Representations for Evolutionary Algorithms

Franz Rothlauf
University of Mainz
D-55099 Mainz/Germany
rothlauf@uni-mainz.de

GECCO'08, July 12-16, 2008, Atlanta, Georgia, USA.
ACM 978-1-60558-131-6/08/07.
slides can be downloaded at

http://wi.bwl.uni-mainz.de/Bilder_allgemein/rothlauf-tutorial.pdf

Scope of the Tutorial

- Illustrate the influence of representations on the performance of EAs.
- Illustrate the relationship between problem difficulty and used representation/operator.
- Review design guidelines for high-quality representations.
- Focus on some properties of representations
  - Locality of representations
  - Redundant representations and neutral search spaces
  - Synonymous and non-synonymous redundancy
  - (Exponentially scaled alleles)

Structure of the Tutorial

- A Short Introduction to Representations
  - Defining Representations
  - Representations, Operators, and Metrics
  - Direct and Indirect Representations
- Design Guidelines for Representations
- Properties of Representations
  - High-Locality Representations
  - Redundant Representations and Neutral Networks
  - Domino Convergence and Genetic Drift

Defining Representations

- A representation assigns genotypes to corresponding phenotypes.
- Every search and optimization algorithms needs a representation.
- The representation allows to represent a solution to a specific problem.
- Different representations can be used for the same problem.
- Performance of search algorithm depends on properties of the used representation and how suitable is the representation in the context of the used genetic operators.

Defining Representations (2)

- There are many different representations.
- Standard representations are binary, real-valued vectors, messy encodings, tree structures,...
- ... and we assume that everybody has some experience at least with some of them.

Defining Representations (3)

An optimization problem $f(\boldsymbol{x})$ can be separated into a genotype-phenotype mapping f_g and a phenotype-fitness mapping f_p:

$$f_g(\boldsymbol{x}_g) : \Phi_g \rightarrow \Phi_p,$$
$$f_p(\boldsymbol{x}_p) : \Phi_p \rightarrow \mathbb{R},$$

where $f = f_p \circ f_g = f_p(f_g(\boldsymbol{x}_g))$.

A change of f_g also changes the properties of f.

The genetic operators mutation and crossover are applied to $\boldsymbol{x}_g$, whereas the selection process is based on the fitness of $\boldsymbol{x}_p$.

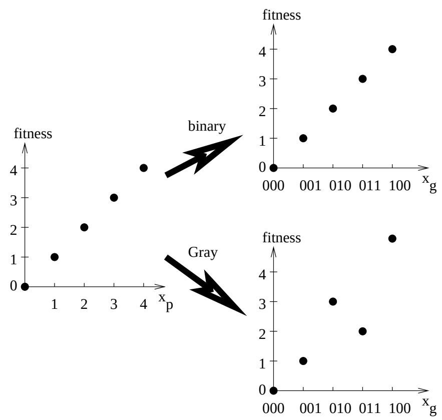
$f_p(\boldsymbol{x}_p)$ determines the difficulty and complexity of a problem.

$f_g(\boldsymbol{x}_g)$ is the used representation.

There are $||\Phi_g||!$ different representations.

Defining Representations (4)

- Representations change the character and difficulty of optimization problems.
- For example $f_p = x_p$, where $x_p \in \mathbb{N}$.
- Different problem depending on the used representations (Gray versus binary).



Defining Representations (5)

- Phenotypic problem easy to solve for hill-climber.
- When using bit-flipping GA the Gray-encoded problem is easier to solve than the binary-encoded problem.
- Gray encoding induces less local optima when used on problems of practical relevance (compare Free Lunch theorem (Whitley, 2000)).
- Resulting problem difficulty depends on used search method. If other search methods (e.g. other operators) are used, then problem difficulty is different (compare (Reeves, 2000)).

Representations, Operators, Metrics

Representation, metric defined on Φ_g and Φ_p, and genetic operators depend on each other and are closely related.

- A representation is just a mapping from Φ_g to Φ_p. It assigns any possible $x_g \in \Phi_g$ to an $x_p \in \Phi_p$.
- In both search spaces, Φ_g and Φ_p, a metric is or has to be defined. The metric determines the distances between the individuals and is the basis for measuring similarities between individuals. In general, the metric used for Φ_p is defined by the considered problem. The metric used for Φ_g is determined by the used search operators.
- Genotypic operators like mutation and crossover are defined based on the used metric.

Representations, Operators, Metrics (2)

Mutation:

The application of mutation to an individual results in a new individual with similar properties. There is a small distance betwen offspring and parent.

Crossover:

Crossover combines the properties of two or more parents in an offspring. The distance between offspring and parent should be smaller than the distance between both parents

(basic idea of "geometric crossover" from (Moraglio and Poli, 2004); compare also (Surry and Radcliffe, 1996a), (Liepins and Vose, 1990), or (Rothlauf, 2002))

Representations, Operators, Metrics (3)

Results:

- Metric on Φ_g and used operators depend on each other. The one determines the other.
- Representations transform the metric on Φ_g to the (problem-dependent) metric on Φ_p. (Compare locality, causality, and distance distortion)

Direct Representations

If the genetic operators are applied directly to the phenotypes it is not necessary to specify a representation and the phenotypes are identical with the genotypes:

$$f_g(\boldsymbol{x}_g) : \Phi_g \rightarrow \Phi_g,$$
$$f_p(\boldsymbol{x}_p) : \Phi_g \rightarrow \mathbb{R}.$$

This means, f_g is the identity function $f_g(\boldsymbol{x}_g) = \boldsymbol{x}_g$.

Using direct representations do not neccessarily make life easier:

- Design of proper operators is difficult
- How can we apply specific types or EAs (like EDAs)?
- Representation issues are not important any more ($\Phi_g = \Phi_p$ and $f_g(x_g) = x_g$).

Direct Representations - Genetic Programming

Representation issues are also relevant to Genetic Programming.

Phenotypes: Programs, logical expressions.
Genotypes: Parse trees, bitstrings, linear structures, ...

Neglecting proper genotype-phenotype mappings can result in low performance of GP approaches.

E.g.: Standard GP (expression tree representation and sub-tree swapping crossover) cannot solve problems where optimal solutions require very full or very narrow trees (Daida et al., 2001). This is due to problems of the representation (interplay between genotypes and used search operators) (Hoai et al., 2006).

Indirect Representations

The use of an explicite genotype-phenotype mapping has some benefits:

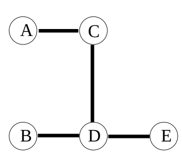
- Specific constraints can be considered.
- Standardized genetic operators with known behavior and properties can be used.
- An indirect representation is necessary if problem-specific operators are either not available or difficult to design.
- Representation can make problem easier by incorporating problem-specific knowledge.

Indirect Representations - Specific Constraints

Example: Tree optimization problems

A tree is a fully connected graph with exactly $n-1$ links (for an n node network). There are no circles in a tree.

A graph can be represented by its characteristic vector.

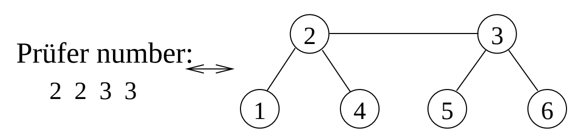


$\Updownarrow$

0	1	0	0	0	1	0	1	0	1
A-B	A-C	A-D	A-E	B-C	B-D	B-E	C-D	C-E	D-E

Indirect Representations - Specific Constraints (2)

Prüfer numbers are a one-to-one mapping between trees and a sequence of integers (like other Cayley codes). A tree with n nodes is represented by a string of length $n-2$ over an alphabet of n symbols.



Therefore, using Prüfer numbers allows to consider the constraint that the graph is a tree (For other representations repair operators are necessary).

Indirect Representations - Standardized Operators

- When mapping many different types of phenotypes on only a few types of different genotypes (binary, integer, or continuous representations), it is possible to use standardized operators.
- Behavior of EAs for standard representations like binary (simple GAs) or continuous (evolution strategies) representations well understood.
- Mapping phenotypes on binary genotypes allows the use of schemata and effective linkage learning GAs (under the assumption that the problem still remains decomposable and that binary encodings allow a natural encoding of the problem).

Indirect Representations - Problem-specific Operators

- Developing of problem-specific operators is difficult and often additional repair mechanisms must be used to ensure a valid solution.
- For some real-world problems there are no problem-specific operators available.

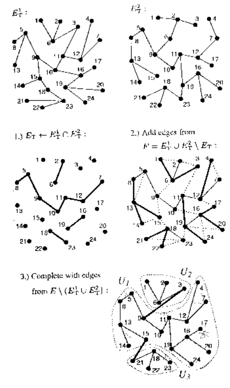
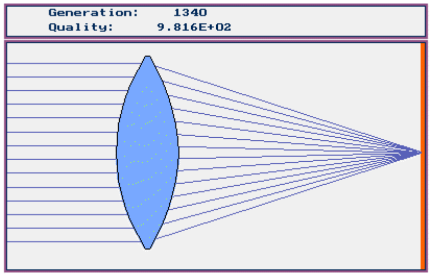


Figure 2: An example for edge crossover ($d = 3$).

(from (Raidl, 2000))

Indirect Representations - Problem-specific Operators (2)

For some types of problems no problem-specific operators exist that can be applied to direct representations.



Indirect Representations - Problem-specific Knowledge

Incorporating problem-specific knowledge in the representations to increase GA performance:

- Increase the initial supply of solutions that are similar to the optimal solution.
- Use high-locality representations for easy problems.
- Consider specific properties of the optimal solution (e.g. stars and trees).
- Use representations that make a problem easier for a specific optimization method.

- A Short Introduction to Representations
- **Design Guidelines for Representations**
- Properties of Representations
  - High-Locality Representations
  - Redundant Representations and Neutral Networks
  - Domino Convergence and Genetic Drift

Goldberg's Recommendations

- Principle of meaningful building blocks: The schemata should be short, of low order, and relatively unrelated to schemata over other fixed positions.
- Principle of minimal alphabets: The alphabet of the encoding should be as small as possible while still allowing a natural representation of solutions (qualified by (Goldberg, 1991))

from (Goldberg, 1989))

Goldberg's Recommendations (2)

- The recommendations caused a lot of critics (Radcliffe, 1997; Fogel and Stayton, 1994).
- What is a natural representation of a problem? (For example, is using binary representations for encoding real-valued phenotypes a natural representation?)
- Principles mainly aimed at binary representations and crossover-based GAs that process schemata. Not big help for other search methods like evolution strategies or evolutionary programming as these search methods do not process schema.

Radcliffe's Recommendations

Representation and operators belong together and can not be separated from each other (Radcliffe, 1992).

Design of representation-independent evolutionary algorithms is possible if the following properties are considered (Surry and Radcliffe, 1996b):

- **Respect**: Offspring produced by recombination are members of all formae to which both their parents belong.
- **Transmission**: Every gene is set to an allele which is taken from one of the parents.
- **Assortment**: Offspring can be formed with any compatible characteristics taken from the parents.
- **Ergodicity**: Iterative use of operators allow to reach any point in the search space.

Palmer's Recommendations

- An encoding should be able to represent all possible phenotypes.
- An encoding should be unbiased in the sense that all possible individuals are equally represented in the set of all possible genotypic individuals.
- An encoding should encode no infeasible solutions.
- The decoding of the phenotype from the genotype should be easy.
- An encoding should possess locality. Small changes in the genotype should result in small changes in the phenotype (compare statements about metric).

from (Palmer, 1994))

Ronald's Recommendations

- Encodings should be adjusted to a set of genetic operators in a way that the building blocks are preserved from the parents to the offspring (Fox and McMahon, 1991).
- Encodings should minimize nonlinearities in fitness functions (Beasley et al., 1993). This means, representations should make the problem easier (for local search methods!).
- Feasible solutions should be preferred.

Ronald's Recommendations (2)

- The problem should be represented at the correct level of abstraction.
- Encodings should exploit an appropriate genotype-phenotype mapping process if a simple mapping to the phenotype is not possible.
- Isomorphic forms, where the phenotype of an individual is encoded with more than one genotype, should not be used.

from (Ronald, 1997))

Design Guidelines - Summary

- Based on observations for specific test problems there are some common, fuzzy ideas about what is a good representation.
- Recommendations too general to be helpful for designing or evaluating representations.
- There is a lack of analytical models describing the influence of representations on EAs.
- To verify (or reject) observations analytical models are necessary.

- A Short Introduction to Representations
- Design Guidelines for Representations
- Properties of Representations
  - **High-Locality Representations**
  - Redundant Representations and Neutral Networks
  - Domino Convergence and Genetic Drift

Locality

- Representations (genotype-phenotype mappings) can change the structure of the neighborhood and the structure of the fitness landscapes.
- Each neighbor can be reached directly by a move (mutation, crossover, etc.). Therefore, the neighborhood structure depends on the used operator and the used metric.
- The set of neighbors can be different for the genotypes and phenotypes.
- The distance between two individuals is determined by the number of moves between both individuals.

Locality (2)

The locality of a representation describes how well neighboring genotypes correspond to neighboring phenotypes. Therefore, the locality of a representation is high, if neighboring genotypes correspond to neighboring phenotypes.

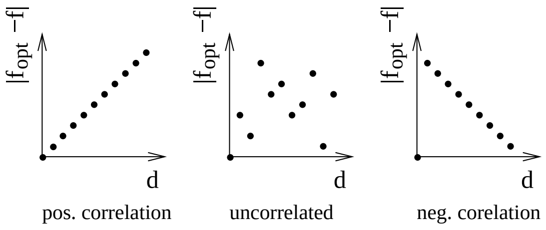
Locality, causality, and distance distortion describe how well the metric on Φ_p fits to the metric on Φ_g. If they fit well the locality is high.

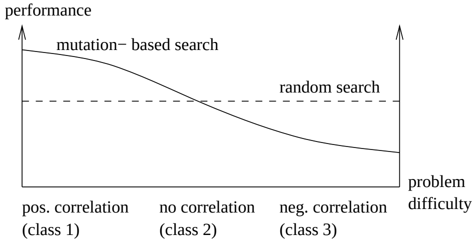
Representations f_g that change the distances between corresponding genotypes and phenotypes modify the difficulty of the problem (difficulty$(f) \neq$ difficulty(f_p)).

Locality - Different Types of Phenotype-Fitness Mappings (Jones and Forrest, 1995)

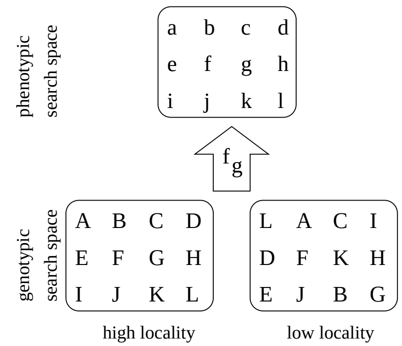
1. **Class1:** Fitness difference to optimal solution is positively correlated with the distance to optimal solution. Structure of the search space guides local search methods to the optimal solution → easy for mutation-based search.
2. **Class 2:** No correlation between fitness difference and distance to optimal solution. Structure of the search space provides no information for guided search methods → difficult for guided search methods.
3. **Class 3:** Fitness difference is negatively correlated to distance to optimal solution. Structure of search space misleads local search methods to sub-optimal solutions → deceptive problems.

Locality - Different Types of Phenotype-Fitness Mappings(2)

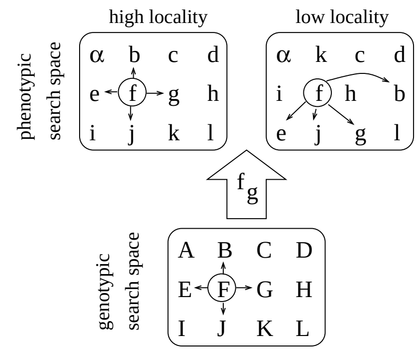




Locality - Low versus High-Locality Representations



Influence of high versus low-locality representations on genotype-phenotype mappings



Effect of mutation for high versus low-locality representations

Locality - Low versus High-Locality Representations(2)

Class 1:
High-locality representations preserve difficulty of problem. Easy problems remain easy for guided search.

Low-locality representations make easy problems more difficult. Resulting problem becomes of class 2.

Class 2:
High-locality representations preserve difficulty of problem. Problems remain difficult for guided search.

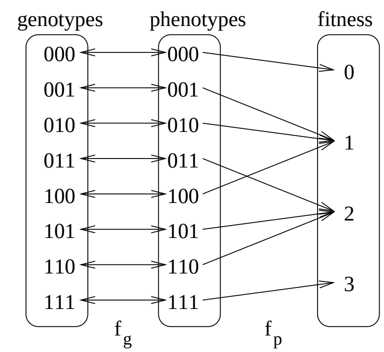
Low-locality representations on average do not change class of problem. Problems remain difficult.

Class 3:
High-locality representations preserve deceptiveness of problem. Traps remain traps.

Low-locality representations transform problem to class 2 problem. Deceptive problems become more easy to solve for guided search.

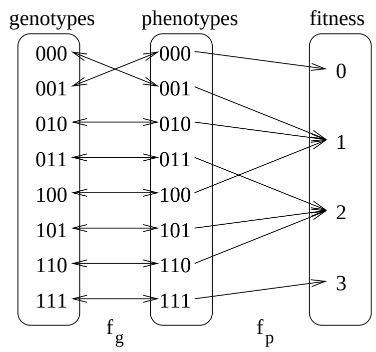
Locality - An Example

- Both, genotypes and phenotypes are binary.
- We use the bit-flipping operator as a move (Hamming distance).
- One-max problem (class 1).
- All building blocks (regarding genotypes and phenotypes) are of size $k = 1$. Therefore, problem is easy for selectorecombinative GAs.

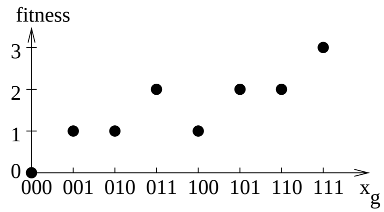


Locality - An Example (2)

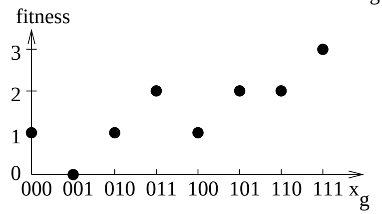
- A representation with lower locality.
- The neighborhood structure changes.
- Not all genotypic building blocks are of size 1. Although, f_p remains unchanged, f becomes more difficult.



Locality - An Example (3)



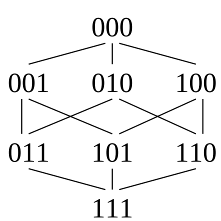
- High-locality representation.
- Problem easy for selectorecombinative GAs.



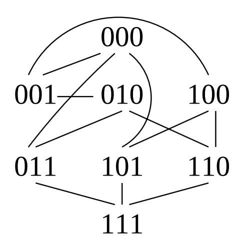
- Different fitness for genotypes 000 and 001.
- Problem more difficult for selectorecombinative GAs.
- Neighborhood not preserved by representation.

Locality - An Example (4)

Neighborhood structure of the genotypes:



Resulting neighborhood structure of the phenotypes:



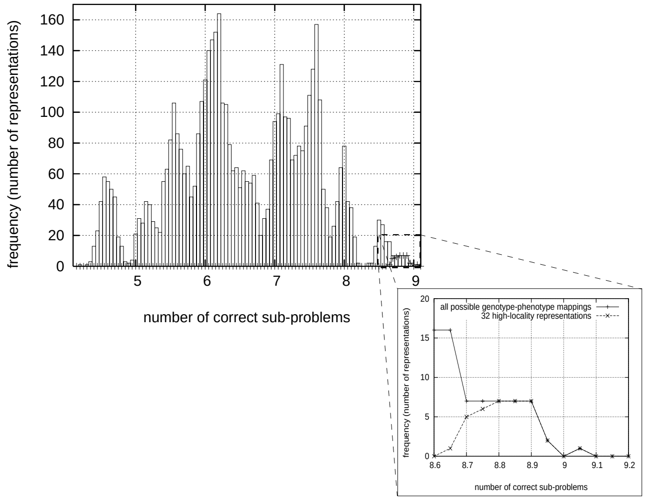
Comparing Representations

- We compare the performance of selectorecombinative GAs over all different representations for the one-max problem.
- When focusing on binary bitstrings and assigning l-bit genotypes to l-bit phenotypes, there are $2^l!$ different representations.
- For $l = 3$ there are 8 different genotypes, resp. phenotypes, and $8! = 40,320$ different representations.
- 36 different representations result in the same overall problem f (for the one-max problem).

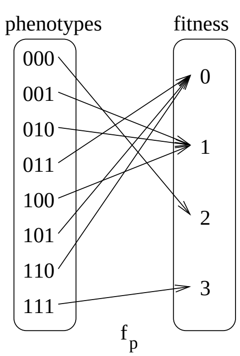
Comparing Representations (2)

- To reduce problem complexity, $x_g = 111$ is always assigned to $x_p = 111$. Therefore, there are $7! = 5040$ different representations.
- We concatenate ten 3-bit problems and use a GA with tournament selection of size 2, uniform crossover, and $N = 16$.

Comparing Representations (3)

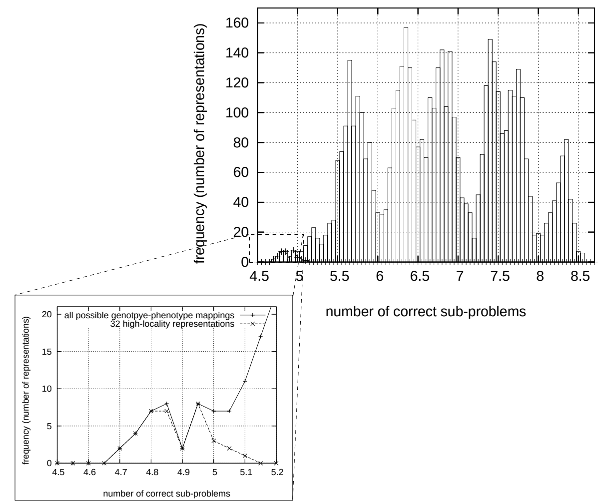


Comparing Representations (4)



- We compare the performance of selectorecombinative GAs over all different representations for the deceptive trap problem.
- To reduce problem complexity, $x_g = 111$ is always assigned to $x_p = 111$. Therefore, there are $7! = 5040$ different representations.
- We concatenate ten 3-bit problems and use a GA with tournament selection of size, uniform crossover, and $N = 16$.

Comparing Representations (5)



High-Locality Representations - Summary

- When using high locality representations, genotypic neighbors correspond to phenotypic neighbors.
- High locality representations do not change the structure and difficulty of the problem.
  - Easy problems remain easy.
  - Difficult problems remain difficult.
- Locality depends on the used distance metrics which depend on the used operators.

- A Short Introduction to Representations
- Design Guidelines for Representations
- Properties of Representations
  - High-Locality Representations
  - **Redundant Representations and Neutral Networks**
  - Domino Convergence and Genetic Drift

Redundant Representations

Representations are redundant if the number of genotypes is larger than the number of phenotypes.

- Using redundant representations f_g means changing $f = f_p(f_g)$. There are additional plateaus in the fitness landscape.
- Redundant representations are more "inefficient" encodings which use a higher number of alleles but do not increase the amount of encoded information.
- Redundant representations are not an invention of AI researchers but are commonly used in nature.

Redundant Representations (2)

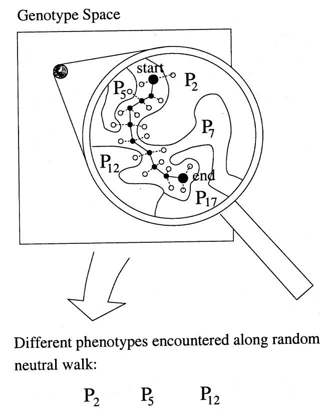
There are different opinions regarding the influence of redundant representation on the performance of EAs.

- Redundant representations reduce EA performance due to loss of diversity (Davis, 1989; Eshelman and Schaffer, 1991; Ronald et al., 1995)
- Redundant representations increase EA performance (Gerrits and Hogeweg, 1991; Cohoon et al., 1988; Julstrom, 1999)

Redundant Representations (3)

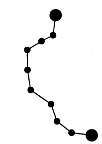
- Large amount of work considers the *neutral theory* (Kimura, 1983). This theory assumes that not natural selection fixing advantageous mutations but the random fixation of neutral mutations is the driving force of molecular evolution.
- Following these ideas redundant representations (neutral networks) are used in EAs with great enthusiasm.
- There is hope that increasing the evolvability of a system also increases the performance of the system (Barnett, 1997; Barnett, 1998; Shipman, 1999; Shipman et al., 2000b; Shackleton et al., 2000; Shipman et al., 2000a; Ebner et al., 2001; Smith et al., 2001c; Smith et al., 2001a; Smith et al., 2001b; Barnett, 2001; Yu and Miller, 2001; Yu and Miller, 2002; Toussaint and Igel, 2002).
- (Knowles and Watson, 2002) showed exemplarily that this is not true!

Redundant Representations (4)



from (Ebner et al., 2001)

Neutral Network: Set of genotypes connected by single-point mutations that map to the same phenotype.



Redundant Representations (5)

Benefits of Neutral Networks

- Population can drift along these neutral networks.
- Reducing the chance of being trapped in sub-optimal solutions.
- Population is quickly able to recover after a change has occurred.
- Evolvability of the system increases.

Problems

- Higher evolvability → Randomization of search
- Genetic drift?

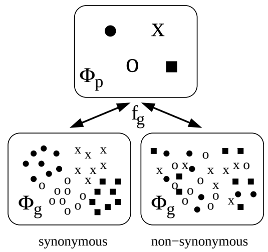
In the following slides we study

- how to distinguish between synonymously and non-synonymously redundant encodings (Rothlauf and Goldberg, 2003),
- how synonymous redundancy changes the performance of EAs (quantitative predictions) (Rothlauf and Goldberg, 2003), and
- the properties of non-synonymously redundant representations (Choi and Moon, 2003; Choi and Moon, 2007).

Synonymously versus Non-synonymously Redundant Representations

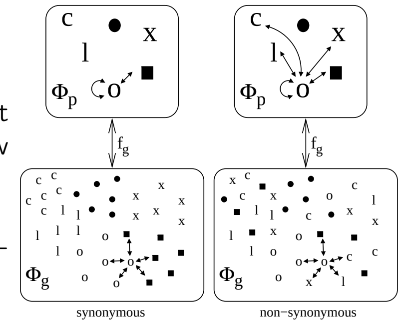
When using redundant representations it can be distinguished between:

- **Synonymously redundant representations**: All genotypes that encode the same phenotype are similar to each other.
- **Non-synonymously redundant representations**: Genotypes that encode the same phenotype are not similar to each other.



Synonymously versus Non-synonymously Redundant Rep. (2)

- Non-synonymously redundant representations do not allow guided search.
- EA search becomes random.
- Similar effect as low locality representations.



Effects of small mutation steps

Synonymously versus Non-synonymously Redundant Rep. (3)

- (Choi and Moon, 2003) defined uniformly redundant encodings that are *maximally non-synonymous* and proved that such encodings induce uncorrelated search spaces (fitness-distance correlation is equal to zero).

- For a maximally non-synonymous redundant encoding the expected distance between any two genotypes that correspond to the same phenotype is invariant and about equal to the problem size n.

- Normalization (transformation of one parent to be consistent with the other) can transform uncorrelated search spaces into correlated search spaces with higher locality (Choi and Moon, 2007).

Synonymously versus Non-synonymously Redundant Rep. (4)

Some selected examples for problems with maximally non-synonymous redundant encodings (Choi and Moon, 2007):

- **Partitioning problems in graphs**: k subsets are represented by integers from 0 to $k-1$ where nodes are contained in the same group if they are represented by the same number. Each phenotype is represented by $k!$ different genotypes.

- **HIFF problems** (Watson et al., 1998): binary encoding where each phenotype is represented by a pair of bitwise complementary genotypes.

- **TSP**: Order-based crossover, in which vertices are indexed from 1 to n and each tour is represented by a permutation of the vertex indices. Each phenotype is represented by $2n$ genotypes.

Modeling Redundant Representations

Synonymously redundant representations can be described using

- order of redundancy $k_r = \frac{\log(|\Phi_p|)}{\log(|\Phi_g|)}$ and
- over-, resp. underrepresentation r of the optimal solution due to the problem representation f_g.

When using the notion of BBs and binary representations:

- $k_r = \frac{k_g}{k_p}$
- r: Number of genotypic BBs of order k_g that represent the optimal phenotypic BB of order k_p.

Modeling Redundant Representations (2)

Example 1:

genotypes x_g	x_p
00 00, 00 01, 01 00, 01 01	0 0
10 00, 10 01, 11 00, 11 01	1 0
00 10, 01 11, 00 11, 01 11	0 1
10 10, 10 11, 11 10, 11 11	1 1

- $k = 2$ (order of phenotypic BBs)
- $k_r = 2$ (One allele of a phenotype is represented using two alleles of a genotype)
- Uniform redundancy: $r = 4$ (the best BB (e.g.. $x_p = 11$) is represented by four genotypic BBs)

Modeling Redundant Representations (3)

Example 2:

genotypes x_g	x_p
000, 001, 010, 100, 101, 110, 011	0
111	1

- $k = 1$ (order of phenotypic BBs)
- $k_r = 3$ (One phenotypic allele is represented using three genotypic alleles)
- Non-uniform redundancy: $r = 1$ (best BB ($x_p = 1$) is represented by one genotypic BB ($x_g = 111$))

Population Sizing for GAs

The Gambler's ruin model (Feller, 1957) can be used for modeling the iterated decision making in GAs.

A gambler with initial stake x_0 wishes to increase his funds to a total of N units by making a sequence of bets against a gaming house. Each bet has fixed probability p of winning ($q = 1 - p$ of losing), and we wish to know the probability of succeeding (getting N units) or failing (losing all units).

Following (Harik et al., 1997) the probability that a GA with a population size N converges after t_{conv} generations to the correct solution is

$$P_n = \frac{1 - (q/p)^{x_0}}{1 - (q/p)^N}$$

Population Sizing for GAs (2)

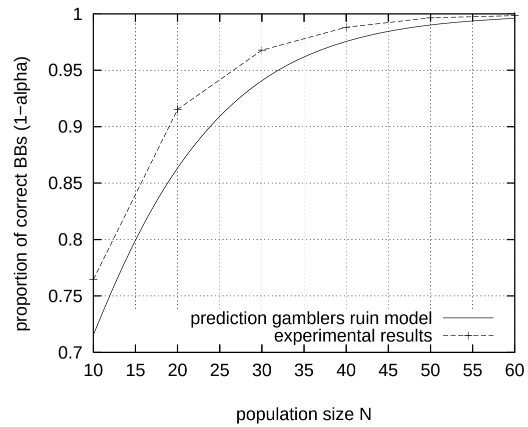
After some calculations we get:

$$N \approx -2^{k-1} \ln(\alpha) \frac{\sigma_{BB}\sqrt{\pi m'}}{d}$$

N is the necessary population size, $\alpha = 1 - P_n$ the probability P_n that the optimal BB cannot be found (probability of failure) and k is the order of the BBs.

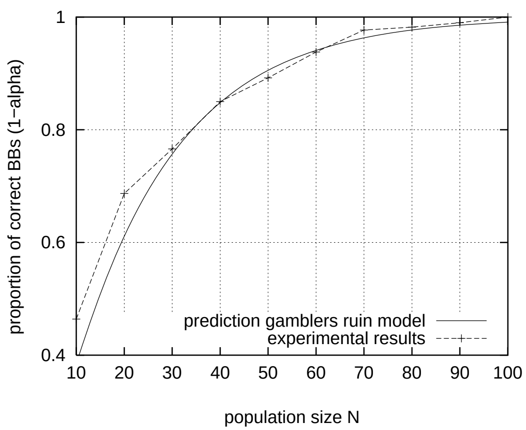
σ_{BB} (variance of BBs), d (fitness difference between best and second best BB), $m' = m - 1$ (number of BBs) and k are problem-dependent.

Population Sizing for GAs (3)



150-bit one-max problem ($k = 1$, $\sigma_{BB} = 0.25$, $d = 1$ and $m = 150$)

Population Sizing for GAs (4)



Ten concatenated 3-bit deceptive traps ($k = 3$, $\sigma_{BB} = 1$, $d = 1$ and $m = 10$)

Population Sizing for GAs (5)

Now we have to ask how the redundancy of a representation influences GA performance?

Observation: Redundant representation change the initial supply x_0 of BBs.

For binary problem representation:

$$x_0 = N \frac{r}{2^{kk_r}},$$

where N is the population size.

Population Sizing for GAs (6)

When using synonymously redundant representations the existing model can be extended:

$$N \approx -\frac{2^{k_r k-1}}{r}\ln(\alpha)\frac{\sigma_{BB}\sqrt{\pi m'}}{d}$$

The population size N that is necessary to find the optimal solution with probability $P_n = 1-\alpha$ goes with $O\left(\frac{2^{k_r}}{r}\right)$.

Population Sizing for GAs (7)

Conclusions from this model:

- Redundant representations can change the performance of EAs.
- If representations are synonymously redundant:
  - Uniformly redundant representations do not change the performance of EAs!
  - If the optimal BB is overrepresented GA performance increases.
  - If the optimal BB is underrepresented GA performance decreases.
- Redundant representations can not be used systematically if there is no problem-specific knowledge!

Population Sizing for GAs (8)

What must be considered when using redundant representations?

1. How does the used representation change the size of the search space?
2. Is the representation synonymously redundant?
3. Are some solutions overrepresented?

Examining these properties allows the user to increase the performance of EAs!

In the following slides we show how this theory can be used for predicting EA performance when using the trivial voting mapping for binary problems.

Trivial Voting Mapping

- The trivial voting mapping (TVM) assigns binary phenotypes to binary genotypes.
- One bit of the phenotype is represented by k_r genotypic bits.
- In general, a phenotypic bit is 0 if less than u genotypic bits are zero. If more than u genotypic bits are 1 then the phenotypic bit is 1.
- For $u = k_r/2$ the value of the phenotypic bit is determined by the majority of the genotypic bits (majority vote)

In general:

$$x_i^p = \begin{cases} 0 \text{ if } & \sum_{j=0}^{k_r-1} x^g_{k_r i+j} < u \\ 1 \text{ if } & \sum_{j=0}^{k_r-1} x^g_{k_r i+j} \geq u, \end{cases}$$

where $u \in \{1, \dots, k_r\}$.

Trivial Voting Mapping (2)

Examples:

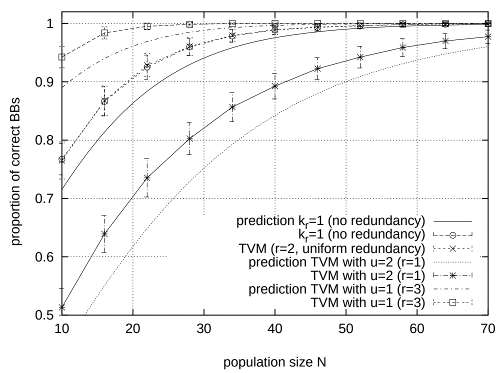
genotypes x_g	x_p
000, 001, 010, 100	0
110, 101, 011, 111	1

- $k = 1$
- $k_r = 3$
- $u = 2$

genotypes x_g	x_p
000	0
001, 010, 100,110, 101, 011, 111	1

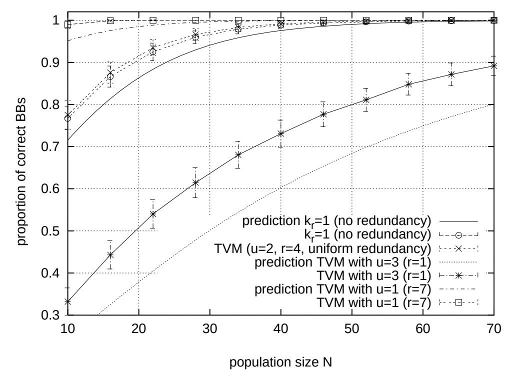
- $k = 1$
- $k_r = 3$
- $u = 1$

Trivial Voting Mapping (3)



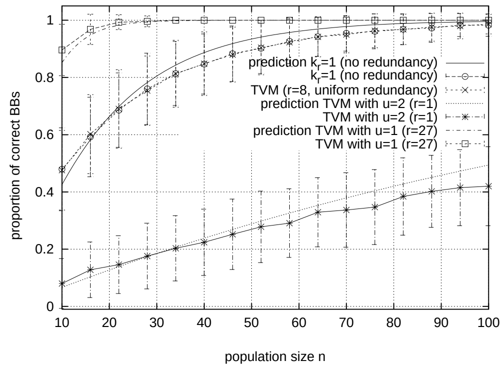
Experimental and theoretical results of the proportion of correct BBs on a 150-bit one-max problem using the trivial voting mapping for $k_r = 2$.

Trivial Voting Mapping (4)



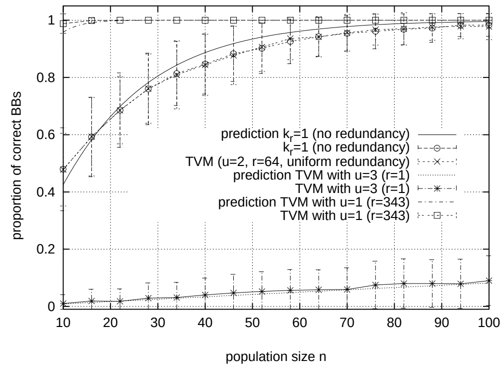
Experimental and theoretical results of the proportion of correct BBs on a 150-bit one-max problem using the trivial voting mapping for $k_r = 3$.

Trivial Voting Mapping (5)



Experimental and theoretical results of the proportion of correct BBs for ten concatenated 3-bit deceptive traps and $k_r = 2$.

Trivial Voting Mapping (6)



Experimental and theoretical results of the proportion of correct BBs for ten concatenated 3-bit deceptive traps and $k_r = 3$.

Redundant Representations - Summary

- There are theoretical models that allow us to predict the expected GA performance when using redundant representations ($N = O(2^{k_r}/r)$).
- There are guidelines for the design of redundant representations:
  - Do not use non-synonymously redundant representations!
  - If you use redundant representations you have to investigate:
    - How does the representation change the size of the search space?
    - Are solutions similar to the optimal solution overrepresented?
  - If there is no knowledge about the optimal solution use a uniformly redundant representation.

- A Short Introduction to Representations
- Design Guidelines for Representations
- Properties of Representations
  - High-Locality Representations
  - Redundant Representations and Neutral Networks
  - **Domino Convergence and Genetic Drift**

Exponentially Scaled Alleles

The alleles of a genotype can be of different importance for the construction of the phenotype.

In many real-world problems it is unclear if the genotypic alleles are uniformly or non-uniformly scaled.

A GA solves the most important alleles first and continues with lower salient alleles (domino convergence)

Genotypic alleles that have little influence on the phenotype are randomly fixed due to genetic drift.

Domino Convergence

The contribution of the genotypic alleles to the construction of the phenotype can be either uniformly or non-uniformly scaled.

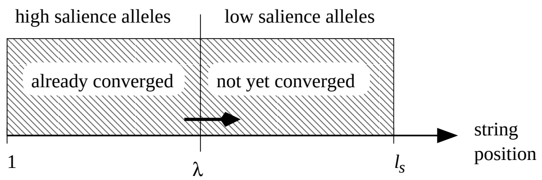
- Uniformly scaled representations:
  - Unary encoding, Gray encoding
  - All alleles are solved implicitly in parallel.
- Exponentially scaled representations:
  - Binary encoding
  - The alleles are solved step by step and domino convergence occurs.

Domino Convergence (2)

The BinInt problem: $f(x) = \sum_{i=0}^{l-1} x_i 2^{l-i-1}$ can be decomposed in

- the exponentially scaled representation $f_g(\boldsymbol{x}_g) = \sum_{i=0}^{l-1} 2^i x_{g,i}$,
- and the problem $f_p(\boldsymbol{x}_p) = x_p$.

When using GAs and non-uniformly scaled representations domino convergence occurs.



Domino Convergence (3)

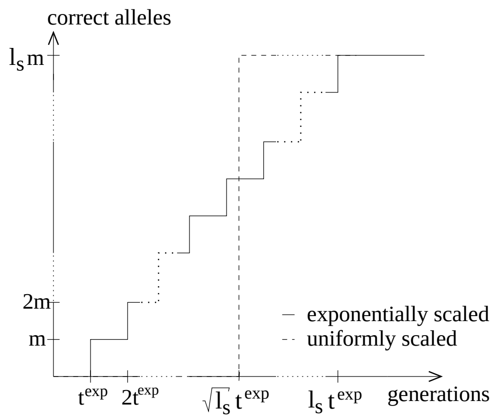
Domino convergence changes the dynamics of selectorecombinative GAs.

Time complexity (neglecting genetic drift):

Uniformly scaled alleles		Exponentially scaled alleles	
const sel. int.	prop. sel.	const sel. int.	prop. sel.
$O(\sqrt{l})$	$O(l\ln(l))$	$O(l)$	$O(2^l)$

Exponentially scaled representations result in longer GA runs!

Domino Convergence (4)



Comparison of time complexity using constant selection intensity:

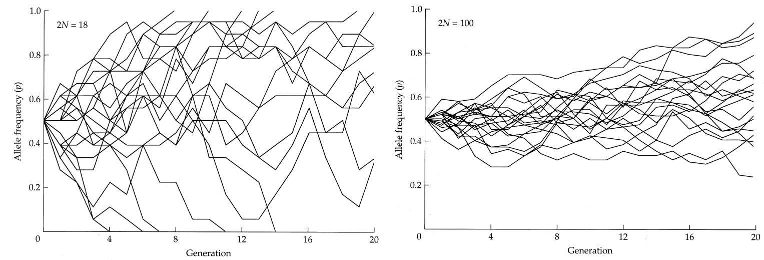
t^{exp}: time for solving one exponentially scaled allele

m: number of exponentially scaled building blocks

l_s: length of one exponentially scaled building block

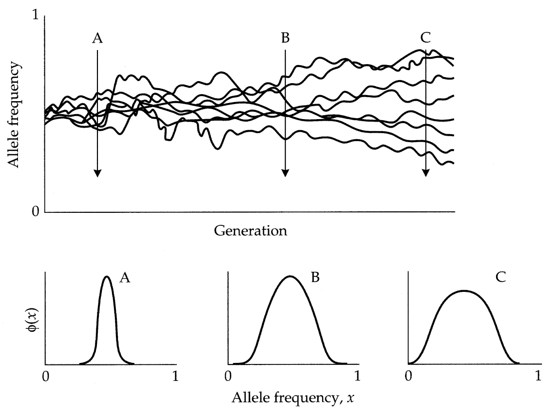
Genetic Drift

If there is no selection pressure, genetic drift occurs. The random process of sampling individuals can result in in a population with only one type of allele.



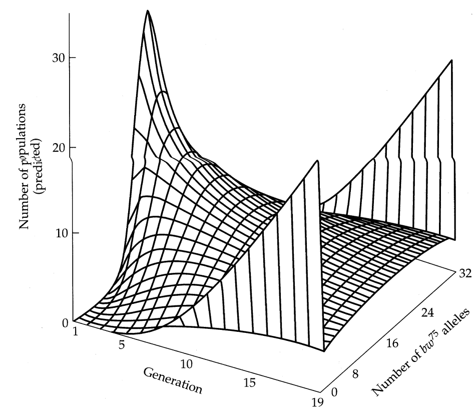
from: (Hartl and Clark, 1997, p. 271)

Genetic Drift (2)



from: (Hartl and Clark, 1997, p. 274)

Genetic Drift (3)



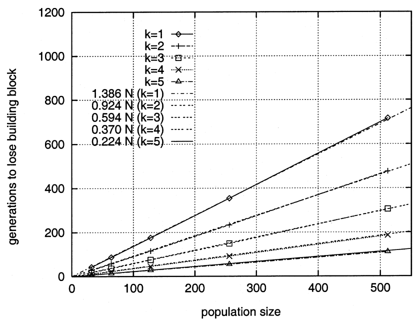
from: (Hartl and Clark, 1997, p. 281)

Genetic Drift (4)

The drift time – using random sampling with replacement – in GAs is proportional to the population size N:

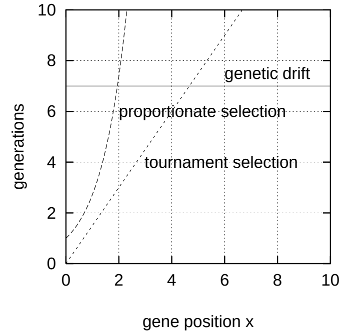
$t_{drift} = cN$,

where c depends on the initial proportion.



from: (Lobo et al., 2000)

Genetic Drift and Domino Convergence

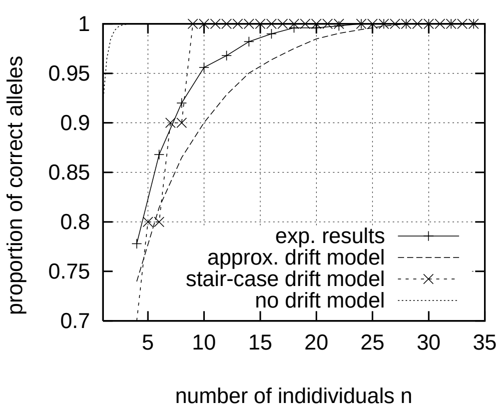


from: (Thierens et al., 1998)

Combining domino convergence and drift models:

- Drift models predict a constant generations upper boundary.
- Lower salient alleles are fixed randomly due to genetic drift.

Genetic Drift and Domino Convergence - Empirical Results



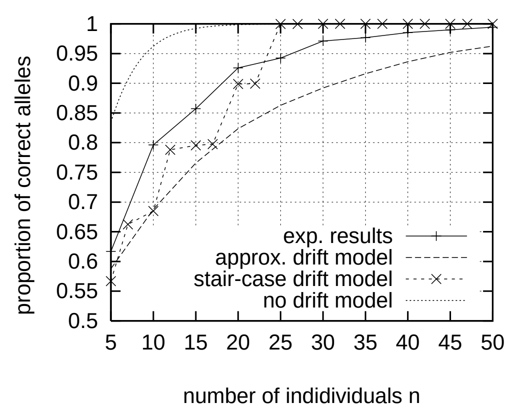
$m = 1,\ l_s = 5$

Simple GA with uniform crossover, no mutation, and tournament selection of size 2 without replacement.

Approx. drift model: Population sizing model based on the drift model of (Kimura, 1964).

Stair-case drift model: Solving time for one exp. scaled allele t^{exp} stays constant. Unsolved alleles remain in their initial state for $t < t_{drift}$.

Genetic Drift and Domino Convergence - Empirical Results



$m = 10,\ l_s = 5$

Simple GA with uniform crossover, no mutation, and tournament selection of size 2 without replacement.

Approx. drift model: Population sizing model based on the drift model of (Kimura, 1964).

Stair-case drift model: Solving time for one exp. scaled allele t^{exp} stays constant. Unsolved alleles remain in their initial state for $t < t_{drift}$.

Exponentially scaled Representations - Summary

- Representations using exponentially scaled alleles change the dynamics of selectorecombinative GAs.
  - Exponentially scaled representations allow to find rough approximations after short time.
  - Uniformly scaled representations allow to find the best solution in shorter overall time.
- Due to genetic drift GAs using exponentially scaled representations need a larger population size.

Take home message

- **Representations are important!**
- Representations change the difficulty of a problem!
- Distinguish carefully between genotypes and phenotypes!
- Representations that are non-synonymously redundant are no good idea.
- Synonymously redundant representations can help you if you have problem-specific knowledge!
- Representations should have high locality if you want to solve **easy** problems.
- (Scaling of alleles changes dynamics of search. Non-uniformly scaled alleles are fast, but inaccurate.)

Last remark

Thanks for your attention and patience!

Further reading:

Rothlauf, Franz (2006). *Representations for Genetic and Evolutionary Algorithms*. Springer, Berlin, 2nd edition.

References

Barnett, L. (1997). Tangled webs: Evolutionary dynamics on fitness landscapes with neutrality. Master's thesis, School of Cognitive Sciences, University of East Sussex, Brighton.

Barnett, L. (1998). Ruggedness and neutrality: The NKp family of fitness landscapes. In Adami, C., Belew, R. K., Kitano, H., and Taylor, C., editors, *Proceedings of the 6th International Conference on Artificial Life (ALIFE-98)*, pages 18–27, Cambridge, MA, USA. MIT Press.

Barnett, L. (2001). Netcrawling - optimal evolutionary search with neutral networks. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC01*, pages 30–37, Piscataway, NJ. IEEE Press.

Beasley, D., Bull, D. R., and Martin, R. R. (1993). Reducing epitasis in combinatorial problems by expansive coding. In Forrest, S., editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 400–407, San Mateo, CA. Morgan Kaufmann.

Choi, S.-S. and Moon, B.-R. (2003). Normalization in genetic algorithms. In et al., E. C.-P., editor, *Proceedings of the Genetic and Evolutionary Computation Conference 2003*, pages 862–873, Heidelberg. Springer.

Choi, S.-S. and Moon, B.-R. (2007). Normalization in genetic algorithms. forthcoming in IEEE Transactions on Evolutionary Computation.

Cohoon, J. P., Hegde, S. U., Martin, W. N., and Richards, D. (1988). Floorplan design using distributed genetic algorithms. In *IEEE International Conference on Computer Aided-Design*, pages 452–455. IEEE.

Daida, J. M., Bertram, R., Stanhope, S., Khoo, J., Chaudhary, S., Chaudhri, O., and Polito, J. (2001). What makes a problem GP-hard? analysis of a tunably difficult problem in genetic programming. *Genetic Programming and Evolvable Machines*, 2(2):165–191.

Davis, L. (1989). Adapting operator probabilities in genetic algorithms. In Schaffer, J. D., editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 61–69, San Mateo, CA. Morgan Kaufmann.

Ebner, M., Langguth, P., Albert, J., Shackleton, M., and Shipman, R. (2001). On neutral networks and evolvability. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 1–8, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea. IEEE Press.

Eshelman, L. J. and Schaffer, J. D. (1991). Preventing premature convergence in genetic algorithms by preventing incest. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 115–122, San Mateo, CA. Morgan Kaufmann.

Feller, W. (1957). *An Introduction to Probability Theory and its Applications*, volume 1. John Wiley & Sons, New York, 1st edition.

Fogel, D. B. and Stayton, L. C. (1994). On the effectiveness of crossover in simulated evolutionary optimization. *BioSystems*, 32:171–182.

Fox, B. R. and McMahon, M. B. (1991). Genetic operators for sequencing problems. In Rawlins, G. J. E., editor, *Foundations of Genetic Algorithms*, pages 284–300, San Mateo, CA. Morgan Kaufmann.

Gerrits, M. and Hogeweg, P. (1991). Redundant coding of an NP-complete problem allows effective Genetic Algorithm search. In Schwefel, H.-P. and Männer, R., editors, *Parallel Problem Solving from Nature*, pages 70–74, Berlin. Springer-Verlag.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, MA.

Goldberg, D. E. (1991). Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems*, 5(2):139–167. (Also IlliGAL Report 90001).

Harik, G. R., Cantú-Paz, E., Goldberg, D. E., and Miller, B. L. (1997). The gambler's ruin problem, genetic algorithms, and the sizing of populations. In Bäck, T., editor, *Proceedings of the Forth International Conference on Evolutionary Computation*, pages 7–12, New York. IEEE Press.

Hartl, D. L. and Clark, A. G. (1997). *Principles of population genetics*. Sinauer Associates, Sunderland, Massachusetts, 3rd edition.

Hoai, N. X., BobMcKay, R. I., and Essam, D. L. (2006). Representation and structural difficulty in genetic programming. *IEEE Trans. Evolutionary Computation*, 10(2):157–166.

Jones, T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 184–192.

Julstrom, B. A. (1999). Redundant genetic encodings may not be harmful. In Banzhaf, W., Daida, J., Eiben, A. E., Garzon, M. H., Honavar, V., Jakiela, M., and Smith, R. E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference: Volume 1*, page 791, San Francisco, CA. Morgan Kaufmann Publishers.

Kimura, M. (1964). Diffusion models in population genetics. *J. Appl. Prob.*, 1:177–232.

Kimura, M. (1983). *The Neutral Theory of Molecular Evolution*. Cambridge University Press.

Knowles, J. D. and Watson, R. A. (2002). On the utility of redundant encodings in mutation-based evolutionary search. In Merelo, J. J., Adamidis, P., Beyer, H.-G., Fernandez-Villacanas, J.-L., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature, PPSN VII*, pages 88–98, Berlin. Springer-Verlag.

Liepins, G. E. and Vose, M. D. (1990). Representational issues in genetic optimization. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:101–115.

Lobo, F. G., Goldberg, D. E., and Pelikan, M. (2000). Time complexity of genetic algorithms on exponentially scaled problems. In Whitley, D., Goldberg, D. E., Cantú-Paz, E., Spector, L., Parmee, L., and Beyer, H.-G., editors, *Proceedings of the Genetic and Evolutionary Computation Conference 2000*, pages 151–158, San Francisco, CA. Morgan Kaufmann Publishers.

Moraglio, A. and Poli, R. (2004). Topological interpretation of crossover. In Deb, Kalyanmoy et al., editor, *gecco2004*, pages 1377–1388, Heidelberg. Springer.

Palmer, C. C. (1994). *An approach to a problem in network design using genetic algorithms*. unpublished PhD thesis, Polytechnic University, Troy, NY.

Radcliffe, N. J. (1992). Non-linear genetic representations. In Männer, R. and Manderick, B., editors, *Parallel Problem Solving from Nature- PPSN II*, pages 259–268, Berlin. Springer-Verlag.

Radcliffe, N. J. (1997). Theoretical foundations and properties of evolutionary computations: schema processing. In Bäck, T., Fogel, D. B., and Michalewicz, Z., editors, *Handbook of Evolutionary Computation*, pages B2.5:1–B2.5:10. Institute of Physics Publishing and Oxford University Press, Bristol and New York.

Raidl, G. R. (2000). An efficient evolutionary algorithm for the degree-constrained minimum spanning tree problem. In *Proceedings of 2000 IEEE International Conference on Evolutionary Computation*, pages 43–48, Piscataway, NJ. IEEE.

Reeves, C. (2000). Fitness landscapes: A guided tour. Joint tutorials of SAB 2000 and PPSN 2000, tutorial handbook.

Ronald, S. (1997). Robust encodings in genetic algorithms: A survey of encoding issues. In *Proceedings of the Forth International Conference on Evolutionary Computation*, pages 43–48, Piscataway, NJ. IEEE.

Ronald, S., Asenstorfer, J., and Vincent, M. (1995). Representational redundancy in evolutionary algorithms. In *1995 IEEE International Conference on Evolutionary Computation*, volume 2, pages 631–636, Piscataway, NJ. IEEE Service Center.

Rothlauf, F. (2002). *Representations for Genetic and Evolutionary Algorithms*. Number 104 in Studies on Fuzziness and Soft Computing. Springer, Heidelberg, 1 edition.

Rothlauf, F. and Goldberg, D. E. (2003). Redundant representations in evolutionary computation. *Evolutionary Computation*, 11(4):381–415.

Shackleton, M., Shipman, R., and Ebner, M. (2000). An investigation of redundant genotype-phenotype mappings and their role in evolutionary search. In *Proceedings of the 2000 Congress on Evolutionary Computation CEC00*, pages 493–500, La Jolla Marriott Hotel La Jolla, California, USA. IEEE Press.

Shipman, R. (1999). Genetic redundancy: Desirable or problematic for evolutionary adaptation? In *Proceedings of the 4th International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA)*, pages 1–11. Springer Verlag.

Shipman, R., Shackleton, M., Ebner, M., and Watson, R. (2000a). Neutral search spaces for artificial evolution: A lesson from life. In Bedau, M., McCaskill, J., Packard, N., and Rasmussen, S., editors, *Proceedings of Artificial Life VII*, page section III (Evolutionary and Adaptive Dynamics). MIT Press.

Shipman, R., Shackleton, M., and Harvey, L. (2000b). The use of neutral genotype-phenotype mappings for improved evoutionary search. *British Telecom Technology Journal*, 18(4):103–111.

Smith, T., Husbands, P., and O'Shea, M. (2001a). Evolvability, neutrality and search space. Technical Report 535, School of Cognitive and Computing Sciences, University of Sussex.

Smith, T., Husbands, P., and O'Shea, M. (2001b). Neutral networks and evolvability with complex genotype-phenotype mapping. In *Proceedings of the European Converence on Artificial Life: ECAL2001*, volume LNAI 2159, pages 272–281, Berlin. Springer.

Smith, T., Husbands, P., and O'Shea, M. (2001c). Neutral networks in an evolutionary robotics search space. In of Electrical, I. and Engineers, E., editors, *Proceedings of 2001 IEEE International Conference on Evolutionary Computation*, pages 136–145, Piscataway, NJ. IEEE Service Center.

Surry, D. and Radcliffe, N. (1996a). Formal algorithms + formal representations = search strategies. *Parallel Problem Solving from Nature- PPSN IV*, pages xx–xx.

Surry, D. and Radcliffe, N. (1996b). Formal algorithms + formal representations = search strategies. In Voigt, H.-M., Ebeling, W., Rechenberg, I., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature- PPSN IV*, pages 366–375, Berlin. Springer-Verlag.

Thierens, D., Goldberg, D. E., and Pereira, Â. G. (1998). Domino convergence, drift, and the temporal-salience structure of problems. In of Electrical, I. and Engineers, E., editors, *Proceedings of 1998 IEEE International Conference on Evolutionary Computation*, pages 535–540, Piscataway, NJ. IEEE Service Center.

Toussaint, M. and Igel, C. (2002). Neutrality: A necessity for self-adaptation. In Fogel, D. B., El-Sharkawi, M. A., Yao, X., Greenwood, G., Iba, H., Marrow, P., and Shackleton, M., editors, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1354–1359. IEEE Press.

Watson, R. A., Hornby, G. S., and Pollack, J. B. (1998). Modeling building-block interdependency. *Parallel Problem Solving from Nature, PPSN V*, pages 97–106.

Whitley, D. (2000). Walsh analysis, schemata, embedded landscapes and no free lunch. Joint Tutorials of SAB 2000 and PPSN 2000.

Yu, T. and Miller, J. (2001). Neutrality and evolvability of Boolean function landscapes. In *Proceedings of the 4th European Conference on Genetic Programming (EuroGP)*, volume LNCS 2038, pages 204–217. Springer.

Yu, T. and Miller, J. (2002). Finding needles in haystacks is not hard with neutrality. In *Proceedings of the 5th European Conference on Genetic Programming (EuroGP)*, volume LNCS, pages 13–25. Springer.