

Self-Emergence of Structures in Gene Expression Programming

By: Xin Li (xli1@cs.uic.edu)

**Department of Computer Science
University of Illinois at Chicago**



Outline



- Research Overview
- Main Contributions
 - Constant Creation in GEP
 - P-GEP: A New Representation Scheme
 - Emergent Structures in P-GEP
- Future Research Plans

Research Scope

- **Scenario**

- Data mining tasks are pivotal for the improvement of manufacturing and design processes.
- Some of the hidden patterns or relationship among the data are very complex.
- Example applications: cell phone drop tests, failure call detection, wave filter design, driving simulation, etc.

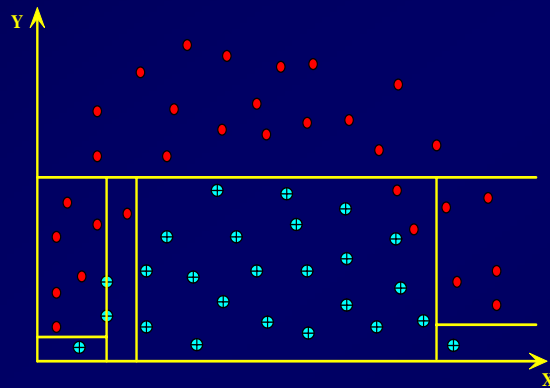


- **What would help?**

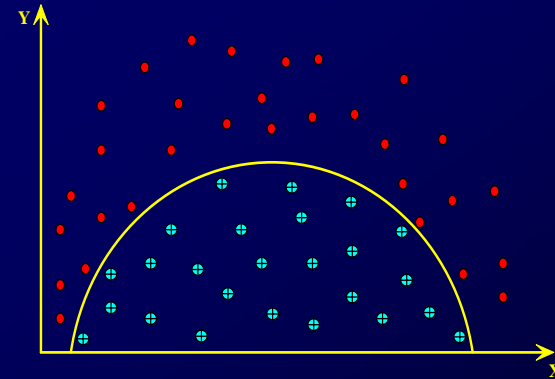
- Data preprocessing methods
- Flexible, automated and expressive data mining techniques for complex knowledge discovery

Traditional Methods

- Decision trees, rule induction, association rule mining, clustering, statistical modeling, Bayesian classification, artificial neural networks, etc.
- However...
 - Division of the data space



Vs.

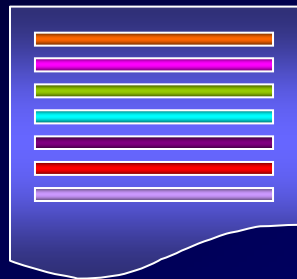


- Interpretability of the solutions
- Pre-knowledge and assumption

Evolutionary Computation

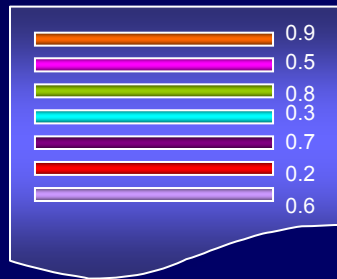
- An adaptive and parallel search procedure

Initial Random Population

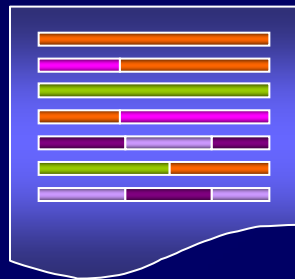


Evaluation

Evaluated Population



Newly Generated Population



Return the best individual



YES

Stop criteria satisfied?

NO

Selection, reproduction and variation via genetic operations

- Minimum Pre-knowledge and unconstrained approximation to the final solution (mined knowledge)
- Successful applications: symbolic regression, classification, optimization, time series analysis, logic synthesis, and cellular automata, etc.

Gene Expression Programming

- A functional programming language paradigm
- Linear genotype (chromosome)
- Expression Tree (ET) phenotype

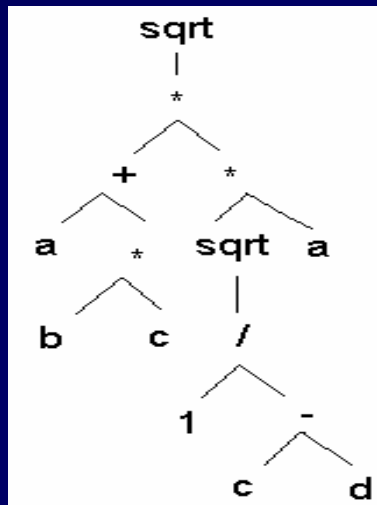


Mapping: Kava notation

Chromosome: sqrt.*.+.*.a.*.sqrt.a.b.c./1.-.c.d



ET:



Mathematical form:

$$\sqrt{(a + bc) * a \sqrt{\frac{1}{c - d}}}$$

Function set:

{+, -, *, /, sqrt}

Terminal set:

{a, b, c, d, 1}

Research Objectives

- Apply **Gene Expression Programming (GEP)** algorithm (Ferreira, 2001) for the general data mining tasks.
- Improve the problem solving ability of **GEP** to fulfill complex data mining tasks by **preserving and utilizing the self-emergence of structures** during its evolutionary process.

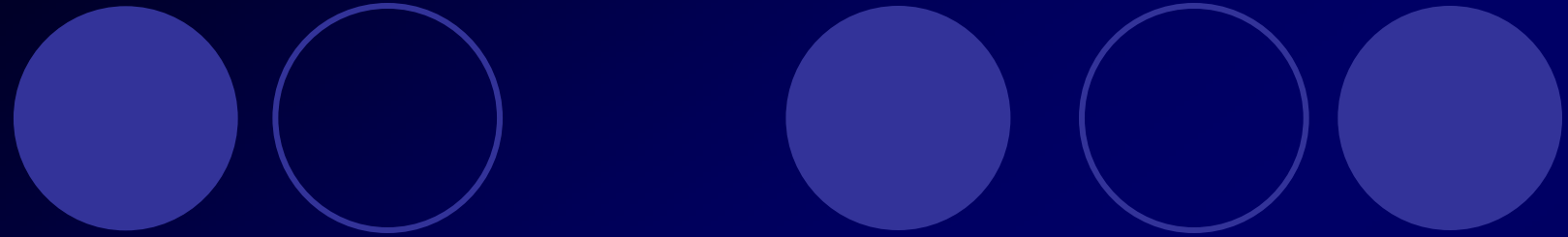
Main Contributions

- Major hypotheses

- An efficient search procedure should be guided toward good solution structures
- Decomposition of the functionality and hierarchical evolution of solutions are the ways to discover complex hidden knowledge.

- An incremental approach

- Constant creation in GEP
- Prefix-GEP: a new representation scheme
- Emergent structures in Prefix-GEP



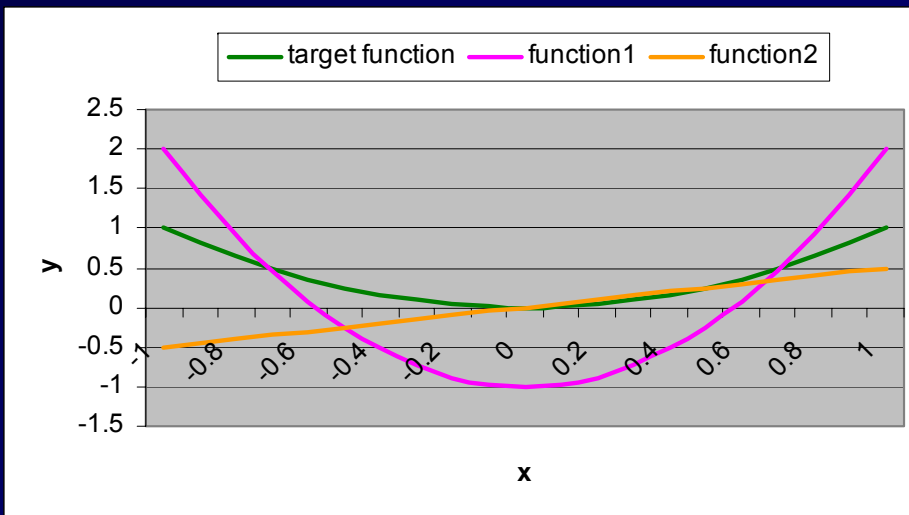
Constant Creation in Gene Expression Programming

-- On the way to promote good solution structures

Constant Creation in GEP -- Why?

- Finding the optimal solution structure and the appropriate constant coefficients are **intermingled** in GEP

Example of functional approximation



target function: $y=x^2$
function1: $y=3x^2-1$
function2: $y=0.5x$

fitness(function1) = 0.29
fitness(function2) = 0.42

- **Local optimization of constants** may help speed up the learning process

Constant Mutation Operations

- Single-point constant mutation

Function set: {+, -, *, /, sqrt}

Terminal set: {x, 1, 2, 3, 5, 7}

1. Single-point creep mutation:

sqrt.-.*.***5**.3.x.x./.2.5.+.-.1.x → sqrt.-.*.***3**.3.x.x./.2.5.+.-.1.x

2. Single-point random mutation:

sqrt.-.*.***5**.3.x.x./.2.5.+.-.1.x → sqrt.-.*.***1**.3.x.x./.2.5.+.-.1.x

- GEP constant (creep or random) mutation operation

- For every constant gene in a chromosome, perform a **single-point constant mutation (creep or random)** in a **greedy** manner.

Constant Creation in GEP

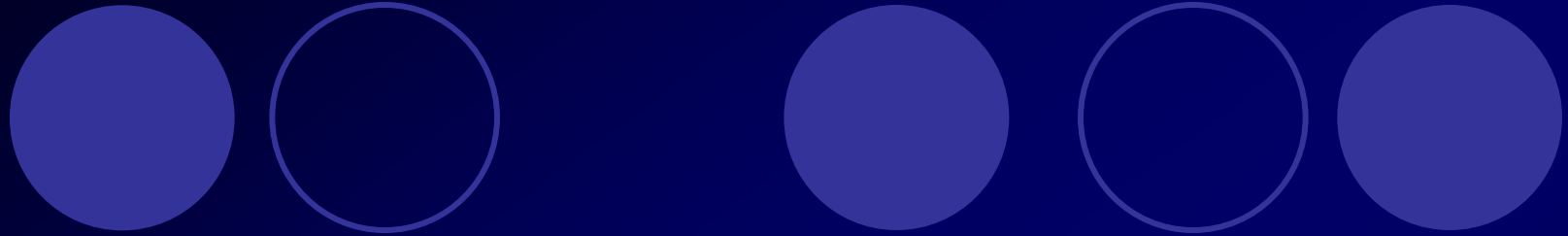
- Five **Constant Creation Methods** proposed for investigation:
 - Creep mutation on best individuals
 - Random mutation on best individuals
 - Creep mutation for first $\alpha\%$ generations
 - Random mutation for first $\alpha\%$ generations
 - Random mutation for generations at intervals

Experimental Results

- GEP is strong in finding or composing the most suitable combination of constants and functional structures
- CCMs applied to the whole population can significantly improve the fitness of average individuals, and higher fitness scores have been achieved for the final best solution
- When emergent useful solution structural information during the evolution is taken into account in fitness measurement, the problem solving ability of GEP can be improved

Related Work

- Long-term attention in Genetic Programming community
 - Creep and uniform constant mutation (Ryan & Keijzer, 2003)
 - Hill climbing (Evelt & Fernandez, 1998)
 - Simulated annealing (Evelt & Fernandez, 1998)
 - Local gradient search (Topchy & Punch, 2001)
 - Digit concatenation (O'Neill, Dempsey, et al., 2003)
 - Linear scaling (Keijzer, 2003)



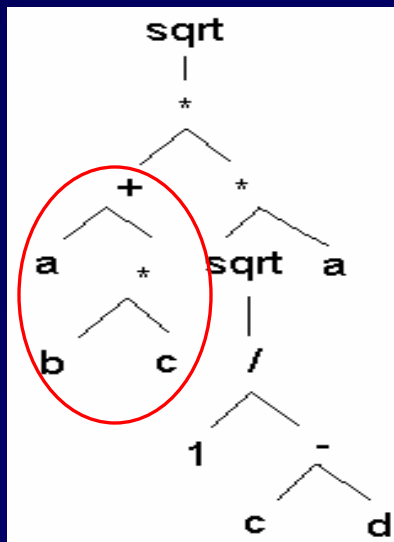
Prefix Gene Expression Programming

-- To define a friendly genotype representation for the evolution of the solution structures

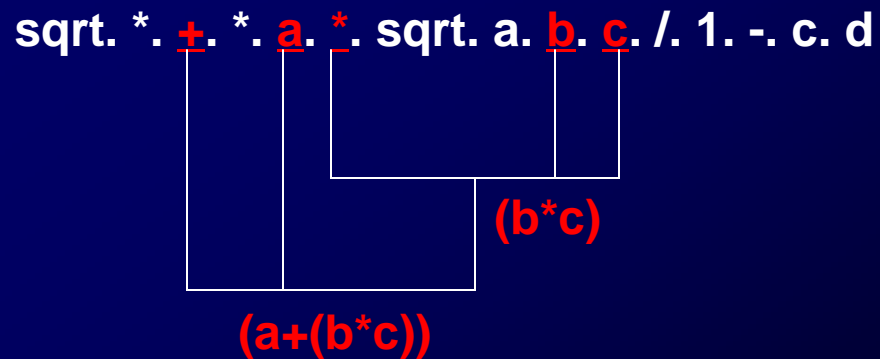
Prefix GEP -- Why?

- Limitations of GEP's linear representation
 - A single substructure does not stay contiguously in genotype
 - The levels of functional complexity in the phenotype are not directly reflected in the genotype

ET:



Chromosome:



Introducing P-GEP

- A new genotype-phenotype mapping mechanism

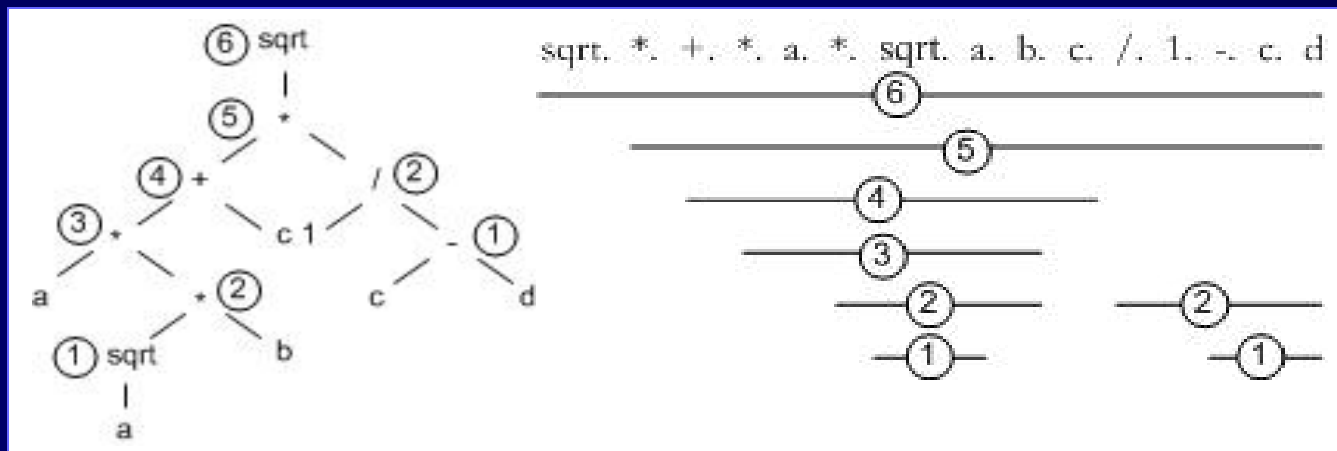
Chromosome: <code>sqrt.*.+*.a*.sqrt.a.b.c./1.-.c.d</code>	
GEP (Karva notation based)	P-GEP (Prefix notation based)
<p>ET:</p>	<p>ET:</p>
<p>Function: $\sqrt{(a + bc) * a \sqrt{\frac{1}{c - d}}}$</p>	<p>Function: $\sqrt{(a \sqrt{ab} + c) (\frac{1}{c - d})}$</p>

Major Characteristics of P-GEP

- Substructure preserving characteristics: less destructive genetic operators

Genetic Operators	Initial Chromosomes	Offspring
One-point Crossover	$\text{sqrt}.*.+.*.a.*.\text{sqrt}.a.b.c./1.-.c.d$ $\text{sqrt}.*.+a.*.b.c.*.\text{sqrt}./1.-.c.d.a$	$\text{sqrt}.*.+.*.a.b.c.*.\text{sqrt}./1.-.c.d.a$ $\text{sqrt}.*.+a.*.*.\text{sqrt}.a.b.c./1.-.c.d$

- Inherent hierarchy in forming the solution



A Schema Theorem for P-GEP

- **Schema of P-GEP** (Li, et al., 2005)
 - a set of non-overlapping substructures in the instance chromosome
 - position dependent
- **Estimation of the occurrence of schema** in the next generation:

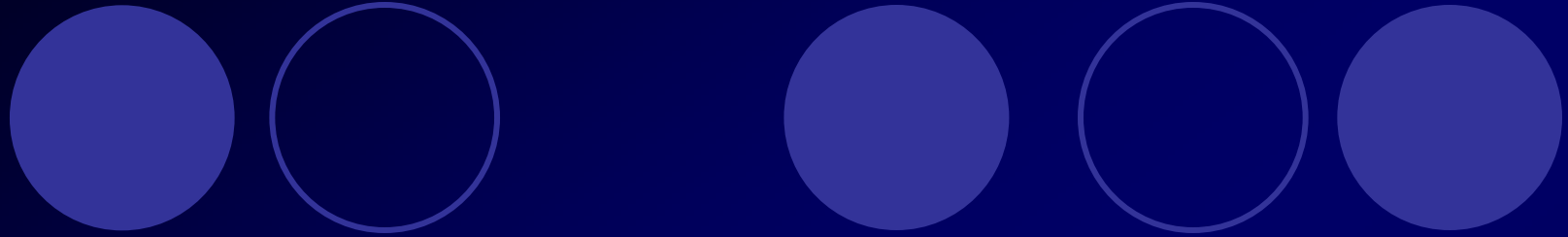
$$m(H, t+1) \geq \frac{f(H, t)}{f(t)} (1 - \varepsilon_c)(1 - \varepsilon_r)(1 - \varepsilon_m) m(H, t)$$

In other words, **smaller** and **fitter substructure** units will enjoy a prosperity in later generations and **serve as the building blocks** of the final solution

Related Work



- **Linear Genetic Programming** (e.g., Brameier & Banzhaf, 2001; Langdon & Banzhaf, 2004)
 - Chromosomes are composed of **assembly language type instructions**
- **Grammatical Evolution** (e.g., Ryan, Collins and Neil, 1998)
 - Chromosomes are composed of random numbers, mapping into **BNF grammar rules**
- **Stack-based Genetic Programming** (e.g., Perkis 1994; Spector & Robinson, 2002)
 - Chromosomes are composed of **FORTH-style operand stacks**



Emergent Structures in Prefix-GEP

-- Toward automatic identification of potential
subroutines and hierarchical evolution

Emergent Structures -- Why?

- Structures largely determine the functionality of the corresponding solution
- Structures adapted to the environment via fitness measurement during the evolution are the most competitive candidates for desirable structures of the unknown optimal solution
- Reusable subroutines are essential for saving the computational effort

Emergent Substructures in P-GEP

- **Substructures in P-GEP**

- Building blocks as **coherent functional elements in linear genotype** that help compose the solution.

- **Self-emergence of Substructures in P-GEP**

- The procedure that the substructures are generated, preserved, and evolved automatically during the P-GEP process.

- **Emergent Substructures in P-GEP**

- Substructures that have a **high appearance frequency** in a group of individuals with the highest fitness values (called as **elite group** and **elite** for every individual in this group) among the whole population.

Recognition of Emergent Substructures

- To determine the minimum required appearance frequency in the elite group
- To extract the emergent substructures into a single denoting symbol (called **derived gene**)

Example chromosome: `sqrt.*.*.a.*.sqrt.a.b.c./1.-.c.d`

Derived gene	Substructures	Complexity level	Expanded substructures
DG0	<code>sqrt.a</code>	0	<code>sqrt.a</code>
DG1	<code>-.c.d</code>	0	<code>-.c.d</code>
DG2	<code>*.DG0.b</code>	1	<code>*.sqrt.a.b</code>
DG3	<code>/.1.DG1</code>	1	<code>/.1.-.c.d</code>
DG4	<code>*.a.DG2</code>	2	<code>*.a.*.sqrt.a.b</code>
DG5	<code>+.DG4.c</code>	3	<code>+.*.a.*.sqrt.a.b.c</code>
DG6	<code>*.DG5.DG3</code>	4	<code>*.*.*.a.*.sqrt.a.b.c./1.-.c.d</code>
DG7	<code>sqrt.DG6</code>	5	<code>sqrt.*.*.*.a.*.sqrt.a.b.c./1.-.c.d</code>

Compression and Expansion Operators for Substructures

- **Compression operator:** to compress a discovered substructure into a single gene symbol
- **Expansion operator:** to restore the derived gene symbol back to the gene segment it denotes before the compression

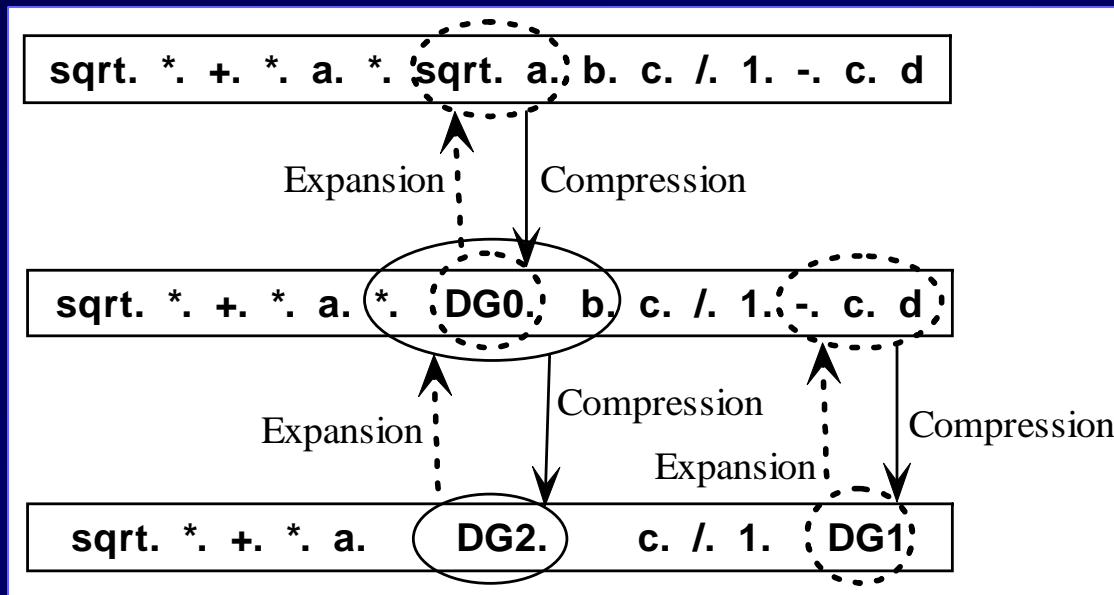
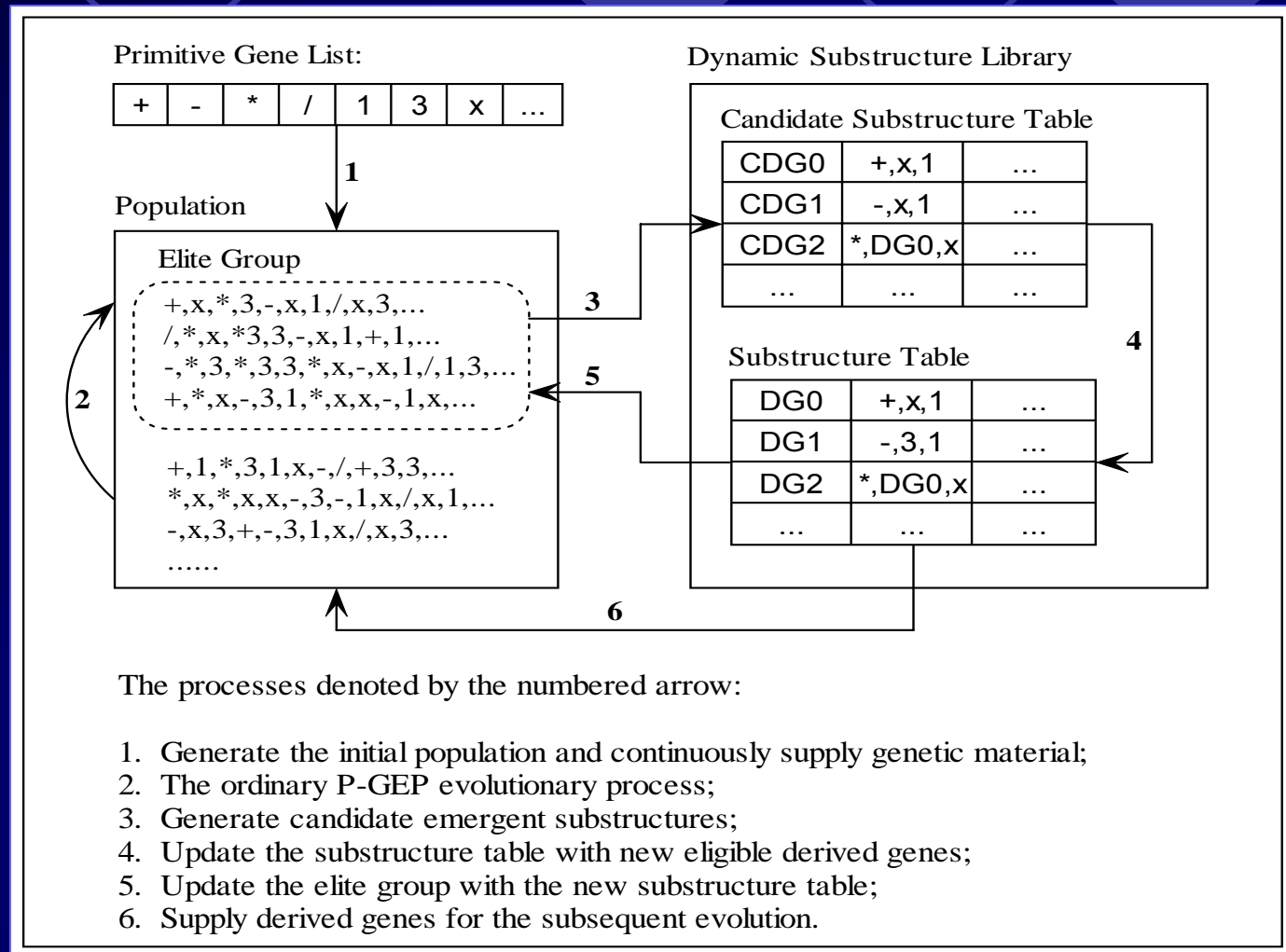


Illustration of compression and expansion operators

Dynamic Substructure Library

A Systematic Overview of the Dynamic Substructure Library



Experimental Results

- P-GEP algorithms with the substructure library have better convergence curves
- Dynamic substructure library helps find useful derived genes

○ Testing problem:

$$y = 3 * (x + 1)^3 + 2 * (x + 1)^2 + (x + 1)$$

○ Evolved optimal solution:

Solution	<code>((3*(x*((DA4*x)+5)))-DA3)+(7-(1+x))</code>
Derived genes	<code>DA0: *.x.1; DA1: /.3.5; DA2: +.x.5; DA3: *.x.x; DA4: -.DA2.1</code>

Related Work

- Schemata and building blocks in GAs (Holland, 1975; Goldberg, 1989)
- Substructure, modules and hierarchy in GP
 - **Automatically defined functions (ADF)** (Koza, 1992; Brock 1994; Yu & Clack, 1997)
 - **Module acquisition (MA)** (Angeline, 1994)
 - **Adaptive representation** (Rosca & Ballard, 1995)
 - **Hierarchical evolution (hGP)** (Banzhaf, Banscheraus & Dittrich, 1999)
 - **Subtree encapsulation** (Roberts, Howard, and Koza 2001)
 - **Transferable library** (Keijzer, Ryan and Cattolico 2004)
- Multi-genetic system in GEP (Ferreira, 2002)

Future Research Plans

- Try to **define the solution structure** based on P-GEP's genotype
 - The skeleton of the solution: extract all of the functional genes in the corresponding chromosome.
 - Use the probabilistic distribution of the functional genes to denote the overall solution structure of the elite group.
- Design a method to evaluate the **structural fitness of a solution**, and combine this value into the final fitness to guide the search procedure
 - Compare the structure of a specific solution to the established probabilistic model.

The End...



- Thank you!
- Questions & Suggestions?