Artificial Immune Systems for Optimisation

Thomas Jansen
Aberystwyth University
Aberystwyth SY23 3DB, UK
t.jansen@aber.ac.uk

Christine Zarges
University of Birmingham
Birmingham B15 2TT, UK
c.zarges@cs.bham.ac.uk

GECCO'13 Companion, July 6–10, 2013, Amsterdam, The Netherlands.
ACM 978-1-4503-1964-5/13/07.

Artificial Immune Systems are. . .

- a model of the natural immune system
  if you are interested in the natural immune system
- computational systems inspired by the natural immune system with natural applications in anomaly detection & classification
  if you are interested in solving a classification problem
- nature-inspired algorithms using the natural immune system as metaphor for problem-solving
  if you are interested in solving difficult problems
- nature-inspired randomised search heuristics like many others, e. g., evolutionary algorithms, ACO, SA, . . .
  if you are interested in randomised search heuristics
- a fascinating area of research
  in any case

Good News We cover all these aspects. ($\rightsquigarrow$ structure governed by this)

Plans for Today

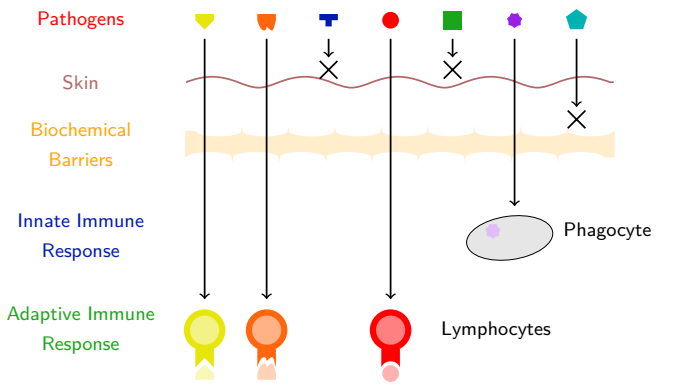
Biological Inspiration: The Immune System of Vertebrates

"The immune system recognizes infection and induces protective responses." [30]

Main Tasks

- Immunological recognition
- Immune effector functions
- Immune regulation
- Immunological memory

Multilayer Structure of the Immune System



(reproduced from [6, 35])

Innate vs. Adaptive Immunity

Innate Immunity

- Non-specific response against large number of bacteria
- First line of defense:
  Controls infection before adaptive immune response kicks in
- Initiates and controls adaptive immune response
- Dendritic cells form bridge between innate/adaptive immunity

Adaptive Immunity

- Specific and preventive immune response
- Mediated by lymphocytes in the lymph nodes
- Two main types: B cells and T cells
- Develops immunological memory
- Described by the Clonal Selection Theory

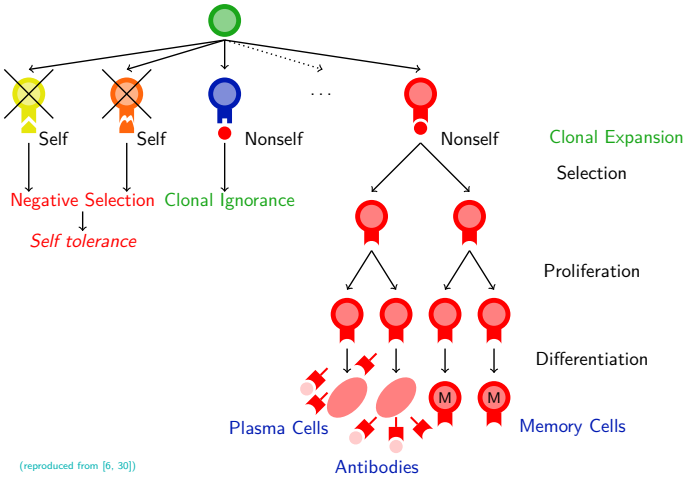
Adaptive Immunity – Immunological Recognition

- Naïve lymphocytes: not yet involved in an immune response
- Carry antigen receptors of single specificity
- Receptor diversity due to
  - Random recombination of gene fragments from several libraries
  - Somatic hypermutation to increase antigen-antibody affinity
- Become active due to interaction with antigenic stimulus
- Recognition based on complementarity between binding region of receptor and epitope of antigen on molecular level
- Antigens may present several epitopes
- Require co-stimulatory signals
- B cell receptor interacts directly while T cell receptor requires preprocessing and presenting by other cells

Clonal Selection Theory (1) [2]

- Describes basic properties of adaptive immune response
- Only cells recognizing an antigen proliferate and differentiate into effector cells
- B cells:
  - Subject to somatic hypermutations
  - B effector cells secrete antibodies
- T cells:
  - Not subject to mutation
  - T effector cells secrete lymphokine
- B cell clonal selection similar to natural selection
- Learning through increasing population size and affinity
- Immune repertoire evolved from a random base to reflection of actual antigenetic environment

Clonal Selection Theory (2) [2]

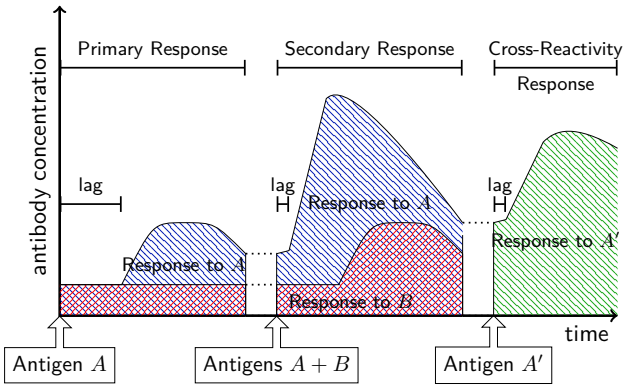


(reproduced from [6, 30])

Clonal Selection Theory (3) [2]

- Diversity via somatic hypermutations, receptor editing and newcomer cells
- Non-functional and harmful anti-self specifities are eliminated
- Variants with higher affinity dominate immune response and enter immune memory
- Some low affinity cells enter repertoire to maintain diversity
- Hypermutations
  - Point mutations allow for exploring local regions
  - On average one mutation per cell division
  - Short burst of somatic hypermutation followed by a pause to allow for selection and clonal expansion
  - Regulation of the hypermutation process by selection depending on receptor affinity
- Receptor editing
  - Instead of clonal deletion development of new receptors
  - Allows for larger steps through the landscape

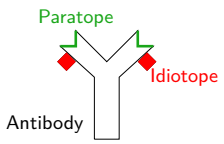
Immunological Memory and Cross-Reactive Response

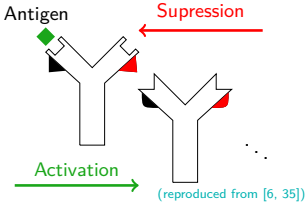


(reproduced from [6, 30])

Immune Network Theory [27]

- Remember Clonal Selection Theory:
  Immune system $\widehat{=}$ set of discrete cells and molecules originally at rest; triggered only by a foreign antigenic stimulation
- Now: different perspective Immune Network Theory
  Immune system $\widehat{=}$ regulated network of cells and molecules that recognize one another even in the absence of antigens
- Network is autonomous, self-regulated and aims at maintaining a specific range of activity
- Immune tolerance, learning and memory as inherent global properties

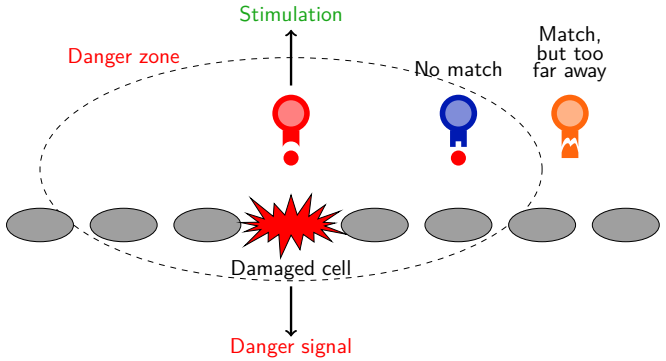




(reproduced from [6, 35])

Danger Theory [29]

Idea Immune system rather detects danger than nonself



(reproduced from [35])

Artificial Immune Systems as Classifiers

Remember artificial immune systems
inspired by natural immune system
$\rightsquigarrow$ 'perform self-nonself discrimination and react accordingly' most natural application

Observation self-nonself discrimination
$\widehat{=}$ two-class classification problem

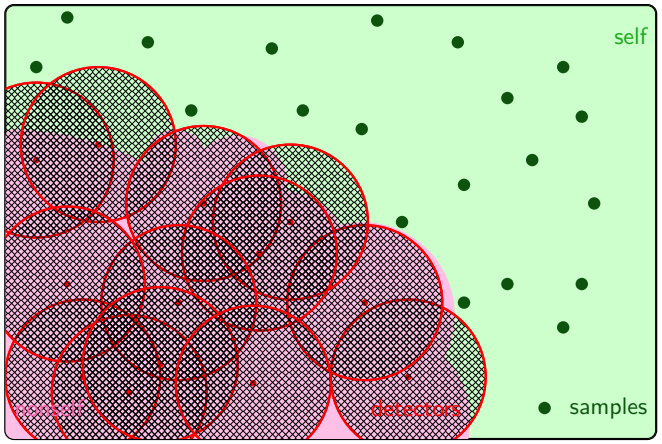
Fact many different AIS for this task
based on different immune principles

Today consider three examples

- negative selection (inspired by self-nonself discrimination)
- receptor density algorithm (inspired by T cell signalling)
- dendritic cell algorithm (inspired by the danger theory)

Self-Nonself Discrimination

Consider crude, unrealistic, partly misleading example



Simple Negative Selection Algorithm

Problem formal formulation
Input finite alphabet Σ, string space Σ^l,
training set $S \subseteq \Sigma^l$ of self strings
Output set of detectors D that match only self
by means of a partial match of length r

Algorithm works in two phases (outline)
Learning randomly generate detectors
keep those that do not match any $s \in S$
Detection mark everything that matches some $d \in D$
as nonself
(early algorithm, see e. g., Forrest et al. (1994) [12])

Fact very inefficient (for different types of detectors)
(see e. g., Stibor et al. (2004) [34])

Efficient Negative Selection

Algorithm outline, main ingredients
use prefix trees as main data structure
efficiently build finite automaton to represent detectors
(note: no explicit detector set)
construction of automaton works in time $O(|S| \cdot l \cdot r)$
classification works in time $O(l)$
(for details see Elberfeld, Textor (2011) [11])
(for implementation see
`http://bioinformatics.bio.uu.nl/textor/negativeselection.html`)

Lesson Learned immune metaphor useful for ideas
algorithmic implementation following the IS
may be very far from optimal
immune-ideas can be implemented efficiently
using 'classical' algorithmic ideas

More Modern AIS-Approaches to Classification

Remember self-nonself discrimination
based on a simplistic understanding of the immune system
can be implemented efficiently
using clever algorithms/data structures

Fact many more AIS-approaches to classification exist
based on different aspects of immunology
too many to cover all here

Today two current approaches
based on different immunological theories
current $\widehat{=}$ both considered and further developed
in current publications

1. based on T cell signalling: receptor density algorithm
2. based on danger theory: dendritic cell algorithm

Receptor Density Algorithm

Motivation two-class classification performed by T cells
depending on history

Basis model of T cell receptor called receptor
having a state c, position p, negative feedback n,
a negative feedback barrier β, length $l > \beta$
reacts to input u by
updating the position (adding u, subtracting n),
increasing negative feedback for positions above β,
decaying negative feedback otherwise
combining receptors spatially in form of a grid
with a stimulation kernel function
yields receptor density algorithm
(for details see Owens (2010) [32], Owens et al. (2013) [33])

Dendritic Cell Algorithm

Motivation danger theory specifically dendritic cells
model immune systems responds to danger/safe signals
(does not perform self-nonself discrimination)

Basis model of dendritic cells being either immature,
semi-mature or mature, having a lifespan
processing input classified as either danger, safe or PAMP,
computes two values:
DCM, indicating the amount of processed information,
K, indicating the classification as normal or anormal
a collection of such cells (with different lifespans) forms
dendritic cell algorithm performing classification
(for details see Greensmith (2007) [13])
fully formalised, simplified deterministic version
deterministic dendritic cell algorithm available
(for details see Gu (2011) [14], Gu et al. (2013) [15])

Artificial Immune Systems as Optimisers

Remember most natural application
$\widehat{=}$ classification, pattern recognition, IT security

Observation some algorithms also suitable for optimisation tasks

In particular clonal selection and immune network theory

Consider minimsation/maximisation of
pseudo-Boolean functions $f\colon \{0,1\}^n \to \mathbb{R}$ or
real-valued functions $f\colon \mathbb{R}^n \to \mathbb{R}$

Observation structure similar to evolutionary algorithms
concrete implementation very different

Today characteristics and concrete algorithms

- Mutation and Metadynamics in AIS
- Clonal Selection Algorithms: CLONALG, B-Cell Algorithm, opt-IA
- Immune Network: opt-aiNet

Additionally simple Java applet to 'play along'
`http://www.cs.bham.ac.uk/~zargesc/ais-tutorial/`

Mutation in Artificial Immune Systems (1)

Usually Mutations at high rate $\rightsquigarrow$ Hypermutations

Inverse Fitness-"Proportional" Hypermutation

- **Idea** Apply mutations with lower mutation rate to good search points
- **Usually** Normalised fitness value $\hat{f} \in [0,1]$ used
  - Using optimal function value f_{opt}: $\hat{f}(x) = f(x)/f_{\text{opt}}$
  - Using best known function value f_{best}: $\hat{f}(x) = f(x)/f_{\text{best}}$
- **Examples** for some parameter $\rho \in \mathbb{R}^+$, maximisation
  - CLONALG: $p_m = \exp(-\rho \cdot \hat{f})$
  - opt-aiNet: $p_m = (1/\rho) \cdot \exp(-\hat{f})$
- **Remark**
  In continuous optimisation p_m equals the mutation strength

Mutation in Artificial Immune Systems (2)

Contiguous Hypermutations

Idea Perform mutations with probability $r \in (0,1]$ only in contiguous region.

1. Choose hotspot $p \in \{0,\dots,n-1\}$, length $\ell \in \{0,1,\dots,n\}$.
   $p = 8$, $\ell = 4$

0	1	2	3	4	5	6	7	8	9

not wrapping around

0	1	2	3	4	5	6	7	8	9

wrapping around

2. Choose two hotspots $a, b \in \{0,\dots,n-1\}$.
   $a = 8$, $b = 5$

0	1	2	3	4	5	6	7	8	9

Mutation in Artificial Immune Systems (3)

Hypermutation with Mutation Potential

- **Idea** Determine number of local mutation steps during a single hypermutation.
- **Different** classes: static, inversely proportional, proportional
- **Example** for some constant $c \in]0,1[$, minimisation
  $M_c(v) = \lceil (1 - f_{\text{best}}/v) \cdot c \cdot n \rceil$
- Variants for the hypermutation of $x \in \{0,1\}^n$:
  tabu, stop at first constructive mutation
  1. Set $y := x$. Set $v := f(x)$.
  2. Repeat the following $M_c(v)$ times:
     - If tabu $= 0$ select $i \in \{1,\dots,n\}$ uniformly at random else select $i \in \{1,\dots,n\}$ uniformly at random, i not previously chosen.
     - $y[i] := 1 - y[i]$
     - If fcm $= 1$ and $f(y) < f(x)$ Then break

Metadynamics in Artificial Immune Systems

Sometimes worst search points replaced by new random ones.

Popular mechanism: Ageing

- Idea:
  Increase diversity by removing old and non-improving points
- General implementation:
  - Assign age to each search point
  - Increase age in each round
  - If new search point improves over parent
    Then assign age 0 to new search point
    Else inherit age of parent
  - Optional: Fill up population with new random search points
- Variants for some parameter $\tau_{\max}$
  - Static Pure Ageing
    remove search points older than $\tau_{\max}$
  - Stochastic Ageing
    remove each search point with probability $1 - 2^{-1/\tau_{\max}}$

CLONALG [8]

Parameters μ: population size d: selection pressure
β: offspring population size factor

1. Initialisation Create an initial population $P = \{x_1, x_2, \ldots, x_\mu\}$.
2. Clonal Selection and Expansion
   For all $i \in \{1, 2, \ldots, \mu\}$:
   a) Create $\lfloor \beta\mu \rfloor$ clones of x_i and place them in a clonal pool
   $C_i = \{y_i^1, \ldots, y_i^{\lfloor \beta\mu \rfloor}\}$.
   b) For all $j \in \{1, \ldots, \lfloor \beta\mu \rfloor\}$:
   Apply inversely fitness-proportional hypermutation to y_i^j.
3. Selection for Replacement
   Keep the μ best search points from $P \cup C_1 \cup \ldots \cup C_\mu$,
   breaking ties uniformly at random.
4. Metadynamics
   Replace d search points with lowest fitness by new random ones.
5. Stopping If stopping criterion not met continue at line 2.

Variants
1. Non-elist selection for replacement
2. Keep best search point from $x_i \cup C_i$ for all $i \in \{1, \ldots, \mu\}$

The B-Cell Algorithm (BCA) [28]

Parameters μ: population size λ: offspring population size

1. Initialisation Create an initial population $P = \{x_1, x_2, \ldots, x_\mu\}$.
2. Clonal Selection and Expansion
   For all $i \in \{1, 2, \ldots, \mu\}$:
   a) Create λ clones of x_i and place them in a clonal pool
   $C_i = \{y_i^1, \ldots, y_i^\lambda\}$.
   b) Select $j \in \{1, \ldots, \lambda\}$ uniformly at random:
   Flip each bit of y_i^j with probability p_m.
   c) For all $j \in \{1, \ldots, \lambda\}$:
   Apply somatic contiguous hypermutation to y_i^j.
3. Selection for Replacement
   For all $i \in \{1, 2, \ldots, \mu\}$:
   If $\min\{f(y_i^1), \ldots, f(y_i^\lambda)\} \leq f(x_i)$:
   Replace x_i by some randomly chosen y_i^j with minimal f-value.
4. Stopping If stopping criterion not met continue at line 2.

opt-IA [3, 4, 5]

Parameters μ: population size λ: offspring population size
H, M: flags for mutation operators

1. Initialisation Create an initial population $P = \{x_1, x_2, \ldots, x_\mu\}$.
2. Clonal Selection and Expansion
   For all $i \in \{1, 2, \ldots, \mu\}$:
   a) Create λ clones of x_i and place them in a clonal pool $C_i = \{y_i^1, \ldots, y_i^\lambda\}$.
   b) For all $j \in \{1, \ldots, \lambda\}$:
   If (H) Then Apply hypermutation with mutation potential to $y_i^j \rightsquigarrow C_i^H$.
   Else $C_i^H = \emptyset$.
   If (M) Then Apply contiguous hypermutations to $y_i^j \rightsquigarrow C_i^M$.
   Else $C_i^M = \emptyset$.
3. Metadynamics Apply aging to P, C_i^H, and C_i^M.
4. Selection for Replacement
   Set $P = P \cup C_1^H \cup \ldots \cup C_\mu^H \cup C_1^M \cup \ldots \cup C_\mu^M$.
   If $|P| \geq \mu$ Then Keep the μ best search points from P,
   breaking ties u.a.r. and removing duplicates.
   Else Keep all search points in P; fill up P with random points.
5. Stopping If stopping criterion not met continue at line 2.

opt-aiNet [7]

Parameters μ: initial population size λ: offspring population size
σ : affinity threshhold δ : average fitness threshold
d : selection pressure

1. Initialisation Create an initial population $P = \{x_1, x_2, \ldots, x_\mu\}$.
2. Clonal Selection and Expansion
   For all $i \in \{1, 2, \ldots, \mu\}$:
   a) Create λ clones of x_i and place them in a clonal pool $C_i = \{y_i^1, \ldots, y_i^{\beta\mu}\}$.
   b) For all $j \in \{1, \ldots, \lambda\}$: Apply inversely fitness-proportional mutation to y_i^j.
   c) For all $i \in \{1, 2, \ldots, \mu\}$: Keep the best search point from $x_i \cup C_i$.
3. Network Dynamics
   If change of average normalised fitness less than δ
   Then Calculate pairwise affinity $d(x_i, x_j)$ of all search points.
   If $d(x_i, x_j) < \sigma$ Then remove worse search point. Update μ.
   Else continue at line 2
4. Metadynamics Introduce $\lfloor d\mu \rfloor$ new search points in the network.
5. Stopping If stopping criterion not met continue at line 2.

Variants Non-elist selection for replacement
Remark Population size not fixed during optimisation

Analysing Artificial Immune Systems as Optimisers

Observation artificial immune systems as optimisers
are randomised search heuristics used for optimisation
just as evolutionary algorithms, ant colony optimisation,
particle swarm optimisation, simulated annealing,
iterated random local search, ...

Consequence AIS as optimisers should be considered
the same way as other RSH as optimisers
$\widehat{=}$ applied as other RSH
analysed as other RSH

Fact analysis of RSH as optimisers is important topic

introduction to runtime analysis	Pietro Oliveto, Per Kristian Lehre
bio-inspired computation	Frank Neumann, Carsten Witt
black-box complexity	Benjamin Doerr, Carola Doerr

Analysing Artificial Immune Systems

Why?

Because 'gaining a better understanding'

- of general limitations (black-box complexity)
- of behaviour in typical situations (example functions)
- of impact of specific operators (operators in (1+1)-frame)
- of parameter settings (simple algorithms with 1 parameter)
- for particular problem classes (classes of functions; combinatorial optimisation problems)

Because 'design of better randomised search heuristics'

- know when not to apply
- have an idea of when to apply
- have an idea of 'good' operators
- have an idea of 'good' parameter values
- have an idea of what kind of RSH
- ...

Analysing Randomised Search Heuristics – What?

Observation most important efficiency
$\widehat{=}$ time

Measuring time in randomised search heuristics

counting what	advantage	disadvantage	remark
computation steps	very precise	very tedious	rarely done (see [24])
function eval.	often good enough easier to handle	not exactly time still tedious	very common
rounds	convenient	inprecise can be misleading	very common

Usually count X until optimum found
$\widehat{=}$ optimisation time (RV $\rightsquigarrow$ expectation ...)

Sometimes count X until good enough solution found
$\widehat{=}$ approximation time (RV $\rightsquigarrow$ expectation ...)

Alternative analyse solution quality after X
$\widehat{=}$ fixed budged compuation (RV $\rightsquigarrow$ expectation ...)
(see [10], [25])

Analysing Randomised Search Heuristics

Artificial Immune Systems?
Yet another class of Randomised Search Heurisitcs?
Why should I care?

Facts artificial immune systems offer

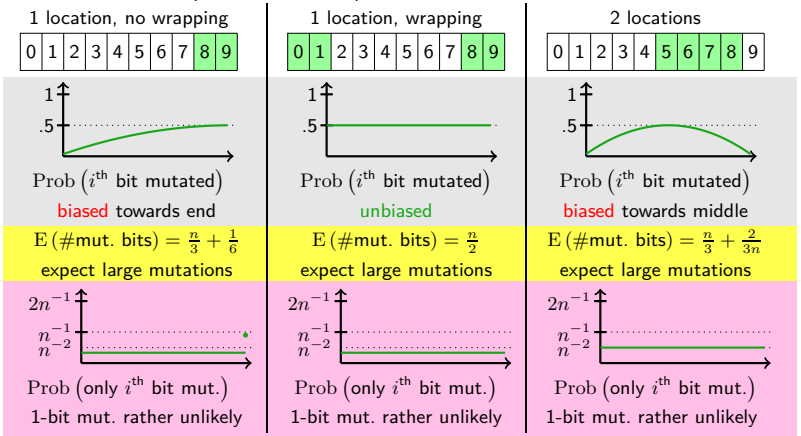
- useful alternative design paradigm for RSHs
- have different operators with different properties
  ⇝ useful in different situations
- can be a simpler and at least equally efficient alternative to crossover-based EAs

Now

- overview of three different types of AIS-specific mutation
- considering ageing as example for a 'meta-dynamic'
- example of a complete AIS in combinatorial optimisation

Contiguous Hypermutations

Remember (here with $r = 1$)



Analysing Contiguous Hypermutations

Method

- Insert mutation operator in (1+1)-framework.
  ⇝ study of effects in isolation
- Prove general observations.
  ⇝ knowledge of general properties
- Compare with (1+1) EA (with mutation probability $1/n$) on well-known example functions.
  ⇝ assessment of effects under well-known circumstances
- Find examples with extremely differing performance.
  ⇝ clear understanding of benefits and drawbacks

Observation method not unique to contiguous hypermutations but generally applicable for study of operators

Remember use Java applet, try for yourself
`http://www.cs.bham.ac.uk/~zargesc/ais-tutorial/`

Results About Contiguous Hypermutations (Part 1)

General Observation $\forall f$ with unique global optimum:

$$\mathrm{E}\left(T_{\mathsf{CHM}_{1,\text{no w.},f}}\right) = \Omega(n)$$
$$\mathrm{E}\left(T_{\mathsf{CHM}_{1,\text{w.},f}}\right) = \Omega(n^2)$$
$$\mathrm{E}\left(T_{\mathsf{CHM}_{2},f}\right) = \Omega(n^2)$$

due to probability of final mutation
(all bounds tight)

Comparison for $\textsc{OneMax}(x) = \sum_{i=1}^{n} x[i]$

$$\mathrm{E}\left(T_{(1+1)\ \mathsf{EA},\textsc{OneMax}}\right) = \Theta(n \log n)$$
$$\mathrm{E}\left(T_{\mathsf{CHM}_{1,\text{no w.}},\textsc{OneMax}}\right) = O(n^2 \log n)$$
$$\mathrm{E}\left(T_{\mathsf{CHM}_{1,\text{w.}},\textsc{OneMax}}\right) = \Theta(n^2 \log n)$$
$$\mathrm{E}\left(T_{\mathsf{CHM}_{2},\textsc{OneMax}}\right) = \Theta(n^2 \log n)$$

due to difficulty of making 1-bit improvements

Results About Contiguous Hypermutations (Part 2)

Comparison for $\textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x[j]$

$\mathrm{E}\left(T_{\text{(1+1) EA},\textsc{LeadingOnes}}\right) = \Theta(n^2)$

$\mathrm{E}\left(T_{\mathsf{CHM}_{1,\text{no w.}},\textsc{LeadingOnes}}\right) = O(n^2)$

$\mathrm{E}\left(T_{\mathsf{CHM}_{1,\text{w.}},\textsc{LeadingOnes}}\right) = \Theta(n^2 \log n)$

$\mathrm{E}\left(T_{\mathsf{CHM}_2,\textsc{LeadingOnes}}\right) = \Theta(n^2 \log n)$

since only position of left-most flipping bit matters

Comparison for $n \cdot \textsc{LeadingOnes}(x) - \textsc{OneMax}(x)$, init. in 0^n

$\mathrm{E}\left(T_{\text{(1+1) EA},\textsc{LeadingOnes}}\right) = \Theta(n^2)$

$\mathrm{E}\left(T_{\mathsf{CHM}_{1,\text{no w.}},\textsc{LeadingOnes}}\right) = O(n)$

$\mathrm{E}\left(T_{\mathsf{CHM}_{1,\text{w.}},\textsc{LeadingOnes}}\right) = O(n^2 \log n)$

$\mathrm{E}\left(T_{\mathsf{CHM}_2,\textsc{LeadingOnes}}\right) = \Theta(n^2)$

since sequence of 0-bits at end only significant advantage if improving mutations easy to find

Results About Contiguous Hypermutations (Part 3)

Demonstrate very large performance difference
to demonstrate understanding of benefits and drawbacks

$$\mathsf{CLOB}_{b,k}(x) = n \cdot \left(\sum_{h=1}^{k} \sum_{i=1}^{n/(bk)} \prod_{j=1}^{i \cdot b} x\left[(h-1)\cdot(n/k)+j\right] \right) - \mathsf{OneMax}(x)$$

Example with $n = 24$, $k = 3$, $b = 2$

$\mathsf{CLOB}_{2,3}(x) = 24 \cdot (1+1+2) - 17 = 79$

Comparison for $\mathsf{CLOB}_{b,k}$ (with $n/(k \cdot b) \in \mathbb{N}$, $l := n/k$)

$\mathrm{E}\left(T_{\text{(1+1) EA},\mathsf{CLOB}_{b,k}}\right) = \Theta(k \cdot l^b \cdot (l/b + \log k))$

$\mathrm{E}\left(T_{\mathsf{CHM},\mathsf{CLOB}_{b,k}}\right) = O(n^2 \log n)$ (all 3 variants)

since length of block does not matter

Summary Contiguous Hypermutations

- difficulties with flipping single bits
  ⇝ bad at locating optima precisely
  ⇒ combine with other operators if locating optima precisely matters
- in expectation mutate $\Theta(n)$ bits
  ⇝ advantages when huge mutations are needed
  ⇒ worth a try when hill-climbing not effective
- some variants with strong positional bias
  ⇝ advantages/disadvantages depending on function
  ⇒ only use variants with positional bias if known facts about objective function make that appear useful
- all noticeable effects rely on $r \approx 1$
  ⇝ even $r = 1 - \varepsilon$ ($\varepsilon > 0$ constant) not useful
  ⇒ use $r = 1 - o(1)$, e. g., $r = 1 - 1/n$

(for details see Jansen/Zarges (2011) [20])

Hypermutations with Mutation Potential

Remember Hypermutation(x) (for $x \in \{0,1\}^n$, minimise f)

1. number of mutations steps $m(f(x)) := \left\lceil \left(1 - \frac{f_{\text{opt}}}{f(x)}\right) \cdot c \cdot n \right\rceil$
2. Repeat m times
   If tabu=false then select $i \in \{1,2,\ldots,n\}$ u. a. r.
   Else select $i \in \{1,2,\ldots,n\}$ not previously chosen u. a. r.
3. local mutation: $x[i] := 1 - x[i]$

Consider four variants

- $\mathsf{MP}_{\text{no tabu, blind}}$ (as above, tabu=false)
- $\mathsf{MP}_{\text{tabu, blind}}$ (as above, tabu=true)
- $\mathsf{MP}_{\text{no tabu}}$ (tabu=false, evaluate and stop at first improvement)
- $\mathsf{MP}_{\text{tabu}}$ (tabu=true, evaluate and stop at first improvement)

Results About Hypermut. with Mutation Potential (Part 1)

Comparison for $\textsc{ZeroMin}(x) = n + 1 - \textsc{OneMax}(x)$

$\mathrm{E}\left(T_{(1+1)\text{ EA},\textsc{ZeroMin}}\right) = \Theta(n \log n)$

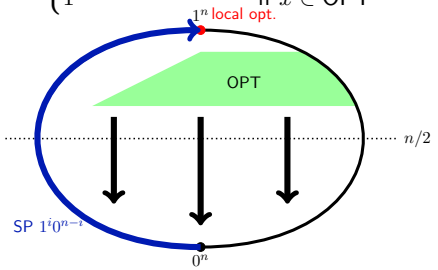
$\mathrm{E}\left(T_{\text{MP}_{\text{no tabu, blind}},\textsc{ZeroMin}}\right) = 2^{\Omega(n)}$
(even with high probability)
$\mathrm{E}\left(T_{\text{MP}_{\text{tabu, blind}},\textsc{ZeroMin}}\right) = 2^{\Omega(n)}$
(even with high probability)
due to drift to middle (due to blindness)

$\mathrm{E}\left(T_{\text{MP}_{\text{no tabu}},\textsc{ZeroMin}}\right) = \Theta(n^2 \log n)$
$\mathrm{E}\left(T_{\text{MP}_{\text{tabu}},\textsc{ZeroMin}}\right) = \Theta(n^2 \log n)$
due to additional evaluations

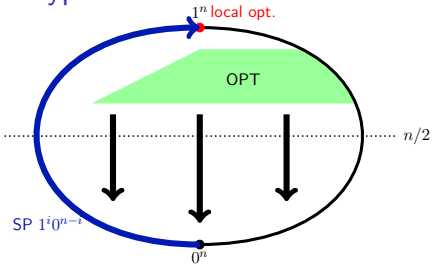
Results About Hypermut. with Mutation Potential (Part 2)

Show large difference with example function f using
short path $\text{SP} = \{1^i 0^{n-i} \mid 0 \le i \le n\}$
optima $\text{OPT} = \{x \mid \text{ZeroMin}(x) < n/4, \text{dist}(x, \text{SP}) \ge \gamma n\}$
$(\gamma = \omega(1/n),\ \gamma < 3/20)$

$$f(x) = \begin{cases} 3n - \text{ZeroMin}(x) & \text{if } x \notin (\text{SP} \cup \text{OPT}) \\ 2n - i & \text{if } x \in \text{SP} \\ 1 & \text{if } x \in \text{OPT} \end{cases}$$



Results About Hypermut. with Mutation Potential (Part 3)



Results for f with $\gamma = \Theta(1)$
$\mathrm{E}\left(T_{(1+1)\text{ EA},f}\right) = 2^{\Omega(n)}$ (even with high probability)
due to 'large distance'
$\mathrm{E}\left(T_{\text{MP}_{\text{tabu}},f}\right) = O(n^3)$
due to $\Theta(n^2 \log n)$ to reach 0^n,
$O(n^3)$ to reach 1^n, $O(n)$ for drifting down

Summary Hypermutations with Mutation Potential

- blind variants difficulties locating specific points
  - ⇝ bad at locating optima precisely
  - ⇒ combine w. other operators if precise hits matter
- blind variants performs mostly blind random walk
  - ⇝ hardly ever useful
  - ⇒ if used at all, only in combination with other operators
- first improvement version can do local search (less efficient)
  - ⇝ no replacement for local search/standard bit mutations
  - ⇒ prefer local search if you want local search
- first improvement version can locate remote optimal regions
  - ⇝ useful for such objective functions
  - ⇒ use as costly alternative to local search/standard bit mutation if such properties are suspected
- depends heavily on actual function values
  - ⇝ sensitive with respect to trivial transformations
  - ⇒ prefer rank-based variants

(for details see Jansen/Zarges (2011) [21])

Inverse Fitness-"Proportional" Hypermutations

Remember

Normalisation
- opt: $\hat{f}(x) = f(x)/f_{\text{opt}} \in [0,1]$
- best: $\hat{f}(x) = f(x)/f_{\text{current best}} \in [0,1]$

Mutation probabilities
- CLONALG: $e^{-\rho \hat{f}(x)}$
- opt-aiNet: $e^{-\hat{f}(x)}/\rho$

resulting in four variants

- CLONALG$_{\text{opt}}$
- CLONALG$_{\text{best}}$
- opt-aiNet$_{\text{opt}}$
- opt-aiNet$_{\text{bext}}$

Results About Inverse Fitness-"Prop." Hypermut. (Part 1)

Results for OneMax

$\mathrm{E}\left(T_{\text{(1+1) EA,OneMax}}\right) = \Theta(n \log n)$

$\mathrm{E}\left(T_{\text{CLONALG}_{\text{opt}},\text{OneMax}}\right) = 2^{\Omega(n)}$ for $\rho = O(1)$

(even with high prob.) since mutation probability too large

$\mathrm{E}\left(T_{\text{CLONALG}_{\text{opt}},\text{OneMax}}\right) = 2^{\Omega(n)}$ for $\rho = \Omega(n)$

(even with high prob.) since mutation probability too small

$\mathrm{E}\left(T_{\text{CLONALG}_{\text{opt}},\text{OneMax}}\right) = 2^{\Omega\left(n^{.5-\varepsilon}\right)}$ for $\rho = \ln n$

(even with high prob.)

but $O(n \log n)$ once $\text{OneMax}(x) = n - O(n/\log n)$

$\mathrm{E}\left(T_{\text{CLONALG}_{\text{best}},\text{OneMax}}\right) = \Theta(\mu n + n \log n)$ for $\rho = \ln n$

using population of size μ

$\mathrm{E}\left(T_{\text{opt-aiNet}_{\text{opt}},\text{OneMax}}\right) = 2^{\Omega(n)}$ for $\rho = 1$

(even with high prob.) since mutation probability too large

$\mathrm{E}\left(T_{\text{opt-aiNet}_{\text{opt}},\text{OneMax}}\right) = \Theta(n \log n)$ for $\rho = \Theta(n)$

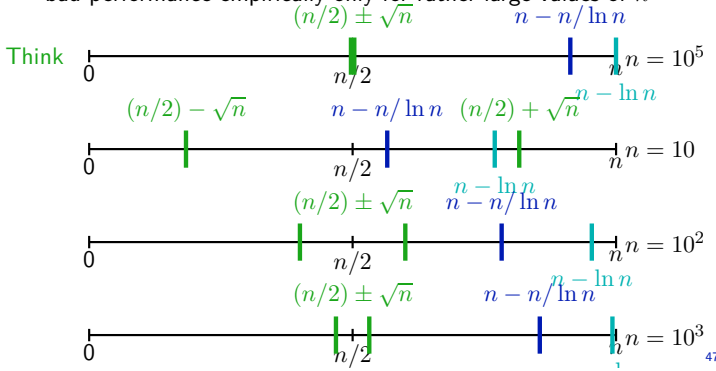
Results About Inverse Fitness-"Prop." Hypermut. (Part 2)

Remember CLONALG$_{\text{opt}}$ inefficient even with $\rho = \ln n$

How is this possible in practice?

- under-estimating opt improves (see CLONALG$_{\text{best}}$)
- OneMax not necessarily realistic
- bad performance empirically only for rather large values of n

Think



Summary Inverse Fitness-"Proportional" Hypermutations

- can be very inefficient in simple situations
  - ⇝ e. g., bad at hill climbing
  - ⇒ use only when needed
- using 'current best' appears superior to 'optimal value' for normalisation
  - ⇝ populations useful
  - ⇒ prefer population-based approaches and 'current best' for normalisation
- CLONALG very sensitive with respect to ρ
  - ⇝ very bad performance easy to achieve
  - ⇒ prefer opt-aiNet
- only analytical results for OneMax
  - ⇝ most points open
  - ⇒ investigate more

(for details see Zarges (2008), (2009), (2011) [36, 37, 38])

Metadynamics in Artificial Immune Systems

Remember metadynamics influence behaviour of algorithm in a more global way
$\rightsquigarrow$ more difficult to analyse than an operator
Example ageing

Remember ageing
has parameter maximal age $\tau_{\max}$
comes in different variants (static pure, stochastic, ...)
depends non-trivially on implementation details

Remember method for analysis/work programme

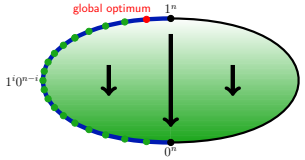
- insert in simple algorithmic framework
- prove general observations
- compare with known algorithms on known problems
- find extreme examples to understand benefits and drawbacks

A Simple Framework for Ageing

- use population of μ search points
  Reason ageing only effective in populations
- increase age of all search points deterministically in each round
  Reason most commonly used ageing variant
- create only one new search point per round, using a well understood variation operator
  Reason introduce as little other complexity as possible
- implement ageing variant as simple as possible
  Static Pure Ageing
  1. new search point gets age 0 in case of an improvement, otherwise inherits age.
  2. remove all search points exceeding $\tau_{\max}$; fill population with new random search points as needed

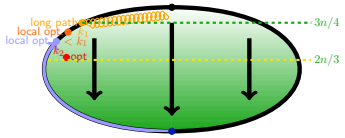
Parameter Study: The Maximal Age

Note maximal age $\tau_{\max}$ must not be too small



$\tau_{\max} = o\big(n^k \log n\big)$
$\rightsquigarrow$ very inefficient
$\tau_{\max} = \omega\big(\log n(n^k + \mu \log n)\big)$
$\rightsquigarrow$ efficient

Note maximal age $\tau_{\max}$ must not be too large



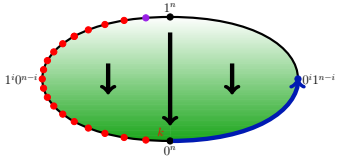
$\tau_{\max} = \Omega\big(\log n(n^{k_1} + \mu \log n)\big)$
$\rightsquigarrow$ very inefficient
$\tau_{\max} = O\big(n^{k_1 - k_2}\big)$ and
$\tau_{\max} = \omega\big(\log n(n^2 + \mu n \log n)\big)$
$\rightsquigarrow$ efficient

See appropriate range for $\tau_{\max}$ can be extremely narrow
$\tau_{\max} = o\Big(n^k \log n\Big)$ or $\tau_{\max} = \Omega\Big(n^{k_1} \log n\Big)$ $\rightsquigarrow$ very inefficient
$\tau_{\max} = \omega\Big(n^k \log n + \mu n \log n\Big)$ and $\tau_{\max} = O\Big(n^{k_1 - k_2}\Big)$ $\rightsquigarrow$ efficient

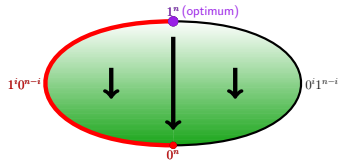
(for details see Horoba, Jansen, Zarges (2009) [17])

Comparing Ageing Variants

Static Pure Ageing
new search points get age 0
if they improve
superior in local optima



Evolutionary Ageing
new search points get age 0
always
superior on plateaus



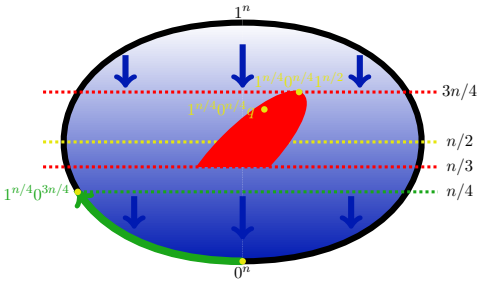
Combine both into genotypic ageing
'new search points get age 0 unless they are copy or worse'
combines advantages, good on plateaus and at local optima

(for details see Jansen, Zarges (2011) [22])

Understanding Specific Benefits

Observation ageing performs restarts in a complicated way
And nothing more?

Idea ageing can perform partial restart
e. g, useful when crossover combines new and old search points



(for details see Jansen, Zarges (2010) [19])

Paying Attention to Details

Remember keep algorithmic framework as simple as possible
since every bit of complexity complicates things a lot

Still pay close attention to each (innocent looking) detail
since it may be very important

Now one small example in the context of ageing

Remember one new search point per round
replacing one of the μ other search points
if it is not worse than the worst and none died of age

Which one is replaced?

Obvious Answers one worst search point and among those

1. a random one
2. one with min. age distance from new one
3. one with most frequent age
4. one with rarest age

And this makes a difference?

Paying Attention to Details (cont.)

Ageing Variants replace worst search point and among those

1. a random one
2. one with min. age distance from new one
3. one with most frequent age
4. one with rarest age

1. $\mathrm{E}(T) = 2^{\Omega(n)}$ even with high probability
2. $\mathrm{E}(T) = O((\mu + (n/\log\mu)) \cdot (\tau_{\max} + n^2 + \mu n \log n))$
   $\mathrm{E}(T) = \Omega((1 + n/(\mu \log \mu)) \cdot (\tau_{\max} + n^2 + \mu n \log n))$
3. $\Theta((1 + n/(\mu \log \mu)) \cdot (\tau_{\max} + n^2 + \mu n \log n))$
4. $\mathrm{E}(T) = 2^{\Omega(n)}$ even with high probability

(for details see Jansen, Zarges (2011) [23])

Summary Ageing

- ageing adds new dynamics and new capabilities
  $\rightsquigarrow$ increased potential at the price of additional parameter
  $\Rightarrow$ use with care
- ageing very sensitive with respect to maximal age
  $\rightsquigarrow$ difficult to set additional parameter
  $\Rightarrow$ perform careful parameter study
- different ageing variants have different capabilties
  $\rightsquigarrow$ no 'one size fits all' solution
  $\Rightarrow$ try different variants
- ageing very sensitive with respect to implementation details
  $\rightsquigarrow$ algorithmic details need to be reported precisely
  $\Rightarrow$ pay attention to details, communicate choices precisely

Analysing AIS in Combinatorial Optimisation

Fact analysis for example functions
(either commonly used or specifically designed)
important first step in understanding, not end of story

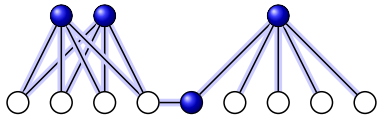
Observation more interesting, relevant, realistic
analysis on combinatorial optimisation problems
(see Neumann, Witt (2010) [31] for EAs)

Fact has been started for AIS, too
namely for the B-Cell algorithm
for Vertex Cover and Longest Common Subsequences

The Vertex Cover Problem (VC)

Input undirected Graph $G = (V, E)$
Output smallest subset $V' \subseteq V$ covering all edges, i. e.
$V' \subseteq V$ with $\forall e \in E \colon e \cap V' \neq \emptyset$

Example



Facts

- "classic" NP hard optimisation problem
- simple 2-approximation algorithm
- no 1.3606-approximation (if P $\neq$ NP)
(no $(2 - \varepsilon)$-approximation under stronger assumptions)

Summary Vertex Cover

- BCA alternative without crossover to EAs
- Ordering heuristic for encoding instead of "cheating" possible
- Known analyses for EAs reproducible for BCA
- On complete bipartite graph more efficient than (1+1) EA;
only slightly more inefficient than (1+1) EA with restarts
- On amplified complete bipartite graphs considerably more efficient than mutation-based EAs
- No need for crossover and population on example graph
- Difficult to find hard instances with "bad approximation ratio with high probability"
- BCA alternative to EAs with respect to efficiency;
easier to analyse
- for details see [18]

The Longest Common Subsequence Problem

Input m sequences $X_1, X_2, \ldots, X_m \in \Sigma^*$
Output common subsequence Y
with $\forall Y' \in \Sigma^* \colon Y'$ is common subsequence $\Rightarrow |Y'| \leq |Y|$

Examples

- Finite alphabet Σ: $\Sigma = \{0, 1\}$, $\Sigma = \{\mathsf{A}, \mathsf{C}, \mathsf{G}, \mathsf{T}\}$
- Finite sequences $\in \Sigma^*$: $X_1 = ACTGTGCAA$
- Subsequences of a sequence:
$AGTA$ of $ACTGTCAA$

Facts

- General case is NP hard
- In P with fixed m
- Solvable using dynamic Programming in $O\left(m \cdot \prod_{i=1}^{m} |X_i|\right)$

Summary Longest Common Subsequence

- Another comparison of EAs and AIS on a combinatorial optimisation problem
- Reconsideration of previous analyses for EAs
- EAs and BCA with random initialisation very inefficient
- EAs do not benefit from deterministic initialisation with empty solutions
- B-Cell algorithm clearly benefits from deterministic initialisation
- Further example where AIS excel EAs
- For details see [26]

Summary

We have seen overviews and introductions of

- the natural immune system
- application of AIS as classifiers
- application of AIS as optimisers
- analysis of AIS as optimisers

all as invitation to

- learn more about AIS
- apply AIS
- explore and understand AIS

Conclusions

Artificial Immune Systems are

- models of the natural IS $\widehat{=}$tool for research in immunology
- heuristic approach to classification based on an example of complex classification from nature
- randomised search heuristics capable of optimisation
  - based on a quite different natural metaphor (compared to EAs)
  - an alternative approach to optimisation, with different characteristics and capabilities
  - an alternative solution if your favourite approach fails
- randomised search heuristics like many others
  - another field of study, worthy of analysis just like EAs/ACO/PSO/...
  - another example of a complex class of RSHs $\widehat{=}$ another opportunity to study differences and similarities hopefully some day leading to useful taxonomy
- a fascinating area of research

References I

[1] J. Brownlee (2011).
Clever Algorithms: Nature-Inspired Programming Recipes.
LuLu Enterprises. http://www.cleveralgorithms.com

[2] F. M. Burnet (1959).
The Clonal Selection Theory of Acquired Immunity.
Cambridge University Press.

[3] V. Cutello, G. Narzisi, G. Nicosia, and M. Pavone (2005).
Clonal selection algorithms: A comparative case study using effective mutation potentials.
In C. Jacob, M. L. Pilat, P. J. Bentley, and J. Timmis, editors, *Proceedings of the 4rd International Conference on Artificial Immune Systems (ICARIS 2005)*, volume 3627 of *Lecture Notes in Computer Science*, pages 13–28. Springer.

[4] V. Cutello, G. Nicosia, and M. Pavone (2004).
Exploring the capability of immune algorithms: A characterization of hypermutation operators.
In G. Nicosia, V. Cutello, P. J. Bentley, and J. Timmis, editors, *Proceedings of the 3rd International Conference on Artificial Immune Systems (ICARIS 2004)*, volume 3239 of *Lecture Notes in Computer Science*, pages 263–276. Springer.

[5] V. Cutello, G. Nicosia, M. Pavone, and J. Timmis (2007).
An immune algorithm for protein structure prediction on lattice models.
IEEE Transactions on Evolutionary Computation, 11(1):101–117.

[6] L. N. de Castro and J. Timmis (2002a).
Artificial Immune Systems: A New Computational Intelligence Approach.
Springer.

[7] L. N. de Castro and J. Timmis (2002b).
An artificial immune network for multimodal function optimization.
In *Proceedings of the 4th IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 699–704. IEEE Press.

References II

[8] L. N. de Castro and F. J. Von Zuben (2002).
Learning and optimization using the clonal selection principle.
IEEE Transactions on Evolutionary Computation, 6(3):239–251.

[9] D. Dasgupta and L. F. Niño (2008).
Immunological Computation: Theory and Applications.
Auerbach.

[10] B. Doerr, T. Jansen, C. Witt, and C. Zarges (2013).
A method to derive fixed budget results from expected optimisation times.
In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2013)*. To appear.

[11] M. Elberfeld and J. Textor (2011).
Negative selection algorithms on strings with efficient training and linear-time classification.
Theoretical Computer Science 412:534–542.

[12] S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri (1994).
Self-nonself discrimination in a computer.
In *Proceedings of the 1992 IEEE Symposium on Security and Privacy*, pp. 202–212.

[13] J. Greensmith (2007).
The Dendritic Cell Algorithm.
PhD Thesis, University of Nottingham.

[14] F. Gu (2011).
Theoretical and Empirical Extensions of the Dendritic Cell Algorithm.
PhD Thesis, University of Nottingham.

[15] F. Gu, J. Greensmith, and U. Aickeln (2013).
Theoretical formulation and analysis of the deterministic dendritic cell algorithm.
Biosystems 111(2):127–135.

References III

[16] J. Hilder, N. Owens, P. Hickey, S. Cairns, D. Kilgour, J. Timmis, and A. Tyrrel (2011).
Parameter optimisation of the receptor density algorithm.
In *Proceeding of the 10th International Conference on Artificial Immune Systems (ICARIS 2011)*, pp. 226–239.

[17] C. Horoba, T. Jansen, and C. Zarges (2009).
Maximal age in randomized search heuristics with aging.
In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2009)*, p. 803–810.

[18] T. Jansen, P. S. Oliveto, and C. Zarges (2011).
On the Analysis of the Immune-Inspired B-Cell Algorithm for the Vertex Cover Problem.
In *Proceedings of the 10th International Conference on Artificial Immune Systems (ICARIS 2011)*. LNCS 6825, pp. 117-131.

[19] T. Jansen and C. Zarges (2010):
Aging beyond restarts.
In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2010)*, pp. 705–712.

[20] T. Jansen and C. Zarges (2011).
Analyzing different variants of immune inspired somatic contiguous hypermutations.
Theoretical Computer Science 412(6):517-533.

[21] T. Jansen and C. Zarges (2011).
Variation in artificial immune systems: Hypermutations with mutation potential.
In *Proceedings of the 10th International Conference on Artificial Immune Systems (ICARIS 2011)*. LNCS 6825, pp. 132–145.

[22] T. Jansen and C. Zarges (2011).
On benefits and drawbacks of aging strategies for randomized search heuristics.
Theoretical Computer Science 412(6):543–559.

References IV

[23] T. Jansen and C. Zarges (2011).
On the role of age diversity for effective aging operators.
Evolutionary Intelligence 4(2):99–125.

[24] T. Jansen and C. Zarges (2011).
Analysis of evolutionary algorithms: From computational complexity analysis to algorithm engineering.
In *11th ACM SIGEVO Workshop on Foundations of Genetic Algorithms (FOGA 2011)*, p. 1–14.

[25] T. Jansen and C. Zarges (2012).
Fixed budget computations: A different perspective on run time analysis.
In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2012)*, pp. 1325–1332.

[26] T. Jansen and C. Zarges (2012).
Computing longest common subsequences with the B-Cell Algorithm.
In *Proceedings of the 11th International Conference on Artificial Immune Systems (ICARIS 2012)*. LNCS 7597, pp. 111–124.

[27] N. Jerne (1974).
Towards a network theory of the immune system.
Annals of Immunology, 125C(1–2):373–389.

[28] J. Kelsey and J. Timmis (2003).
Immune inspired somatic contiguous hypermutation for function optimisation.
In *Proceedings of the 5th Annual Conference on Genetic and Evolutionary Computation (GECCO 2003)*, LNCS 2723, pages 207–218. Springer.

[29] P. Matzinger (1994).
Tolerance, danger, and the extended family.
Annual Reviews of Immunology, 12:991–1045.

[30] K. Murphy (2012).
Janeway's Immunobiology.
Garland Science, 8th edition.

References V

[31] F. Neumann and C. Witt (2010).
Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity.
Springer.

[32] N. D. L. Owens (2010).
From Biology To Algorithms.
PhD Thesis, University of York.

[33] N. D. L. Owens, A. Greensted, J. Timmis, and A. Tyrrell (2013).
The receptor density algorithm.
Theoretical Computer Science 481:51–73.

[34] T. Stibor, K. M. Bayarou, and C. Eckert (2004).
An investigation of r-chunk detector generation on higher alphabets.
In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, pp. 299–307.

[35] J. Timmis (2010)
Lecture Course on Collaborative Bio-Inspired Algorithms
Available online: `http://www.artificial-immune-systems.org/teaching.shtml`

[36] C. Zarges (2008).
Rigorous runtime analysis of inversely fitness proportional mutation rates.
In *Parallel Problem Solving from Nature (PPSN X)*. LNCS 5199, pp. 112–122.

[37] C. Zarges (2009).
On the utility of the population size for inversely fitness proportional mutation rates.
In *Proceedings of the 10th ACM SIGEVO Workshop on Foundations of Genetic Algorithms (FOGA 2009)*, pp. 39–46.

[38] C. Zarges (2011).
Theoretical Foundations of Artificial Immune Systems.
PhD Thesis, TU Dortmund.