Stoppregel benötigt

Pseudocode VNS I

Algorithmus 3 : VNS (für ein Minimierungsproblem)

`// Input: Zulässige Lösung` x^0
SETZE $t := 0$ (Iterationszähler) und $k := 1$ (Nachbarschaftszähler)
repeat
  (Schütteln) Wähle zufällig eine Nachbarlösung $x' \in \mathcal{N}_k(x^t)$
  (Lokale Suche, optional) $x'' =$ LokaleSuche(x') mittels Nachbarschaft $\mathcal{N}$
  (Akzeptanzkriterium)
  if $c(x'') < c(x^t)$ **then**
    SETZE $x^{t+1} = x''$, $t := t + 1$ und $k := 1$
  else
    SETZE $k := \begin{cases} k+1 & k < k_{max} \\ 1 & k = k_{max} \end{cases}$
until *Stoppkriterium erfüllt*
`// Output: beste gefundene Lösung` x^t

Pseudocode VNS II

Bemerkungen:

- Für das Schütteln werden die Nachbarschaften zyklisch in der Folge $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_{k_{max}}, \mathcal{N}_1, \mathcal{N}_2, \ldots$ durchlaufen.
- Nur wenn eine bessere Lösung x'' gefunden wird, startet man wieder mit $\mathcal{N}_1$.
- Die Auswahl eines $x' \in \mathcal{N}_k(x^t)$ sollte zufällig aufgrund einer Gleichverteilung auf allen Elementen der Nachbarschaft erfolgen.
- Typischerweise sind die Nachbarschaften $\mathcal{N}_1, \ldots, \mathcal{N}_{k_{max}}$ „groß“, während die Lokale Suche mit einer „kleineren“ Nachbarschaft $\mathcal{N}$ durchgeführt wird.

Geschichte der VNS/VND

Zur Historie:

- Die Begriffe „VND“ und „VNS“ stammen von Pierre Hansen und Nenad Mladenovic (Hansen und Mladenovic, 2001, 1997) und wurden erstmalig Mitte der 90er Jahre genannt.
- Die Ideen von VND und VNS wurden bereits deutlich früher in vielen Metaheuristiken für verschiedenste Probleme benutzt.

Zur Vertiefung...

- (Glover, 1986)
- (Aarts und Lenstra, 1997): Kapitel 4
- (Domschke und Drexl, 2007): Abschnitt 6.6.1
- (Grünert und Irnich, 2005): Abschnitt 3.8
- (Zimmermann, 2005): Kapitel 6
- (Hoos und Stützle, 2005): Abschnitt 2.3
- (Hansen und Mladenovic, 1997, 2001)

[Aarts und Lenstra 1997] Aarts, E. ; Lenstra, J.K.: *Local Search in Combinatorial Optimization*. Chichester : Wiley, 1997

[Domschke und Drexl 2007] Domschke, W. ; Drexl, A.: *Einführung in Operations Research*. 7. Auflage. Berlin : Springer, 2007

[Glover 1986] Glover, Fred: Future paths for integer programming and links to artificial intelligence. In: *Computers & Operations Research* 5 (1986), S. 533–549

[Grünert und Irnich 2005] Grünert, T. ; Irnich, S.: *Optimierung im Transport Band I: Grundlagen*. Aachen : Shaker Verlag, 2005

[Hansen und Mladenovic 2001] Hansen, P. ; Mladenovic, N.: Variable neighborhood search: Principles and applications. In: *European Journal of Operational Research* 130 (2001), Nr. 3, S. 449–467

[Hansen und Mladenovic 1997] Hansen, Pierre ; Mladenovic, Nenad: Variable Neighborhood Search. In: *Computers and Operations Research* 24 (1997), S. 1097–1100

[Hoos und Stützle 2005] Hoos, H.H. ; Stützle, T.: *Stochastic Local Search Foundations and Applications*. San Francisco, CA : Morgan Kaufmann Publishers, Elsevier, 2005. – ISBN 1558608729

[Zimmermann 2005] Zimmermann, H.J.: *Operations Research Methoden und Modelle*. Wiesbaden : Vieweg, 2005