Karlsruher Institut für Technologie

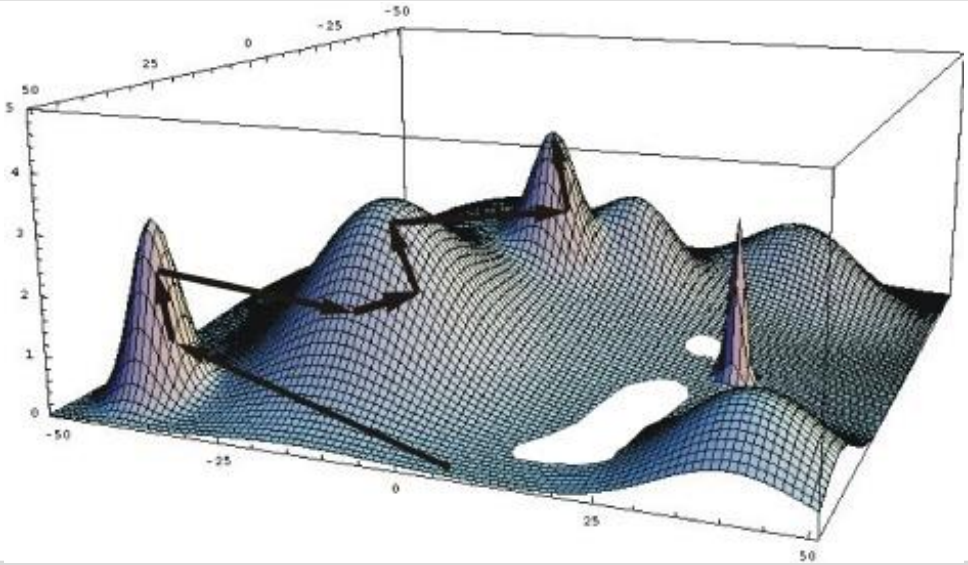
Vorlesung Computational Intelligence

Teil 4: Evolutionäre und Memetische Algorithmen

4.2 Ausgewählte Grundlagen der Optimierung

Ralf Mikut, Wilfried Jakob, Markus Reischl

Institut für Automation und angewandte Informatik (IAI) / Campus Nord



4.2 Ausgewählte Grundlagen der Optimierung

Übersicht:

- Formale Definitionen
- Komplexität
- Mehrzieloptimierung
- Einige etablierte mathematische Verfahren
- Einige etablierte Metaheuristiken
- No-free-Lunch-Theoreme

Grundlagen der Optimierung – Formale Definitionen

Formale Definition eines Optimierungsproblems

Gegeben: n-dimensionale auf beliebige Mengen definierte Funktion:

$$F: M \subseteq M_1 \times M_2 \times \ldots \times M_n \rightarrow \mathbb{R}, \; M \neq \emptyset$$

Gesucht ist ein $x_{opt} \in M$ x **heißt Vektor der *Entscheidungsvariablen***

für das gilt: $\forall x \in M: \left(F(x) \leq F\left(x_{opt}\right)\right) = F_{opt}$ ***globales Optimum***

Häufig sind die M_i die Mengen der reellen oder der ganzen Zahlen.

Man spricht von einem *lokalen Optimum* $F_{lopt} = F(x_{lopt})$**, wenn gilt:**

$$\exists \varepsilon > 0 \; \forall x \in M: \left\| x - x_{lopt} \right\| < \varepsilon \Rightarrow F(x_{lopt}) \geq F(x)$$

Wegen $max\{F(x)\} = -min\{-F(x)\}$ ist eine Minimumsuche immer in eine Maximumsuche überführbar.

Grundlagen der Optimierung – Formale Definitionen

Beschränkungen von M: $G_j(x) \geq 0$

Explizite Beschränkung: $G_j(x)$ hängt von einer Vektorkomponente x_i ab.
Obere/Untere Schranken von x_i

Implizite Beschränkung: $G_j(x)$ hängt von mehreren Vektorkomponenten ab.
Unzulässige Bereiche im oder
am Rand des Parameterraums

Reale Probleme sind in der Regel beschränkt.

Anzahl der Optima:

- **Unimodal:** Ein globales Optimum
- **Multimodal:** Mindestens ein lokales Optimum

Weitere Eigenschaften wie *Definitionslücken*, *Verrauschung*, fehlende *Stetigkeit*, *Differenzierbarkeit* oder *Konvexität*, usw. sind kein Hindernis für einen EA-Einsatz.

Grundlagen der Optimierung – Komplexität

Polynomialzeit

Zeitobergrenze, die als Polynomfunktion *n*-ten Grades der Größe *k* eines Problems angegeben werden kann, mit der eine *deterministische sequentielle* Rechenmaschine das Problem löst.

$$\sum_{i=1}^{n} a_i \cdot k^i, \quad a_n \neq 0$$

Vereinfachende O-Notation:

Notation	Bedeutung	Beispiele für Laufzeiten
$O(1)$	beschränkte Komplexität	indizierter Zugriff
$O(\log x)$	logarithmisches Wachstum	Binäre Suche im geordneten Feld der Größe x
$O(\sqrt{x})$	Wachstum gemäß der Wurzelfunktion	naiver Primzahlentest $(Teiler \leq \sqrt{x})$
$O(x)$	lineares Wachstum	lineare Suche im unsortierten Feld der Größe x
$O(x \log x)$	super-lineares Wachstum	fortgeschrittenere Sortieralgorithmen (Größe x)
$O(x^2)$	quadratisches Wachstum	z.B. Sortieralgorithmus *Bubblesort*
$O(2^x)$	exponentielles Wachstum	rekursive Berechnung der Fibonacci-Folge

Grundlagen der Optimierung – Komplexität

O-Notation

- dient der Beschreibung der Zeit- oder Speicherkomplexität eines Algorithmus (auf einer sequentiellen Maschine)
- ist eine obere Schranke
- Vernachlässigung von Faktoren oder Polynomgliedern geringerer Potenz

Die Vorgehensweise bei der O-Notation erlaubt auch die Bildung von Komplexitätsklassen für Algorithmen.

Die Klasse der P-Probleme:

Alle Probleme, die von einer deterministischen, sequentiellen Rechenmaschine (Turingmaschine) in Polynomialzeit lösbar sind.

Grundlagen der Optimierung – Komplexität

Die Klasse der NP-Probleme:

Alle Probleme, die von einer nichtdeterministischen Turingmaschine in Polynomialzeit lösbar sind.

Nichtdeterministische Turingmaschine (NTM):

- **theoretisches Maschinenmodell (Konzept der theoretischen Informatik)**
- **Der Folgezustand einer NTM lässt sich nicht aus dem aktuellen Zustand und den aktuellen Eingabezeichen ableiten.**
- **NTM wählen einen Nachfolgezustand aus mehreren möglichen aus:**
  → daher nichtdeterministisch
- **Annahme, dass die Auswahl „gut“ sei, auf jeden Fall besser als zufällig**

→ Randomisierte Algorithmen sind nicht identisch mit nichtdeterministischen!

Grundlagen der Optimierung – Komplexität

Es ist unbekannt, ob P = NP ?

Ob also die NP-Probleme nicht doch auf einer deterministischen Maschine in Polynomialzeit lösbar sind.

NP-vollständige Probleme:

- **Untermenge der NP-Probleme**
- **lassen sich vermutlich nicht effizient (d.h. in Polynomialzeit) lösen**

  →ab einer *problemabhängig kleinen* Datenmenge nicht in praktikabler Zeit lösbar

Einige Beispiele NP-vollständiger Probleme:

- **Travelling Salesman Problem (TSP), Problem des Handlungsreisenden**
  Ermittlung der kürzesten Städtetour
- **Schedulingprobleme, meist mit einschränkenden Nebenbedingungen**
- **Knapsack Problem (Rucksackproblem)**
  Auswahl von Objekten derart, dass der größte Nutzwert unter Einhaltung einer Gewichtsschranke erreicht wird

Grundlagen der Optimierung – Mehrzieloptimierung

Mehrzieloptimierung (multikriterielle Optimierung)

- **Bisher: eine zu optimierende Zielfunktion**
- **praktische Probleme haben i.d.R. mehrere zu optimierende Kriterien Beispiele:**
  - **große Nutzlast**
  - **große Geschwindigkeit**
  - **geringer Energieverbrauch**

  oder
  - **kurze Bearbeitungszeit**
  - **geringe Maschinenkosten**
  - **hohe Auslastung**

→ sich widersprechende Kriterien!

Grundlagen der Optimierung – Mehrzieloptimierung

Mehrzieloptimierung (multikriterielle Optimierung)

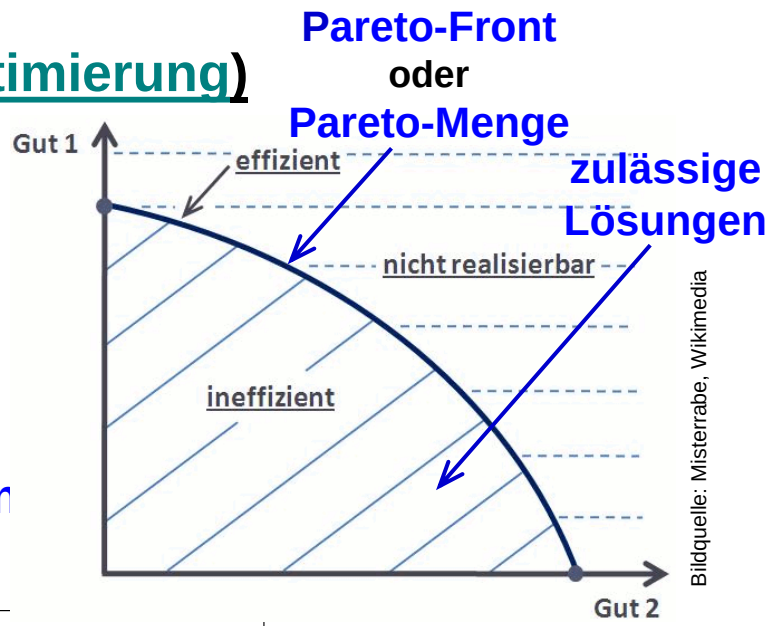
Pareto-Optimierung:

Bestimmung der Pareto-Menge:

Alle Lösungen, bei denen die Verbesserung eines Kriteriums nur auf Kosten eines anderen möglich ist.

Diese Lösungen liegen auf der Pareto-Fron

→ Menge aller optimalen Kompromisse



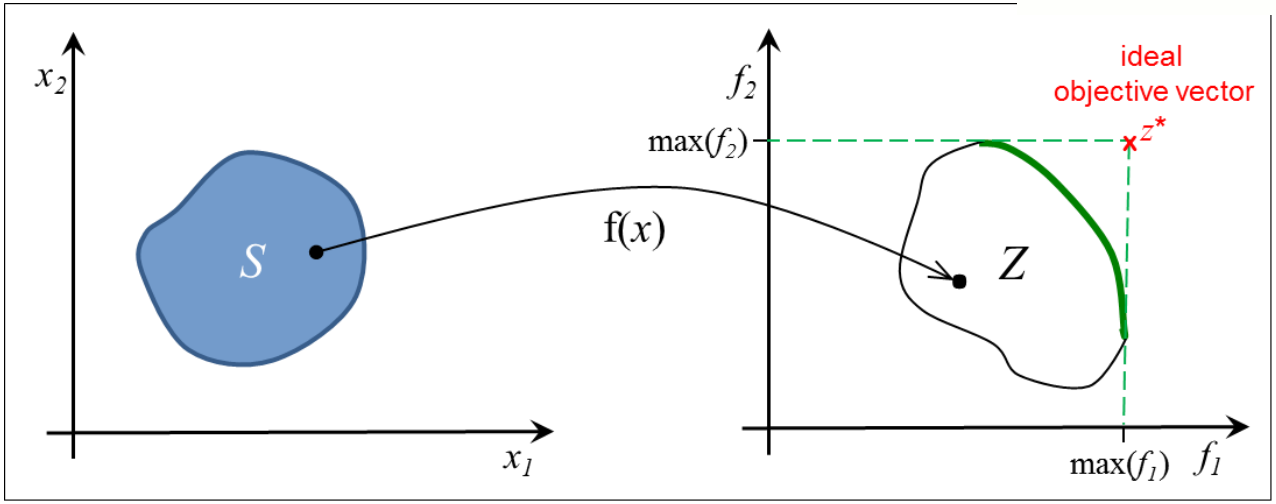


Abbildung der zulässigen Parametermenge S

durch die Fitnessfunktion

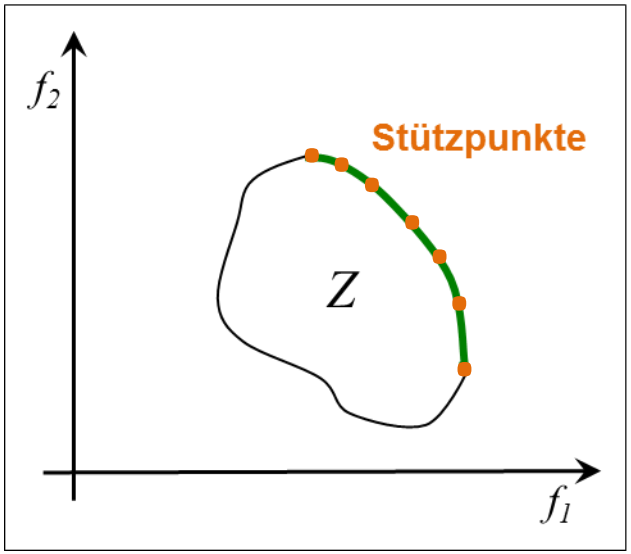
$f(x) = (f_1(x), ..., f_k(x))^T, \; x \in S$

in die zulässige Kriterienmenge Z

mit Paretofront

Mehrzieloptimierung (multikriterielle Optimierung)

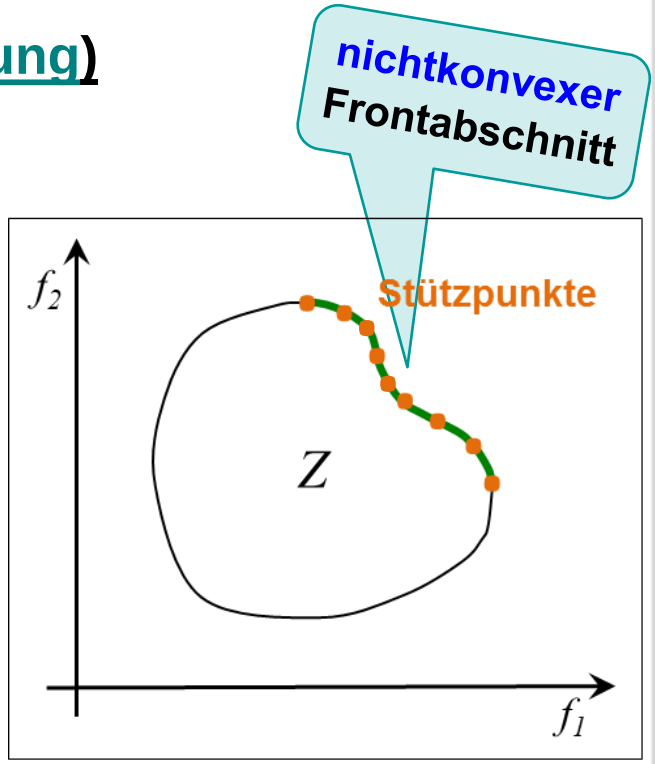
Aufwand zur Approximation der Pareto-Front:



Wie viel Stützpunkte werden mindestens benötigt?

7 Stützpunkte können bei günstiger Lage und halbwegs regelmäßiger konvexer Front zur Approximation genügen.

Mehr sind aber besser:



Aufwand zur Approximation der Pareto-Front:

Bei s Stützpunkten pro zusätzlichem Kriterium werden für eine Approximation der Pareto-Front von $k > 1$ Kriterien $s^{(k-1)}$ Stützpunkte (Pareto-optimale Lösungen) in der (Hyper-)Fläche benötigt, die weit genug auseinander liegen.

Wie nennt man dieses Wachstum?

Grundlagen der Optimierung – Mehrzieloptimierung

Mehrzieloptimierung (multikriterielle Optimierung)

Pareto-Optimierung:

Vorteile:

- Bestimmung vieler gleichwertiger Lösungsalternativen (Kompromisse)
- Abschätzung von erreichbaren Wertebereichen der Kriterien
- Reduktion einer *subjektiven Auswahl* auf *optimale Kompromisse*

Nachteile:

- keine direkte quantitative Vergleichbarkeit aller Lösungsalternativen
- Exponentielle Steigerung des Aufwands bei steigender Kriterienanzahl
- bei mehr als 4 Kriterien kaum noch darstellbar (→ Aggregation eines Teils der Kriterien)

Multi- und Many-Objective Evolutionary Algorithms (MOEAs):

- NSGA-II und NSGA-III (Non-dominated Sorting Genetic Algorithm) [Deb02, Deb14, Jain14]
  Komplexität der Suche: $O(m \cdot n^2)$ bei m Kriterien und einer Population der Größe n
- SPEA2 (Strength Pareto Evolutionary Algorithm) [Zit01]
- Bewertungsverfahren für Evolutionäre Algorithmen
  zur Bestimmung einer möglichst divergenten Pareto-Menge [Beu06]

Grundlagen der Optimierung – Mehrzieloptimierung

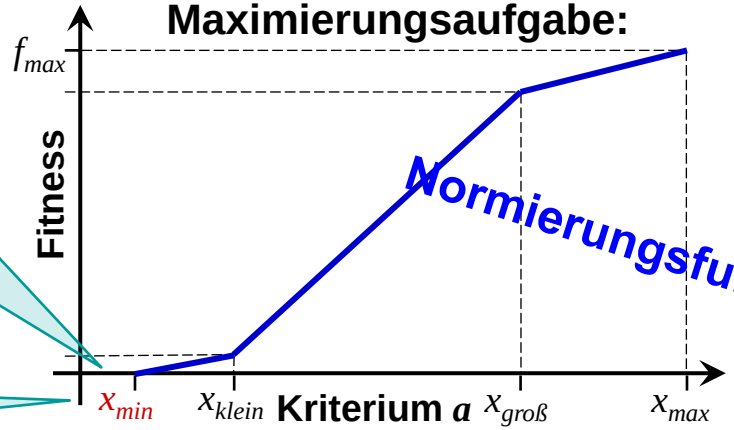
Mehrzieloptimierung (multikriterielle Optimierung)

Aggregierung der Kriterien, z.B. gewichtete Summe:

Wie soll das bei unterschiedlichen Dimensionen (z.B. Zeit und Kosten) und Skalen funktionieren?

Beispiele:

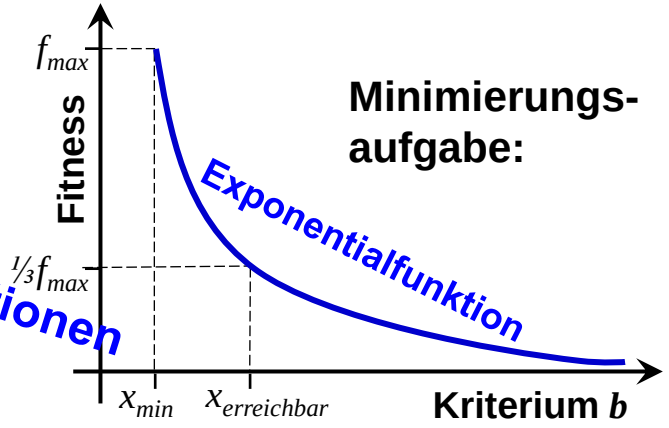
Maximierungsaufgabe:



Was tun bei Unterschreitung ?

Schätzwerte

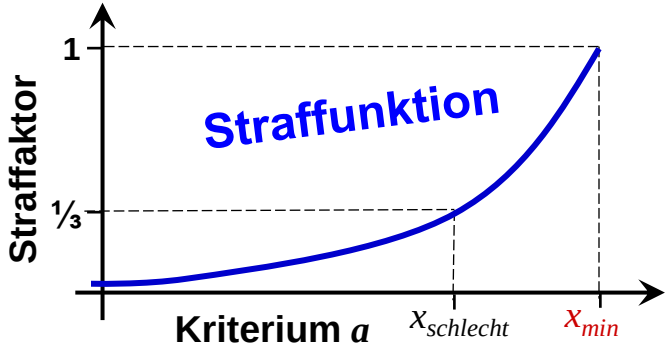
Minimierungsaufgabe:



Maximalwerte seien schlecht schätzbar

Behandlung der Verletzung von Restriktionen:

Straffaktor für die gewichtete Summe



Mehrzieloptimierung (multikriterielle Optimierung)

Aggregierung der Kriterien, z.B. gewichtete Summe:

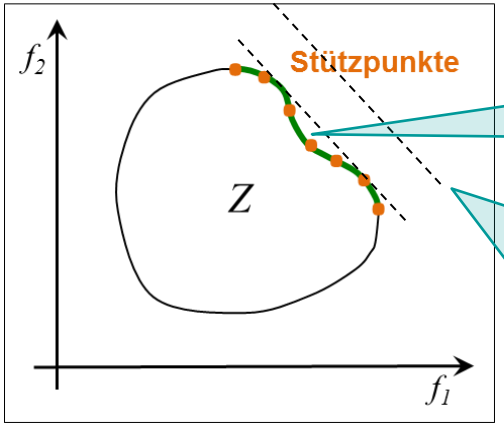
- Getrennte Normierung der Kriterien auf ein einheitliches Maß
- (subjektive) Gewichtung der Kriterien (eventuell ohne Vorwissen!)
- Bildung der Summe → ein Qualitätswert (Zielfunktion)
- Leichte Integration von Straffunktionen in Form von Straffaktoren $\in [0, 1]$
- Abbildung auf *Kosten* entspricht einer Form der gewichteten Summe!

Vorteile:

- quantitative Vergleichbarkeit aller Lösungsalternativen
- keine Begrenzung der Kriterienanzahl
- leichte Berechenbarkeit

Nachteil:

- Bei nichtkonvexen Lösungsmengen werden Lösungen unerreichbar!
- nur für Experten nachvollziehbar
- subjektive Gewichtung



Die Nichtkonvexen hingegen nicht!

Durch geeignete Gewichtung sind die konvexen Teile der Paretofront erreichbar.

Alternative: Kaskadierte gewichtete Summe [Jak14]

Einige etablierte mathematische Verfahren (lokale Suchverfahren, LSV)

Zwei Arten:

- **indirekte Verfahren benötigen neben dem Funktionswert Ableitungen**
- **direkte Verfahren arbeiten nur mit dem Funktionswert**

Einige Eigenschaften:

- **startpunktabhängig, daher meist mehrfache Anwendung erforderlich**
- **meist vergleichsweise schnelle Konvergenz**
- **in der Regel deterministisch**
- **Anwendungsgebiet: numerische Probleme**
- **Komplexität: $O(n^2)$ ... $O(n^3)$**

Warum?

Was bedeutet das?

Grundlagen der Optimierung – Mathematische Verfahren

Indirekte Verfahren:

- **Gradientenverfahren, konjugierte Gradientenverfahren**
  Ermitteln die Richtung des stärksten Anstiegs
- **Quasi-Newton-Verfahren**
  verwenden ein quadratisches Modell der Zielfunktion durch Taylor-Reihenansatz und deren Ableitung

  Fülle von Varianten, z.B. DFP-Verfahren mit und ohne Ableitungen

. . .

Einige Eigenschaften:

- **Beschränkungen werden nicht berücksichtigt**
- **konvergieren schnell**
- **Aufwand: nicht unter $O(n^3)$**

Direkte Verfahren:

- ableitungsfrei
- Suchrichtung und Schrittweite werden heuristisch bestimmt.
- keine Berücksichtigung von Beschränkungen
- Konvergenzprobleme möglich

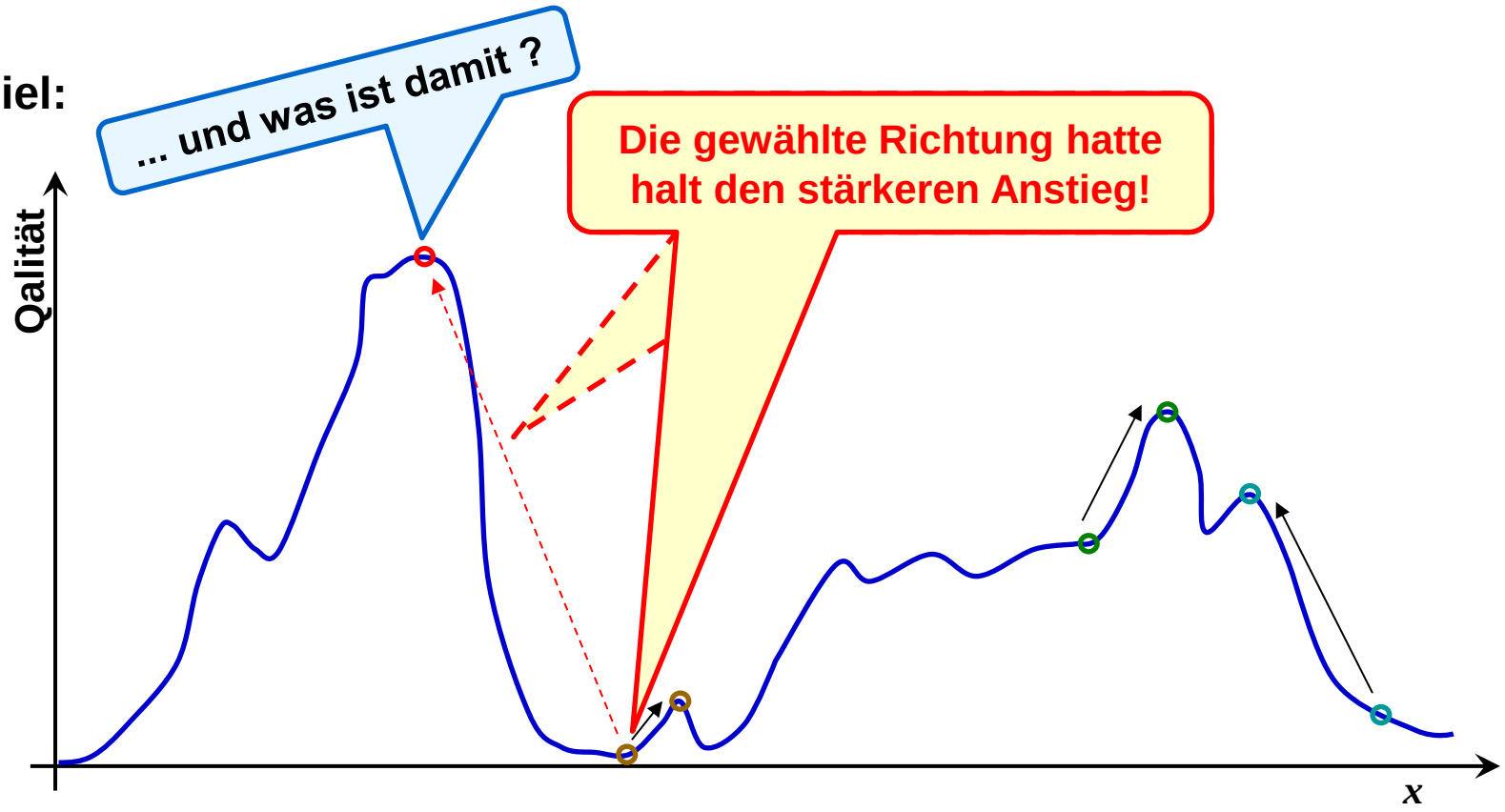
- **Gauß-Seidel-Verfahren**
  Iterative Suche mit fester Schrittweite entlang den Achsen
  bis Verbesserung ausbleibt.
  - Konvergenzgeschwindigkeit stark suchraumabhängig
  - Fülle von Varianten zur Verbesserung der Konvergenzgeschwindigkeit
- **Pattern-Strategien**
  Anpassung der Suchrichtung entsprechend der Zielfunktion
  - Verfahren von Hooke und Jeeves
  - Verfahren von Powell (konjugierte Richtungen)

. . .

Verhalten von lokalen Suchverfahren:

LSVs suchen in der Regel in die Richtung des stärksten Anstiegs.

Beispiel:



Zwei ausgewählte lokale Suchverfahren:

Rosenbrock-Verfahren

Suche entlang eines im Raum rotierenden Koordinatensystems, das in die Richtung zeigt, die den größten Fortschritt verspricht.

- Anpassung der Schrittweiten
- Berücksichtigung von Beschränkungen durch interne Straffunktion
- Benötigt gültigen Startpunkt

Gilt als robustes, ableitungsfreies lokales Suchverfahren. Aufwand: $O(n^3)$

[Ros60]

Complex-Verfahren von Box

Arbeiten mit mehreren Punkten (n+1), die einen Polyeder bilden.

- **Veränderung durch Kontraktion, Expansion und Reflexion des Polyeders**
  - Reflexion des schlechtesten Punktes am Flächenschwerpunkt
  - Bei Verbesserung Versuch einer Expansion
  - Bei Verschlechterung erfolgt eine Kontraktion
- **Abbruch bei zu großer Annäherung der Punkte an den Mittelpunkt** **[Box60]**

Complex-Verfahren von Box (2)

Weitere Eigenschaften:

- **mindestens ein Startpunkt, der Rest wird ausgewürfelt (stochastische Initialisierung)**
- **Abbruch, wenn fünf mal hintereinander keine Verbesserung eintritt.**
- **Berücksichtigung von Beschränkungen**
- **Bewegung eines Polyeders im Raum, wobei kleinere lokale Minima übersprungen werden können.**
- **Gilt bei wenigen Parametern (bis zu 10) als robust.**
  Aufwand: $O(n^2)$, Anzahl der Funktionsaufrufe bei $n \leq 10$: $O(n^{2.11})$

Einige etablierte Metaheuristiken

Meist allgemein anwendbare, global suchende und stochastische Verfahren

- **Ameisen-Algorithmen (Ant Colony Optimisation)**
  Idee: Kürzere Wege zu einem Futterplatz werden öfter von Ameisen frequentiert als längere.
  → höhere Pheromonkonzentration, Entstehung einer „Ameisenstraße“
  Sehr gut für kombinatorische Probleme geeignet
- **Partikelschwarm-Optimierung**
  Idee: Nachbildung des Schwarmverhaltens von Tieren zur Auffindung attraktiver Plätze
  Für kontinuierliche und diskrete Aufgabenstellungen konzipiert
- **Evolutionäre Algorithmen:**
  Idee: Nutzung der Mechanismen der biol. Evolution zur Verbesserung von Lösungen
  Sehr gut für kontinuierliche, diskrete und kombinatorische Probleme geeignet

Gemeinsame Eigenschaften:

- **Jeder Optimierungslauf kann andere Ergebnisse liefern (stochastische Verfahren)**
- **Keine Optimalitätsgarantie**
- **Wenig Vorwissen erforderlich**
- **Existierende (Teil-)Lösungen oder Lösungen eines ähnlichen Problems leicht integrierbar**
- **Mehr Rechenleistung bringt bessere Ergebnisse! Gut parallelisierbar**

Grundlagen der Optimierung – No-free-Lunch-Theoreme

No-free-Lunch-Theoreme:

(Nichts ist umsonst)

Bezogen auf die Menge aller mathematisch möglichen Probleme sind alle Suchalgorithmen im Durchschnitt gleich gut (oder gleich schlecht).

[Wol95, Wol97]

→ **Metaheuristik mit möglichst viel Anwendungswissen anpassen:**

- **an das Problem oder besser**
- **an die Problemklasse**

Umkehrschluss:

Es gibt keinen Universalalgorithmus, der alle Aufgaben am effizientesten löst!

... genauso wenig, wie es ein Perpetuum mobile gibt!