GECCO 2021 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

GECCO'21 Companion, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8351-6/21/07. . . $15.00
https://doi.org/10.1145/3449726.3461417

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)

What ?

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

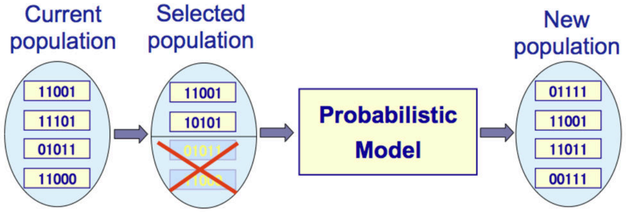
- Little known about the structure of the problem
- Clean separation optimizer from problem definition
- Easy and generally applicable

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

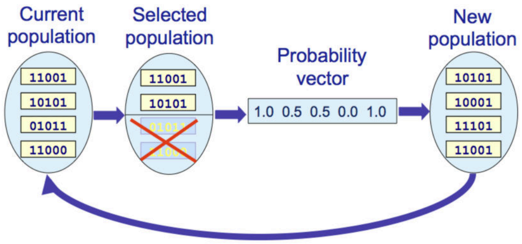
Type of Models

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

Univariate PMBGA

Model

- * Model: probability vector $[p_1, \dots, p_\ell]$ (ℓ: 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

Data
000000
111111
010101
101010
000010
111000
010111
111000
000111
111111

Marginal Product (MP) FOS		
	$\hat{P}(X_0X_1X_2)$	$\hat{P}(X_3X_4X_5)$
000	0.3	0.3
001	0.0	0.0
010	0.2	0.2
011	0.0	0.0
100	0.0	0.0
101	0.1	0.1
110	0.0	0.0
111	0.4	0.4

Univariate FOS						
	$\hat{P}(X_0)$	$\hat{P}(X_1)$	$\hat{P}(X_2)$	$\hat{P}(X_3)$	$\hat{P}(X_4)$	$\hat{P}(X_5)$
0	0.5	0.4	0.5	0.5	0.4	0.5
1	0.5	0.6	0.5	0.5	0.6	0.5

- What is the probability of generating 111111?
- Univariate FOS: $0.5 \cdot 0.6 \cdot 0.5 \cdot 0.5 \cdot 0.6 \cdot 0.5 = 0.0225$
- 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.
- $p(X) = \prod_{i=1}^{\ell-1} p(x_{i+1}|x_i)p(x_1)$.
- MIMIC greedily searches for the optimal permutation of variables that minimizes Kullack-Leibler divergence.

Bivariate PMBGA

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 forrest.
- Pairwise interactions measured by Pearson's chi-square statistics.

Bivariate PMBGA

DSMGA

- Dependency Structure Matrix Genetic Algorithm (Yu, Goldberg, Sastry, Lima, Pelikan; 2009)
- Dependency Structure Matrix (DSM) contains the information of pairwise interactions.
- DSMGA constructs the DSM by using mutual information metric.
- DSM clustering aims to transfer the pair-wise interaction information into higher-order interaction information.
- DSM Clustering Metric based on the minimum description length principle (MDL).

Bivariate PMBGA

DSMGA-II

- Extended version ⇒ DSMGA-II (Hsu, Yu; 2015).
- DSMGA-II consists of four major components:
  1. pair-wise linkage detection
  2. model building
  3. restricted mixing
  4. back mixing
- Clustering the DSM leads to the Incremental Linkage Set: starting from one gene, incrementally add the next most dependent gene one-by-one.
- Restricted mixing: focus on building-block supply.
- Back mixing: when no improvement occurs, switch to the equal-acceptance criterion to reduce unnecessary evaluations on plateaus.

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).
- $p(X) = \prod_{g=1}^{G} p(X_g)$
- 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)

- $MDL(M, D) = D_{Model} + D_{Data}$
- 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.

Multivariate PMBGA

Learning MP model

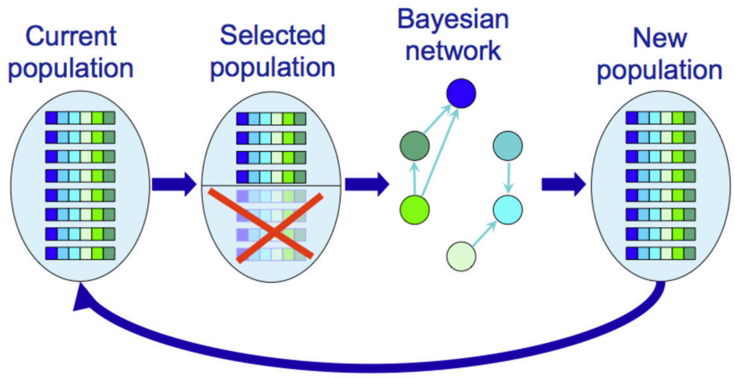
1. Start from univariate FOS:
$$\{\{0\}, \{1\}, \{2\}, \ldots, \{l-2\}, \{l-1\}\}$$
2. All possible pairs of partitions are temporarily merged:
$$\begin{gathered}\{\{0,1\}, \{2\}, \ldots, \{l-2\}, \{l-1\}\}\\ \{\{0,2\}, \{1\}, \ldots, \{l-2\}, \{l-1\}\}\\ \vdots\\ \{\{0\}, \{1,2\}, \ldots, \{l-2\}, \{l-1\}\}\\ \vdots\\ \{\{0\}, \{1\}, \{2\}, \ldots, \{l-2, l-1\}\}\end{gathered}$$
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.

Multivariate PMBGA

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



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.

Multivariate PMBGA

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

Multivariate PMBGA

Markov Network

- Markov Netwok EDA
  (MN-EDA: Santana, 2005) (DEUM: Shakya & McCall, 2007).
- 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 ℓ problem variables $x_0, x_1, \dots, x_{\ell-1}$.
- Let S be a set of all variable indices $\{0, 1, \dots, \ell-1\}$.
- A FOS $\mathcal{F}$ is a set of subsets of the set S.
- FOS $\mathcal{F}$ is a subset of the powerset of S ($\mathcal{F} \subseteq \mathcal{P}(S)$).

Family Of Subsets (FOS) model

- FOS can be written more specifically as:

$$\mathcal{F} = \{\mathbf{F}^0, \mathbf{F}^1, \dots, \mathbf{F}^{|\mathcal{F}|-1}\}$$

where

$$\mathbf{F}^i \subseteq \{0, 1, \dots, l-1\}, \quad i \in \{0, 1, \dots, |\mathcal{F}|-1\}$$

- Every variable is in at least one subset in the FOS, i.e.:
  $\forall i \in \{0, 1, \dots, l-1\} : \left(\exists j \in \{0, 1, \dots, |\mathcal{F}|-1\} : i \in \mathbf{F}^j\right)$

The Univariate Structure

- The univariate FOS is defined by:

$$\mathbf{F}^i = \{i\}, \quad i \in \{0, 1, \dots, l-1\}$$

- For $l = 10$ 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 varibables.

The Marginal Product Structure

- The marginal product (MP) FOS is a FOS such that:

$$\mathbf{F}^i \cap \mathbf{F}^j = \emptyset, \quad i, j \in \{0, 1, \dots, l-1\}.$$

- Univariate FOS is a MP FOS.
- For $l = 10$ 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.

The Linkage Tree Structure

- The linkage tree (LT) FOS is a hierarchical structure.
- Group of all variables is in there.
- For any subset $\mathbf{F}^i$ with more than one variable, there are subsets $\mathbf{F}^j$ and $\mathbf{F}^k$ such that:

$$\mathbf{F}^j \cap \mathbf{F}^k = \emptyset, \quad |\mathbf{F}^j| < |\mathbf{F}^i|, \quad |\mathbf{F}^k| < |\mathbf{F}^i| \ \text{ and } \ \mathbf{F}^j \cup \mathbf{F}^k = \mathbf{F}^i$$

- For $l = 10$ a possible LT FOS is

$$\mathcal{F} = \{\{7, 5, 8, 6, 9, 0, 3, 2, 4, 1\},$$
$$\{7, 5, 8, 6, 9\}, \{0, 3, 2, 4, 1\}, \{7\}, \{5, 8, 6, 9\},$$
$$\{0, 3, 2, 4\}, \{1\}, \{5, 8, 6\}, \{9\}, \{0, 3\}, \{2, 4\},$$
$$\{5, 8\}, \{6\}, \{0\}, \{3\}, \{2\}, \{4\}, \{5\}, \{8\}\}$$

- Variables sometimes independent, sometimes dependent.
- $\approx$ Path through dependency space, from univariate to joint.

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}, {7,8,9}, {0,1}, {4,5}, {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 ℓ the linkage tree has ℓ leaf nodes (the clusters having a single problem variable) and $\ell - 1$ internal nodes.

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 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 Algorithm (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

---

Gene-pool Optimal Mixing EA (GOMEA)

- FOS linkage models specify the linked variables.
- A subset of the FOS is used as mixing mask
- Mixing is greedy: accept only improvements (or equal).
- A new FOS model is built from the population every generation.
- FOS model may also be pre-specified, of course.
- Gene-pool Optimal Mixing Evolutionary Algorithm (GOMEA)
  - For each solution in the population
    - all subsets of the FOS are tried with a donor solution randomly picked from the population

Gene-pool Optimal Mixing EA

```
GOMEA()
    Pop ← InitPopulation()
    while NotTerminated(Pop)
       FOS ← BuildFOS(Pop)
       forall Sol ∈ Pop
          forall SubSet ∈ FOS
             Donor ← Random(Pop)
             Sol ← OptimalMixing(Sol,Donor,Subset,Pop)
    return Sol

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

Linkage Tree Genetic Algorithm

- The LTGA is an instance of GOMEA that uses a Linkage Tree as FOS model (Thierens & Bosman, 2010, 2011).
- AKA LT-GOMEA.
- Each generation a new hierarchical cluster tree is built.
- For each solution in population, traverse tree (random order).
- Nodes (= clusters) in the linkage tree form FOS.

Benchmark problems

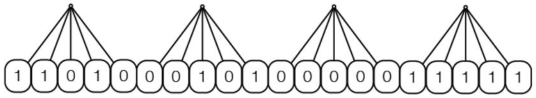
- Onemax (counting ones)

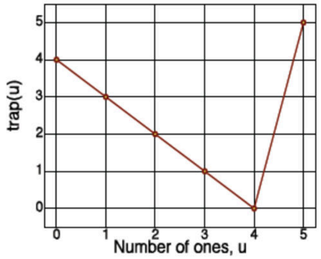
$$f_{\text{Onemax}}(\mathbf{x}) = \sum_{i=0}^{\ell-1} x_i$$

Deceptive Trap Function

Interacting, non-overlapping, deceptive groups of variables.

$$f_{\text{DT}}(x) = \sum_{i=0}^{l-k} f_{\text{DT}}^{\text{sub}}\left(x_{(i,\ldots,i+k-1)}\right)$$



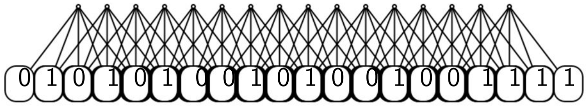


Nearest-neighbor NK-landscape

- Overlapping, neighboring random subfunctions

$$f_{\text{NK-S1}}(x) = \sum_{i=0}^{l-k} f_{\text{NK}}^{\text{sub}}\left(x_{(i,\ldots,i+k-1)}\right) \quad \text{with} \quad f_{\text{NK}}^{\text{sub}}\left(x_{(i,\ldots,i+k-1)}\right) \in [0..1]$$

- eg. 16 subsfcts, length $k = 5$, overlap $o = 4 \Rightarrow$ stringlength $\ell = 20$



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

Benchmark problems

- Hierarchical If-and-only-iFF (HIFF)
  (Watson, Hornby and Pollack, 1998)
- Computed over multiple layers
- Nodes are combined as a perfectly balanced binary tree
- Problem lengths are powers of two (i.e., $\ell = 2, 4, 8, 16, 32, \ldots$)
- Each variable is considered to be a leaf
- Leaf contributes 1
- Internal node contributes 2^{height} if children both 0 or both 1
- Internal node is 0 if children both 0; 1 if both 1; NIL else

---

Benchmark problems

- Weighted MAX-CUT
  (Karp, 1972)
- Given a weighted graph (V, E), divide nodes into two sets so that total weight of edges between sets is maximized
- Identify binary variable x_i with each node v_i

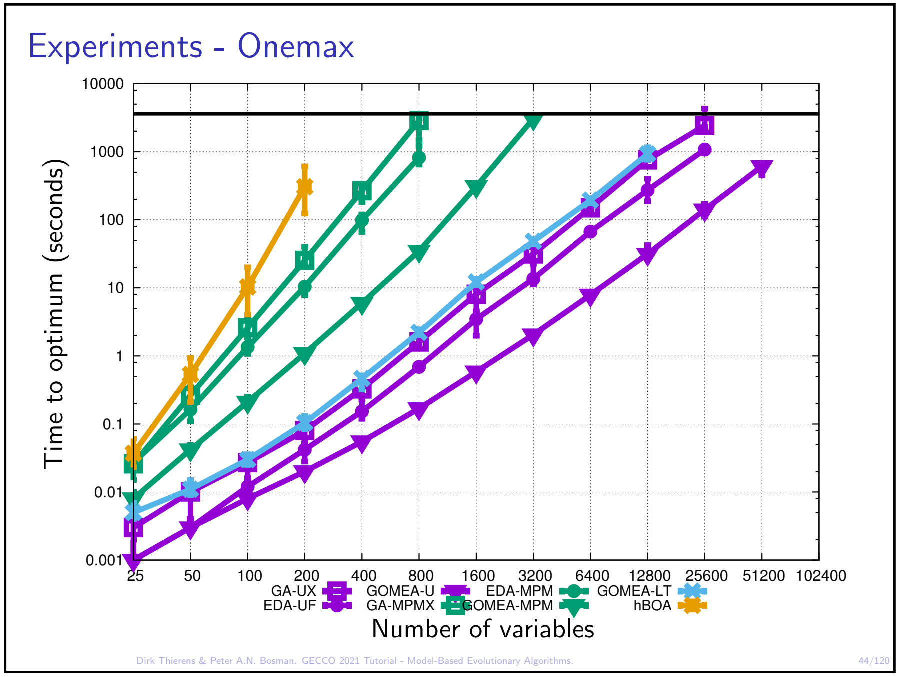
$$f_{\text{Weighted MAX-CUT}}(\mathbf{x}) = \sum_{(v_i,v_j)\in E} \begin{cases} w_{ij} & \text{if } x_i \neq x_j \\ 0 & \textit{otherwise} \end{cases}$$

- For now, considered as a black-box problem
  (no partial evaluations allowed)
- 5 instance types:
  - Fully connected graphs (β-distributed, $\alpha = 100$, $\beta = 1$)
  - 2D Square-grid graphs (β-distributed, $\alpha = 100$, $\beta = 1$)
  - 3D Square-torus graphs (β-distributed, $\alpha = 100$, $\beta = 1$)
  - Uniformly distributed in a box, fully connected
  - Uniformly distributed in a box, $\lfloor\sqrt{(\ell)}\rfloor$ nearest neighbors

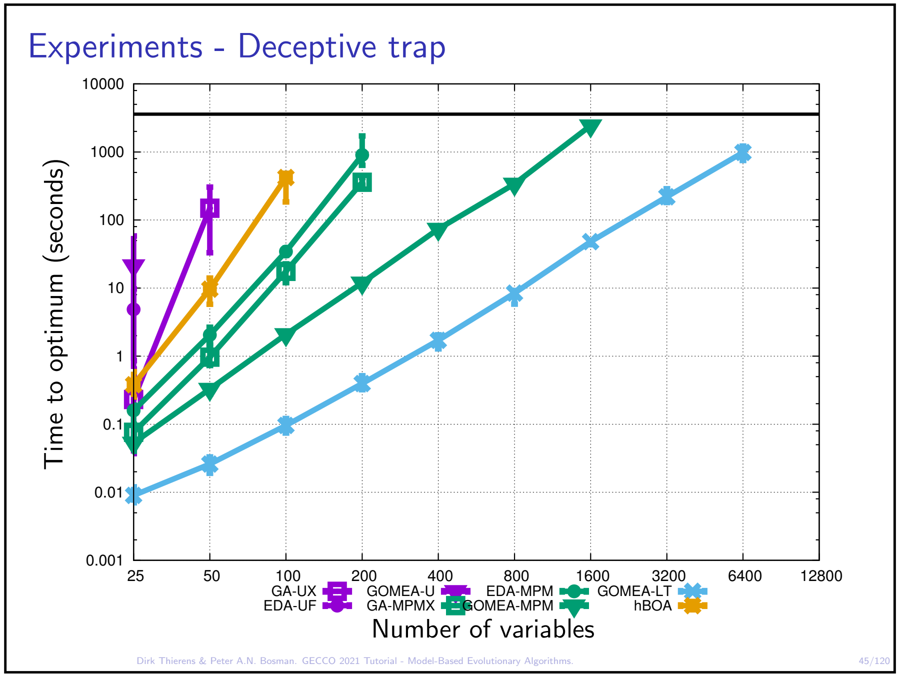
---

Experimental setup

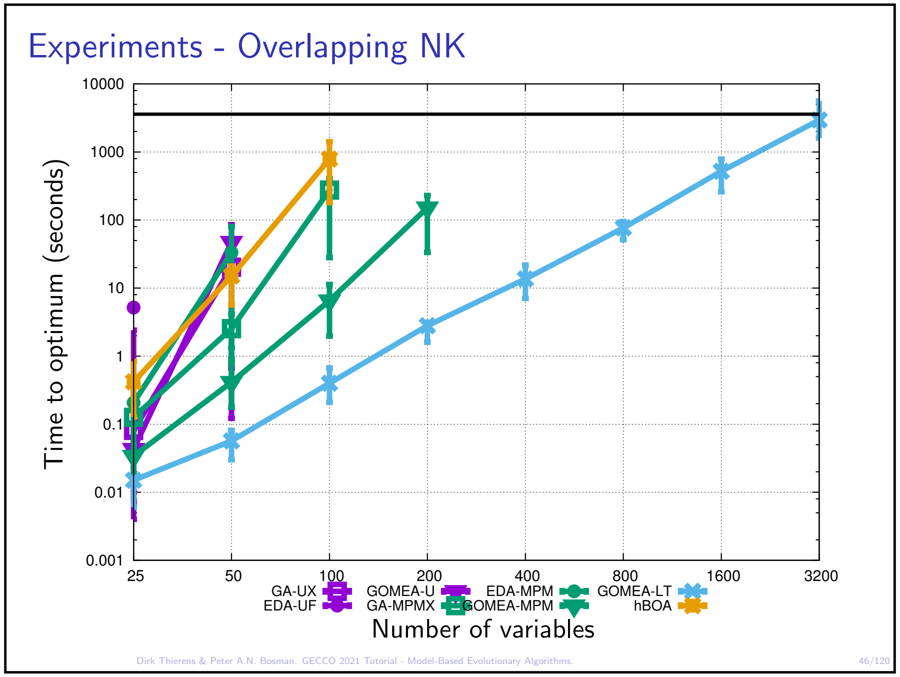
- 100 independent runs
- Fully black-box evaluated
- Observe time required to reach optimum
- On a relatively slow CPU: 2.8 GHz AMD core
- Harik-Lobo Population-sizing-free scheme
  - Setting population size optimally is hard/impossible
  - Use interleaving of different runs with different population sizes
  - Generations in larger population sizes tick slower
  - Smaller populations converge first
  - Overhead: typically somewhere between 1-4 times slower
  - Can actually also be faster!
  - Gains outweigh the overhead!



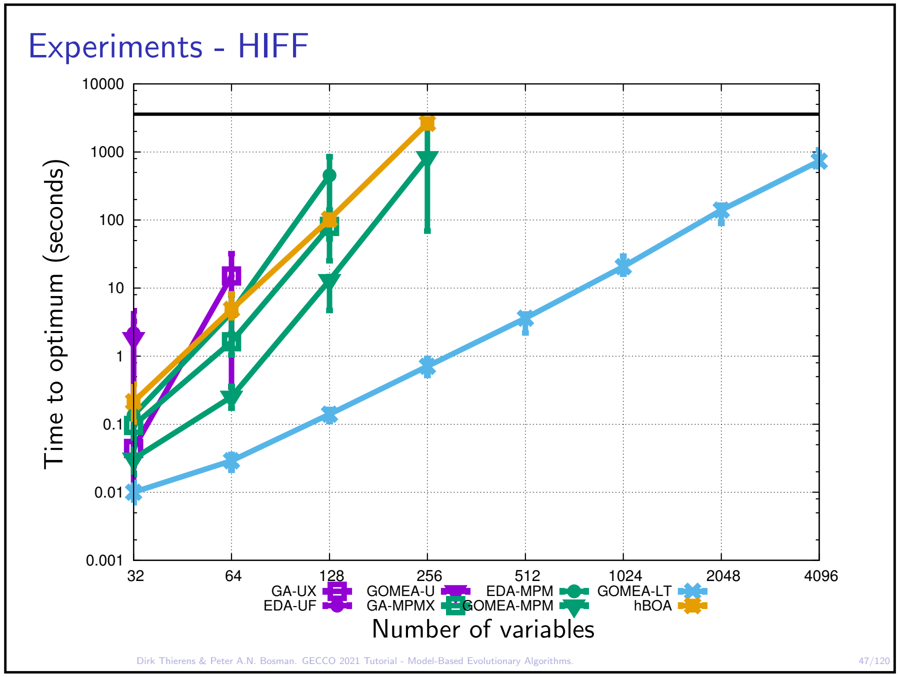
Experiments - Onemax



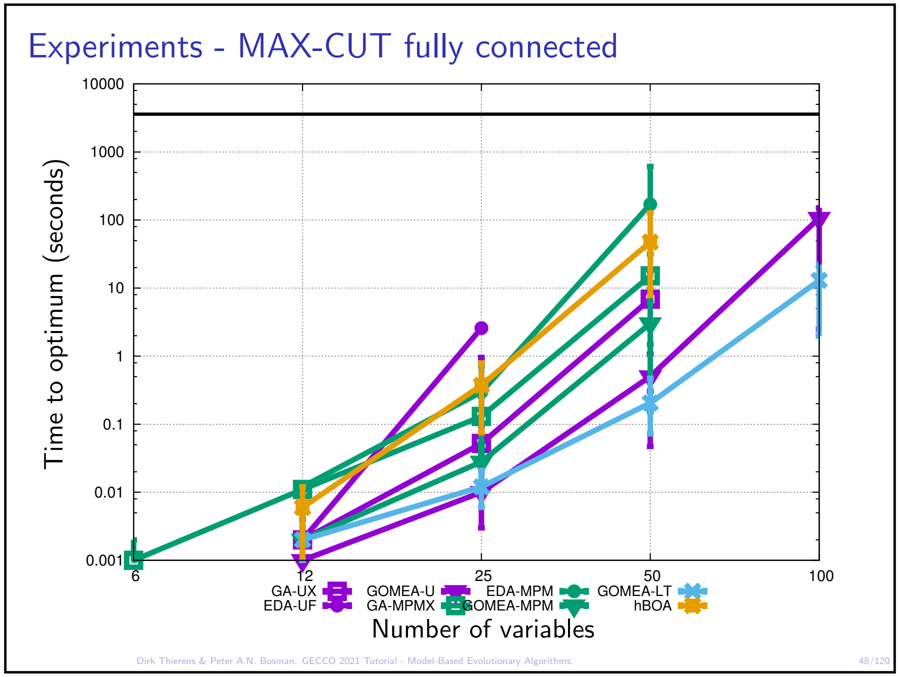
Experiments - Deceptive trap



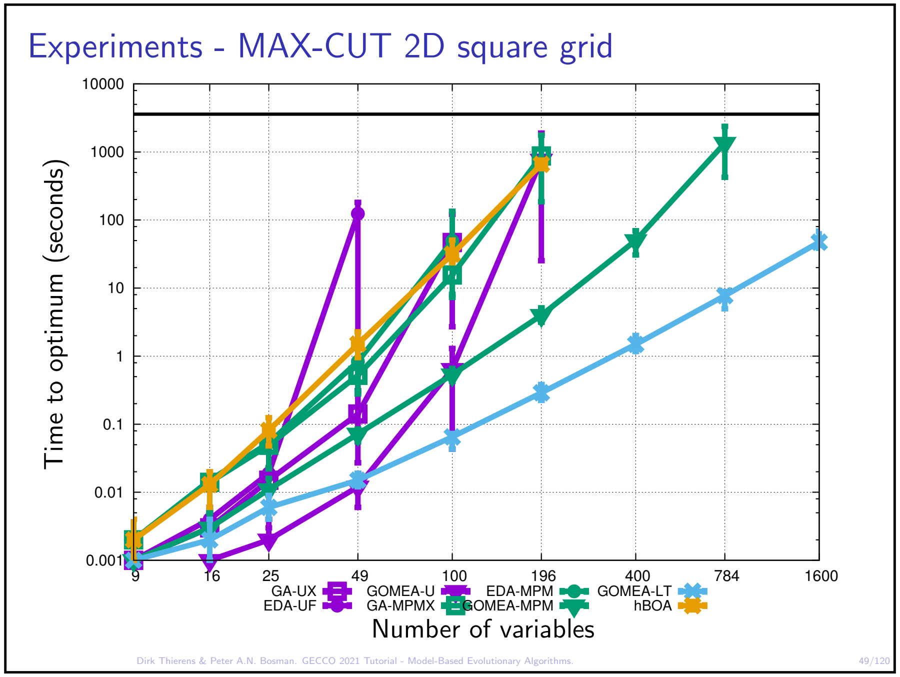
Experiments - Overlapping NK



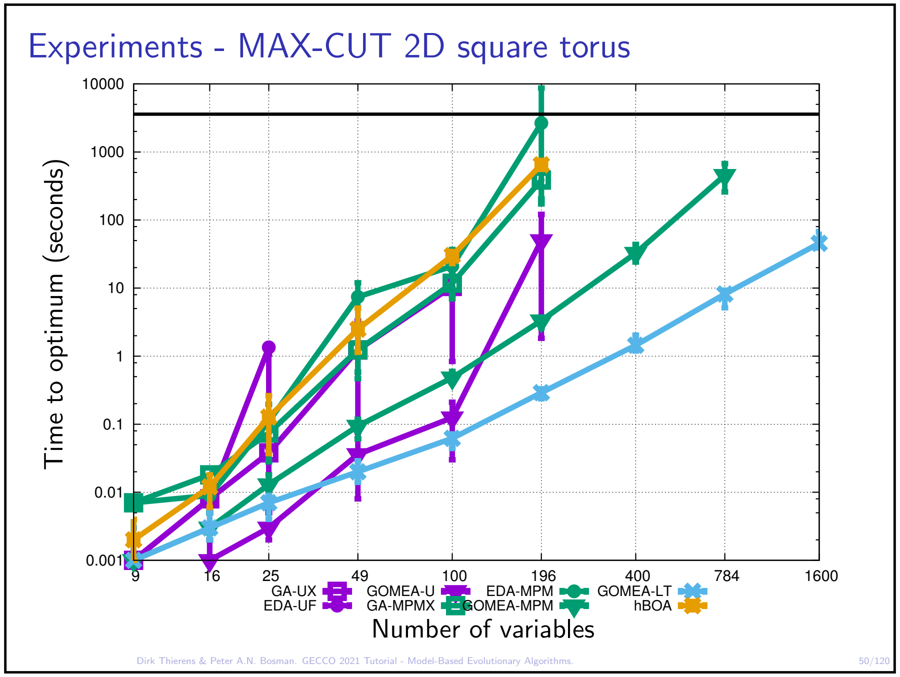
Experiments - HIFF



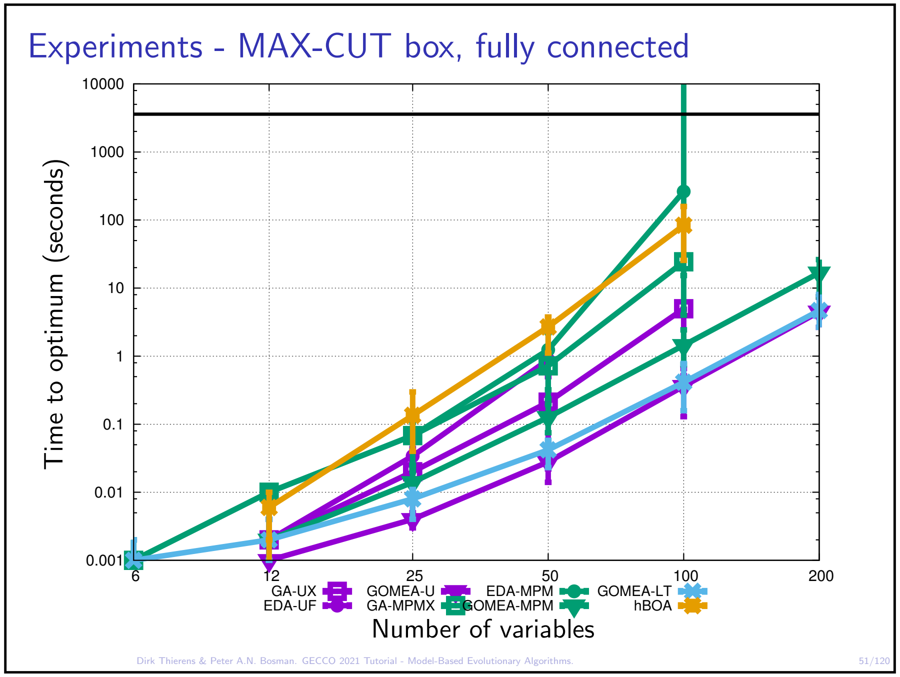
Experiments - MAX-CUT fully connected



Experiments - MAX-CUT 2D square grid

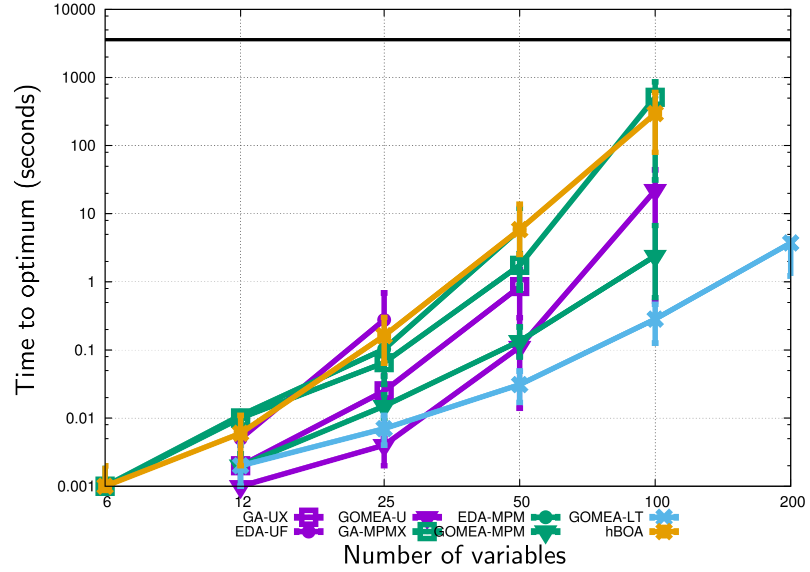


Experiments - MAX-CUT 2D square torus



Experiments - MAX-CUT box, fully connected

Experiments - MAX-CUT box, $\lfloor\sqrt(\ell)\rfloor$ nearest neighbors



Discussion

- Use of univariate structure leads to exponential scale-up on non-trivial problems
- GOMEA mixing is then the worst
- Use of learned structure leads to polynomial scale-up
- GOMEA mixing is then the best
- Most efficient: LT-GOMEA (a.k.a. LTGA)

Experiments: conclusion

- LTGA (= GOMEA with LT FOS) very efficient on Deceptive Trap function, Nearest-Neighbor NK landscape, and Hierarchical Trap function.
- 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 ?

Parameter-less Population Pyramid

- P3 (Goldman, Punch; 2014)
  - Similar to Harik-Lobo scheme: eliminates population-size parameter
- Each level of a pyramid-like structure is a population of solutions.
- Solutions are always hill-climbed.
- All solutions encountered are stored in the pyramid structure.
- At each level a Linkage Tree GA is run.
- Solutions climb the pyramid ladder with increasing fitness.
- Whenever a solution enters a level the linkage tree is relearned.

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).

Model-based optimization

- Assumption: problems are somehow structured
- Use induction to find structure
- Exploit structure for increased efficiency
- Preferable to enumeration or iterated random sampling
- What to induce?
- Use a model that defines reasonable structures
- Induce instance of the model
- Model capacity determines bias strength

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)

```
EDA
1 Initialize $\mathcal{P}$ with $n$ random solutions
2 Repeat until termination criterion met
    2.1 Select subset $\mathcal{S}$ from $\mathcal{P}$
    2.2 Estimate distribution from $\mathcal{S}$
    2.3 Draw new set of solutions $\mathcal{O}$ from distribution
    2.4 Update $\mathcal{P}$ with $\mathcal{O}$
```

Model-based optimization

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

- Optimal Mixing Evolutionary Algorithm (OMEA)

The Optimal Mixing Evolutionary Algorithm (OMEA)

- 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)

```
OMEA
1 Initialize $\mathcal{P}$ with $n$ random solutions
2 Repeat until termination criterion met
    2.1 Select subset $\mathcal{S}$ from $\mathcal{P}$
    2.2 Learn linkage model from $\mathcal{S}$
    2.3 Apply linkage-model guided optimal mixing
        to every individual in $\mathcal{P}$ to generate $\mathcal{O}$
    2.4 Replace $\mathcal{P}$ by $\mathcal{O}$
```

Model-based optimization

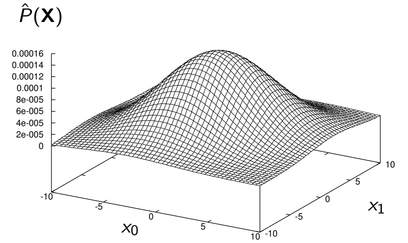
- General concepts
- Can be applied to different types of optimization problems/domains
- In second part of tutorial: focus on other domains than binary/integer

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:



- Only $\mathcal{O}(l^2)$ parameters (mean, covariance matrix)
- maximum-likelihood (ML) estimates well known

$$\hat{\boldsymbol{\mu}} = \frac{1}{|\boldsymbol{\mathcal{S}}|}\sum_{j=0}^{|\boldsymbol{\mathcal{S}}|-1}(\boldsymbol{\mathcal{S}}_j), \quad \hat{\boldsymbol{\Sigma}} = \frac{1}{|\boldsymbol{\mathcal{S}}|}\sum_{j=0}^{|\boldsymbol{\mathcal{S}}|-1}((\boldsymbol{\mathcal{S}}_j) - \hat{\boldsymbol{\mu}})((\boldsymbol{\mathcal{S}}_j) - \hat{\boldsymbol{\mu}})^T$$

- Can only model linear dependencies

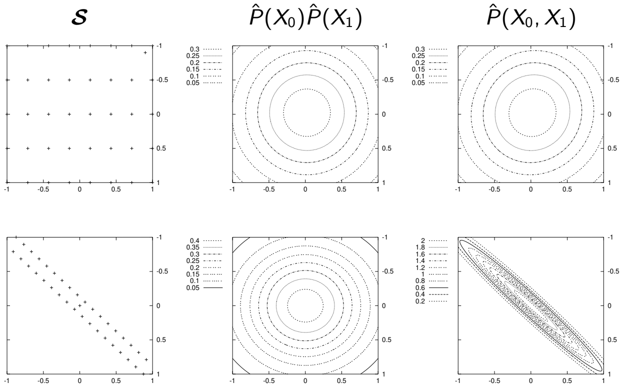
EDAs based on the Normal Distribution

- 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. $\hat{P}(X_0)\hat{P}(X_1)$)

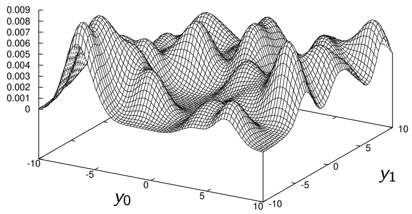

$\boldsymbol{\mathcal{S}}$ $\quad \hat{P}(X_0)\hat{P}(X_1)$ $\quad \hat{P}(X_0, X_1)$

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, Etxeberria, Lozano, and Peña (2000)
  - Salinas-Gutièrrez, Hernàndez-Aguirre, and Villa-Diharce (2011)
  - 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

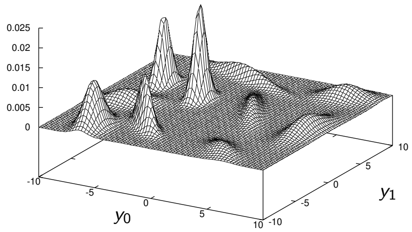
$\hat{P}^{\mathcal{N}_K}(Y_0, Y_1)(y_0, y_1)$



  - 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

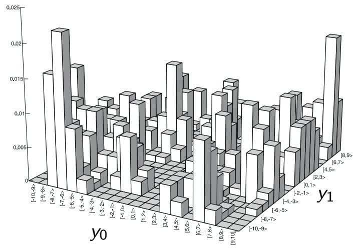
$\hat{P}^{\mathcal{N}_M}(Y_0, Y_1)(y_0, y_1)$



  - Gallagher, Fream, and Downs (1999)
  - Bosman and Thierens (2001)
  - Cho and Zhang (2002)
  - Ahn, Ramakrishna, and Goldberg (2004)
  - Li, Goldberg, Sastry, and Yu (2007)
  - Maree, Alderliesten, Thierens, and Bosman (2017)
- Trade–off between normal and normal kernels.
- Maximum-Likelihood Estimate is lot of effort (EM algorithm).
- Alternative: cluster, then est. normal (with max. likelihood).

EDAs based on the Histogram Distribution

$\hat{P}^{\mathcal{H}}(Y_0, Y_1)(y_0, y_1)$



  - 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.

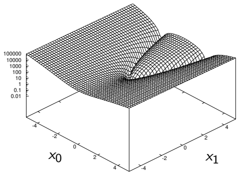
EDAs based on latent variable models

- 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:
      $\mathfrak{F}(\mathbf{x}) = \sum_{i=0}^{l-2} 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2$



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

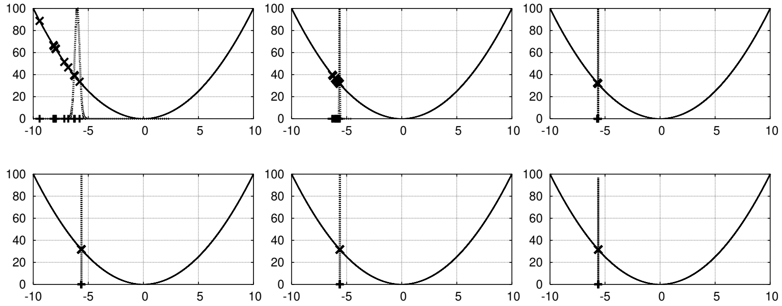
No attention for the gradient

- 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

$$\mathfrak{F}(\mathbf{x}) = x_0^2$$

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

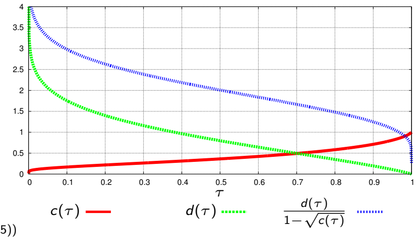
- Variance decreases (exponentially fast)

$$\lim_{t\to\infty}\{\hat{\sigma}(t)\} = \lim_{t\to\infty}\{\hat{\sigma}(0)c(\tau)^t\} = 0$$

- 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)

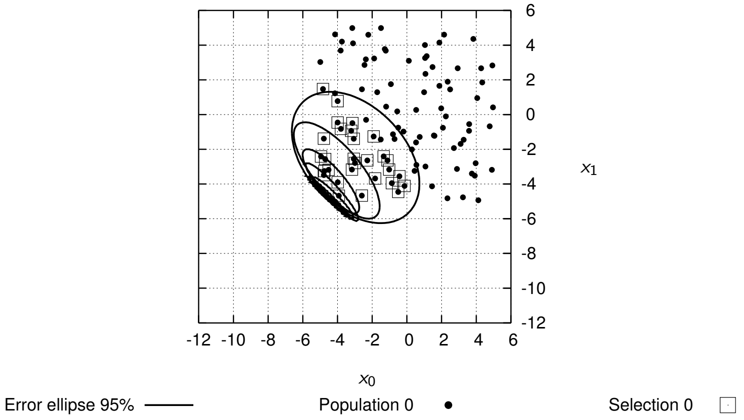


(Bosman and Grahl (2005))

Illustration on the 2-D plane function

$$\mathfrak{F}(\mathbf{x}) = x_0 + x_1$$

Progression in first 6 generations



What is missing?

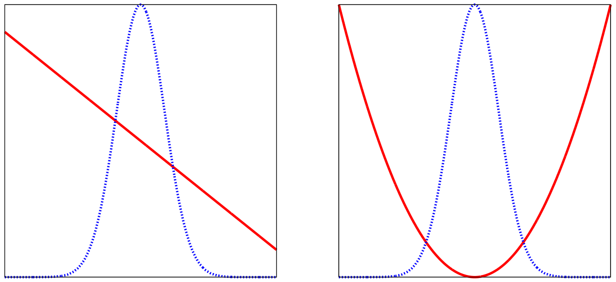
- Structure of landscape can be very complicated
- “Simple” normal distr. hardly matches global structure
- More involved distributions possible, but
  - harder, or even impossible, to estimate with ML
  - requires lots of data
- Local structure can be approximated but…
  - there is no generalization 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 $\hat{\mathbf{\Sigma}}$

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

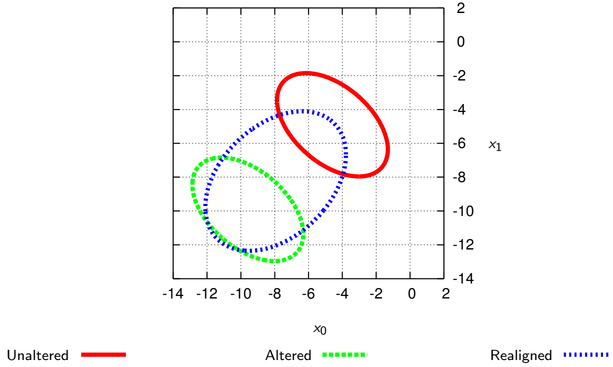
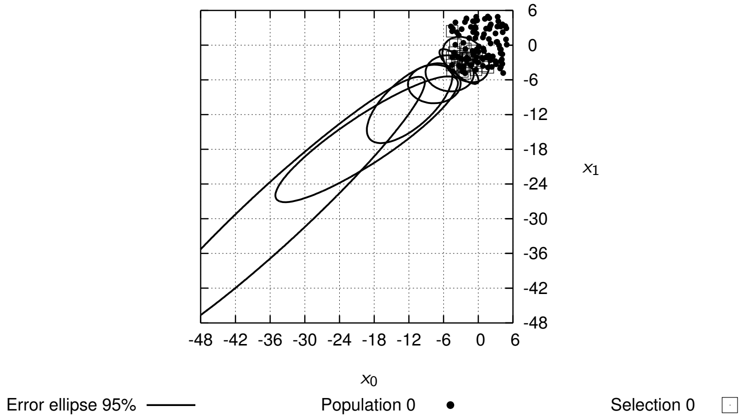


Illustration on a 2-D slope

$$\mathfrak{F}(\mathbf{x}) = x_0 + x_1$$

Progression in first 6 generations



AMaLGaM, CMA-ES, NES, and RP

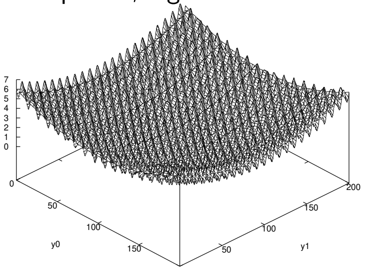
- AMaLGaM IDEA (or AMaLGaM for short)
  Adapted Maximum–Likelihood Gaussian Model Iterated Density-Estimation Evolutionary Algorithm
- Natural question:
  what is the relation to CMA-ES (Hansen (2001)) and NES (Wierstra, Schaul, Peters, and Schmidhuber (2008))
- Answer: the probability distribution
- 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.

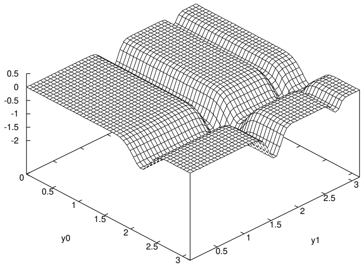
AMaLGaM, CMA-ES, NES, and RP

- 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) and RP uses ensembles of random projections to lower dimensions to estimate covariance matrices more efficiently.
- 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, NES scale better than AMaLGaM on such problems

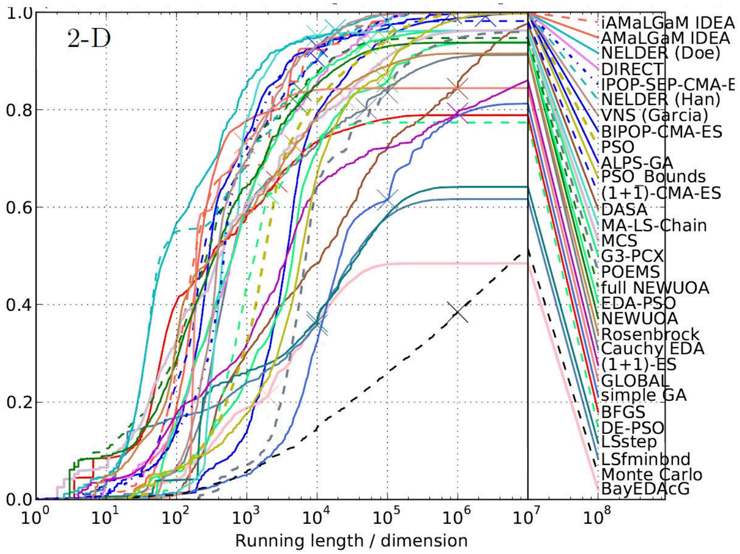
Parameter-free Gaussian EDAs

- 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)

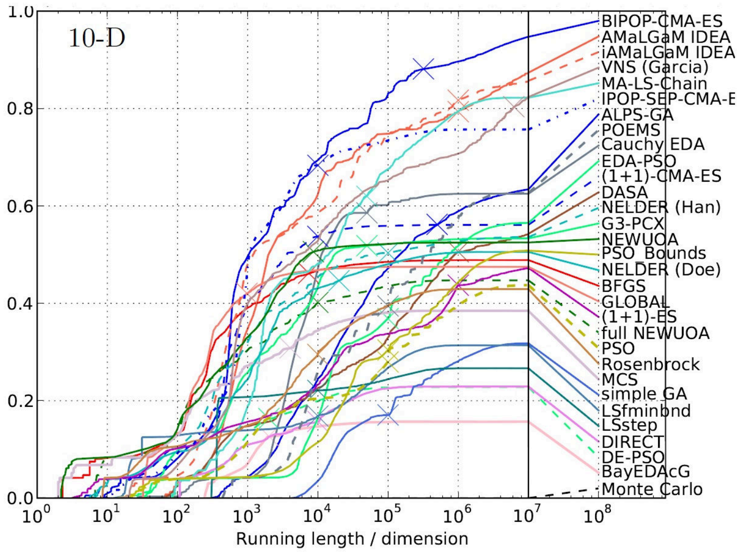




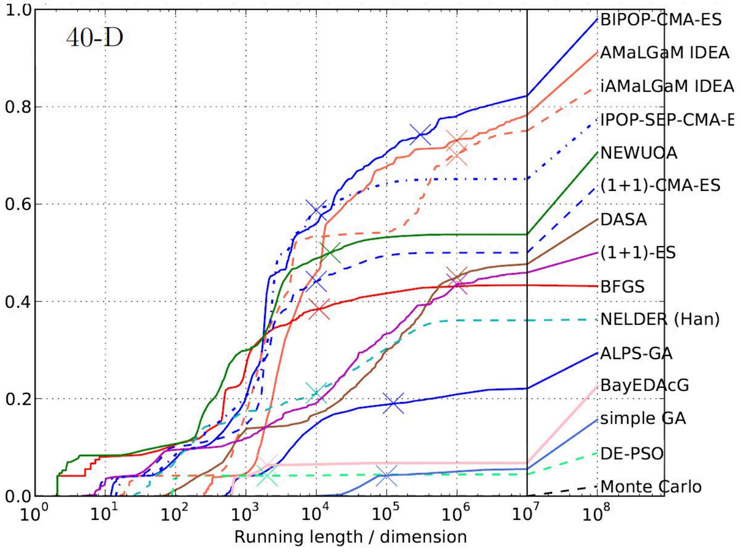
Noiseless BBOB comparison with other algorithms



Noiseless BBOB comparison with other algorithms



Noiseless BBOB comparison with other algorithms



Dimensionality reduction and problem-specific models

- Real-world problems may be high(er) dim. (at least, $\ell \gg 40$)
- Handling a full covariance matrix becomes expensive
- Restrict size of covariance matrix somehow
  - Random projections, tested up to $\ell = 10^3$ (Kabán, Bootkrajang, and Durrant (2013))
  - Projection-based restricted CMA-ES, tested up to $\ell = 10^3$ (Akimoto and Hansen (2016))
  - GOMEA-based, tested up to $\ell = 5 \cdot 10^6$ (with partial eval.'s) (Bouter, Alderliesten, Witteveen, and Bosman (2017))

Permutation Model-Based Evolutionary 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)

0	1	2	3
0.61	0.51	0.62	0.31

⇒

3	1	0	2
0.31	0.51	0.61	0.62

- Real-valued approaches can thus be used directly
  - Bosman and Thierens (2001) (normal EDA)
  - Larrañaga et al (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, but redundant encoding
- 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
  - Bengoetxea et al (2000)
  - 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: $P(\mathbf{X}) = P(X_0, X_4)P(X_1)P(X_3, X_2)$.
- Permutation: $P(\mathbf{X}) = \frac{2!1!2!}{5!}P(X_0, X_4)P(X_1)P(X_3, X_2)$.
- Random variable X_i: position of integer i in the permutation $\rightarrow$ 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

Permutation Model-Based Evolutionary Algorithms

- Gaussian “equivalent” in permutation space: Mallows model (GM-EDA)
  - 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
  - Ceberio, Irurozki, Mendiburu, and Lozano (2014)

Permutation Model-Based Evolutionary Algorithms

- GOMEA variant for permutations based on random keys
  - Bosman, Luong, and Thierens (2016)
- Requires a distance measure between permutations. Used product of:
  - Relative ordering information
  - Adjacency information
- Possibly add random rescaling and re-encoding
- First results are promising (permutation flow shop), mostly outperforming GM-EDA

Model-Based Genetic Programming

- 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?

Model-Based Genetic Programming

- Early works did not use grammar, e.g PIPE (Probabilistic Incremental Program Evolution)
  - Salustowicz and Schmidhuber (1997)
- Store probabilities of options (operators/terminals) for any node in the solution tree, bound maximum size
- All nodes thus independent

Model-Based Genetic Programming

- 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 its limitations

Model-Based Genetic Programming

- Extensions to Bayesian factorizations are also possible
- POLE does exactly this
  - Hasegawa and Iba (2008)
- MOSES does not use fixed template, but has incremental tree complexity (and model complexity) using special operators
  - Looks, Goertzel, and Pennachin (2004)
  - Looks (2006)

Model-Based Genetic Programming

- 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)

Model-Based Genetic Programming

- 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)

Model-Based Genetic Programming

- Impossible to cover everything in this tutorial, see literature
    - Kim, Shan, Nguyen, and McKay (2014)

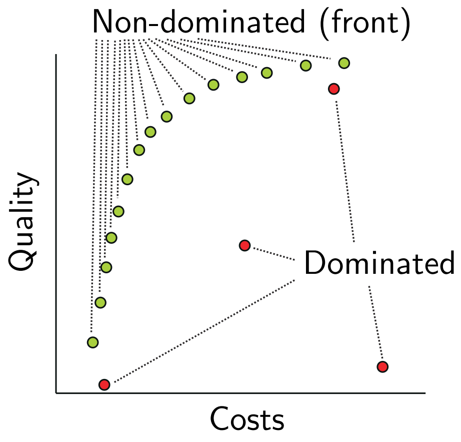
Representation			Models	Positional		Context (Label)		Context (Ancestral)	
Genotype		Model		Det.	Dep.	Det.	Dep.	Det.	Dep.
Tree GP like	Expression Tree	PPT	PIPE	1		-1	-1	-1	-1
			EDP	1		-1	-1	-1	1
			ECGP	1		-1	-1	-1	1
			POLE	1		-1	-1	-1	1
			AP	1		-1	-1	-1	-1
		N-gram	OFGP	-1	-1	1		-1	1
	Derivation Tree	Stochastic Chomsky Grammar	sSG-GP	-1	-1	1		-1	-1
			vSG-GP	-1	1	1		-1	1
			PEEL	-1	1	1		(-1,1)	1
			GMPE	-1	(-1,1)	1		(-1,1)	1
			PAGE	-1	(-1,1)	1		(-1,1)	1
			CFGT	-1	(-1,1)	1		(-1,1)	1
			CSGR	-1	(0,1)	1		-1	1
Graph			GNP-EDA						
GA like	Prefix Exp. Tree		bACP						
			gACP	-1		-1	-1	-1	
			GACP	1		-1	-1	-1	1
			DAP	-1	(0,1)	1		(-1,1)	1
	Prefix Der. Tree		GAP	-1	-1	1		1	
			EGAP	-1	-1	1		1	
			GBAP	-1	-1	1		1	
	Linear Genotype		BAP	1		-1	-1	-1	1
			BOAP	1		-1	-1	-1	1
			CGP-EDA	1		-1	-1	-1	-1
			N-gramGP	-1	-1	1		-1	1
			AntTAG	-1	-1	1		-1	-1
Hybrid			CFGR MOSES PAM-DGP						

Model-Based Genetic Programming

- Hybrid approach:
    - Use GOMEA on template
    - Learn new candidate functions using entropy of wrong cases
    - Best-ever performance by non-specific EA on even-parity problem
        - Virgolin, Alderliesten, Witteveen, and Bosman (2017)

Multi-objective Model-Based Evolutionary Algorithms

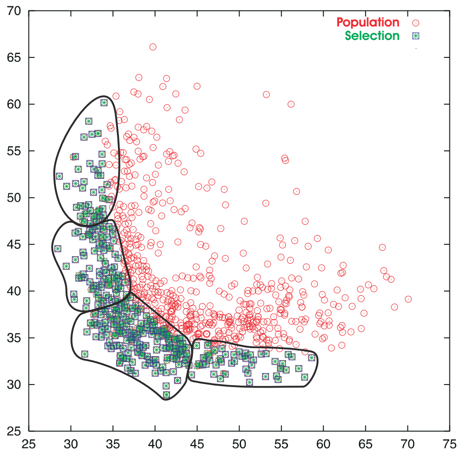
- Multiple objectives should be optimized simultaneously
- Conflicting objectives, no expression of weights
- Can't combine the objectives in a single scalar objective
- Want to present a set of promising alternatives to a decision maker
- Example: Maximize the quality and minimize the production costs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



Multi-objective Model-Based Evolutionary Algorithms

- 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
    - Bosman and Thierens (2002)
    - Pelikan, Sastry, and Goldberg (2009)

Multi-objective Model-Based Evolutionary Algorithms



Multi-objective Model-Based Evolutionary Algorithms

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

$$P_{(\varsigma,\boldsymbol{\theta})}(\mathcal{Z}) = \sum_{i=0}^{k-1} \beta_i P_{(\varsigma_i,\boldsymbol{\theta}_i)}(\mathcal{Z})$$

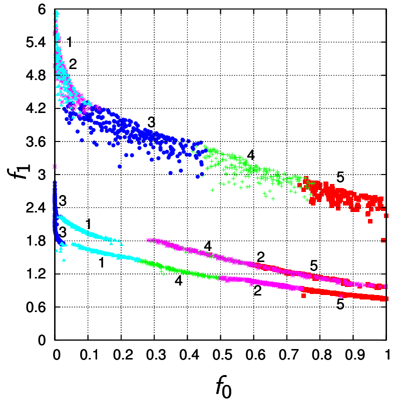
- Cluster solutions in objective space (e.g. k-means)
- Estimate a simpler distribution $P_{(\varsigma_i,\boldsymbol{\theta}_i)}(\mathcal{Z})$ in each cluster
- Set all mixing coefficients to $\beta_i = \frac{1}{k}$
- Parallel, specialized exploration along front

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)

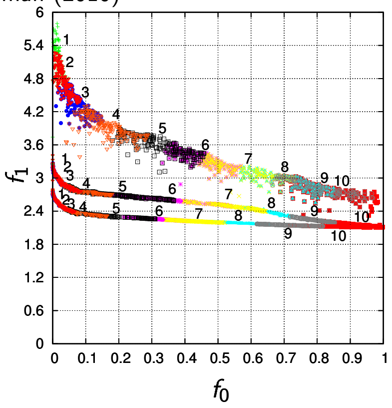
Multi-objective Model-Based Evolutionary Algorithms

- 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)



Multi-objective Model-Based Evolutionary Algorithms

- Other model-based MOEAs or MOEA concepts:
    - MOEA/D: simultaneously evolve different directions (Tchebycheff)
    - Can be combined with model-based EAs
        - e.g. CMA-ES, see: Wang, Liaw, and Ting (2016)
    - Copula-based EDAs
        - Martí, de Mello Jr., Sanchez-Pi, and Vellasco (2016)

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?

Conclusions

- 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)

Conclusions

- Books
  - Larrañaga and Lozano (eds) (2001). Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer.
  - Lozano, Larrañaga, Inza, Bengoetxea (2006). Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms, Springer.
  - Pelikan, Sastry, Cantú-Paz (eds) (2006). Scalable Optimization via Probabilistic Modeling: From Algorithms to Applications, Springer.

Acknowledgements

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