Discovering the preference hypervolume: an interactive model for real world computational co-creativity
Hagg, A.

Citation

Version: Publisher's Version
License: Licence agreement concerning inclusion of doctoral thesis in the Institutional Repository of the University of Leiden
Downloaded from: https://hdl.handle.net/1887/3245521

Note: To cite this publication please use the final published version (if applicable).
Evolutionary Algorithms

This appendix is based on previous published work by Asteroth and Hagg (2015). In EA, an approximative solution to an optimization problem is iteratively found. In each iteration, a set of solution candidates, called a population is maintained in a stochastic procedure. This procedure consists of 4 basic steps (see Figure A.1):

1. *evaluation* – assignment of a real valued fitness to the candidates
2. *selection* – survival of the fittest candidates
3. *crossover* – recombination to produce offspring
4. *mutation* – randomized changes to the individual offspring

*Figure A.1:* Four basic steps of an EA.

In this repeated procedure the balance between *exploitation*, the refinement of
A. EVOLUTIONARY ALGORITHMS

individuals with high fitness, and exploration, the search for new solutions, possibly through a part of the search space that has lower fitness, plays a crucial role.

Members of the population are called individuals (see Figure A.2). Each is represented by its genotype, which here can be considered a one-dimensional array. The array’s cells are called loci and their value a gene. All genes of the individual make up its genome, an instance of the genotype. Before the individual is assigned a fitness the genome is first mapped to its phenotype (1:1 or n:1 mappings are possible).

Individuals from the current population survive and breed based on their fitness and therefore the fitness function must provide sufficient information to direct the search towards an optimum. Formally, the fitness function is a function from phenotype space $P$ to the positive real numbers $\mathbb{R}^+$.

$$f : P \rightarrow \mathbb{R}^+$$

While in classical optimization algorithms, search is driven by the gradient of an error function, in EA a gradient is not necessary but fitness must still represent some distance to the solution.

The exploration rate plays a crucial role for the convergence speed and approximation quality. During the selection process, exploration happens if individuals with submaximal fitness survive. If on the other hand exploitation is high, the search will converge very fast but will get stuck in local extrema. This problem does not occur when exploration is high, but the convergence speed will then be much lower.
A.1 Selection

A balance between refining good individuals and nurturing promising ones has to be found. Good solutions are often combinations of building blocks found in less fit individuals, therefore diversity plays a central role. On the other hand, selection must assure the survival of the fittest individuals, otherwise, the search will be more or less undirected. Common search strategies include:

- *fitness proportionate* selection, such as roulette wheel selection or stochastic universal sampling
- *rank-based* selection, e.g. linear ranking or tournament selection

A.2 Crossover

A new population of children is created by recombining the selected individuals. This breeding process is done by crossover of genomes. The process is highly dependent on the genotype, the representation of the individuals. Simple strategies for crossover include:

- *N-point crossover:* two genomes form two children. Between crossover-points, genes are taken from alternating parents, as shown in Figure A.3
- *uniform crossover:* for all loci, choose a gene from either parent with a given probability (usually dependent on the fitness)
- *arithmetic crossover:* calculate the offspring’s genes by arithmetic combination of the parents genes

Usually two parents are chosen for crossover, though it is possible to create offspring from more parents. A *crossover rate* (CR) defines the probability of an individual to undergo crossover, otherwise it is copied to the next generation.

A.3 Mutation

After individuals are selected as parents and recombined by crossover, there is a certain probability for *mutation* of the offspring. For binary genes it is an obvious
option to flip values with a certain probability. For real valued genes a random value can be drawn from some distribution (e.g. Gaussian) and added to the gene. In many cases it is better to permutate a genome. For example, if the genome encodes a TSP tour it is better to permutate two cities than to randomly mutate one city because by random mutation the vast majority of variants will become invalid.

The idea behind mutation is to introduce new information into the population. Thus a high mutation rate corresponds to a high exploration rate. A balance between exploration and exploitation has to be found.

Another important secondary parameter is the mutation distance $s$. This distance represents the strength (and therefore also amount of disruption) a single mutation can provoke. In case of a Gaussian mutation on a real number, this distance would be controlled by changing the $\sigma$ of the underlying distribution.

### A.4 Diversity Management

Precautions must be taken to prevent premature convergence and ensure exploration to take place even after convergence to local suboptimal extrema. In particular if the search space contains disconnected regions, it is necessary to sustain diversity in the population. Techniques to implement this include

- *niching*, in crowded regions the probability to reproduce is lower
**A.5 Representation**

- *speciation*, only “similar” solutions are recombined

Niching prevents all individuals from populating only the region around one particular extremum by fitness sharing.

\[
\text{fitness}_{\text{shared}} = \frac{\text{fitness}}{|\text{close individuals}|}
\]

Niching allows for dissimilar solutions to be recombined. While this is desired since it ensures exploration it usually results in less fit individuals if two solutions from dissimilar regions/optima are combined. A solution to this problem is *speciation* which allows only similar individuals to be recombined. This can be implemented by dividing the population into species (using some metric). Each species is assigned a number of children according to their fitness and these are created solely from individuals of the corresponding species.

**A.5 Representation**

The representation of a possible solution is probably the most important deciding factor, whether an EA will find a solution quickly or at all. The fitness of an individual must represent the distance to a solution in some way or other. For mutation to work best, minor changes in the genome should result in minor changes to the fitness value, otherwise the search conducted by an EA will be rather erratic and convergence will be slow.
**T-Distributed Stochastic Neighborhood Embedding**

Commonly used for visualization, the dimensionality reduction method t-SNE has been shown to be capable of retaining the local structure of the data, as well as revealing clusters at several scales. It does so by finding a lower-dimensional distribution of points $Q$ that is similar to the original high-dimensional distribution $P$. The similarity of data point $x_j$ to datapoint $x_i$ is the conditional probability ($p_{j|i}$ for $P$ and $q_{j|i}$ for $Q$, Eq. B.1), that $x_i$ would pick $x_j$ as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian distribution centered at $x_i$. The Student-t distribution is used to measure similarities between low-dimensional points $y_i \in Q$ in order to allow dissimilar objects to be modeled far apart in the archive (Eq. B.1).

$$p_{j|i} = \frac{e^{-\frac{\|x_i-x_j\|^2}{2\sigma_i^2}}}{\sum_{k\neq i} e^{-\frac{\|x_i-x_k\|^2}{2\sigma_i^2}}}, \quad q_{j|i} = \frac{1 + \|y_i - y_j\|^2}{\sum_{k\neq i} (1 + \|y_i - y_k\|^2)^{-1}}$$ (B.1)

The local scale $\sigma_i$ is adapted to the density of the data (smaller in denser parts). The parameter $\sigma_i$ is set such that *perplexity* of the conditional distribution equals a predefined value. The perplexity of a distribution defines how many neighbors for each data point have a significant $p_{j|i}$ and can be calculated using the Shannon entropy $H(P_i)$ of the distribution $P_i$ around $x_i$ (Eq. B.2).

$$\text{Perp}(P_i) = 2^{-\sum_j p_{j|i}\log_2 p_{j|i}}$$ (B.2)  

$$KL(P \| Q) = \sum_{i \neq j} p_{ij} \log \left( \frac{p_{ij}}{q_{ij}} \right)$$ (B.3)
Using the bisection method, $\sigma_i$ are changed such that $\text{Perp}(P_i)$ approximates the preset value (commonly 5–50). The similarity of $x_j$ to $x_i$ and $x_i$ to $x_j$ is absorbed with the joint probability $p_{ij}$. A low-dimensional archive is learned that reflects all similarities $p_{ij}$ as well as possible. Locations $y_i$ are determined by iteratively minimizing the Kullback-Leibler divergence of the distribution $Q$ from the distribution $P$ (Eq. B.3) with gradient descent.
Gaussian Process Regression

To perform interpolation or regression on a given data set, GP models (introduced by Rasmussen (2004)) assume that the underlying data is sampled from a Gaussian process – a process that generates points that are distributed in a Gaussian fashion, and are correlated in a local and smooth fashion. For an in-depth introduction to GP regression and its application in model-based optimization, refer to Forrester et al. (2008). What follows is a summarized explanation.

The models assume that the objective function is smooth: the closer a candidate is to a known example, the closer their function values will be to each other. Here, the training data of the model is denoted as a set of $n$ solutions $X = \{x^{(i)}\}_{i=1...n}$ in a $k$-dimensional search space. The corresponding $n$ observations are denoted with $y = \{y^{(i)}\}_{i=1...n}$. For an unknown point in our search space, $x^*$, Gaussian process regression intends to estimate the unknown function value $\hat{y}(x^*)$. In its core, the model assumes that the observations at each location $x$ are correlated via a kernel function. Kernel functions of the following type are considered here:

$$k(x, x') = \exp \left( -\theta d(x, x') \right). \quad (C.1)$$

This essentially expresses the correlation of two samples $x$ a $x'$, based on their distance $d(x, x')$, and a kernel parameter $\theta \in \mathbb{R}^+$. Kernel parameters are usually determined by Maximum Likelihood Estimation (MLE), that is, they are chosen such that the data has the maximum likelihood under the resulting model. MLE usually involves a numerical optimization procedure. The distance measure $d(x, x')$ can potentially be any measure, though not all ensure that the kernel is positive semi-definite, a common requirement. By using the Manhattan distance, the
distance measure is less affected by issues related to high-dimensional data, see Aggarwal et al. (2001). This distance is defined as:

\[ d_{\text{Man}}(x, x') = \sum |x_i - x'_i| \quad \text{(C.2)} \]

Rather than a single parameter \( \theta \), a different \( \theta \) can be used for each dimension \( i \) of the input samples, enabling the model to estimate the influence of each individual dimension on the observed values. However, in the interest of simplicity and computational efficiency we opt for an isotropic kernel with a single \( \theta \).

Once the pairwise correlations between all training samples are collected in a matrix \( K \), the GP predictor can be specified with

\[ \hat{y}(x^*) = \hat{\mu} + k^T K^{-1} (y - 1\hat{\mu}), \quad \text{(C.3)} \]

where \( \hat{\mu} \) is another model parameter (estimated by MLE), \( k \) is the vector of correlations between training samples \( X \) and the new sample \( x^* \), and \( 1 \) is a vector of ones. The error or uncertainty of the prediction can be estimated with

\[ \hat{s}^2(x) = \hat{\sigma}^2 (1 - k^T K^{-1} k^T), \quad \text{(C.4)} \]

where \( \hat{\sigma}^2 \) is a further model parameter to be estimated by MLE.
Bibliography


Association for Computational Creativity (“2019 (accessed October 2, 2020)”). https://computationalcreativity.net/home/about/computational-creativity.


Jung, C. (1923). Psychological types: or the psychology of individuation.


BIBLIOGRAPHY


Shir, O. M., M. Preuss, B. Naujoks, and M. Emmerich (2009). Enhancing decision
space diversity in evolutionary multiobjective algorithms. Lecture Notes in
Computer Science (including subseries Lecture Notes in Artificial Intelligence
and Lecture Notes in Bioinformatics) 5467 LNCS, 95–109.

Monotonic Improvements versus Nonmonotonic Variants. Creativity Research

exploration through hierarchical spatial partitioning. Lecture Notes in Computer
Science (including subseries Lecture Notes in Artificial Intelligence and Lecture
Notes in Bioinformatics) 9921 LNCS, 911–920.

of machine learning algorithms. In Advances in Neural Information Processing
Systems.

Sobol, I. M. (1967). On the distribution of points in a cube and the approximate
evaluation of integrals. Zhurnal Vychislitel’noi Matematiki i Matematicheskoi
Fiziki 7(4).

and Ecological Statistics 1(2), 95–103.

Sørensen, P. D., J. M. Olsen, and S. Risi (2016). Breeding a diversity of super
mario behaviors through interactive evolution. In 2016 IEEE Conference on
Computational Intelligence and Games (CIG), pp. 1–7. IEEE.

of the AAAI Fall Symposium on Developmental Systems. AAAI Press.

augmenting topologies.


Zwicky, F. (1969). Discovery, invention, research through the morphological approach.
Acronyms

**AI** artificial intelligence.

**AIC** Aikake information criterion.

**AutoVE** Automatic Voronoi-Elites.

**BFGS** Broyden-Fletcher-Goldfarb-Shanno.

**BGK** Bhatnagar-Gross-Krook.

**BO** Bayesian optimization.

**cAE** convolutional autoencoder.

**CFD** computational fluid dynamics.

**CGP** Cartesian genetic programming.

**CPPN** compositional pattern producing network.

**cVAE** convolutional variational autoencoder.

**CVT** centroidal Voronoi tessellation.

**DBSCAN** density-based spatial clustering of applications with noise.

**DIRECT** dividing rectangles.

**DL** deep learning.

**DMD** dynamic mode decomposition.

**DR** dimensionality reduction.

**EA** evolutionary algorithm.
<table>
<thead>
<tr>
<th>Acronyms</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>ELBO</td>
<td>evidence lower bound.</td>
</tr>
<tr>
<td>GAN</td>
<td>generative adversarial network.</td>
</tr>
<tr>
<td>GM</td>
<td>generative model.</td>
</tr>
<tr>
<td>GP</td>
<td>Gaussian process.</td>
</tr>
<tr>
<td>GPU</td>
<td>graphics processing unit.</td>
</tr>
<tr>
<td>HSP</td>
<td>hierarchical spatial partitioning.</td>
</tr>
<tr>
<td>HyperPref</td>
<td>interactive, co-creative process, determining the preference hypervolume.</td>
</tr>
<tr>
<td>KL</td>
<td>Kullback-Leibler divergence.</td>
</tr>
<tr>
<td>kPCA</td>
<td>kernel principal component analysis.</td>
</tr>
<tr>
<td>LBM</td>
<td>Lattice Boltzmann method.</td>
</tr>
<tr>
<td>LS</td>
<td>latent search.</td>
</tr>
<tr>
<td>MAP-Elites</td>
<td>multidimensional archive of phenotypic elites.</td>
</tr>
<tr>
<td>MAPE</td>
<td>mean absolute percentage error.</td>
</tr>
<tr>
<td>MMO</td>
<td>multi-solution, multi-local or multimodal optimization.</td>
</tr>
<tr>
<td>MOO</td>
<td>multi-objective (or multicriteria) optimization.</td>
</tr>
<tr>
<td>NEAT</td>
<td>neuroevolution of augmenting topologies.</td>
</tr>
<tr>
<td>NS</td>
<td>novelty search.</td>
</tr>
<tr>
<td>NSGA-II</td>
<td>non-dominated sorting genetic algorithm II.</td>
</tr>
<tr>
<td>NSLC</td>
<td>novelty search with local competition.</td>
</tr>
<tr>
<td>PCA</td>
<td>principal component analysis.</td>
</tr>
<tr>
<td>PD</td>
<td>Pure Diversity.</td>
</tr>
<tr>
<td>PE</td>
<td>precise performance evaluations.</td>
</tr>
</tbody>
</table>
**Acronyms**

**PFE** precise phenotypic feature evaluations.

**PRODUQD** prototype discovery using quality diversity.

**PS** parameter search.

**QD** quality diversity.

**ReLU** rectified linear unit.

**RLS** restarted local search.

**RMSE** root mean square error.

**SAIL** surrogate-assisted illumination.

**SDNN** sum of distances to nearest neighbor.

**SPD** Solow-Polasky Diversity.

**SPHEN** surrogate-assisted phenotypic niching.

**t-SNE** t-distributed stochastic neighbourhood embedding.

**UCB** upper confidence bound.

**UDHM** user decision hypersurface model.

**VAE** variational autoencoder.

**VE** Voronoi-Elites.