Algorithmen für Chip-Entwurfswerkzeuge

Metriken und Simulated Annealing (VPR und DAST)

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorlesung
WS 2013/2014

Florian Stock, Andreas Koch

Eingebette Systeme und Anwendungen
Technische Universität Darmstadt

Embedded Systems & Applications

Optimierungsziele

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gesamtverbindungslänge
  - Optimiert indirekt auch andere Ziele
- Verdrahtungsdichte / Verdrahtbarkeit
- Signalverzögerung
- Energieverbrauch
- Thermisches Verhalten

$\Rightarrow$ Zielfunktion (objective function) entsprechend wählen

Am gebräuchlichsten:

(Gewichtete) Gesamtverbindungslänge

Gesamtverbindungslänge
Exakte Kosten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Optimal: Exaktes (und perfektes) Routing während der Platzierung
- Exaktes Routing zu aufwendig

$\Rightarrow$ Verbindungslänge für jedes Netz schätzen und aufsummieren

  - Annahme optimales Routing möglich (d.h. keine Umwege)
  - Für bis 3 Pins trivial: HPWL Abschätzung exakt
  - Für (viel) mehr: HPWL Abschätzung zunehmend ungenau
  - Andere Möglichkeiten RSMT, RMST, korrigierte HPWL, FLUTE, quadratische Verbindungslänge, LSE, WA, Star+, ...

Metrik

TECHNISCHE UNIVERSITÄT DARMSTADT

- Abstand $d(x, y)$ zweier Punkte (Pins) im Raum (in der Ebene)
- Definit: $x = y \Leftrightarrow d(x, y) = 0$
- Symmetrisch: $d(x, y) = d(y, x)$
- Erfüllt Dreiecksungleichung: $d(x, y) \leq d(x, z) + d(z, y)$
- Üblicherweise benutzt man von Norm induzierte Metrik

$$d(x, y) := ||x - y||$$

- Wie handhabt man Netze mit mehr als 2 Pins?

1-Norm
Manhattan-/Taxi-Metrik

TECHNISCHE UNIVERSITÄT DARMSTADT

- *p*-Norm induzierte Metrik

$$d(x, y) = \sqrt[p]{\sum_i |x_i - y_i|^p}$$

- 1-Norm gibt Abstand wenn man sich rechtwinklig bewegt
- Gibt exakte Länge in einem Raster
- Sehr schnell zu berechnen

$\Rightarrow$ Sehr weit verbreitet

HPWL
Half-Perimeter-Wirelength

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verallgemeinerung der 1-Norm für mehr als 2 Pins
- Halber Umfang des umschliessenden Rechtecks aller Pins
- Exakt für bis zu 3 Pins
- Untere Grenze für $>$ 3 Pins

Vorteile

- Sehr schnell zu berechnen:

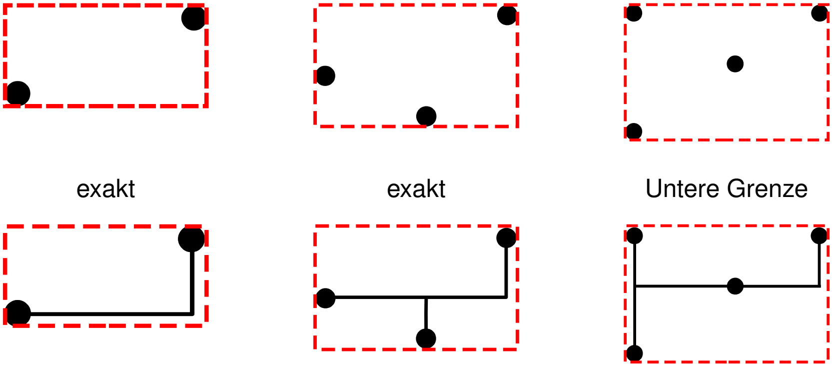
$$\begin{aligned} HPWL = &(\max x_i - \min x_i) + \\ &(\max y_i - \min y_i) \end{aligned}$$

Nachteile

- Für große Anzahl Pins wird der schnell Fehler groß
  $\Rightarrow$ Korrekturfaktoren für große Netze
- Nicht differenzierbar

HPWL
Beispiele

TECHNISCHE UNIVERSITÄT DARMSTADT



RSMT
Rectiliniear Steiner Minimal Tree

- Rechtwinkliger Minimaler Steiner Baum, d.h.
  Steinerbaum mit rechtwinkliger Metrik
- Steinerbaumproblem: Finde kleinsten geometrisch aufspannenden Baum eines Graph (Verzweigungen nicht zwangsweise an Knoten)

Vorteile

- Exakt

Nachteile

- NP-Vollständig $\Rightarrow$ Impraktikabel
- Normalerweise sogar Heuristiken lange Laufzeit
- NEU: FLUTE
  Tabellenbasierte RSMT-Heuristik
- Nicht differenzierbar

RMST
Rectilinear Minimum Spanning Tree

- Rechtwinkliger minimal aufspannender Baum

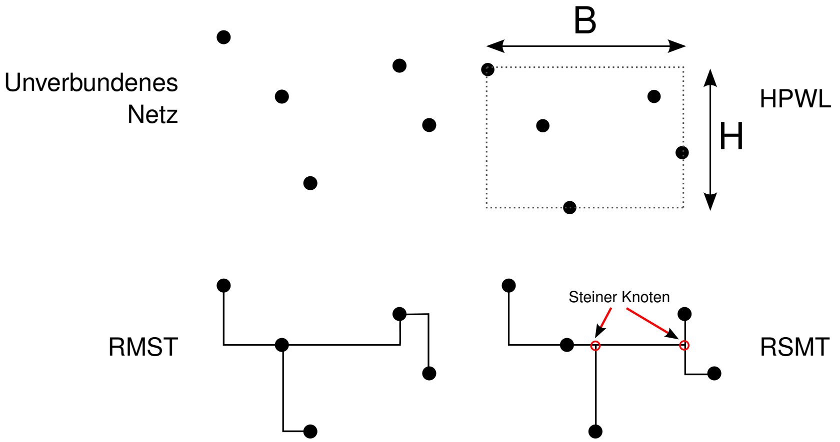
Vorteile

- Viel schneller als RSMT: $\mathcal{O}(n \log n)$
- Gütegarantie: Ergebnis max. 50% schlechter

Nachteile

- Schlechter als RMST
- Vielfaches langsamer als HPWL
- Nicht differenzierbar

Vergleich
HPWL – RMST – RSMT



Nichtlineare Metriken
Quadratische Verdrahtungslänge

TECHNISCHE UNIVERSITÄT DARMSTADT

- 2-Norm basiert, allerdings ohne Wurzel und mit Faktor

$$QWL = \frac{1}{2}((x_i - x_j)^2 + (y_i - y_j)^2)$$

- Da oft mit Ableitung gearbeitet wird: Faktor von $\frac{1}{2}$

Vorteile

- Differenzierbar
- Konvex

Nachteile

- Weicht von linearer ($p = 1$) Norm ab $\Rightarrow$ Funktion linearisieren (z.B. Star+)

Nichtlineare Metriken
Log-Summe-Exp-Verdrahtungslänge (LSE)

- Logarithmus-Summe-Exponentialfunktion:

$$LSE(z_i) = \alpha(\log(\sum_i e^{\frac{z_i}{\alpha}}))$$

- Approximation der max-Funktion
- α Genauigkeit der Approximation
- Für $\alpha \to 0$: LSE konvergiert gegen max
- Approximiert somit HPWL

$$\alpha \times ((\log(\sum_i e^{\frac{x_i}{\alpha}})) + (\log(\sum_i e^{\frac{-x_i}{\alpha}})) + (\log(\sum_i e^{\frac{y_i}{\alpha}})) + (\log(\sum_i e^{\frac{-y_i}{\alpha}}))$$

- Differenzierbar und konvex

Netze mit mehr als 2 Pins

- *Auflösen* des Multi-Pin-Netzes in mehrere 2-Pin-Netze

  Cliquen Modell Netz wird durch vollständigen Teilgraphen (Clique) ersetzt

  Stern Modell Netz wird durch zusätzlichen Knoten und 2-Pin-Netze zu diesen Stern-Knoten ersetzt

Verbesserung der Kostenfunktion
Gewichtete Netzkosten

- Bisher: Jedes Netz gleich behandelt
- Gesamtkosten = $\sum$ Netzkosten
- Verbesserung: Gewichtung einzelner Teilnetze bzgl. bestimmter (anderer) Optimierungsziele
- Nun: Gesamtkosten = $\sum c_i(...) \cdot$ Netzkosten
- Gewicht für jedes Netz muß seperat bestimmt/berechnet werden

VPR

- Versatile Place and Route
  - Betz und Marquardt, University of Toronto
  - Ab hier Auszüge aus Paper (auf Web-Seite)
- Platzierer
  - Simulated Annealing-basiert
    - Mit adaptivem cooling schedule
  - Optimiert gleichzeitig
    - Leitungslänge
    - Verzögerung

VPR
Züge

- Paarweises Austauschen von Blöcken
  - N_{blocks} = Größe der Schaltung
- Aber nicht ganz zufällig:
  Beschränkung der Entfernung

VPR
Starttemperatur

- Wird automatisch bestimmt
  passend für aktuelle Schaltung
- Idee:
  - Anfangs fast alle Züge akzeptieren
  - Wie hoch muß die Starttemperatur sein?
- Vorgehen:
  - N_{blocks} Blöcke paarweise austauschen
  - Beobachte Änderung der Kostenfunktion c (Standardabweichung)

$$s_c = \sqrt{\frac{1}{n-1}((\sum_i c_i^2) - n\bar{c}^2)}$$

  - Starttemperatur = $20 \cdot s_c$

VPR
Thermisches Gleichgewicht

TECHNISCHE UNIVERSITÄT DARMSTADT

- Anzahl von Schritten pro Temperaturstufe:

$$10 \cdot N_{blocks}^{\frac{4}{3}}$$

- $10\times$ schneller, aber nur ca. 10% schlechter:

$$N_{blocks}^{\frac{4}{3}}$$

VPR
Abkühlen

TECHNISCHE UNIVERSITÄT DARMSTADT

Beobachtung:

- Anfangs: T hoch, fast alle Züge akzeptiert
  - Im wesentlichen zufälliges Bewegen
  - Keine echte Verbesserung der Kosten
- Ende: T niedrig, kaum Züge akzeptiert
  - Fast keine Bewegung mehr
  - Wenig Veränderung in Kosten

Idee:

- Meiste Optimierung passiert zwischen Anfangs- und Endphase
- Bringe T schnell in den produktiven Bereich
- Halte T möglichst lange im produktiven Bereich

Vorgehen:

- Steuere T anhand der Akzeptanzrate R_a
  Akzeptanzrate R_a: Anteil der Züge die akzeptiert wurde
  (egal, ob verbesserend oder verschlechternd)

- Cooling Schedule $T_{new} = \alpha T_{old}$

α	Acceptance Rate R_a
0.50	$R_a > 0.96$
0.90	$0.80 < R_a \leq 0.96$
0.95	$0.15 < R_a \leq 0.80$
0.80	$R_a \leq 0.15$

VPR
Abkühlen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Vorahnung
  - Gute Fortschritte bei $R_a \approx 0.5$
- Am effizientesten $R_a = 0.44$
  Beste Fortschritte
- Idee
  - R_a möglichst auf diesem Wert halten, aber wie?
  - *Nicht* temperaturbasiert (kühle nur ab!)
  - Sondern: *Auswirkungen* der Züge beeinflussen
  - Beobachtung:
    - Weite Züge: Große Änderung der Kosten
    - Kurze Züge: Kleine Änderung der Kosten
- Vorgehen:
  - Variiere Zugweite R_{limit}, um $R_a \approx 0.44$ zu halten

VPR
Abkühlen

TECHNISCHE UNIVERSITÄT DARMSTADT

R_{limit} klein

- Kleine Zugreichweite
- Kleine Änderungen der Kosten
- Kleine Verschlechterungen (Werden eher angenommen)
- R_a steigt

R_{limit} groß

- Große Zugreichweite
- Große Änderungen der Kosten
- Große Verschlechterungen (Werden eher abgelehnt)
- R_a sinkt

VPR
Abkühlen

TECHNISCHE UNIVERSITÄT DARMSTADT

- Anfangs:

$$R_{limit} = \text{ ganzer Chip } L_{Chip}$$

- Bei jedem Abkühlschritt:

$$R_{limit}^{new} = R_{limit}^{old}(1 + R_a^{old} - 0.44) \qquad \text{mit } 1 \leq R_{limit}^{new} \leq L_{Chip}$$

  - Zuviel akzeptiert: R_{limit} größer machen
  - Zuwenig akzeptiert: R_{limit} kleiner machen

VPR
Abbruchbedingung

- Wann Abkühlung beenden?
- Idee:
  - Stillstand erkennen
- Vorgehen:
  - Jeder Zug beeinflußt mindestens ein Netz
  - Bestimme die durchschnittlichen Kosten pro Netz
  - Wenn T kleiner als ein Bruchteil davon ...
    - Nur noch kleine Chance, dass Zug akzeptiert wird
    - $T < 0.005 \cdot \mathit{AvgCostPerNet}$
  - Auch einfachere Realisierung möglich
    - Letzte k Züge ohne akzeptierten Zug
    - Letzte k Züge ohne Verbesserung von BSF
    - ...

VPR
Kostenfunktion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Gleichzeitiges optimieren von
  1. Verdrahtungslänge
  2. Zeitverhalten

$\Rightarrow$ Kombination von 2 Kostenfunktionen

1. Korrigierter HPWL: $c_w = \sum_{n \in N} q(n_{pincount})HPWL(n)$
   Korrekturfaktor q_n um Unterschätzung vorzubeugen
   ($q(1) = 1, \ldots, q(50) = 2.79$, für Details siehe Paper auf Web-Seite [Cheng 1994])
2. Zeitverhaltensabschätzung c_t

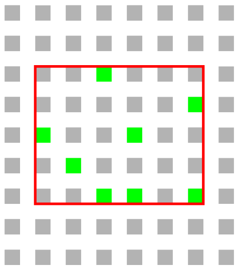
VPR
Optimierung HPWL

- Berechnung HPWL
  - Simpel: $\mathcal{O}(k)$, k Anzahl der Pins
  - Problem: k = 100 ... 1000 realistisch
  - Nach jedem Zug neu berechnen
- Besser:
  - Nach Möglichkeit nur bewegte Pins neu berechnen
    - Ein Pin ist nur in einem Netz
    - Ein Block hat aber mehrere Pins
  - Vorgehen:
    - Je Netz umspannendes Rechteck speichern:
      Position der Seiten: $(x_{min}, x_{max}, y_{min}, y_{max},)$
      Anzahl der Pins direkt auf den Seiten: $(N_{xmin}, N_{xmax}, N_{ymin}, N_{ymax},)$

VPR
Optimierung HPWL

- Als Beispiel nur linke Seite
- Bewege Terminal von x_{old} nach x_{new}
- Netz an Terminal: n

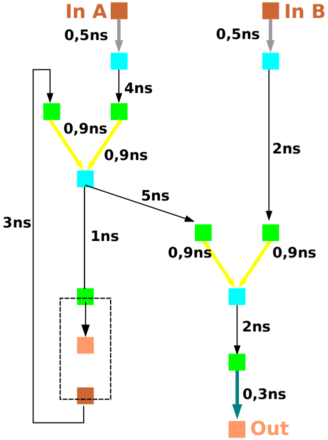
if $x_{new} \neq x_{old}$ **then**
    if $x_{new} < n.xmin$ **then**
        n.xmin := x_{new} ;
        n.Nxmin := 1
    else if $x_{new} = n.xmin$ **then**
        n.Nxmin++
    else if $x_{old} = n.xmin$ **then**
        if *(n.Nxmin > 1)* **then**
            n.Nxmin–
        else BruteForce(n) ;



$$(x_{min}, x_{max}, y_{min}, y_{max},) = (2, 7, 3, 7)$$
$$(N_{xmin}, N_{xmax}, N_{ymin}, N_{ymax},) = (1, 2, 3, 1)$$

VPR
Kostenfunktion Zeitverhalten

TECHNISCHE UNIVERSITÄT DARMSTADT



- Betrachte Platzierungsabhängiges Zeitverhalten
- Punkt-zu-Punkt Verbindungen
- Von Netzquelle *u*
- Zu jeder Netzsenke *v*
- Sicht: *Two-Terminal-Nets*
- Zeitverhalten:
  - Bestimmt aus Slacks
  - Nicht auf Pfaden (langsam)

Embedded Systems & Applications

VPR
Kostenfunktion Zeitverhalten

TECHNISCHE UNIVERSITÄT DARMSTADT

- *Wichtigkeit* einer Verbindung
  - Punkt-zu-Punkt zwischen Terminals u und v

$$Criticality(u, v) = 1 - \frac{slack(u, v)}{D_{max}}$$

  - (u, v) auf kritischem Pfad:
    $slack(u, v) = 0 \Leftrightarrow Criticality(u, v) = 1$
  - (u, v) absolut unkritisch:
    $slack(u, v) = D_{max} \Leftrightarrow Criticality(u, v) = 0$
- Timing Cost: $Delay(u, v)$ ist Schätzung!
  Noch kein *echtes* Routing!

$$c_t = \sum_{(u,v) \in E_{NetTiming}} Delay(u, v) \cdot Criticality(u, v)^{ce}$$

VPR
Kostenfunktion Zeitverhalten

- Criticality Exponent *ce*
  - Gewichtet kritischere Verbindungen höher
  - Untergeweichtet unkritischere Verbindungen
- Idee:
  - Gegen Ende auf kritische Netze konzentrieren
- Vorgehen:
  - Steigern von $ce_{start} = 1$ auf $ce_{final} = 8$ (experimentell)

$$ce = \left(1 - \frac{R_{limit}^{now} - 1}{R_{limit}^{start} - 1}\right) \cdot (ce_{final} - ce_{start}) + ce_{start}$$

VPR
Kostenfunktion Zeitverhalten

TECHNISCHE UNIVERSITÄT DARMSTADT

- *slack*() ist platzierungsabhängig
  - Unkritische Netze können kritisch werden
    Zu lange Leitungslängen
  - Kritische Netze können unkritisch werden
    Sehr kurze Leitungslängen
- Slack-Werte müssen (zeitaufwendig!) aktualisiert werden
  Timing-Analyse: T_a, T_r
- Wie oft?
  - Nach jedem Zug? Nach N Zügen?
  - N-mal pro Temperaturstufe?
  - Alle N Temperaturstufen?
- $1\times$ pro Temperaturstufe

VPR
Gesamtkostenfunktion

TECHNISCHE UNIVERSITÄT DARMSTADT

- Selbstnormierend:

$$\Delta c_w = c_w(g) - c_w(f)$$
$$\Delta c_t = c_t(g) - c_t(f)$$
$$\Delta c = \lambda \frac{\Delta c_t}{c_t^{old}} + (1 - \lambda) \frac{\Delta c_w}{c_w^{old}}$$

- λ gewichtet Zeit- gegenüber Längenoptimierung
  - Aber $\lambda = 1$ erzeugt nicht die schnellste Lösung
  - Netze wechselnd kritisch/unkritisch
    Nicht erkannt, da Timing-Analyse nur $1\times$ pro Temperaturstufe
  - Besser $\lambda = 0.5$
    Längenmaß wirkt als Dämpfer für Oszillation

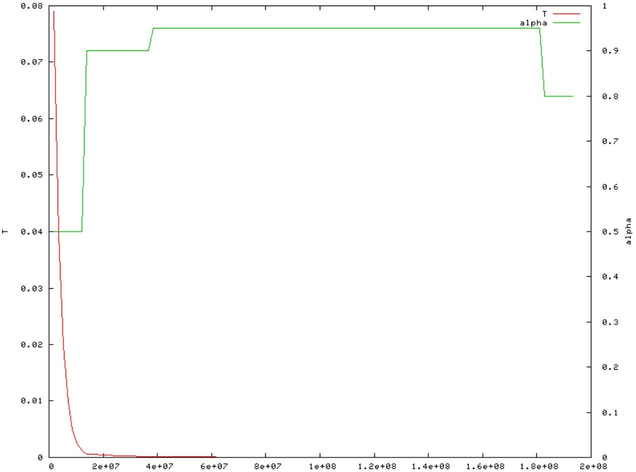
VPR
Gesamtalgorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

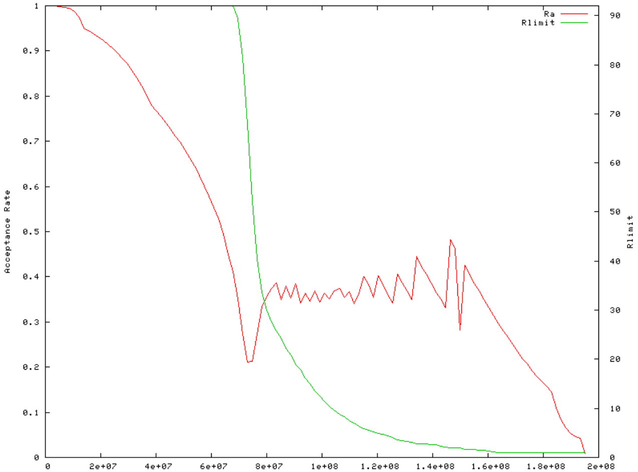
```
S := RandomPlacement() ;
T := InitialTemperature() ;
Rlimit := InitialRlimit() ;
CritExp := ComputeNewExponent(Rlimit);
repeat
    /* Bestimme Ta, Tr und slack()        */
    TimingAnalyze() ;
    /* für Normalisierung der Kostenterme
       */
    OldWiringCost := WiringCost(S) ;
    OldTimingCost := TimingCost(S) ;
    while !InnerLoopCriterion() do
        ⋮
        /* eine Temperaturstufe          */
        ⋮
    T := UpdateTemp() ;
    Rlimit := UpdateRlimit() ;
    CritExp := ComputeNewExponent(Rlimit) ;
until ExitCriterion();
```

```
⋮
/* eine Temperaturstufe:                                   */
Snew := GenerateSwap(S, Rlimit) ;
ΔtimingCost := TimingCost(Snew) – TimingCost(S) ;
ΔwiringCost := WiringCost(Snew) – WiringCost(S) ;
ΔC := λ (ΔtimingCost/OldTimingCost) +
(1-λ)(ΔwiringCost/OldWiringCost);
if ΔC ≤ 0 then
    S = Snew
else
    if random(0,1) < exp(-ΔC/T) then
        S = Snew
⋮
```

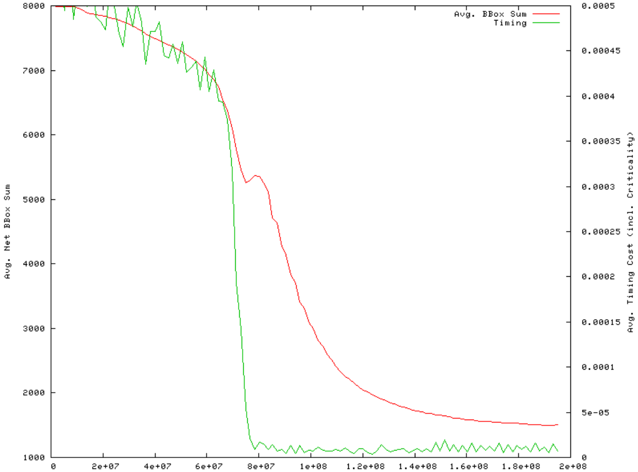
VPR
Temperatur- und α-Graph



VPR
R_a- und R_{limit}-Graph



VPR
Kosten-Graphen



DAST
STUN

- Weitere SA Verbesserung
- Dynamisch Adaptives STUN ⇒ DAST
  (Lin & Warzynek 2010)
- Idee: Verbessertes entkommen lokaler Minima
- Stochastisches Tunneln ⇒ STUN
  (Hamacher 1999)
- Zusätzlich:
  - Multimodale Bewegungen
  - Lokale Minima Detektion ⇒ Nur dann STUN benutzen

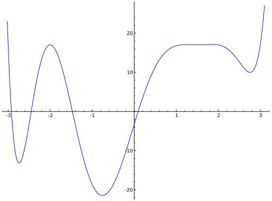
STUN
Stochastic Tunneling

- SA kann immer noch in lokalen Minima stecken bleiben (Freezing Problem)
- Idee: Tunneln durch (lokales) Maximum
- Kostenfunktion wird modifiziert: $c'(x) = c(x) \cdot g(x)$
  - *Kleine* Extrema werden geglättet
  - *Große* werden hervorgehoben
  - *Klein* und *groß* relativ zu der bisher besten Lösung mit Kosten c_{BSF}
  - Mehere Funktionen möglich (γ Tunnel Parameter der die Stäre des Glättens steuert): $e^{-\gamma(c-c_{BSF})}$, $\frac{1-sgn(c-c_{BSF})}{2}c$, $\tanh(-\gamma(c-c_{BSF}))$, $\sinh(-\gamma(c-c_{BSF}))$, ...
  - Empirisch festgestellt:
    - Funktion beeinflußt Ergebnis bis zu 20%
    - Für das Platzierungsproblem am besten: $1 - e^{-\gamma(c-c_{BSF})}$
    - $\gamma \in [0, \ldots, 5]$ beeinflußt Ergebnis bis zu 30%
      Wird manuell angepaßt
- Veränderung der Kostenfunktion

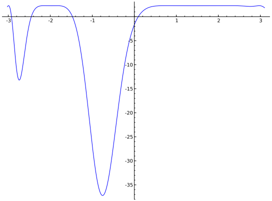
STUN
Glättungsfunktion

TECHNISCHE UNIVERSITÄT DARMSTADT

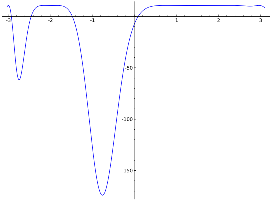
$$g_{STUN}(x) = e^{-\gamma(c(x) - c_{BSF})}$$



1-D Kostenfunktion

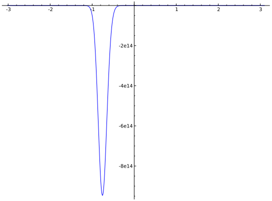


Kostenfunktion mit Glättung, BSF bei $x_{BSF} = -2.73$

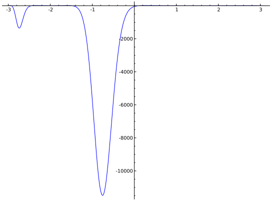


Kostenfunktion mit Glättung, BSF bei $x_{BSF} = 2.76$

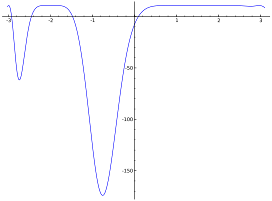
Glättungsfunktion
Einfluß γ



$\gamma = 1$



$\gamma = 5$



$\gamma = 15$

In DAST: Adapativ, so dass $\frac{c - c_{BSF}}{\gamma} \approx 0.05$

DAST
Multimodale Bewegungen

- Verallgemeinerung von VPRs R_{limit}
- Es gibt verschiedene Arten von Zügen
  (z.B. Züge mit maximalen Reichweiten: $\frac{2}{3}L, \frac{4}{3}L, 2L$)
- Aktzeptanzrate a_i für jede Zugart m_i mitprotokollieren
- Gibbs-Sampling (spez. Metropolis-Hastings-Algorithmus)
  - Anfänglich werden Züge mit gleicher Wahrscheinlichkeit gewählt
  - Später mit Wahrscheinlichkeit $p(m_i) = \frac{a_i}{\sum_{i=0} na_i}$

Für Details: Siehe Paper

DAST
Lokale Minima Erkennung

TECHNISCHE UNIVERSITÄT DARMSTADT

- STUN besonders gut wenn in lokalem Minimum
- Permanentes stochastisches Tunneln negativ:
  - Golfkurs-Effekt (Ebene mit einem Loch)
  - Lokale Züge sind weniger effektiv

$\Rightarrow$ Versuche Minima zu erkennen
und nur wenn nötig STUN zu benutzen

- DAST: Nutzt jede 10000. Iteration DFA (Detrended Fluctuation Analysis)
  - Für Details: Siehe Paper
  - Alternativen möglich: z.B. Ableitung

DAST
Zusammenfassung

TECHNISCHE UNIVERSITÄT DARMSTADT

- Stochastisches Tunneln
- Multimodale Züge
- Minima Erkennung
- Vergleich mit VPR: Verbesserung von ...

Laufzeit:	$\approx$ 30%
Verzögerung:	(kritischer Pfad) $\approx$ 12%
Verdrahtbarkeit:	(min. Tracks) 3%

Zusammenfassung

- Längenmaße
  - Lineare: HPWL
  - Nichtlineare: Quadratische und LSE-basierte Kostenfunktion
- VPR
  - Adaptives Simulated Annealing
  - Schnelle HPWL-Berechnung
  - Timingbasierte Kostenfunktion
  - Selbstnormalisierende Kostenfunktion
  - Gesamtalgorithmus
- DAST
  - STUN