Representations for Evolutionary Algorithms

Amsterdam, July 2013

Franz Rothlauf
Johannes Gutenberg Universität Mainz
D-55099 Mainz/Germany
rothlauf@uni-mainz.de

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

About the Presenter

- Franz Rothlauf received a Diploma in Electrical Engineering from the University of Erlangen, Germany, a Ph.D. in Information Systems from the University of Bayreuth, Germany, a Habilitation from the University of Mannheim, Germany, in 1997, 2001, and 2007, respectively.
- He is currently chair of Information Systems, Johannes Gutenberg-University Mainz, Germany. He has published more than 50 technical papers in the context of evolutionary computation, co-edited several conference proceedings, and is author of the books "Representations for Genetic and Evolutionary Algorithms" and "Design of Modern Heuristics".
- His main research interests are problem representations for heuristic search approaches especially evolutionary algorithms. He is a member of the Editorial Board of Evolutionary Computation Journal. He has been organizer of several workshops on representations issues, chair of EvoWorkshops in 2005 and 2006, co-organizer of the European workshop series on "Evolutionary Computation in Communications, Networks, and Connected Systems", co-organizer of the European workshop series on "Evolutionary Computation in Transportation and Logistics", and co-chair of the program commitee of the GA track at GECCO 2006. He was conference chair of GECCO 2009. He is member of the executive board of SIGEVO and treasurer of SIGEVO.

Objectives of the Tutorial

- Illustrate the influence of representations on the performance of EAs.
- Illustrate the relationship between problem difficulty and used for 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

Agenda

1. A Short Intro to Representations
   1. Defining Representations
   2. Standard Genotypes
   3. Representations and Operators
   4. Direct versus Indirect Representations
2. Design Guidelines for Representations
3. Properties of Representations
   1. Locality
   2. Redundancy

Basics: Modern heuristics

- Modern heuristics
  - Can be applied to a wide range of problems
  - Use intensification (exploitation) and diversification (exploration) steps
- Intensification steps shall improve quality
- Diversification explores new areas of search space, also accepting complete or partial solutions that are inferior to current solution.

Rothlauf: Representations for Evolutionary Algorithms 5

Basics: Modern Heuristics

- Start with one or more solutions
  - One solutions: Local search methods
  - Population of solutions: Recombination-based methods
- In iterative steps modify solution(s) to generate one or more new solution(s)
- New solutions are created by search operators (variation operators)
- Regularly perform intensification and exploration phases
  - During intensification, use objective function value and focus variation on high-quality solutions
  - During diversification, objective function less relevant. Modify solutions such that new areas of search space are explored.

Rothlauf: Representations for Evolutionary Algorithms 6

Intro – Defining Representations

Defining Representations (1)

- A representation assigns genotypes to corresponding phenotypes.
- Every search and optimization algorithms needs a representation.
- The representation allows us 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.
- There are many different representations.
- Standard representations are binary, real-valued vectors, messy encodings, trees ...
- ... and we assume that everybody has some experience at least with some of them.

Rothlauf: Representations for Evolutionary Algorithms 7

Intro – Defining Representations

Defining Representations (2)

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

  $f_g(\boldsymbol{x}_g): \Phi_g \rightarrow \Phi_p,$

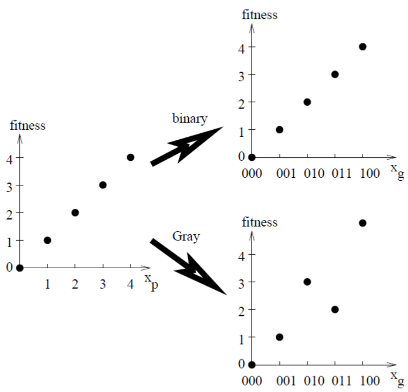
  $f_p(\boldsymbol{x}_p): \Phi_p \rightarrow \mathbb{R},$

  where $f = f_p \circ f_g = f_p(f_g(x_g))$
- A change of f_g also changes the properties of f
- The genetic operators mutation and crossover are applied to x_g whereas the selection process is based on the fitness of x_p
- $f_p\ (x_p)$ determines the fitness and complexity of the problem
- $f_g\ (x_g)$ determines the used representations
- There are $||\Phi_g||$ different representations!

Rothlauf: Representations for Evolutionary Algorithms 8

Representations make the difference

- Representations change the character and difficulty of optimization problems
- Example $f_p = x_p$, $x \in N$
- Different problem depending on the used representation (Gray versus Binary)



Representations make the difference (2)

- Phenotype 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).
- Search performance depends on used search method. If other search methods (e.g. different operators) are used, then search performance is different (compare Reeves, 2000).

Binary Genotypes

- Commonly used in Genetic Algorithms
- Recombination is main operator, mutation used as background noise.
- Search space is $\Phi_g = \{0,1\}^l$ where l is length of a binary vector $x^g = (x^g_1, \ldots, x^g_l)$
- Representation (genotype-phenotype mapping) depends on problem to be solved
- Sometimes natural for combinatorial problems
- When using binary representations for integers, decide between unary, Gray, or binary.
- Are binary genotypes are good choice for continuous phenotypes?

Integer Genotypes

- Use χ-ary alphabet instead of binary, where $\{\chi \in N | \chi > 2\}$
- Instead of coding 2^l solutions, size of search space becomes χ^l
- Fine for integer phenotypes.
- Definition of recombination operators can be difficult (e.g. permutation problems).

Continuous Genotypes

- The search space is Φ_g = R^l where l is the size of the real-valued vector
- Often used in evolution strategies, rely on local search
- Can also encode permutations, trees, schedules, or tours.

Representations, Operators, and Metrics

- Representation, metric defined on Φ_g and Φ_p, and genetic operators 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 individuals and allows measuring similarities between individuals.
  - In general, the metric used for Φ_p is defined by the problem. The metric used for Φ_g is determined by the used search operators.
  - Genotype operators like mutation and crossover are defined based on the used metric.

Representations, Operators, and Metrics (2)

- Mutation:
  - The application of mutation to a genotype individual results in a new individual with similar properties. There is a small genotype distance between offspring and parent.

- What happens if mutation (small genotype change) does not result in a small phenotype change? ->low locality

Representations, Operators, and Metrics (3)

- Crossover:
  - Crossover combines the genotype properties of two or more parents in an offspring. The genotype distance between offspring and parent should be equal or smaller than the distance between both parents.
  - Basic idea of "geometric crossover" from Moraglio and Poli (2004); compare also Surry and Radcliffe (1996), Liepins and Vose (1990), or Rothlauf, (2002)
  - What happens if genotype and phenotype distances do not fit? -> low locality

Representations, Operators, and Metrics (4)

- 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)

Drawbacks of Direct Representations

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

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

This means, f_g is the identity function $f_g(x_g)=x_g$. Using direct representations do not necessarily 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 $f_g(x_g)=x_g$).

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.
- Example:
  - Standard GP (expression tree representation and subtree swapping crossover) cannot solve problems where optimal solutions require very full or very narrow trees (Daida et al., 2001). This is an inherent problem of the construction of the phenotype from the genotype (Hoai et al., 2006).
  - Linear GP (e.g. Grammatical Evolution) has larger problems finding optimal full trees. Some phenotypes can not be encoded (binary LID problem)

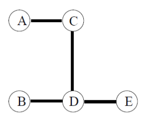
Benefits of indirect representations

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.

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.

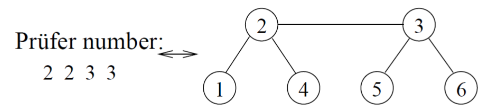


⇕

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

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 us to consider the constraint that the graph is a tree (For other representations repair operators are necessary).

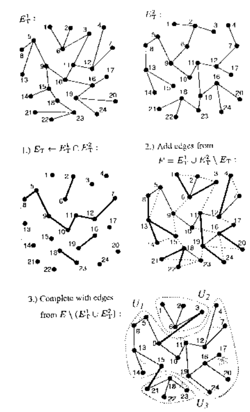


Standardized operators

- When mapping many different types of phenotypes on only a few types of different genotypes (binary, integer, or continuous), it is possible to use standardized operators.
- Behavior of EAs for standard genotypes like binary (simple GAs) or continuous (evolution strategies) genotypes is 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).

Problem-specific operators

Developing of problem-specific operators is difficult and often additional repair mechanisms must be used to ensure a valid solution

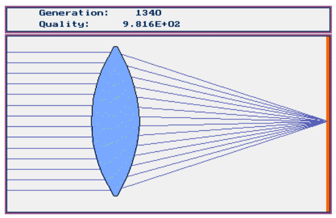


from Raidl (2000)

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

Problem-specific operators (2)

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



Ultimate Goal for Design of Representations

Incorporate problem-specific knowledge such that EA performance increases:

- 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 particular optimization method.

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 much 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. No 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 EAs is possible if the following properties are considered (Surry and Radcliffe, 1996)
  - **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 allows the search method to reach any point in the search space.

Representation Invariant Genetic Operators (Rowe et al., 2010)

- Fact: Performance of genetic algorithms using one-point crossover depends on order of objects (e.g. knapsack problem). Thus, one-point crossover is not invariant under changes in the order of objects.
- Evolutionary operators are invariant with respect to a set of representations if EA performance is independent of used representation (how objects are encoded).
- Rowe et al. (2010) propose an approach to generate invariant search operators.
- Examples for appropriate (representation-independent) search operators for some types of problems (subset problems, permutation problems, and balanced partition problems).

Palmer’s Recommendations (Palmer, 1994)

- 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 th genotype should result in small changes in the phenotyp (compare statements about metric).

Ronald‘s Recommendations (Ronald, 1997)

- 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.
- 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.
- Redundant encodings should not be used?!?

Summary

- Based on observations for specific test problems there are some common, but fuzzy ideas about what makes a good representation.
- Some recommendations are too general to be helpful for designing or evaluating representations.
- Analytical models describing the influence of representations on EAs are on their way.
- To verify (or reject) observations analytical models are necessary.

Locality

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

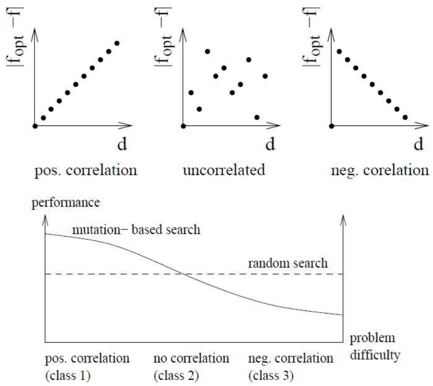
Locality of a Representation

- The locality of a representation describes how well neighboring genotypes correspond to neighboring phenotypes.
- Locality of a representation is high, if neighboring genotypes correspond to neighboring phenotypes.
- Locality describes how well the phenotype metric fits to the genotype metric. If they fit well, locality is high.
- Representations f_g that change the distances between corresponding genotypes and phenotypes modify the performance of particular optimization problems (method performance(f) $\neq$ method performance(f_p)).

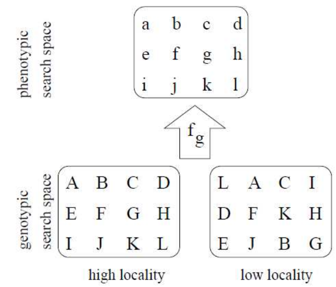
Different Phenotype-Fitness Mappings (Jones and Forrest, 1995)

- Class 1: 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.
- 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.
- 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

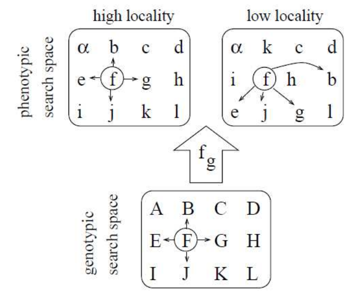
Different Phenotype-Fitness Mappings (2)



Low-locality versus high-locality representations



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



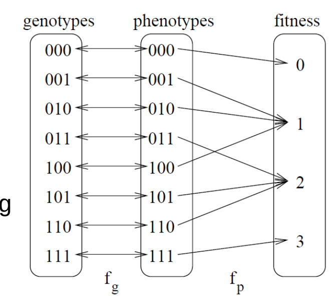
Effect of mutation for high versus low-locality representations

Low-locality 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.

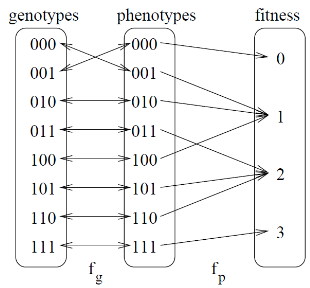
An Example

- 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.

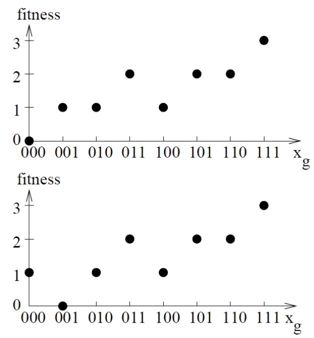


An Example

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



An Example

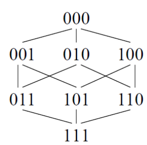


- High-locality representation.
- Problem easy for Selectorecombinative GAs.

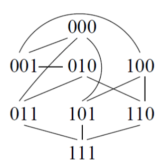
- Different fitness values for genotypes 000 and 001.
- Problem more difficult for selectorecombinative GAs.
- Neighborhood not preserved by representation.

An Example

- Neighborhood structure of the genotypes



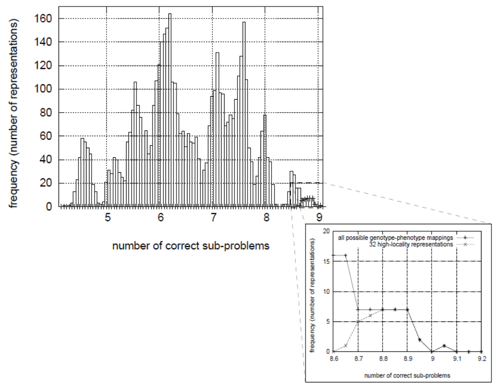
- Resulting neighborhood structure of phenotypes



An Example: comparing representations

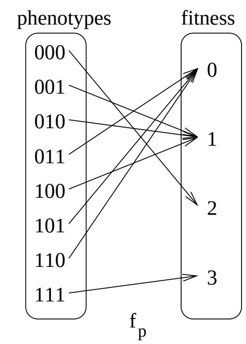
- We compare the performance of selectorecombinative Gas over all different representations for the one-max problem.
- When focusing on binary bitstrings and assigning l-bit genotypes to l-bit phenotypes, there are 2^l! different representations.
- For l=3 there are 8 different genotypes and phenotypes, resp., and 8! = 40,320 different representations.
- 36 different representations result in the same overall problem f (for the one-max problem).
- To reduce problem complexity, x_g = 111 is always assigned to x_p=111. Therefore, we consider 7! = 5.040 different representations.
- We concatenate ten 3-bit problems and use a GA with tournament selection of size 2, uniform crossover, and N=16.

An Example: Results

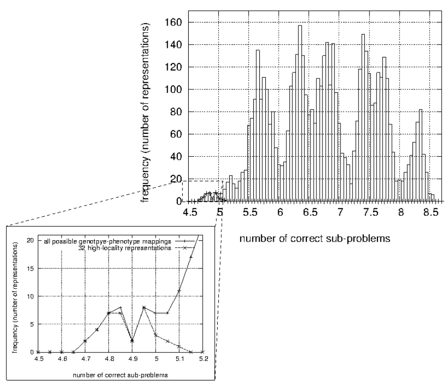


An Example: Deceptive Trap

- We compare the performance of selectorecombinative GAs over all different representations for the deceptive trap prtexoblem.
- 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.



An Example: Results for Deceptive Traps



Conclusions on Locality

- When using high locality representations, genotype neighbors correspond to phenotype 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).

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.

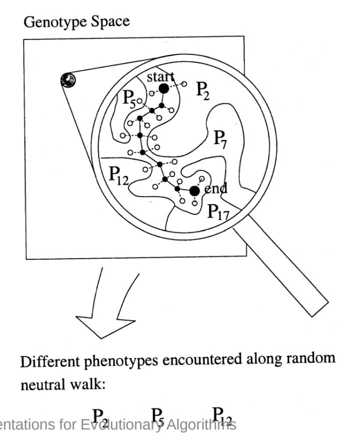
- Redundant representations reduce EA performance due to loss of diversity (Davis, 1989; Eshelman and Schaffer, 1991; Ronald et al., 1995)
- Redundant representations increase EA performance (Gerrits and Hogeweg, 1991; Cohoon et al., 1988; Julstrom, 1999)

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) have been used in EAs with great enthusiasm.
- There was hope that increasing the "evolvability of a system" (reachability of solutions) 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).
- This is not true! (Knowles and Watson, 2002)

Redundant representations (4)

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



Ebner et al. (2001)

Redundant representations (5)

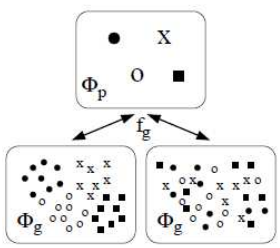
- Benefits of Neutral Networks
  - Population can drift along these neutral networks.
  - Reducing the chance of being trapped in sub-optimal solutions.
  - Population is quickly able to recover after a change has occurred.
  - Evolvability and connectivity of the system increases.
- Problems
  - Higher evolvability and connectivity **randomizes search.** No guided search is possible any more.
  - Genetic drift?

Synonymous and Non-Synonymous Redundancy

- We study
  - how to distinguish between synonymously and non-synonymously redundant encodings (Rothlauf and Goldberg, 2003),
  - how synonymous redundancy changes performance of EAs (quantitative predictions) (Rothlauf and Goldberg, 2003),and
  - the properties of non-synonymously redundant representations (Choi and Moon, 2003; Choi and Moon, 2008).

Categorization of Redundancy

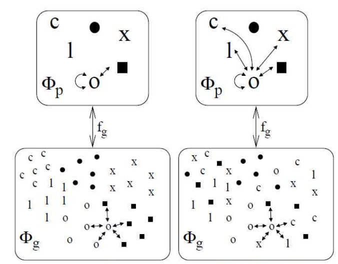
- For redundant representations, we can distinguish between:
  - **Synonymously redundant representations**: All genotypes that encode the same phenotype are similar to each other.
  - **Non-synonymously redundant representations**: Genotypes that encode the same phenotype are not similar to each other.



synonymous non−synonymous

Non-synonymous Redundancy

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



synonymous non−synonymous

Effects of small mutation steps

Non-synonymous Redundancy

- 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, 2008).

Non-synonymous Redundancy

Some selected examples for problems with maximally non-synonymous redundant encodings :

- **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

Synonymous Redundancy

- Synonymously redundant representations can be described using
  - order of redundancy $k_r = \frac{\log|\Phi_p|}{\log|\Phi_g|}$
  - 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 genotype BBs of order k_g that represent the optimal phenotype BB of order k_p.

Examples of synonymously redundant representation

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

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

Examples of synonymously redundant representation

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

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

Excursus: 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_o 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

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

Excursus: Population sizing for GAs (2)

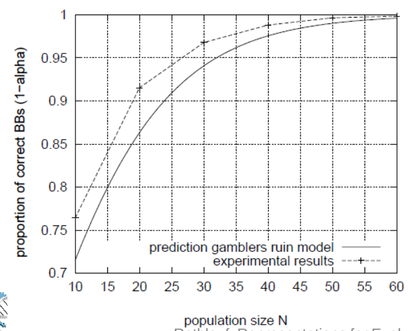
- How does population size N depends on α?

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

- N is the necessary population size, α = 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.

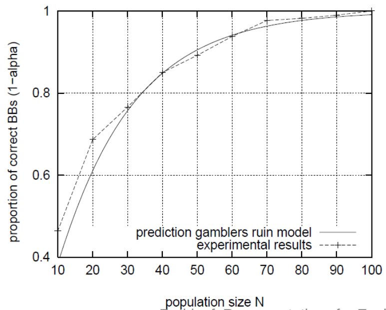
Excursus: Population sizing for GAs (3)

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



Excursus: Population sizing for GAs (4)

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



What about synonymously redundant representations?

- How does the redundancy of a representation influence GA performance?
- Observation: Redundant representation change the initial supply x_o of BBs.
- For binary problem representation:

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

where N is the population size.

Performance of GAs using synonymously redundant representations

- When using synonymously redundant representations the Gambler's ruin 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−α goes with $O\left(\frac{2^{k_r}}{r}\right)$

Example: 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∈ {1,…,k_r}.

Examples of Trivial Voting Mappings

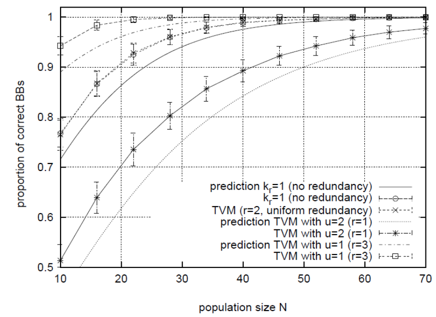
- k=1
- k_r=3
- u=2

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

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

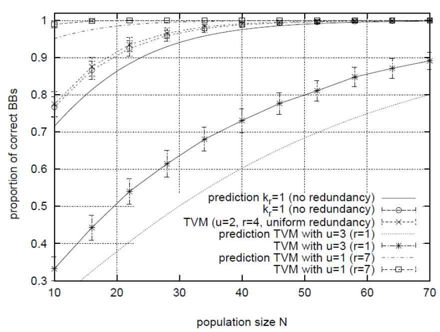
genotypes x_g	x_p
000	0
001, 010, 100,110, 101, 011, 111	1

Results for trivial voting mapping



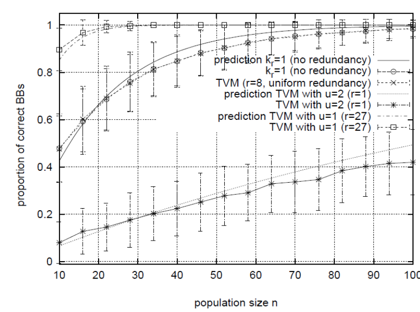
Experimental and theoretical results of the proportion of correct BBs on a 150-bit one-max problem using the trivial voting mapping for k_r=2.

Results for trivial voting mapping (2)



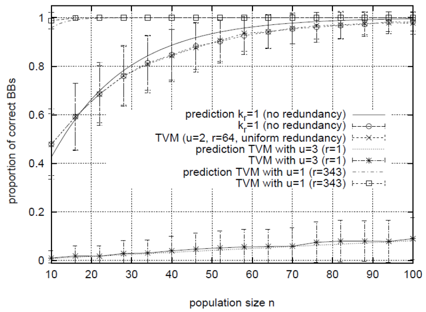
Experimental and theoretical results of the proportion of correct BBs on a 150-bit one-max problem using the trivial voting mapping for k_r=3.

Results for trivial voting mapping (3)



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

Results for trivial voting mapping (3)



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

Summary: Synonymously Redundant Representations

- 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!

Cookbook: How to deal with redundant representations?

1. How does the redundant representation change the size of the search space?
   1. Are additional phenotypes encoded?
   2. Are some phenotypes not encoded?
2. Is the representation non-synonymously redundant?
   1. yes -> you have a problem: guided search fails and only traps can be solved!
   2. no -> fine. We have a synonymously redundant encoding.
3. Is the representation uniformly redundant?
   1. yes -> fine! EA performance is not affected by redundancy.
   2. no -> Be careful! Which types of solutions are overrepresented? EAs perform only well if high-quality solutions are overrepresented.

Take home message for redundant representations

- There are theoretical models that allow us to predict the expected GA performance when using redundant representations.
- Do **not use** non-synonymously redundant representations!
- If there is **no** knowledge about the optimal solution use a uniformly redundant representation.

Thanks for your attention and interest!

Further reading

- Rothlauf, F. (2006). Representations for Genetic and Evolutionary Algorithms. Springer, Berlin.
- Rothlauf, F. (2011). Design of Modern Heuristics. Springer, Berlin.

References

- Bäck, T., Michalewicz, Z., and Yao, X., editors (1997). Proceedings of the Fourth International Conference on Evolutionary Computation, Piscataway. IEEE Service Center.
- 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. E., editors, Proceedings of the 6th International Conference on Artificial Life (ALIFE-98), pages 18–27, Cambridge. MIT Press.
- Barnett, L. (2001). Netcrawling - optimal evolutionary search with neutral networks. In (Kim et al., 2001), pages 30–37.
- 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 Francisco. Morgan Kaufmann.
- Choi, S.-S. and Moon, B.-R. (2003). Normalization in genetic algorithms. In Cant´u-Paz, E., Foster, J. A., Deb, K., Davis, D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K., Jonoska, N., and Miller, J., editors, Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2003, pages 862–873, Heidelberg. Springer.

- Choi, S.-S. and Moon, B.-R. (2008). Normalization for genetic algorithms with nonsynonymously redundant encodings. IEEE Trans. Evolutionary Computation, 12(5):604–616.
- 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, Piscataway. 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, Burlington. Morgan Kaufmann.
- Ebner, M., Langguth, P., Albert, J., Shackleton, M., and Shipman, R. (2001). On neutral networks and evolvability. In (Kim et al., 2001), pages 1–8.
- 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, Burlington. 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.
- Fonseca, C., Kim, J.-H., and Smith, A., editors (2000). Proceedings of 2000 IEEE Congress on Evolutionary Computation, Piscataway. IEEE Press.
- 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. 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 M¨anner, R., editors, Parallel Problem Solving from Nature - PPSN I, volume 496 of LNCS, pages 70–74, Berlin. Springer.
- Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading.
- Goldberg, D. E. (1991). Real-coded genetic algorithms, virtual alphabets, and blocking. Complex Systems, 5(2):139–167.
- Harik, G. R., Cant´u-Paz, E., Goldberg, D. E., and Miller, B. L. (1997). The gambler's ruin problem, genetic algorithms, and the sizing of populations. In (Bäck et al., 1997), pages 7–12.

- Hoai, N. X., McKay, 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, GECCO '99, page 791, Burlington. Morgan Kaufmann.
- Kim, J.-H., Zhang, B.-T., Fogel, G., and Kuscu, I., editors (2001). Proceedings of 2001 IEEE Congress on Evolutionary Computation, Piscataway. IEEE Press.
- 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 mutationbased 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.
- Liepins, G. E. and Vose, M. D. (1990). Representational issues in genetic optimization. Journal of Experimental and Theoretical Artificial Intelligence, 2:101–115.

- Moraglio, A. and Poli, R. (2004). Topological interpretation of crossover. In Deb, K., Poli, R., Banzhaf, W., Beyer, H.-G., Burke, E., Darwen, P., Dasgupta, D., Floreano, D., Foster, J., Harman, M., Holland, O., Lanzi, P. L., Spector, L., Tettamanzi, A., Thierens, D., and Tyrrell, A., editors, gecco2004, pages 1377–1388, Heidelberg. Springer.
- Palmer, C. C. (1994). An approach to a problem in network design using genetic algorithms. PhD thesis, Polytechnic University, Brooklyn, NY.
- Radcliffe, N. J. (1992). Non-linear genetic representations. In M¨anner, R. and Manderick, B., editors, Parallel Problem Solving from Nature – PPSN II, pages 259–268, Berlin. Springer.
- 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 (Fonseca et al., 2000), pages 43–48.
- Reeves, C. R. (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 (Bäck et al., 1997), pages 43–48.

- Ronald, S., Asenstorfer, J., and Vincent, M. (1995). Representational redundancy in evolutionary algorithms. In Fogel, D. B. and Attikiouzel, Y., editors, Proceedings of the 1995 IEEE International Conference on Evolutionary Computation, volume 2, pages 631–636, Piscataway. IEEE Service Center.
- Rothlauf, F. (2002). Representations for Genetic and Evolutionary Algorithms. Number 104 in Studies on Fuzziness and Soft Computing. Springer, Heidelberg, 1st edition.
- Rothlauf, F. and Goldberg, D. E. (2003). Redundant representations in evolutionary computation. Evolutionary Computation, 11(4):381–415.
- Rowe, J. E., Vose, M. D., and Wright, A. H. (2010). Representation invariant genetic operators. Evolutionary Computation, 18(4):635–660.
- Shackleton, M., Shipman, R., and Ebner, M. (2000). An investigation of redundant genotypephenotype mappings and their role in evolutionary search. In (Fonseca et al., 2000), pages 493–500.
- Shipman, R. (1999). Genetic redundancy: Desirable or problematic for evolutionary adaptation? In Dobnikar, A., Steele, N. C., Pearson, D. W., and Albrecht, R. F., editors, Proceedings of the 4th International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA), pages 1–11, Berlin. Springer.
- 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 (Kim et al., 2001), pages 136–145.
- Surry, P. D. and Radcliffe, N. (1996). 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.

- 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 2002 IEEE Congress on Evolutionary Computation, pages 1354–1359, Piscataway. IEEE Press.
- Watson, R. A., Hornby, G. S., and Pollack, J. B. (1998). Modeling building-block interdependency. In Eiben, A. E., B¨ack, T., Schoenauer, M., and Schwefel, H.-P., editors, Parallel Problem Solving from Nature – PPSN V, volume 1498 of LNCS, pages 97–106, Berlin. Springer.
- Whitley, L. 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 Miller, J., Tomassini, M., Lanzi, P. L., Ryan, C., Tetamanzi, A. G. B., and Langdon, W. B., editors, Proceedings of the Fourth European Conference on Genetic Programming (EuroGP-2001), volume 2038 of LNCS, pages 204–217, Berlin. Springer.
- Yu, T. and Miller, J. (2002). Finding needles in haystacks is not hard with neutrality. In Foster, J. A., Lutton, E., Miller, J., Ryan, C., and Tettamanzi, A. G. B., editors, Proceedings of the Fifth European Conference on Genetic Programming (EuroGP- 2002), volume 2278 of LNCS, pages 13–25, Berlin. Springer.