Model-Based Evolutionary Algorithms
Part 2: Linkage Tree Genetic Algorithm

Dirk Thierens

Universiteit Utrecht
The Netherlands
Evolutionary Algorithms

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

(Probabilistic) Model-Based Evolutionary Algorithms

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

FOS $\mathcal{F}$

- MBEAs learn a **model** of good solutions to match the **structure** of the optimization problem.
- Model does not have to be probabilistic.
- Key idea is to 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 processed as a block = **building block**.
- The dependency structure is a **set of subsets** of the problem variables.
Family Of Subsets (FOS) model

<table>
<thead>
<tr>
<th>FOS $\mathcal{F}$</th>
<th>Dependency structure generally called a <strong>Family Of Subsets</strong> (FOS).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Let there be $\ell$ <strong>problem variables</strong> $x_0, x_1, \ldots, x_{\ell-1}$.</td>
</tr>
<tr>
<td></td>
<td>Let $S$ be a set of all variable <strong>indices</strong> ${0, 1, \ldots, \ell-1}$.</td>
</tr>
<tr>
<td></td>
<td>A FOS $\mathcal{F}$ is a <strong>set of subsets</strong> of the set $S$.</td>
</tr>
<tr>
<td></td>
<td>FOS $\mathcal{F}$ is a <strong>subset</strong> of the <strong>powerset</strong> of $S$ ($\mathcal{F} \subseteq \mathcal{P}(S)$).</td>
</tr>
<tr>
<td></td>
<td>Example FOS $\mathcal{F}$ structures:</td>
</tr>
<tr>
<td></td>
<td>1. <strong>Univariate</strong> FOS structure</td>
</tr>
<tr>
<td></td>
<td>2. <strong>Marginal Product</strong> FOS Structure</td>
</tr>
<tr>
<td></td>
<td>3. <strong>Linkage Tree</strong> FOS Structure</td>
</tr>
</tbody>
</table>
Family Of Subsets (FOS) model

- FOS can be written more specifically as:

\[ \mathcal{F} = \{F^0, F^1, \ldots, F^{\lvert \mathcal{F} \rvert - 1}\} \]

where

\[ F^i \subseteq \{0, 1, \ldots, l - 1\}, \quad i \in \{0, 1, \ldots, \lvert \mathcal{F} \rvert - 1\} \]

- Every variable is in at least one subset in the FOS, i.e.:

\[ \forall i \in \{0, 1, \ldots, l - 1\} : \left( \exists j \in \{0, 1, \ldots, \lvert \mathcal{F} \rvert - 1\} : i \in F^j \right) \]
The Univariate Structure

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

- For \( l = 10 \) the univariate FOS is:
  \[ F = \{\{0\}, \{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{9\}\} \]

- Every variable is modeled to be independent of other variables.
The Marginal Product Structure

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

- 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 $F^i$ with more than one variable, there are subsets $F^j$ and $F^k$ such that:
  \[ F^j \cap F^k = \emptyset, \quad |F^j| < |F^i|, \quad |F^k| < |F^i| \quad \text{and} \quad F^j \cup F^k = F^i \]
- For $l = 10$ a possible LT FOS is
  \[ \mathcal{F} = \{\{7, 5, 8, 6, 9, 0, 3, 2, 4, 1\}, \]
  \[ \{7, 5, 8, 6, 9\}, \{0, 3, 2, 4, 1\}, \{7\}, \{5, 8, 6, 9\}, \]
  \[ \{0, 3, 2, 4\}, \{1\}, \{5, 8, 6\}, \{9\}, \{0, 3\}, \{2, 4\}, \]
  \[ \{5, 8\}, \{6\}, \{0\}, \{3\}, \{2\}, \{4\}, \{5\}, \{8\}\} \]
Problem variables in subset are considered to be dependent on each other but become independent in a child subset.

≈ Path through dependency space, from univariate to joint.

For a problem of length $\ell$ the linkage tree has $\ell$ leaf nodes (the clusters having a single problem variable) and $\ell - 1$ internal nodes.
Linkage Tree Learning

- Start from univariate structure.
- Build linkage tree using **bottom-up** hierarchical clustering algorithm.
- **Similarity** measure:
  1. Between individual variables $X$ and $Y$: mutual information $I(X, Y)$.
     
     $$I(X, Y) = H(X) + H(Y) - H(X, Y)$$

  2. Between cluster groups $X_{Fi}$ and $X_{Fj}$: average pairwise linkage clustering (= unweighted pair group method with a arithmetic mean: UPGMA).
     
     $$I^{UPGMA}(X_{Fi}, X_{Fj}) = \frac{1}{|X_{Fi}||X_{Fj}|} \sum_{X \in X_{Fi}} \sum_{Y \in X_{Fj}} I(X, Y).$$

$(H(X), H(Y), H(X, Y)$ are the marginal and joint entropies)
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.

Linkage Tree Learning
Reciprocal nearest neighbor chain clustering

- Construct a **nearest-neighbor chain**: start from an arbitrary start cluster, which is followed by its nearest neighbor, which is again followed by its nearest neighbor from among the remaining points, and so on.
- When the nearest neighbor for the last chain element $n$ is the element $n-1$, we stop with the generation with the NN chain, we have found a **RNN pair** and can **agglomerate** it.
- The nearest-neighbor assignments stay valid for the remaining chain members, which can thus be **reused** for the next iteration of agglomeration.
- If we do not find any more RNN pairs in the remaining NN chain, we create a new one starting from a random start point.
**Optimal Mixing Evolutionary Algorithms (OMEA)**

- **OMEA** is a Model-Building EA that uses a **FOS** as its linkage model (Thierens & Bosman, 2011).
- Characteristic of **Optimal Mixing Evolutionary Algorithm (OMEA)** is the use of **intermediate** function evaluations (inside variation)
- Can be regarded as **greedy improvement** of existing solutions
- Coined “**Optimal” Mixing** because **better** instances for substructures are **immediately accepted** and not dependent on “**noise”** coming from other parts of the solution
Optimal Mixing EA (GOMEA)

- **FOS** linkage models specify the linked variables.
- A subset of the FOS is used as **crossover mask**
- Crossover is **greedy**: only **improvements** (or **equal**) are accepted.
- Each generation a new FOS model is build from selected solutions.
- For each solution in the population, **all subsets** of the FOS are tried with a **donor** solution randomly picked from the population
- **Recombinative OM (ROM)** and **Gene-pool OM (GOM)**
  - ROM is GA-like: select **single** donor solution to perform OM with
  - GOM is EDA-like: select **new** donor solution for each subset in OM
Gene-pool Optimal Mixing EA

GOMEA()

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

GreedyRecomb(Sol, Donor, SubSet, Pop)

    NewSol ← ReplaceSubSetValues(Sol, SubSet, Donor)
    if ImprovementOrEqual(NewSol, Sol)
    then Sol ← NewSol
    return Sol
Recombinative Optimal Mixing EA

ROMEAEA()

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

GreedyRecomb(Sol, Donor, SubSet, Pop)

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

return Sol
Linkage Tree Genetic Algorithm

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

Interacting, non-overlapping, deceptive groups of variables.

\[ f_{DT}(x) = \sum_{i=0}^{l-k} f^{\text{sub}}_{DT}(x(i, \ldots, i+k-1)) \]
Nearest-neighbor NK-landscape

- **Overlapping**, neighboring random subfunctions

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

- eg. 16 subfcts, 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.
Experiments

- Benchmark functions: randomly linked deceptive trap function and randomly linked nearest-neighbor NK function.
- Compare GA, EDA, and GOMEA while each are learning the Marginal Product (MP) FOS structure, and GOMEA learning the Linkage Tree (LT) as FOS structure.
- Note:
  - EDA using MP = Extended Compact GA (ECGA).
  - GOMEA using LT = Linkage Tree Genetic Algorithm (LTGA).
Experiments

# Function Evaluations / Problem size

![Graphs showing function evaluations vs problem size for MP and LT models for different problem types and algorithms.](image)
Experiments

Minimal Population Size / Problem Size

![Graphs showing minimal population size and problem size](image-url)
Experiments

Runtime (seconds) / Problem Size

![Graphs showing runtime (seconds) vs problem size for different algorithms and landscapes.](image-url)
Experiments

**Figure:** LTGA vs. ILS: 100 NK problems

Iterated Local Search: perturbation size each time randomly picked between 2 and 10 bits (= better than any fixed value).
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.
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 the most efficient FOS structure

Dynamically learned tree model (often) more efficient than predetermined, fixed FOS models that mirror the problem structure
Conclusions

- “Blind” Evolutionary Algorithms are limited in their capability to detect and mix/exploit/re-use partial solutions (building blocks).
- Requires knowledge, ‘luck’ or analysis and design of problem structure and exploit this in the problem representation and search operators.
- Having a configurable model can help overcome this.
- Algorithm then must learn a model of dependencies and exploit structure online during optimization (e.g. EDAs, OMEAs).