CZECH TECHNICAL
UNIVERSITY
IN PRAGUE

F3

Faculty of Electrical Engineering
Department of Cybernetics

Bachelor's Thesis

Differential Evolution Crossover with Dependency Detection

Vojtěch Voráček

March 2021

Acknowledgement / Declaration

Chtěl bych poděkovat své manželce Ludmile za podporu nejen finanční. Díky tomu mohu na svém pracovišti dělat, co mě baví, a nejsem stresován výplatní páskou.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 13. 13. 2013

....

Abstrakt / Abstract

Abstrakt stručně a přesně reprezentuje obsah práce, shrnuje cíl, metody, výsledky a závěry.

Klíčová slova: Klíčová slova jsou odborné termíny vyjadřující obsah práce.

Překlad titulu: Křížení pro diferenciální evoluci s detekcí závislostí

This document shows and tests an usage of the plainTEX officially (may be) recommended design style CTUstyle for bachelor (Bsc.), master (Ing.), or doctoral (Ph.D.) theses at the Czech Technical University in Prague. The template defines all thesis mandatory structural elements and typesets their content to fulfil the university formal rules.

This is version 2 of this template which implements the Technika font recommended by CTU graphics identity reference since 2016.

Keywords: document design template; bachelor, master, Ph.D. thesis; TEX.

/ Contents

Chapter 1
Introduction

1.1 Motivation

Striving for the best solution of a certain problem is an important part of many fields of human interest. The process of finding the best solution according to some criteria is called optimization. Optimization in mathematical notation:

In the real world, there are a large number of engineering optimization problems whose input-output relationships are noisy and indistinct, so it cannot be assumed anything about the optimized function but it's possible to observe its outputs on given inputs. In these cases, the function is called a black box function and an optimization as a black box optimization.

Due to these limited capabilities, all black box optimization algorithms are allowed to perform just these three steps:

- Create a candidate solution
- Check if a candidate is feasible or not
- Evaluate its fitness by using the objective function

From the mid-1950s, a new family of optimization algorithms called Evolutionary algorithms has started to be developed. [1–2] Evolutionary algorithms have proven to be very effective while optimizing black box functions. [3] Among evolutionary algorithms, *Differential Evolution* (DE) has achieved excellent results on real-valued black box functions. [4].

However, there exists a class of functions containing dependent solution components and the recognition of those components may be a crucial task which could lead to significantly enhanced performance. Nevertheless, DE does not provide any tool capable of recognizing the dependent components of a solution. Thus it can be seen that this particular class of functions is the weakness of DE.

This work aims to find a way how to find dependencies between parts of solutions and how to represent a dependency structure. It would lead to the proposal of a new crossover operator for DE well suited for functions with dependent solution components. This new operator should partially eliminate the above-mentioned weakness of DE.

Chapter 2
Evolutionary algorithms

This chapter is mainly based on these references: [5–7]

Evolutionary algorithms (EAs) is a set of stochastic metaheuristic optimization algorithms inspired by Darvin's theory of evolution by natural selection. [8] The theory describes the process of development of organisms over time as a result of changes in heritable traits. Changes which allow an organism to better adapt to its environment will help it survive and reproduce more offspring. This phenomenon is commonly called as "Survival of the fittest" first used by Herbert Spencer. [9]

In analogy, EA maintains a "population" of potential solutions (*individuals*) for the given problem. Population is iteratively evolved by encouraging the reproduction of fitter individuals. The fitness is usually the value of the objective function in the optimization problem being solved. New candidate solutions are created either by combining existing individuals (crossover) or by modification of an individual (mutation). The algorithm runs until a candidate solution with sufficient quality is found or a user-defined computational limit is reached.

2.1 Components of evolutionary algorithms

In this section, certain parts of evolutionary algorithms are discussed in detail. In general, EAs can be divided into various components, procedures, or operators, which are:

- representation of individuals
- objective function
- population
- parent selection
- crossover operator
- mutation operator
- replacement strategy

To define a particular EA, it is necessary to specify these components. In addition, the initialization procedure and the termination condition must be defined to obtain working algorithm.

2.1.1 Representation of individuals

Each individual is encoded in so called *chromosomes*. Representation of chromosome is called *genotype* . While *phenotype* refers to the interpretation of the genotype, in other words, how the objective function treats the genotype. The Representation also involves a genotype-phenotype mapping. For instance, given an optimization problem on integers, if one decide to represent them by their binary code, then 20 would be seen as a phenotype and 10100 as a genotype representing it.

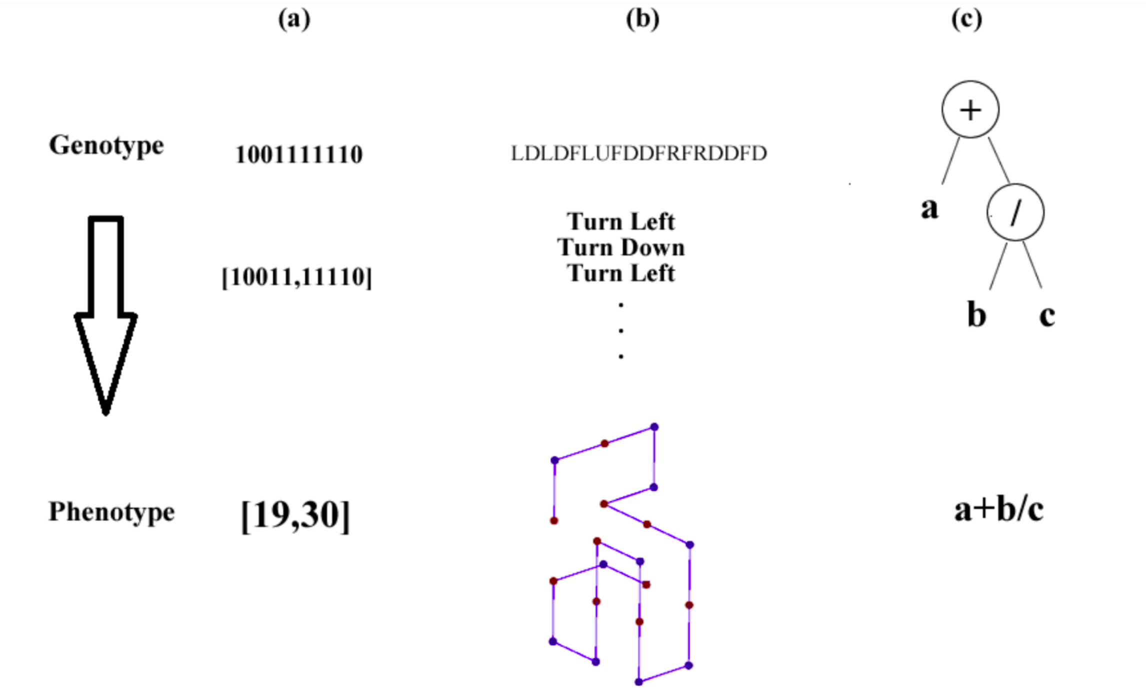

Figure 2.1. Examples of genotype-phenotype mapping (a) Integere representation (b) Protein structure representation on a lattice model (c) Tree representation for a mathematical expression [6]

2.1.2 Objective function

The role of the objective function, is to represent the requirement to adapt to. Objective function defines how quality individual is with respect to the problem in consideration. Technically, it is a function which takes an individual as an input and produces a the measure of quality of a given individual as an output. The measure of quality is called *fitness* and the objective function is called *fitness function*.

To remain with the above-mentioned example, the problem was to minimise x^2 on integers. The fitness of the individual represented by the genotype 10100 would be defined as a square of its corresponding phenotype: $20^2 = 400$

2.1.3 Population

Population in a evolutionary algorithm means a set of individuals. Population can be specified only by setting the population size, in other words, how many individuals are in population. This parameter is usually specified by user.

2.1.4 Parent selection

During each *generation* (one iteration of algorithm), a certain part of population is selected to breed offspring. The choice is made similarly to natural selection, in other words, fitter individuals are preferred, nevertheless, low quality individuals are given a small, but positive change to be selected. Otherwise, the EA could become too greedy and get stuck in the local optimum. Parent selection along with the replacement strategy pushes quality improvements. Parent selection as well as other EA procedures are usually stochastic.

Individuals selected by parent selection are called *parents*.

2.1.5 Crossover operator

Crossover is a genetic operator used to combine typically two parents to generate new offsprings. The idea behind crossover is that by mating two individuals with different but desirable features, it is possible to produce offsprings which combines both of those features. Similarly to other genetic operators, crossover is stochastic.

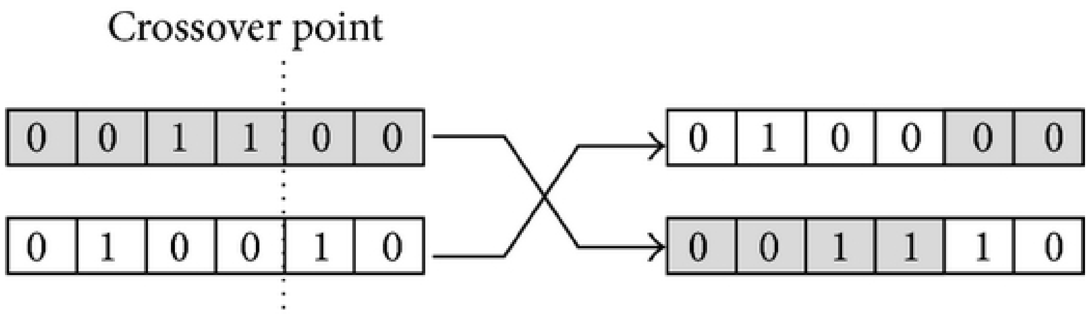

Figure 2.2. Example of One-point crossover (part of chromosome right to the Crossover point are swapped between two parents chromosomes) [10]

2.1.6 Mutation operator

Mutation is an unary genetic operator which changes parts of the chromosome of an individual, typically randomly. In mutation, the mutated individual may change entirely from the original individual. Mutation is used to maintain and introduce diversity in the genetic population.

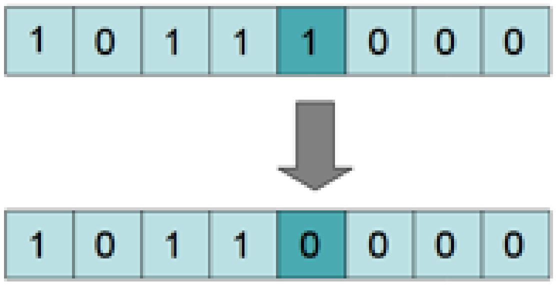

Figure 2.3. Example of mutation [11]

2.1.7 Replacement strategy

Replacement strategy defines which individuals survive and become members of subsequent generation. Typically, the decision is based on the quality of individuals, prefering those with higher fitness. The replacement strategy is similar to parent selection, as both are responsible for promoting quality improvement. However, parent selection is usually stochastic while the replacement strategy is often deterministic.

2.1.8 Initialization

It generates a defined number of individuals of the given representation, thereby creating the initial population. *Initialization* is often done randomly due to lack of knowledge when optimizing black box functions.

2.1.9 Termination condition

The algorithm runs until the *termination condition* has been reached. If we know the optimum of the optimized problem, then reaching the optimum (with a given precision $\epsilon \geq 0$) is a natural termination condition. However, since EAs are stochastic, there is usually no guarantee to reach an optimum and the condition would be never satisfied. Therefore, this condition is extended to a condition which certainly stops the algorithm, such as the limited number of fitness function calls.

2.2 General scheme

In the previous section, the main parts of an EA were introduced individually. By merging all above-mentioned components, the evolutionary algorithm is formed. This section describes the way an EA works as a whole.

Algorithm 1: Evolutionary algorithm

Result: best individual in $Population(t)$
$t \leftarrow 0$;
$Population(t) \leftarrow initializeation()$;
$fitnessFunctionEvalutation(Population(t))$;
while $termination\_condition$ *not met* **do**
  $Parents \leftarrow parentSelection(Population(t))$;
  $Offspring \leftarrow crossover(Parents)$;
  $Offspring \leftarrow mutation(Offspring)$;
  $evalutate(Offspring)$;
  $Population(t+1) \leftarrow replacementStrategy(Population(t), Offspring)$;
  $t \leftarrow t+1$;
end

Figure 2.4. General scheme of an evolutionary algorithm

Firstly, an initial population is generated by an initialization procedure. The population is subsequently evaluated by a fitness function. Then starts a generational process.

The generational process is repeated until a terminal condition is not satisfied. Generational process starts by parent selection, usually based on a fitness, some individuals are chosen to seed the new generation. The chosen individuals are combined by a crossover operator to produce offsprings, which are then modified by the mutation operator. Offspring are then evaluated by a fitness function and the generational process ends by creating a new population. Creating a new population is performed with respect to the replacement strategy that selects some newly created offsprings to replace some members of the old population.

The algorithm returns the best individual found so far, eventually some statistics concerning the run of algorithm.

2.3 Differential evolution

This chapter is mainly based on these references: [4, 7]

Differential evolution was introduced by Storn and Price [4] as an efficient evolutionary algorithm initially designed for multidimensional real-valued spaces.

DE utilizes a population of real vectors. The initial population is chosen randomly. After initialization, for each member $\vec{x}_i$ of a population P is generated a so called *mutatant vector*. A mutatant vector is generated by adding the weighted difference between two individuals ($\vec{x}_{r_2}$, $\vec{x}_{r_3}$) to a third individual ($\vec{x}_{r_1}$). These three individuals are mutually exlusive. The mutant vector is then crossed over with $\vec{x}_i$ The offspring $\vec{o}_i$ is then created by crossing over the mutatnt vector with $\vec{x}_i$.

Note that a size of mutatant vector is largely based on the actual variance in the population. The mutant vector will make major changes if the population is spread, on the other hand mutant vecotr will be small if the population is condensed in a particular region. Thus DE belongs to the family of adaptive mutation algorithms.

Lastly, newly created offspring is compared to his parent using the greedy criteria. If the offspring is better, it replace its parent in the population.

To put it more formally, standard DE is defined by specifying of following components of EA:

2.3.1 Representation

Individuals are represented by real-valued vectors:

$$\vec{x_i} = \{x_{i,1}, x_{i,2}, ..., x_{i,D}\}, \forall j : x_{i,j} \in \mathbb{R},$$

where i represents the index of the individual in the population P and D stands for dimension of the optimized function. Population is represented as following:

$$P = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_{NP}\}, NP \geq 4,$$

where NP is the size of population.

2.3.2 Mutation

For each individual in the population $\vec{x}_i, i = 1, 2, ..., NP$, DE generates mutant vector $\vec{m}_i$ as following:

$$\vec{m}_i = \vec{x}_{r_1} + F * (\vec{x}_{r_2} - \vec{x}_{r_3}),$$

with random, mutually exclusive indexes $r_1, r_2, r_3 \in \{1, 2, ..., NP\}$, which are also chosen to be different from the running index i. F, called *differential weight*, is a constant factor $\in [0, 2]$, representing the amplification of the random deviation $(\vec{x}_{r_2} - \vec{x}_{r_3})$. Individual $\vec{x}_i$ is called the parent of $\vec{m}_i$.

2.3.3 Crossover

After mutation the mutant vector $\vec{m}_i$ undergoes a crossover with the its parent $\vec{x}_i$ to generate the offspring $\vec{o}_i$. Standard DE take advantage of binomial crossover, where offspring is generated as following:

$$\vec{o}_i = \vec{x}_{r_1} + F * (\vec{x}_{r_2} - \vec{x}_{r_3}),$$

Chapter 3
Modelovani zavislosti

Chapter 4
Hledani zavislosti

Chapter 5
Setup

Chapter 6
Results

References

[1] R. M. Friedberg. A Learning Machine: Part I. *IBM J. Res. Dev.*. 1958, 2 2-13.

[2] R. M. Friedberg, B. Dunham, and J. H. North. A Learning Machine: Part II. *IBM J. Res. Dev.*. 1959, 3 282-287.

[3] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík. Comparing Results of 31 Algorithms from the Black-Box Optimization Benchmarking BBOB-2009. *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10 - Companion Publication*. 2010, 1689-1696. DOI 10.1145/1830761.1830790.

[4] Rainer Storn, and Kenneth Price. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*. 1997, 11 341-359. DOI 10.1023/A:1008202821328.

[5] A. E. Eiben, and James E. Smith. *Introduction to Evolutionary Computing*. 2nd edition. Springer Publishing Company, Incorporated, 2015. ISBN 3662448734.

[6] Ka-Chun Wong. *Evolutionary Algorithms: Concepts, Designs, and Applications in Bioinformatics: Evolutionary Algorithms for Bioinformatics*. 2015.

[7] Sean Luke. *Essentials of Metaheuristics* . 2009 .
`http://cs.gmu.edu/~sean/book/metaheuristics/` .

[8] Darwin Charles. *On the Origin of Species by Means of Natural Selection*. London: Murray, 1859.

[9] Spencer Herbert. *The Principles of Biology*. London: William and Norgate, 1864.

[10] Marco Ferreira, J. Bagari■, Jose Lanza-Gutierrez, Sílvio Mendes, Jo■o Pereira, and Juan A. Gomez-Pulido. On the Use of Perfect Sequences and Genetic Algorithms for Estimating the Indoor Location of Wireless Sensors. *International Journal of Distributed Sensor Networks*. 2015, 2015 1-12. DOI 10.1155/2015/720574.

[11] Kendall Park. *0-1 Knapsack Approximation with Genetic Algorithms* . 2014 .
`https://github.com/KendallPark/genetic-algorithm` .