5. Epistasis. Linkage identification. Optimization by model fitting.

Petr Pošík

Dept. of Cybernetics
ČVUT FEL

Contents

Introduction to Epistasis 3 / 22

Black-box Optimization and Genetic Algorithms

In BBO, we need to specify:

- ✔ the representation of candidate solution
- ✔ the objective function (gives the quality of a candidate solution)
  - ✘ can have almost any form (non-differentiable, discontinuous, multimodal, noisy, . . .)

How to solve BBO problems?

- ✔ Algorithm applicable in the BBO scenario can only provide a candidate solution and have the objective function to evaluate it.
- ✔ It cannot require (assume, take advantage of) any other knowledge about the objective function. (It can estimate the needed knowledge. . .)
- ✔ Hill-climbing, simulated annealing, taboo, GAs, . . .

GAs are popular:

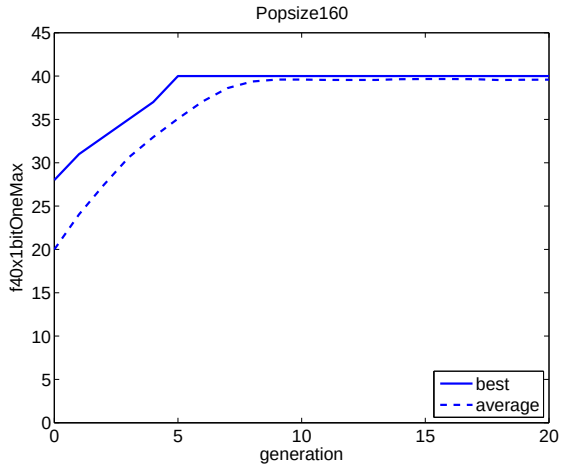
- ✔ they are easy to use,
- ✔ applicable without prior knowledge, and
- ✔ easy to parallelize.

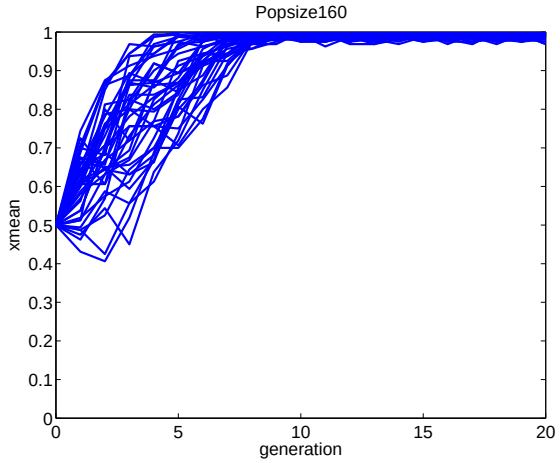
GAs are great, but not perfect!!!

GA works well...

Problem f_1:

- ✔ defined over 40-bit strings
- ✔ the quality of the worst solution: $f_1(x^{\text{worst}}) = 0$.
- ✔ the quality of the best solution: $f_1(x^{\text{opt}}) = 40$.
- ✔ the best solution: $x^{\text{opt}} = (1111\ldots1)$.



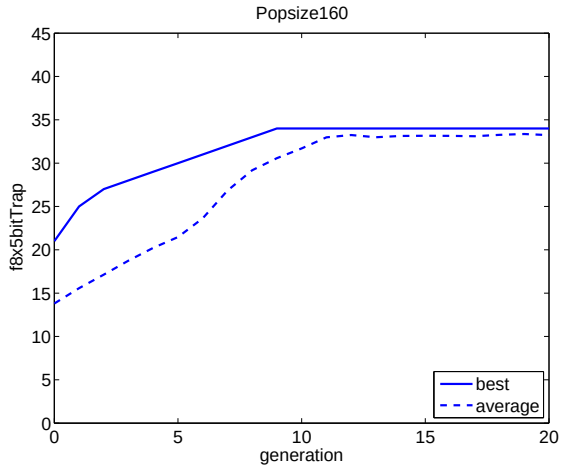


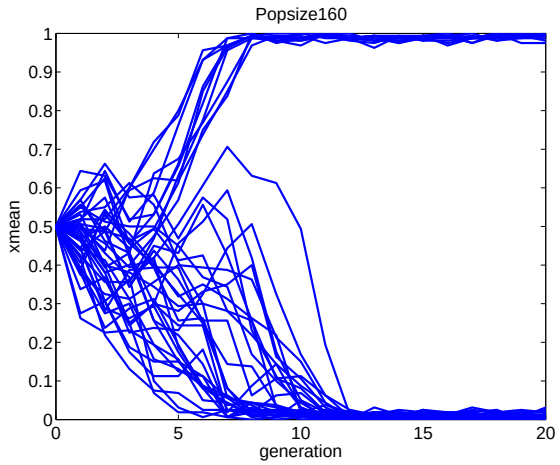
The f_1 problem contains no epistatic interactions among design variables.

GA fails...

Problem f_2:

- ✔ defined over 40-bit strings
- ✔ the quality of the worst solution: $f_2(x^{\text{worst}}) = 0$.
- ✔ the quality of the best solution: $f_2(x^{\text{opt}}) = 40$.
- ✔ the best solution: $x^{\text{opt}} = (1111\ldots1)$.





The f_2 problem contains some interactions among variables, *GA is not aware of them* and works with the individual bits as if they were truly independent of each other.

None of the above mentioned problem characteristics is important to judge if the GA will work well!!!

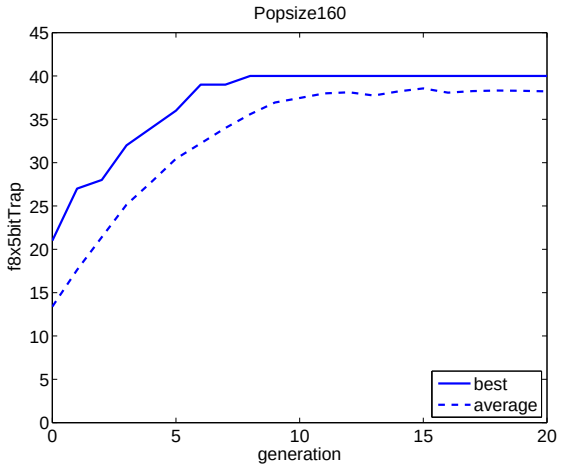
GA works again...

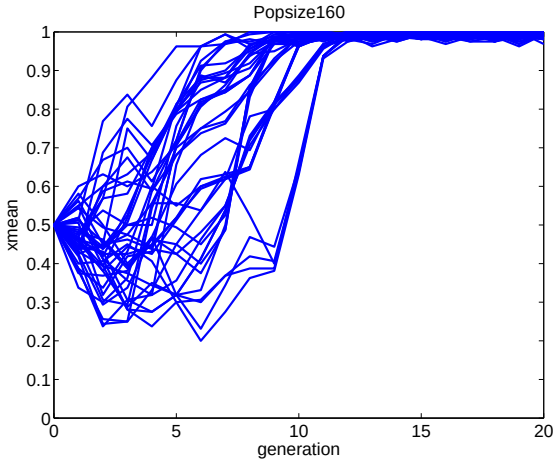
Still solving f_2:

- ✔ defined over 40-bit strings
- ✔ the quality of the worst solution: $f_2(x^{\text{worst}}) = 0$.
- ✔ the quality of the best solution: $f_2(x^{\text{opt}}) = 40$.
- ✔ the best solution: $x^{\text{opt}} = (1111\ldots1)$.

Instead of the uniform crossover,

- ✔ let us allow the crossover only after each 5th bit.





The f_2 problem contains some interactions among variables and *GA knows about them*.

Epistasis

Epistasis:

- ✔ Effects of one gene are dependent on (influenced, conditioned by) other genes.

Linkage:

- ✔ Tendency of certain loci or alleles to be inherited together.

Other names:

- ✔ dependencies
- ✔ interdependencies
- ✔ interactions

When optimizing the following functions, which of the variables are linked together?

$$f = x_1 + x_2 + x_3 \tag{1}$$
$$f = 0.1x_1 + 0.7x_2 + 3x_3 \tag{2}$$
$$f = x_1 x_2 x_3 \tag{3}$$
$$f = x_1 + x_2^2 + \sqrt{x_3} \tag{4}$$
$$f = \sin(x_1) + \cos(x_2) + e^{x_3} \tag{5}$$
$$f = \sin(x_1 + x_2) + e^{x_3} \tag{6}$$

Discussion on semestral projects

Which of the semestral projects contain interactions? Why?

- ✔ Japanese puzzle
- ✔ Circles in the square
- ✔ Image compression
- ✔ MTSP
- ✔ CD compilation
- ✔ Shortest common supersequence
- ✔ ATP rankings
- ✔ Binary opt. problem

Notes:

- ✔ Almost all real-world problems contain interactions among design variables.
- ✔ The "amount" and "type" of interactions depend on the representation and the objective function.
- ✔ Sometimes, by a clever choice of the representation and the objective function, we can get rid of the interactions.

Linkage Identification Techniques

Problems:

- ✔ How to detect dependencies among variables?
- ✔ How to use them?

Techniques used for linkage identification:

1. Indirect detection along genetic search (messy GAs)
2. Direct detection of fitness changes by perturbation
3. Model-based approach: classification
4. Model-based approach: distribution estimation (EDAs)

Linkage identification by non-linearity check (LINC)

LINC

- ✔ Developed by Munetomo [MG98], based on gene expression messy GA [Kar95]

Linearity and nonlinearity

- ✔ consider a bitstring $\mathbf{x} = (x_1, x_2, \ldots, x_D)$
- ✔ perturb loci i, j, and both i and j

$\mathbf{x} = (x_1, \ldots, x_i, \ldots, x_j, \ldots, x_D)$	
$\mathbf{x}_i = (x_1, \ldots, \bar{x}_i, \ldots, x_j, \ldots, x_D)$	$\Delta f_i = f(\mathbf{x}_i) - f(\mathbf{x})$
$\mathbf{x}_j = (x_1, \ldots, x_i, \ldots, \bar{x}_j, \ldots, x_D)$	$\Delta f_j = f(\mathbf{x}_j) - f(\mathbf{x})$
$\mathbf{x}_{ij} = (x_1, \ldots, \bar{x}_i, \ldots, \bar{x}_j, \ldots, x_D)$	$\Delta f_{ij} = f(\mathbf{x}_{ij}) - f(\mathbf{x})$

If $\Delta f_{ij} = \Delta f_i + \Delta f_j$,

- ✔ fitness is linear, interaction did not show up,
- ✔ x_i and x_j needn't be part of the same BB.

If $\Delta f_{ij} \neq \Delta f_i + \Delta f_j$,

- ✔ fitness is nonlinear, interaction exists,
- ✔ x_i and x_j are members of the same BB,
- ✔ add i to j's linkage set and j to i's linkage set.

[Kar95] H. Kargupta. Search, polynomial complexity, and the fast messy genetic algorithm. Technical Report IlliGAL Report No. 95008, Illinois Genetic Algorithm Laboratory, Urbana Champaign, Illinois, 1995.

[MG98] Masaharu Munetomo and David E. Goldberg. Identifying Linkage by Nonlinearity Check. Technical Report IlliGAL Report No. 98012, University of Illinois, Urbana-Champaign, October 1998.

LINC: Example

3-bit Trap Function:

$u(\mathbf{x})$	0	1	2	3
$f(\mathbf{x})$	0.6	0.3	0	1

- ✔ Let us consider string $\mathbf{x} = (010)$ and bits 1 and 2:

$\mathbf{x} = (010)$				
$\mathbf{x}_1 = (110)$	Δf_1	$= f(110) - f(010)$	$= 0 - 0.3$	$= -0.3$
$\mathbf{x}_2 = (000)$	Δf_2	$= f(000) - f(010)$	$= 0.6 - 0.3$	$= 0.3$
$\mathbf{x}_{12} = (100)$	Δf_{12}	$= f(100) - f(010)$	$= 0.3 - 0.3$	$= 0$

$\Delta f_{12} = \Delta f_1 + \Delta f_2 \longrightarrow i$ and j needn't be part of the same BB

- ✔ The same string $\mathbf{x} = (010)$ and bits 1 and 3:

$\mathbf{x} = (010)$				
$\mathbf{x}_1 = (110)$	Δf_1	$= f(110) - f(010)$	$= 0 - 0.3$	$= -0.3$
$\mathbf{x}_3 = (011)$	Δf_3	$= f(011) - f(010)$	$= 0 - 0.3$	$= -0.3$
$\mathbf{x}_{13} = (111)$	Δf_{13}	$= f(111) - f(010)$	$= 1 - 0.3$	$= 0.7$

$\Delta f_{13} \neq \Delta f_1 + \Delta f_3 \longrightarrow i$ and j are part of the same BB

Analysis of the string 010 suggests linkage between 1st and 3rd bit only.
Additional string (e.g. 011) would suggest that all three bits are linked together.

Linkage identification by non-monotonicity detection (LIMD)

LIMD:

- ✔ Developed by Munetomo [MG99a, MG99b]
- ✔ Not all nonlinearities detected by LINC are bad for GAs (or hill-climber)
- ✔ Consider 4 bitstrings (as in case of LINC): $\mathbf{x}$, $\mathbf{x}_i$, $\mathbf{x}_j$, and $\mathbf{x}_{ij}$
- ✔ Reorder them so that $\mathbf{x} = \arg\min(f(\mathbf{x}), f(\mathbf{x}_i), f(\mathbf{x}_j), f(\mathbf{x}_{ij}))$
- ✔ Monotonicity constraints:
  - ✘ $f(\mathbf{x}) < f(\mathbf{x}_i) < f(\mathbf{x}_{ij})$
  - ✘ $f(\mathbf{x}) < f(\mathbf{x}_j) < f(\mathbf{x}_{ij})$
- ✔ If monotonicity is violated, add i to j's linkage set and j to i's linkage set.

[MG99a] Masaharu Munetomo and David E. Goldberg. Identifying linkage groups by nonlinearity/non-monotonicity detection. In *Proceedings of the 1999 Genetic and Evolutionary Computation Conference*, pages 433–440, 1999.

[MG99b] Masaharu Munetomo and David E. Goldberg. Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation*, 7:377–398, December 1999.

Tightness detection

For each pair of variables, we know if the two variables are linked. So what?

- ✔ Construct linkage groups (decompose the problem)
- ✔ Use modified crossover: always transfer the bits in the linkage group together
- ✔ Use modified mutation: allow mutations of the whole linkage group

What if the problem cannot be decomposed sufficiently?

- ✔ No clearly separated groups of bits
- ✔ Solution: **tightness detection**

Algorithm:

1. Given: linkage groups from LIMD (or LINC) procedure
2. Tightness definition:

$$tightness(i,j) = \frac{n(i \wedge j)}{n(i \vee j)} \in \langle 0, 1 \rangle,$$

   where
   $n(i \wedge j)$ is the number of linkage groups that contain both i and j, and
   $n(i \vee j)$ is the number of linkage groups that contain i or j.
3. If $tightness(i,j) < \delta$, remove locus j from linkage set of locus i and vice versa.

Modeling the Interactions

What kind of "models" should we use?

- ✔ The perturbation techniques
  - ✘ extract *global information* from the population, and
  - ✘ use modified recombination operators that use the learned information, i.e. they used modified sampling process.
- ✔ Classification techniques can be used to
  - ✘ build a model distinguishing the "good" individuals from the "bad" individuals, and
  - ✘ sample offspring from the "good" areas (more often than from the "bad" ones).
- ✔ Probability distribution models can be used to
  - ✘ build a model describing the distribution of "good" individuals, and
  - ✘ sample offspring from that model.

Optimization by Model Fitting: The Algorithm

Algorithm 1: General Evolutionary Scheme

```
1  begin
2      M^(0) ← InitializeModel()
3      X^(0) ← Sample(M^(0))
4      f^(0) ← Evaluate(X^(0))
5      g ← 1
6      while not TerminationCondition() do
7          {S, D} ← Select(X^(g−1), f^(g−1))
8          M^(g) ← Update(g, M^(g−1), X^(g−1), f^(g−1), S, D)
9          X_Offs ← Sample(M^(g))
10         f_Offs ← Evaluate (X_Offs)
11         {X^(g), f^(g)} ← Replace(X^(g−1), X_Offs, f^(g−1), f_Offs)
12         g ← g + 1
```

Descriptive vs. Generative Models

Many models are *descriptive* in the sense that

- ✔ given a point, they are able to
  - ✘ classify it to the "good" or "bad" class, or
  - ✘ evaluate its probability of being a "good" individual,
- ✔ but they are *not able to generate* a "good" point directly.
- ✔ *Generative* models can be used as data generators.

How to turn a descriptive model into generative?

- ✔ Use (weighted) rejection sampling (see [Luk09, algorithms 115 and 117])

Problems with rejection sampling?

- ✔ In later stages of evolution, the acceptance region becomes very small.
- ✔ It becomes increasingly difficult to hit that region.

[Luk09] Sean Luke. *Essentials of Metaheuristics*. 2009. available at http://cs.gmu.edu/∼sean/book/metaheuristics/.

Learnable Evolution Model (LEM)

LEM: Evolutionary process guided by machine learning

- ✔ created by Ryszard Michalski [Mic00]
- ✔ originally, alternates between 2 phases:
  - ✘ machine learning mode
  - ✘ Darwinian evolution mode
- ✔ uses AQ decision rules as the classification model

LEM3: the most recent implementation of the LEM approach

- ✔ implemented by Janusz Wojtusiak [WM06, Woj07]
- ✔ alternates between more phases including local search, randomization, and representation adjustment
- ✔ good results reported, especially in the initial phases of the search

[Mic00] Ryszard S. Michalski. Learnable evolution model: Evolutionary processes guided by machine learning. *Machine Learning*, 38:9–40, 2000.

[WM06] Janusz Wojtusiak and Ryszard S. Michalski. The LEM3 system for non-darwinian evolutionary computation and its application to complex function optimization. Reports of the Machine Learning and Inference Laboratory MLI 04-1, George Mason University, Fairfax, VA, February 2006.

[Woj07] Janusz Wojtusiak. *Handling Constrained Optimization Problems and Using Constructive Induction to Improve Representation Spaces in Learnable Evolution Model*. Reports of the machine learning and inference laboratory, Fairfax, VA, November 2007.

Summary

- ✔ GAs are not very good for problems containing epistatic interactions among design variables
  - ✘ because the assume all variables independent of each other
- ✔ There are methods that identify interactions and create the right linkage
  - ✘ Perturbation techniques are the most straightforward
  - ✘ Optimization by model fitting is a general approach
- ✔ Using classification models is not so common
  - ✘ LEM is a prominent example
  - ✘ another example for the real-valued spaces using elliptic classifiers will be given later in the course
- ✔ Using probabilistic models of the distribution of promising individuals is very popular:
  - ✘ The class of estimation-of-distribution (EDA) algorithms
  - ✘ EDAs are covered in the next 2 lectures