

Modellierung, Simulation und Optimierungsverfahren II

Dr. Colin W. Glass

Institut für Höchstleistungsrechnen

07.05.2014

Grundkonzepte

Jede Optimierung basiert auf drei grundsätzlichen Konzepten:

- Ziel (aka "Objective")
- Repräsentation
- Evaluierungsfunktion

Drei berühmte Standardprobleme

- Erfüllbarkeitsproblem der Aussagenlogik (SAT, von satisfiability)
- Nonlinear programming problem (NLP)
- Traveling salesman problem (TSP)

Viele Optimierungsaufgaben lassen sich auf (diese) Standardprobleme reduzieren oder zumindest gut annähern.

Diese Standardprobleme bilden die Basis unserer Betrachtungen. Weitere Probleme werden nach Bedarf betrachtet.

Nomenklatur

- Das **Ziel** ist, was wir wirklich haben wollen.
- Eine **Repräsentation** modelliert das Problem auf Basis von Variablen.
- Die Anzahl der Variablen ist die **Dimension** des Problems.
- Eine **Lösung** ist ein Set an Werten - ein Wert für jede Variable des Problems.
- Eine Lösung ist zulässig, wenn sie gegen keine Zwangsbedingungen verstösst (fortan "**(hard) Constraints**" genannt).
- Der **Suchraum** $\mathcal{S}$ ist der Raum aller Lösungen, alle zulässigen Lösungen bilden den Unterraum $\mathcal{F}$ (feasible).

Nomenklatur

- Ein **Kriterium** ist eine Grösse, die sich auf die Güte bzw. die Fitness auswirkt.
- Die **Evaluierungsfunktion** $F(\mathbf{x})$ (auch **Fitnessfunktion** genannt) bildet eine Lösung auf einen spezifischen Wert ab.
- Dieser Wert ist die **Fitness** einer Lösung und quantifiziert deren Güte.
- Die **Fitnesslandschaft** ist die (Hyper-)Fläche der Fitnesswerte.

Ausblick

Die wichtigsten nächsten Schritte

- Constraints
- Distanz
- Nachbarschaft

Nomenklatur

- Die **Distanz** basiert auf der Repräsentation und legt den Abstand zwischen zwei Lösungen fest.
- Die **Nachbarschaft** einer Lösung x besteht aus allen Lösungen, die sich innerhalb einer gewissen Distanz von x befinden.
- Ein **lokales Optimum** ist eine zulässige Lösung, deren benachbarten, zulässigen Lösungen alle schlechter sind.
- Das **globale Optimum** ist die bestmögliche, zulässige Lösung des Problems.

Constraints

Hard Constraints müssen erfüllt werden. Diese nehmen wir als default (Constraints $\hat{=}$ hard Constraints).

Das NLP hat beispielsweise Constraints, in Form von Ungleichungen.

Einerseits verringern Constraints die Menge der Lösungen die wir in Betracht ziehen müssen, andererseits machen Constraints die Suche oft schwieriger, weil wir uns in $\mathcal{F}$ bewegen wollen.

Beispiel: Constraints

Es soll ein Stundenplan für die Uni erstellt werden. Es ergeben sich folgende Constraints:

- Jeder Saal darf nur einfach belegt werden.
- Jede Vorlesung muss in einem hinreichend grossen Saal stattfinden.
- Studenten können verschiedene Vorlesungen nicht zeitgleich hören.
- Dozenten können verschiedene Vorlesungen nicht zeitgleich halten.

Nun ist es nicht leicht, zulässige Lösungen zu finden. Die Schwierigkeit hängt vom spezifischen Szenario ab.

Beispiel: Constraints

Zusätzlich gibt es "soft" Constraints. Diese sollen möglichst gut erfüllt werden:

- Studenten sollten hinreichend viel Zeit haben, um den Vorlesungssaal zu wechseln.
- Dozenten sollten hinreichend viel Zeit haben, um den Vorlesungssaal zu wechseln.
- Aufeinanderfolgende Vorlesungen sollten in möglichst nah beieinanderliegenden Sälen stattfinden.
- Vorlesungen die mehrmals die Woche stattfinden sollten möglichst regelmässig auf die Woche verteilt sein (z.B. Di und Fr)
- ...

Hier zeigt sich sofort die Schwierigkeit, gute Evaluierungsfunktionen (Modelle) zu erzeugen.

Umgang mit Constraints

In erster Linie gibt es drei Möglichkeiten, was bei unzulässigen Lösungen geschehen soll:

- Sehr schlechte Fitness geben.
- Auf die naheliegendste zulässige Lösung verschieben ("flicken").
- Maß an Unzulässigkeit in $F(\mathbf{x})$ berücksichtigen.

Umgang mit Constraints

- Sehr schlechte Fitness geben.

Dies ist die einfachste Herangehensweise. Sie ist dann vernünftig, wenn die unzulässigen Gebiete klein sind und wir erwarten, dass sich die gesuchte, optimale Lösung weit entfernt davon befinden wird.

Ausserdem ist diese Herangehensweise immer möglich.

Umgang mit Constraints

- Auf die naheliegendste zulässige Lösung verschieben.

Dies ist bei zulässigen Wertebereichen pro Dimension (z.B. $0 \leq x_i \leq 10$) sehr leicht und dann sehr sinnvoll, wenn die gesuchte optimale Lösung nahe oder auf der Grenze zum unzulässigen Gebiet liegen könnte.

Es gibt wiederum zwei Möglichkeiten:

- Die Lösung x durch die naheliegendste Zulässige ersetzen.
- Anstelle von x die naheliegendste Zulässige evaluieren.

Umgang mit Constraints

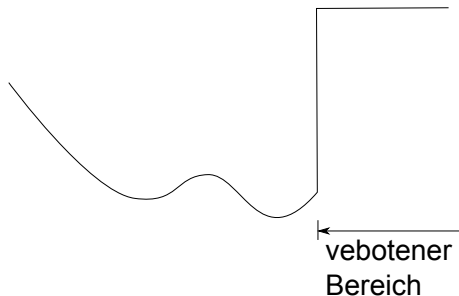
- Maß an Unzulässigkeit in $F(\mathbf{x})$ berücksichtigen.

Dies erfordert eine Quantifizierung der Unzulässigkeit, was schwierig sein kann.

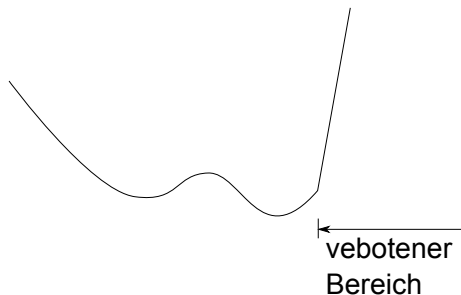
Es erlaubt uns jedoch auch in unzulässigen Gebieten eine zielgerichtete Suche durchzuführen.

Umgang mit Constraints

- $F(\mathbf{x})$ wird auf schlechten, konstanten Wert gesetzt



- Ein z.B. linear ansteigender Penalty-Term wird zu $F(\mathbf{x})$ addiert



Distanz

Das Distanzmaß hängt stark von der Repräsentation ab.

Die Distanz legt den Abstand zwischen zwei Lösungen fest. Sie soll also quantifizieren, wie ähnlich zwei Lösungen sind.

Folglich haben wir es hier mit einem Modell zu tun, welches "Ähnlichkeit" quantifizieren soll.

Distanz: NLP

Für das NLP ist die euklidische Distanz eine intuitive Wahl:

$$\text{dist}(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Eine Alternative wäre z.B.:

$$\text{dist}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^n |(x_i - y_i)|$$

P.s.: Diese beiden Distanzmaße verhalten sich unterschiedlich, obschon sie beide auf $(x_i - y_i)$ aufbauen.

Distanz: SAT

Bei booleschen Repräsentationen (SAT) ist die Anzahl ungleicher Bits eine sinnvolle Distanz. Für ein SAT hat jede Lösung genau n Nachbarn mit Distanz 1.

Z.B. x mit $n = 10$:

0110010010

Hat innerhalb Distanz 1 die Nachbarn:

1110010010 (erstes Bit geflipt)

0010010010 (zweites Bit geflipt)

...

0110010011 (letztes Bit geflipt)

Distanz: SAT

Vorsicht: wenn wir das SAT Problem als Integer $[0, 2^n - 1]$ repräsentieren (und dann vor der Auswertung erst umrechnen) und die Distanz von 1 anschauen (Differenz der Integer $\hat{=}$ Distanz), verhält sich diese ganz anders:

Z.B. für $n = 6$:

Lösung 31 $\hat{=}$ 011111

Lösung 32 $\hat{=}$ 100000

Mit dem vorherigen Distanzmaß entspricht dies einer Distanz von 6.

Distanz: TSP

Beim TSP kann ein Distanzmass z.B. ein Zwei-Städte-Tausch sein. Die Distanz 1 entspricht dann dem Vertauschen von zwei Städten. Für ein TSP hat jede Lösung genau $n(n-1)/2$ Nachbarn mit Distanz 1.

Z.B. x mit $n = 10$:

4 - 9 - 10 - 5 - 2 - 1 - 6 - 8 - 7 - 3

Hat innerhalb Distanz 1 die Nachbarn:

4 - 2 - 10 - 5 - 9 - 1 - 6 - 8 - 7 - 3 (Stadt 2 und 9 vertauschen)

4 - 9 - 10 - 6 - 2 - 1 - 5 - 8 - 7 - 3 (Stadt 5 und 6 vertauschen)

7 - 9 - 10 - 5 - 2 - 1 - 6 - 8 - 4 - 3 (Stadt 4 und 7 vertauschen)

...

Distanz 2 wäre 2 Mal zwei Städte vertauschen:

7 - 9 - 10 - 5 - 2 - 6 - 1 - 8 - 4 - 3 (Stadt 4 und 7 vertauschen, danach Stadt 1 und 6 vertauschen)

Distanz: TSP

Bei diesem Distanzmaß sei zu beachten, dass es schwierig ist, die Distanz zwischen weit auseinanderliegenden Lösungen zu bestimmen.

Selbst wenn ausgehend von einer Lösung k Mal zwei Städte vertauscht werden und kein Tausch wiederholt werden darf, ist unklar, ob die resultierende Lösung Distanz k hat.

Ausgehend von zwei beliebigen Lösungen ist die Distanzbestimmung i.A. sehr schwer, da es zahlreiche verschiedene Wege gibt und der Kürzeste die Distanz bestimmt.

Distanz: Gebiete einfärben

Betrachten wir folgendes Optimierungsproblem:

- Gebieten sollen Farben zugeordnet werden.
- Es sollen keine angrenzenden Gebiete die selbe Farbe haben.
- Es sollen möglichst wenige Farben verwendet werden.

Die Repräsentation sei:

- Die Gebiete werden durchnummeriert.
- Eine Lösung: Ein Vektor mit Integern.
- Die Position der Elemente im Vektor entspricht der Nummer i des Gebiets.
- Die Integer werden auf Farben abgebildet.

Distanz: Gebiete einfärben

Z.B. Grün $\hat{=}$ 1, Blau $\hat{=}$ 2, Gelb $\hat{=}$ 3, Rot $\hat{=}$ 4 ...

Was ist ein sinnvolles Distanzmaß?

In zwei Lösungen kann das gleiche Gebiet:

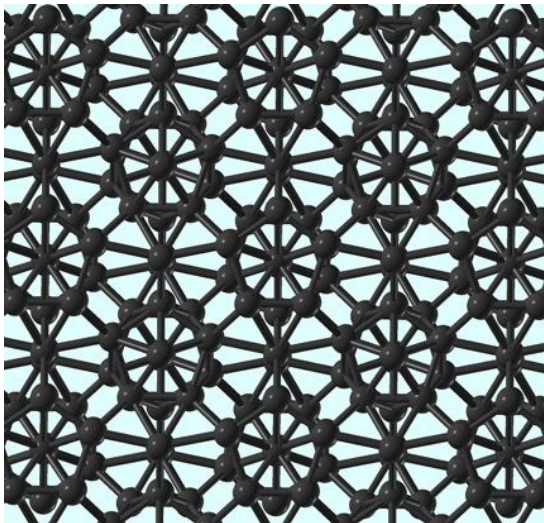
- Die gleiche Farbe haben.
- Zwei unterschiedliche Farben haben.

D.h., für $x_i - y_i$ hat nur $= 0$ und $\neq 0$ Bedeutung.

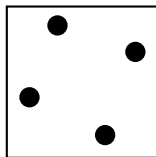
Ob $x_i - y_i$ z.B. $= 3$ oder $= 2$ ist, macht im Sinne einer Distanz keinen Unterschied.

Folglich: Ein sinnvolles Distanzmaß ist die Anzahl ungleich gefärbter Gebiete.

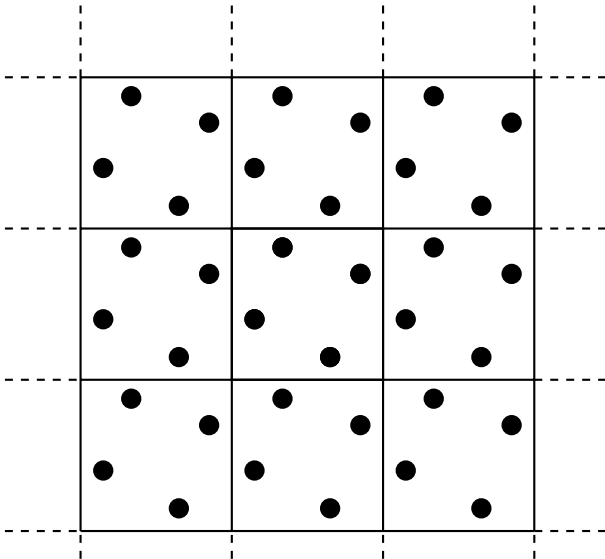
Kristallstrukturen: Beispiel für schwierige Distanzbestimmung



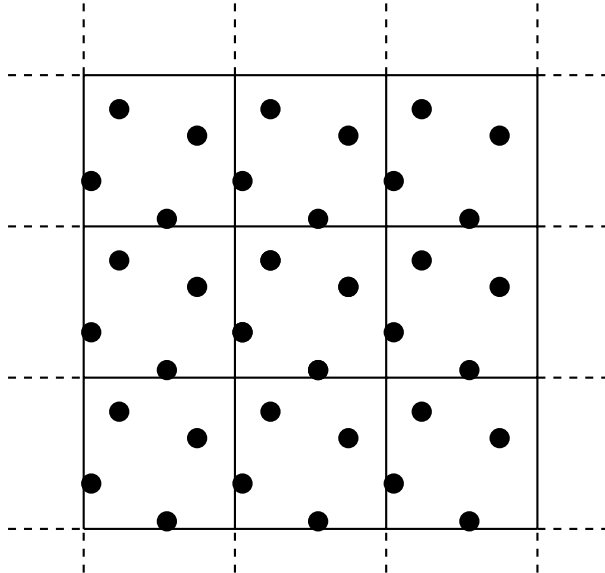
Schwierige Distanzbestimmung: Kristallstrukturen



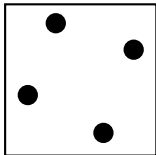
Schwierige Distanzbestimmung: Kristallstrukturen



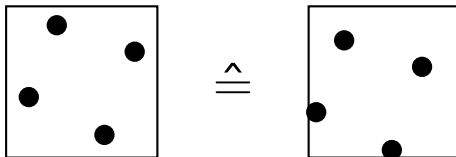
Schwierige Distanzbestimmung: Kristallstrukturen



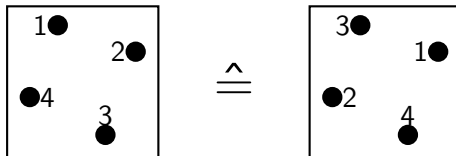
Schwierige Distanzbestimmung: Kristallstrukturen



Schwierige Distanzbestimmung: Kristallstrukturen



Schwierige Distanzbestimmung: Kristallstrukturen



Richtung

Die Distanz quantifiziert den absoluten Abstand. Manchmal lässt sich auch eine sinnvolle Richtung bestimmen.

Beispielsweise ein Parametrisierungsproblem (Maschine, Verfahren, ...) hat meist eine sinnvolle Richtung: wird Parameter x_i um $+1$ verändert, so ist dies eine andere Richtung als $x_j + 1$ oder auch $x_i - 1$.

Wird eine Lösung $\mathbf{x}$ verändert und dadurch besser, kann die Richtung beibehalten werden oder von einer Lösung $\mathbf{y}$ kann in die selbe Richtung gesucht werden.

Richtung

Gewisse Optimierungsverfahren können/sollten nur dann verwendet werden, wenn eine sinnvolle Richtung feststellbar ist.

- Optimierungsverfahren die mit Ableitungen von $F(\mathbf{x})$ arbeiten.
- Evolutionärer Algorithmus (EA) dessen Vererbungsoperator auf (gewichteten) Mittelwerten basiert.
- Schwarmalgorithmus
- CMA-ES
- ...

Richtung

Beim NLP kann man sich unter der Richtung direkt etwas vorstellen.

Beim SAT gibt es nur zwei Zustände pro Dimension (0 und 1). Folglich kann z.B. eine 1D Richtung auf eine andere Lösung z.T. übertragen werden, aber es kann nicht zwei Mal hintereinander in die selbe Richtung gegangen werden.

Beim TSP hat ein 2-Städte-Tausch keine eindeutige Richtung.

Nachbarschaft

Gegeben sei ein Suchraum $\mathcal{S}$. Die Nachbarschaft $\mathcal{N}$ einer Lösung $\mathbf{x}$ ist definiert als alle Lösungen, die sich innerhalb einer gewissen Distanz (ϵ) von $\mathbf{x}$ befinden.

$$\mathcal{N}(\mathbf{x}) = \{\mathbf{y} \in \mathcal{S} : \text{dist}(\mathbf{x}, \mathbf{y}) \leq \epsilon\}$$

Lokales Optimum

Eine Lösung ist dann ein lokales Optimum, wenn sie zulässig ist und sich innerhalb der Nachbarschaft keine bessere zulässige Lösung befindet.

Das heisst $\mathbf{x} \in \mathcal{F}$, so dass gilt:

$$F(\mathbf{x}) \leq F(\mathbf{y}) \quad \forall \mathbf{y} \in \mathcal{N}(\mathbf{x}) \cap \mathcal{F}$$

Hillclimber

Der Hillclimber (Name geht von einer Maximierung aus) ist das Paradebeispiel für lokale Optimierung: er sucht in seiner Nachbarschaft nach einer besseren Lösung, bewegt sich da hin, sucht eine bessere Lösung, bewegt sich da hin ...

Oft wiederholen:

- Wähle neue Lösung $\mathbf{x}_{neu} \in \mathcal{N}(\mathbf{x})$
- if $(F(\mathbf{x}_{neu}) \geq F(\mathbf{x}))$ then $(\mathbf{x} = \mathbf{x}_{neu})$

Hillclimber

Grundsätzlich kann ein Hillclimber pro Iteration:

- Eine zufällige Lösung $\in \mathcal{N}(\mathbf{x})$ generieren und übernehmen, wenn diese besser ist.
- Alle Lösungen $\in \mathcal{N}(\mathbf{x})$ generieren und die Beste übernehmen.

Die erste Option kommt i.d.R. schneller vorwärts, bei der Zweiten wird immer der bestmögliche Schritt gemacht und es ist sofort klar wann ein (lokales) Optimum erreicht wurde.

Wir gehen im folgenden von der ersten Option aus.

P.s: Natürlich sind dies die beiden Extreme – es können auch k Lösungen generiert werden.

Hillclimber

Der Hillclimber besticht durch seine Schlichtheit - notwendig ist:

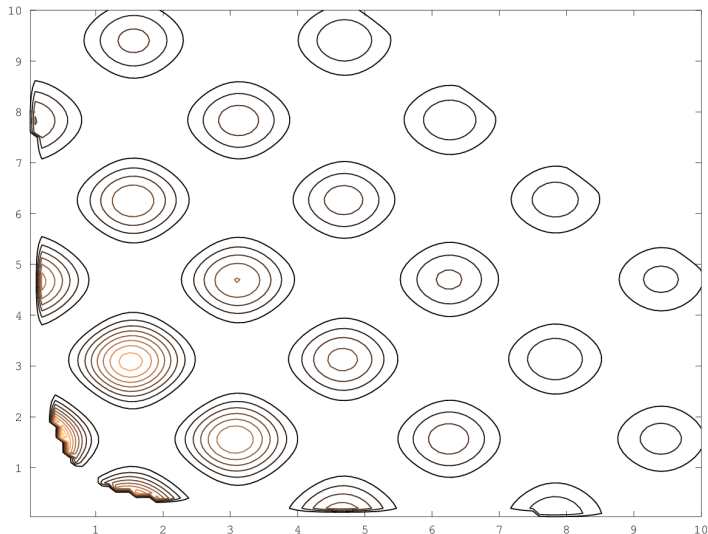
- Eine Methode um, ausgehend von einer Lösung $\mathbf{x}$, eine (zufällige) Lösung $\mathbf{x}_{neu} \in \mathcal{N}(\mathbf{x})$ zu generieren.

Dadurch kann der Hillclimber schnell (und hoffentlich fehlerfrei) implementiert werden.

P.s.: Diese Methode fällt in die Klasse der **Operatoren**. Sie ist unabhängig vom Hillclimber und kann z.B. für einen EA direkt übernommen werden (Mutation).

Problem der lokalen Optimierung:

Viele lokale Optima!



Lokalen Optima entkommen

Eine offensichtliche Erweiterung eines lokalen Optimierungsverfahrens ist die Fähigkeit, lokalen Optima zu entkommen.

Stochastic Hillclimber

Um dem Hillclimber die Möglichkeit zu geben, einem lokalen Optimum zu entkommen, bedarf es:

- einer gewissen Wahrscheinlichkeit, eine schlechtere Lösung zu wählen.

Dies hat relevante Auswirkungen.

Stochastic Hillclimber

Die Wahrscheinlichkeit sieht so aus (Maximierungsproblem):

$$p = \frac{1}{1 + e^{\frac{F(\mathbf{x}) - F(\mathbf{x}_{neu})}{T}}}$$

T ist ein neuer Parameter ("Temperatur").

$e^0 = 1 \rightarrow p = 0.5$, das heisst wenn $F(\mathbf{x}) = F(\mathbf{x}_{neu})$ haben beide die gleiche Wahrscheinlichkeit.

Je besser $\mathbf{x}_{neu}$ (vs. $\mathbf{x}$), desto eher wird es übernommen. Je schlechter, desto eher bleibt man bei $\mathbf{x}$.

Stochastic Hillclimber

Wie soll T gewählt werden?

Für $T \rightarrow 0$ liegt ein Hillclimber vor (besser $\hat{=}$ nehmen, schlechter $\hat{=}$ verwerfen).

Für $T \rightarrow \infty$ geht $p \rightarrow 0.5$, egal ob $\mathbf{x}_{neu}$ besser oder schlechter ist als $\mathbf{x}$.
Also: Große T führen zu einem "Random Walk".

Stochastic Hillclimber

Folglich brauchen wir für den Stochastic Hillclimber:

- Eine quantitativ sinnvolle Evaluierungsfunktion.

Dies ist überhaupt nicht selbstverständlich!

Und es stellen sich die Fragen, wie groß T sein soll und wann die Optimierung abgebrochen werden soll.

Simulated Annealing

Simulated Annealing basiert auf dem Stochastic Hillclimber, nur werden noch zwei Dinge getan:

- Bessere Lösungen werden immer genommen!
- T wird im Verlaufe der Optimierung gesenkt.

Simulated Annealing

Simulated Annealing basiert auf dem Stochastic Hillclimber, nur werden noch zwei Dinge getan:

- Bessere Lösungen werden immer genommen!
- T wird im Verlaufe der Optimierung gesenkt.

Anschaulich bedeutet dies:

- Am Anfang (große T) können schlechtere Lösungen trotzdem genommen zu werden ($p \approx 0.5$).
- Gegen Ende (kleine T) verhält sich Simulated Annealing genau wie der Hillclimber (nur bessere Lösungen.)

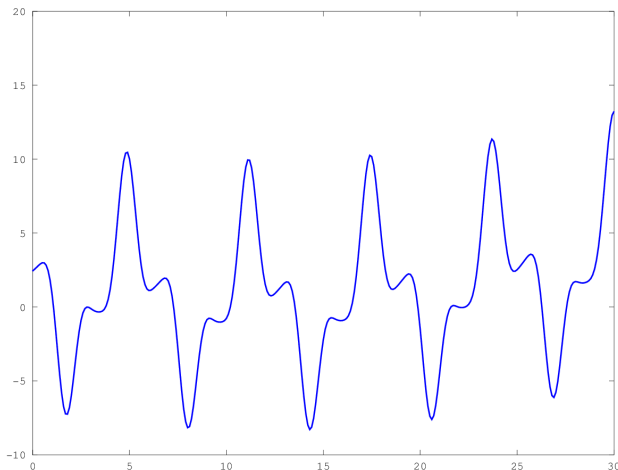
Simulated Annealing

Für Simulated Annealing ergibt sich folglich noch eine weitere Frage, die es zu beantworten gilt:

- Wie schnell sollen wir T "runterkühlen"?

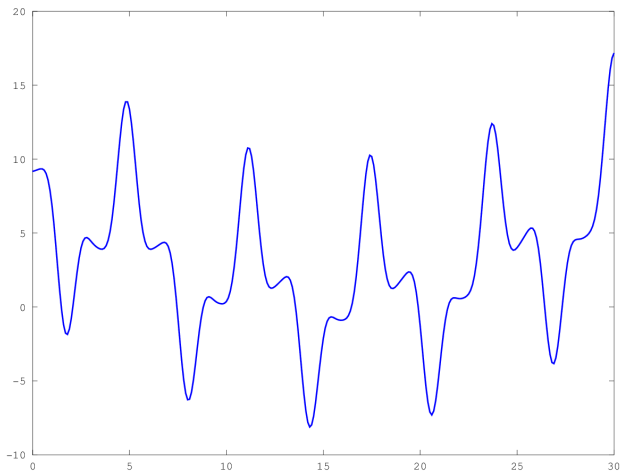
Simulated Annealing

Sehr problematisch: Relative Güte von lokalen Optima gering, Barrieren hoch. Guter Bereich schwer zu identifizieren, Barrieren schwer zu überwinden.



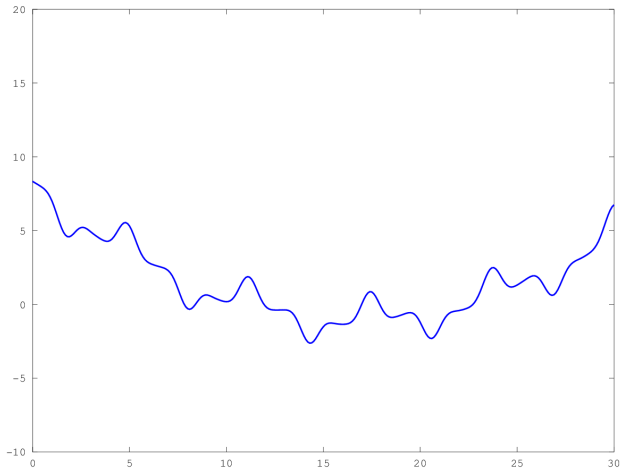
Simulated Annealing

Besser: Relative Güte ausgeprägter, aber die Barrieren sind noch immer sehr hoch. Probleme beim Überwinden.

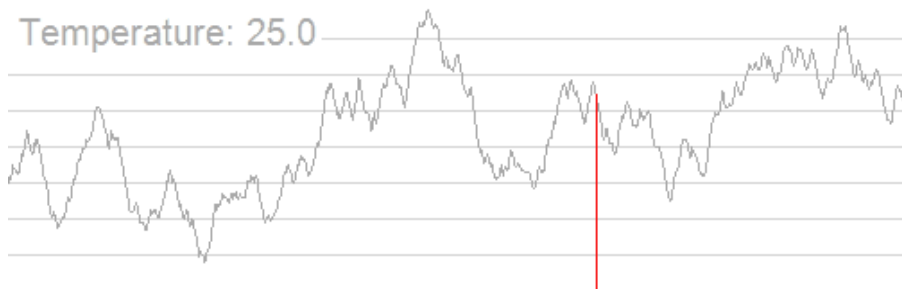


Simulated Annealing

Diese Landschaft ist z.B. viel besser geeignet für Simulated Annealing.



Simulated Annealing



Globale vs. lokale Optimierung

Bei einer lokalen Optimierung wird, ausgehend von einer Lösung, ein umliegendes lokales Optimum gesucht (je nach Fitnesslandschaft und Nachbarschaft können das mehrere sein). Üblicherweise werden dabei nur Lösungen aus der Nachbarschaft der bisher besuchten Lösungen betrachtet.

Bei einer globalen Optimierung wird nach der besten, zulässigen Lösung gesucht – dem globalen Optimum (wie bereits als "Optimierungsproblem" dargestellt). Globale Optimierung sei der Default von Optimierung.

Ausblick

Es geht weiter mit (lokaler) Optimierung basierend auf Ableitungen:

- Ableitungen von $F(\mathbf{x})$
- Newtonverfahren
- Gradientenverfahren
- Konjugierte Gradienten