Motivation

Estimation of Distribution Algorithms (EDAs)
Probabilistic Model-Building Genetic Algorithms (PMBGAs)
Iterated Density-Estimation Evolutionary Algorithms (IDEAs)

- modern form of evolutionary algorithm
- solve problem classes where standard GAs fail
- trajectories
Road Map

EDA Principles

Advanced EDAs

Comparison with GAs

Billion-Variable EDA

Descriptive Probability Models

Linkage Learning

Disruptive Crossover

Case Study
Quick Review of Probability I

7 random observations of my state of mind

if \( X \) is a random variable representing my state of mind, can estimate its distribution as:

\[
\begin{align*}
\mathbb{P}(X = \text{happy}) &= \frac{4}{7} \\
\mathbb{P}(X = \text{sad}) &= \frac{3}{7}
\end{align*}
\]
Conditional Probability

- if $D$ is the day,
  
  $\mathbb{P}(X = \text{happy}|D = \text{Monday}) = \frac{1}{3}$
  
  $\mathbb{P}(X = \text{sad}|D = \text{Monday}) = \frac{2}{3}$

- enables a more ‘refined’ model

- conditional probability can be calculated using:

  $\mathbb{P}(X = x|D = d) = \frac{\mathbb{P}(X = x, D = d)}{\mathbb{P}(D = d)}$

<table>
<thead>
<tr>
<th>Day</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday</td>
<td>sad</td>
</tr>
<tr>
<td>Monday</td>
<td>sad</td>
</tr>
<tr>
<td>Friday</td>
<td>happy</td>
</tr>
<tr>
<td>Friday</td>
<td>happy</td>
</tr>
<tr>
<td>Friday</td>
<td>sad</td>
</tr>
<tr>
<td>Friday</td>
<td>happy</td>
</tr>
<tr>
<td>Friday</td>
<td>happy</td>
</tr>
</tbody>
</table>
Estimation of Distribution Algorithm Process

- Selection
- Estimation
- Initialisation
- Termination
- Generation
- Probability
  - Model
  - Distribution
A Simple Example - Configuration

Genome (Representation)

4 genes \( (X_i, \ i = 0, 1, 2, 3) \); each gene is either A or B

Probability Model

Assume each gene is independent

Probability Distribution

\[(p_0, p_1, p_2, p_3)\]

where \( P(X_i = A) = p_i \) and thus \( P(X_i = B) = 1 - p_i \)
A Simple Example - Process

Initialisation

(0.5, 0.5, 0.5, 0.5)

Termination

Estimation

Generation

Selection
A Simple Example - Process

Initialisation

Estimation

(0.5, 0.5, 0.5, 0.5)

Termination

Generation

Selection
A Simple Example - Process

Initialisation

Selection

Estimation

Termination

Generation

(1, 0.25, 0.75, 0.75)
A Simple Example - Process

Initialisation

Estimation

Selection

Generation

Termination

(1, 0.25, 0.75, 0.75)
A Simple Example - Process

Initialisation → Estimation → Selection → Generation → Termination

(0, 1, 2, 3) → (1, 0.25, 0.75, 0.75)
A Simple Example - Process

Initialisation

Termination

(1, 0, 1, 1)

0 1 2 3

Estimation

Generation

Selection
A Simple Example - Process

Initialisation

(1, 0, 1, 1)

Generation

Estimation

Selection

Termination

(1, 0, 1, 1)

Generation
**Initialisation**

Initially, don’t know distributions of A and B in best solutions, so assume equally likely: \( p_i = 0.5 \).

**Generation**

Could use the following method to pick the value of each gene, \( X_i \):

1. pick a (uniformly distributed) random number, \( \gamma \), between 0 and 1
2. if \( \gamma \leq p_i \), then set \( X_i \) to A, otherwise to B

Note: The values in the generated population will match the distribution closely, but not necessarily exactly.
A Simple Example - Key Points II

Selection
Can use same selection methods as for standard GAs, e.g. proportional selection (roulette wheel).

Estimation
In this example, simply count the number of As for gene $X_i$ and divide by the number of individuals to give $p_i$.

Termination
Sensible criterion is for all $p_i$ to be either 0 or 1.
Note: The solution is ABAA; it is not (1,0,1,1). The latter is the probability distribution at termination.
EDAs as GAs with Variance Operator
EDAs as GAs with Variance Operator
This model is used by the following EDAs (although the algorithm itself differs slightly):

- Univariate Marginal Distribution Algorithm (UMDA) [Mühlenbein and Paaß, 1996]
- Population-Based Incremental Learning (PBIL) [Baluja, 1994]
- Compact Genetic Algorithm (cGA) [Harik et al., 1999]

But . . .

Is the assumption of independent probability distributions for each gene an oversimplification?
Road Map

EDA Principles

Advanced EDAs

Comparison with GAs

Billion-Variable EDA

Descriptive Probability Models

Linkage Learning

Disruptive Crossover

Case Study
Probability distribution for a gene depends on (conditional on) the value of other genes.

Example

- distribution of $X_1$ is independent (as before)
- but, distribution of $X_3$ depends on value of $X_1$
- distribution of $X_2$ depends on value of $X_3$
- distribution of $X_0$ depends on values of $X_1$ and $X_2$

(arrows go from parent(s) to dependent child)

Need to order genes appropriately in order to generate from, and estimate, the distribution.
Estimation Example

\[ P(X_1 = A) = 0.5 \]
\[ P(X_0 = A | X_1 = A) = 0.5 \]
\[ P(X_0 = A | X_1 = B) = 1 \]

and so \[ P(X_1 = B) = 0.5 \]
and so \[ P(X_0 = B | X_1 = A) = 0.5 \]
and so \[ P(X_0 = B | X_1 = B) = 0 \]
Conditional Probability Model Calculations II

Generation Example

- \( P(X_1 = A) = 0.5 \)
- \( P(X_0 = A | X_1 = A) = 0.5 \)
- \( P(X_0 = A | X_1 = B) = 1 \)

1. randomly pick \( \gamma_1 \) between 0 and 1, say \( \gamma_1 = 0.428 \ldots \)
2. since \( \gamma_1 \leq P(X_1 = A) \), set \( X_1 \) to A
3. now pick \( \gamma_0 \) between 0 and 1, say \( \gamma_0 = 0.732 \ldots \)
4. since \( \gamma_0 > P(X_0 = A | X_1 = A) \), set \( X_0 \) to B
5. so in our generated individual, \( X_0 = B, X_1 = A \)
Probability distributions considered for a subset of genes taken as a whole.

**Example**

for each subset, need to store probability of all combinations, e.g.:

\[
\begin{align*}
\mathbb{P}(X_1 = A, X_3 = A) \\
\mathbb{P}(X_1 = A, X_3 = B) \\
\mathbb{P}(X_1 = B, X_3 = A) \\
\mathbb{P}(X_1 = B, X_3 = B)
\end{align*}
\]
Why Use More Complex Models?

- Better able to model structure of underlying problem in terms of the relationship between genes
- Processing for estimating and generating from a more complex model is not usually significant compared to fitness evaluation
- Factorised Distribution Algorithm (FDA) [Mühlenbein, Mahning, and Rodriguez, 1998] uses a predefined model using conditional probability and subsets

But ...

Is is realistic that we define structure of probability model for problems in general?
So far, examples have used a predefined probability model that stays the same throughout the algorithm.

Many powerful EDAs ‘learn’ the probability model as they go.

Often the probability model is derived during the estimation step of each generation.
To be able to choose from all possible models, need to have a measure of how good a particular model is at representing the selected population.

Examples of metrics include:
- Bayesian Dirichlet metric
- Kullback-Leibler divergence
- Pearson’s chi-square statistic
- minimum description length
Deriving the Model

Given a metric, a possible method of deriving the model from the selected population is the following greedy algorithm:

1. Assume no connections (all genes independent)

Consider all valid operations on the model (e.g., adding a link from a parent to a child).

If no operation improves the metric, stop.

Otherwise, perform the operation that improves the metric the most.

Repeat from step (2).
Given a metric, a possible method of deriving the model from the selected population is the following greedy algorithm:

1. assume no connections (all genes independent)
2. consider all valid operations on the model (e.g. adding a link from a parent to a child)
Given a metric, a possible method of deriving the model from the selected population is the following greedy algorithm:

1. assume no connections (all genes independent)
2. consider all valid operations on the model (e.g. adding a link from a parent to a child)
3. if no operation improves the metric, stop
4. otherwise perform the operation that improves the metric the most
Given a metric, a possible method of deriving the model from the selected population is the following greedy algorithm:

1. assume no connections (all genes independent)
2. consider all valid operations on the model (e.g. adding a link from a parent to a child)
3. if no operation improves the metric, stop
4. otherwise perform the operation that improves the metric the most
5. repeat from step (2)
Examples of Linkage Learning EDAs I

Mutual Information Maximizing Input Clustering (MIMIC)

De Bonet et al., 1997

Bivariate Marginal Distribution Algorithm (BMDA)

Pelikan and Mühlenbein, 1999
Examples of Linkage Learning EDAs II

**Extended Compact Genetic Algorithm (ECGA)**

Harik, 1999

**Bayesian Optimization Algorithm (BOA)**

Pelikan, Goldberg and Cantú-Paz, 2000
Building Blocks

- A *schema* is a bit pattern template using the alphabet \( \{0,1,*\} \) where * is a wildcard.
- *Defining length* is the distance between the first and last non-wildcard symbols.
- *Order* is the number of non-wildcard symbols.

**Example**

Schema: \( H = *10* \)
- Representatives: 0100, 0101, 1100, 1101
- Defining length: \( \delta(H) = 1 \)  
  Order: \( o(H) = 2 \)

- *Building blocks* are short, low order, highly fit schemata.
- GAs work well when building blocks propagate through the population and are combined to produce fit individuals.
Some crossover operators can disrupt building blocks.

Example - One-Point Crossover

```
0 1 0 0
1 0 1 1
```
Disruptive Crossover

Some crossover operators can disrupt building blocks.

Example - One-Point Crossover

```
0 1 0 0
1 0 1 1
1 0 1 1
0 0 1 1
```
Some crossover operators can disrupt building blocks.

Example - One-Point Crossover

```
0 1 0 0
1 0 1 1

↓
1 0 1 0
0 1 0 1
```
Some crossover operators can disrupt building blocks.

Example - One-Point Crossover

```
0 1 0 0
1 0 1 1
1 0 0 0
0 1 1 1
```
Case Study - Additive Deceptive Function

Genome

<table>
<thead>
<tr>
<th>X₀</th>
<th>X₁</th>
<th>X₂</th>
<th>X₃</th>
<th>X₄</th>
<th>X₅</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Fitness

\[ f = g(X₀, X₁, X₂) + g(X₃, X₄, X₅) \]

where \( g(\cdot) \) is:

![Graph showing fitness function with peaks at 0 and 3 ones, and a local optimum at 000000.]

Global optimum is clearly 111111, but deceptive nature of \( g(\cdot) \) tends to move population towards local optimum at 000000.
Results Using Standard GA

Figure: proportion of population having schemata 111*** and 000*** at each generation; average over 10 runs

- population size 1000
- one-point crossover (with probability 1)
- no mutation
- fitness proportional selection
Hypothesis

- Schemata with 2 ones are quickly eliminated from population.
- Crossover between 111 and other schemata is more often destructive than not.
- Crossover between schemata is unlikely to produce 111.
- Therefore, schemata with few ones begin to dominate.
- Since 000 is the fitter of the few ones schemata, algorithm eventually converges to this solution.
Figure: proportion of population having schemata 111*** and 000*** at each generation; average over 10 runs

- population size 1000
- fitness proportional selection
- predefined model
Hypothesis 1

- In initial random population, individuals where $X_0 = 0$ are on average fitter than $X_0 = 1$
- So schemata with $X_0 = 0$ occur more frequently in each new generation
Hypothesis II

- by selection over a number of generations, algorithm then establishes probability distribution for model:
  - given $X_0 = 0$, fitter individuals occur when $X_1 = 0$ and $X_2 = 0$
  - given $X_0 = 1$, fitter individuals occur when $X_1 = 1$ and $X_2 = 1$

- so probability distribution now results in generation of schemata 000 and 111 more often than others

- when this occurs, individuals where $X_0 = 0$ are now on average less fit than $X_0 = 1$

- so 111 schema begins to dominate, and algorithm converges on this solution
Towards Billion Bit Optimization via Efficient Genetic Algorithms
Kumara Sastry, David E Goldberg, Xavier Llorà

IlliGAL Report No. 2007007
Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign

Best EDA paper award at Genetic and Evolutionary Computation Conference (GECCO) 2007
### Problem - Noisy, OneMax

<table>
<thead>
<tr>
<th>Representation</th>
<th>10⁹ variables $x_i \in {0, 1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Objective</td>
<td>Optimal solution has all $x_i = 1$ (‘OneMax’)</td>
</tr>
<tr>
<td>Fitness</td>
<td>$f = \sum x_i + \mathcal{N}(0, \sigma^2)$</td>
</tr>
</tbody>
</table>
Solution Method

Algorithm

Compact Genetic Algorithm (CGA) - a univariate EDA

Implementation

Processor A

Processor B
Extrapolation from Trajectory

For $10^9$ variables, algorithm would take too long to converge even on large parallel computing cluster.

Measured time for algorithm to reach point where all probabilities were $> 0.501$ (from initial probability of 0.5). Extrapolated results from small problems where full convergence was possible.

Novelty

- Real-world problem size
- Very efficient parallel implementation of CGA
- Although simple EDA, superior (more scalable) to other approaches such as hill-climbing on this problem
EDAs are a modern form of evolutionary algorithm.

Wide variety of algorithms ranging from simple (e.g. CGA) to advanced, state-of-the-art (e.g. BOA).

Demonstrate advantages over standard GAs on some problem classes.
Selected Resources

Survey of Bit-String EDAs
Martin Pelikan, David Goldberg and Fernando Lobo
A Survey of Optimization by Building and Using Probabilistic Models
IlliGAL Report No. 99018, University of Illinois, 1999

Missouri Estimation of Distribution Algorithms Laboratory
http://medal.cs.umsl.edu/