Kodierung
Genetische Algorithmen und Simulated Annealing

Referenten
Dipl.-Ing. (FH) Christian Benjamin Ries
Dipl.-Ing. (FH) Matthias Vollmer

Agenda

Erklärung des Genetischen Algorithmus

f(x)=x² (2-dimensional)

Verschiedene Codierungen

Binärcode, Gray-Code,

Testreihen

Parameter
Funktionen

Messergebnisse

Konvergenzverhalten
Zeitverhalten

Fazit

Erklärung des Genetischen Algorithmus

1. Popsize erzeugen
2. Fitnesswerte zuweisen
3. Selektion (RouletteWheel)
4. Crossover
5. Mutation

Beispielfunktion:

f(x)=x² (2-dimensional)
um das Minimum zu finden, wird von der 1 das Maximum abgezogen

→ f(x)=-x²

1. Popsize festlegen, mit den Grenzen des Suchbereichs

popsize=8
xl=-6; xr=6

Werte: $I_1 = -6$, $I_2 = -4$, $I_3 = -3$, $I_4 = -1$, $I_5 = 1$, $I_6 = 2$, $I_7 = 5$, $I_8 = 6$,

Beispielfunktion:

$f(x)=x^2$ (2-dimensional)

→ $f(x)=-(x^2)$

2. Fitnesswerte zuweisen

popsize=8

xl=-6; xr=6

Werte: $l_1 = -6$, $l_2 = -4$, $l_3 = -3$, $l_4 = -1$, $l_5 = 1$, $l_6 = 2$, $l_7 = 5$, $l_8 = 6$,

f(x) : $l_1 = 36$, $l_2 = 16$, $l_3 = 9$, $l_4 = 1$, $l_5 = 1$, $l_6 = 4$, $l_7 = 25$, $l_8 = 36$,

fit(c)= m- f(x) (m ≥ max (f(x) xЄS) m=36

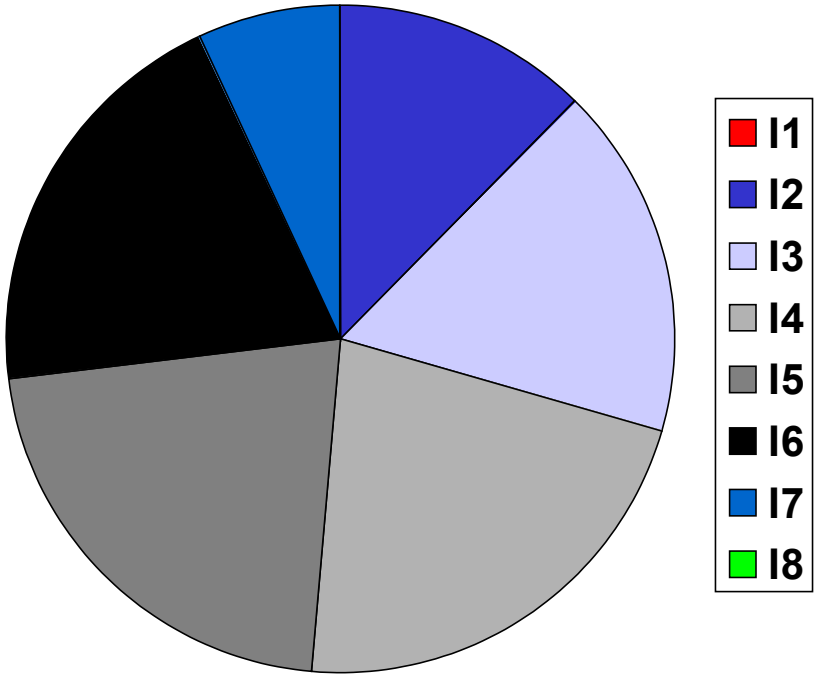
fit(c): $l_1 = 0$, $l_2 = 20$, $l_3 = 27$, $l_4 = 35$, $l_5 = 35$, $l_6 = 32$, $l_7 = 11$, $l_8 = 0$,

p(c) = 1 - fit(c) / F

p(c): $l_1 = 0$ $l_2 = 0{,}125$ $l_3 = 0{,}16875$ $l_4 = 0{,}21875$

$l_5 = 0{,}21875$ $l_6 = 0{,}2$ $l_7 = 0{,}06875$ $l_8 = 0$

3. Selektion (RouletteWheel)



Wähle popsize x mal (8x) zufällig ein Individuum aus

Die Überlebenswahrscheinlichkeit entspricht dabei den Rouletteradsegmenten

Annahme der wahrscheinlichen neuen Population:

$I_1 = 2, I_2 = 1, I_3 = -1, I_4 = 1,$

$I_5 = 2, I_6 = 2, I_7 = -3, I_8 = 5,$

4. Crossover

- Erzeuge Zufallszahlen zwischen [0;1], für jedes Individuum der neuen Population

```
for i=1:popsize
    r = rand(1);
```

- Vergleiche die Zufallszahl r mit der Wahrscheinlichkeit p(c) und schreibe die Indizes raus, wenn r < p (c)

```
if  r < c_next_select_feasibility(i)
    indezes(indezes_counter) = i;
    indezes_counter = indezes_counter + 1;
```

- Bei ungeraden Indizes wird der letzte gelöscht (reparieren)
- Erzeugen der Zufallszahlen, an welcher Stelle der Chromosomen ein Crossover stattfinden soll
  (1 point-x-over, 2 point-x-over, Inversion, Parametrisierter Uniform)
- Durchführen des Crossover mit zwei benachbarten Chromosomen
- Chromosomen außerhalb des Suchraums werden auf den Rand gelegt

Individuen nach Crossover: $I_1 = 3$, $I_2 = 0$, $I_3 = -1$, $I_4 = 1$, $I_5 = 2$, $I_6 = 2$, $I_7 = 3$, $I_8 = -5$,

1. Mutation

- Wähle zufällig Individuen für die Mutation aus
- Wähle zufällig die Stelle die mutiert (invertiert) werden soll
  Da die Informationen des Gens binär (0, 1) dargestellt werden, gibt es nur 2 unterschiedliche Allele

Bsp.: $I_1 = 3, I_2 = 0, I_3 = -1, I_4 = 1, I_5 = 2, I_6 = 2, I_7 = 3, I_8 = -5,$

Der Suchraum lässt sich mit 4 Bit beschreiben

Invertierung des 1. Bit: ± 1
Invertierung des 2. Bit: ± 2
Invertierung des 3. Bit: ± 4
Invertierung des 4. Bit: Vorzeichenänderung

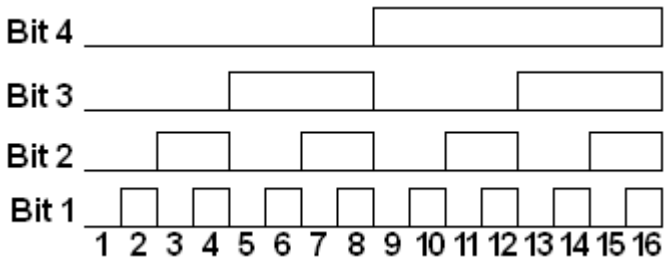
Verschiedene Codierungen

1. Binärcode
2. Gray-Code
3. Floating point

Parametrierung über

```
codecid:
      [1] = binary
      [2] = gray-code
      [3] = floating point
```

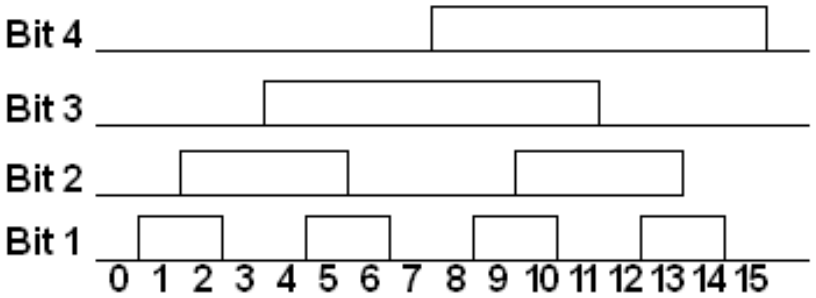
1. Binärcode



Bit 4	Bit 3	Bit 2	Bit 1	Dezi
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

2. Gray-Code

- verwendet ebenfalls zur Darstellung von Informationen die Basis 2
  Dadurch können beispielsweise mit einer Länge von 8 Bit
  $2^8 = 256$ Informationen dargestellt werden
- Bei 2 benachbarten Zahlen ändert sich immer nur 1 Bit



Bit 4	Bit 3	Bit 2	Bit 1	Dezi
0	0	0	0	0
0	0	0	1	1
0	0	1	1	2
0	0	1	0	3
0	1	1	0	4
0	1	1	1	5
0	1	0	1	6
0	1	0	0	7
1	1	0	0	8

Umrechnungen zwischen Binärcode, Gray-Code und Dezimalzahlen

1. Binär → Dezimalzahl

$$Ergebnis_{10} = \sum_{i=10}^{n-1} Wert_i * 2^n$$

Bsp.:

$$101001 = 1*2^6 + 0*2^5 + 1*2^4 + 0*2^3 + 0*2^2 + 1*2^1 = 82$$

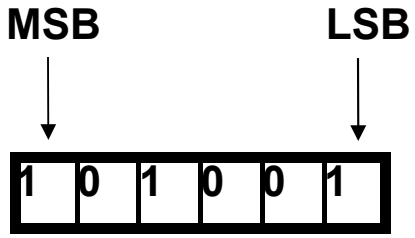
2. Gray-Code → Dezimalzahl

Bsp.:

$$101001 = 1*2^6 - (0*2^5 - 1*2^4 - 0*2^3 - 0*2^2 - 1*2^1) = 46$$

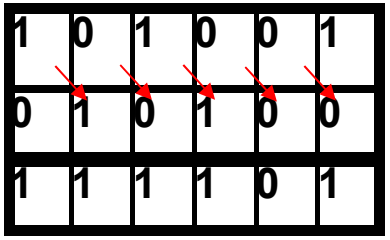
Umrechnungen zwischen Binärcode, Gray-Code und Dezimalzahlen

3. Binär → Gray-Code



MSB = most significant bit
LSB = least significant bit

Das LSB des Binärcodes wird gelöscht und die beiden Zahlen dann mit dem XOR-Gatter verknüpft.



A	B	A XoR B
0	0	0
0	1	1
1	0	1
1	1	0

Testreihenparameter

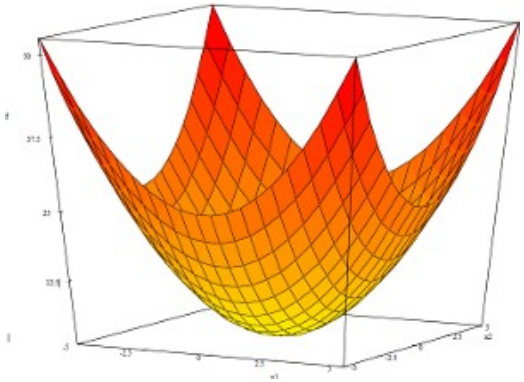
Parameter für GA

- Anwachsende Population, von *50-1000*
- Mittelung der Ergebnisse bei *10* Messreihen
- Selektionswahrscheinlichkeit ist stochastisch (RouletteWheel)
- Mutationswahrscheinlichkeit := *5%*
- Crossover-Operator: Two-Point-Crossover
- Crossover-Wahrscheinlichkeit := Wahrscheinlichkeit des vorherigen Durchlaufs

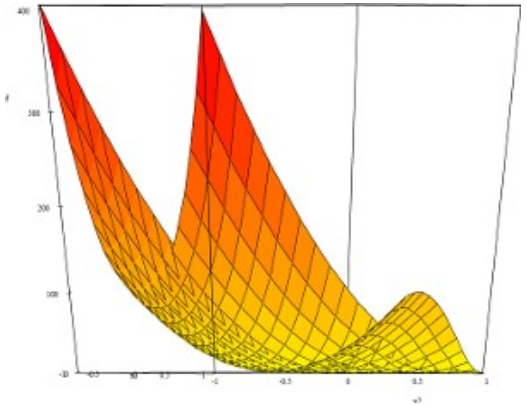
Parameter für Simulated Annealing

- Mittelung der Ergebnisse bei 10 Messreihen
- $s\vec{v}ec_0 = Grenzwerte$, *T=10*, *a = 0.895* (sollte zwischen *0.8-0.99* liegen), *abort = 1000*

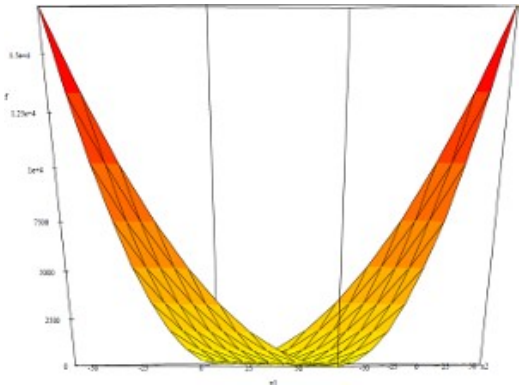
Testfunktionen



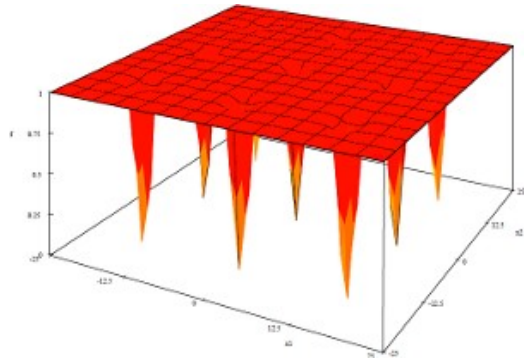
Sphäre Funktion



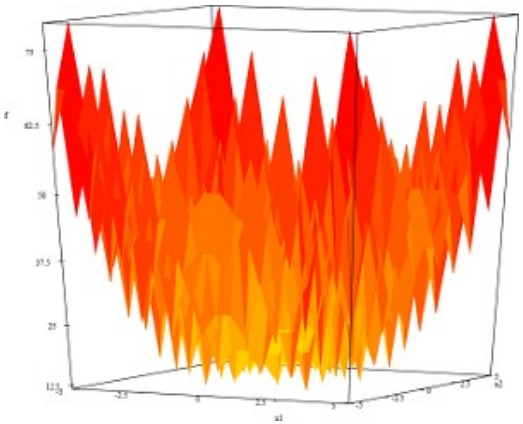
Rosenbrock's Sattel



Schwefel Funktion



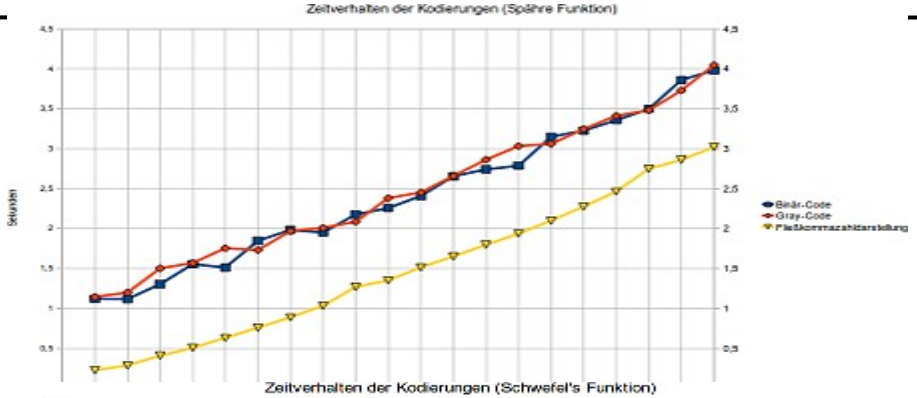
Shekel's
Fuchsbauten

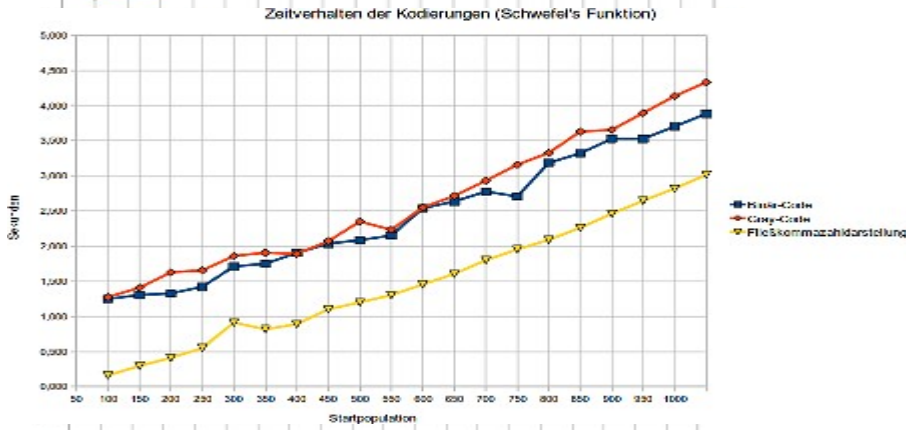


Rastrigin Funktion

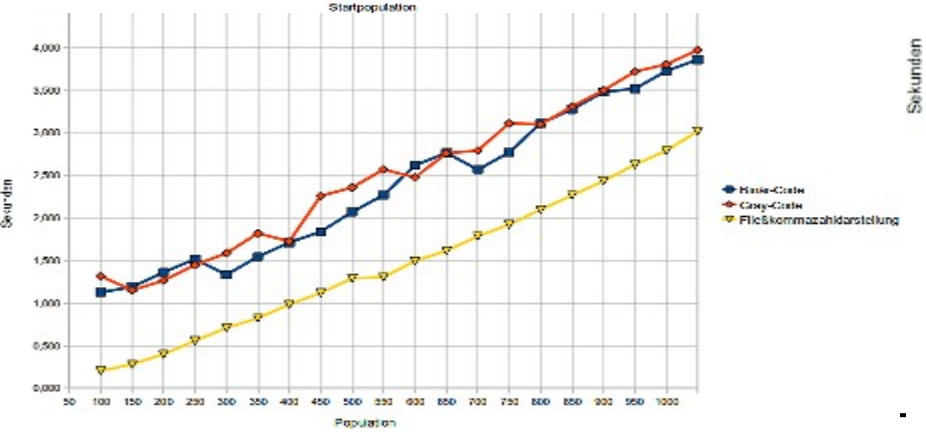
Quelle: Marion Riedel,
Diplomarbeit, 2002

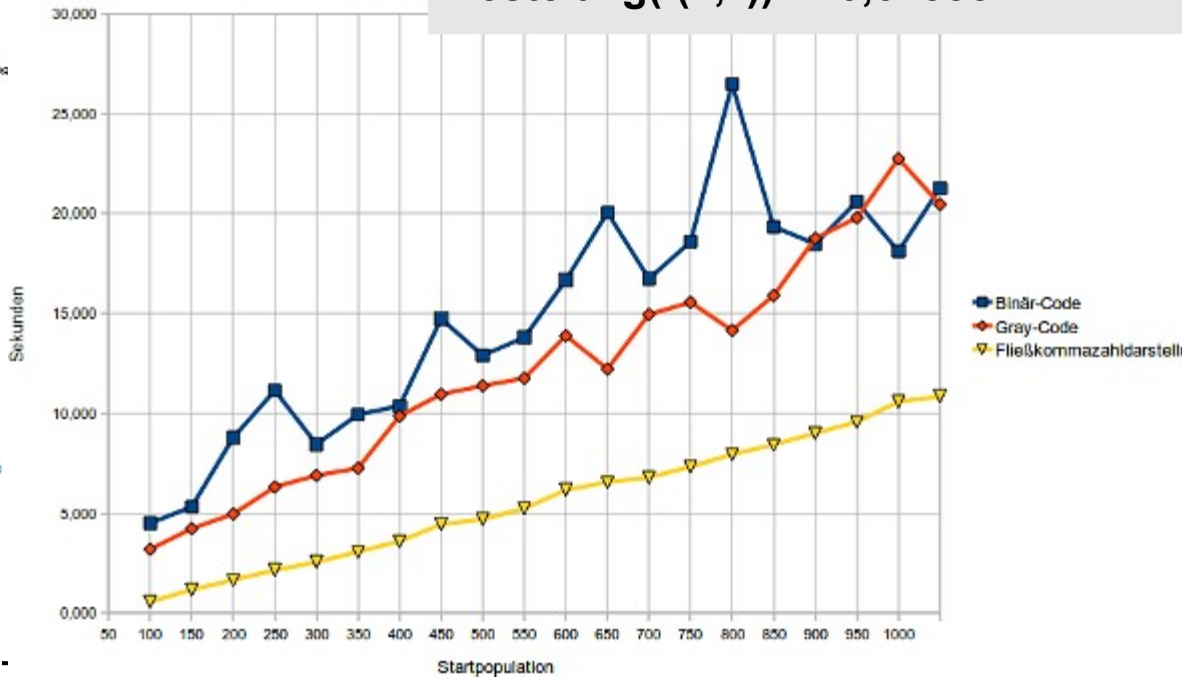
Messergebnisse – Zeitverhalten GA





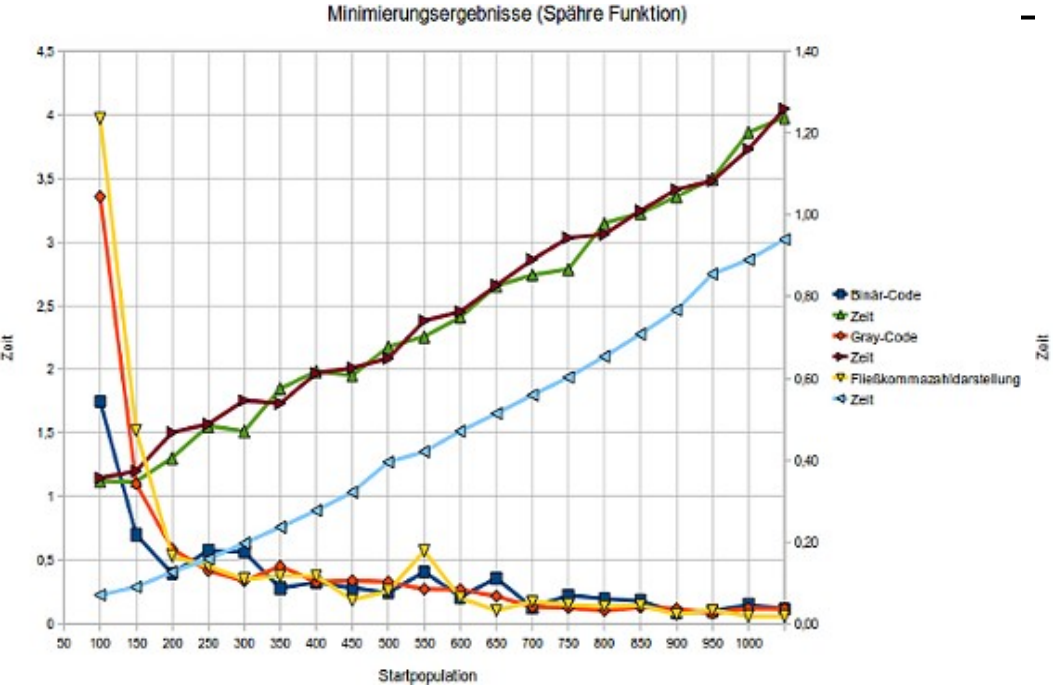
Shekel's Fuchsbauten
Beste f(x,z) = -2,4603
Beste avg(f(x,z)) = -0,64555

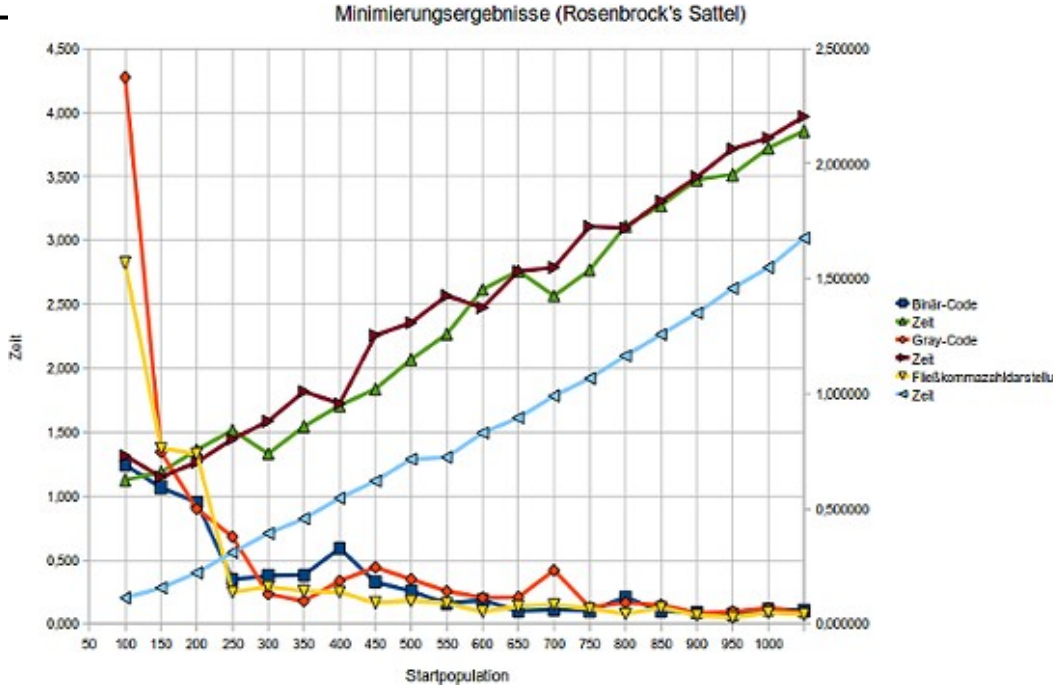


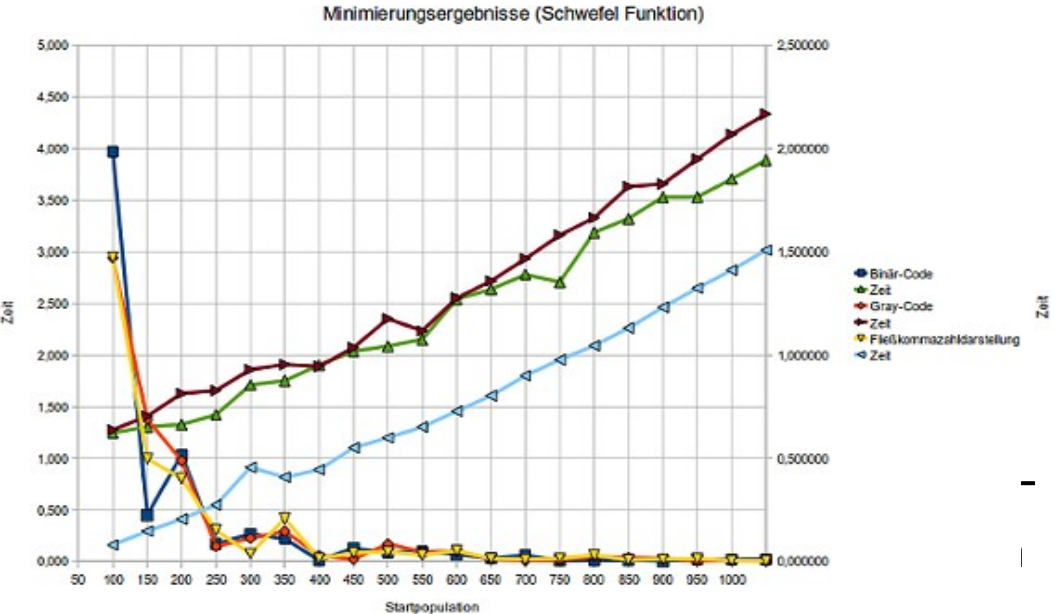


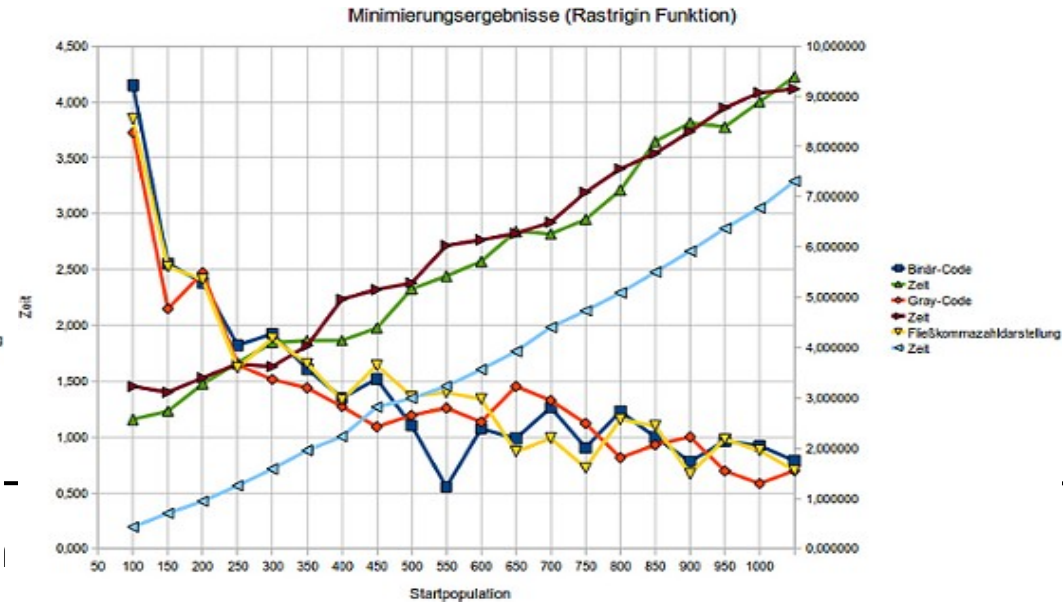
Christian Benjamin Ries
Matthias Vollmer

Messergebnisse – Konvergenzverhalten GA







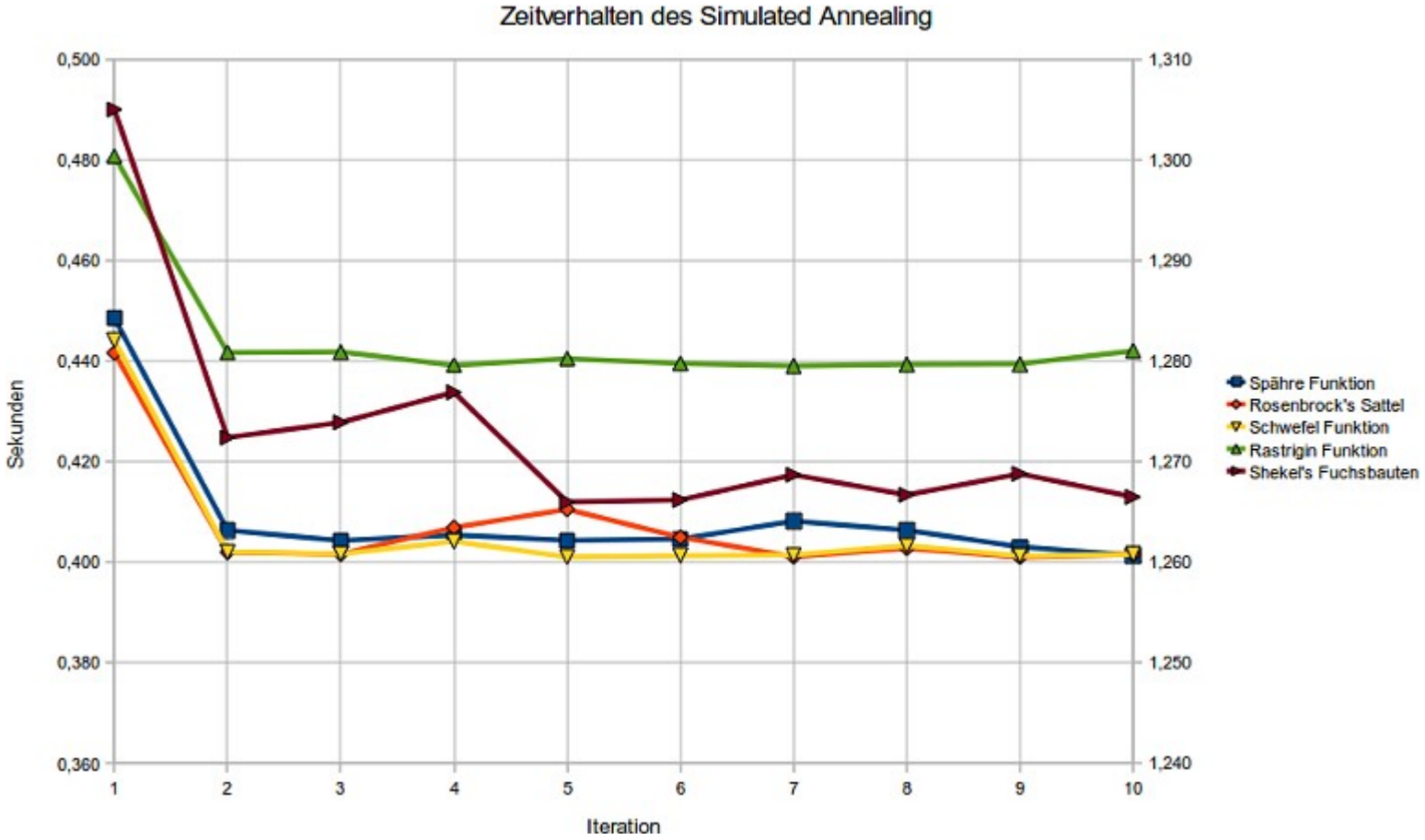


Messergebnisse – Konvergenzverhalten GA - Zahlenwerte

- Annähernd gutes Minima in den Messungen bis zu einer Startpopulation := 300

	Spähren Funktion		Rosenbrock's Sattel		Schwefel Funktion		Rastrigin Funktion		Shekel's Fuchsbauten	
	Popsize	Wert	Popsize	Wert	Popsize	Wert	Popsize	Wert	Popsize	Wert
Binär-Code Kodierung	300	0,086801	300	0,193070	200	0,084505	300	3,575400	50	-0,645550
Gray-Code Kodierung	250	0,104060	300	0,099546	200	0,074256	300	3,197600	300	-1,963400
Fließkommazahl Kodierung	250	0,107970	200	0,138290	250	0,038247	200	3,594300	50	-0,780830

Messergebnisse – Zeitverhalten *Simulated Annealing*



- Zeitverhalten des *Simulated Annealing* für die Testfunktionen, Shekel's Funktion auf der rechten y-Achse, alle anderen auf der linken y-Achse
- Viel viel weniger Iterationen, Ergebnis daher nicht verwunderlich

Messergebnisse – *Simulated Annealing* - Zahlen

Funktion	best fx	Iteration
Sphäre Funktion	0,0000053943	6
Rosenbrock's Sattel	0,0000067863	5
Schwefel Funktion	0,0000000001	9
Rastrigin Funktion	7,9609000000	3
Shekel's Fuchsbauten	0,0051622000	6

Messergebnisse – Fazit

Funktion	KGA beste	KGA Kodierung	SA beste
Sphäre Funktion	0,086801	Binär-Code	**0,0000053943**
Rosenbrock's Sattel	0,099545	Gray-Code	**0,0000067863**
Schwefel Funktion	0,074256	Gray-Code	**0,0000000076**
Rastrigin Funktion	**3,197600**	**Gray-Code**	7,9609000000
Shekel's Fuchsbauten	-0,645550	Binär-Code	**0,0051622000**

- Bei 4/5 der Messungen liefert der SA ein besseres Ergebnis
- Der Gray-Code ist bei der Kodierung und Anwendung der GA Operatoren am effektivsten

Sonstiges

- Dokumentation
- Messergebnisse
- Matlab Implementierung
- Präsentation

In der aktuellen Version und Ausarbeitung zu finden unter:

Webseite: http://ti.fh-bielefeld.de/~cries/