

Evolutionstrategien für das HeuristicLab

Mihaela Ionescu, Mihaela.Ionescu@students.uni-linz.ac.at

11. Januar 2004

Zusammenfassung

Im Rahmen des Projektes HeuristicLab wurden verschiedene Arten von Evolutionstrategie erzeugt und getestet. HeuristicLab ist ein Optimierungsframework, das die Entwicklung von stochastischen Optimierungsalgorithmen sowie den Test mittels entsprechender Problemimplementierung unterstützt. Die Evolutionstrategie gehört zu den Evolutionären Algorithmen (EA), die auf der biologischen Evolution basieren. Die Funktionalität der verschiedenen Arten der Evolutionstrategien wurde anhand von Testfunktionen verglichen.

Inhaltsverzeichnis

1	Einleitung	2
2	Evolutionstrategie	3
2.1	Notation	3
2.2	Mutation	3
2.3	Rekombination	5
2.4	Selektion	6
2.5	Algorithmus	7
3	Implementierung	9
3.1	HeuristicLab	9
3.2	ES (Standard Evolution Strategy)	9
3.3	SAES(Selfadaptive Evolution Strategy)	10
4	Test	11
4.1	Testfunktionen	11
4.2	Testkonfigurationen	12
4.3	Testergebnisse	13

1 Einleitung

Die Evolutionsstrategie ist ein Optimierungsverfahren. Sie gehört zur Klasse der Evolutionären Algorithmen (EA), die zur Lösung komplexer Optimierungsprobleme, bei denen klassische Verfahren aufgrund der Problemkomplexität oder fehlender mathematischer Eigenschaften des Optimierungsproblems nicht eingesetzt werden können, verwendet werden. Das Modell der EAs wird von der natürlichen Evolution abgeleitet, die ein sehr komplexer Suchprozess im Raum der genetischen Information ist. Sie soll Individuen erzeugen, die für die vorgegebene Umgebung möglichst überlebensfähig sind.

Die Evolutionären Algorithmen (EA) sind jedoch nur eine starke Vereinfachung des biologischen Vorbildes. Im Bereich der EAs werden einzelne, potentielle Lösungen eines Optimierungsproblems als Individuum bezeichnet. Eine Menge von Individuen, die sich im gleichen Optimierschritt befinden, bilden eine Population. Ein Optimierschritt wird im Rahmen der Evolution "Generation" genannt. Die Individuen der Anfangspopulation werden beliebig initialisiert. Die Güte eines Individuums bezüglich eines Optimierungsproblems wird mit dem Fitnesswert dargestellt. Die Suche im Lösungsraum erfolgt mittels gerichteter und ungerichteter Suchprozesse. Dazu gehören die Rekombination, Selektion und die Mutation. Ein gerichteter Suchprozess ist ein Suchprozess der nach fixen Regeln vorgeht und deterministisch ist. Ungerichtete Suchprozesse beruhen auf dem Zufallsprinzip. Die Selektion soll im Laufe des Optimierungsprozesses die Individuen mit dem besseren Fitnesswert bevorzugen. Speziell für die Evolutionsstrategie (ES) ist die Selektion ein gerichteter Suchprozess. Die Mutation soll Innovation in den Evolutionsprozess einführen. Sie ist der ungerichtete Teil des Suchprozesses. Die Rekombination soll die Elterninformationen miteinander kombinieren um neue Individuen zu erzeugen. Sie ist sowohl gerichtet als auch ungerichtet, da sie zwar zufällig die Eltern für die Rekombination auswählt, jedoch diese nach gewissen Gesetzmäßigkeiten miteinander kombiniert.

Die Evolutionären Algorithmen werden in drei Hauptrichtungen unterteilt: die Genetischen Algorithmen (GA), Evolutionäres Programmieren (EP) und die Evolutionsstrategien (ES). Die verschiedenen Richtungen unterscheiden sich in der unterschiedlichen Codierung des Problems, in der Mutation, Rekombination und Selektion. Schwefel und Bäck beschreiben die Unterschiede und testen die Algorithmen an Hand von Testfunktionen in [1].

2 Evolutionsstrategie

2.1 Notation

n	Problemdimension
I	Raum aller Individuen
\vec{a}	$\vec{a} \in I$ bezeichnet ein Individuum
μ	Anzahl der Eltern bzw. Größe der Population $\mu \geq 1$.
λ	Anzahl der Nachkommen, die durch Rekombination und Mutation erzeugt werden, wobei $\lambda \geq \mu$ sein muß.
$P(t)$	Population einer bestimmten Generation t , $P(t) = \{\vec{a}_1, \vec{a}_2, \dots, \vec{a}_\mu\}$
$r: I^\mu \rightarrow I$	Rekombinationsoperator für einen Nachkommen.
$m: I \rightarrow I$	Mutationsoperator für einen Nachkommen.
$s: (I^\lambda \cup I^{\mu+\lambda}) \rightarrow I$	Selektionsoperator.

Eine Population $P(t)$ besteht aus μ Individuen. Ein Individuum $\vec{a} \in I$ besteht aus einer Objektkomponente und einer Strategiekomponente. Die Objektkomponente x wird als n -dimensionaler reelwertiger Parametervektor $\vec{x} = (x_0, x_1, \dots, x_n) \in \mathbf{R}^n$ dargestellt und gibt die Position eines Individuums im Suchraum an. Die Strategiekomponente gibt $\vec{\sigma} = (\sigma_0, \sigma_1, \sigma_2, \dots, \sigma_\omega) \in \mathbf{R}^\omega$ beinhaltet Parameter, die für die Mutation von Bedeutung sind (siehe Mutation). Demnach ist $I = \mathbf{R}^{n+\omega}$. Die Dimension der Strategiekomponente hängt von der Variante der Evolutionsstrategie ab. Die Fitness eines Individuums wird mit einer Funktion $f: \mathbf{R}^n \rightarrow \mathbf{R}$ angegeben und wird nur von der Objektkomponente x bestimmt. Je nach Optimierungsproblem (Minimum-, bzw. Maximumproblem) suchen wir jenes Individuum mit dem geringsten, bzw. größten Fitnesswert.

2.2 Mutation

Bei der Evolutionsstrategie wird die Objektkomponente jedes Individuums \vec{a} um einen zufälligen Faktor verändert. Genauer betrachtet wird jede Dimension eines Individuums mittels folgender Vorschrift berechnet (allgemeine Definition):

$$x'_i(t) = x_i(t) + N(0, \sigma_i(t))$$

$N(0, \sigma_i(t))$ (Schrittweite) steht für einen normalverteilten Wert mit dem Erwartungswert 0 und Standardabweichung σ . Die Normalerteilung eignet sich deshalb gut, weil sie kleine Veränderungen eher bevorzugt als große und die Mutationen im Schnitt ausgeglichen sein sollten. Nachkommen sollten ihren Eltern schließlich ähnlich sein. Varianten der Evolutionsstrategien unterscheiden sich hauptsächlich in der Art wie σ bestimmt wird. Bekanntlich werden bei einer

Normalverteilung (um 0) mit großer Standardabweichung häufiger betragsmäßig große Zufallsvariablen (hier Schrittweite) zurückgegeben, als bei Normalverteilung mit kleiner Standardabweichung. Entsprechend dieser Tatsache werden wir im Folgenden von einer Erhöhung der Schrittweite reden, obwohl wir eigentlich meinen: Erhöhung der Standardabweichung für die Normalverteilung, zur Berechnung der Schrittweite.

Standard-Mutation

In der Klassischen Evolutionsstrategie wird σ am Beginn des Algorithmus festgelegt und wird im Laufe der Optimierung nicht verändert. Sie ist für jede Dimension eines Individuums, für jedes Individuum der Population und für jede Generation gleich.

$$x'_i(t) = x_i(t) + N(0, \sigma)$$

Mutation mit der 1/5 Erfolgsregel

Wir bezeichnen im Folgenden eine Mutation, welche zu einer Verbesserung des Fitness-Wertes führt, als eine erfolgreiche Mutation. Die 1/5 Erfolgsregel ist ein deterministisches Schema zur Veränderung der Variationsbreite im Laufe der Optimierung und abhängig vom Optimierungserfolg. Sie besagt, dass das Verhältnis p von erfolgreichen Mutationen zu allen Mutationen 1/5 betragen sollte. Falls es größer als 1/5 ist, muss die Schrittweite erhöht werden (da viele Lösungen eine Verbesserung der vorhergehenden Generation sind, nimmt man an, dass die Suche weit entfernt vom Globalen Optimum ist), falls es kleiner als 1/5 ist (die Suche befindet sich in der Nähe des Optimums), muss die Schrittweite verringert werden. Der theoretische Hintergrund für diese Behauptung wird in [1] beschrieben. Die Erfolgsrate wird zwar von Rechenberg auf 1/5 festgelegt, jedoch gibt es Problemstellungen für die eine kleinere bzw. eine größere Erfolgsrate eher zum Optimum führt und wird im Laufe dieser Arbeit auch als Parameter betrachtet. Für diese Mutations-Variante existiert für jede Generation eine Schrittweite $\sigma(t)$, die für alle Individuen und alle Dimensionen eines Individuums identisch ist. Durch die variable Schrittweite kann das globale Minimum genauer angenähert werden (wegen der kleineren Schrittweite in späteren Generationen). Durch die Anpassung der Schrittweite ist es jedoch möglich, dass die Suche frühzeitig konvergiert. Um dies zu verhindern muss die Erfolgsrate und die Schrittweite aufwendig und problemabhängig ermittelt werden.

$$\sigma(t+1) = \left\{ \begin{array}{ll} c * \sigma(t) & \text{falls } p > 1/5 \\ d * \sigma(t) & \text{falls } p < 1/5 \\ \sigma(t) & \text{sonst} \end{array} \right\}$$

Die Konstanten c und d sollen nach Untersuchungen von Rechenberg $c = 1.22$ und $d = 0.82$ gewählt werden.

$$x'_i(t) = x_i(t) + N(0, \sigma(t))$$

Self Adaption

Die Schrittweite unterliegt einer stochastischen Veränderung. Sie wird also genauso wie der Objektvektor mutiert. Als erstes wird die Strategiekomponente jedes Individuums und dann die Objektkomponente mutiert. Es gibt die Möglichkeit pro Individuum (1-Adaption) oder pro Dimension des Individuums eine eigene Schrittweite (n-Adaption) mitzuführen. Diese Variante erlaubt die automatische Anpassung der Schrittweiten an das jeweilige Problem. Definition für die n-Adaption:

$$\begin{aligned}\sigma'_i(t) &= \sigma_i(t) * \exp(\tau' * N(0, 1) + \tau * N_i(0, 1)) \\ \vec{x}'_i(t) &= \vec{x}_i(t) + N(0, \vec{\sigma}'_i(t))\end{aligned}$$

Die Definition für die 1-Adaption:

$$\begin{aligned}\sigma'(t) &= \sigma(t) * \exp(\tau' * N(0, 1)) \\ \vec{x}'_i(t) &= \vec{x}_i(t) + N(0, \vec{\sigma}'(t))\end{aligned}$$

Die Faktoren τ , τ' sollen laut Schwefel folgendermaßen gewählt werden:

$$\begin{aligned}\tau &\propto \left(\sqrt{2 * \sqrt{n}}\right)^{-1} \\ \tau' &\propto \left(\sqrt{2 * n}\right)^{-1}\end{aligned}$$

2.3 Rekombination

Die Rekombination soll die Information zweier oder mehrerer Elternindividuen miteinander kombinieren und daraus einen Nachkommen erzeugen. Es gibt mehrere Möglichkeiten die genetische Information zu kombinieren. Die Grundarten sind (lokal/global) diskrete bzw. (lokal/global) intermediäre Rekombination. Für die lokale Variante werden zwei Elternindividuen S und T gleichverteilt aus der Menge aller Elternindividuen für die Rekombination bestimmt und anschließend diskret oder intermediär rekombiniert. Für die global-diskrete Variante wird für jedes Element eines Nachkommens ein Elternindividuum gleichverteilt aus der Menge aller Elternindividuen bestimmt. Für die global-intermediäre Variante werden für jedes Element eines Nachkommens zwei Elternindividuen S und T bestimmt und rekombiniert.

$$x'_i = \left\{ \begin{array}{ll} x_{S,i} & \text{keine Rekombination} \\ x_{S,i} \text{ OR } x_{T,i} & \text{lokal - diskrete Rekombination} \\ (x_{S,i} + x_{T,i})/2 & \text{lokal - intermed. Rekombination} \\ x_{S,i} \text{ OR } x_{T,i} & \text{global - diskrete Rekombination} \\ (x_{S,i} + x_{T,i})/2 & \text{global - intermed. Rekombination} \end{array} \right\}$$

Die Rekombination kann sowohl auf die Objektkomponente als auch auf die Strategiekomponente angewendet werden. Siehe [1] für Details.

2.4 Selektion

Die Selektion wählt aus den Nachkommen der aktuellen Generation die Elternindividuen der nächsten Generation. In der von Rechenberg und Schwefel entwickelten Evolutionsstrategie ist dieser Operator deterministisch und soll die besten μ Individuen auswählen. Es gibt jedoch, je nach Anwendungsgebiet, verschiedenste Varianten dieses Operators unter anderem auch probabilistische Selektion, die hier nicht behandelt wird. In der klassischen Evolutionsstrategie unterscheidet man zwei verschiedene Selektionsstrategien:

- (μ, λ) Die Kommastrategie, die die besten μ Individuen der mutierten und eventuell rekombinierten Nachkommen für die nächste Generation gewählt werden.
- $(\mu + \lambda)$ Die Plusstrategie berücksichtigt bei der Auswahl der Population der nächsten Generation nicht nur die mutierten und rekombinierten Nachkommen sondern auch die Elternpopulation. Deshalb kann ein Individuum, im Gegensatz zur (μ, λ) -Selektion, mehrere Generationen überleben und das fitteste Individuum einer Population Populationsbeste kann von Generation zu Generation nie schlechter werden. Dies kann möglicherweise bei unimodalen Funktionen schneller zu einem Ergebnis führen (für die Definition einer "unimodalen Funktion" siehe Kapitel 4.1). Der Nachteil dieser Strategie (und somit der Vorteil der Kommastrategie) liegt am schweren Entkommen aus lokalen Minima wenn das zu optimierende Problem nicht unimodal ist.

2.5 Algorithmus

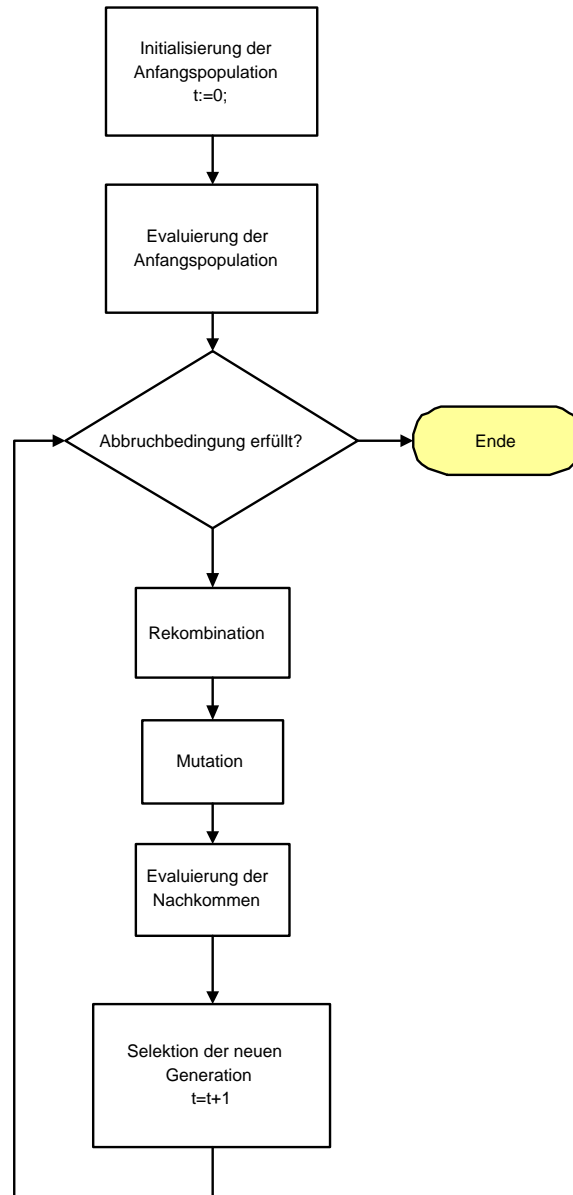


Abbildung 1: Flussdiagramm der Evolutionsstrategie

Die grobe Struktur aller Evolutionsstrategien wird in Abbildung 1 dargestellt. Die Individuen der Anfangspopulation werden mit beliebigen Werten initialisiert

und anschließend evaluiert. Als nächstes werden zufällig ausgewählte Elternindividuen miteinander rekombiniert. Die daraus entstandenen Kinder werden um einen zufälligen Faktor mutiert und anschließend evaluiert. Die besten Kinder werden dann für die neue Generation selektiert. Rekombination, Mutation und Selektion werden so lange wiederholt bis das Abbruchkriterium erfüllt wird. Der Algorithmus kann abgebrochen werden wenn zum Beispiel: das Ergebnis sich nicht mehr signifikant verändert oder eine vorgegebene Anzahl von Generationen oder maximale Ausführungszeit erreicht wurde. Im folgenden Kapitel werden die hier implementierten Versionen der Evolutionsstrategie näher betrachtet.

3 Implementierung

3.1 HeuristicLab

Die Anwendung HeuristicLab soll komplexe Optimierungsprobleme mit Hilfe verschiedener Optimierungsalgorithmen lösen. Die Struktur des Frameworks erleichtert die unabhängige Implementierung von Algorithmen und Problemen. Einmal für ein Algorithmus entwickelte Problemdarstellung kann dann mit geringem Aufwand für einen neuen Algorithmus angepasst werden. So kann man verschiedene Algorithmen mit verschiedenen Problemen kombiniert vergleichen.

3.2 ES (Standard Evolution Strategy)

Die Implementierung des Algorithmus ES ist die einfachste Version eines evolutionären Algorithmus mit einigen einfachen Erweiterungen (siehe Algorithmus 1). Die Eingabe besteht aus der Anzahl der Generationen, die durchlaufen werden bis die Ausführung abbricht, die Anzahl der Eltern und Kinder jeder Generation, die Art der Selektion (Komma oder Plus) sowie die Varianz mit der mutiert werden soll. Zusätzlich kann man eine 1/n-Erfolgsregel mit einer variablen Erfolgsrate und eine diskrete und intermediäre Rekombination aktivieren.

Algorithm 1 Klassische Evolutionsstrategie - ES

```
t := 0;
initialize  $P(0) := \{\vec{x}_1(0), \dots, \vec{x}_\mu(0)\} \in \mathbf{R}^n$ 
evaluate  $P(0) : \{f(\vec{x}_1(0)), \dots, f(\vec{x}_\mu(0))\}$ 
while (Abbruchkriterium nicht erfüllt) do
  if (userecombination = true) then
    recombine  $\vec{x}'_k(t) := r'(P(t)), \wedge k \in \{1, \dots, \lambda\}$ 
  end if
  if (1/n - rule, userecombination = false) then
    mutate  $\vec{x}''_k(t) := m_{\{\sigma(t)\}}(\vec{x}'_k(t)), \wedge k \in \{1, \dots, \lambda\}$ 
  else
    mutate  $\vec{x}''_k(t) := m_{\{\sigma\}}(\vec{x}'_k(t)), \wedge k \in \{1, \dots, \lambda\}$ 
  end if
  evaluate  $P''(t) := \{\vec{a}_1(t), \dots, \vec{a}_\lambda(t)\} : \{f(\vec{x}_1(t)), \dots, f(\vec{x}_\lambda(t))\}$ 
  // berechne  $\sigma(t)$  neu, falls die 1/n-Regel verwendet wird
  if (( $\mu, \lambda$ ) - selection) then
    select  $P(t+1) := s_{(\mu, \lambda)}(P''(t))$ 
  else
    select  $P(t+1) := s_{(\mu+\lambda)}(P(t) + P''(t))$ 
  end if
  t := t + 1;
end while
```

3.3 SAES(Selfadaptive Evolution Strategy)

SAES ist eine Erweiterung des ES - Algorithmus und entspricht der im Kapitel 2.2 beschriebenen Strategie (siehe Algorithmus 2).

Algorithm 2 Selbstadaptive Evolutionsstrategie - SAES

```
t := 0;
initialize  $P(0) := \{\vec{a}_1(0), \dots, \vec{a}_\mu(0)\} \in I^\mu$ 
  where  $I = \mathbf{R}^{n+\omega}$ 
  and  $\vec{a}_k = (x_i, \sigma_i, i \in \{1, \dots, n\})$ 
evaluate  $P(0) : \{f(\vec{x}_1(0)), \dots, f(\vec{x}_\mu(0))\}$ 
while (Abbruchkriterium nicht erfüllt) do
  if (userecombination = true) then
    recombine  $\vec{a}'_k(t) := r(P(t)), \wedge k \in \{1, \dots, \lambda\}$ 
  end if
  mutate  $\vec{a}''_k(t) := m_{\{\tau, \tau'\}}(\vec{a}'_k(t)), \wedge k \in \{1, \dots, \lambda\}$ 
  evaluate  $P''(t) := \{\vec{a}''_1(t), \dots, \vec{a}''_\lambda(t)\} :$ 
     $\{f(\vec{x}''_1(t)), \dots, f(\vec{x}''_\lambda(t))\}$ 
  if ( $(\mu, \lambda)$  - selection) then
    select  $P(t+1) := s_{(\mu, \lambda)}(P''(t))$ 
  else
    select  $P(t+1) := s_{(\mu+\lambda)}(P(t) + P''(t))$ 
  end if
  t := t + 1;
end while
```

4 Test

4.1 Testfunktionen

Für den Test und Vergleich der implementierten Evolutionsstrategien wurde eine, aus der Literatur bekannte Sammlung von Funktionen mit unterschiedlichen Eigenschaften verwendet (siehe [2]), um möglichst viele Testsituationen abzudecken. Hier werden nur die repräsentativen Beispiele jeder Gruppe verwendet. Da die Definition einer unimodalen Funktion für das Verständnis der folgenden Tests von Bedeutung ist, wird sie hier kurz angeführt.

Definition (unimodale Funktion).

1. Gegeben sei ein Intervall $[a, b]$ mit \bar{x} als Minimum der Funktion $f(x)$ in diesem Intervall. Sei $x_1, x_2 \in [a, b]$ und $x_1 < x_2$. Die Funktion $f(x)$ ist unimodal wenn:
 - (1) $(x_2 < \bar{x}) \Rightarrow f(x_2) \leq f(x_1)$, und
 - (2) $(x_1 < \bar{x}) \Rightarrow f(x_1) \leq f(x_2)$.
2. Gegeben sei ein Intervall $[a, b]$ mit \bar{x} als Maximum der Funktion $f(x)$ in diesem Intervall. Sei $x_1 < x_2 \in [a, b]$. Die Funktion $f(x)$ ist unimodal wenn:
 - (1) $(x_2 < \bar{x}) \Rightarrow f(x_2) \geq f(x_1)$, und
 - (2) $(x_1 > \bar{x}) \Rightarrow f(x_1) \geq f(x_2)$.

Sphere-Funktion

Die Sphere-Funktion ist stetig, unimodal und symmetrisch.

$$f_{SPH}(\vec{x}) = \sum_{i=1}^n x_i^2$$

Treppenfunktion

Die Treppenfunktion ist eine unstetige, unimodale Funktion mit vielen Plateauflächen.

$$f_{STR}(\vec{x}) = \sum_{i=1}^n \text{integer}(x_i)$$

Ackleys-Funktion

Ackleys-Funktion ist eine multimodale Funktion.

$$f_{ACKL}(\vec{x}) = 20 + e - 20 * e^{\left(-0.2 * \sqrt{\frac{1}{n} * \sum_{i=1}^n x_i^2}\right)} - e^{\left(\frac{1}{n} * \sum_{i=1}^n \cos(2\pi * x_i)\right)}$$

Shekels Fuchsbauten

Die Shekels Fuchsbauten sind stetig, multimodal, nicht linear und besitzen tiefe und schmale lokale Minima eingebettet in Plateau-Flächen. Plateauflächen sind problematisch für Optimierungsalgorithmen, die sich nach dem Funktionswert richten, weil sie für die Suche nach dem Optimum keine günstige Richtung angeben.

$$f_{SF}(\vec{x}) = \sum_{j=1}^{25} \frac{1}{1 + \sum_{i=1}^2 (x_i - a_{ij})^6}$$

$$a_{ij} = \begin{pmatrix} -32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -32, -16, 0, 16, 32, -16, 0, 16, 32 \\ -32, -32, -32, -32, -32, -16, -16, -16, -16, -16, 0, 0, 0, 0, 16, 16, 16, 16, 32, 32, 32, 32 \end{pmatrix}$$

Rastrigin-Funktion

Die Rastrigin-Funktion ist eine multimodale Funktion mit vielen kleinen Hügeln und Tälern.

$$f_{RASTR}(\vec{x}) = n * A + \sum_{i=1}^n x_i^2 - A * \cos(2 * \pi * x_i)$$

4.2 Testkonfigurationen

Test No.	f	S	n	$Optimum$
T_1	f_{SPH}	$[-5.12, 5.12]^2$	2	$f_{SPH}\{min\}(0, 0) = 0$
T_2	f_{STR}	$[-5.12, 5.12]^2$	2	$f_{STR}\{min\}(-5.12 + \epsilon, -5.12 + \epsilon) = -10, \epsilon \in [0, 0.12]$
T_3	f_{ACKL}	$[-32, 32]^2$	2	$f_{ACKL}\{min\}(0, 0) = 0$
T_4	f_{SF}	$[-65.536, 65.536]^2$	2	$f_{SF}\{max\}(\approx 0, \approx 0) \approx 1$
T_5	f_{RASTR}	$[-5.12, 5.12]^2$	2	$f_{RASTR}\{min\}(0, 0) = 0$
T_6	f_{SPH}	$[-5.12, 5.12]^{30}$	30	$f_{SPH}\{min\}(0, 0, 0, \dots, 0) = 0$
T_7	f_{ACKL}	$[-32, 32]^{30}$	30	$f_{ACKL}\{min\}(0, 0, 0 \dots 0) = 0$
T_8	f_{RASTR}	$[-5.12, 5.12]^{30}$	30	$f_{RASTR}\{min\}(0, 0, 0 \dots 0) = 0$

Tabelle 1: Testfunktionen

Alg. No.	Alg.	μ	γ	Std.Dev	Recomb.	Mutation	Selektion
A_1	ES	100	700	0,02	keine	Standard	(μ, γ)
A_2	ES	100	700	0,02	keine	Standard	$(\mu + \gamma)$
A_3	ES	100	700	0,02	keine	1/5-Erfolgsregel	(μ, γ)
A_4	ES	100	700	3	keine	1/5-Erfolgsregel	(μ, γ)
A_5	ES	100	700	0,02	Diskret	Standard	(μ, γ)
A_6	SAES	100	700	3	Diskret	1-Stddev.	(μ, γ)

Tabelle 2: Algorithmenkonfiguration

4.3 Testergebnisse

Die Endergebnisse ergeben sich aus dem Mittelwert von 5 Ausführungen. Die Ausführung wird nach 1000 Generationen abgebrochen.

Tabelle 3: Testergebnisse

TestNo.	Algorithmus	Test Funktion	durchschnittl. Endergebnis
1	A_1	T_1	0,000133950600728877
2	A_2	T_1	8,31048143642824E-08
3	A_3	T_1	3,36883587423626E-90
4	A_4	T_1	1,11853325994286E-88
5	A_5	T_1	8,25636464145624E-05
6	A_6	T_1	0
7	A_1	T_2	-10
8	A_2	T_2	-10
9	A_3	T_2	-9
10	A_4	T_2	-10
11	A_5	T_2	-10
12	A_6	T_2	-10
13	A_1	T_3	4,18904103601031
14	A_2	T_3	0,516507351696122
15	A_3	T_3	3,97260967592685
16	A_4	T_3	-4,44089209850063E-16
17	A_5	T_3	1,7357599814466
18	A_6	T_3	-4,44089209850063E-16

Tabelle 3: Testergebnisse

TestNo.	Algorithmus	Test Funktion	durchschnittl. Endergebnis
19	A_1	T_4	1,00000027560995
20	A_2	T_4	1,00000027605174
21	A_3	T_4	1,0000003255039
22	A_4	T_4	1,00000029446839
23	A_5	T_4	1,00000025715275
24	A_6	T_4	1,00000036989358
25	A_1	T_5	1,00213150186167
26	A_2	T_5	0,795983108863249
27	A_3	T_5	0,795967245674632
28	A_4	T_5	0
29	A_5	T_5	0,232244236355694
30	A_6	T_5	0
31	A_1	T_6	divergiert
32	A_2	T_6	stagniert
33	A_3	T_6	stagniert
34	A_4	T_6	stagniert
35	A_5	T_6	divergiert
36	A_6	T_6	2,79360936384892E-15
37	A_6	T_7	1,37667655053519E-14
38	A_6	T_8	0,994959057093331

- Test 1 Um mit der einfachsten Evolutionsstrategie ein besseres Ergebnis zu erzielen müsste man die Schrittweite kleiner wählen um der Mutation kleinere Änderungen zu erlauben.
- Test 2 Eine kleine Verbesserung kann man mit der Berücksichtigung der Eltern in der Selektion (Plus-Strategie) erreichen, da die Lebensdauer eines Individuums sich nicht nur auf eine Generation beschränkt. Wie in Abbildung 2 dargestellt ist, hat die Plus-Strategie für die unimodale Sphere-Funktion eine schnellere Konvergenzzeit als die Komma-Strategie. Für multimodale Funktionen ist die Plus Version jedoch ungeeignet, weil sie leichter in lokale Minima hängen bleibt.
- Test 3,4 Mit der 1/5 - Erfolgsregel erzielt man ein viel genaueres Ergebnis, da die Schrittweite in der Nähe des Optimums immer

kleiner wird.

Test 5	Die Rekombination für die Optimierung der Sphere-Funktion bringt nur eine kleine Verbesserung zur einfachen ES.
Test 9,10	Mit der 1/5-Erfolgsregel und der Anfangsstandardabweichung von 0,02 konvergiert die Schrittweite beim Testen der Treppenfunktion frühzeitig. Um dies zu verhindern kann man in diesem Fall eine höhere Anfangsschrittweite einstellen (siehe Abbildung 4).
Test 13-17	Ohne Anpassung der Schrittweite kann die Standard-ES nur sehr schwer das globale Optimum einer multimodalen Funktion finden. Mit der richtigen Konfiguration der Parameter (Anfangsschrittweite und Erfolgsrate) liefert die ES mit der 1/5 Erfolgsregel wesentlich bessere Ergebnisse.
Test 18	Im Gegensatz zu A4 konvergiert A6 schneller (siehe Abbildung 5)
Test 19-23	A1, A2, A3 und A5 sind sehr ungeeignet für die Lösung der Foxhole-Funktion, weil sie schnell in lokale Minima hängen bleiben. A4 (1/5-Erfolgsregel) findet auf in 1/5 der Testfälle das globale Optimum. A6 (Selbstadaptiv) findet in 3/5 der Testfälle das globale Optimum.(mit 20 Testfälle getestet)
Test 31-36	Für höhere Dimensionen (30-Dim Sphere-Funktion) eignet sich die SAES besser.(siehe Abbildung 6)

Die erfolgreichsten Algorithmen des Tests sind ES (mit der 1/5-Regel) und SAES. Für unimodale Funktionen eignet sich die einfache ES ebenfalls weil sie ähnlich gute Optima findet, jedoch mit einer besseren Konvergenzzeit (siehe Abbildung 3). Für multimodale Funktionen ist jedoch die Anpassung der Schrittweite sehr wichtig. Bei niedrigen Dimensionen besteht nicht viel Unterschied zwischen ES(1/5-Regel) und SAES, erst bei höheren Dimensionen werden die Vorteile von SAES deutlich.

In den folgenden Abbildungen werden die Algorithmen an verschiedenen Funktionen gegenübergestellt. Die vertikale Achse entspricht dem Funktionswert und die horizontale der Anzahl der Generationen. Die untere Kurve stellt die besten Ergebnisse jeder Generation dar. Die obere Kurve entspricht dem Mittelwert aller Individuen jeder Generation.

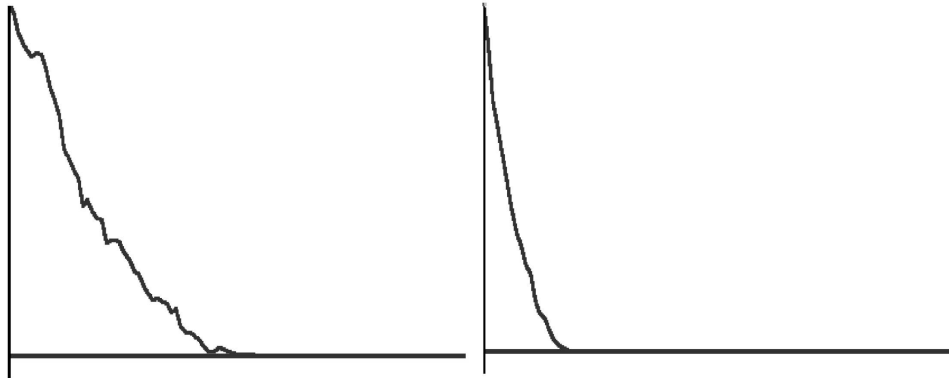


Abbildung 2: Vergleich von (1,7)-ES und (1+7)-ES anhand der 2-Dim. Sphere-Funktion (50 Generationen)

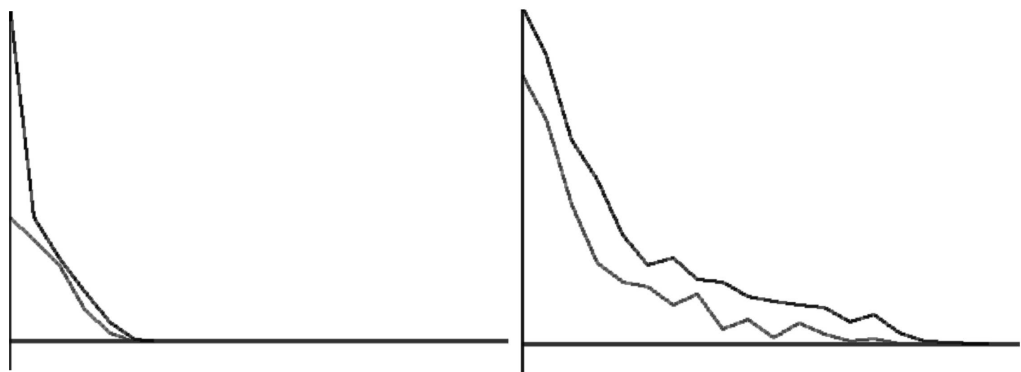


Abbildung 3: Vergleich von (10,70)-ES mit 1/5-Erfolgsregel $\sigma(0) = 0,02$ und (10,70)-SAES anhand der 2-Dim.Sphere-Funktion (20 Generationen)

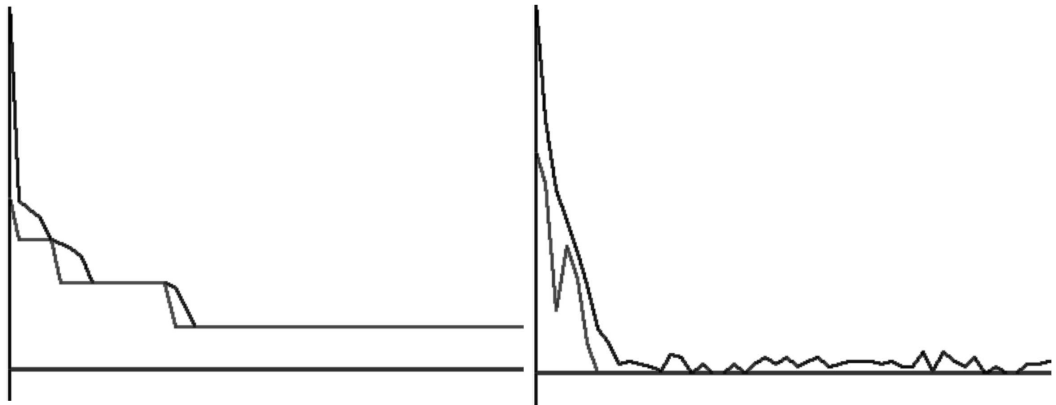


Abbildung 4: Vergleich von (10,70)-ES mit 1/5-Regel und $\sigma(0) = 0,02$ und (10,70)-SAES mit 1/5-Regel und $\sigma(0) = 3$ anhand der 2-Dim Treppenfunktion (50 Generationen)

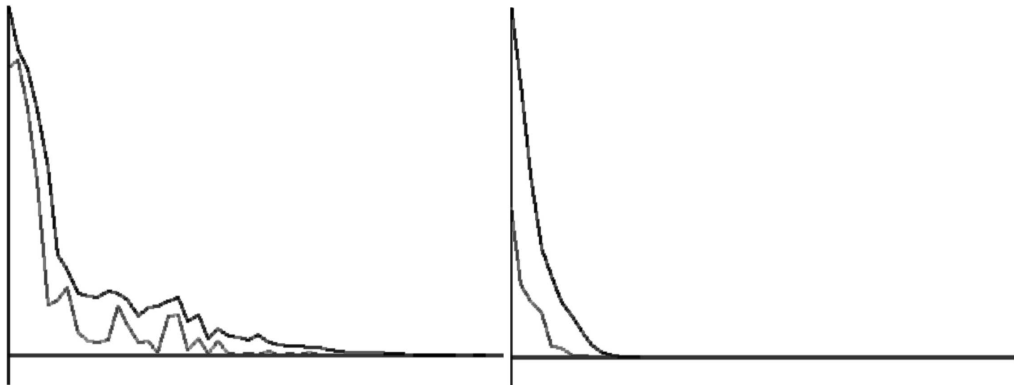


Abbildung 5: Vergleich von A4 und A6 anhand der 2-Dim. Ackley-Funktion(50 Generationen)

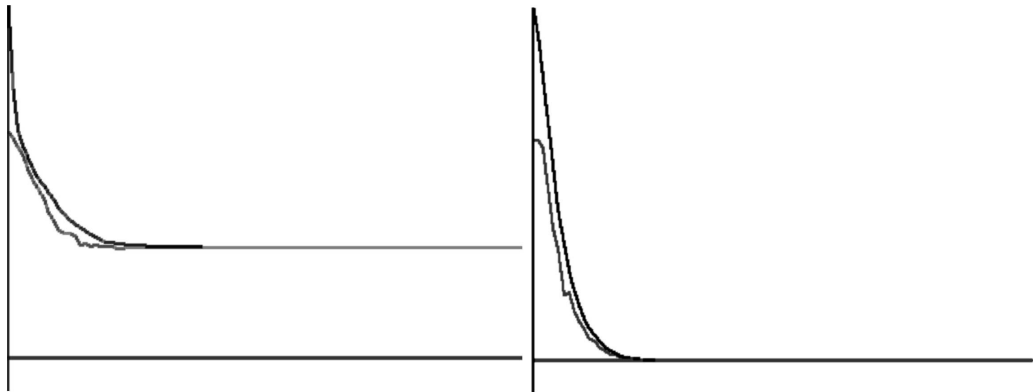


Abbildung 6: Vergleich von A_3 und A_6 anhand der 30-Dim. Sphere-Funktion (100 Generationen)

Literatur

- [1] T. Bäck und H.P. Schwefel, (1993). "An Overview of Evolutionary Algorithms for Parameter Optimization", Universität Dortmund, Department of Computer Science, Chair of Systems Analysis.
- [2] X.Yao und Y.Liu, (1997). "Fast Evolution Strategies", University of New South Wales, Australian Defence Force Academy, Canberra.
- [3] L.Hildebrand,(2001). "Asymmetrische Evolutionsstrategien"., Dissertation zur Erlangung des Grades eines Doktors der Naturwissenschaften der Universität Dortmund.
- [4] M.Affenzeller, (2000). "Stochastische Optimierung", Vorlesungsunterlagen, Institut für Systemwissenschaften an der J.K Universität Linz.