Model-Based Evolutionary Algorithms

Thierens, Dirk; Bosman, Peter A.N.

DOI
10.1145/3638530.3648423

Publication date
2024

Document Version
Final published version

Published in
GECCO '24 Companion

Model-based Evolutionary Algorithms

a GECCO 2024 Tutorial

Dirk Thierens
Utrecht University
Department of Information and Computing Sciences
Utrecht, The Netherlands
D.Thierens@uu.nl

Universiteit Utrecht

Peter A.N. Bosman

Senior Researcher at
Centrum Wiskunde & Informatica (CWI)
(Center for Mathematics and Computer Science)
Life Sciences and Health Research Group
Amsterdam, The Netherlands
Peter.Bosman@cwi.nl

Professor of Evolutionary Algorithms at
Delft University of Technology
Department of Software Technology
Algorithmics group
Delft, The Netherlands
P.A.N.Bosman@tudelft.nl

CWI Centrum Wiskunde & Informatica

TU Delft Delft University of Technology

Outline

What ?

Evolutionary Algorithms

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

Model-Based Evolutionary Algorithms (MBEA)

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

MBEA = Evolutionary Computation + Machine Learning

Why ?

Goal: Black Box Optimization

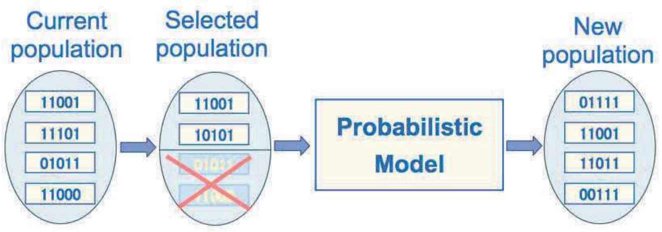
Little known about the structure of the problem.

* Classical EAs: need suitable representation & variation operators

* Model-Based EAs: learn structure from good solutions

Probabilistic Model-Building GA (PMBGA)

Discrete Representation



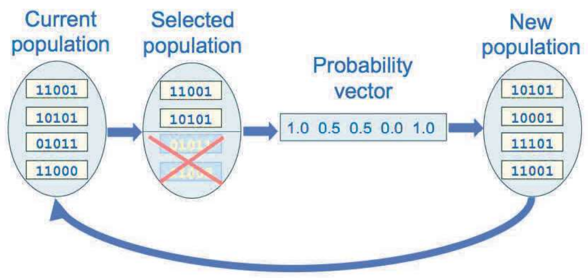
Type of Models

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

Univariate PMBGA

Model

- Model: probability vector $[p_1, \ldots, p_\ell]$ (ℓ: string length)
- $p(X) = \prod_{i=1}^{\ell} p(x_i)$ ($p(x_i)$: univariate marginal distribution)



A hard problem for the univariate model

Data
000000
111111
010101
101010
000010
111000
010111
111000
000111
111111

Marginal Product model

	$\hat{P}(X_0X_1X_2)$	$\hat{P}(X_3X_4X_5)$
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 model

	$\hat{P}(X_0)$	$\hat{P}(X_1)$	$\hat{P}(X_2)$	$\hat{P}(X_3)$	$\hat{P}(X_4)$	$\hat{P}(X_5)$
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 model: $0.5 \cdot 0.6 \cdot 0.5 \cdot 0.5 \cdot 0.6 \cdot 0.5 = 0.0225$
- MP model: $0.4 \cdot 0.4 = 0.16$

Learn problem structure while searching

- Without a good decomposition of the problem, important partial solutions - building blocks- get disrupted in variation.
- Disruption leads to inefficiency.
- Selection increases proportion of good building blocks and thus correlations between variables of these building blocks.
- Learn the model structure - this is, which variables are correlated.

Bivariate PMBGA

Model pairwise interactions: conditional probabilities

COMIT

- Probabilistic model: Dependency Tree (Baluja, Davies; 1997).
- Fully connected weighted graph between problem variables.
- Weights are the mutual information $I(X, Y)$ between the variables.
- Compute maximum spanning tree of this weighted graph.
- Resulting tree minimizes the Kullback-Leibler divergence between the joint probability distribution and the second order approximation probability model:

$$p(X) = \prod_{i=1}^{n} p(X_i | X_{parent(i)})$$

- $p(X)$ is the class of distributions with a tree as graphical model.

Multivariate PMBGA

ECGA: Harik, 1999

Extended Compact GA (ECGA)

- ECGA: first PMBGA going beyond pairwise dependencies.
- $p(X) = \prod_{g=1}^{G} p(X_g)$, with X_g mutual exclusive groups.
- ECGA greedily searches for the Marginal Product Model that minimizes the minimum description length (MDL).
- $MDL(M, D) = D_{Model} + D_{Data}$
  1. Model complexity D_{Model}: complexity of describing the model.
  2. Compressed population complexity D_{Data}: complexity of describing the data within the model

ECGA

Minimum Description Length score

- Model Complexity $D_{Model} = log_2(N+1) \sum_i (2^{S_i} - 1)$.
- Compressed Population Complexity $D_{Data} = N \sum_i H(M_i)$.

N : Population size
S_i : size of partition i
M_i : marginal distribution of the partition i
$H(M_i)$: entropy of the marginal distribution of the partition i

Small example

population size $N = 8$, string length $l = 4$

1	0	0	0
1	1	0	1
0	1	1	1
1	1	0	0
0	0	1	0
0	1	1	1
1	0	0	0
1	0	0	1

- Start from simplest model: the univariate factorization.
- Join two groups that give largest improvement in MDL score.
- Stop when joining of two groups no longer improves MDL score.

MPM	Combined Complexity
$[I_1][I_2][I_3][I_4]$	44.0
$[I_1, I_2][I_3][I_4]$	46.7
$[\mathbf{I_1}, \mathbf{I_3}][\mathbf{I_2}][\mathbf{I_4}]$	**39.4**
$[I_1, I_4][I_2][I_3]$	46.7
$[I_1][I_2, I_3][I_4]$	46.7
$[I_1][I_2, I_4][I_3]$	45.6
$[I_1][I_2][I_3, I_4]$	46.7
$[I_1, I_3, I_2][I_4]$	48.6
$[I_1, I_3, I_4][I_2]$	48.6
$[I_1, I_3][I_2, I_4]$	41.4

MPM $[I_1, I_3]$, $[I_2]$, $[I_4]$ has the lowest combined complexity:
⇒ it is the best Marginal Product Model to compress the population,
⇒ it captures the most dependencies in the set of solutions.

Multivariate PMBGA

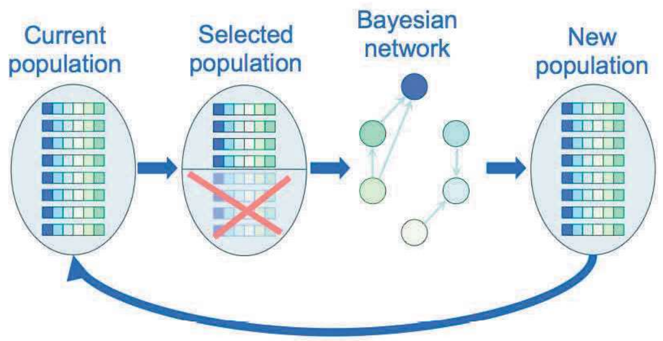
Bayesian network model

Bayesian Network

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

Multivariate PMBGA

Bayesian network model



Multivariate PMBGA

Pelikan, Goldberg, Cantú-Paz, 1998; Pelikan, Goldberg, 2001

Hierarchical Bayesian Optimization Algorithm (hBOA)

- Model: Bayesian Network
- 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

MN-EDA: Santana, 2005; DEUM: Shakya, McCall, 2007; MOA: Shakya, Santana, 2008

Markov Network

- Probability model is undirected graph.
- Factorise the joint probability distribution in cliques of the undirected graph.
- Markovian Optimisation Algorithm (MOA): does not explicitly factorise the distribution but uses the local Markov property and Gibbs sampling to generate new solutions.

Dependency Model = Family Of Subsets ($\mathcal{FOS}$)

$\mathcal{FOS}$ models instead of Probabilistic models

$\mathcal{FOS}$ model

- Identify groups of problem variables that together make an important contribution to the quality of solutions.
- These variable groups are interacting in a non-linear way and should be recombined as a block = building block.
- $\mathcal{FOS}$ model: the dependency structure is a set of subsets of the problem variables.

Example $\mathcal{FOS}$ models

- Univariate $\mathcal{FOS}$:

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

$\rightarrow$ Every variable is modeled to be independent of other variables.

- Marginal product $\mathcal{FOS}$:

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

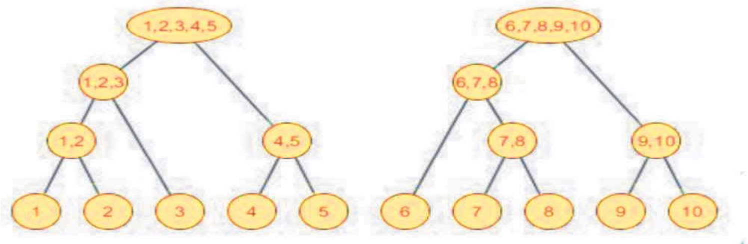
$\rightarrow$ Every group of variables is modeled to be independent of other variables.

Example $\mathcal{FOS}$ models

- Linkage Tree $\mathcal{FOS}$: hierarchical cluster tree.

$$\mathcal{FOS} = \{1,2,3,4,5\},\{6,7,8,9,10\},\{1,2,3\},\{6,7,8\},\{1,2\},\{4,5\},$$
$$\{7,8\},\{9,10\},\{1\},\{2\},\{3\},\{4\},\{5\},\{6\},\{7\},\{8\},\{9\},\{10\}$$

$\rightarrow$ Problem variables in subset are modeled to be dependent at one level but become independent at lower level.



Linkage Tree Learning

- Start from univariate structure.
- Build linkage tree using bottom-up hierarchical clustering.
- 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 $I(X,Y)$.

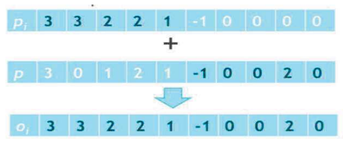
$$I^{clust}(X_{F^i}, X_{F^j}) = \frac{1}{|X_{F^i}||X_{F^j}|} \sum_{X \in X_{F^i}} \sum_{Y \in X_{F^j}} I(X,Y).$$

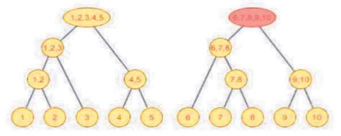
Computational Efficiency

- Computing the mutual information between pairs of variables $= O(\ell^2)$ computation.
- Bottom-up hierarchical clustering $= O(\ell^2)$ computation with the *reciprocal nearest neighbor chain* algorithm.

Gene-pool Optimal Mixing Evolutionary Algorithm

- Each generation a new $\mathcal{FOS}$ model is learned.
- For each solution in the population, all subsets of the $\mathcal{FOS}$ are used as crossover mask.
- Randomly select a donor solution from the population, and its values at the crossover mask replace the variable values from the current solution.
- Crossover is greedy: only improvements (or equal) are accepted.





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
```

Dependency Structure Matrix Genetic Algorithm

DSMGA-II: Hsu, Yu; 2015

- DSMGA-II stores pairwise dependency information, measured by mutual information, in a Dependency Structure Matrix.
- The $\mathcal{FOS}$ model consists of Incremental Linkage Sets, one for each problem variable.
- The Incremental Linkage Sets are constructed by incrementally adding - one-by-one - the next most dependent variable.
- A variant of the optimal mixing operator - called restricted mating - is used to generate new solutions.

Parameter-less Population Pyramid (P3)

Goldman, Punch; 2014

- Each level of a pyramid-like structure is a population of solutions.
- All solutions encountered are stored in the pyramid structure.
- P3 uses multiple Linkage Tree $\mathcal{FOS}$ models, one at each level of a pyramid structure.
- At each level, the Optimal Mixing procedure is executed.
- Solutions climb the pyramid ladder with increasing fitness.
- Whenever a solution enters a level, the Linkage Tree is relearned.

Empirical Linkage Learning

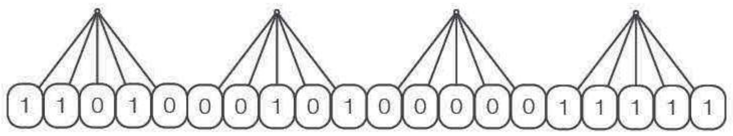
Przewozniczek, Komarnicki, Frej; 2020, 2021

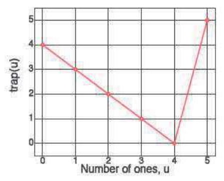
- Statistical Linkage Learning:
  learn dependencies using mutual information.
- Empirical Linkage Learning:
  learn dependencies using local optimization.
- Advantage: only learns true dependencies (no false linkage).
- Disadvantage: computationally more expensive.
  $\rightarrow$ can exploit the use of local search in hybrid EAs
  (Tinós, Przewozniczek, Whitley; 2022).

Deceptive Trap Function

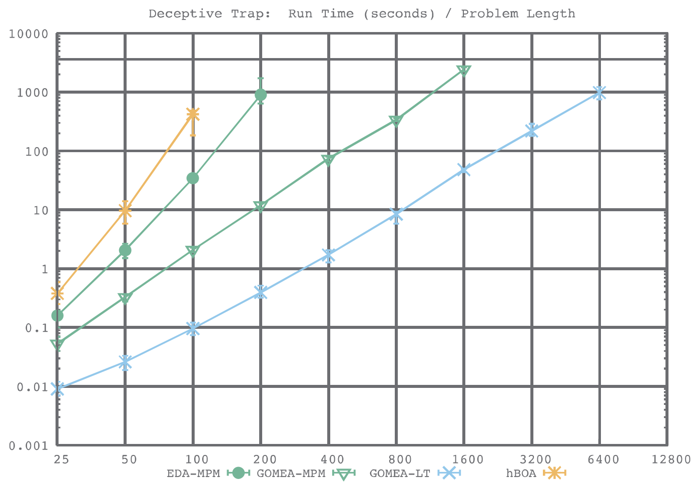
Interacting, non-overlapping, deceptive groups of variables.

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



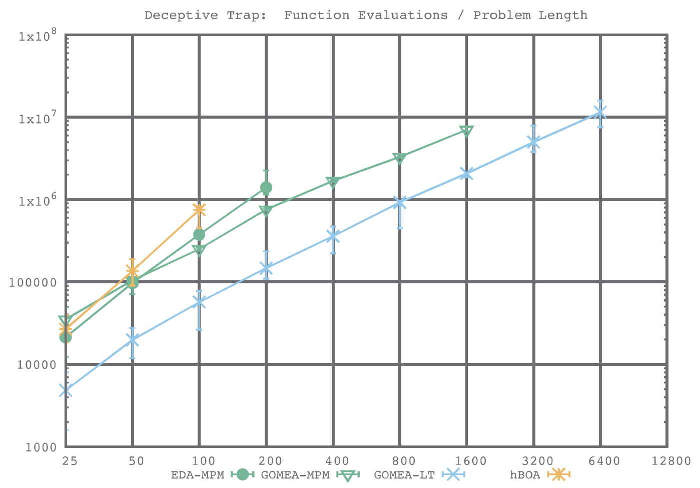


Deceptive, Randomly Shuffled Trap Function (k=5)



(99/100 runs successful, adaptive population sizing)

Deceptive, Randomly Shuffled Trap Function (k=5)



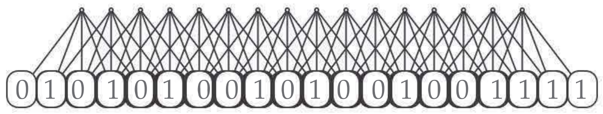
(99/100 runs successful, adaptive population sizing)

Adjacent NK-landscape

- Overlapping, neighboring random subfunctions

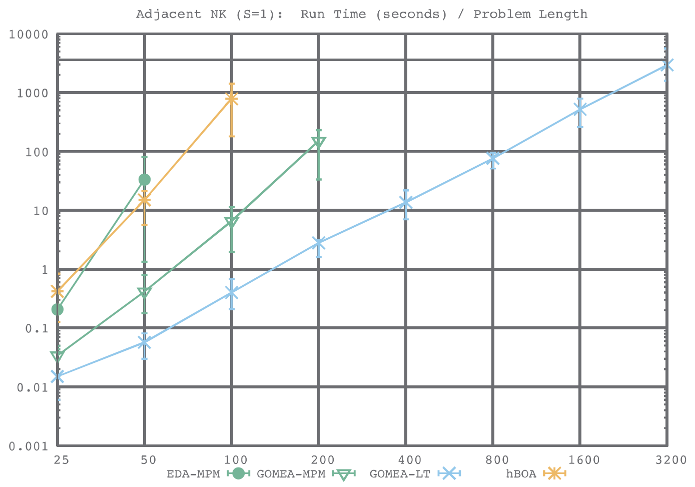
$$f_{\text{NK-S1}}(x) = \sum_{i=0}^{l-k} f^{\text{sub}}_{\text{NK}}\left(x_{(i,\ldots,i+k-1)}\right) \quad \text{with } f^{\text{sub}}_{\text{NK}}\left(x_{(i,\ldots,i+k-1)}\right) \in [0..1]$$

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



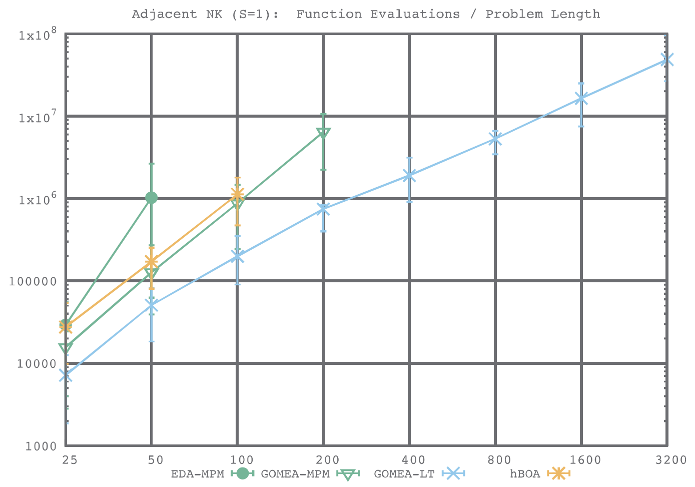
- Global optimum computed by dynamic programming
- Benchmark function: structural information is not known !
- $\Rightarrow$ Randomly shuffled variable indices.

Adjacent NK Landscape, Randomly Shuffled (k=5)



(99/100 runs successful, adaptive population sizing)

Adjacent NK Landscape, Randomly Shuffled (k=5)



(99/100 runs successful, adaptive population sizing)

Conclusion

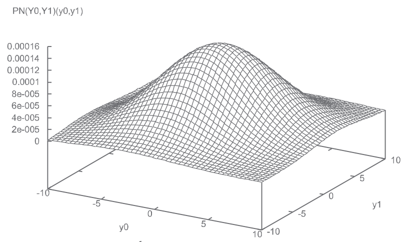
- Optimal Mixing removes the noise from building block decision making.
- Optimal Mixing requires far smaller populations than Probabilistic-Model Building EAs Algorithms.
- Optimal Mixing more efficient and scales up better than multivariate PMBGAs (hBOA, ecGA).
- Average pairwise mutual-information measure allows fast Linkage Tree building.

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

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

- Can only model linear dependencies

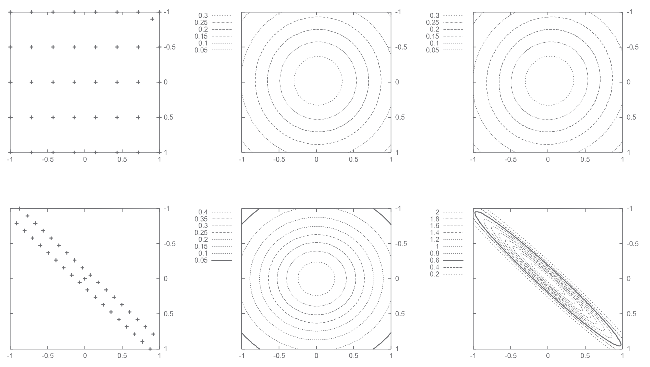
EDAs based on the Normal Distribution

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

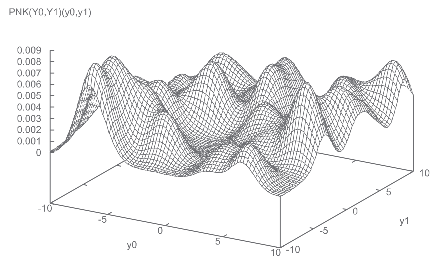
$\mathcal{S}$ $\qquad$ $\hat{P}(X_0)\hat{P}(X_1)$ $\qquad$ $\hat{P}(X_0, X_1)$



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

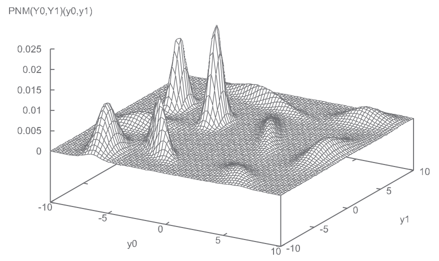
EDAs based on the Normal–kernels distribution



- 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

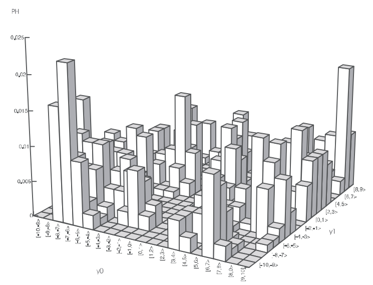
EDAs based on the Normal–mixture distribution



- Gallagher, Fream, and Downs (1999)
- Bosman and Thierens (2001)
- 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



- Bosman and Thierens (2000)
- Tsutsui, Pelikan, and Goldberg (2001)

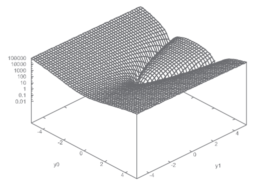
- Easy to implement and map to integers.
- Require many bins to get a good estimate.
- Curse of dimensionality.
- Greedy incr. factorization selection hardly possible.

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

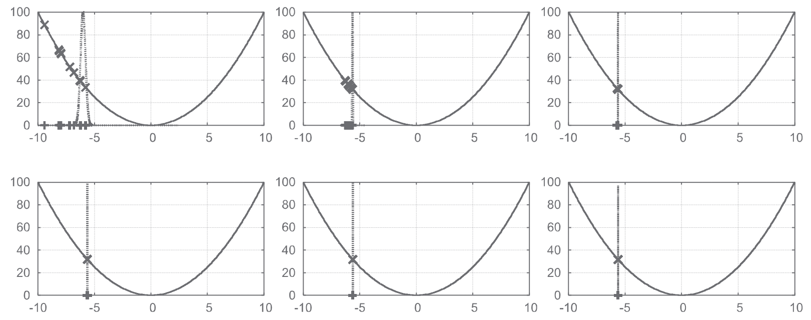
No attention for the gradient

- 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

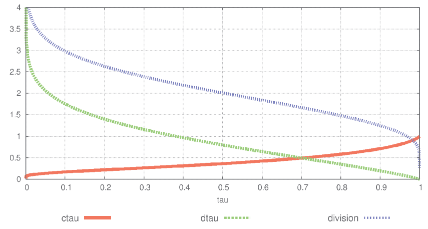
- Variance decreases (exponentially fast)

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

- 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)

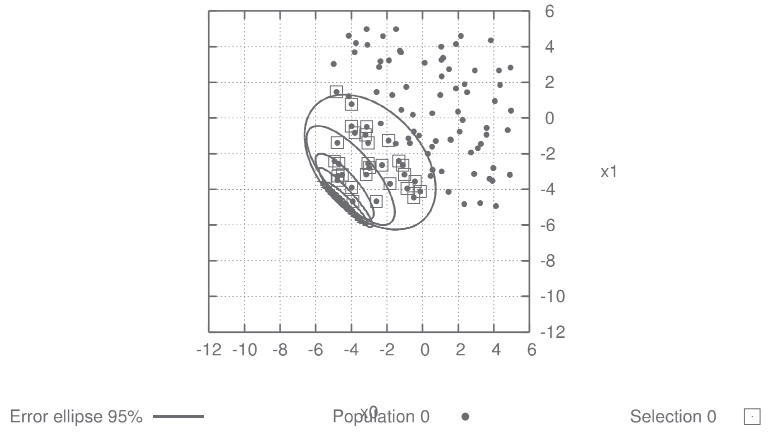


(Bosman and Grahl (2005))

Illustration on the 2-D plane function

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

Progression in first 6 generations



What is missing?

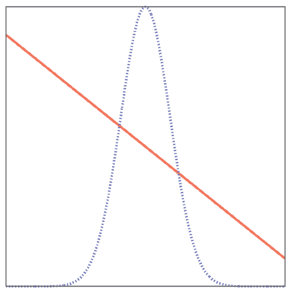
- Structure of landscape can be very complicated
- "Simple" normal distr. hardly matches global structure
- More involved distributions possible, but
  - harder, or even impossible, to estimate with ML
  - requires lots of data
- Local structure can be approximated but. . .
  - there is no generalization 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

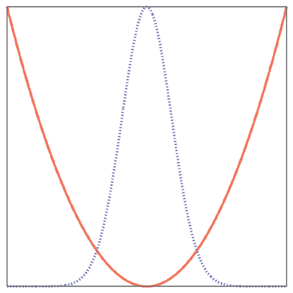
Ways to improve

- 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

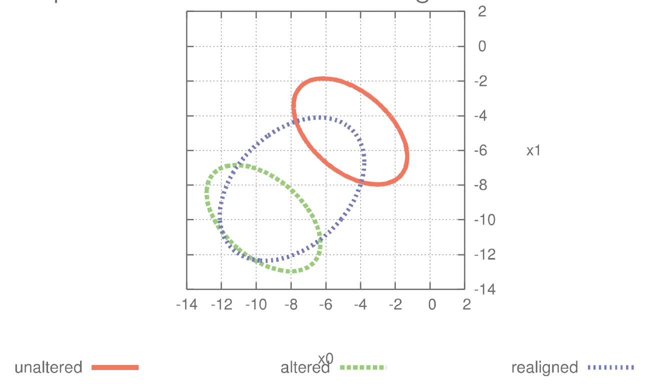
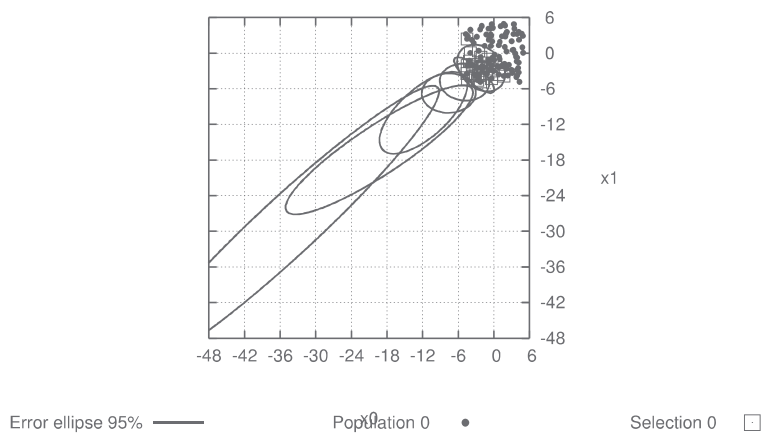


Illustration on a 2-D slope

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

Progression in first 6 generations



AMaLGaM, CMA-ES, NES, and RP

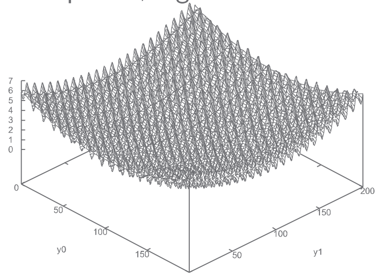
- AMaLGaM IDEA (or AMaLGaM for short)
  Adapted Maximum–Likelihood Gaussian Model Iterated Density-Estimation Evolutionary Algorithm
- Natural question:
  what is the relation to CMA-ES (Hansen (2001)) and NES (Wierstra, Schaul, Peters, and Schmidhuber (2008))
- Answer: the probability distribution
-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

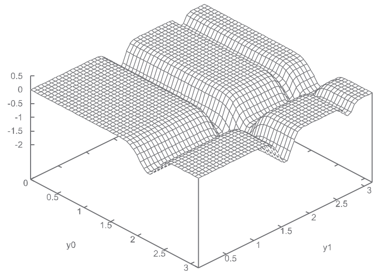
AMaLGaM, CMA-ES, NES, and RP

- 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

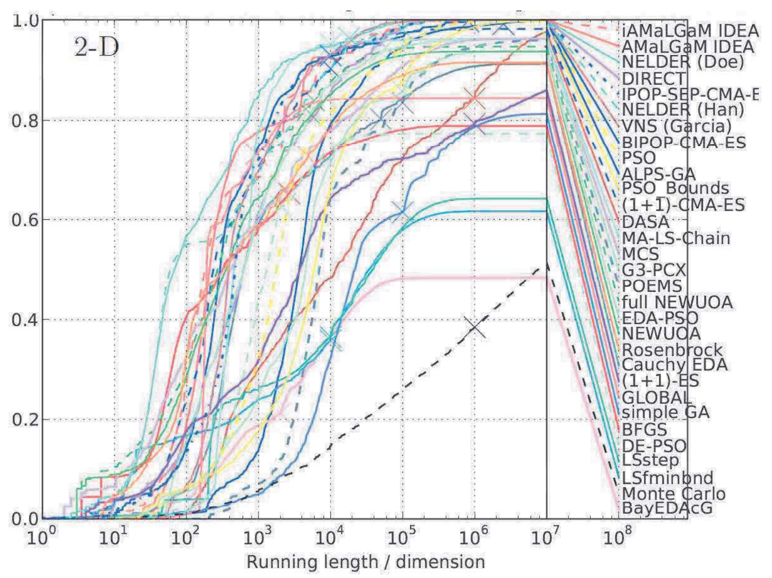
Parameter-free Gaussian EDAs

- 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)

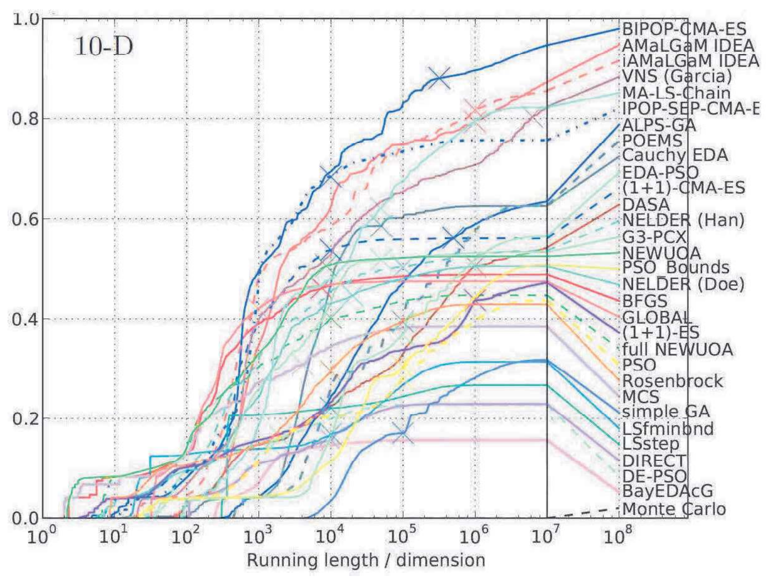




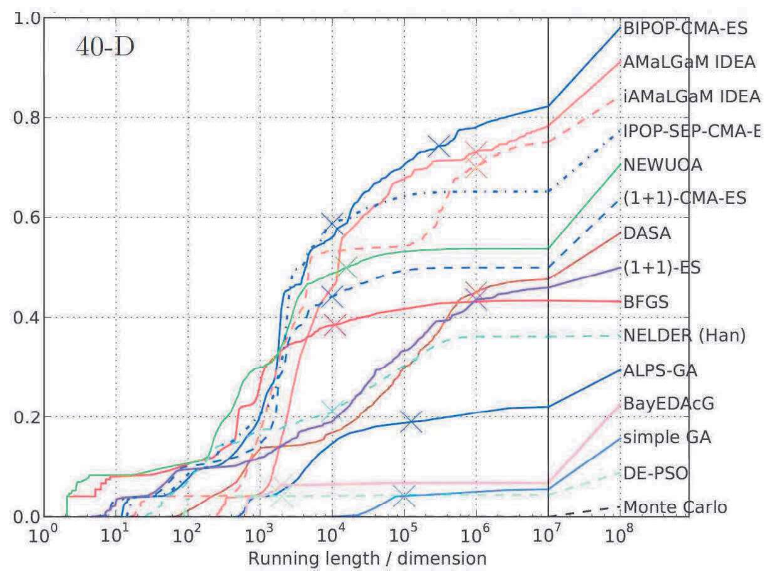
Noiseless BBOB comparison with other algorithms



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))

Model-based EAs & Decomposition

- Recall **decomposability** and **optimal mixing**:
  - Key **assumption**/**hypothesis**:
    - Many **problems** can be (non-trivially) **decomposed**, i.e. (small) **sets of variables** can together make an **important** (above average) **contribution** to a solution's **quality**
  - Logical **structure** to capture decomposition structure: **Family Of Subsets (FOS)**
  - Leverage **FOS** in **(Gene-pool) Optimal Mixing** operator
  - Better **scalability** in general & higher **parallelizability**
- Can we do this for other representations too?
  - **Yes**! Today will show **RV** and **GP**.

Fundamentally improve scalability?

Slides: thanks to Anton Bouter

- Parallels between exploiting linkage information in **discrete** and **real-valued** optimization.
- Real-Valued Gene-Pool Optimal Mixing Evolutionary Algorithm (**RV-GOMEA**)
- **Gray-Box** Optimization

Problem Separability/Decomposability

- Problem is **fully separable** iff (assume minimization):

$$\arg\min_{x_1,\ldots,x_\ell} f(x)$$
$$=$$
$$\left(\arg\min_{x_1} f(x_1,\ldots),\ldots,\arg\min_{x_\ell} f(\ldots,x_\ell)\right)$$

- All problem variables are then **independent**.
- Global optimum can be found by optimizing **one variable** at a time, regardless of other variables.

Problem Separability

- Problem is **additively separable** if:

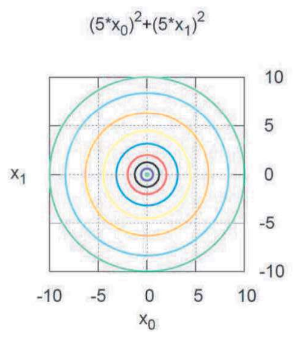
$$f(x) = \sum_{i=1}^{k} f_i(X_i)$$

- All X_i are **disjoint** subsets of variables of x, e.g.:

$$f(x) = f_1(x_1,x_2) + f_2(x_3) + f_3(x_4,x_5)$$

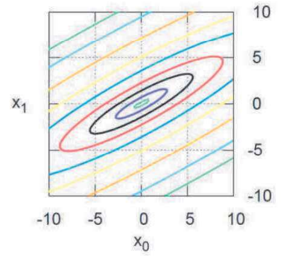
Problem Separability

- **Sphere** function is **fully separable**.



- **Rotated ellipsoid** function is **non-separable**.

$$(1 \times (\cos(-0.5) \times x_0 - \sin(-0.5) * x_1))^2 +$$
$$(5 \times (\sin(-0.5) \times x_0 + \cos(-0.5) * x_1))^2$$

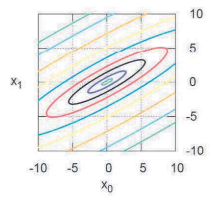


Problem Separability

- **Sum of rotated ellipsoid blocks** function is **additively separable**.

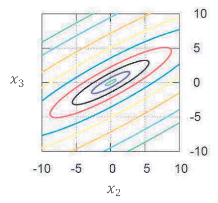
$$f(x) = f_{Rot.Ell}(x_0, x_1) + f_{Rot.Ell}(x_2, x_3) + \cdots$$

$$(1 \times (\cos(-0.5) \times x_0 - \sin(-0.5) * x_1))^2 +$$
$$(5 \times (\sin(-0.5) \times x_0 + \cos(-0.5) * x_1))^2$$



$+$

$$(1 \times (\cos(-0.5) \times x_2 - \sin(-0.5) * x_3))^2 +$$
$$(5 \times (\sin(-0.5) \times x_2 + \cos(-0.5) * x_3))^2$$



Exploiting Linkage

- Ability of solving problems with **dependent variables** depends on complexity of **covariance matrix**.
- Full covariance matrix models **all** $O(\ell^2)$ pair-wise **dependencies**.
- Sampling takes $O(\ell^3)$ time due to **Cholesky decomposition**.
- **Univariate** models perform very **poorly** on **non-separable** problems (if ill-conditioned).

Exploiting Linkage

- Efficiently **solving** decomposable problems.
  - Variation operators **matching** the dependencies of the optimization problem.
  - Full covariance matrix on decomposable problem: **suboptimal scalability**.

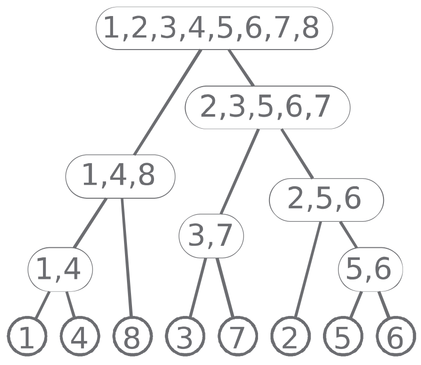
Exploiting Linkage

- Mixing building blocks **separately**.
  - Mixing multiple building blocks can **decrease fitness** of some building blocks despite **overall better fitness**.
  - → **Gene-pool Optimal Mixing** (GOM): Apply variation **building-block-wise**, **accept** if **not worse***.
  - **Only few variables change** per evaluation.
    - **Potential downside in BBO**
    - **Potential upside in GBO**

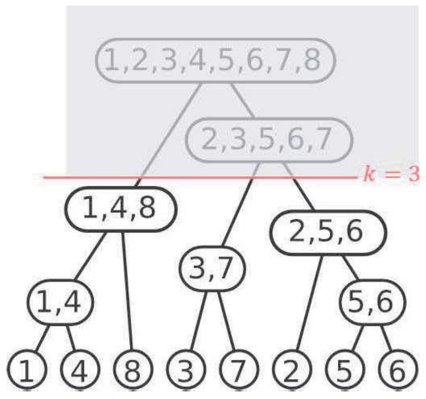
RV-GOMEA

- Real-Valued Gene-pool Optimal Mixing Evolutionary Algorithm (**RV-GOMEA**)
  - Integration of **GOMEA** and **AMaLGaM**
  - Can also integrate with **CMA-ES**
- Variation by sampling from multivariate normal distributions (similar to **AMaLGaM** (/ CMA-ES))…
- …applied to building blocks according to **GOMEA**.

Linkage Models (Linkage Tree)



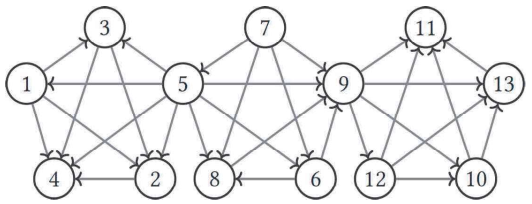
Linkage tree



Bounded linkage tree ($k = 3$).

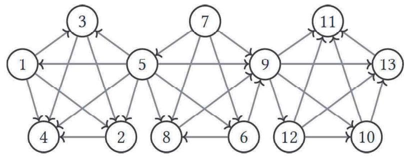
Linkage Model (Conditional Linkage)

- **Conditional linkage**
  - Explicitly represent overlap between linkage sets
  - Simplest: **each variable conditionally dependent** on **set of other variables** (e.g., Bayesian (Gaussian) network $P(\boldsymbol{X}) = \prod_{i=0}^{\ell-1} P(X_i|X_{\boldsymbol{\pi}_i})$)



Linkage Model (Conditional Linkage)

$$P(\boldsymbol{X}) = \prod_{i=0}^{\ell-1} P(X_i | X_{\pi_i})$$



- Sample variable i **conditionally** on parent variables $\boldsymbol{\pi}_i$
  - Requires **conditional Gaussians** (not covered further)
  - **GOM** per conditionally sampled variable, accept if **not worse***
  - Can still break dependencies due to mixing

    → sample **entire** Gaussian network **at once** afterward to "repair", accept if not worse

(Bouter et al, Proc. GECCO 2020; Andreadis et al, Proc. GECCO 2024)

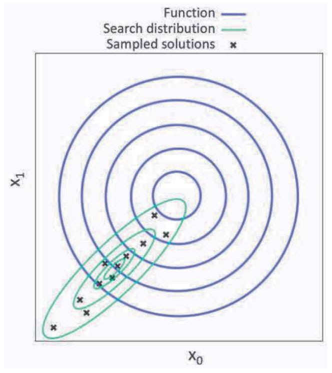
Linkage Learning

- Mutual information **less effective** in **real-valued** optimization.
- Gaussian:

$$I(X,Y) = \log\left(\sqrt{\frac{1}{1-\rho^2}}\right), \rho = \frac{cov(x,y)}{\sigma_x \sigma_y}$$

- Search distribution is **not always aligned** with **fitness function** (contours).
- Alternative: **fitness-based** linkage learning (next slide).

(Olieman et al, IEEE TEVC, 2020)



Linkage Learning

Dependency Estimation – Differential Grouping

- Identify dependency between x_1 and x_2.
  - Choose initial points $a = [x_1, x_2, x_3, \ldots]$ and b $= [x_1, x_2', x_3, \ldots]$
  - $a' = [x_1 + \delta, x_2, x_3, \ldots]$
    $b' = [x_1 + \delta, x_2', x_3, \ldots]$
  - $\Delta_a = f(a') - f(a)$
    $\Delta_b = f(b') - f(b)$
  - Considered independent if $|\Delta_a - \Delta_b| < \epsilon$

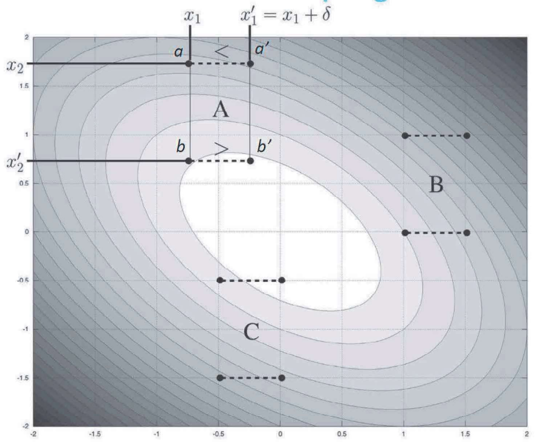
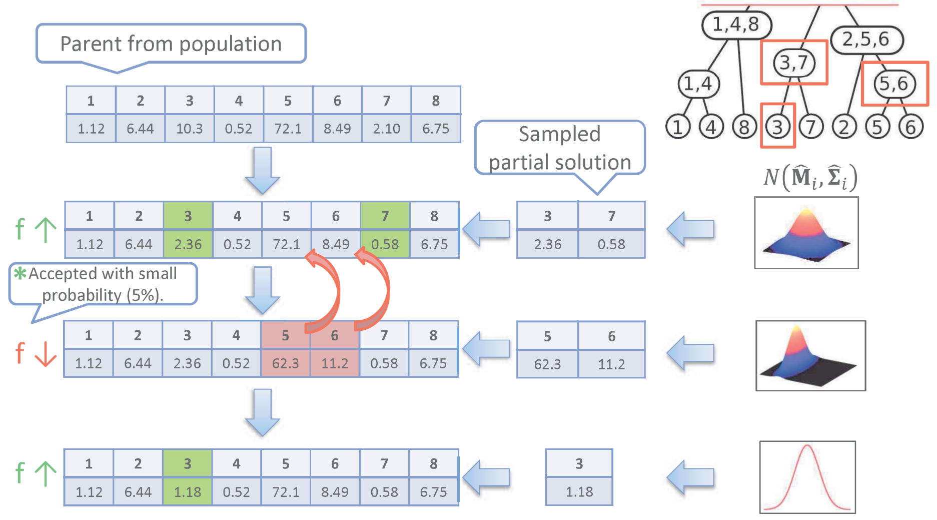


Image credit: Omidvar et al., 'Cooperative Co-Evolution With Differential Grouping for Large Scale Optimization.'

Gene-pool Optimal Mixing



Parent from population
Sampled partial solution
f ↑
f ↓
f ↑
*Accepted with small probability (5%).
1,4,8
2,5,6
3,7
1,4
5,6
1
4
8
3
7
2
5
6
$N(\hat{\mathbf{M}}_i, \hat{\boldsymbol{\Sigma}}_i)$

1	2	3	4	5	6	7	8
1.12	6.44	10.3	0.52	72.1	8.49	2.10	6.75
1.12	6.44	2.36	0.52	72.1	8.49	0.58	6.75
1.12	6.44	2.36	0.52	62.3	11.2	0.58	6.75
1.12	6.44	1.18	0.52	72.1	8.49	0.58	6.75

3	7
2.36	0.58

5	6
62.3	11.2

3
1.18

Gray-box Optimization

- **If**
  - **Sub-functions** of optimization fct. are known, e.g.:
    $f(x) = f(x_1, x_2) + f(x_2, x_3) + f(x_3, x_4) + f(x_4, x_5)$
  - Or otherwise known how to **update fitness "locally"**
- **Then**
  - **Partial evaluations** are possible.
    - Efficient update of fitness after modification of a subset of variables.

Partial Evaluations

$$x = [x_1\ x_2\ x_3\ x_4\ x_5]$$

$$f(x) = \boxed{f(x_1, x_2)} + \boxed{f(x_2, x_3)} + \boxed{f(x_3, x_4)} + \boxed{f(x_4, x_5)}$$

Partial Evaluations

$$x = [x_1\ x_2\ x_3\ x_4\ x_5]$$

$$f(x) = \boxed{f(x_1, x_2)} + \boxed{f(x_2, x_3)} + \boxed{f(x_3, x_4)} + \boxed{f(x_4, x_5)}$$

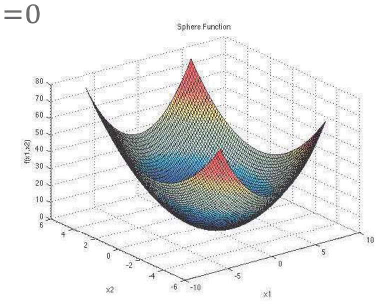
$$x' = [x_1\ x'_2\ x_3\ x_4\ x_5\ ]$$

$$f(x') = f(x) - \boxed{f(x_1, x_2)} - \boxed{f(x_2, x_3)} + \boxed{f(x_1, x'_2)} + \boxed{f(x'_2, x_3)}$$

Gray-box Optimization

- **Sphere function**:

$$f(x) = \sum_{i=0}^{\ell-1} x_i^2$$



- "Fully" **additively separable**

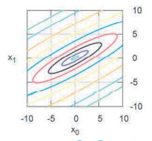
Gray-box Optimization

- **SOREB function**:

$$f(x) = \sum_{i}^{m-1} f_{Rot.Ell}(x_{ik}, x_{ik+1}, \dots, x_{ik+k-1})$$

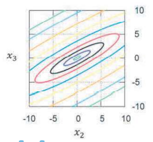
where m is **number of blocks**, k is **block length**

$$(1 \times (\cos(-0.5) \times x_0 - \sin(-0.5) * x_1))^2 + (5 \times (\sin(-0.5) \times x_0 + \cos(-0.5) * x_1))^2$$



$+$

$$(1 \times (\cos(-0.5) \times x_2 - \sin(-0.5) * x_3))^2 + (5 \times (\sin(-0.5) \times x_2 + \cos(-0.5) * x_3))^2$$



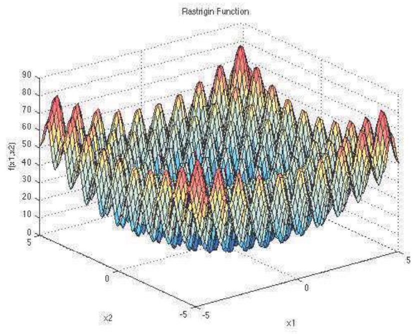
- Block-wise **additively separable**

Gray-box Optimization

- **Rastrigin function**:

$$f(x) = 10\ell + \sum_{i=0}^{\ell-1} x_i^2 - 10\cos(2\pi x_i)$$

- Also fully **additively separable**
- **Still hard!** Multi-modal problem.

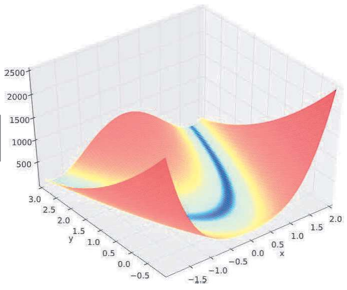


Gray-box Optimization

- **Rosenbrock function**:

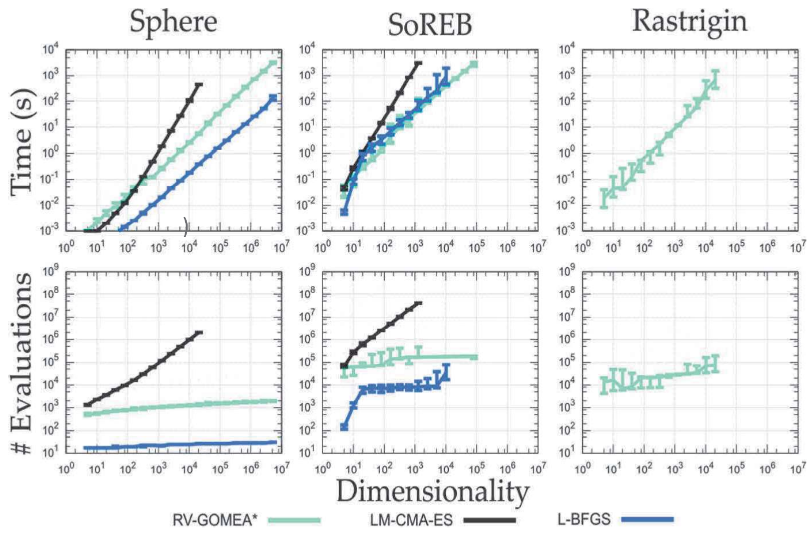
$$f(x) = \sum_{i=0}^{\ell-2} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2\right]$$

$$f^{GBO}(x) = \sum_{i=0}^{\ell-2} [f_i^{sub}(x_i, x_{i+1})]$$



- **Not separable**.
  - No decomposition into **disjoint** sets of variables.
- Partial evaluations can **still be applied!**
  - Modification of **1 variable** affects up to **2 sub-functions**.

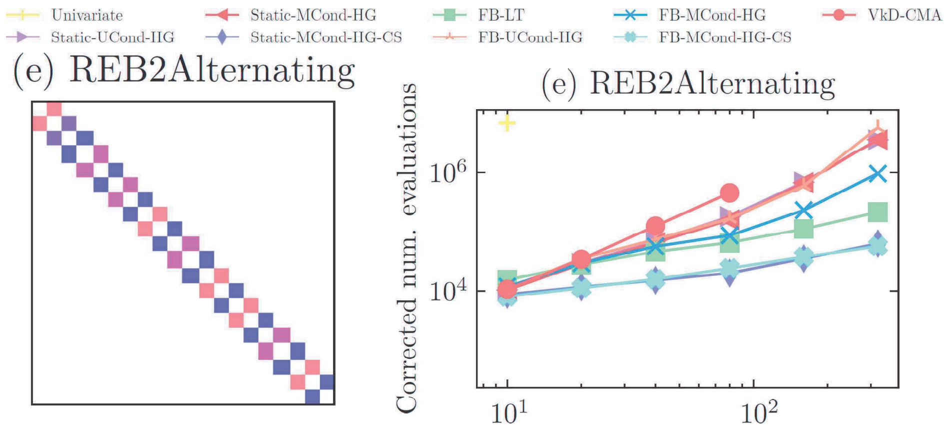
Gray-box Optimization - Scalability



(Bouter et al., Proc. GECCO, 2017; Bouter et al., Evolutionary Computation, 2021)

Take-home Message for RV

- Can combine with **online FB-learning and conditional (overlap) modelling** *(Andreadis et al., GECCO 2024)*



Take-home Message for RV

- **Linkage information** can be exploited for real-valued optimization, for instance with **optimal mixing** (RV-GOMEA)
- RV-GOMEA mostly effective in combination with **Gray-Box Optimization**

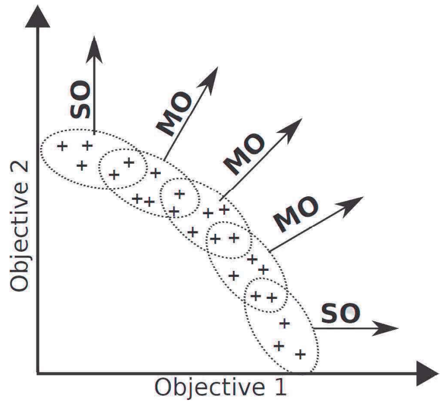
What about OMEA for GP and MO?

- What about the 2 remaining “generic” topics **GP** and **MO**?
- Standard **MO-GOMEA** is relatively **straightforward**:
  - **Cluster** population into k clusters in obj. space
  - Perform **learning** and **mixing** for each cluster separately

What about OMEA for GP and MO?

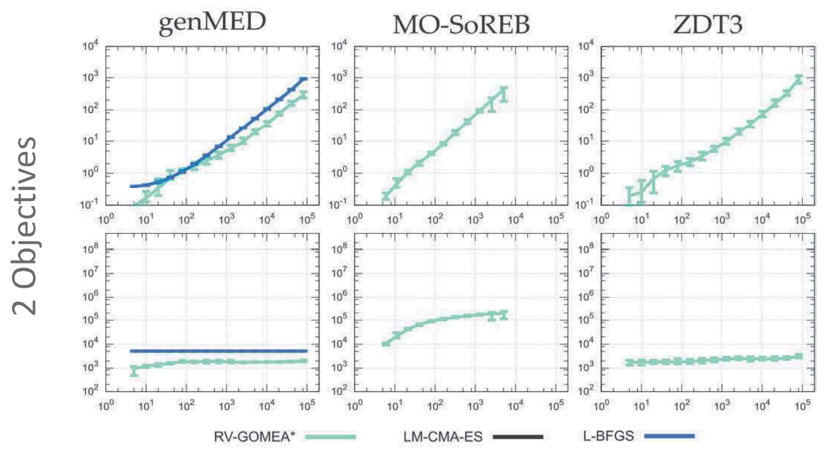
- For “extreme” clusters: use **single-objective OM**
- For other clusters: accept OM step based on **Pareto dominance** or **acceptance into Elitist archive**

(Luong et al, Proc. GECCO, 2014; Bouter et al, Proc. GECCO, 2017; Luong et al, SWEVO, 2017; Bouter et al, EVCO 2019)



What about OMEA for GP and MO?

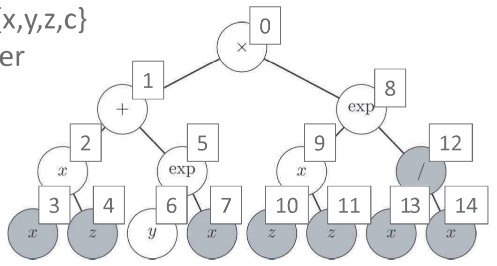
- MO-RV-GOMEA also scales **excellently**



(Bouter et al., Proc. GECCO, 2017; Bouter et al., Evolutionary Computation, 2021)

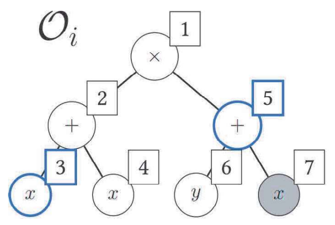
What about OMEA for GP?

- So, what about **GP**?
- Bit more involved, but **main principle same**
  - Basically, encode tree in a **fixed template**.
  - E.g., in case of operators of arity max. 2:
    - Function set, e.g.: {+,-,x,/,exp}
    - Terminal set, e.g.: {x,y,z,c}
    - Index terminals after functions, e.g.: genotype string 205574654277355 corresponds to: (grey nodes are **introns**)

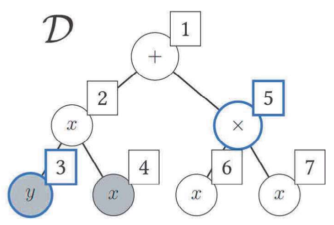


What about OMEA for GP?

- Now, we have **fixed-length strings** and we can do crossover/mixing and learn **dependencies**
- Note, with **FOS** concept (and use of LT learning) in GOMEA, can mix sets of nodes that are **not necessarily a subtree** together:



$F_j = \{3, 5\}$

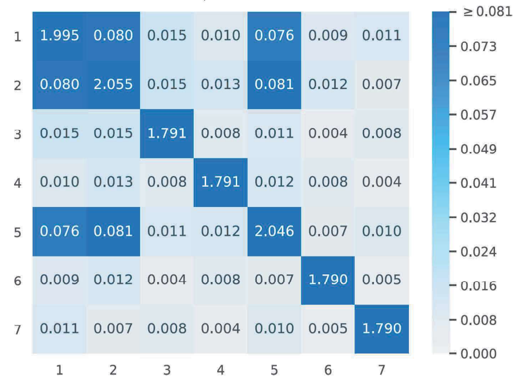


What about OMEA for GP?

- Key issue with **linkage learning**
  - Initially, expect **no linkage** in population (randomly initialized, no selection/fitness influence yet)
  - However, genes are **not uniformly randomly sampled**, typically
    - Leaf nodes can never be functions
    - Ramped half-and-half method often used for initialization
  - Leads to possible **spurious linkage**

What about OMEA for GP?

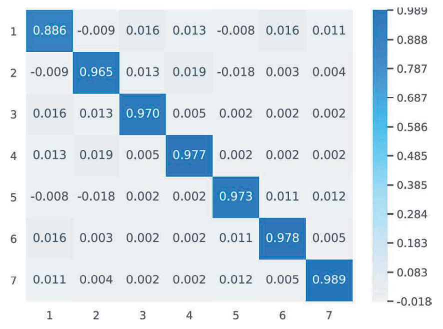
- Example **mutual information (MI) matrix** upon **initialization** (some pairs 10 × larger MI):



	1	2	3	4	5	6	7
1	1.995	0.080	0.015	0.010	0.076	0.009	0.011
2	0.080	2.055	0.015	0.013	0.081	0.012	0.007
3	0.015	0.015	1.791	0.008	0.011	0.004	0.008
4	0.010	0.013	0.008	1.791	0.012	0.008	0.004
5	0.076	0.081	0.011	0.012	2.046	0.007	0.010
6	0.009	0.012	0.004	0.008	0.007	1.790	0.005
7	0.011	0.007	0.008	0.004	0.010	0.005	1.790

What about OMEA for GP?

- **Adjust** MI matrix based on **initial sample**, so that matrix becomes **identity**
- Use adjustment factors **throughout run**
- Example MI matrix in generation 2:



	1	2	3	4	5	6	7
1	0.886	-0.009	0.016	0.013	-0.008	0.016	0.011
2	-0.009	0.965	0.013	0.019	-0.018	0.003	0.004
3	0.016	0.013	0.970	0.005	0.002	0.002	0.002
4	0.013	0.019	0.005	0.977	0.002	0.002	0.002
5	-0.008	-0.018	0.002	0.002	0.973	0.011	0.012
6	0.016	0.003	0.002	0.002	0.011	0.978	0.005
7	0.011	0.004	0.002	0.002	0.012	0.005	0.989

What about OMEA for GP?

- This and several other things incorporated into **GP-GOMEA**
- Different versions/variants:
  - Binary function learning
    (Virgolin et al., Proc. GECCO, 2017)
  - Symbolic regression
    (Virgolin et al., Evolutionary Computation, 2021)
  - Feature construction
    (Virgolin et al., Swarm and Evolutionary Computation, 2020)
  - Multi-objective Multi-modal Multi-tree
    (Sijben et al., Proc. GECCO, 2022)
  - Function **class** learning
    (Sijben et al., Proc. GECCO, 2024)
  - Higher-cardinality operators
    (Schlender et al., Proc. GECCO, 2024)

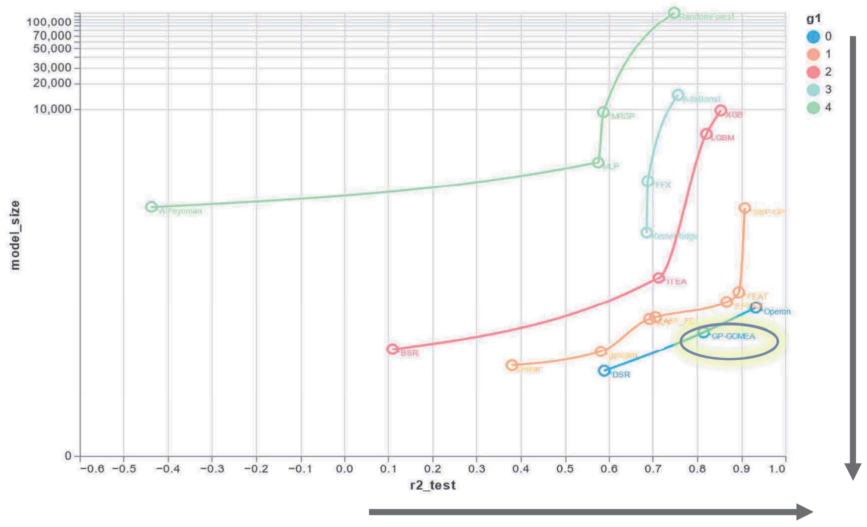
What about OMEA for GP?

- Mostly tailored at searching for **compact, small solutions** (because of template and OM)

- Since 2021 **benchmark** especially for symbolic regression exists, like BBOB for continuous variables: **SR-BENCH**

- We submitted **GP-GOMEA** in 2021

What about OMEA for GP?



Large-scale Parallelization Compatibility

- **GOMEA** has **massive speed-up** potential
  - **Parallellization**
    - **Population**: $\pm$ **factor n** (but holds for any EA)
    - **Linkage sets**: up to $\pm$ **factor ℓ** (for fully decomposable problem)
  - **Partial evaluations**
    - Typically, $\pm$ **factor ℓ**
  - Combined $\pm$ **factor $\ell^2 n$**, can have **huge** impact
  - Can even leverage **massive** #cores of **GPUs**

(Bouter et al., Proc. GECCO, 2018; Proc. CEC 2021; Proc. GECCO 2022)

Large-scale Parallelization Compatibility

Graphics Processing Unit (GPU)

Most suitable for 'parallel evaluation of single solution', in parallel for entire population.

Pros:

- High peak performance.

Cons:

- Restricted by SIMD programming.
- Overhead caused by data transfer.
- Requires large degree of parallelism.

Large-scale Parallelization Compatibility

Parallelization in Gray-Box Optimization (GBO)

- Large scale of parallelization required to utilize GPU capacity
  - Generally larger than population size.
  - Instead, 'Parallel evaluation of single solution' for each individual in population.
  - Requires known sub-functions, thus a GBO setting.

Large-scale Parallelization Compatibility

$$x = [\boxed{x_1}\ \boxed{x_2}\ \boxed{x_3}\ \boxed{x_4}\ \boxed{x_5}]$$

$$f(x) = \boxed{f_1(x_1,x_2)} + \boxed{f_2(x_2,x_3)} + \boxed{f_3(x_3,x_4)} + \boxed{f_4(x_4,x_5)}$$

$$x'' = [\boxed{x_1}\ \boxed{x_2'}\ \boxed{x_3}\ \boxed{x_4'}\ \boxed{x_5}]$$

Can be computed in parallel.

$$\Delta f' = -\boxed{f_1(x_1,x_2)} - \boxed{f_2(x_2,x_3)} + \boxed{f_1(x_1,x_2')} + \boxed{f_2(x_2',x_3)}$$

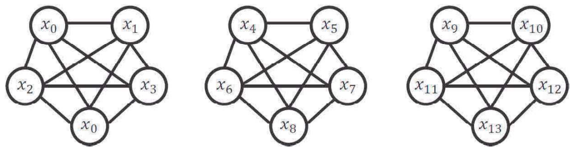
$$\Delta f'' = -\boxed{f_3(x_3,x_4)} - \boxed{f_4(x_4,x_5)} + \boxed{f_3(x_3,x_4')} + \boxed{f_4(x_4',x_5)}$$

$$f(x'') = f(x) + \Delta f' + \Delta f''$$

Large-scale Parallelization Compatibility

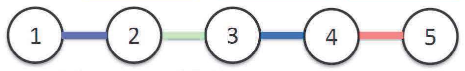
Variable Interaction Graph (VIG)

- Graph describing interactions/dependencies between problem variables.
- VIG: Undirected, unweighted graph $G = (V, E)$
  - V : set of vertices; 1 per problem variable
  - E : set of edges; $(u, v) \in E$ iff problem variables x_u and x_v are dependent (i.e., have a non-linear interaction).
- Example:
  VIG of deceptive trap (trap size 5)



Large-scale Parallelization Compatibility

- In a BBO setting, can be estimated using fitness-based dependency testing, e.g., differential grouping.
  - May require a large number of evaluations.
  - May not be able to identify all dependencies.
- In a GBO setting, can be estimated from problem definition.
- Edge between each pair of variables that are part of the input of the same subfunction.
- Example:
  - Optimization function: $f(x) = \boxed{f_1(x_1,x_2)} + \boxed{f_2(x_2,x_3)} + \boxed{f_3(x_3,x_4)} + \boxed{f_4(x_4,x_5)}$
  - Estimated VIG:



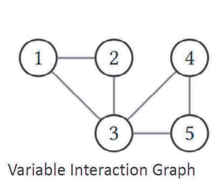
- 'Conservative estimate' of the true VIG
  - Variables used within the same subfunction may not always have a non-linear dependency.

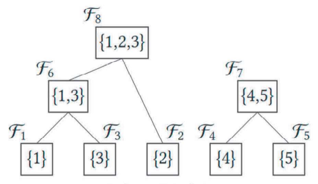
Large-scale Parallelization Compatibility

- Exploiting parallel partial evaluations in GOMEA:
  - Identify independent linkage sets.
  - Perform GOM (fitness-improving) variation steps in parallel for independent linkage sets.
- Linkage Model Interaction Graph (LMIG)
  - Shows dependencies between linkage sets.
  - Undirected, unweighted graph
  - $G = (V, E)$ with $(u, v) \in E$ if any variable in F_u is dependent on any variable in F_v.

Large-scale Parallelization Compatibility

Linkage Model Interaction Graph

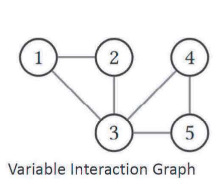

Variable Interaction Graph

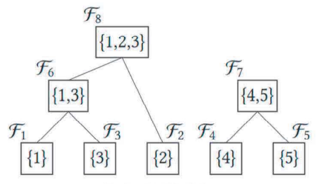

Linkage Model

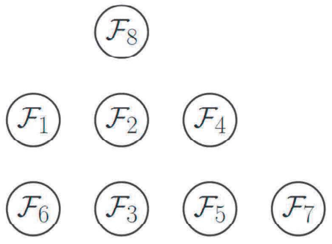
Large-scale Parallelization Compatibility

Linkage Model Interaction Graph

- One node per linkage set.


Variable Interaction Graph

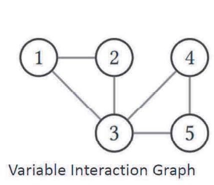

Linkage Model

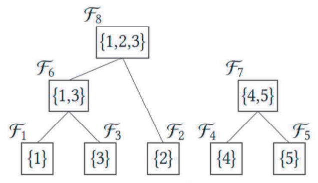

Linkage Model Interaction Graph

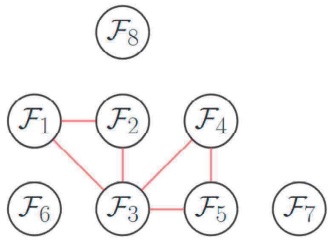
Large-scale Parallelization Compatibility

Linkage Model Interaction Graph

- One node per linkage set.
- Initial dependencies equal to VIG.


Variable Interaction Graph

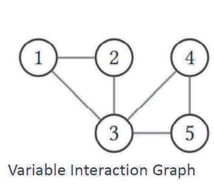

Linkage Model

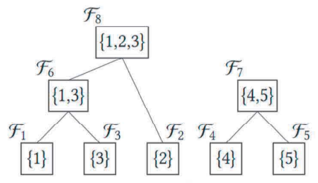

Linkage Model Interaction Graph

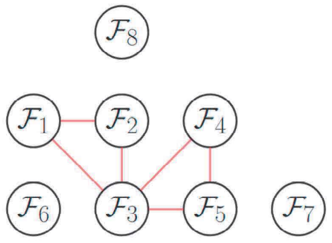
Large-scale Parallelization Compatibility

Linkage Model Interaction Graph

- One node per linkage set.
- Initial dependencies equal to VIG.
- Consider edges connecting F_6:


Variable Interaction Graph

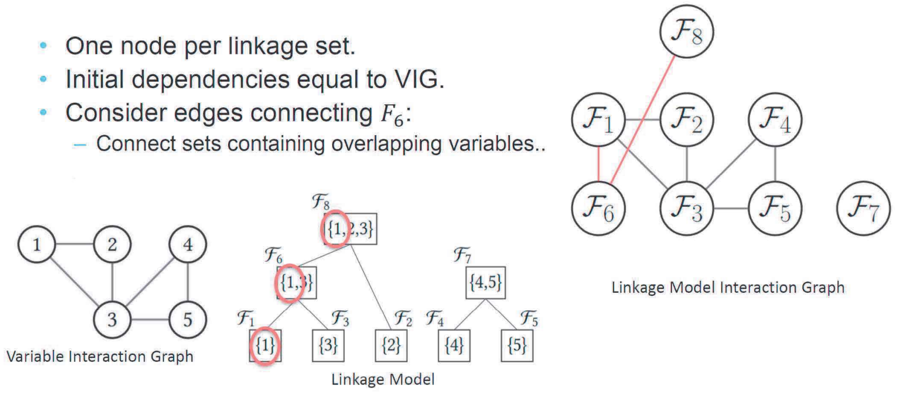

Linkage Model


Linkage Model Interaction Graph

Large-scale Parallelization Compatibility

Linkage Model Interaction Graph

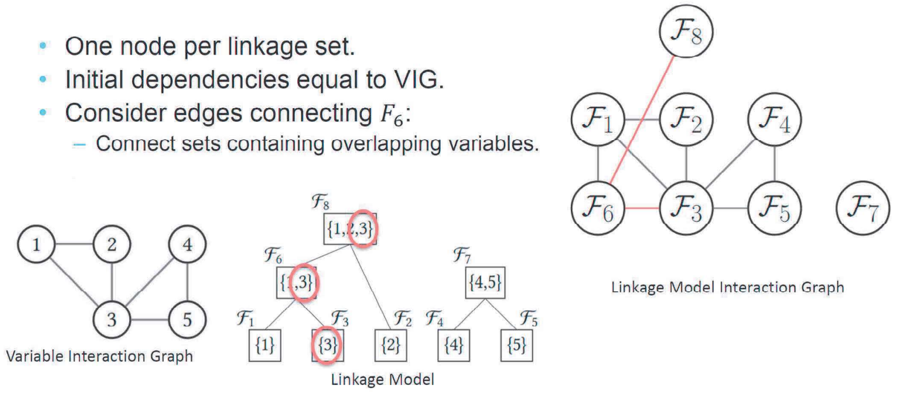
- One node per linkage set.
- Initial dependencies equal to VIG.
- Consider edges connecting F_6:
  - Connect sets containing overlapping variables..



Large-scale Parallelization Compatibility

Linkage Model Interaction Graph

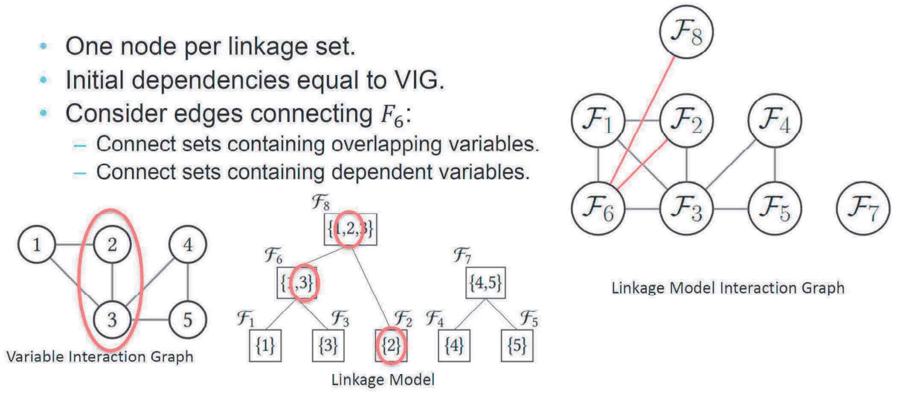
- One node per linkage set.
- Initial dependencies equal to VIG.
- Consider edges connecting F_6:
  - Connect sets containing overlapping variables.



Large-scale Parallelization Compatibility

Linkage Model Interaction Graph

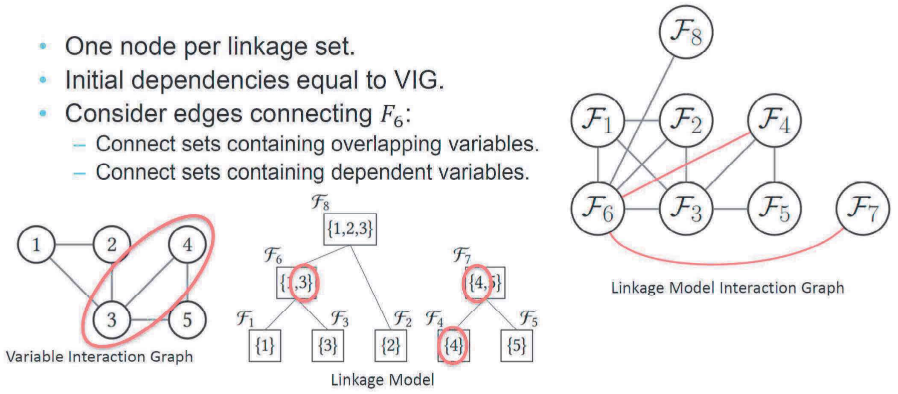
- One node per linkage set.
- Initial dependencies equal to VIG.
- Consider edges connecting F_6:
  - Connect sets containing overlapping variables.
  - Connect sets containing dependent variables.



Large-scale Parallelization Compatibility

Linkage Model Interaction Graph

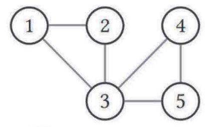
- One node per linkage set.
- Initial dependencies equal to VIG.
- Consider edges connecting F_6:
  - Connect sets containing overlapping variables.
  - Connect sets containing dependent variables.



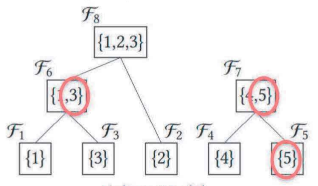
Large-scale Parallelization Compatibility

Linkage Model Interaction Graph

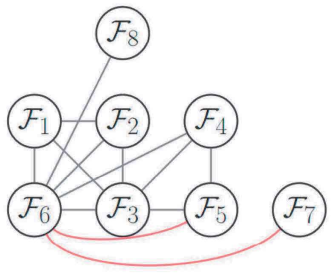
- One node per linkage set.
- Initial dependencies equal to VIG.
- Consider edges connecting F_6:
  - Connect sets containing overlapping variables.
  - Connect sets containing dependent variables.



Variable Interaction Graph



Linkage Model

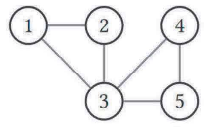


Linkage Model Interaction Graph

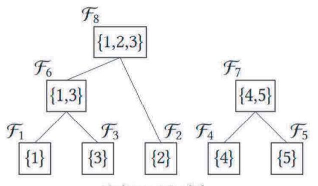
Large-scale Parallelization Compatibility

Linkage Model Interaction Graph

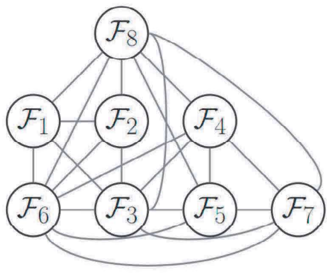
- One node per linkage set.
- Initial dependencies equal to VIG.
- Consider edges connecting F_6:
  - Connect sets containing overlapping variables.
  - Connect sets containing dependent variables.



Variable Interaction Graph



Linkage Model

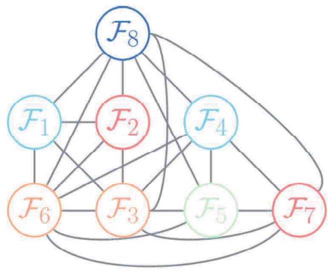


Linkage Model Interaction Graph

Large-scale Parallelization Compatibility

Parallel Gene-pool Optimal Mixing

- Linkage sets are independent if they are not connected in LMIG.
- Apply graph coloring to LMIG
  - Each linkage set with same color is mutually independent.
  - In each round of parallel GOM, apply GOM in parallel to all nodes (linkage sets) of the same color, for each individual in the population.



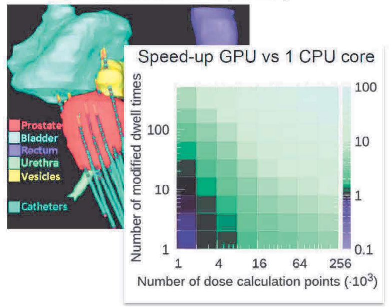
Linkage Model Interaction Graph

Large-scale Parallelization Compatibility

Applications of Large-Scale Parallel GBO

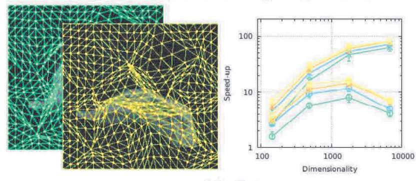
'Parallel evaluation of single solution':

Treatment plan optimization for Prostate HDR brachytherapy

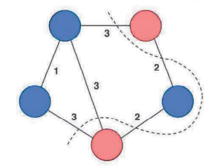


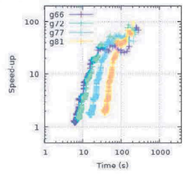
Parallel gene-pool optimal mixing:

Multi-Objective Deformable Image Registration



MaxCut





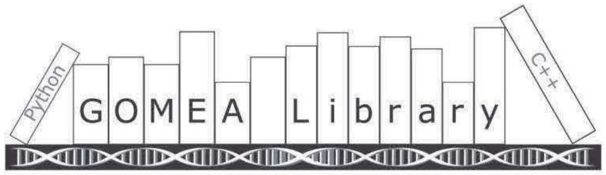
(Bouter et al, Medical Physics, 2019; Bouter et al, Proc. CEC, 2021; Bouter et al, Proc. GECCO, 2022; Andreadis et al, Proc. GECCO 2023)

Discussion

- Other (G)OMEA **developments**
  - GOMEA for **permutations**
    (Bosman et al, Proc. GECCO, 2016; Guijt et al, Swarm and Evolutionary Computation, 2022)
  - **Kernel** GOMEA for better locality exploitation
    (Guijt et al, Proc. GECCO, 2022)
  - GOMEA for discrete **expensive** optimization
    (Dushatskiy et al, Proc. GECCO, 2021)
  - Hypervolume-based GOMEA
    (Maree et al, Evolutionary Computation, 2022)
  - **Constraint handling** in GOMEA
    (Under construction)
  - …

Discussion

- **GOMEA library** *(SO binary & RV for now – more to follow)*
  - **Python** frontend (easily program your fitness functions in Python)
  - **C++** backend (run GOMEA efficiently)
  - Try it yourself, get it here:



https://github.com/abouter/gomea

Discussion

- A bit of a **GOMEA-only story**, but…
- **Parallels to be drawn**
  - Grey-box optimization & **partition crossover** by Darrell Whitley (discrete optimization)
  - **Cooperative co-evolution** by Omidvar (real-valued optimization)
- Key idea is the same: **exploit problem structure** (using a model to capture the structure in)

Take-away Message

- "Blind" metaheuristics 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

Essential MBEA questions

- Can problem structure be represented sufficiently?
- Can problem structure be represented efficiently?
- Can the model be learned from data correctly?
- Can the model be learned (and sampled) efficiently?