

Modellierung, Simulation und Optimierungsverfahren II

Dr. Colin W. Glass

Institut für Höchstleistungsrechnen

30.04.2014

Optimierung

Eine Handlung, Verfahren oder Methode um ein Problem so gut wie möglich zu lösen.

Problem

Ein Problem existiert, wenn eine Disparität zwischen dem gewünschten und dem tatsächlichen Zustand festgestellt ist. Lösungen wiederum sind Wege, die verfügbaren Ressourcen zuzuweisen, um diese Disparität zu minimieren. [Z. Michalewicz, D. Fogel: How to solve it]

P.s: Der gesamte Optimierungsteil der Vorlesung wird sich stark an dieses Buch halten.

Grundkonzepte

Jede Optimierung basiert auf drei grundsätzlichen Konzepten:

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

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.

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.

SAT

Beim SAT Problem soll für eine aussagenlogische Formel eine Lösung gefunden werden, die das Ergebnis "Wahr" hat (Objective). Beispiel in konjunktiver Normalform (jedes SAT lässt sich prinzipiell in diese Form überführen):

$$F(\mathbf{x}) = (x_{12} \vee \neg x_{55}) \wedge (\neg x_{46} \vee x_{23} \vee x_{55}) \wedge (x_{12} \vee x_{35}) \wedge \cdots \wedge (x_3 \vee \neg x_{98} \vee x_{74})$$

Jede Variable x_i (Anzahl Variablen $n = \text{Dimension}$) kann den Wert Wahr oder Falsch.

Repräsentation: SAT

- Jede Variable x_i des SAT kann durch ein Bit repräsentiert werden, $0 \hat{=}$ Falsch, $1 \hat{=}$ Wahr.
- Eine Lösung: Ein Bitvektor der Länge n .
- Die Position der Elemente im Vektor entspricht der Nummer i der Variable (z.B. drittes Element $\hat{=}$ x_3).

Jeder Bitvektor der Länge n ist folglich eine (zulässige) Lösung.

Grösse des Suchraums: SAT

Grösse des Suchraums ($|\mathcal{S}|$): Anzahl möglicher Bitvektoren der Länge n .

Zwei Möglichkeiten pro Bit, folglich $|\mathcal{S}| = 2^n$.

$$n = 10 \quad \hat{=} \quad 2^{10} \quad = \quad 1'024$$

$$n = 20 \quad \hat{=} \quad 2^{20} \quad = \quad 1.05 \cdot 10^6$$

$$n = 50 \quad \hat{=} \quad 2^{50} \quad = \quad 1.13 \cdot 10^{15}$$

$$n = 100 \quad \hat{=} \quad 2^{100} \quad = \quad 1.27 \cdot 10^{30}$$

$$n = 200 \quad \hat{=} \quad 2^{200} \quad = \quad 1.61 \cdot 10^{60}$$

NLP

Ein Beispiel für ein NLP: Es soll der Wert der Funktion G maximiert werden (Objective: Das Maximum einer nichtlinearen Funktion):

$$G(\mathbf{x}) = \left| \frac{\sum_{i=1}^n \cos^4(x_i) - 2 \prod_{i=1}^n \cos^2(x_i)}{\sqrt{\sum_{i=1}^n i x_i^2}} \right|$$

$$\text{mit } \prod_{i=1}^n x_i \geq 0.75 \quad \sum_{i=1}^n x_i \leq 7.5n \quad \text{und} \quad 0 \leq x_i \leq 10$$

für $1 \leq i \leq n$

Repräsentation: NLP

- Jede Variable x_i des NLP kann durch eine reelle Zahl repräsentiert werden.
- Eine Lösung: Ein Vektor der Länge n .
- Die Position der Elemente im Vektor entspricht der Nummer i der Variable (z.B. drittes Element $\hat{=}$ x_3).

Jeder Vektor reeller Zahlen der Länge n ist folglich eine Lösung. Sind die Constraints erfüllt, handelt es sich um eine zulässige Lösung.

Grösse des Suchraums: NLP

Grösse des Suchraums ($|S|$): Anzahl möglicher Vektoren reeller Zahlen der Länge n .

Folglich ist $|S|$ im Prinzip ∞ .

Gehen wir von einer Diskretisierung mit Genauigkeit 10^{-6} aus und beachten direkt einen Teil der Constraints ($0 \leq x_i \leq 10$), gibt es pro Variable eine endliche Anzahl Möglichkeiten ($10'000'000$).

$$|S| = 10'000'000^n = 10^{7n}.$$

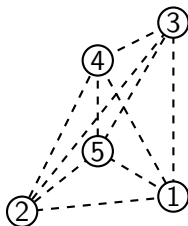
Die Anzahl möglicher Lösungen ist (selbst für kleine n , z.B. $n = 20$) gewissermassen absurd groß (10^{140}).

TSP

Ein Traveling Salesman möchte genau n Städte besuchen (Dimension = n) und in seine Anfangsstadt zurückkehren. Die Reise von Stadt zu Stadt kostet (Geld, Zeit, Nerven, ...). Gesucht wird die günstigste Route (Objective).

Eine Lösung wäre z.B. 1 - 2 - 5 - 4 - 3

Natürlich bedeutet dies, dass am Ende der Route die Strecke 3 - 1 ebenfalls gereist wird.

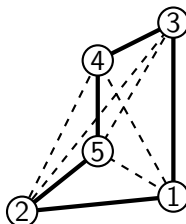


TSP

Ein Traveling Salesman möchte genau n Städte besuchen (Dimension = n) und in seine Anfangsstadt zurückkehren. Die Reise von Stadt zu Stadt kostet (Geld, Zeit, Nerven, ...). Gesucht wird die günstigste Route (Objective).

Eine Lösung wäre z.B. 1 - 2 - 5 - 4 - 3

Natürlich bedeutet dies, dass am Ende der Route die Strecke 3 - 1 ebenfalls gereist wird.



Repräsentation: TSP

- Die Städte werden mit Integern durchnummeriert.
- Eine Lösung: Ein Vektor mit Integern der Länge n , jeder Wert von 1 bis n kommt genau ein Mal vor.
- Die Reihenfolgen der Elemente im Vektor entspricht der Reihenfolge der Stadtbesuche.
- Der Werte eines Elements bezeichnet die Nummer der Stadt.

Jede Permutation der Integern von 1 bis n ist folglich eine (zulässige) Lösung.

Grösse des Suchraums: TSP

Grösse des Suchraums ($|\mathcal{S}|$): Anzahl Permutationen (ohne Wiederholung) der Städte, also $n \cdot (n-1) \cdot \dots \cdot 1$, auch als $n!$ geschrieben.

$$|\mathcal{S}| = n!$$

P.s: jede Route liegt n Mal identisch vor (jede Stadt kann die Anfangsstadt sein) und für ein symmetrisches TSP entsprechend $2n$ Mal (umgekehrte Richtung ist äquivalent). Für eine Repräsentation, die keine identischen Lösungen zulässt, wäre $|\mathcal{S}|$:

$$|\mathcal{S}| = \frac{n!}{2n} = \frac{(n-1)!}{2}$$

Dies ist die Anzahl einzigartiger Lösungen (symmetrisches TSP).

Grösse des Suchraums: TSP

5 Städte	$\hat{=}$	5!	=	120
10 Städte	$\hat{=}$	10!	=	$3.63 \cdot 10^6$
20 Städte	$\hat{=}$	20!	=	$2.43 \cdot 10^{18}$
100 Städte	$\hat{=}$	100!	=	$9.33 \cdot 10^{157}$
200 Städte	$\hat{=}$	200!	=	∞

Große Zahlen

Anzahl möglicher Ergebnisse beim Lotto (6 aus 49): ca. $7.15 * 10^8$

Distanz zur Sonne (in m): ca. $1.5 * 10^{11}$

Alter des Universums (in s): ca. $1.4 * 10^{17}$

Sandkörner auf der Erde: ca. $7 * 10^{20}$

Sterne im beobachtbaren Universum: ca. $6 * 10^{22}$

Masse der Erde (in kg): ca. $6 * 10^{24}$

Atome im beobachtbaren Universum: ca. 10^{80}

Große Zahlen

Betrachten wir nur die einzigartigen (!) Lösungen eines symmetrischen TSP:

Es ist wahrscheinlicher im Lotto zu gewinnen, als die beste Route eines 14-Städte TSP zu erraten.

Wenn wir alle Lösungen eines 23-Städte TSP evaluieren möchten und einen Computer, der pro Sekunde 1'000 Lösungen evaluieren kann, zum Zeitpunkt des Urknalls hätten zu rechnen anfangen lassen, wären wir heute bereits zu ca. 25% durch.

Wenn wir alle Lösungen eines TSP jeweils genau einem Atom zuordnen wollen, bräuchten wir ca. 70% aller Atome des beobachtbaren Universums, für die Lösungen eines 60-Städte TSP.

Evaluierungsfunktion

Die Evaluierungsfunktion ($F(\mathbf{x})$) ist nicht das gleiche wie das Ziel! $F(\mathbf{x})$ soll einen Wert zurück liefern, der eine relative Bewertung von Lösungen ermöglicht. Darüber hinaus ist eine absolute Bewertung ebenfalls interessant. $F(\mathbf{x})$ bestimmt auch, wie $\mathbf{x} \notin \mathcal{F}$ bewertet wird.

Die Evaluierungsfunktion ist nicht vorgegeben, sondern muss bestimmt bzw. geschaffen werden.

Wichtig: Lokale Optima und insbesondere das globale Optimum des realen Problems sollten lokale bzw. globales Optima der Evaluierungsfunktion sein.

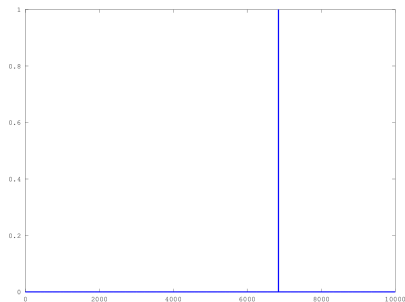
Evaluierungsfunktion: TSP

Die Evaluierungsfunktion des TSP ist trivial:

Die effektiven Kosten der Route.

Evaluierungsfunktion: SAT

Für SAT ist die Bestimmung einer Evaluierungsfunktion schwierig: die meisten Lösungen sind "Falsch", also 0. Die Fitnesslandschaft wäre folglich flach, bis auf die gesuchte Lösung (können auch mehrere sein), die den Wert 1 hat. Hier müssen offenbar noch Schlauheiten in $F(\mathbf{x})$ eingebaut werden, um die relative Güte von Lösungen zu bestimmen.



Eine simple Möglichkeit wäre z.B. die Anzahl erfüllter Klauseln zu nehmen. Sind alle Klauseln erfüllt, ist das Objective erreicht, somit entspricht der maximale Wert dieser Evaluierungsfunktion dem Objective.

Evaluierungsfunktion: NLP

Die Evaluierungsfunktion des NLP ergibt sich in erster Linie aus dem Wert der Funktion G .

Jedoch muss entschieden werden, was im Falle $\mathbf{x} \notin \mathcal{F}$ geschehen soll.

Dafür gibt es viele Möglichkeiten. Die einfachste (nicht unbedingt die beste) wäre, allen Lösungen $\notin \mathcal{F}$ den Wert 0 zuzuweisen.

Vorsicht: Problemspezifisch $\rightarrow$ Maximierung eines Betrags.

Fitnesslandschaft

Die **Fitnesslandschaft** ergibt sich aus den Fitnesswerten aller Lösungen.

Die Form der Fitnesslandschaft hat einen großen Einfluss auf die Schwierigkeit eines Problems.

Fitnesslandschaft: NLP

Wenn möglich, macht es oft Sinn, sich ein grobes Bild von der Fitnesslandschaft zu machen.

Z.B. geht das gut für NLP: Der Suchraum wird gerastert und an den Knoten ausgewertet. Anschliessend visualisiert.

P.s: Solche Bilder lassen sich ganz einfach mit "mesh" in Octave generieren (oder "surf").

```
>> tx = ty = linspace(0, 10, 101)';
```

```
>> Values = NLP_2D(tx, ty);
```

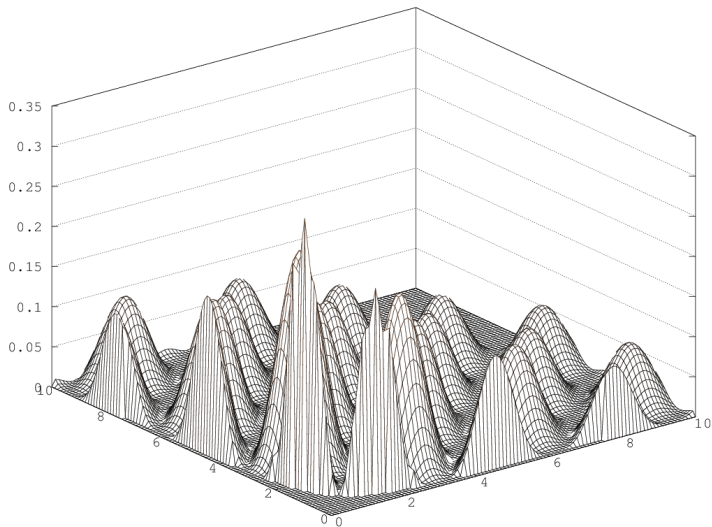
```
>> mesh(tx, ty, Values)
```

Fitnesslandschaft: NLP

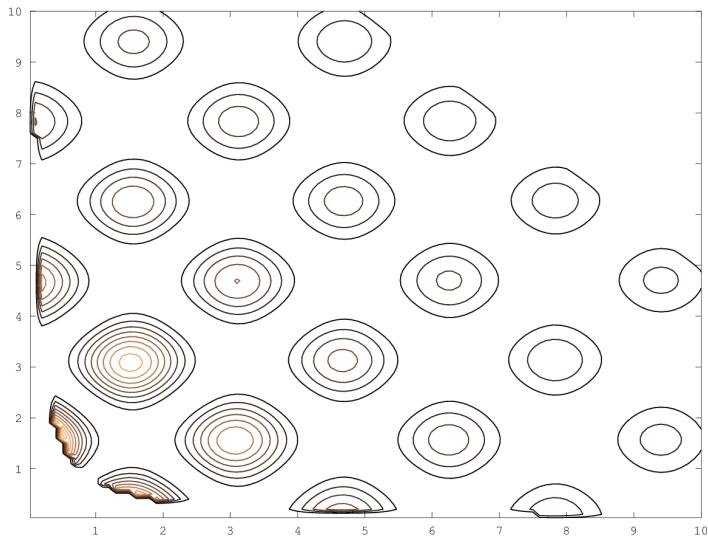
Kurz runtergehackter Code sieht ungefähr so aus:

```
function Values = NLP_2D(tx,ty)
Values = zeros (length(tx),length(ty));
for indx = 1:length(tx)
    for indy = 1:length(ty)
        if (tx(indx)*ty(indy))>0.75
            if (tx(indx)+ty(indy))<15
                Values(indx,indy) = abs((((cos(tx(indx)))^4+(cos(ty(indy)))^4 ...
                    - (2*(cos(tx(indx)))^2*(cos(ty(indy)))^2)) ...
                    / sqrt(tx(indx)^2+2*ty(indy)^2)));
            end
        end
    end
end
```

Fitnesslandschaft: NLP

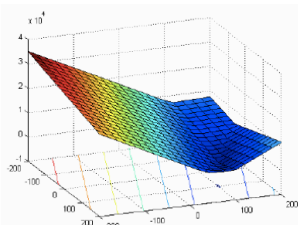


Fitnesslandschaft: NLP

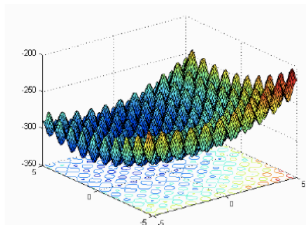


Fitnesslandschaften: CEC

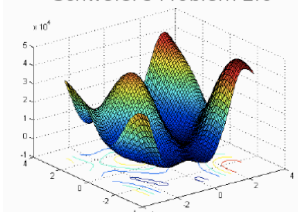
Fitnesslandschaften können total unterschiedlich aussehen.
Beispiele aus der CEC Benchmark Suite:



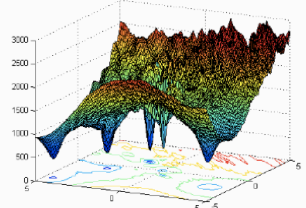
Schwefel's Problem 2.6



Rastrigin's Function



Schwefel Problem 2.13



Hybrid Function

Optimierungsproblem

Definition:

Gegeben sei ein Suchraum $\mathcal{S}$ und dessen zulässiger Bereich $\mathcal{F} \subseteq \mathcal{S}$.

Gesucht ist $\mathbf{x} \in \mathcal{F}$, so dass gilt:

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

P.s: Wir gehen hier von einem Minimierungsproblem aus. Minimierung gilt gemeinhin als der default.

Schwierige Probleme

Es gibt viele Gründe, warum Probleme schwierig zu lösen sein können. Die in unseren Betrachtungen wichtigsten Gründe sind (i.d.R. sind mehrere Gründe zugleich notwendig):

- Der Suchraum ist derart riesig, dass er nicht erschöpfend durchsucht werden kann (fortan "exhaustive search" genannt).
- Die Fitnesslandschaft hat keine schöne Form, ist z.B. Schroff.
- Das Problem lässt sich nicht in Teilprobleme zerlegen (nicht separabel).
- Das Problem ist derart komplex, dass schwer simplifizierte Modelle verwendet werden müssen und die entsprechenden Lösungen kaum noch Signifikanz für das eigentliche Problem haben.
- Die Evaluierungsfunktion ist verrauscht oder zeitabhängig.
- Das Problem hat derart starke Constraints, dass es schwer ist, überhaupt eine zulässige Lösung zu finden.

Schwierige Probleme: nicht separabel

- Das Problem lässt sich nicht in Teilprobleme zerlegen (nicht separabel).

Ist ein Problem separabel, so lässt es sich in Teilprobleme zerlegen, die unabhängig von einander gelöst werden können.

Wir betrachten dies anhand eines neuen Problems:
"Rekonstruktion geschredderter Akten" (RGA).

Rekonstruktion geschredderter Akten (RGA)

Es liegen die Streifen geschredderter Akten vor, welche rekonstruiert werden sollen. Die Streifen haben sich nach dem Schreddern chaotisch angeordnet. Da wir die Trajektorie des chaotischen Prozesses nicht kennen, wissen wir nicht, wie die Akten zu beginn aussahen.

Gesucht wird die korrekte Anordnung der Streifen (Objective).

RGA kann analog zum TSP repräsentiert werden.

Die Evaluierungsfunktion überlegen wir uns noch nicht ...

Grösse des Suchraums: RGA

Nehmen wir an, jede Seite wird in 20 Streifen geschnitten.

Grösse des Suchraums: Anzahl Permutationen (ohne Wiederholung) der Streifen, also $n \cdot (n - 1) \cdot \dots \cdot 1$, auch als $n!$ geschrieben.

$$\begin{array}{lclclcl} 1 \text{ Seite} & = & 20 \text{ Streifen} & \hat{=} & 20! & = & 2.43 \cdot 10^{18} \\ 5 \text{ Seiten} & = & 100 \text{ Streifen} & \hat{=} & 100! & = & 9.33 \cdot 10^{157} \\ 10 \text{ Seiten} & = & 200 \text{ Streifen} & \hat{=} & 200! & = & \infty \end{array}$$

Separieren: RGA

Obschon TSP und RGA beide $|\mathcal{S}| = n!$ haben, gibt es einen fundamentalen Unterschied:

Bei RGA können wir feststellen, ob jeweils zwei Streifen nebeneinander gehören. Das Problem lässt sich folglich in das (wiederholte) Auffinden von Nachbarn zerlegen.

Beim TSP können wir nicht feststellen, ob jeweils zwei Städte nacheinander besucht werden sollen.

Anders: Eine exakte Evaluierung beim TSP ist nur für ganze Lösungen möglich, bei RGA kann jedes Paar Streifen einzeln exakt evaluiert werden. Dadurch lässt sich das RGA zerlegen.

Separieren: RGA

Wenn wir die Akten rekonstruieren möchten: wir fügen iterativ jeweils einen Streifen hinzu. Für jede Stelle, können wir sofort entscheiden, ob der Streifen, den wir eben testen, da hingehört (wenn nicht, nächsten Streifen testen).

Wie viele Operationen (einen Streifen testen) müssen durchgeführt werden, um die Akten zu rekonstruieren?

P.s: Die Evaluierungsfunktion betrifft hier nur jeweils zwei Streifen → wenn benachbart 1, ansonsten 0.

Separieren: RGA

Erste Stellen: max. n Streifen testen. Zweite Stelle: max. $n - 1$ Streifen testen. Also: $n + (n - 1) + \dots + 1 = (n + 1) \cdot n/2$

$$1 \text{ Seite} = 20 \text{ Streifen} \hat{=} 210$$

$$10 \text{ Seiten} = 200 \text{ Streifen} \hat{=} 20100$$

$$100 \text{ Seiten} = 2000 \text{ Streifen} \hat{=} 2 \cdot 10^6$$

$$1000 \text{ Seiten} = 20000 \text{ Streifen} \hat{=} 2 \cdot 10^8$$

P.s: Komplexität: $O(N^2)$

P.p.s: In Teilprobleme zerlegen: aka "Divide and Conquer" (DaC).

P.p.p.s: Wir lernen: einen Schredder nie ganz leeren!

Schwierige Probleme: Simplifiziertes Modell

- Das Problem ist derart komplex, dass schwer simplifizierte Modelle verwendet werden müssen und die entsprechenden Lösungen kaum noch Signifikanz für das eigentliche Problem haben.

Wir betrachten eine vereinfachte Evaluierungsfunktion anhand eines Logistikproblems.

Logistikproblem

Ein Logistikunternehmen möchte die Transportkosten der Sendungen eines Stuhlproduzenten optimieren. Die Kosten eines Transports von m Stühlen von A nach B sind folgendermassen:

$$F(A, B, m) = \text{Distanz}(A, B) * \text{Handhabung}(m)$$

$\text{Handhabung}(m)$ ist eine Funktion, die die günstigste Verpackungsart in Abhängigkeit der Menge bestimmt.

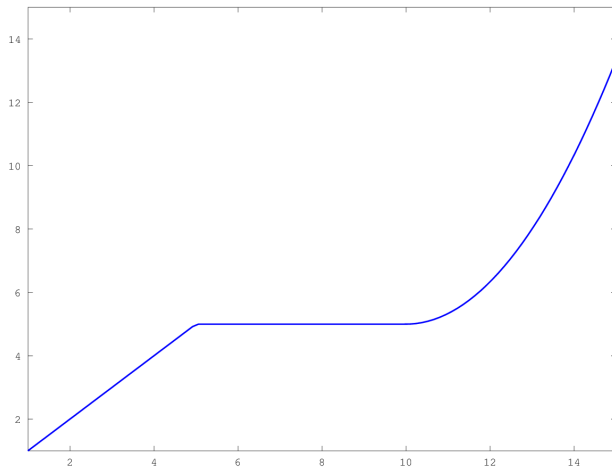
Aus $F(A, B, m)$ soll langfristig die Evaluierungsfunktion einer Lösung x entstehen. Die Lösung steht für den Transport aller produzierten Stühle in einem Monat auf diversen Strecken.

Logistikproblem

Handhabung(m) sieht folgendermassen aus:

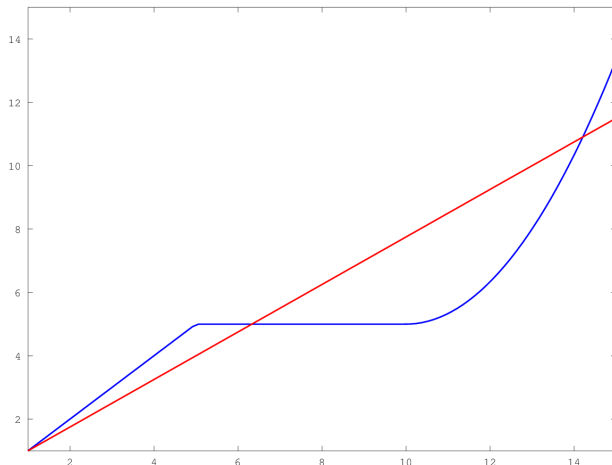
- Bis zu 5 Stühle können lose transportiert werden. Die Kosten steigen linear, da man jeden Stuhl einzeln anfassen und verstauen muss
 $Handhabung_{m < 5}(m) = m$.
- Zwischen 5 und 10 Stühle werden in einer großen Kiste untergebracht, Kosten konstant $Handhabung_{5 \leq m \leq 10}(m) = 5$.
- Über 10 Stühle steigen die Kosten quadratisch an, weil dafür ein Spezialkontainer herhalten muss, der zunehmend (mit m) nervig zu beladen wird $Handhabung_{10 < m}(m) = 10 + (m - 10)^2/3$.

Logistikproblem



Dies sieht nicht sonderlich kompliziert aus, kann jedoch im übergeordneten Logistikproblem zu erheblichen Schwierigkeiten führen.

Logistikproblem



Die lineare Approximation ist für die Optimierung viel leichter zu handhaben, jedoch entspricht die Approximation nicht den realen Zahlen.

Schwierige Probleme

Es gibt viele Gründe, warum Probleme schwierig zu lösen sein können. Die in unseren Betrachtungen wichtigsten Gründe sind (i.d.R. sind mehrere Gründe zugleich notwendig):

- Der Suchraum ist derart riesig, dass er nicht erschöpfend durchsucht werden kann (fortan "exhaustive search" genannt).
- Die Fitnesslandschaft hat keine schöne Form, ist z.B. Schroff.
- Das Problem lässt sich nicht in Teilprobleme zerlegen (nicht separabel).
- Das Problem ist derart komplex, dass schwer simplifizierte Modelle verwendet werden müssen und die entsprechenden Lösungen kaum noch Signifikanz für das eigentliche Problem haben.
- Die Evaluierungsfunktion ist verrauscht oder zeitabhängig.
- Das Problem hat derart starke Constraints, dass es schwer ist, überhaupt eine zulässige Lösung zu finden.

Ausblick

Die wichtigsten nächsten Schritte

- Constraints
- Distanz
- Nachbarschaft