Implementation of Transcriptome Anomalous Diffusion in Genetic Programming (TAD-GP)

Taxonomy: Genetic Programming algorithm is an example of an Evolutionary Algorithm and belongs to the field of Evolutionary Computation and more broadly Computational Intelligence and Biologically Inspired Computation. The GP algorithm is sibling to other Evolutionary Algorithms, *e.g.* the Genetic Algorithm (GA), Evolution Strategies, Evolutionary Programming, and Learning Classifier Systems.

Inspiration: The Genetic Programming algorithm is inspired by population genetics (including heredity and gene frequencies), and evolution at the population level, as well as the Mendelian understanding of the structure (such as chromosomes, genes, alleles) and mechanisms (such as recombination and mutation). TAD instead focuses on the transcriptome structure (transcript classes, *e.g.* mRNA, miRNA, lncRNA and rRNA) and mechanisms (transcription, export, import, silencing, enhancing, decay half-life).

Metaphor: Individuals of a population contribute a portion of their transcriptome (functionally the genotype) proportional to a semantic filter dependent on all of the transcriptome, acting on each transcript individually. The individual transcript contributions are pooled, then distributed to the population in the form of survivors. The next generation is created through a process of implementing transcriptome operations that involves silencing or enhancing specific semantically-selected transcripts in the population, and optionally with the introduction of random errors (mismatch at the semantic level). Alternate process of multicellular growth involves transcript operators such as secretion to extracellular pool, absorption of transcripts from extracellular pool, and the introduction of random copying errors (called mutation) or degradation operators (with some half-life). This iterative process may result in an improved adaptive-fit between the phenotypes of individuals in a population and the environment.

Programs may be evolved and used in a secondary adaptive process, where an assessment of candidates at the end of that secondary adaptive process is used for differential reproductive success in the first evolutionary process. This system may be understood as the inter-dependencies experienced in evolutionary development where evolution operates upon an embryo that in turn develops into an individual in an environment that eventually may reproduce.

Strategy

The objective of the Genetic Programming algorithm is to use induction to devise a computer program. This is achieved by using transcriptome and evolutionary operators on candidate programs with a Cartesian graph or tree structure to improve the adaptive fit between the population of candidate programs and an objective function.

Procedure

Algorithm (below) provides a pseudocode listing of the Genetic Programming algorithm for minimizing a cost function, based on Koza and Poli's tutorial [Koza2005]. Genetic Program uses LISP-like symbolic expressions called S-expressions that represent the graph of a program with function nodes and terminal nodes. While the algorithm is running, the programs are treated like data, and when they are evaluated they are executed.

Algorithm TAD-GP in pseudo code is:

// start with an initial time

t := 0;

// initialize a usually random population transcriptomes of N individuals. Transcripts are programs of
// some length and have calculable Similarity and Reverse Complementarity measures to the whole
// population of transcriptomes.

initPopulation P (t);

// evaluate fitness of all initial individuals of population

evaluate P (t);

// test for termination criterion (time, fitness, etc.)

while not done **do**

    // increase the time counter

    t := t + 1;

    // select fittest sub-population for survival; return P'

    P' := *selectFittest* [P (t), selectionP%, P'];

    // Calculate $T_{\alpha}^{OUT} = T_{\alpha} * F[S, RC, N, T_{\alpha}]$ from Equation 2.
    // Remove transcripts with semantic *filter* F from each survivor to ExtracellularPool
    // Filter acts on selectionF% of transcriptome; return modified individuals as P''
    // Calculate $T^{EC} = T_1^{OUT} + T_2^{OUT} + T_3^{OUT} + \ldots$ from Figure 2 in text.

    ExtracellularPool := *filter* [P' (t), selectionF%, P''(t)];

    // Add selectionD% of ExtracellularPool transcripts to P'' stochastically; return P'''
    // Calculate $T_{\alpha}^{IN} = T^{EC} * F[S, RC, N, T_{\alpha}]$ from Equation 2.

    distribute [P'' (t), ExtracellularPool, selectionD%, P'''(t)] ;

    // implement transcriptome silencers and enhancers, return as P

    P := implement [P''' (t), P];

od

end GA-AD

Heuristics

- The evaluation (fitness assignment) of a candidate solution takes the structure of the program into account, rewarding parsimony.

- The selection process should be balanced between random selection and greedy selection to bias the search towards fitter candidate solutions (exploitation), whilst promoting useful diversity into the population (exploration).
- All transcriptome operations ensure (or should ensure) that syntactically valid and executable programs are produced as a result of their application.
- The Genetic Programming algorithm component is configured with a high-probability of transfer to ExtraCellular Pool and a low-probability of RNA point modifications. Other operators such as transcription, decay, and architecture alterations (Accessible/Inaccessible or A/I functions) are used with moderate-level probabilities and fill in the probabilistic gap.
- Architecture altering operations are not limited to the A/I functions of sub-structures of a given program.
- The function set may also include control structures such as conditional statements and loop constructs.
- The Genetic Programming algorithm can make use of Automatically Defined Functions (ADFs) that are sub-graphs and are promoted to the status of functions for reuse and are co-modified with the programs.
- The genetic operators employed during reproduction in the algorithm may be considered transformation programs for candidate solutions and may themselves be co-evolved in the algorithm.