7. Optimierung von Simulationsmodellen

Ziel vieler Simulationsmodelle ist das finden einer guten/optimalen Lösung

Allgemein formuliert:

Min f(**x**) unter den Nebenbedingungen $\mathbf{x} \in W$

oft sind die Nebenbedingungen durch Gleichungen/Ungleichungen spezifiziert, also z.B. $g_i(\mathbf{x}) \geq 0$ (i=1,2,…,m)

f(**x**) wird per Simulation ermittelt

Dies ist ein nichtlineares Optimierungsproblem!

Was macht die Optimierung von Simulationsmodellen schwierig?

Unterscheidung diskrete stochastische und kontinuierliche deterministische Simulation notwendig!

Eigenschaften kontinuierlicher Modelle:

- Berechnungsdauer einer Auswertung im Sekundenbereich
- Deterministische Resultate
- Gradienten numerisch relativ genau bestimmbar

Eigenschaften diskreter Modelle:

- Berechnungsdauer einer Auswertung im Minuten- bis Stundenbereich
- Stochastische Resultate
- Gradienten können nur geschätzt werden

Hybride Modelle mit stochastischen Anteilen verhalten sich ähnlich wie diskrete Modelle

Wir betrachten im Folgenden die Optimierung diskreter stochastischer Modelle!

Ziele des Kapitels:

- Einordnen der Probleme, die bei der Optimierung stochastischer Simulationsmodelle auftreten
- Kennen lernen von unterschiedlichen Optimierungsansätzen
- Einordnung der Optimierung in den Prozess der modellbasierten Systemanalyse
- Erkennen von simulationsspezifischen Problemen und Chancen bei der Optimierung

Gliederung des Kapitel

7.1 Optimierungsansätze für Simulationsmodelle

7.2 Gradientenbestimmung und Sensitivitätsanalyse

7.3 Nutzung von Polynomen als Metamodelle

7.4 Stochastische und heuristische Optimierung

7.5 Optimierungs-Software

Literatur:

- Law/Kelton Kap. 12.5, 12.6
- Banks et al. Kap. 12.4
- R. H. Myers, D. C. Montgomery. Response Surface Methodology
- Z. Michalewicz, D. B. Fogel. How to Solve It: Modern Metaheuristics. Springer 2004.

7.1 Optimierungsansätze für Simulationsmodelle

Grundsätzliche Probleme:

- Viele Faktoren aber meistens recht einfache Nebenbedingungen
- Mehrere Resultatvariablen
  z.B. Kosten versus Durchlaufzeiten versus Auslastungen
  wir gehen hier von einer Resultatvariablen aus, die z.B. durch eine Linearkombination der Resultate bestimmt werden kann
- Stochastische Beobachtungen erlauben keine exakten Aussagen bzgl. des Vergleichs von Konfigurationen oder bzgl. der Gradienten

Anforderungen an Optimierungsverfahren für Simulationsmodelle:

- Garantierte Wahrscheinlichkeit einer korrekten Auswahl unter Vorgabe einer Konfidenzintervallbreite
  d.h. für den mit dem Algorithmus gewonnenen Schätzer des Optimums $\tilde{\mathbf{x}}$ gilt
  $P(E(f(\tilde{\mathbf{x}})) + \varepsilon > E(f(\mathbf{y})) \mid \forall \mathbf{y} \in W) > \alpha$ globale Optimierung
  $P(E(f(\tilde{\mathbf{x}})) + \varepsilon > E(f(\mathbf{y})) \mid \forall \mathbf{y} \in N(\tilde{\mathbf{x}})) > \alpha$ lokale Optimierung
- Garantierte asymptotische Konvergenz gegen das globale/lokale Optimum, wenn die Simulationszeit jedes Experiments oder die Zahl der Experimente gegen unendlich geht
- Garantierte Konvergenz, wenn davon ausgegangen würde, dass jede Simulation ein deterministisches Ergebnis liefert
- Robustheit der Lösung. d.h. es wird mit hoher Wahrscheinlichkeit eine gute Lösung in akzeptabler Zeit gefunden

Verwendete Optimierungsmethoden:

- Nutzung von Standardoptimierungsalgorithmen
  Je nach Problemstellung existieren unterschiedliche Verfahren
  Einbeziehung von Nebenbedingungen oft durch Änderung der Zielfunktion
  Gradienten werden entweder direkt benötigt oder implizit durch mehrfache Auswertung der Funktion approximiert
- Metamodellbasierte Verfahren
  Simulationsmodell wird durch ein Metamodell approximiert, das Metamodell wird optimiert, anschließend wird im Bereich des Optimums ein neues Metamodell angepasst und damit ein neues Optimum bestimmt (iteratives Vorgehen siehe 7.3)
- Heuristische Suchverfahren
  oft zufallsgesteuerte Heuristiken und Metaheuristiken
  z.B. Random Search, Local Search, evolutionäre Algorithmen,…

7.2 Gradientenbestimmung und Sensitivitätsanalyse

- Basis vieler Optimierungsmethoden ist die Kenntnis der Gradienten
- Gradienten sind auch bedeutsam, um den Einfluss der Faktoren auf des Ergebnis und die Stabilität einer Lösung abzuschätzen
- Sei $df(\mathbf{x})/dx_i$ der partielle Ableitung von f an der Stelle x bzgl. der i-ten Komponente
  - falls der Wert klein ist, so beeinflusst der Parameter den Wert von f nur geringfügig
  - große Werte zeigen eine starke Beeinflussung an
  - zur Optimierung zeigt der Gradient die Richtung der Vergrößerung/Verkleinerung des Funktionswertes

Es existiert keine vollständig befriedigende Methode zur Bestimmung von Gradienten, wenn keine Information über die Verteilung von f(**x**) bekannt ist

Verwendet werden

1. Abschätzung der Gradienten aus Differenzenquotienten
2. Abschätzung der Gradienten aus Metamodellen
3. Pertubationsanalyse zur Bestimmung von Gradienten aus einzelnen Simulationsläufen

- 1. und 2. sind für beliebige Simulationsmodelle ohne Eingriff in den Simulator anwendbar
  (es müssen aber u. U. viele Experimente durchgeführt werden)
- 3. stellt gewisse Bedingungen an den Verlauf von f(x) und erfordert Eingriffe in den Simulator
  (kann aber sehr effizient und genau sein)

Abschätzung aus dem Differenzenquotienten:

$$\frac{df(\mathbf{x})}{dx_i} = \lim_{\delta \to 0} \frac{f(\mathbf{x}+\mathbf{e}_i\delta)-f(\mathbf{x})}{\delta} \quad \Rightarrow$$

$$\frac{df(\mathbf{x})}{dx_i} \approx \frac{f(\mathbf{x}+\mathbf{e}_i\delta)-f(\mathbf{x})}{\delta} \qquad \text{für } \delta > 0$$

Kleines δ liefert genauere Approximation der partiellen Ableitung, ist aber auch weniger robust!

Situation in der stochastischen Simulation:

- f(**x**) und f(**x**+$\mathbf{e}_i\delta$) können nur geschätzt werden
  - damit sind nur Mittelwertschätzer und Konfidenzintervalle bekannt
  - Für kleine Werte von δ unterscheiden sich die Ergebnisse kaum, dadurch ist nicht einmal das Vorzeichen des Gradienten bestimmbar

Simuliere an der Stelle **x** und an allen Stellen $\mathbf{x}+\mathbf{e}_i\delta$ (i=1,…,k) jeweils n unabhängige Replikationen oder so viele Replikationen, bis das Konfidenzintervall eine Breite < ε hat

Sei ε_i die Breite des Konfidenzintervalls für $f(\mathbf{x}+\mathbf{e}_i\delta)$ und ε_0 die Breite des Konfidenzintervalls von $f(\mathbf{x})$

Setze

$$\frac{f(\mathbf{x})}{dx_i} = \begin{cases} \frac{\hat{f}(\mathbf{x}+\mathbf{e}_i\delta)-\hat{f}(\mathbf{x})}{\delta} & \text{if } |\hat{f}(\mathbf{x}+\mathbf{e}_i\delta)-\hat{f}(\mathbf{x})| > \epsilon_0+\epsilon_i \\ 0 & \text{sonst} \end{cases}$$

Alternative: Simuliere mit gemeinsamen Zufallszahlen und untersuche, ob $w = f(\mathbf{x}+\mathbf{e}_i\delta) - f(\mathbf{x}) < 0$ oder > 0 mit den Methoden auf Kap. 5 (Vorsicht, die einzelnen Vergleiche sind bei Verwendung gleicher ZZs nicht unabhängig!)

Schätzung der partiellen Ableitungen aus Metamodellen

Bestimme ein lineares Regressionsmodelle, das Hauptfaktoren und ausgewählte Nebenfaktoren berücksichtigt (siehe Kap. 6.5):

$$\text{z.B. } y = \beta_0 + \sum_{i=1}^{k} \beta_i x_i + \sum_{i=1}^{k} \sum_{j=i}^{k} \beta_{i,j} x_i x_j$$

- Nur Variablen im Modelle verwenden, deren Koeffizienten signifikant von 0 verschieden sind
- Ableitung bzgl. x_i leicht bestimmbar
- Qualität der Ableitungen kann durch Konfidenzintervalle der β_i (die wir nicht behandelt haben) abgeschätzt werden

Vor- und Nachteil der vorgestellten Ansätze:

- Methoden sind einfach durchführbar und erfordert keine Eingriffe in die Simulationssoftware
- Konfidenzintervalle geben zumindest einen groben Hinweis auf die Qualität der ermittelten Ableitungen
- Der ermittelte Gradient kann in „Standardoptimieurngsverfahren“ verwendet werden

Aber

- Der Aufwand ist sehr hoch, da viele Experimente durchgeführt werden müssen
- Fehler in den partiellen Ableitungen können die Optimierungsverfahren in die Irre führen

Eine effizientere, aber auch komplexere Methode der Bestimmung von Ableitungen ist die Pertubationsanalyse
(hier nur grundsätzliche Idee, für Details siehe Originalliteratur):

- Sensitivität bzgl. eines oder mehrerer Parameter soll aus einem Beispiellauf gewonnen werden
- Dazu wird während der Simulation die ursprünglich Trajektorie und die Trajektorie bestimmt, die entstehen würde, wenn der Parameter x_i um einen kleinen Wert ε_i verändert würde
  - Z.B. bei einer exponentiell verteilten Zeit mit Rate λ würde aus der ZZ $-\lambda\ln(u)$ die „gestörte“ ZZ $-(\lambda+\varepsilon)\ln(u)$
  - Während der Simulation müssen zur Bestimmung von k Sensitivitäten k+1 Abläufe nachgebildet werden
    (Eingriffe in den Simulator sind notwendig!)
  - Probleme treten bei Diskontinuitäten in den Trajetorien auf

7.3 Nutzung von Polynomen als Metamodelle

- Zu Grunde liegende Methode wird als Response Surface Methode bezeichnet und hat eine lange Historie in der experimentbasierten Optimierung
  (siehe Buch von Myers, Montgomery)
- Basis der Optimierung sind Polynome, wie in 6.5
  - Nur signifikante Faktoren berücksichtigen
  - Optimierung besteht aus dem iterativen Einsatz von Simulations- und Metamodellergebnissen
  - Zuerst grobe Optimierung durch Modelle erster Ordnung (d.h. nur die Haupteffekte werden berücksichtigt)
  - Zur Verfeinerung Modelle zweiter Ordnung verwenden (d.h. Haupt- und Nebeneffekte der Ordnung 2 berücksichtigen)

Skizze des Vorgehens:

1. Bestimme einen initialen Bereich $[l_i,u_i]$ für jeden Parameter
2. Normiere die Parameterwerte auf [-1,+1],
   d.h. aus $w_i \in [l_i,u_i]$ wird $x_i = (2w_i - (u_i + l_i)) / (u_i - l_i)$
3. Bestimme ein lineares Regressionsmodell, in dem nur die k Haupteffekte berücksichtigt werden
   a. Wähle dazu ein Experimentdesign mit m > k Designpunkten, die so gewählt werden, dass die Spalten der Matrix **X** orthogonal sind, d.h. $\Sigma_{h=1,\ldots,m}\mathbf{X}(h,i)\cdot\mathbf{X}(h,j) = 0$ für $i \neq j$
   und die Werte der Designvariablen -1 oder +1 sind
   Orthogonale Designs minimieren die Varianz der Regression
   b. Führe u.U. Replikationen an jedem Designpunkt durch und ersetze die Resultatwerte durch die Mittelwerte oder berücksichtige die Replikationen im Design

Beispiel für ein orthogonales Design mit 4 Experimenten und 3 Variablen:

$$\mathbf{X} = \begin{pmatrix} & x_1 & x_2 & x_3 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \quad \text{und } \mathbf{X}^T\mathbf{X} = 4 \cdot \mathbf{I}$$

Die Einführung von Replikationen im Design (d.h. Wiederholungen von Zeilen) sollte so erfolgen, dass die Orthogonalität erhalten bleibt, z.B.

$$\mathbf{X} = \begin{pmatrix} & x_1 & x_2 & x_3 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 \end{pmatrix} \quad \text{und } \mathbf{X}^T\mathbf{X} = 8 \cdot \mathbf{I}$$

4. Test, ob die Regression signifikant ist
   (siehe z.B. Kap. 6.5, weitere Alternativen existieren)
   a. Falls nein, verkleinere die Intervalle $[l_i,u_i]$ und
      i. Falls $u_i-l_i > \omega_i$ für alle i, fahre bei 2. fort
      ii. Ansonsten erhöhe die Anzahl der Variablen durch Berücksichtigung von Nebeneffekten der Ordnung 2 und fahre bei 6. fort
   b. Falls ja, gehe zu Schritt 5.
5. Führe eine schrittweise Verbesserung durch:
   Richtung des steilsten Abstiegs/Anstiegs ist durch die Werte der Koeffizienten b_i gegeben
   a. Setze $\Delta = 1 / \max_i |b_i|$ als Schrittweite des Abstiegs

b. Vorgehen für die Maximierung (Minimierung analog dazu)

m = 1 ;

repeat

simuliere das Modelle am Punkt ($m\Delta b_1,\ldots,m\Delta b_k$) ;

m := m+1 ;

until keine Vergrößerung des Funktionswertes mehr beobachtet wird ;

i. Falls m > 1 setzte

m := m – 1;

wähle ($m\Delta b_1,\ldots,m\Delta b_k$) als neuen normierten Mittelpunkt

(unnormierter Wert von Var. i: $(u_i+l_i)+ m\Delta b_i(u_i-l_i)$)

ii. Ansonsten fahre bei 4.a fort

6. Optimierung mit Hilfe des Regressionsmodells der Ordnung 2
   D.h. alle Haupteffekte und Nebeneffekte der Ordnung 2 werden berücksichtigt

Sei

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{pmatrix},$$

$$\mathbf{B} = \begin{pmatrix} b_{1,1} & b_{1,2}/2 & \cdots & b_{1,k-1}/2 & b_{1,k}/2 \\ b_{2,1}/2 & b_{2,2} & \cdots & b_{2,k-1}/2 & b_{2,k}/2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ b_{k-1,1}/2 & b_{k-1,2}/2 & \cdots & b_{k-1,k-1}/2 & b_{k-1,k}/2 \\ b_{k,1}/2 & b_{k,2}/2 & \cdots & b_{k,k-1}/2 & b_{k,k}/2 \end{pmatrix}$$

Matrix B ist symmetrisch, da $b_{i,j} = b_{j,i}$

Stationärer Punkt des Regressionsmodells $y = b_0 + \mathbf{x}^T\mathbf{b} + \mathbf{x}^T\mathbf{B}\mathbf{x}$:

$\mathbf{x}_s = -0.5\mathbf{B}^{-1}\mathbf{b}$

Falls

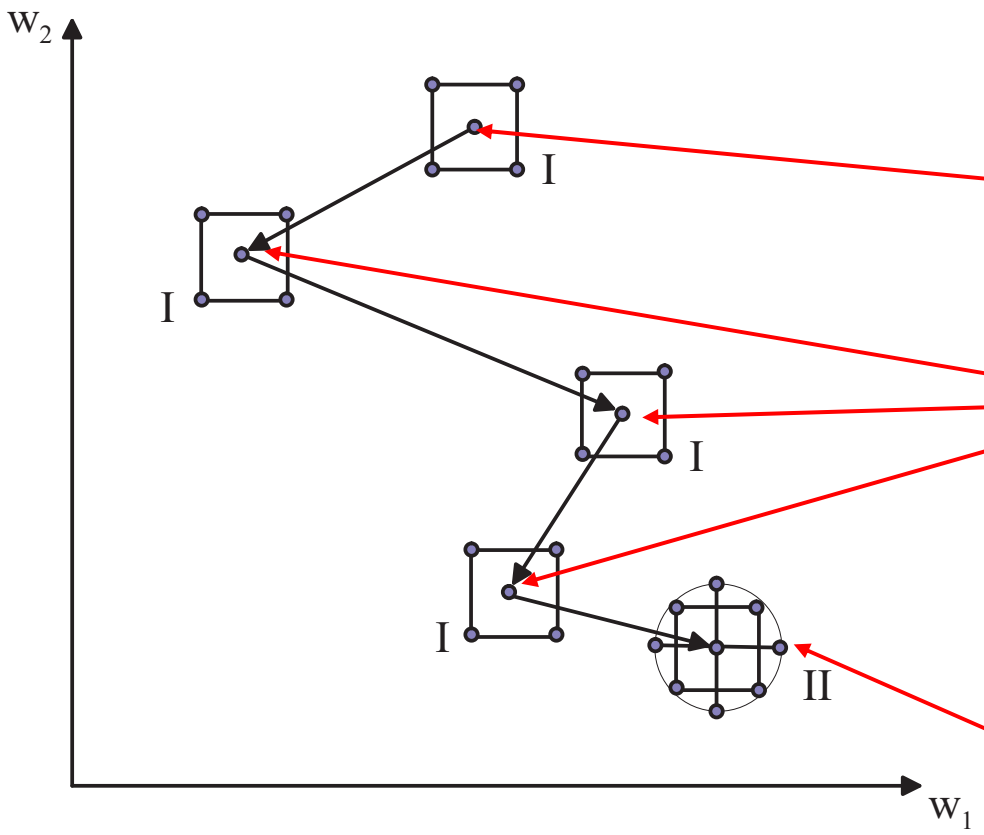
- Falls alle Eigenwerte von **B** negativ, dann ist $\mathbf{x}_s$ ein Maximum
- Falls alle Eigenwerte von **B** positiv, dann ist $\mathbf{x}_s$ ein Minimum
- Ansonsten ist $\mathbf{x}_s$ ein Sattelpunkt

Falls $\mathbf{x}_s$ vom richtigen Typ ist, so analysiere das Modell am Punkt $\mathbf{x}_s$ und gib den erreichten „optimalen“ Punkt aus

Falls $\mathbf{x}_s$ nicht vom richtigen Typ ist, so

- gib entweder den besten Wert aus, der bisher bei einer Simulation erreicht wurde oder
- nutze ein anderes Verfahren zu Optimierung im Bereich $[l_1,u_1],\ldots,[l_k,u_k]$

Skizze des Vorgehens



Startpunkt

Schrittweise Verbesserung

Endgültige Bestimmung des „optimalen“ Punktes mittels eines Modells der zweiten Ordnung

Einige Bemerkungen:

- Die vorgestellte Version muss für die praktische Anwendung noch verfeinert werden
- Problematisch ist insbesondere die automatische Anwendung, bei der alle Entscheidungen algorithmisch getroffen werden
- Parameter müssen reellwertig sein oder sich durch reellwertige Variablen approximieren lassen
- Nebenbedingungen für Parameter können bei der Festlegung neuer Intervallgrenzen berücksichtigt werden
- Variabilität der Simulationsresultate wird nur implizit durch die Signifikanzüberprüfung der Regression berücksichtigt
- Verfahren findet im besten Fall lokale Optima
  Versuch der Abdeckung des Designraums durch Start der Regression an unterschiedlichen Stellen

7.5 Stochastische und heuristische Optimierung

- Bisher vorgestellte Optimierungsansätze stellen Anforderungen an das Verhalten von Simulationsmodellen (z.B. durch eine Regression approximierbar, Gradient bestimmbar),
- können durch statistische Schwankungen in den Beobachtungen in die Irre geführt werden,
- sind nur eingeschränkt automatisierbar,
- finden nur lokale Optima,
- können nicht/kaum mit diskreten oder qualitativen Parametern arbeiten

➢ Es werden robuste und automatisierbare Ansätze benötigt, die die auszuwertende Funktion als „black box“ betrachten

Kandidaten sind heuristsiche Suchverfahren, wie z.B.

- Zufallssuche
- Lokale Suchverfahren, wie
  - Simulated Annealing, Sinflutalgorithmus, …
- Pattern Search
- Tabu Search
- Ameisenkolonieoptimierung
- Genetische Algorithmen
- Evolutionäre Algorithmen
- u.v.a.

➢ Auch diese Algorithmen wurden in der Regel für deterministische Funktionen entwickelt, lassen sich aber gut an die Simulation anpassen

Beispiele evolutionäre Algorithmen und Tabu-Suche:

- In beiden Fällen muss das Basisverfahren an die speziellen Erfordernisse der Simulation angepasst werden
  - Berücksichtigung der stochastischen Schwankungen durch feste oder variable Anzahl von Replikationen
  - Berücksichtigung der u.U. langen Simulationslaufzeiten

Bisherige Realisierung basiert im Wesentlichen auf den Standardalgorithmen, ohne direkte Simulationsanpassung, dann oft ineffizient

Neuere Arbeiten berücksichtigen Simulationsspezifika (aktuelles Forschungsgebiet)

Beispiel Evolutionsstrategien

Individuum (**x**, **s**)

- **x** Parameter
- **s** Strategieparameter

Mutation (hier für kont. Parameter):

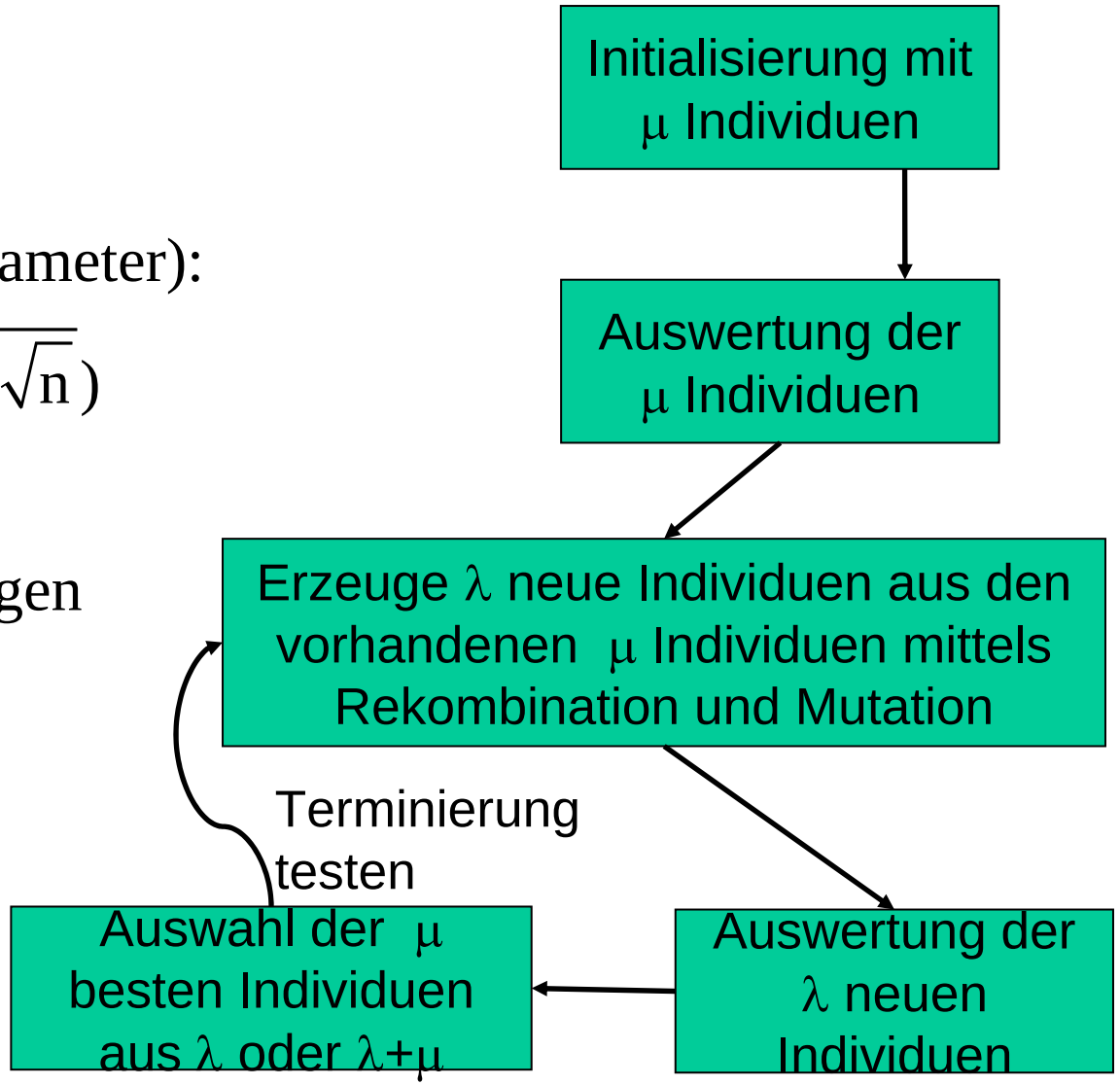
$$s_i = s_i \cdot \exp(u/\sqrt{2n} + u_i/\sqrt{2\sqrt{n}})$$

$$x_i = x_i + s_i \cdot u'_i$$

u, u_i, u'_i aus N(0,1) gezogen

für diskrete Parameter modifizierbar!

Rekombination von zwei Individuen durch Kombination der Vektoren **x** und **s**

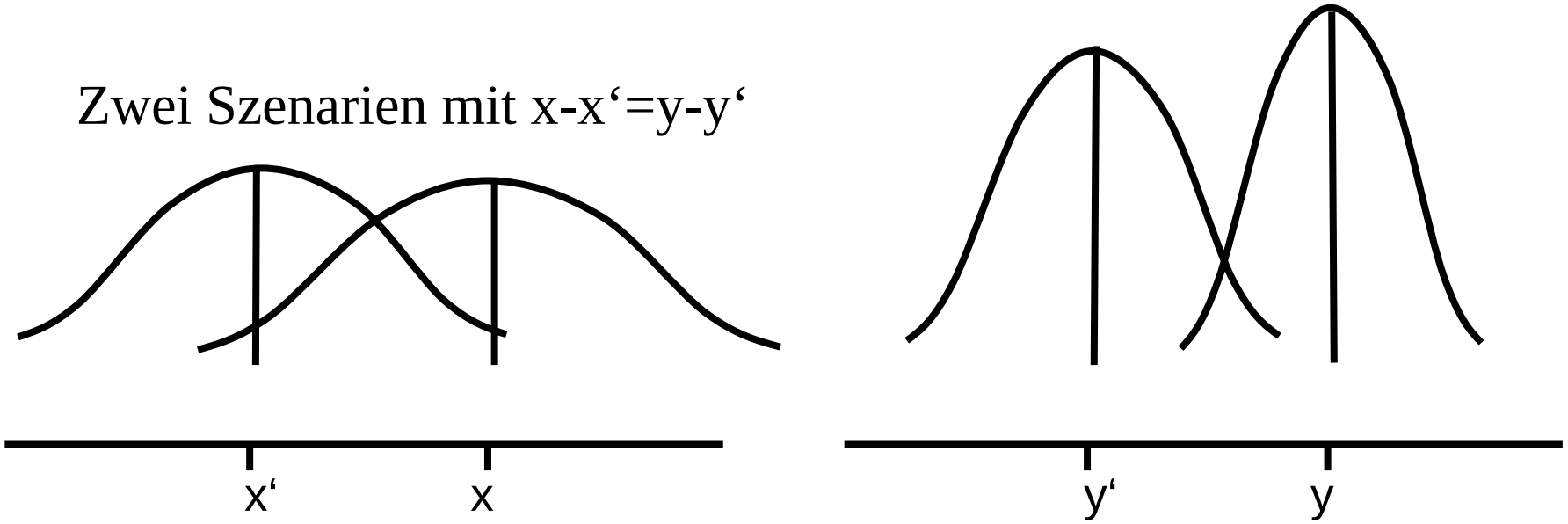


Besonderheit bei stochastischen Modellen: Auswertung von f(**x**)

Wie viele Beobachtungen sind notwendig?

Einfachste Lösung:
Ersetze f(x) durch den Mittelwertschätzer bei vorgegebener Beobachtungszahl

Zwei Szenarien mit x-x‘=y-y‘



y‘<y ist sehr viel wahrscheinlicher als x‘<x, auch wenn der Abstand der Mittelwerte identisch ist

Ziel: Auswahl der besten m Individuen aus l oder m+l Individuen

Zwangsläufige Einschränkung bei stochastischen Systemen

Algorithmus findet die besten Individuen

- mit vorgegebener Wahrscheinlichkeit p,
- wenn sich die Erwartungswerte um mindesten d unterscheiden

Algorithmus sollte berücksichtigen, dass

- für Individuen bereits (unterschiedliche Anzahlen von) Beobachtungen vorhanden sein können
- offensichtlich schlechte oder gute Parameterwerte nur wenige Beobachtungen benötigen

vorhandene Verfahren (ranking and selection, siehe 5.3) berücksichtigen diese Aspekte nicht, deshalb Entwicklung neuer (heuristischer) Algorithmen

Tabu-Suche:

1. Initialisiere den Parametervektor x (z.B. zufällig)
2. Finde in der Nachbarschaft von **x** einen Vektor **x'**, so dass f(**x'**) > f(**x**) (bei Maximierung)
   a. Falls kein **x'** existiert, Ausgabe von x
   b. Ansonsten setze **x** = **x'** und fahre bei 3. fort
3. Speichere alle in Schritt 2 untersuchten Punkte in der Tabuliste und entferne u.U. alte Punkte aus der Tabuliste, fahre bei 2 fort.

Zwei zentrale Aspekte

- Definition der Nachbarschaft durch alle Punkte **y** mit $\|\mathbf{y}-\mathbf{x}\|<\varepsilon_1$ und es existiert kein **z** in Tabuliste mit $\|\mathbf{y}-\mathbf{z}\|<\varepsilon_2$ ($\varepsilon_2 < \varepsilon_1$)
- Finden eines besseren Punktes in der Nachbarschaft durch Zufallssuche, Gittersuche etc.

Viele weitere Ansätze existieren!

Insgesamt ein weites Feld mit

- vielen heuristischen Untersuchungen,
- wenig Systematik
- wenig beweisbaren Resultaten
- aber pragmatischen Lösungen, die in der Praxis mehr oder weniger gut funktionieren

➢ Bedarf an weiteren theoretischen und empirischen Ergebnissen

….

7.5 Optimierungs-Software

Software-Packet	Simulatoren	Algorithmen
AutoStat	AutoMod, AutoShed	Evolutionsstrategien
OptQuest	Arena, Taylor, QUEST	Scatter Search, Tabu Search, Neuronale Netze
OPTMIZ	SIMUL8	Neuronale Netze
SimRunner2	ProModel, ServiceModel	Evolutionsstrategien, Genetische Algorithmen
WITNESS Optmizer	WITNESS	Simulated Annealing, Tabu Search
OPEDo	ProC/B, APNN-Toolbox	RSM, Evolutionsstrategien, Pattern Search