

Probabilistic Model Building and Competent Genetic Programming

Kumara Sastry, and David E. Goldberg

Illinois Genetic Algorithms Laboratory
Department of General Engineering
University of Illinois at Urbana-Champaign

<http://www-illigal.ge.uiuc.edu>

{ksastry,deg}@uiuc.edu

Supported by:

AFOSR F49620-00-0163, F49620-03-1-0129, NSF DMI-9908252, and CSE fellowship

Motivation

- ❖ Genetic Algorithms (GAs)
 - ◆ Design decomposition theory (Goldberg, 2002)
 - ◆ Design of **competent** GAs
 - ★ Solve hard problems *quickly, reliably, and accurately*
 - ★ Polynomial scale-up on boundedly-difficult problems
- ❖ Genetic Programming (GP)
 - ◆ Significant advances in application
 - ★ Problem-specific operators
 - ◆ Limited attention to competent operator design
 - ★ *Generic* operators that adapt linkage
 - ★ Solve hard problems quickly, reliably, and accurately

Outline

- ❖ Background
- ❖ Objective
- ❖ Extended compact GA (eCGA)
- ❖ Probabilistic incremental program evolution (PIPE)
- ❖ Extended compact GP (eCGP)
- ❖ Initial Results: GP-easy and GP-hard problems
- ❖ Future Work & Conclusions

Background

- ❖ Competent GAs
 - ◆ Messy genetic algorithm (Goldberg, Korb, & Deb, 1989)
 - ◆ Operators that identify and exchange building blocks
- ❖ Perturbation techniques:
 - ◆ Messy GA, fast messy GA, GEMGA, LINC, LIMD, ...
- ❖ Linkage adaptation techniques: LLGA
- ❖ Probabilistic model building techniques:
 - ◆ PBIL, CGA, BMDA, BOA, hBOA, IDEA, eCGA, PIPE, ...
- ❖ Goldberg (2002), Pelikan; Goldberg, & Lobo (2002); Pelikan (2003); Larranaga & Lozano (2002); Chen *et al* (In preparation).

Objective

- ❖ Design a **competent** GP
 - ◆ Identify and exchange substructures *effectively*
 - ◆ Polynomial scale-up on boundedly difficult problem
- ❖ Combine ideas from GAs & GP
 - ◆ Extended compact GA (eCGA) (Harik, 1999)
 - ◆ Probabilistic incremental program evolution (PIPE) (Salustowicz & Schmidhuber, 1997)
- ❖ Study scale-up of competent GP
 - ◆ GP-easy and GP-hard problems

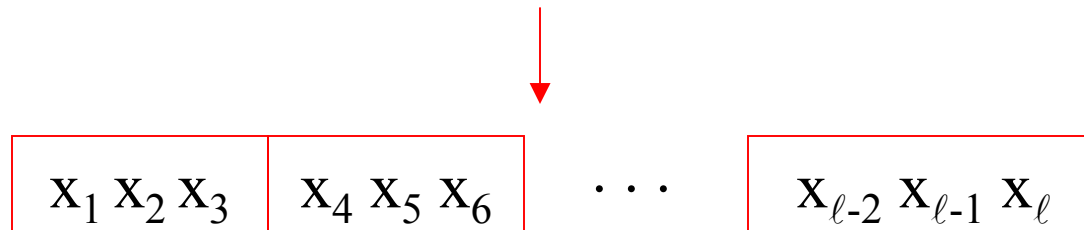
Extended Compact Genetic Algorithm (eCGA)

- ❖ A Probabilistic model building GA (Harik, 1999)
 - ◆ Builds models of good solutions as linkage groups
- ❖ Key idea:
 - ◆ Good probability distribution \equiv *Linkage learning*
- ❖ Key components:
 - ◆ Representation: *Marginal product model* (MPM)
 - ★ Marginal distribution of a gene partition
 - ◆ Quality: *Minimum description length* (MDL)
 - ★ Occam's razor principle
 - ★ All things being equal, simpler models are better
 - ◆ Search Method: Greedy heuristic search

Marginal Product Model (MPM)

- ❖ Partition variables into clusters
- ❖ Product of marginal distributions on a partition of genes
- ❖ Gene partition maps to linkage groups

MPM: $[1, 2, 3], [4, 5, 6], \dots [l-2, l-1, l]$



$\{p_{000}, p_{001}, p_{010}, p_{100}, p_{011}, p_{101}, p_{110}, p_{111}\}$

Minimum Description Length Metric

❖ **Hypothesis:** For an optimal model

- ◆ Model size and error is minimum

❖ **Model complexity, C_m**

- ◆ # of bits required to store all marginal probabilities

$$C_m = \log_2(n) \sum_{i=1}^m [\chi^{k_i} - 1]$$

❖ **Compressed population complexity, C_p**

- ◆ Entropy of the marginal distribution over all partitions

$$C_p = n \sum_{i=1}^m \sum_{j=1}^{\chi^{k_i}} [-p_{ij} \log_2(p_{ij})]$$

❖ **MDL metric, $C_c = C_m + C_p$**

Algorithmic Description of eCGA

1. Initialize the population
2. Evaluate the fitness of individuals
3. Perform selection
4. Build Probabilistic models of selected solutions
 - ◆ Search for an optimal model
 - ★ Optimize both model structure and parameters
5. Generate new solutions using the model
6. Repeat steps 2–5 till termination criteria are met

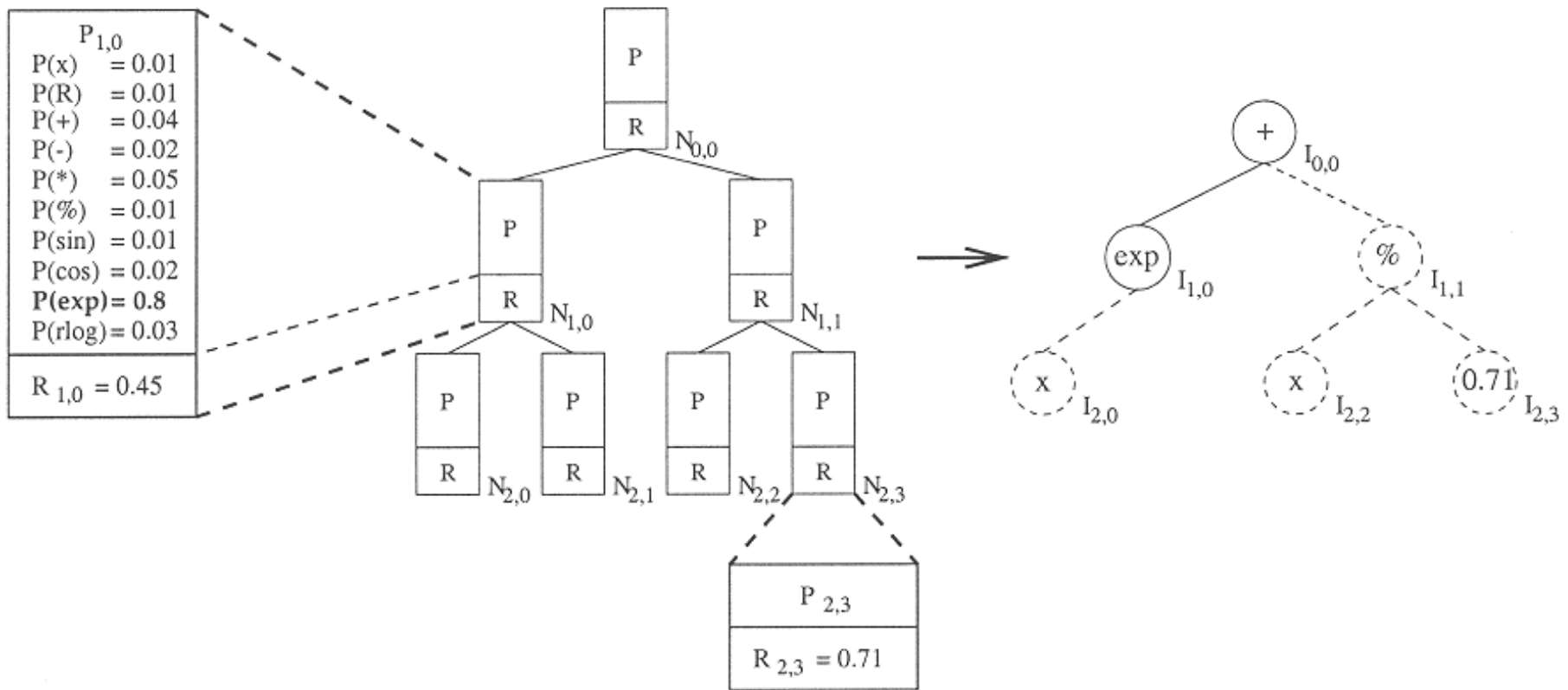
Building an Optimal MPM

1. Assume independent genes ($[1],[2],\dots,[\ell]$)
2. Compute MDL metric, C_c
3. All combinations of two subset merges
 - ◆ Eg., $\{([1,2],[3],\dots,[\ell]), ([1,3],[2],\dots,[\ell]), ([1],[2],\dots,[\ell-1,\ell])\}$
4. Compute MDL metric for all model candidates
5. Select the set with minimum MDL, C'_c
6. If $C_c > C'_c$, accept the model and go to step 2.
7. Else, the current model is optimal

Probabilistic Incremental Program Evolution

- ❖ Salustowicz & Schmidhuber (1997)
 - ◆ Based on PBIL (Baluja, 1994)
 - ◆ Tree representation: Functions and Terminals
- ❖ **Univariate model:** Each node is independent
 - ◆ Complete n-ary tree
 - ◆ Probability of selecting a function or terminal
- ❖ Model metric and search method are not needed
- ❖ Good for simple problems
 - ◆ Cannot handle complex interactions

PIPE: Model Representation & Sampling



- ❖ Start with root node
- ❖ Depth first, left-to-right traversal
- ❖ Choose either a function or terminal based on model

Extended Compact Genetic Programming

❖ Extension of PIPE

- ◆ Handle multivariate interactions
- ◆ Identify and exchange important substructures

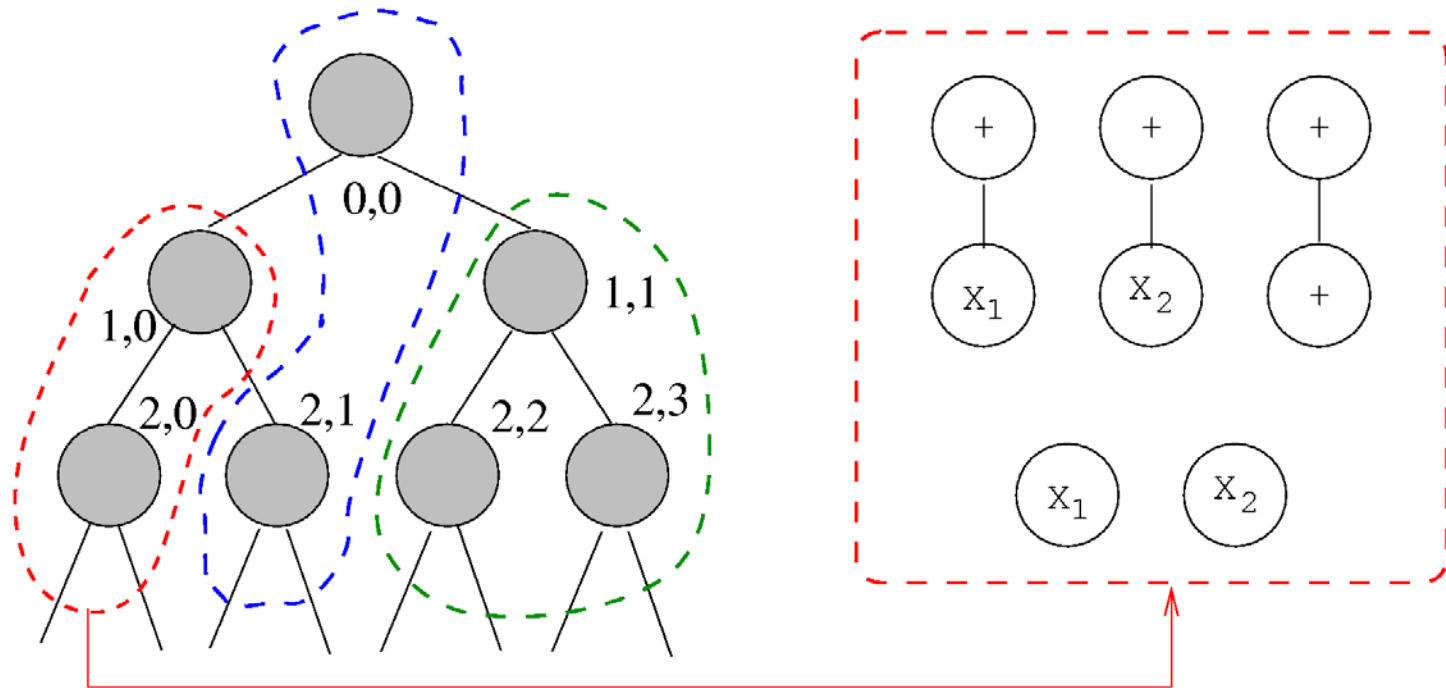
❖ Extension of eCGA

- ◆ Evolve programs instead of bitstrings
- ◆ Handle variable-size problems

❖ Key components:

- ◆ Complete n-ary tree for model building
- ◆ Model representation: Marginal product model
- ◆ Metric: Minimum description length
- ◆ Model Searcher: Greedy heuristic method

eCGP: Model Representation



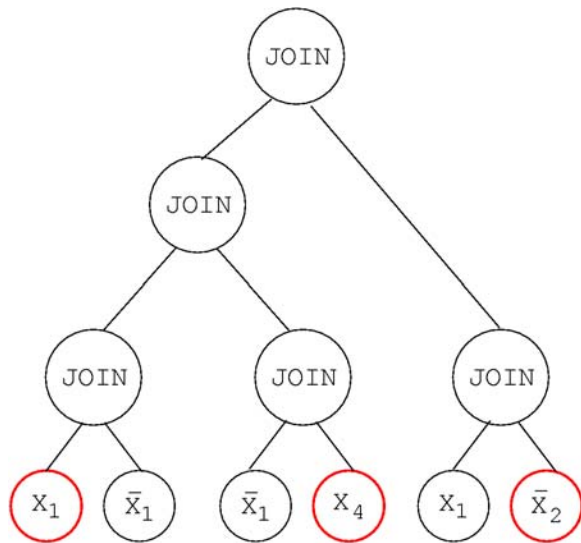
- ❖ Complete n-ary tree (similar to PIPE)
- ❖ Marginal product model (similar to eCGA)
 - ◆ Partition tree-nodes into clusters
 - ◆ Marginal probability distribution of each cluster

Test Problem Design

- ❖ Design **adversarial** problems
 - ◆ Thwart the mechanism of the GP
- ❖ Building-block (BB) identification is critical
 - ◆ BB structure not known to GP
 - ★ No a priori knowledge should be used
- ❖ Tunable problem difficulty
 - ◆ Without changing the functional form
- ❖ Should bound GP performance

GP-Easy Problem: ORDER

- ❖ Primitive set: $\{\text{JOIN}, X_1, \bar{X}_1, X_2, \bar{X}_2, \dots, X_\ell, \bar{X}_\ell\}$
- ❖ Parse program tree inorder
- ❖ Primitive first encountered is **expressed**.



Parsed primitives

$\{X_1, \bar{X}_1, \bar{X}_1, X_4, X_1, \bar{X}_2\}$

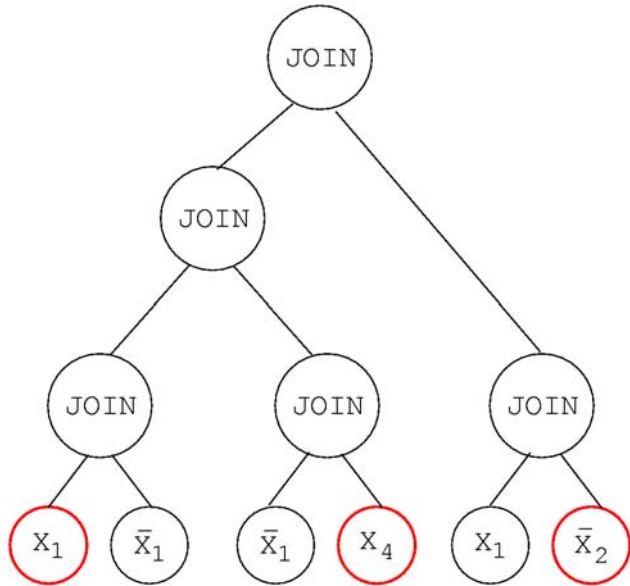
Expressed primitives

$\{X_1, \bar{X}_2, X_4\}$

- ❖ Fitness: # of expressed primitives X_i
- ❖ Similar to OneMax in GAs

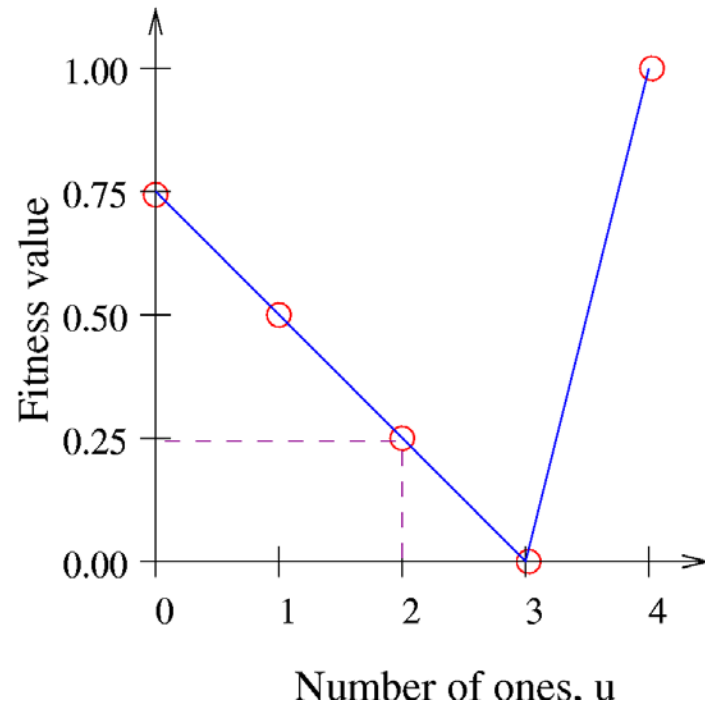
GP-Hard Problem: Deceptive Trap

- ❖ Primitive set: $\{\text{JOIN}, X_1, \bar{X}_1, X_2, \bar{X}_2, \dots, X_\ell, \bar{X}_\ell\}$
- ❖ Expression mechanism: Same as ORDER
- ❖ BB identification and exchange is critical

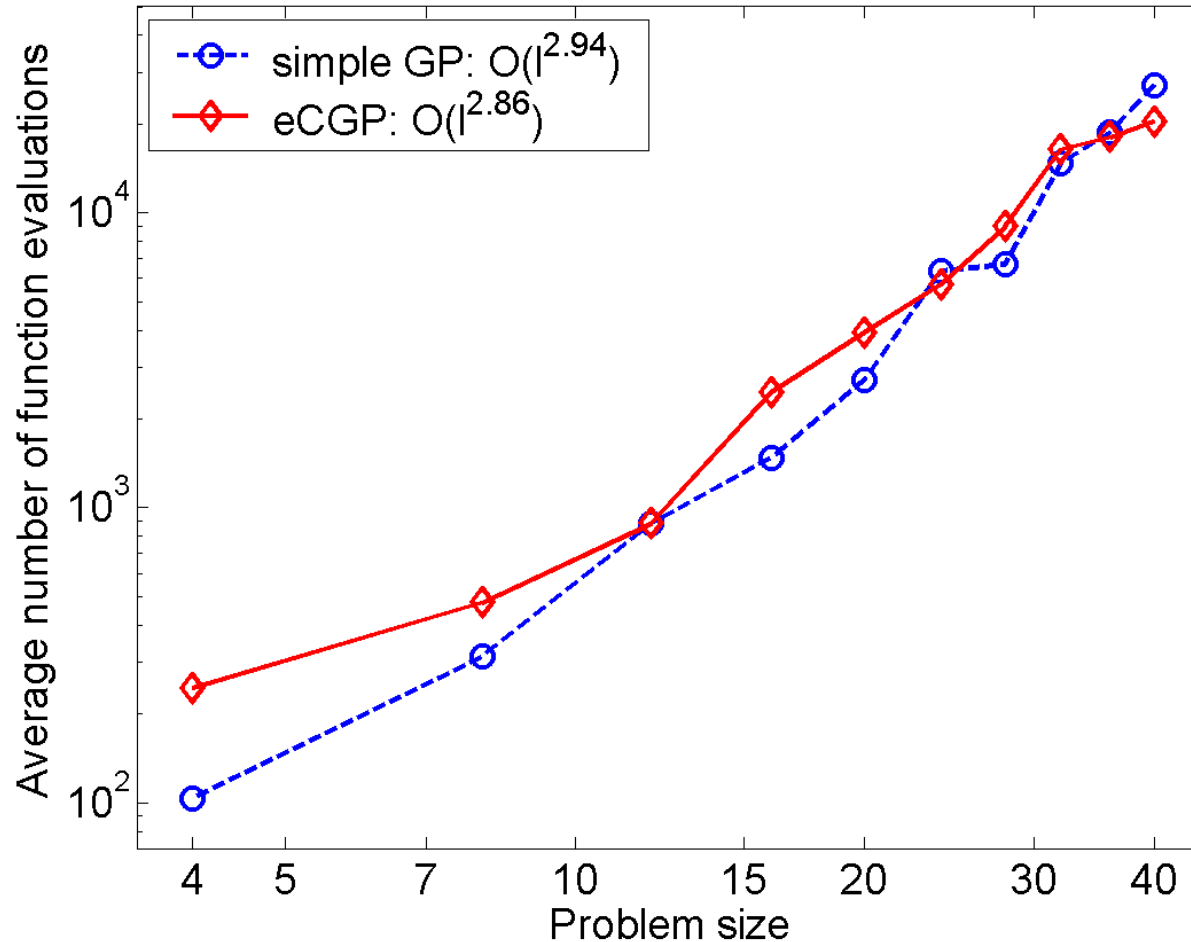


Expressed primitives

$\{X_1, \bar{X}_2, X_4\}$

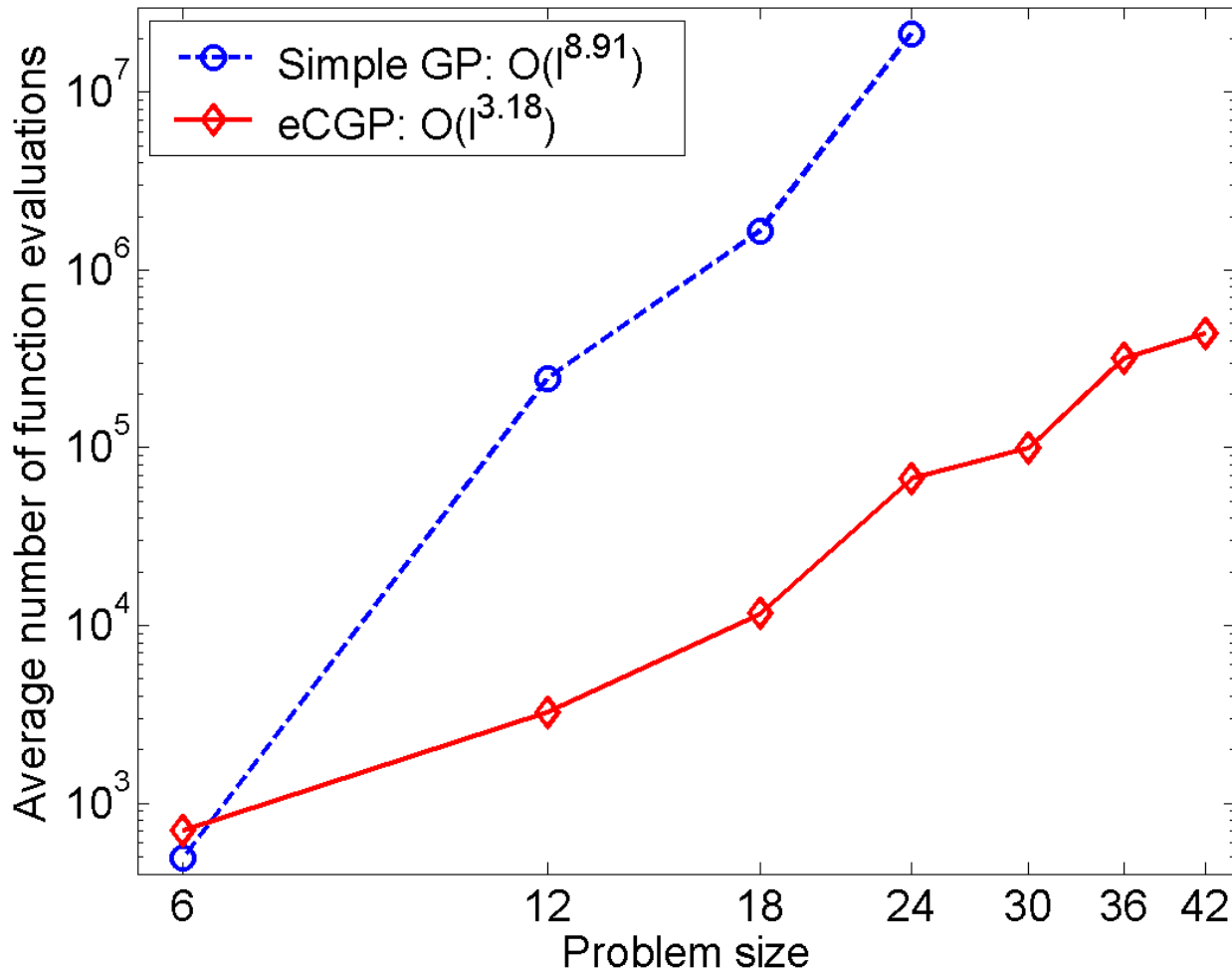


GP-Easy Problem: Competent GP vs. Simple GP



❖ Simple GP & eCGP: Cubic scale-up

GP-Hard Problem: eCGP vs. Simple GP



- ❖ Simple GP: Could not solve problems > 24 terminals
- ❖ eCGP: **Cubic** scale-up

Future Work

- ❖ Handle more complex variable interactions
 - ◆ Overlapping, and hierarchical building blocks
- ❖ Apply eCGP to different class of problems
 - ◆ Symbolic regression
 - ◆ Study scale-up behavior
- ❖ Convergence-time and population-sizing analysis
 - ◆ Theoretical and Empirical
- ❖ Extend eCGP to handle other variable types
 - ◆ Ephemeral random constants (ERCs)
 - ◆ Automatically defined functions (ADFs)

Summary & Conclusions

- ❖ Developed a competent genetic programming
 - ◆ Probabilistic model building GP
 - ★ Combine of eCGA and PIPE
 - ◆ Polynomial scale-up on a GP-hard problem
- ❖ Advantageous when linkage-learning is critical
- ❖ Do such problems exist in GP domain?
 - ◆ Building-block identification & exchange is critical
 - ◆ Broader issue of problem difficulty in GP
 - ★ Context, content, and structure
 - ◆ Optimization vs. System identification