Anwendungen und Algorithmen basierend auf Prinzipien eines künstlichen Immunsystems

Herzlich Willkommen

Übersicht

1. Immunsystem
2. Immun-Algorithmus
3. Testanwendung 1
4. Testanwendung 2
5. Zwischenstand
6. Variation des Algorithmus
7. Testanwendung 3
8. Testanwendung 4

1. Das Immunsystems

1. Aufgabe

- Erkennt und beseitigt gefährliche Substanzen oder Moleküle: Antigene
- Benötigt dazu spezielle Antikörper, die auf die jeweiligen Antigene angepasst sind

- Wie funktioniert diese Anpassung?

1. Immunsystem
2. Klonale Selektion

- Nur Antikörper mit passenden Rezeptoren werden geklont (kopiert) um auf Bedrohungen zu reagieren
- Einige Zellen mit Rezeptoren werden mit langer Lebenserwartung geklont um ein Gedächnis aufzubauen

- Beim Klonen entsteht durch Hypermutation Vielfalt, ermöglicht es neue Erreger besser zu erkennen.

1. Immunsystem

3. Immunreaktion in zwei Phasen

- Ablauf in 2 Phasen:
  - 1. Phase:
    Erkennen und bekämpfen eines neuen Erregers und entwickeln des Gedächnisses
  - 2. Phase:
    Bereits bekannte Antigene werden durch spezialisierte Gedächniszellen erkannt und eine schnellere stärkere Reaktion kann erfolgen.

1. Immunsystem

4. Anwendung auf Problemlösungsverfahren

- Antigen
  = die Parameter einer Problemestellung

- Antikörper
  = mögliche Lösungen

1. Immunsystem

4. Anwendung auf Problemlösungsverfahren

- 1. Phase (Trainingsphase):
  - Man beginnt mit zufälligen möglichen Teil-Lösungen, die nach und nach zu endgültigen optimierten Lösungen heranreifen. Man erhält einen oder mehrere Antikörper (wenn mehrere Lösungen möglich sind) mit korrekten Lösungen

- 2. Phase (Testphase):
  - Trainierte Antikörper können verwendet werden um ähnliche Probleme schneller zu lösen.

2. Immun-Algorithmus

1. Variablen

- Grundlage: Random Search
- Als Antigen verwendet man ein Menge an Variablen des Kombinatorischen Optimierungsproblems
- Als Antikörper mögliche Lösungen als Bit-strings der Länge l
- Die Populationsgrösse: d
- Anzahl der Klone pro Zelle: dub

2. Immun-Algorithmus

2. Vorgehen

- t = 0
- Initialisiere $P^{(0)}$ = zufällige mögliche Lösungen
- Evaluiere $P^{(0)}$
- while (T ($P^{(t)}$) = 0)
- P^{clo} = cloning ($P^{(t)}$,dup)
- P^{hyp} = hypermutation (P^{clo})
- evaluate (P^{hyp})
- $P^{(t+1)}$ = select (($P^{hyp} \cup P^{(t)}$), d) (multiset!)
- t = t+1
- wend

2. Immun-Algorithmus
3. Mutation

- Als Mutation möglich:
  - Bit flip an beliebiger stelle
  - oder beliebige Map-Funktion f: {0,1} → {0,1}

- Repräsentieren verschiedene Suchfunktionen

2. Immun-Algorithmus

4. Wichtige Größe: Population

- Populationsgrösse d
  - Wenn zu klein gewählt:
    - vorzeitige Konvergenz auf falsche Lösung
  - Wenn zu gross gewählt:
    - Fitness Verbesserung dauert zu lange
    - viel überflüssige Verechnungen zu machen
    - unrentable Laufzeit
  - Einfacher Algorithmus verwendet:
    - ohne dynamische Populationsgrösse
    - kann durch Beeinflussung von d und dub auch nichtdominante Lösungen finden

2. Immun-Algorithmus

4. Nichtdominante Lösungen

- Expansions Phase:
  - Population wird grösser
  - Cloning und Hypermutation untersucht die Fitness Landscape nach möglichen Lösungen
- Reduktions Phase:
  - Population wird wieder kleiner
  - Selektion der besten Lösungen

3. Testanwendung 1
1. Problemstellung

- Minimum Hitting Set Problem
  - Gegeben:
    - Menge U
    - Menge S von Untermengen von U
    - Positive Zahl k ≤ |U|
  - Gesucht:
    - eine Untermenge U' von U, sodass |U'| ≤ k und U' enthält aus jeder Menge in S mindestens ein Element
- Problem ist NP-vollständig

3. Testanwendung 1
2. Fitness-Funktion

- Fitness-Funktion:
  - $f_{hs}(x)= |x| + (|S| - Hits(x))$

- Die möglichen Lösungen müssen beide Terme minimieren

- Für |S| - Hits(x) = 0
  hat man eine gültige Lösung gefunden

3. Testanwendung 1

3. Welche Lösung ist besser?

- Die Chance einen Term zu minimieren sind bei beiden Termen gleich gut.
- Beispiel:
  - $|S|=50000$
  - $|x_1|=40$ $Hits(x_1)=49997$ $f_{hs}(x_1)=43$
  - $|x_2|=42$ $Hits(x_2)=49999$ $f_{hs}(x_2)=43$
- Welche Lösung ist besser?

3. Testanwendung 1
4. Ergebnisse

- Algorithmustest:
  - |U| = 100
  - |S| = 1.000 10.000 50.000

Table 1. Minimum Hitting Set Instances

	hs100-1000		hs100-10000		hs100-50000	
d	25	100	50	200	50	200
dup	15	30	15	15	15	10
best	6	6	9	9	39	39
#min	1	3	3	5	3	2
AES	2275	18000	6800	27200	45050	98200

- Höheres d = mehr nichtdominierende Lösungen gefunden
- Höheres dup = schnellere Konvergenz

3. Testanwendung 1

4. Ergebnisse

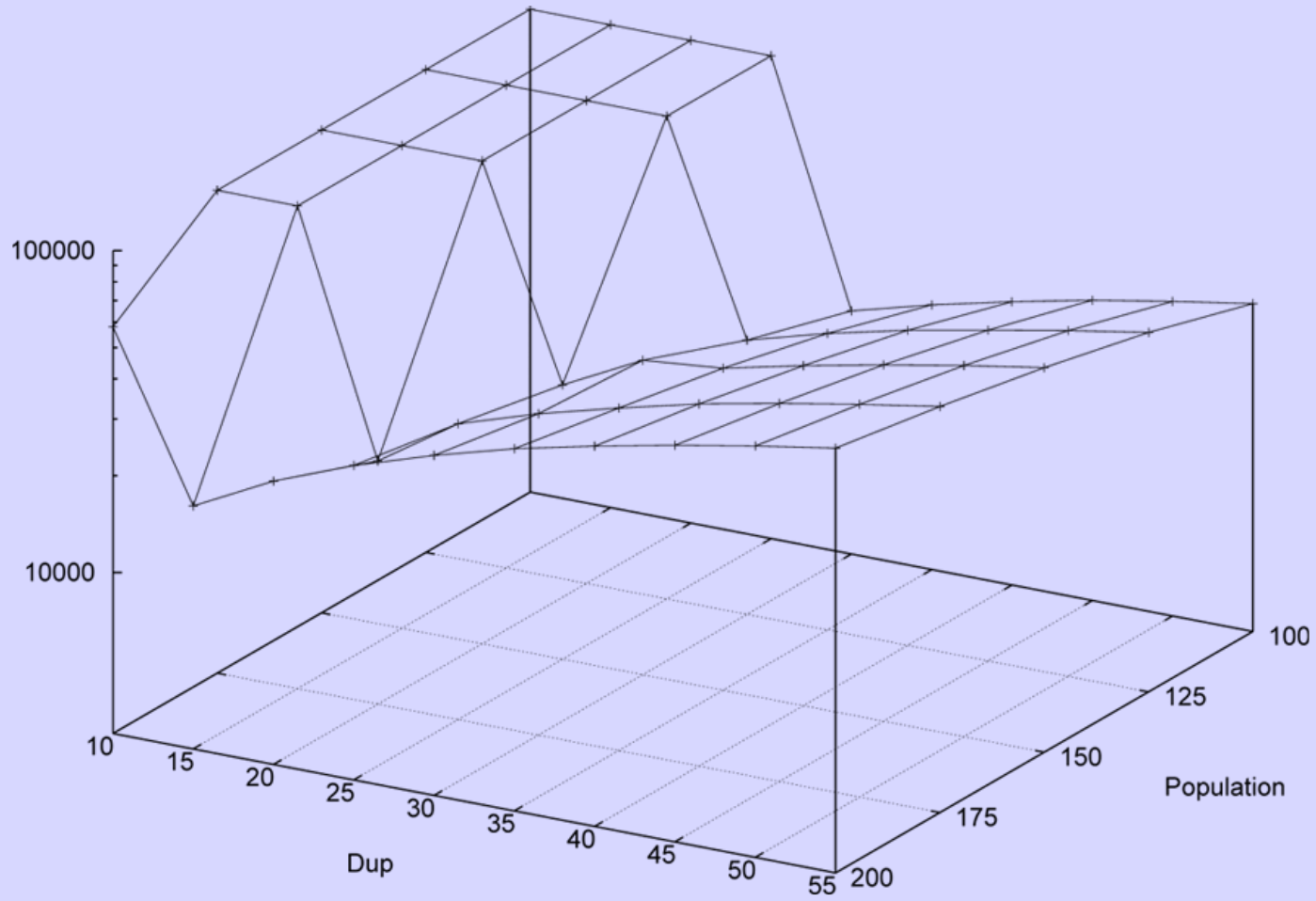


Fig. 1. 3D representation of experimental results, with dimensions: d, dup and AES

3. Testanwendung 1

4. Ergebnisse

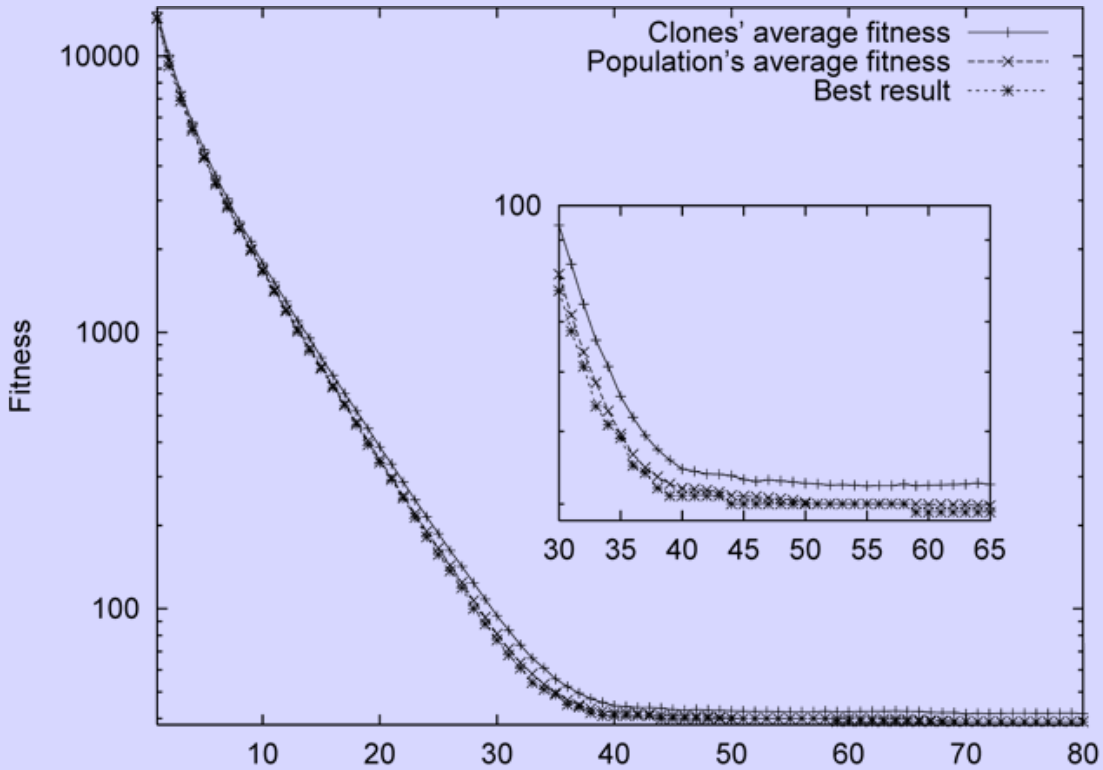


Fig. 2. Fitness function versus generations. Y axis is in log scale and origin is fixed at X-value 39.

- ab Generation 60 Minimal Set gefunden, danach nur noch nichtdominierende Lösungen mit gleicher Kardinalität

4. Testanwendung 2

1. Problemstellung

- 3-Sat Problem:
  - Gegeben:
    - Menge Variablen V
    - Menge C mit Formeln von jeweils 3 Variablen
  - Gesucht:
    - Besetzung der Variablen mit Boolschen Werten, sodass alle Formeln erfüllt sind
- Problem ist NP-vollständig

4. Testanwendung 2

2. Fitness-Funktion

- Fitness-Funktion:
  - $f_{sat}(x)$= #ErfüllteFormeln (c,x)

4. Testanwendung 2

3. Problemerstellung

- A. van Gelders 3-SAT problem instance generator MKCNF.C
  - (i1): sat30-129 (rs: 83791)
  - (i2): sat30-129 (rs: 83792)
  - (i3): sat30-129 (rs: 83792)
  - (i4): sat40-172 (rs: 62222)
  - (i5): sat50-215 (rs: 82635)

4. Testanwendung 2

4. Ergebnisse

Table 2. 3-SAT Instances

	(i1)	(i2)	(i3)	(i4)	(i5)
d	50	50	50	50	75
dup	20	30	20	30	15
AES	14632	292481	11468	24276	50269

- Die Rechenzeit hängt proportional von der Anzahl und Schwierigkeit der Formeln ab

4. Testanwendung 2

5. Erweiterung der Fitness-Funktion

- Mit SAW (stepwise adaption of weights)

Table 3. 3-SAT Instances

Case	V	C	Randseed	SuccessR	AES	SuccessR	AES
1	30	129	83791	1	2708	1	6063
2	30	129	83892	0.94	22804	0.96	78985
3	30	129	83792	1	12142	1	31526
4	40	172	83792	1	9078	1	13328
5	40	172	72581	0.82	37913	1	2899
6	40	172	62222	1	37264	0.94	82031
7	50	215	87112	0.58	17342	1	28026
8	50	215	82635	1	42137	1	60160
9	50	215	81619	0.26	67217	0.32	147718
10	100	430	87654	0.32	99804	0.06	192403
11	100	430	78654	0.04	78816	0.44	136152
12	100	430	77665	0.32	97173	0.58	109091

5. Zwischenstand

- Vorteile:
  - einfacher Algorithmus mit nur 2 Parametern lässt die Dynamik besser vorraussagen
  - Benötigt wenig Population (Grössenordnung 1000 und weniger)
- Nachteile:
  - Lokales Minimum bei zu kleinem d oder dup
  - zu hohe Werte können Overhead produzieren
  - Die Selection benutzt einen elitären Auswahlopera-tor, der zwar die Suche beschleunigt aber zu einem lokalen Minimum führen kann und damit die Vielfalt der Lösungen einschränkt

6. Variation des Algorithmus

1. Variation der Mutation

- **Static Hypermutation**
  Feste Anzahl an Mutationen
- **Proportional Hypermutation**
  Anzahl Mutationen proportional zur Fitness-Funktion
- **Inversely Propotional Hypermutation**
  Anzahl Mutationen ist antiproportional zur Fitness-Funktion
- **Hypermacromutation**
  Anzahl der Mutationen wird aus einem zufällig gewählten Intervall ausgeführt

6. Variation des Algorithmus

2. Aging Operator

- static pure aging
  - Jeder Antikörper bekommt ein Alter τ
  - Wenn Antikörper zu alt ist ($\tau > \tau_B$), dann werden sie entfernt, egal welche Fitness-Funktion sie haben
  - Beim Klonen bekommen geklonte Antikörper das gleiche Alter wie ihre Ursprungsantikörper
  - Wenn bei der Mutation eine Verbesserung der Fitness stattfindet, wird das Alter auf 0 gesetzt
  - τ_B > maximale Generationszahl
    = kein Aging wird benutzt

6. Variation des Algorithmus

3. Selection

- $(\mu + \lambda)$ selection
  - keine Redundanz
  - $\mu = d$
  - $\lambda = Nc$ oder $2Nc$

  - Es werden aus λ Antikörpern immer μ Antikörper ausgewählt
  - Überleben nicht genug Antikörper den Alterungsprozess, so werde neue erzeugt

6. Variation des Algorithmus

4. Ablauf

Immune Algorithm$(\ell, d, dup, \tau_B, c, H, HM)$
$Nc := d * dup;$
$t := 0;$
$P^{(t)} :=$ Initial_Pop();
Evaluate$(P^{(0)})$;
while ($\neg$ Termination_Condition()) **do**
    $P^{(clo)} :=$ Cloning $(P^{(t)}, Nc)$;
    if (H) **then** $P^{(hyp)} :=$ Hypermutation $(P^{(clo)}, c, \ell)$;
        Evaluate$(P^{(hyp)})$;
    if (HM) **then** $P^{(macro)} :=$ Hypermacromutation (P^{clo});
        Evaluate $(P^{(macro)})$;
    $(P_a^{(t)}, P_a^{(hyp)}, P_a^{(macro)}) :=$ Aging$(P^{(t)}, P^{(hyp)}, P^{(macro)}, \tau_B)$;
    $P^{(t+1)} := (\mu + \lambda)$-Selection $(P_a^{(t)}, P_a^{(hyp)}, P_a^{(macro)})$;
    $t := t + 1;$
end_while

7. Testanwendung 3
1. Problemstellung

- Trap Functions
- Eingabe: Anzahl der 1er in einem Bitstring der Länge l:

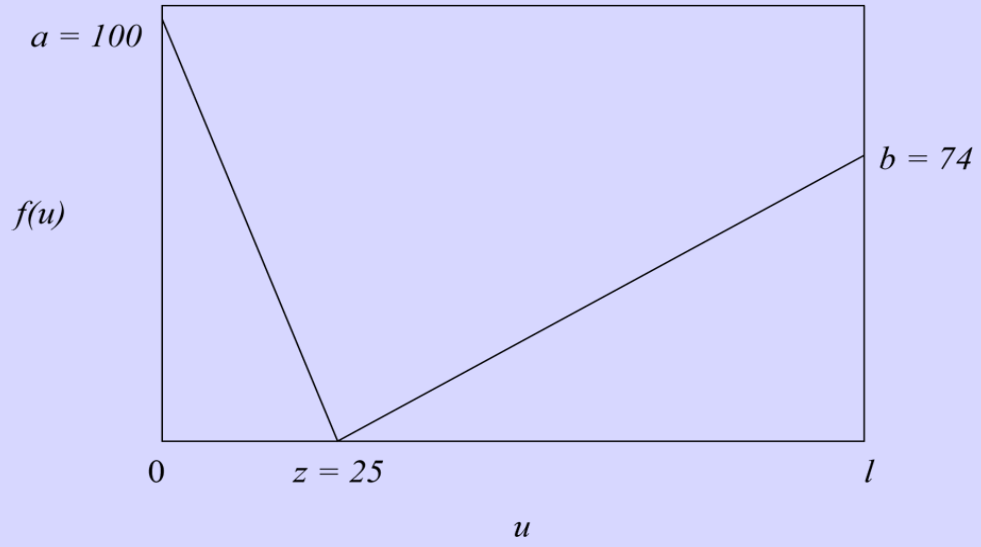
$$f(x) = \widehat{f}(u(x)) = \widehat{f}\left(\sum_{k=1}^{l} x_k\right)$$

- Verwenden zum Test eine „basic trap function“ und eine „complex trap function“

7. Testanwendung 3
1. Problemstellung

- basic trap function

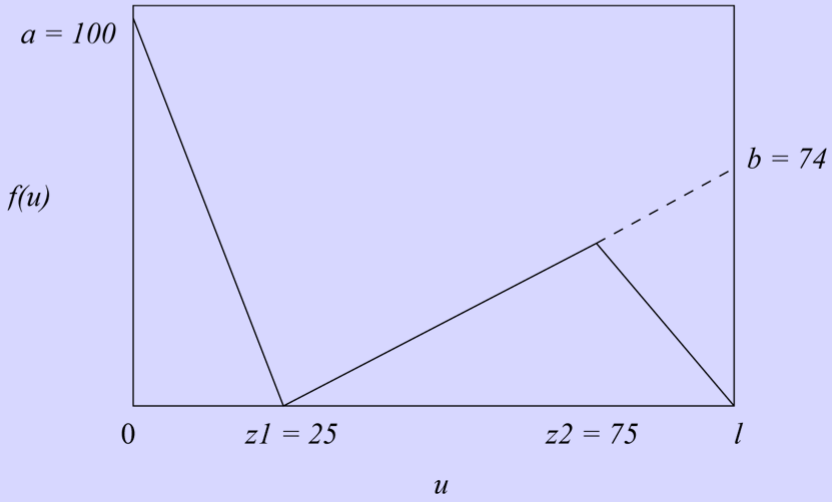
$$\hat{f}(u) = \begin{cases} \frac{a}{z}(z-u), & \text{if } u \leq z \quad ; \\ \frac{b}{l-z}(u-z), & \text{otherwise} \quad . \end{cases}$$



7. Testanwendung 3
1. Problemstellung

- complex trap function

$$\hat{f}(u) = \begin{cases} \frac{a}{z}(z_1 - u), & \text{if } u \leq z_1 \quad ; \\ \frac{b}{l - z_1}(u - z_1), & \text{if } z_1 < u \leq z_2 \quad ; \\ \frac{b(z_2 - z_1)}{l - z_1}(1 - \frac{1}{l - z_2}(u - z_2)), & \text{otherwise} \quad . \end{cases}$$



7. Testanwendung 3

2. Ergebnisse

- Folgende Parametersets würden verwendet:
  - I (l=10, z=3, a=12, b=6)
  - II (l=20, z=5, a=20, b=14)
  - III (l=50, z=10, a=80, b=39)
  - IV (l=75, z=20, a=80, b=54)
  - V (l=100, z=25, a=100, b=74)
  - Für complex trap: $z_1 = z$, $z_2 = l - z_1$

7. Testanwendung 3

2. Ergebnisse

Table 1. The best results obtained by IA with static, proportional, inversely proportional hypermutation and hypermacromutation with $M-$mutations ($M-mut$) strategy.

Trap	*Static*	*Proportional*	*Inversely*	*Hypermacro M-mut*
S(I)	**100**, *576.81*	**100**, *604.33*	**100**, *504.76*	**100**, *4334.94*
	$\tau_B = 25$, $c = 0.9$	$\tau_B = 100$, $c = 0.2$	$\tau_B = 5$, $c = 0.3$	$\tau_B = 1$
S(II)	**34**, *82645.85*	**100**, *50266.61*	**97**, *58092.70*	**5**, *5626.8*
	$\tau_B = 20$, $c = 1.0$	$\tau_B = 25$, $c = 0.8$	$\tau_B = 20$, $c = 0.2$	$\tau_B = 50$
S(III)	**0**	**0**	**0**	**5**, *174458.6*
				$\tau_B = 50$
S(IV)	**0**	**0**	**0**	**0**
S(V)	**0**	**0**	**0**	**0**
C(I)	**100**, *389.67*	**100**, *404.94*	**100**, *371.15*	**100**, *1862.09*
	$\tau_B = 100$, $c = 0.6$	$\tau_B = 50$, $c = 0.1$	$\tau_B = 10$, $c = 0.2$	$\tau_B = 5$
C(II)	**100**, *36607.15*	**100**, *22253.70*	**100**, *44079.57*	**56**, *84001.29*
	$\tau_B = 15$, $c = 0.1$	$\tau_B = 25$, $c = 0.3$	$\tau_B = 10$, $c = 0.2$	$\tau_B = 50$
C(III)	**1**, *15337.00*	**0**	**2**, *236484.50*	**1**, *185318*
	$\tau_B = 50$, $c = 0.7$		$\tau_B = 25$, $c = 0.1$	$\tau_B = 50$
C(IV)	**0**	**0**	**0**	**0**
C(V)	**0**	**0**	**0**	**0**

7. Testanwendung 3

2. Ergebnisse

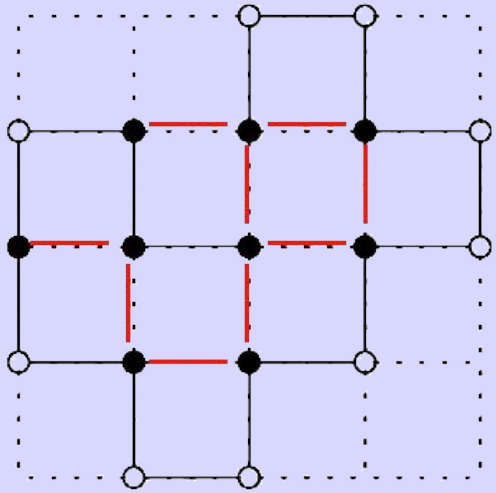
Table 2. The best results obtained by IA with hypermacromutation FCM, and combinations of static, proportional, inversely proportional hypermutation and hypermacromutation FCM.

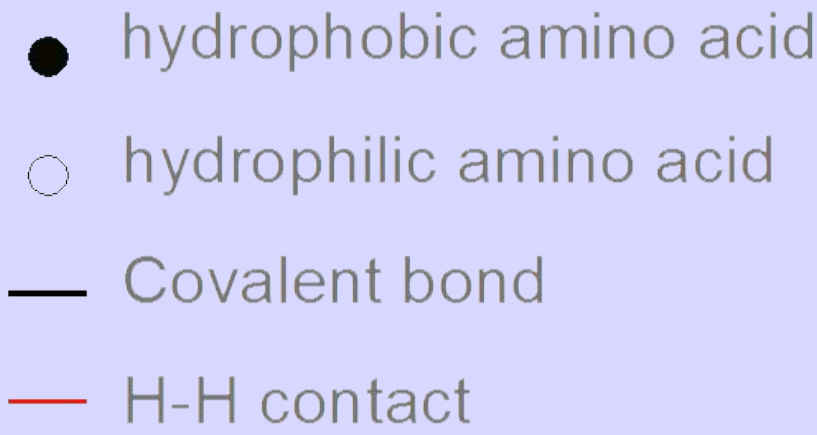
Trap	*Hypermacro FCM*	*Static+Macro*	*Proportional+Macro*	*Inversely+Macro*
S(I)	**100**, *1495.90*	**100**, *452.94*	**100**, *834.01*	**100**, *477.04*
	$\tau_B = 1$	$\tau_B = 25$, $c = 0.2$	$\tau_B = 50$, $c = 0.2$	$\tau_B = 15$, $c = 0.2$
S(II)	**28**, *64760.25*	**99**, *37915.17*	**100**, *51643.11*	**100**, *35312.29*
	$\tau_B = 1$	$\tau_B = 25$, $c = 0.1$	$\tau_B = 20$, $c = 0.7$	$\tau_B = 100$, $c = 0.2$
S(III)	**15**, *15677.53*	**100**, *18869.84*	**1**, *243038.00*	**100**, *20045.81*
	$\tau_B = 100$	$\tau_B = 50$, $c = 0.1$	$\tau_B = 20$, $c = 0.2$	$\tau_B = \infty$, $c = 0.1$
S(IV)	**24**, *40184.83*	**100**, *37871.71*	**0**	**100**, *42082.00*
	$\tau_B = 200$	$\tau_B = 25$, $c = 0.1$		$\tau_B = 25$, $c = 0.2$
S(V)	**27**, *139824.44*	**100**, *78941.79*	**0**	**100**, *80789.94*
	$\tau_B = 1$	$\tau_B = 100$, $c = 0.1$		$\tau_B = 50$, $c = 0.2$
C(I)	**100**, *826.78*	**100**, *367.67*	**100**, *644.67*	**100**, *388.42*
	$\tau_B = 50$	$\tau_B = 10$, $c = 0.1$	$\tau_B = 10$, $c = 0.1$	$\tau_B = 10$, $c = 0.2$
C(II)	**96**, *54783.25*	**100**, *24483.76*	**100**, *23690.82*	**100**, *29271.68*
	$\tau_B = 15$	$\tau_B = 10$, $c = 0.2$	$\tau_B = 50$, $c = 0.3$	$\tau_B = 5$, $c = 0.2$
C(III)	**39**, *112533.18*	**27**, *172102.85*	**1**, *147114.00*	**24**, *149006.5*
	$\tau_B = 15$	$\tau_B = 20$, $c = 0.1$	$\tau_B = 50$, $c = 0.1$	$\tau_B = 20$, $c = 0.1$
C(IV)	**5**, *227135.80*	**2**, *99259.00*	**0**	**2**, *154925.00*
	$\tau_B = 15$	$\tau_B = 25$, $c = 0.5$		$\tau_B = 15$, $c = 0.4$
C(V)	**2**, *353579.00*	**0**	**0**	**0**
	$\tau_B = 15$			

8. Testanwendung 4

1. Problemstellung

- 2D-HP Modell
- 2 Arten Aminosäuren:
  - Hydrophobe / Non Polar (H)
  - Hydrophile / Polar (P)
- Zielsetzung: Maximieren der H-H Kontakte für eine gegebene Sequenz

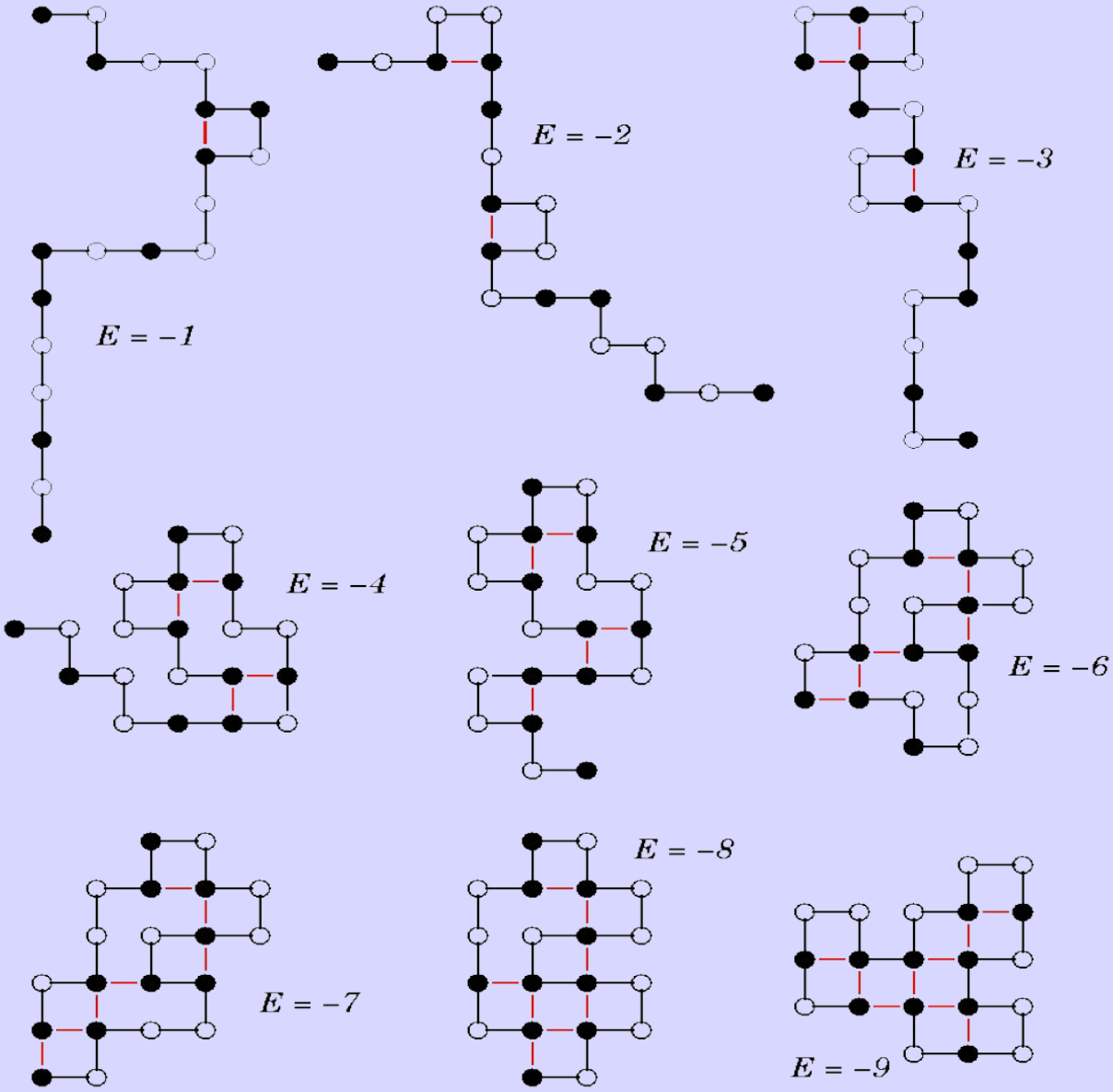




The above conformation has an energy of –9.

8. Testanwendung 4

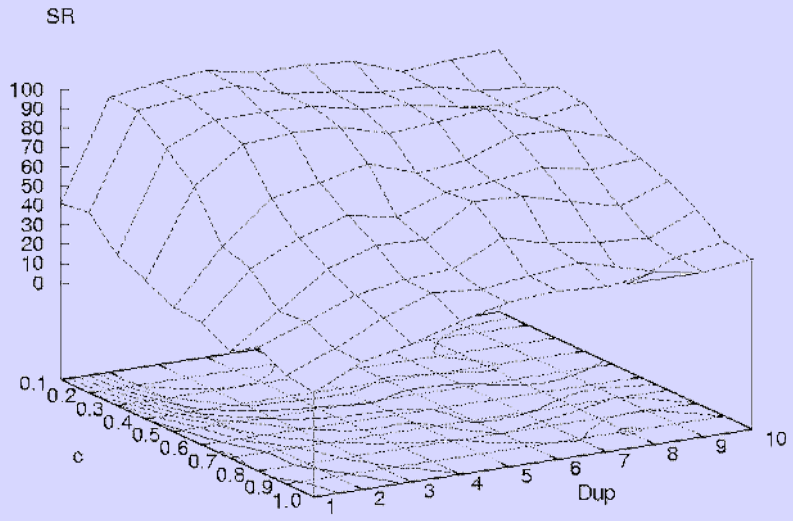
2. Beispiele



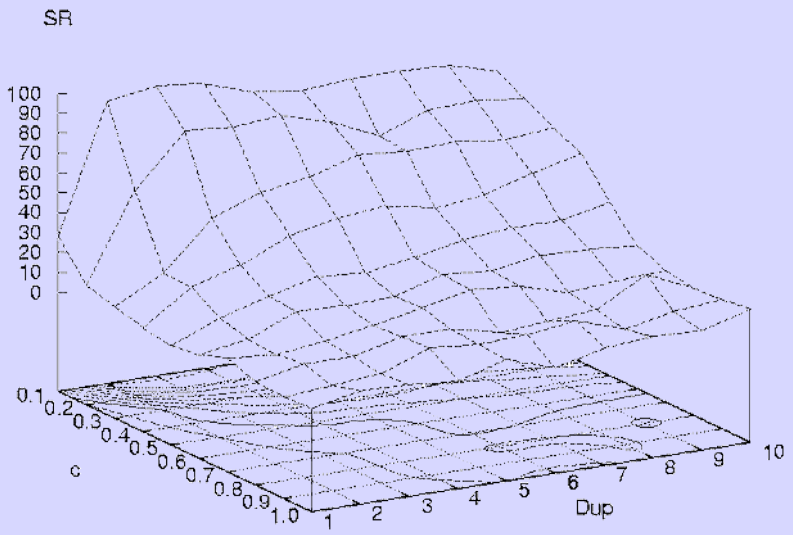
8. Testanwendung 4

3. Ergebnisse

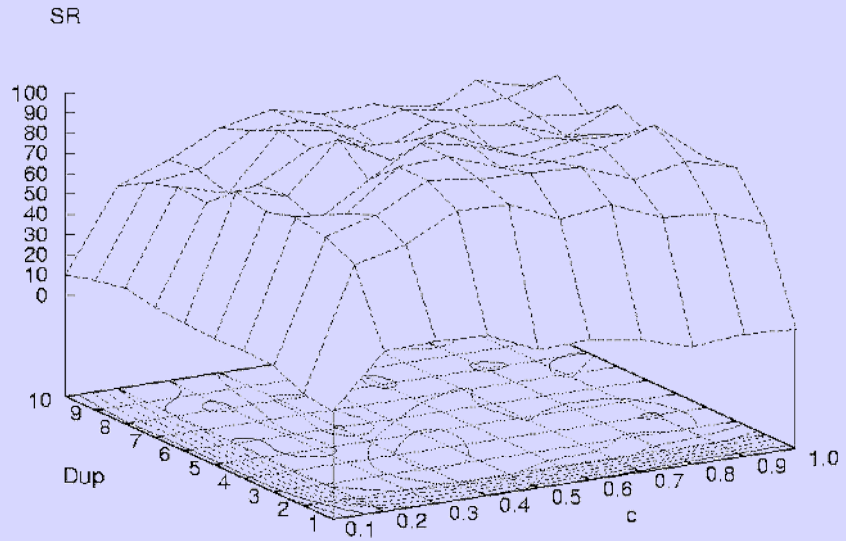
Static Hypermutation, d=10 tauB=5



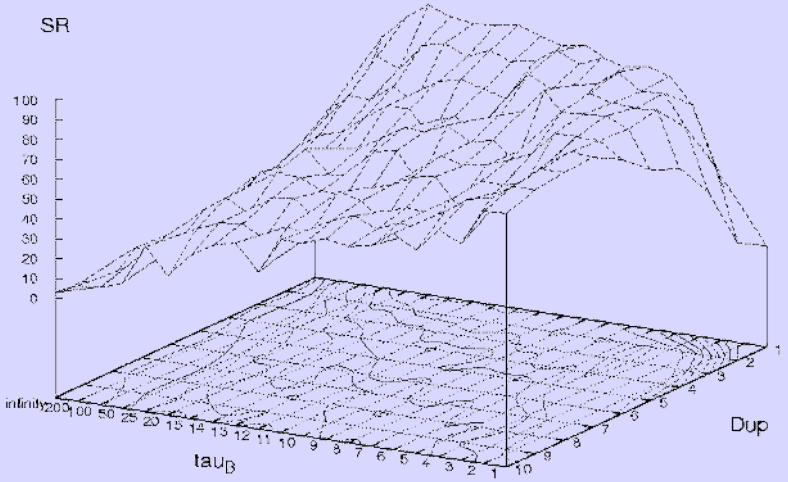
Proportional Hypermutation, d=10 tauB=5



Inversely Proportional Hypermutation, d=10 tauB=5



Macromutation, d=10



8. Testanwendung 4

4. Vergleich

Sequence *E**		1 - 9	2 - 9	3 - 8	4 - 14	5 - 23	6 - 21	7 - 36	8 - 42	9 - 10
New ACO	(Hoos et al., 03)	- 9	- 9	- 8	- 14	- 23	- 21	- 36	- 42	
CG	(Dill et al.,96)	- 9	- 9	- 8	- 14	- 23	- 21	- 35	- 42	
Tabu Search	(Dill et al.,03)	- 9	- 9	- 8	- 14	- 23	- 21		- 42	
IA		**- 9**	**- 9**	**- 8**	**- 14**	**- 23**	**- 21**	**- 35**	**- 39**	**- 10**
EMC	(Liang et al.,01))	- 9	- 9	- 8	- 14	- 23	- 21	- 35	- 39	
PERM	(Hsu et al.,03)	- 9	- 9	- 8	- 14	- 23	- 21	- 36	- 38	
MMA	(Krasnogor et al.,02)	- 9		- 8	- 14	- 22	- 21		- 39	- 10
CI	(Toma et al., 96)	- 9	- 9	- 8	- 14	- 23	- 21	- 35		
GA	(Unger et al.,93)	- 9	- 9	- 8	- 14	- 22	- 21	- 34	- 37	
Pioneer Search	(Konig et al.,99)	- 9			- 14	- 23			- 37	
ACO + LS	(Hoos et al., 02)	- 9	- 9	- 8	- 14	- 23	- 21	- 34	- 32	- 10
SC	(Konig et al.,99)	- 9			- 14	- 22			- 34	
Metropolis MC	(Unger et al.,93)	- 9	- 9	- 8	- 13	- 20	- 21	- 33	- 35	
only LS	(Hoos et al., 02)	- 9	- 9	- 8	- 14	- 21	- 20	- 33	- 33	- 10
Multiple MC	(Unger et al.,93)	- 9	- 9	- 7	- 13	- 19	- 20	- 32	- 32	
SGA	(Konig et al.,99)	- 9			- 14	- 20			- 32	

d= 500 -1500

d=10

The reported energy values are the lowest obtained by each method.

Noch Fragen???

Vielen Dank für Ihre Aufmerksamkeit.