Efficiency Enhancement of Probabilistic Model Building Genetic Algorithms

Kumara Sastry[1], David E. Goldberg[1] and Martin Pelikan[2]

[1]Illinois Genetic Algorithms Laboratory
University of Illinois at Urbana-Champaign

[4]Department of Mathematics and Computer Science
University of Missouri at St. Louis

Supported by AFOSR F49620-03-1-0129, NSF/DMR at MCC DMR-99-76550, NSF/ITR at CPSD DMR-0121695, DOE at FS-MRL DEFG02-91ER45439.

Motivation

- Probabilistic model building genetic algorithms:
  - Solve hard problems quickly, reliably, and accurately
  - Intractability → tractability
  - Polynomial (usually subquadratic) scalability
    - Daunting for complex problems
    - 1000 variables, 10 seconds/evaluation → ~120 days!
- Efficiency-Enhancement Techniques
  - Tractability → Practicality
- Evaluation relaxation via endogenous fitness-estimation model
- Local search in linkage neighborhoods

Outline

- Background and purpose
- Methodology
- Evaluation relaxation using an internal probabilistic fitness-estimation model
  - Methodology
  - Scalability and speed-up analysis
- Incorporation of a competent mutation operator
  - Methodology
  - Scalability and speed-up analysis
- Future work, summary and conclusions

Background and Purpose

- Efficiency Enhancement of GAs
  - Parallelization [Cantu-Paz, 2001]
  - Time utilization [Goldberg, 1999; Goldberg & Srivastava, 2001]
  - Hybridization [Davis, 1990; Goldberg & Voessner, 1999]
  - Evaluation relaxation [Sastry, 2002]
- Solve boundedly difficult problems in polynomial time
  - Propose two principled efficiency-enhancement techniques
- Use probabilistic models built by PMBGAs
  - Develop and use an inexpensive fitness-estimation model
  - Local search in the linkage neighborhood

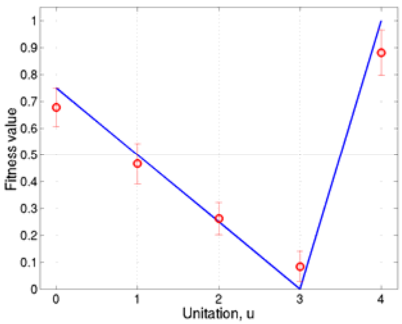
Evaluation Relaxation in PMBGAs

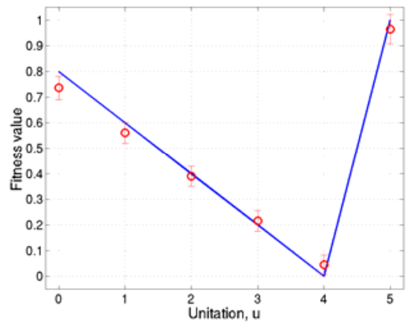
- Replace accurate, but expensive fitness evaluation with less-accurate, but inexpensive fitness estimation
  - Fitness inheritance [Smith *et al*, 1994; Sastry *et al*, 2000]
  - Simple method yields limited speed-up of 1.25
- Develop and use an internal probabilistic fitness model
  - Automatic discovery of fitness structure
  - Estimate sub-structure (building block) fitnesses
  - Individual fitness a function of sub-structural fitness
- Illustrate using extended compact genetic algorithm (eCGA)

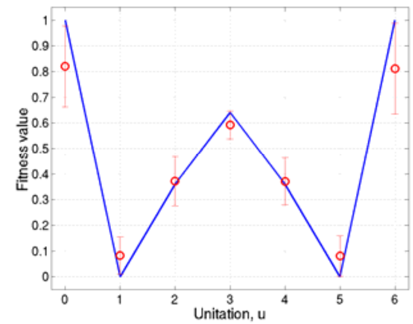
Internal Sub-Structural Fitness Model

- Identify key sub-structures of the search problem
- Estimate the fitness of sub-structure instances

Fitness of substructure instance **=** Average fitness of all evaluated individuals with the schema instance **−** Average fitness of all evaluated individuals in the population







- Individual fitness as a function of sub-structural fitnesses
  - Sum of fitness estimates of sub-structure instances
  - Can use other complex methods such as neural networks

Illustration Of Sub-Structural Fitness Estimation

- Use model-building procedure of extended compact GA
  - Partition genes into (mutually) independent groups

Model Structure	Metric
$[X_0]$ $[X_1]$ $[X_2]$ $[X_3]$ $[X_4]$ $[X_5]$ $[X_6]$ $[X_7]$ $[X_8]$ $[X_9]$ $[X_{10}]$ $[X_{11}]$	1.0000
$[X_0]$ $[X_1]$ $[X_2]$ $[X_3]$ $[X_4X_5]$ $[X_6]$ $[X_7]$ $[X_8]$ $[X_9]$ $[X_{10}]$ $[X_{11}]$	0.9933
$[X_0]$ $[X_1]$ $[X_2]$ $[X_3]$ $[X_4X_5X_7]$ $[X_6]$ $[X_8]$ $[X_9]$ $[X_{10}]$ $[X_{11}]$	0.9819
$[X_0]$ $[X_1]$ $[X_2]$ $[X_3]$ $[X_4X_5X_6X_7]$ $[X_8]$ $[X_9]$ $[X_{10}]$ $[X_{11}]$	0.9644
⋮	⋮
$[X_0]$ $[X_1]$ $[X_2]$ $[X_3]$ $[X_4X_5X_6X_7]$ $[X_8X_9X_{10}X_{11}]$	0.9273
⋮	⋮
$[X_0X_1X_2X_3]$ $[X_4X_5X_6X_7]$ $[X_8X_9X_{10}X_{11}]$	0.8895

- Use population data to estimate:
  - Probabilities: $p(X_0X_1X_2X_3)$, $p(X_4X_5X_6X_7)$, $p(X_8X_9X_{10}X_{11})$
  - Sub-structure fitnesses: $f(X_0X_1X_2X_3)$, $f(X_4X_5X_6X_7)$, $f(X_8X_9X_{10}X_{11})$

Scalability of Evaluation-Relaxation Scheme

- Error in fitness estimation modeled as additive Gaussian noise

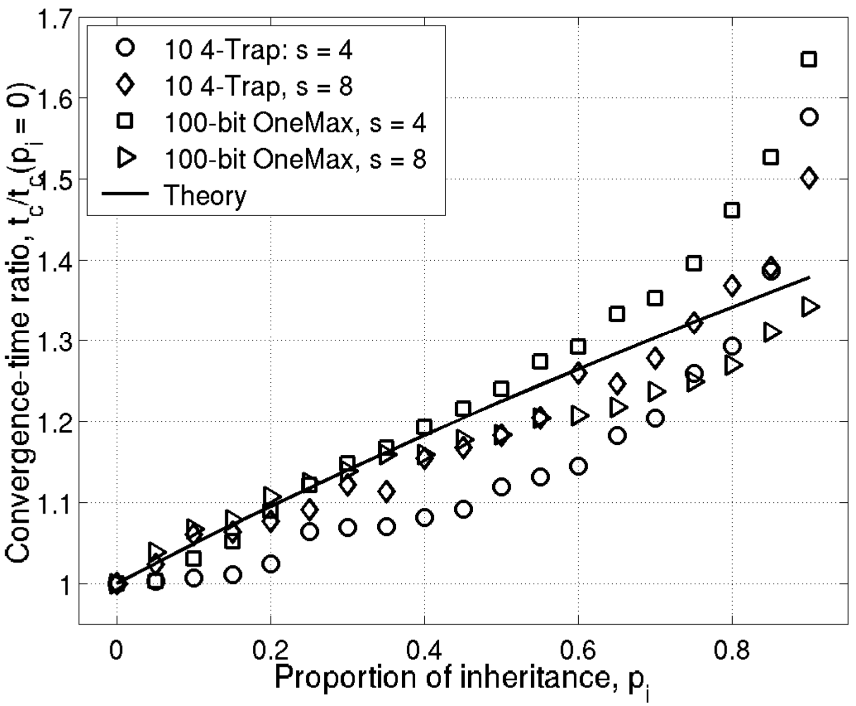
$$\mu_{\epsilon,t} = 0, \quad \sigma^2_{\epsilon,t} = p_i \sigma^2_{f,t}$$

- Convergence time: Selection-Intensity based models

[Mühlenbein & Schlierkamp-Voosen, 1993; Thierens & Goldberg 1993; Bäck, 1994; Miller & Goldberg, 1995 & 1996; Sastry & Goldberg, 2002]

$$t_{c,r} = \frac{t_c}{t_{c,o}} \approx \sqrt{1 + p_i}$$

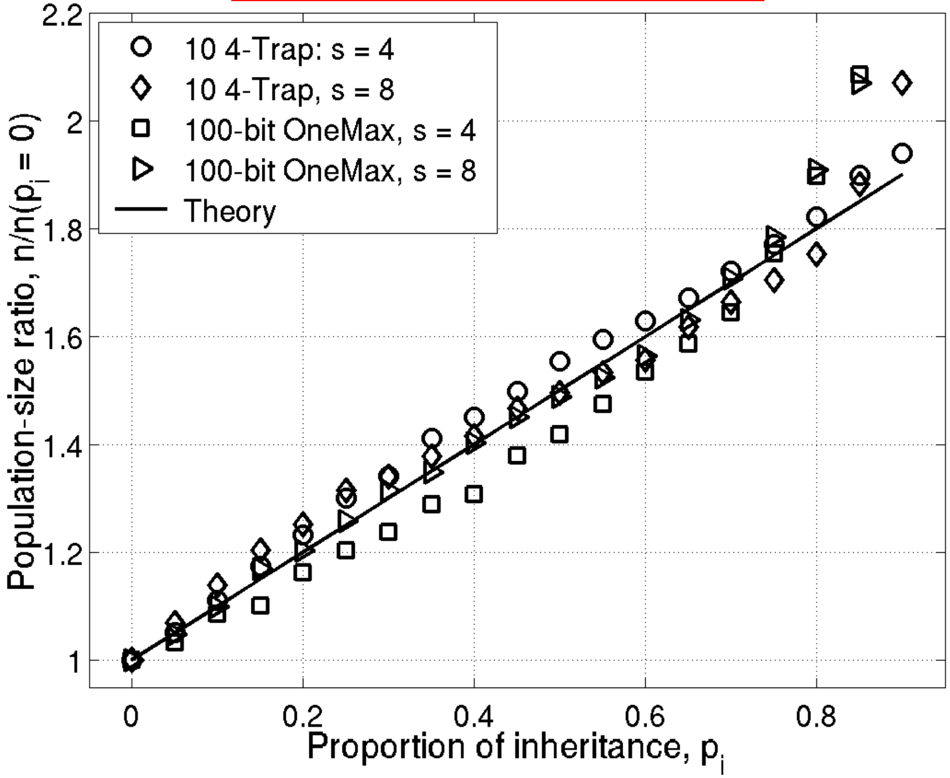
- Neglected cumulative effects of fitness estimation



Scalability of Evaluation-Relaxation Scheme

- Population Sizing Model [Goldberg *et al*, 1992; Harik *et al*, 1997; Pelikan *et al*, 2001]
  - BB supply,
  - Decision-making, and
  - Accurate model building

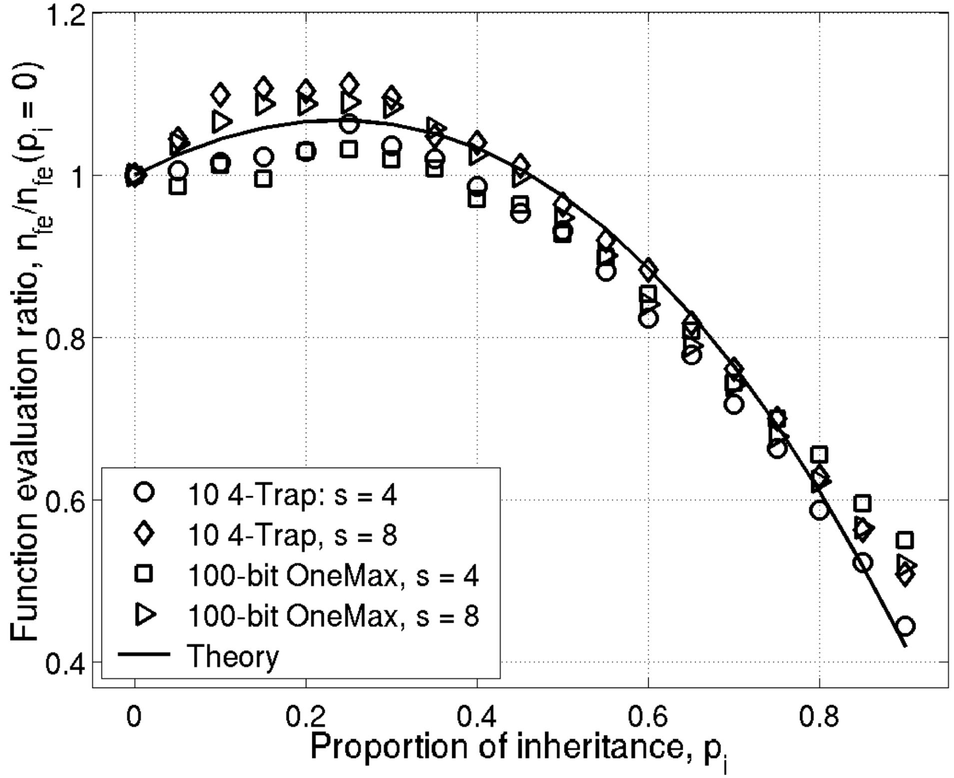
$$n_r = \frac{n}{n_o} \approx 1 + p_i$$



Scalability Of Evaluation-Relaxation Scheme

- **Scalability:** # function evaluations required to obtain a solution of quality 1-1/m
  - $1-p_i$ individuals evaluated
  - Neglected the cost of building and using the fitness-estimation model

$$n_{fe,r} \approx (1-p_i)(1+p_i)^{1.5}$$



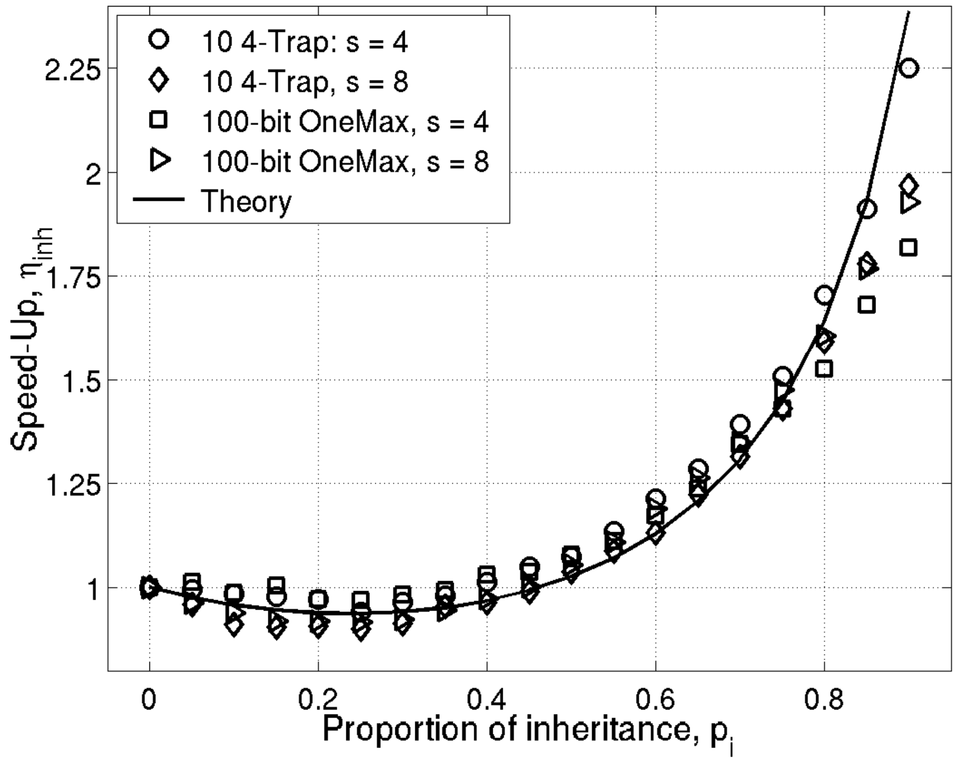
Scalability Of Evaluation-Relaxation Scheme

- Speed-Up: Ratio of # function evaluations with evaluation relaxation to that without it

- Only 1-15% individuals need evaluation
  - Optimal $p_i \sim 0.85$-0.99

- Speed-Up: 1.75-53

$$\eta \approx (1 - p_i)^{-1} (1 + p_i)^{-1.5}$$

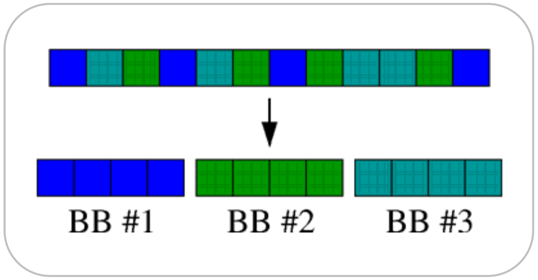


Designing A Competent Mutation

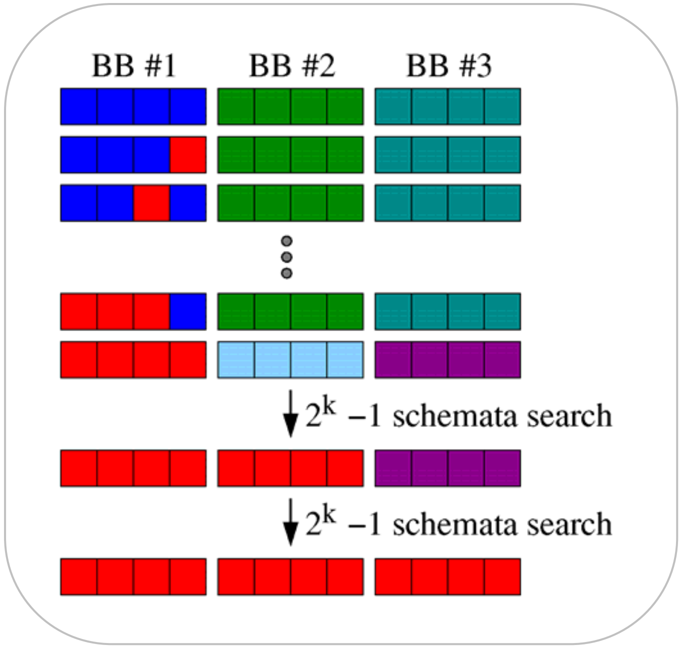
- Systematic induction of good neighborhood information
  - Based on a population of promising candidate solutions
- Polynomial (quadratic) scalability
- Perform scalability analysis of the proposed algorithm
  - Derive and use *facetwise* models
  - Verify on nearly decomposable problems of bounded difficulty
    - Additively separable problems
  - Compare with a competent selectorecombinative GA

Designing A Competent Mutation Operator

- Inducing good neighborhoods
  - Sample of candidate solutions
  - Use linkage-learning procedures developed for selectorecombinative GAs



- Neighborhood: Space vs. Sub-space
- Search: Hillclimbing vs. Random
  - Consider linkage partitions
    - Arbitrary left-to-right order
  - Choose the best schemata
    - Among the 2^k possible ones



How to Induce Neighborhoods?

- Induce good neighborhoods as linkage groups
- Use model-building procedure of extended compact GA
  - Partition genes into (mutually) independent groups
  - Start with the lowest complexity model
  - Search for a least-complex, most-accurate model

Model Structure	Metric
$[X_0]$ $[X_1]$ $[X_2]$ $[X_3]$ $[X_4]$ $[X_5]$ $[X_6]$ $[X_7]$ $[X_8]$ $[X_9]$ $[X_{10}]$ $[X_{11}]$	1.0000
$[X_0]$ $[X_1]$ $[X_2]$ $[X_3]$ $[X_4 X_5]$ $[X_6]$ $[X_7]$ $[X_8]$ $[X_9]$ $[X_{10}]$ $[X_{11}]$	0.9933
$[X_0]$ $[X_1]$ $[X_2]$ $[X_3]$ $[X_4 X_5 X_7]$ $[X_6]$ $[X_8]$ $[X_9]$ $[X_{10}]$ $[X_{11}]$	0.9819
$[X_0]$ $[X_1]$ $[X_2]$ $[X_3]$ $[X_4 X_5 X_6 X_7]$ $[X_8]$ $[X_9]$ $[X_{10}]$ $[X_{11}]$	0.9644
⋮	⋮
$[X_0]$ $[X_1]$ $[X_2]$ $[X_3]$ $[X_4 X_5 X_6 X_7]$ $[X_8 X_9 X_{10} X_{11}]$	0.9273
⋮	⋮
$[X_0 X_1 X_2 X_3]$ $[X_4 X_5 X_6 X_7]$ $[X_8 X_9 X_{10} X_{11}]$	0.8895

Algorithmic Description of Selectomutative GA

1. Initialize the population (usually random initialization)

2. Evaluate the fitness of individuals

3. Select promising solutions (e.g., tournament selection)

4. Build the probabilistic model
   - Optimize the structure and parameter of the probabilistic model to best fit the selected individuals
   - Automatic discovery linkage neighborhoods

5. Apply enumerative BB-wise mutation
   - Search for the best building block in each partition

Scalability of Competent Mutation Operator

- Population size required for accurate modeling of neighborhood information [Pelikan, Sastry, and Goldberg, 2002]:

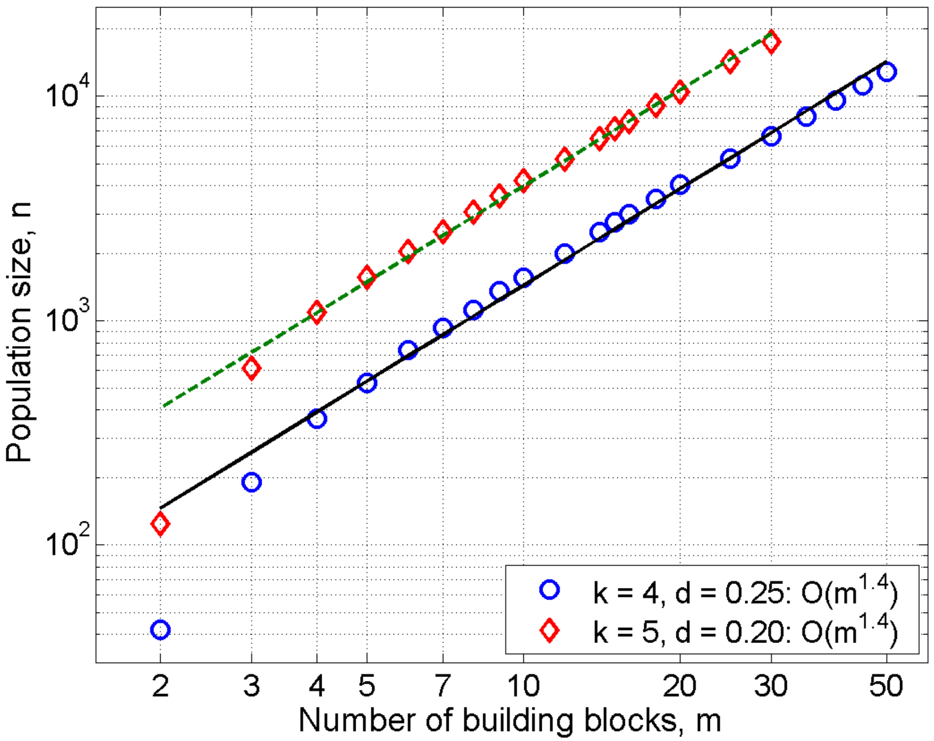
$$\mathcal{O}\left(2^k \cdot m^{1.05}\right) \leq n \leq \mathcal{O}\left(2^k \cdot m^{2.1}\right)$$

- Empirical verification:

$$n = \mathcal{O}\left(2^k \cdot m^{1.5}\right)$$

Sub-Component size (BB size)

Components (# BBs)



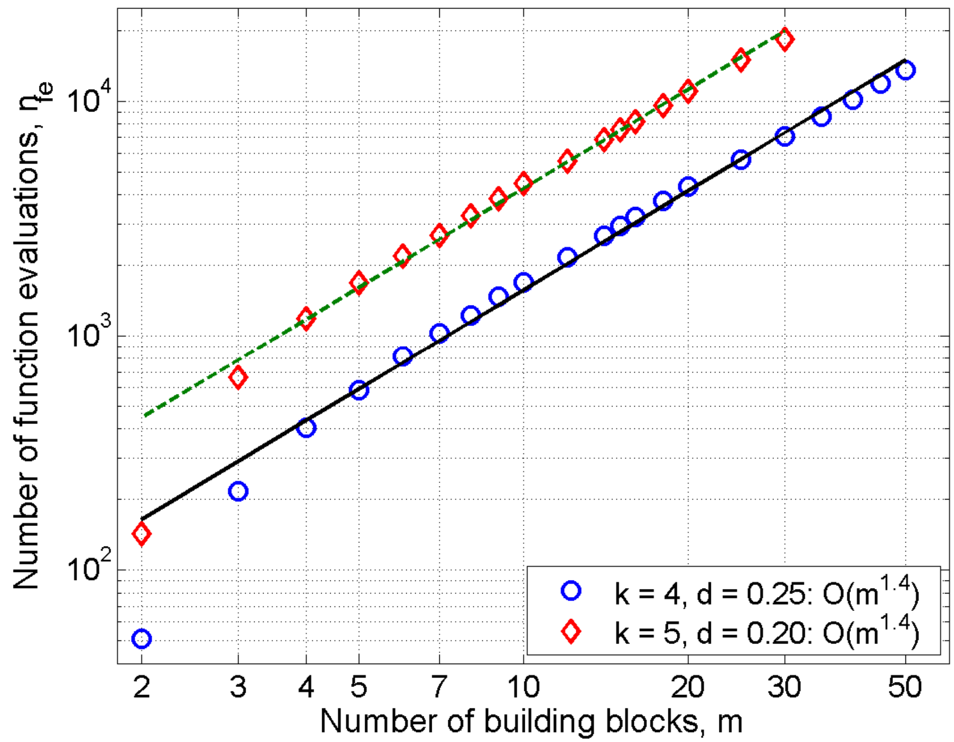
Scalability of Competent Mutation Operator

- Model building cost: $$\mathcal{O}\left(2^k \cdot m^{1.05}\right) \leq n_{fe} \leq \mathcal{O}\left(2^k \cdot m^{2.1}\right)$$

- BB-wise enumeration cost: $$n_{fe} = \left(2^k - 1\right) m = \mathcal{O}\left(2^k \cdot m\right)$$

- Empirical verification

$$n_{fe,M} = \mathcal{O}\left(2^k \cdot m^{1.5}\right)$$



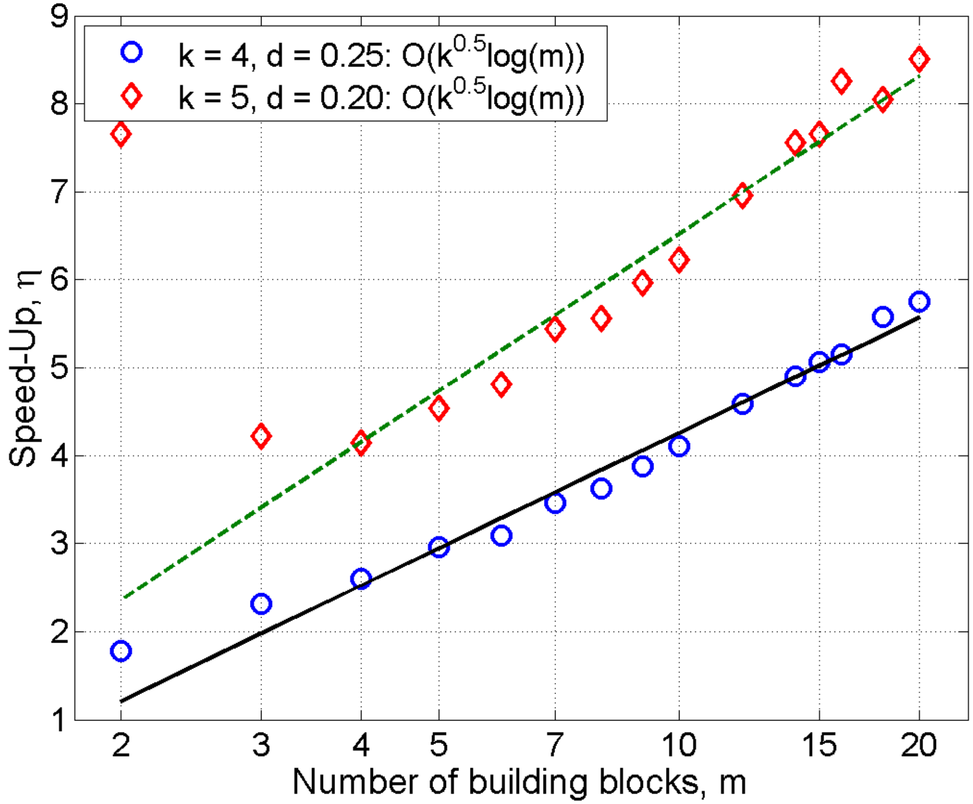
Competent Crossover vs. Competent Mutation

Competent mutation:

$$\mathcal{O}\left(2^k m^{1.5}\right)$$

Competent crossover:

$$\mathcal{O}\left(2^k m^{1.5} \log m\right)$$



- Speed-Up: Scalability ratio of mutation to that of crossover

$$\eta = \frac{n_{fe,M}}{n_{fe,XO}} = \mathcal{O}\left(\sqrt{k} \log m\right)$$

Future Work

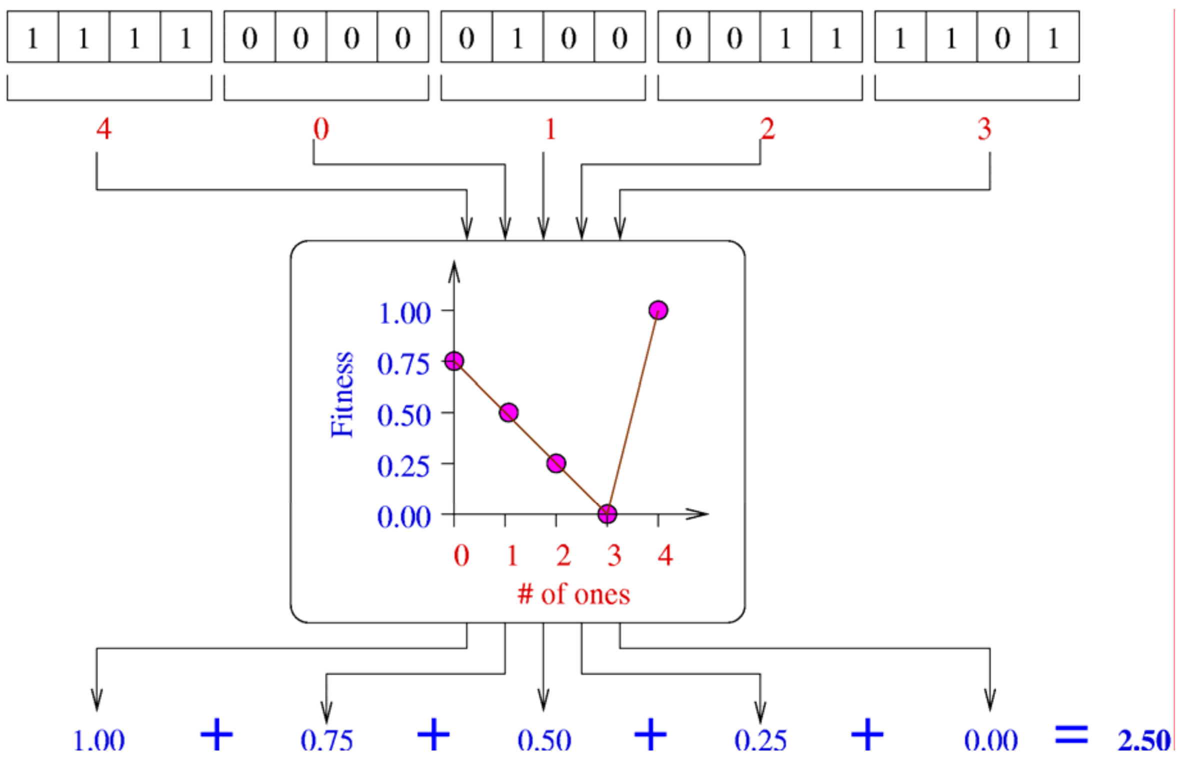
- Hybridization of competent crossover *and* mutation
  - Consider a hybrid GA with *oscillating* population
- Problems with overlapping building blocks
  - Effect is similar to that of exogenous noise
  - Speed-Up likely to be lower
  - More complex model-building schemes (E.g., BOA, DSMGA)
- Problems with non-uniform salience
  - Account for sequentiality
- Hierarchical problems
  - Account for effect of niching

Summary & Conclusions

- Evaluation relaxation using internal fitness model
  - Uses sub-structural fitness estimates as basis
  - Only 1-15% of the individuals need fitness evaluation
  - Yields a speed-up of 1.75-53
- Competent mutation operator:
  - Systematic induction of good neighborhoods
  - Neighborhood: Entire search space → Linkage subspace
  - Search: Hillclimbing → Random/Enumerative
  - Subquadratic scalability: $O(2^k m^{1.5})$
    - $e^k m^k$ scalability of simple mutation [Mühlenbein, 1991]

Test Problems: m-k Deceptive Trap Functions

- Building block identification is critical to GA success
- *m* concatenated deceptive trap functions [Ackley, 1987; Goldberg, 1987; Deb & Goldberg, 1993]



Algorithmic Description of eCGA

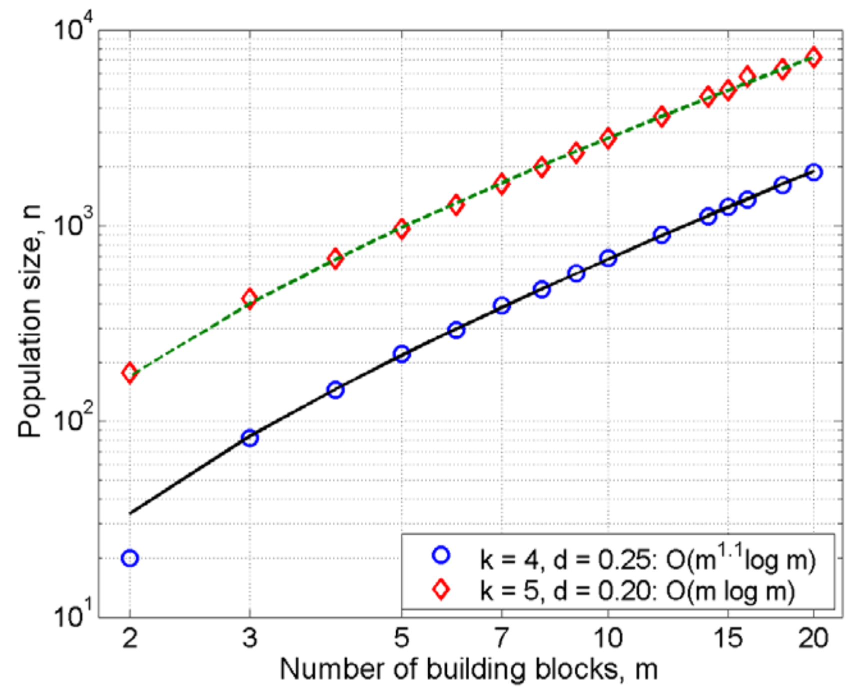
1. Initialize the population (usually random initialization)
2. Evaluate the fitness of individuals
3. Select promising solutions (e.g., tournament selection)
4. Build the probabilistic model
   - Optimize the structure and parameter of the probabilistic model to best fit the selected individuals
   - Automatic identification of building blocks
5. Sample the model to create new candidate solutions
   - Effective exchange of building blocks
6. Repeat steps 2-7 till some convergence criteria are met

Scalability of eCGA

- **Population sizing:** [Pelikan, Goldberg, Cantú-Paz, 2000; Pelikan, Sastry, & Goldberg, 2003]
  - Accounts for BB supply, decision-making, and model accuracy requirements

$$n = -\log(\alpha) \cdot \left(\frac{\sigma_{bb}}{d}\right) \cdot 2^k \cdot m$$

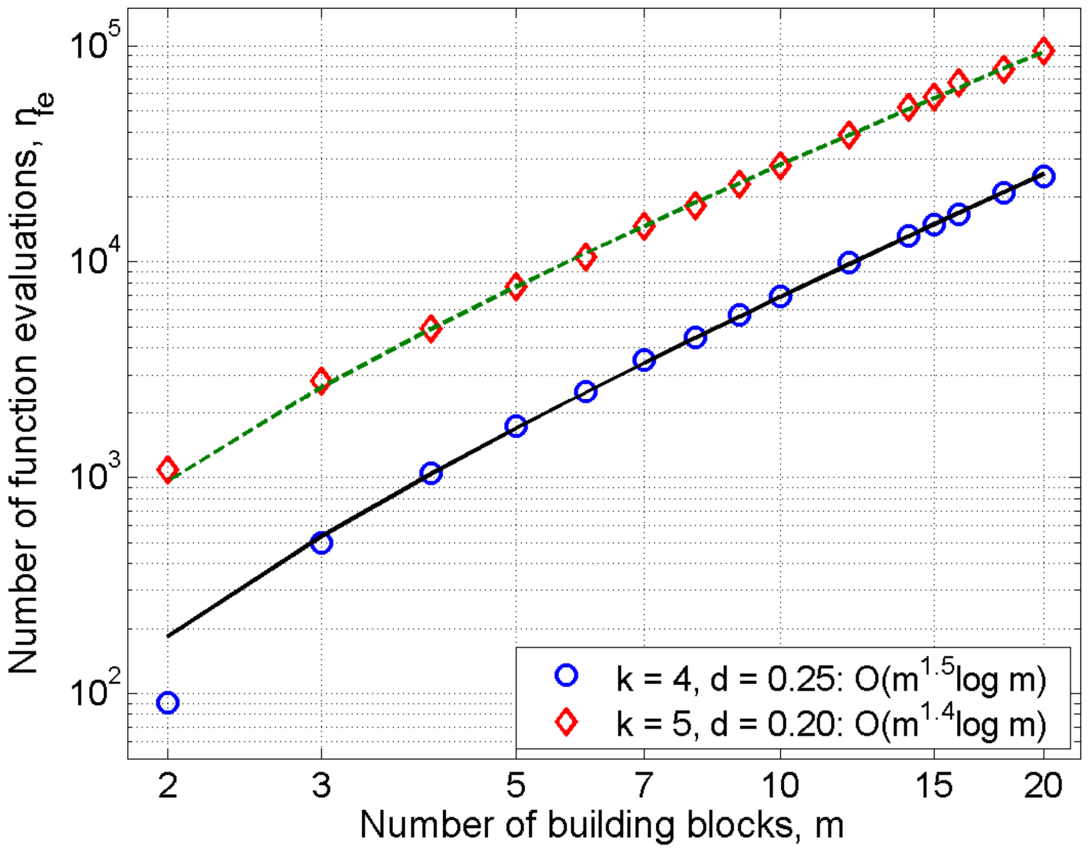
- BB fitness variance (σ_{bb})
- Signal between competing BBs (d)
- Failure probability (α)
- Sub-Component size (BB size) (k)
- Error tolerance (α)
- Signal-to-Noise ratio (σ_{bb}/d)
- # Competing sub-components (2^k)
- # Components (# BBs) (m)



Scalability of eCGA

- Number of function evaluations:
  - Error probability α = 1/m

$$
\begin{aligned}
n_{fe,XO} &\propto \left(\frac{\sigma_{bb}}{d}\right) \cdot 2^k \cdot m^{1.5} \log m \\
&= \mathcal{O}\left(2^k m^{1.5} \log m\right)
\end{aligned}
$$



Related Work

- Mutation operators in genetic algorithms
  - Random walk locally around individuals
- Mutation operators in evolution strategies [Rechenberg, 1973; Schwefel, 1977; Bäck, 1996; Beyer, 1996; Hansen & Ostermeier, 2001]
  - Neighborhood information based on entire search space
- Local search literature [Barnes *et al*, 2003; Watson, 2003; Vaughn *et al*, 2000; Armstrong & Jacobson, 2004; Glover & Laguna, 1997]
  - Emphasis on good *neighborhood* operators
  - Lack systematic methods for inducing good neighborhoods
- Mutation with known linkage neighborhoods [Sastry & Goldberg, 2004]
  - $e^k m^k$ scalability → $2^k m^2$ scalability
  - Neglected the cost of inducing linkage neighborhoods