GECCO 2013 Tutorial

Model-Based Evolutionary Algorithms

Dirk Thierens
Utrecht University
Department of Information and Computing Sciences
Utrecht, The Netherlands

Peter A.N. Bosman
Centrum Wiskunde & Informatica – CWI
(Centre for Mathematics and Computer Science)
Amsterdam, The Netherlands

Copyright is held by the author/owner(s).
ACM 978-1-4503-1964-5/13/07.

Outline

Model-Based Evolutionary Algorithms (MBEA)

- Introduction
- Part I: Discrete Representation
- Part II: Real-Valued, Permutation, and Program Representations

What?

Evolutionary Algorithms
- Population-based, stochastic search algorithms
- Exploitation: selection
- Exploration: mutation & crossover

Model-Based Evolutionary Algorithms
- Population-based, stochastic search algorithms
- Exploitation: selection
- Exploration:
  1. Learn a model from selected solutions
  2. Generate new solutions from the model (& population)

Model-Based Evolutionary Algorithms (MBEA)

- a.k.a. Estimation of Distribution Algorithms (EDAs)
- a.k.a. Probabilistic Model-Building Genetic Algorithms
- a.k.a. Iterated Density Estimation Evolutionary Algorithms

MBEA = Evolutionary Computing + Machine Learning

Note: model not necessarily probabilistic
Why?

Goal: Black Box Optimization
- Little known about the structure of the problem
- Clean separation optimizer from problem definition
- Easy and generally applicable

Approach
- Classical EAs: need suitable representation & variation operators
- Model-Based EAs: learn structure from good solutions

Discrete Representation
- Typically binary representation
- Higher order cardinality: similar approach

Probabilistic Model-Building Genetic Algorithm

Type of Models
- Univariate: no statistical interaction between variables considered.
- Bivariate: pairwise dependencies learned.
- Multivariate: higher-order interactions modeled.

Univariate PMBGA

Model
- Model: probability vector \( [p_1, \ldots, p_\ell] \) (\( \ell \): string length)
- \( p_i \): probability of value 1 at string position \( i \)
- \( p(X) = \prod_{i=1}^\ell p(x_i) \) (\( p(x_i) \): univariate marginal distribution)
- Learn model: count proportions of 1 in selected population
- Sample model: generate new solutions with specified probabilities
Univariate PMBGA

Different Variants

- **PBIL** (Baluja; 1995)
  - Prob. vector incrementally updated over successive generations
- **UMDA** (Mühlenbein, Paass; 1996)
  - No incremental updates: example above
- **Compact GA** (Harik, Lobo, Goldberg; 1998)
  - Models steady-state GA with tournament selection
- **DEUM** (Shakya, McCall, Brown; 2004)
  - Uses Markov Random Field modeling

A hard problem for the univariate FOS

<table>
<thead>
<tr>
<th>Data</th>
<th>Marginal Product (MP) FOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>000000</td>
<td>000 (\cdot) 0.3 (\cdot) 0.3</td>
</tr>
<tr>
<td>111111</td>
<td>111 (\cdot) 0.0 (\cdot) 0.0</td>
</tr>
<tr>
<td>010101</td>
<td>001 (\cdot) 0.0 (\cdot) 0.0</td>
</tr>
<tr>
<td>101010</td>
<td>010 (\cdot) 0.2 (\cdot) 0.2</td>
</tr>
<tr>
<td>000010</td>
<td>011 (\cdot) 0.0 (\cdot) 0.0</td>
</tr>
<tr>
<td>111000</td>
<td>100 (\cdot) 0.0 (\cdot) 0.0</td>
</tr>
<tr>
<td>010111</td>
<td>101 (\cdot) 0.1 (\cdot) 0.1</td>
</tr>
<tr>
<td>111000</td>
<td>110 (\cdot) 0.0 (\cdot) 0.0</td>
</tr>
<tr>
<td>000011</td>
<td>111 (\cdot) 0.4 (\cdot) 0.4</td>
</tr>
<tr>
<td>111111</td>
<td>111 (\cdot) 0.4 (\cdot) 0.4</td>
</tr>
</tbody>
</table>

- What is the probability of generating 111111?
- **Univariate FOS**: \(0.5 \cdot 0.6 \cdot 0.5 \cdot 0.6 \cdot 0.5 = 0.0225\)
- **MP FOS**: \(0.4 \cdot 0.4 = 0.16\) (7 times larger!)

Learning problem structure on the fly

- Without a “good” decomposition of the problem, important partial solutions (building blocks) are likely to get disrupted in variation.
- **Disruption** leads to inefficiency.
- Can we automatically configure the model structure favorably?
- Selection increases proportion of good building blocks and thus “correlations” between variables of these building blocks.
- So, learn which variables are “correlated”.
- See the population (or selection) as a data set.
- Apply statistics / probability theory / probabilistic modeling.

Bivariate PMBGA

Model

- Need more than just probabilities of bit values
- Model pairwise interactions: conditional probabilities

- **MIMIC** (de Bonet, Isbell, Viola; 1996)
  - Dependency Chain
- **COMIT** (Baluja, Davies; 1997)
  - Dependency Tree
- **BMDA** (Pelikan, Mühlenbein; 1998)
  - Independent trees (forest)
### Bivariate PMBGA

**MIMIC**
- Model: chain of pairwise dependencies.
- $p(X) = \prod_{i=1}^{l-1} p(x_{i+1} | x_i) p(x_1)$.
- MIMIC greedily searches for the optimal permutation of variables that minimizes Kullback-Leibler divergence.

**COMIT**
- Optimal dependency tree instead of linear chain.
- Compute fully connected weighted graph between problem variables.
- Weights are the mutual information $I(X, Y)$ between the variables.
- $I(X, Y) = \sum_{y \in Y} \sum_{x \in X} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}$.
- COMIT computes the maximum spanning tree of the weighted graph.

### Bivariate PMBGA

**BMDA**
- BMDA also builds tree model.
- Model not necessarily fully connected: set of trees or forest.
- Pairwise interactions measured by Pearson’s chi-square statistics.

### Multivariate PMBGA

**Marginal Product Model**
- Extended Compact GA (ECGA) (Harik; 1999) was first EDA going beyond pairwise dependencies.
- Greedily searches for the Marginal Product Model that minimizes the minimum description length (MDL).
- $p(X) = \prod_{g=1}^{c} p(X_g)$.
- Choose the probability distribution with the lowest MDL score.
- Start from simplest model: the univariate factorization.
- Join two groups that result in the largest improvement in the used scoring measure.
- Stop when no joining of two groups improves the score further.
Multivariate PMBGA

Minimum Description Length (MDL)

- \[ \text{MDL}(M, D) = D_{\text{Model}} + D_{\text{Data}} \]
- Best factorization = the one with the lowest MDL score.
- MDL is a measure of complexity.
  1. Compressed population complexity: how well the population is compressed by the model (measure of goodness of the probability distribution estimation).
  2. Model complexity: the number of bits required to store all parameters of the model.

Learning MP model

1. Start from univariate FOS:
   \[
   \{\{0\}\}, \{\{1\}\}, \{\{2\}\}, \ldots, \{\{l-2\}\}, \{\{l-1\}\} \]
2. All possible pairs of partitions are temporarily merged:
   \[
   \{\{0, 1\}\}, \{\{2\}\}, \ldots, \{\{l-2\}\}, \{\{l-1\}\} \]
   \[
   \{\{0, 2\}\}, \{\{1\}\}, \ldots, \{\{l-2\}\}, \{\{l-1\}\} 
   \]
   \[
   \vdots
   \]
   \[
   \{\{0\}\}, \{\{1, 2\}\}, \ldots, \{\{l-2\}\}, \{\{l-1\}\} 
   \]
   \[
   \vdots
   \]
   \[
   \{\{0\}\}, \{\{1\}\}, \{\{2\}\}, \ldots, \{\{l-2, l-1\}\} 
   \]
3. Compute MDL score of each factorization.
4. Choose the best scoring factorization if better than current.
5. Repeat until no better scoring factorization is found.

Bayesian Network

- Probability vector, dependency tree, and marginal product model are limited probability models.
- Bayesian network much more powerful model.
  - Acyclic directed graph.
  - Nodes are problem variables.
  - Edges represent conditional dependencies.
Multivariate PMBGA

Bayesian network learning

- Similar to ECGA: scoring metric + greedy search
- **Scoring metric**: MDL or Bayesian measure
- **Greedy search**:
  - Initially, no variables are connected.
  - Greedily either add, remove, or reverse an edge between two variables.
  - Until local optimum is reached.

Bayesian Network PMBGAs variants

- Bayesian Optimization Algorithm (BOA) (Pelikan, Goldberg, Cantú-Paz; 1998)
- Estimation of Distribution Networks Algorithm (EBNA) (Etxeberria, Larrañaga; 1999)
- Learning Factorized Distribution Algorithm (LFDA) (Mühlenbein, Mahnig, Rodriguez; 1999)

- **Similarities**: All use Bayesian Network as probability model.
- **Dissimilarities**: All use different method to learn BN.

Hierarchical BOA

- hBOA (Pelikan, Goldberg; 2001)
- **Decomposition** on multiple levels.
  - Bayesian network learning by BOA
- **Compact** representation.
  - Local Structures to represent conditional probabilities.
- **Preservation** of alternative solutions.
  - Niching with Restricted Tournament Replacement

Markov Network

- Probability model is **undirected graph**.
- **Factorise** the joint probability distribution in cliques of the undirected graph and sample it.
- **Most recent version**: Markovian Optimisation Algorithm (MOA) (Shakya & Santana, 2008).
- MOA does not explicitly factorise the distribution but uses the **local Markov property** and **Gibbs sampling** to generate new solutions.
Family Of Subsets (FOS) model

FOS $\mathcal{F}$
- PMBGAs learn a probabilistic model of good solutions to match the structure of the optimization problem.
- Key idea is to identify groups of problem variables that together make an important contribution to the quality of solutions.
- Dependency structure generally called a Family Of Subsets (FOS).
- Let there be $\ell$ problem variables $x_0, x_1, \ldots, x_{\ell-1}$.
- Let $S$ be a set of all variable indices $\{0, 1, \ldots, \ell - 1\}$.
- A FOS $\mathcal{F}$ is a set of subsets of the set $S$.
- FOS $\mathcal{F}$ is a subset of the powerset of $S$ ($\mathcal{F} \subseteq \mathcal{P}(S)$).

The Univariate Structure

- The univariate FOS is defined by:
  
  $F^i = \{i\}, \quad i \in \{0, 1, \ldots, \ell - 1\}$

- For $\ell = 10$ the univariate FOS is:
  
  $\mathcal{F} = \{\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{9\}\}$

- Every variable is modeled to be independent of other variables.

The Marginal Product Structure

- The marginal product (MP) FOS is a FOS such that:
  
  $F^i \cap F^j = \emptyset, \quad i, j \in \{0, 1, \ldots, \ell - 1\}$

- Univariate FOS is a MP FOS.
- For $\ell = 10$ a possible MP FOS is:
  
  $\mathcal{F} = \{\{0, 1, 2\}, \{3\}, \{4, 5\}, \{6, 7, 8, 9\}\}$

- Every group of variables is modeled to be independent of other variables.

FOS can be written more specifically as:

$\mathcal{F} = \{F^0, F^1, \ldots, F^{\mathcal{F} - 1}\}$
where

$F^i \subseteq \{0, 1, \ldots, \ell - 1\}, \quad i \in \{0, 1, \ldots, |\mathcal{F}| - 1\}$

- Every variable is in at least one subset in the FOS, i.e.:
  
  $\forall i \in \{0, 1, \ldots, \ell - 1\} : (\exists j \in \{0, 1, \ldots, |\mathcal{F}| - 1\} : i \in F^j)$
The Linkage Tree Structure

- The linkage tree (LT) FOS is a hierarchical structure.
- Group of all variables is in there.
- For any subset \( F_i \) with more than one variable, there are subsets \( F_j \) and \( F_k \) such that:
  \[ F_j \cap F_k = \emptyset, \quad |F_j| < |F_i|, \quad |F_k| < |F_i| \quad \text{and} \quad F_j \cup F_k = F_i \]
- For \( l = 10 \) a possible LT FOS is
  \[ F = \{ \{ 7, 5, 8, 6, 9, 0, 3, 2, 4, 1 \}, \{ 7, 5, 8, 6, 9 \}, \{ 0, 3, 2, 4, 1 \}, \{ 7 \}, \{ 5, 8, 6, 9 \}, \{ 0, 3, 2, 4 \}, \{ 2, 4 \}, \{ 0 \}, \{ 1 \}, \{ 3 \}, \{ 4 \}, \{ 5 \}, \{ 6 \}, \{ 7 \}, \{ 8 \}, \{ 9 \} \} \]
- Variables sometimes independent, sometimes dependent.
- \( \approx \) Path through dependency space, from univariate to joint.

Linkage Tree Learning

- Start from univariate structure.
- Build linkage tree using bottom-up hierarchical clustering algorithm.
- Similarity measure:
  1. Between individual variables \( X \) and \( Y \): mutual information \( I(X, Y) \).
  2. Between cluster groups \( X_{F_i} \) and \( X_{F_j} \): average pairwise linkage clustering (= unweighted pair group method with a arithmetic mean: UPGMA).
  \[ I_{UPGMA}(X_{F_i}, X_{F_j}) = \frac{1}{|X_{F_i}| |X_{F_j}|} \sum_{X \in X_{F_i}} \sum_{Y \in X_{F_j}} I(X, Y). \]

Linkage Tree

- Linkage Tree structure: subsets of FOS \( F \) form a hierarchical clustering.
- \( F = \{ \{ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \}, \{ 0, 1, 2, 3, 4, 5 \}, \{ 6, 7, 8, 9 \}, \{ 0, 1 \}, \{ 2, 3 \}, \{ 4 \}, \{ 5 \}, \{ 6 \}, \{ 7 \}, \{ 8 \}, \{ 9 \} \} \)
- Each subset (of length > 1) is split in two mutually exclusive subsets.
- Problem variables in subset are considered to be dependent on each other but become independent in a child subset.
- For a problem of length \( \ell \) the linkage tree has \( \ell \) leaf nodes (the clusters having a single problem variable) and \( \ell - 1 \) internal nodes.

Linkage Tree Learning

- This agglomerative hierarchical clustering algorithm is computationally efficient.
- Only the mutual information between pairs of variables needs to be computed once, which is a \( O(\ell^2) \) operation.
- The bottom-up hierarchical clustering can also be done in \( O(\ell^2) \) computation by using the reciprocal nearest neighbor chain algorithm.
Optimal Mixing Evolutionary Algorithms (OMEA)

- OMEA is a Model-Building EA that uses a FOS as its linkage model (Thierens & Bosman, 2011).
- Characteristic of Optimal Mixing Evolutionary Algorithm (OMEA) is the use of intermediate function evaluations (inside variation).
- Can be regarded as greedy improvement of existing solutions.
- Coined “Optimal” Mixing because better instances for substructures are immediately accepted and not dependent on “noise” coming from other parts of the solution.
- Recombinative OM (ROM) and Gene-pool OM (GOM)
  - ROM is GA-like: select single solution to perform OM with.
  - GOM is EDA-like: select new solution for each substructure in OM.

Gene-pool Optimal Mixing EA

GOMEA()

Pop ← InitPopulation()
while NotTerminated(Pop)
  FOS ← BuildFOS(Pop)
  forall Sol ∈ Pop
    forall SubSet ∈ FOS
      Donor ← Random(Pop)
      Sol ← GreedyRecomb(Sol,Donor,SubSet,Pop)
  return Sol

GreedyRecomb(Sol,Donor,SubSet,Pop)

NewSol ← ReplaceSubSetValues(Sol,SubSet,Donor)
if ImprovementOrEqual(NewSol,Sol) then Sol ← NewSol
return Sol

Recombinative Optimal Mixing EA

ROMEA()

Pop ← InitPopulation()
while NotTerminated(Pop)
  FOS ← BuildFOS(Pop)
  forall Sol ∈ Pop
    Donor ← Random(Pop)
    forall SubSet ∈ FOS
      Sol ← GreedyRecomb(Sol,Donor,SubSet,Pop)
  return Sol

GreedyRecomb(Sol,Donor,SubSet,Pop)

NewSol ← ReplaceSubSetValues(Sol,SubSet,Donor)
if ImprovementOrEqual(NewSol,Sol) then Sol ← NewSol
return Sol

Optimal Mixing EA (GOMEA)

- FOS linkage models specify the linked variables.
- A subset of the FOS is used as crossover mask.
- Crossover is greedy: only improvements (or equal) are accepted.
- Each generation a new FOS model is build from selected solutions.
- For each solution in the population, all subsets of the FOS are tried with a donor solution randomly picked from the population.
- Recombinative OM (ROM) and Gene-pool OM (GOM)
  - ROMEA: each solution uses a single donor solution.
  - GOMEA: new donor selected for each FOS subset.
Linkage Tree Genetic Algorithm

- The LTGA is an instance of GOMEA that uses a Linkage Tree as FOS model (Thierens & Bosman, 2010, 2011).
- Each generation a new hierarchical cluster tree is build.
- For each solution in population, traverse tree starting at the top.
- Nodes (= clusters) in the linkage tree used as crossover masks.
- Select random donor solution, and its values at the crossover mask replace the variable values from the current solution.
- Evaluate new solution and accept if better/equal, otherwise reject.

Deceptive Trap Function

Interacting, non-overlapping, deceptive groups of variables.

\[
f_{DT}(x) = \sum_{i=0}^{l-k} f_{sub}^{DT}(x(i,...,i+k-1))
\]

Nearest-neighbor NK-landscape

- Overlapping, neighboring random subfunctions

\[
f_{NKSS}(x) = \sum_{i=0}^{l-k} f_{sub}^{NK}(x(i,...,i+k-1)) \text{ with } f_{sub}^{NK}(x(i,...,i+k-1)) \in [0..1]
\]

- eg. 16 subsfcts, length \( k = 5 \), overlap \( o = 4 \) ⇒ stringlength \( \ell = 20 \)

- Global optimum computed by dynamic programming
- Benchmark function: structural information is not known!
- ⇒ Randomly shuffled variable indices.

Experiments

# Function Evaluations / Problem size
Hierarchical Trap function

HTrap

- Combine deceptive trap functions at each level in tree.
- Balanced $k$–ary tree
- Internal nodes are 0 (resp. 1) if all their children are 0 (resp. 1).
- Global optimum is all ones, yet at each level search is biased towards zeroes.

Hierarchical Trap function

HTrap: LTGA and hBOA

- HTrap problems:
  - block length $k = 3$; problem lengths 27, 81, 243 & 729.
  - Number of evaluations & minimal population size.
Experiments: conclusion

- LTGA (= GOMEA with LT FOS) very efficient on Deceptive Trap function, Nearest-Neighbor NK landscape, and Hierarchical Trap function.
- Tree not always suitable linkage model: for instance spin-glasses LTGA vs. hBOA (Pelikan, Hauschild & Thierens, 2011).
- Other FOS models possible: Linkage Neighborhood OM (Bosman & Thierens, 2012).
- Linkage Tree seems to be good compromise between FOS model complexity and search efficiency.

Predetermined vs. Learned FOS

- Problem structure unknown: learn a FOS model.
- Problem structure Information available: predetermined FOS model.
- What is a good predetermined FOS model?
- Direct mapping of dependency structure of problem definition to a predetermined FOS model?
- Predetermined linkage models mirroring the static structure of the problem not sufficient (Thierens & Bosman, 2012).
- Dynamically learned tree model superior to mirror structured models and to static tree model.
- Question: is there an optimal, predetermined linkage model that outperforms the learned (tree) model?

Conclusions

- “Blind” Evolutionary Algorithms are limited in their capability to detect and mix/exploit/re-use partial solutions (building blocks).
- One requires luck or analyzing and designing ways of structure exploitation directly into problem representation and search operators.
- Having a configurable model can help overcome this.
- Algorithm then must learn to configure the model and thereby exploit structure online during optimization (e.g. EDAs, OMEAs).

Black-Box Optimization (BBO)

- Maximize $\mathcal{F}(x), x \in \mathbb{P}$
- No prior knowledge of $\mathcal{F}$
- Guess a new $x$ and evaluate it
- Can only use previously evaluated solutions
- Minimize number of evaluations and/or actual time
- Needed when not much known about a problem (e.g. simulations)
Black-Box Optimization (BBO)

- Assumption: problems are somehow structured
- Use induction to find structure
- Exploit structure for increased efficiency
- Preferable to enumeration or iterated random sampling

Model-based optimization

- What to induce?
- Use a model that defines reasonable structures
- Induce instance of the model
- Model capacity determines bias strength

Stochastic optimization

- Random initial populations
- Randomized (but potentially structured) variation operators
- Why optimize stochastically?
- More robust against
  - Noise
  - Unreliable gradients (e.g. numerically unstable)
  - Discontinuities
  - Local optima
  - ...

Stochastic model-based optimization

- Model: a parameterized (function) class
- Given observed solutions \( \{(x', \hat{y}(x'))\} \)
  - Induction: configure the model (construct an instance)
  - Variation: generate new solution(s) from model (stochastically)
  - Repeat
Stochastic model-based optimization

- Model = probability distribution
- Induction = learning/estimation
- Variation = sampling

- Estimation-of-Distribution Algorithm (EDA)

The Estimation-of-Distribution Algorithm (EDA)

- Use a set of $n$ solutions for distribution estimation
- Focus on better solutions by selection
- Estimate from selection
  - EDA: Mühlenbein and Paaß, 1996

<table>
<thead>
<tr>
<th>EDA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Initialize $P$ with $n$ random solutions</td>
</tr>
<tr>
<td>2 Repeat until termination criterion met</td>
</tr>
<tr>
<td>2.1 Select subset $S$ from $P$</td>
</tr>
<tr>
<td>2.2 Estimate distribution from $S$</td>
</tr>
<tr>
<td>2.3 Draw new set of solutions $O$ from distribution</td>
</tr>
<tr>
<td>2.4 Update $P$ with $O$</td>
</tr>
</tbody>
</table>

Stochastic model-based optimization

- Model = description of linkages/dependencies
- Induction = learning/statistical testing
- Variation = mixing

- Optimal Mixing Evolutionary Algorithm (OMEA)

The Estimation-of-Distribution Algorithm (EDA)

- Use a set of $n$ solutions for linkage detection
- Focus on better solutions by selection within variation
- Estimate from selection
  - OMEA: Thierens and Bosman, 2011

<table>
<thead>
<tr>
<th>OMEA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Initialize $P$ with $n$ random solutions</td>
</tr>
<tr>
<td>2 Repeat until termination criterion met</td>
</tr>
<tr>
<td>2.1 Select subset $S$ from $P$</td>
</tr>
<tr>
<td>2.2 Learn linkage model from $S$</td>
</tr>
<tr>
<td>2.3 Apply linkage-model guided optimal mixing to every individual in $P$ to generate $O$</td>
</tr>
<tr>
<td>2.4 Replace $P$ by $O$</td>
</tr>
</tbody>
</table>
Real-valued Model-Based Evolutionary Algorithms

- Essentially similar questions to case of binary/integer variables
- We don’t have the optimal model...
- Approximate the optimal model
- Match inductive search bias and problem structure
- How to learn and perform variation efficiently and effectively
- Trade-offs:
  - Quality versus complexity of approximation
  - Efficiency in # evaluations versus time
- Essential model questions:
  - Can key problem structure be represented?
  - Can key problem structure be represented efficiently?
  - Can the model be learned from data?
  - Can the model be learned (and used for variation) efficiently?

Normal distribution

- Require practically useful models.
- For instance normal distribution:
  \[ P(x) \]
  
  - Only \( O(l_2) \) parameters (mean, covariance matrix)
  - maximum-likelihood (ML) estimates well known
  \[ \hat{\mu} = \frac{1}{|S|} \sum_{j=0}^{S-1} (S_j), \quad \hat{\Sigma} = \frac{1}{|S|} \sum_{j=0}^{S-1} ((S_j) - \hat{\mu})(S_j) - \hat{\mu}^T \]
- Can only model linear dependencies

EDAs based on the Normal Distribution

- First uses were adaptations of PBIL
  - Rudlof and Köppen, 1996
  - Sebag and Ducoulombier, 1998
- Although initial results were interesting, quickly found that some problems were solved more efficiently if dependencies were modeled

EDAs based on the Normal Distribution

- Make decisions based on better fit and increased complexity (e.g. \( \hat{P}(X_0, X_1) \) vs. \( \hat{P}(X_0)\hat{P}(X_1) \) )
**EDAs based on the Normal Distribution**

- EDAs with factorized Normal Distributions (MIMIC, COMIT, Bayesian, Copula selection, Multivariate (Markov networks))
  - Bosman and Thierens, 2000, 2001
  - Larrañaga, Etcheberria, Lozano and Peña, 2000
  - Karshenas, Santana, Bielza and Larrañaga (2012)
- On selected problems, improvements were found when using higher-order dependencies
- On some problems, results didn’t get much better however
- Initially mainly attributed to mismatch between model and search space
- Clearly true to some extent

**EDAs based on the Normal–kernels distribution**

- Bosman and Thierens, 2000
- Ocenasek and Schwarz, 2002
- Ocenasek, Kern, Hansen, Müller and Koumoutsakos, 2004
- Natural tendency to fit structure of data (linear or not)
- But also tendency to overfit
- Maximum-likelihood estimate not usable
- Quality of estimation depends heavily on size of kernel

**EDAs based on the Normal–mixture distribution**

- Gallagher, Fream and Downs, 1999
- Bosman and Thierens, 2001
- Ahn, Ramakrishna and Goldberg, 2004
- Trade–off between normal and normal kernels.
- Requires a lot of effort to estimate with maximum likelihood (EM algorithm).
- Clustering, followed by normal–distribution estimate can be used alternatively.

**EDAs based on the Histogram Distribution**

- Bosman and Thierens, 2000
- Tsutsui, Pelikan and Goldberg, 2001
- Easy to implement and map to integers.
- Require many bins to get a good estimate.
- Curse of dimensionality.
- Greedy incr. factorization selection hardly possible.
EDAs based on the Normal–mixture Distribution Revisited

- Cluster first, then estimate (factorized) normal distribution in each cluster
  - Bosman and Thierens, 2001
  - Cho and Zhang, 2002

- "Reverse" also possible (more focus on separability)
- Factorize, then estimate mixture distr. per set of variables
- Still need to way to factorize however (select pdf to base on)
  - Li, Goldberg, Sastry and Yu (2007)

EDAs based on latent variable models

- Build models by projecting data onto model of lower dimensionality
- Helmholtz machines, mixture of factor analyzers, etc
  - Shin and Zhang, 2001
  - Cho and Zhang, 2001
  - Shin, Cho and Zhang, 2001
  - Cho and Zhang, 2002
  - Cho and Zhang, 2004

- Better results than standard normal EDA on some problems, but still unable to come close to the optimum of 10-dimensional Rosenbrock function

Direct use of normal distribution

- Bad results
  - Rosenbrock:
    $$f(x) = \sum_{i=0}^{n-2} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$$

- because...
  - Rosenbrock has narrow valley leading to minimum
  - Quickly samples no longer centered around minimum

No attention for the gradient

- Distribution estimation makes no assumption on source
- Source is just selected points in parameter space
- Gradient info is ignored in maximum-likelihood estimate
- For normal distribution:
  Variance goes to zero too fast
Illustration on the 1-D sphere function
\[ f(x) = x^2 \]
Progression in first 6 generations (top-left to bottom-right)

Analysis of the premature-convergence problem
- Theoretical analysis reveals indeed limits
  - Gonzalez, Lozano and Larrañaga, 2000
  - Grahl, Minner and Rothlauf, 2005
  - Bosman and Grahl, 2005
  - Yuan and Gallagher, 2006
- There is for instance a bound on how far the mean can shift

Analysis of the premature-convergence problem
- Variance decreases (exponentially fast)
  \[ \lim_{t \to \infty} \{ \hat{\sigma}(t) \} = \lim_{t \to \infty} \{ \hat{\sigma}(0) c(t)^t \} = 0 \]
- This limits mean shift to a fixed factor times initial spread!
  \[ \lim_{t \to \infty} \{ \hat{\mu}(t) \} = \hat{\mu}(0) + \frac{d(\tau)}{1 - \sqrt{c(\tau)}} \hat{\sigma}(0) \]
- \( c(\tau) \) and \( d(\tau) \) functions of
  - \( \phi() \) (standard normal distribution) and
  - \( \Phi() \) (inverse cumulative normal distribution)

Illustration on the 2-D plane function
\[ f(x) = x_0 + x_1 \]
Progression in first 6 generations
What is missing?

- Structure of landscape can be very complicated
- “Simple” normal distr. hardly matches global structure
- More involved distributions possible, but
  - harder, or even impossible, to estimate with ML
  - requires lots of data
- Local structure can be approximated but...
  - there is no generalization outside of the data range
  - Once optimum “lost” outside data range, EDA converges elsewhere, possibly not even a local optimum!
- EDA based on maximum-likelihood estimate not efficient

Ways to improve

- Gradient hybridization
  - Explicit use of gradient information
  - Apply gradient-based search to certain solutions (e.g. conjugate gradients)
  - Requires gradient computation
    - not always possible
    - not always reliable
- Adapt(ive) (ML) estimation
  - Derivative Free
  - Maintain EDA properties for valley case
  - Adapt in other cases (to explore beyond selected solutions)
  - How to distinguish?
  - Three ingredients:
    - Adaptive Variance Scaling (AVS)
    - Standard-Deviation Ratio (SDR)
    - Anticipated Mean Shift (AMS)

Adapted Maximum-Likelihood Gaussian Model

- Adaptive Variance Scaling (AVS) & Standard-Deviation Ratio (SDR)
- If improvements are found
  - a) far from the mean, enlarge $\Sigma$
  - b) close to the mean, do nothing
- Close to the mean: within one standard deviation

Adapted Maximum-Likelihood Gaussian Model

- Anticipated Mean Shift (AMS)
- Anticipate where the mean is shifting
- Alter part of generated solutions by shifting
- On a slope, predictions are better (further down slope)
- Require balanced selection to re-align covariance matrix
Illustration on a 2-D slope
\[ f(x) = x_0 + x_1 \]

Progression in first 6 generations

AMaLGaM, CMA-ES and NES

- AMaLGaM IDEA (or AMaLGaM for short)
  Adapted Maximum-Likelihood Gaussian Model Iterated Density-Estimation Evolutionary Algorithm
- Natural question: what is the relation to CMA-ES (Hansen, 2001) and NES (Wierstra, Schaul, Peters and Schmidhuber, 2008)?
  Answer: the probability distribution
  All can be seen to be EDAs: every generation they estimate/update a probability distribution (which also happens to be the normal distribution in all three cases) and perform variation by generating new samples from this distribution.
  Differences are only in how the distribution is obtained. Where AMaLGaM uses maximum-likelihood estimates from the current generation, CMA-ES and NES base estimates on differences between subsequent generations as well as many elaborate enhancements (see tutorial on CMA-ES).

AMaLGaM, CMA-ES and NES

- On typical unimodal benchmark problems (sphere, (rotated) ellipsoid, cigar, etc) these algorithms exhibit polynomial scalability in both minimally required population size and required number of function evaluations
- CMA-ES and NES scale better than AMaLGaM on such problems

Parameter-free Gaussian EDAs

- Parameters get in the way of ease-of-use
- Remove all parameters: derive and implement guidelines
- Restart mechanism to increase success probability
- Typical restart scheme: increase size exponentially
- Works well on Griewank (left), not so much on Michalewicz (right)
- Many different schemes exist therefore (also algorithm specific, e.g. BIPOP-CMA-ES and IPOP-CMA-ES)
Noiseless BBOB comparison with other algorithms

- Binary/Integer representations are discrete, but also Cartesian
- Other discrete search spaces exist that are non-Cartesian
- Most notably: permutation-based problems
- Important real-world relevance, e.g. routing and scheduling
- Brings different challenges than Cartesian spaces however
  - Relative ordering problems
  - Absolute ordering problems
  - Neighbor ordering problems
  - Combinations of these
- Different types of models are more suited for specific types of ordering problem
Permutation Model-Based Evolutionary Algorithms

- Building permutation models directly not straightforward
- Potential aid in the form of random keys (Bean, 1997)
- Random keys encode permutations in real-valued space (via sorting)

\[
\begin{array}{ccc}
0 & 1 & 2 & 3 \\
0.61 & 0.51 & 0.62 & 0.31
\end{array}
\Rightarrow
\begin{array}{ccc}
3 & 1 & 0 & 2 \\
0.31 & 0.51 & 0.61 & 0.62
\end{array}
\]

- Real-valued approaches can thus be used directly
  - Bosman and Thierens (2001) (normal EDA)
- Inefficient scale-up behavior on deceptive additively decomposable relative ordering problems
- Highly redundant encoding that is hard to model with a normal distribution

Permutation Model-Based Evolutionary Algorithms

- Use crossover on the basis of a factorization of the normal distribution instead
  - Bosman and Thierens, 2001
- Now obtain polynomial scale-up behavior
- How about a direct modelling of probabilities of permutations?
- Consider a marginal product factorization (i.e. mutually exclusive subsets of variables as in ECGA)
- Once an instance is sampled for a subset of variables, other variables can’t use these values anymore
- One way to deal with this is explicit repair of probability tables during sampling
  - Pelikan et al (2007)
- Requires very large sample sizes
- Sampling repair can introduce unwanted biases

Permutation Model-Based Evolutionary Algorithms

- For relative-ordering variables, a probabilistically correct factorization approach is possible
  - Bosman, 2003
- Continuous, Binary: \( P(X) = P(X_0, X_4)P(X_1)P(X_3, X_2) \)
- Permutation: \( P(X) = \frac{2!1!2!}{5!} P(X_0, X_4)P(X_1)P(X_3, X_2) \)
- Random variable \( X_i \): position of integer \( i \) in the permutation → tackle relative-ordering permutation problems.
- Normalization required, because there are 5! permutations.
- “Oddities” specific to permutations exist (spurious dependencies between “low” variables in one building block and “high” variables in another)
- Require specialized adaptations of standard linkage learning / factorization techniques

Permutation Model-Based Evolutionary Algorithms

- Generate instance for each subset of variables independently
- Then map to the real-valued domain using random keys and then translate the entire string into a valid permutation
- Preserves relative ordering of variables in subsets
- Can sample directly instead of using crossover (crossover still more robust however)
- Scales polynomially and much better than normal-pdf induced crossover
Permutation Model-Based Evolutionary Algorithms

- **Edge-histogram** based sampling
  - Tsutsui, Pelikan and Goldberg, 2003
  - Maps well to problems with neighboring variable relations
- Model is a **matrix** with probabilities of edges
- Matrix needs to be **adjusted while sampling**
- For problems with neighboring relations works **better** than random keys

- **Gaussian** “equivalent” in permutation space: **Mallows** model
  - Ceberio, Mendiburu and Lozano (2011)
  - Requires a **distance measure** between permutations and a central permutation
  - Also requires a **spread parameter** (not estimated from data)
  - Most commonly used **distance**: Kendall-τ, allows factorization
  - Finding central permutation is **NP-hard** however
  - Fast **heuristics** are possible (linear in $l$ and $n$
  - Final parameter estimation and sampling are not trivial and require **dedicated** algorithms
  - First results are **promising** (permutation flow shop), outperforming Tsutsui

Tree (GP) Model-Based Evolutionary Algorithms

- Not tree-models for dependencies, but tree-models for tree-based solutions
- Estimation-of-Distribution Programming (EDP)
- Typically **grammar** based, but not always
- Grammar Guided Genetic Programming (GGGP)
- Grammars very **useful** to limit search space
- But how do we use it **learn** structural features?
Tree (GP) Model-Based Evolutionary Algorithms

- If looking at solutions node-based, and using a fixed template, essentially have Cartesian fixed-length representation
- Can use existing integer-based model-based EAs on this
  - eCGP (ECGA for GP) does exactly this
    - Sastry and Goldberg, 2003
- Better results for selected problems, but use of a template has it limitations

Tree (GP) Model-Based Evolutionary Algorithms

- Extensions to Bayesian factorizations are also possible
- Incremental tree complexity (and model complexity) using special operators
  - Looks (2006)

Tree (GP) Model-Based Evolutionary Algorithms

- Alternative approach: grammar-based
- Start with basic production rules
- Learning: assign probabilities to rules and increase complexity and specificity of rules using heuristics
- Sampling: select probabilistically from appropriate production rules
- Results are promising in that less function evaluations are often needed than standard GP, but time-complexity is (much) larger
  - Shan, McKay, Baxter, Abbass and Essam, 2003
  - Bosman and de Jong, 2004
  - Shan, McKay, Baxter, Abbass, Essam and Hoai, 2004
  - Hasegawa and Iba, 2007

Tree (GP) Model-Based Evolutionary Algorithms

- Intermediate approach: n-grams
- Focus probabilities on most important relationships (local, e.g. with parents and grandparents)
- Enumerate all possible relationships beforehand
- Learning: estimate probabilities for the n-grams
- Sampling: recursively employ the n-grams
- Advantage: learning is much faster than with grammar transformations
  - Hemberg, Veeramachaneni, McDermott, Berzan and O’Reilly (2012)
Multi-objective Model-Based Evolutionary Algorithms

- **Multiple objectives** should be optimized simultaneously
- **Conflicting objectives**, no expression of weights
- Can’t combine the objectives in a single **scalar objective**
- Want to present a set of **promising alternatives** to a decision maker
- **Example**: Maximize the **quality** and minimize the **production costs** of a product
- **NOTE**: This is NOT an MO tutorial

### Example

Maximize the quality and minimize the production costs of a product

### Algorithm

Algorithm attempts to obtain **improvements** all along the current Pareto front
- Different **regions** along Pareto front may be very different
- E.g. what are far ends of the **optimal** Pareto front? Optimal solutions for individual objectives \( f_i \)
- **Restrict** variation to clusters (restricted mating)
- For instance: obtain clusters **along** Pareto front: cluster **selected solutions**
  - Pelikan, Sastry and Goldberg, (2009)

In EDAs, this clustering corresponds to use of **mixture** probability distributions

\[
P(\varsigma, \theta)(Z) = \sum_{i=0}^{k-1} \beta_i P(\varsigma_i, \theta_i)(Z)
\]

- **Cluster** solutions in **objective** space (e.g. k-means)
- Estimate a **simpler** distribution \( P(\varsigma_i, \theta_i)(Z) \) in each cluster
- Set all **mixing coefficients** to \( \beta_i = \frac{1}{k} \)
- **Parallel**, specialized exploration **along** front
Multi-objective Model-Based Evolutionary Algorithms

- Each distribution explores own region
- Learning may however by incremental (CMA-ES, iAMaLGaM, iBOA, etc)
- Assign each distribution own adaptive incremental mechanisms
- Cannot combine directly with clustering each generation
- Need correspondence over generations
- Number of clusters fixed beforehand ($k$)

Multi-objective Model-Based Evolutionary Algorithms

- Implicit cluster registration
- Keep clusters spatially separated during run.
- Assign new solution to its nearest, non-full cluster
- Can over time lead to inefficient cluster movement

Multi-objective Model-Based Evolutionary Algorithms

- Explicit cluster registration
- Minimize sum of cluster distance over all permutations of clusters in subsequent generations
  - Bosman, 2010

Conclusions

- “Blind” metaheuristics are limited in their capability to detect and mix/exploit/re-use structural features of an optimization problem (e.g. partial solutions, building blocks, promising search directions, etc).
- One requires luck or analyzing and designing ways of structure exploitation directly into problem representation and search operators.
- Having a configurable model can help “overcome” this / help to do this automatically.
- Algorithm then must learn to configure the model and thereby exploit structure online during optimization.
- Having an explicitly tunable model can really help
Conclusions

▷ We don’t have the optimal model...
▷ Approximate the optimal model
▷ Match inductive search bias and problem structure
▷ How to learn and perform variation efficiently and effectively
▷ Trade-offs:
  ▷ Quality versus complexity of approximation
  ▷ Efficiency in # evaluations versus time
▷ Essential model questions:
  ▷ Can key problem structure be represented?
  ▷ Can key problem structure be represented efficiently?
  ▷ Can the model be learned from data?
  ▷ Can the model be learned (and used for variation) efficiently?

Efficient model-based evolutionary algorithms (EDAs/IDEAs/PMBGAs/OMEAs) exist
▷ Binary/Integer/Permutation/Real-valued/GP & multi-objective
▷ Research is ongoing
▷ Especially useful when optimizing from a black-box perspective (e.g. complex simulations)
▷ Also useful from a white-box perspective
  ▷ Can learn more about the problem through learnt models
  ▷ Models configurable by hand (remove “expensive” learning overhead)

Books

Acknowledgements

▷ Selected images were re-used from the 2012 GECCO tutorial “Probabilistic Model-building Genetic Algorithms” by Martin Pelikan.