The University of Nottingham

The Tree-String Problem

Steven Gustafson

smg@cs.nott.ac.uk

www.cs.nott.ac.uk/~smg

The Problem

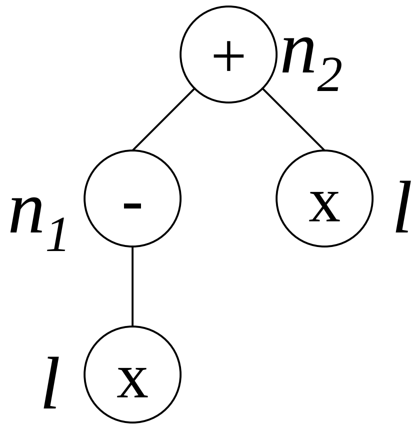
- Define
  - properties of target solution structure, and
  - properties of target solution contents
- Use
  - mapping from solution to a structure representation and a content representation
  - measure of similarity between candidate solution and target solution
- Goal
  - understand how easy/hard instances are for algorithm
  - understand how algorithm behaves on instances

The Problem for GP

- Solution structure found in tree shapes
- Solution content defined by tree node contents
- Structure representation
  – emphasise hierarchical nature of structure
  – e.g. breadth-first tree traversal
- Content representation
  – emphasise juxtaposition of contents
  – e.g. depth-first tree traversal
- Similarity
  – compatible with representations
  – e.g. with above, longest common substring

TS for GP : Details

- Ξ = structure set, e.g. $\{n_2 n_1, l\}$
- Ψ = content set, e.g. {+,-,x}
- t = instance from Ξ, Ψ
- α(t) maps to structure string (Ξ*)
- γ(t) maps to content string (Ψ*)
- δ(s1,s2) = longest common substring



$\alpha(t) = n_2\, n_1\, ll$

$\gamma(t) = x\text{-}{+}x$

$\Pi = (\Xi, \Psi, t, \alpha, \gamma, \delta)$

Similarity $\delta()$

- Compatible with structure and content representations
- Dependent on research objectives
- Longest common substring provides some rigidity to flexibility allowed by similarity and string representations of structure and content

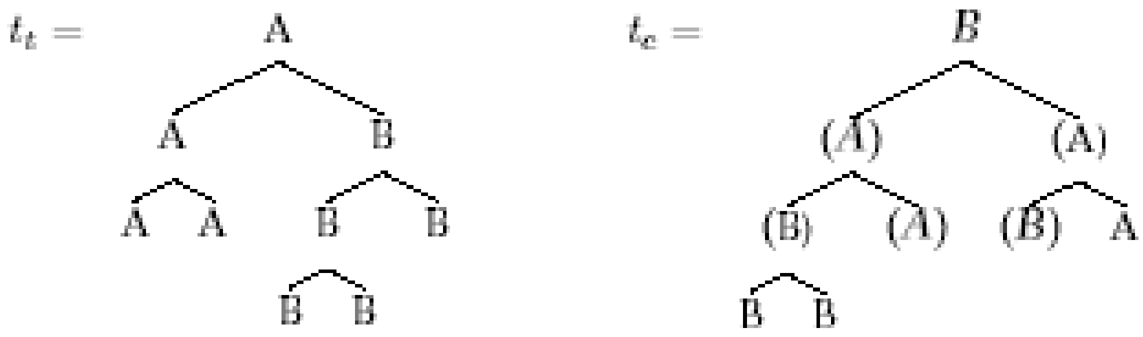
Selection

- Find structure and content similarity
  - $\delta(\alpha(t), \alpha(c)) \rightarrow i$
  - $\delta(\gamma(t), \gamma(c)) \rightarrow j$
- Use a multi-objective selection strategy

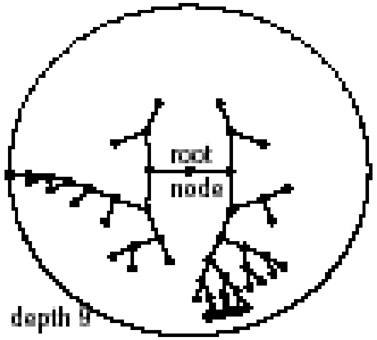
- Pareto criterion
  - better-then( ) = better in one objective, not worse in the other
  - equal( ) = equal in both, or better in one AND worse in other

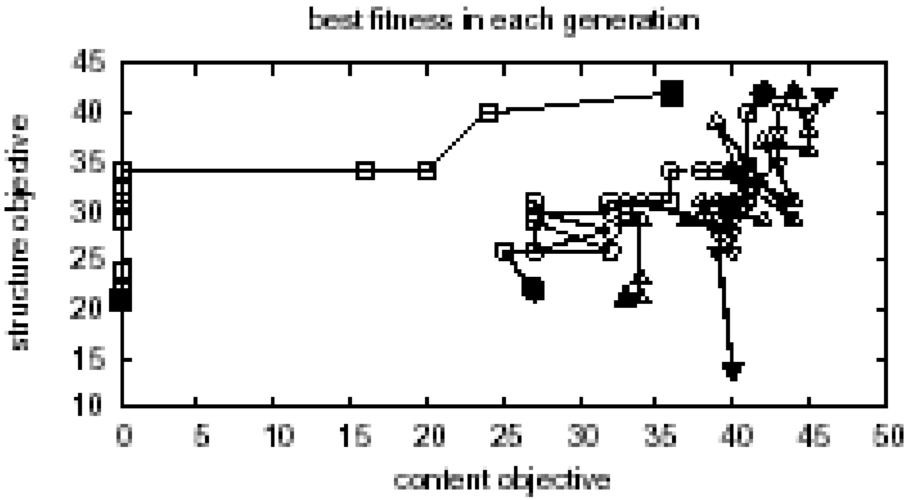
Examples

- $\delta(\alpha(t_t), \alpha(t_e))$ =LCS($\underline{nnnll}nlll$, $n\underline{nnnll}lll$) = 5 and
- $\delta(\gamma(t_t), \gamma(t_e))$ =LCS($AA\underline{AABB}BBB$, $BBB\underline{AABB}AA$) = 4.



Examples





$$\alpha(t) = nnnnnnnllnnnnnllllnnlnlnnnnnnlllnllllnnnllnlllllnll$$

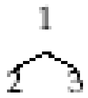
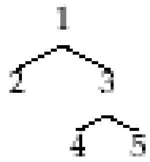
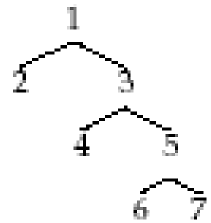
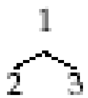
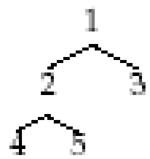
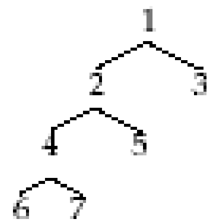
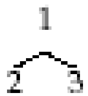
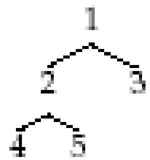
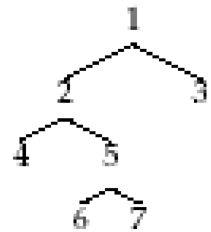
$$\Psi_1 = \mathrm{AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA}$$

$$\Psi_2 = \mathrm{BBAABABAAABBBBBAABBAABAABAAABBBAAAAAAAABBBBAABABAAAAA}$$

$$\Psi_3 = \mathrm{BBACABABBCABCBBCABBBCACABCACAABCBBACABACBCCCCCBACAB}$$

$$\Psi_4 = \mathrm{CBBADADBCCABABBBCBDCABCBBBCCDCBBDAADDCCBAACDADCCBCA}$$

Reduced Conflict Instances

Trees			
Content	2 1 3	2 1 4 3 5	2 1 4 3 6 5 7
Structure	n l l	n l n l l	n l n l n l l
Trees			
Content	2 1 3	4 2 5 1 3	6 4 7 2 5 1 3
Structure	n l l	n n l l l	n n l n l l l
Trees			
Content	2 1 3	4 2 5 1 3	4 2 6 5 7 1 3
Structure	n l l	n n l l l	n n l l n l l

Parity Instances

The University of Nottingham

A Tree-String problem is then designed where:

- $\Xi = \{n_2, n_1, l\}$, where n_2 has two arguments and n_1, one, representing the two argument functions AND, NAND, OR and NOR and the single argument NOT function.
- $\Psi = \{d1, \ldots, dN, F\}$, where $d1, \ldots, dN$ are leaf contents and F is used for all non-leaf node contents.
- Tree shapes are generated of size m of various shapes.
- Content strings are generated such that an even distribution of labels $(d1, \ldots, dN)$ are placed in positions representing leaf nodes in the tree shape, and all node positions with an F label.

Motivations

The University of Nottingham

- Recent research into structural mechanisms and content/context conflicts
  - find the right content
  - put content into correct order
  - some content likely to prefer certain structure
- Study interdependencies between solution structure and content

Motivations (2)

- Inter-play between Structure and Content
  - implicit presence in search process
  - Tree-String attempts to make it more explicit
- Bridge gap between easy-to-analyse constructed problems and more-real world, difficult-to-analyse problems
  - flexibility to design instances, similarity, etc.
  - random instances that create tunable GP search

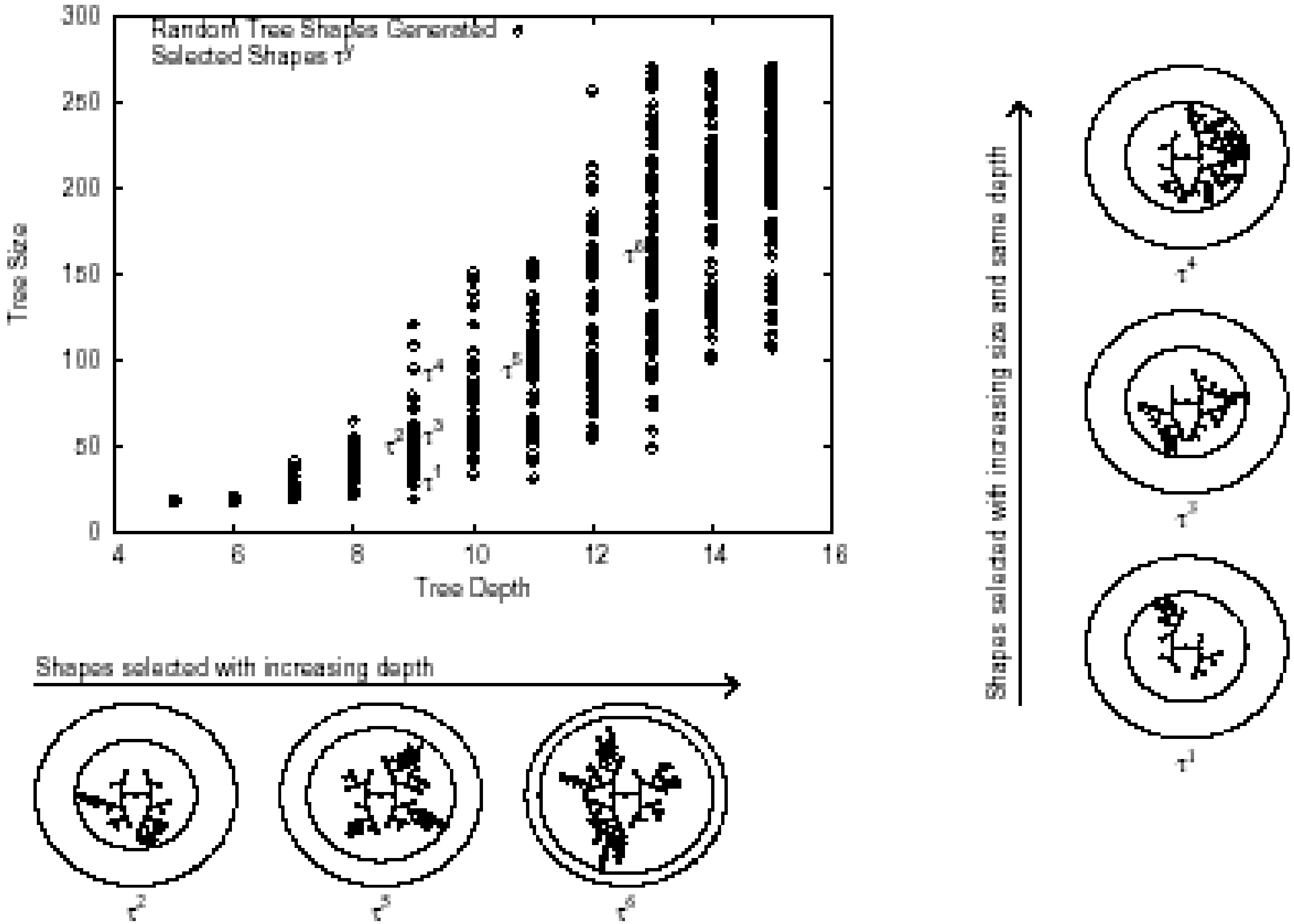
Difficulty Study

- Given some random tree shapes or various depths and size,
- How does GP search behave when an increasing number of content symbols are required?
- Can do a lot of other things:
  - find `hard’ content strings
  - find `hard’ tree shapes

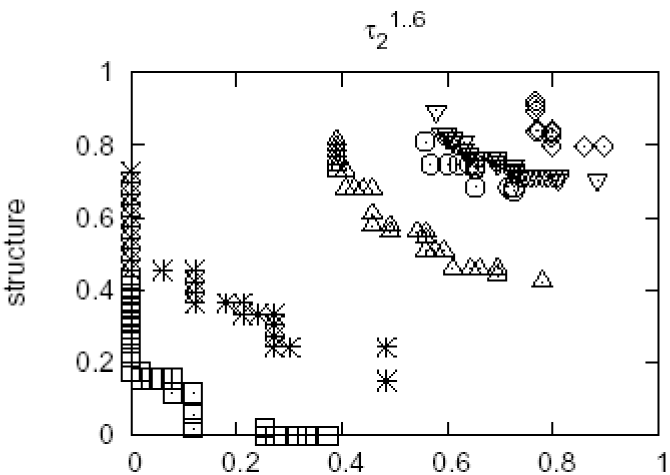
GP System

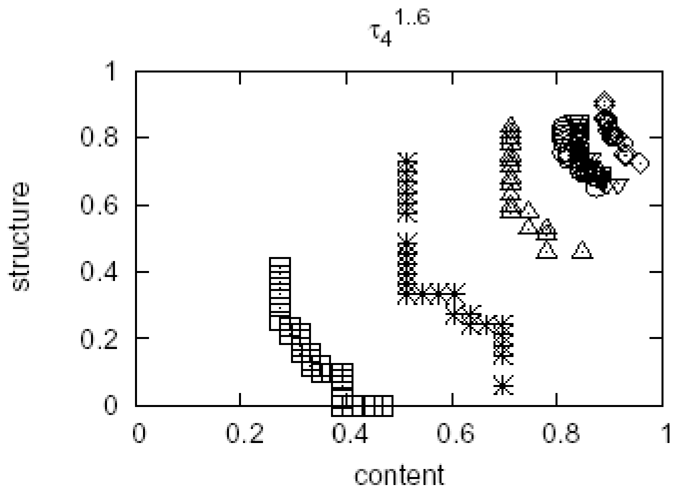
- Standard, generational GP algorithm
- Population size is small (50)
- Trees are bounded to depth 17
- Only subtree crossover is used
- Create many instances (t) consisting of a string representing tree shape and a string representing tree contents

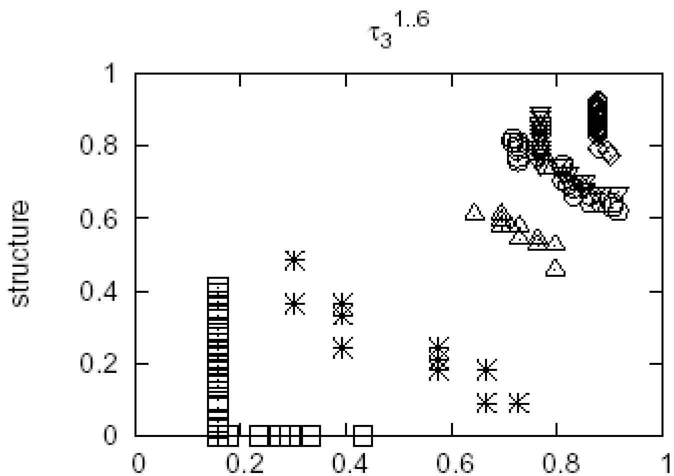
TS Instances



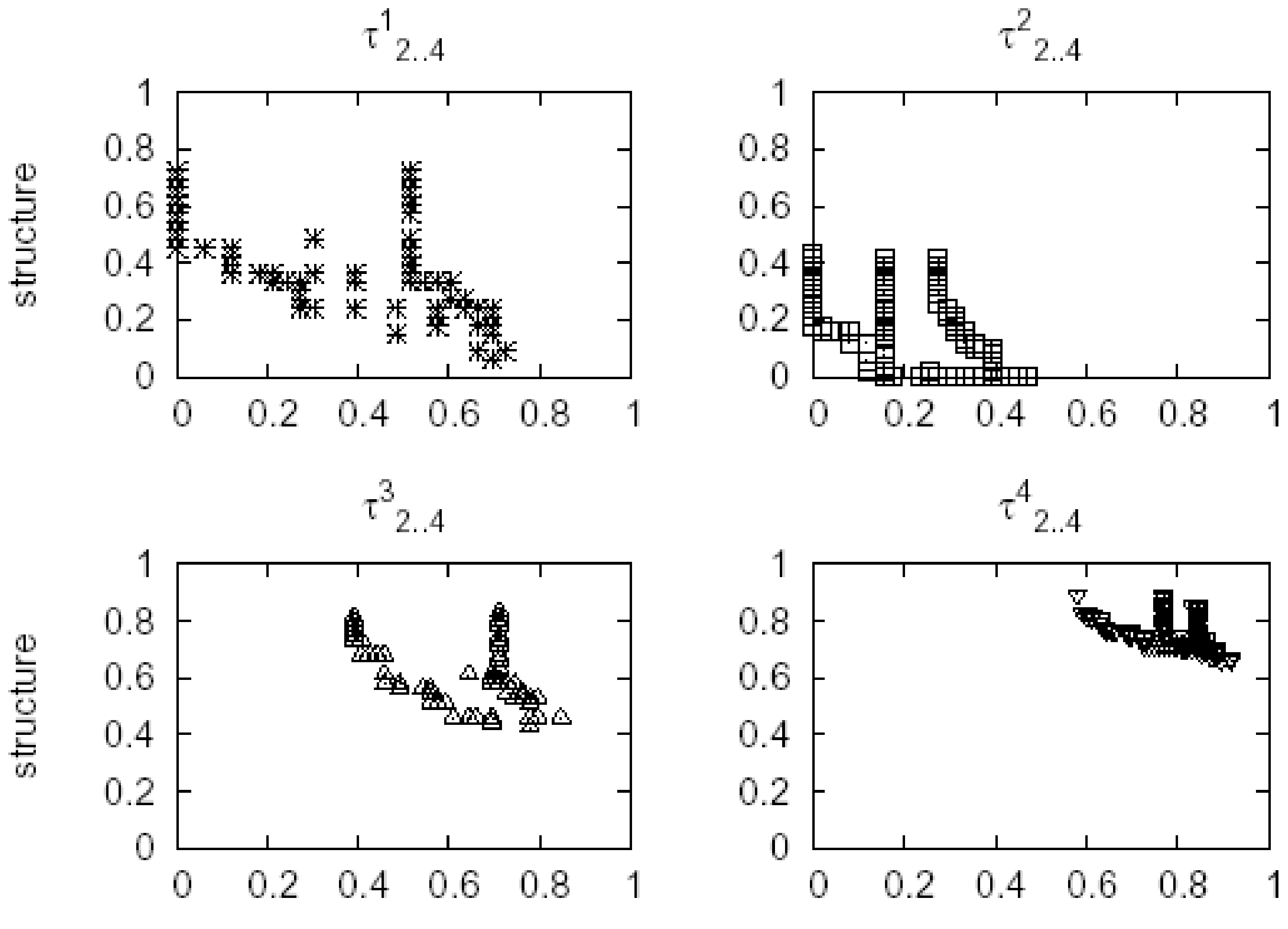
Results - Content vs. Structure



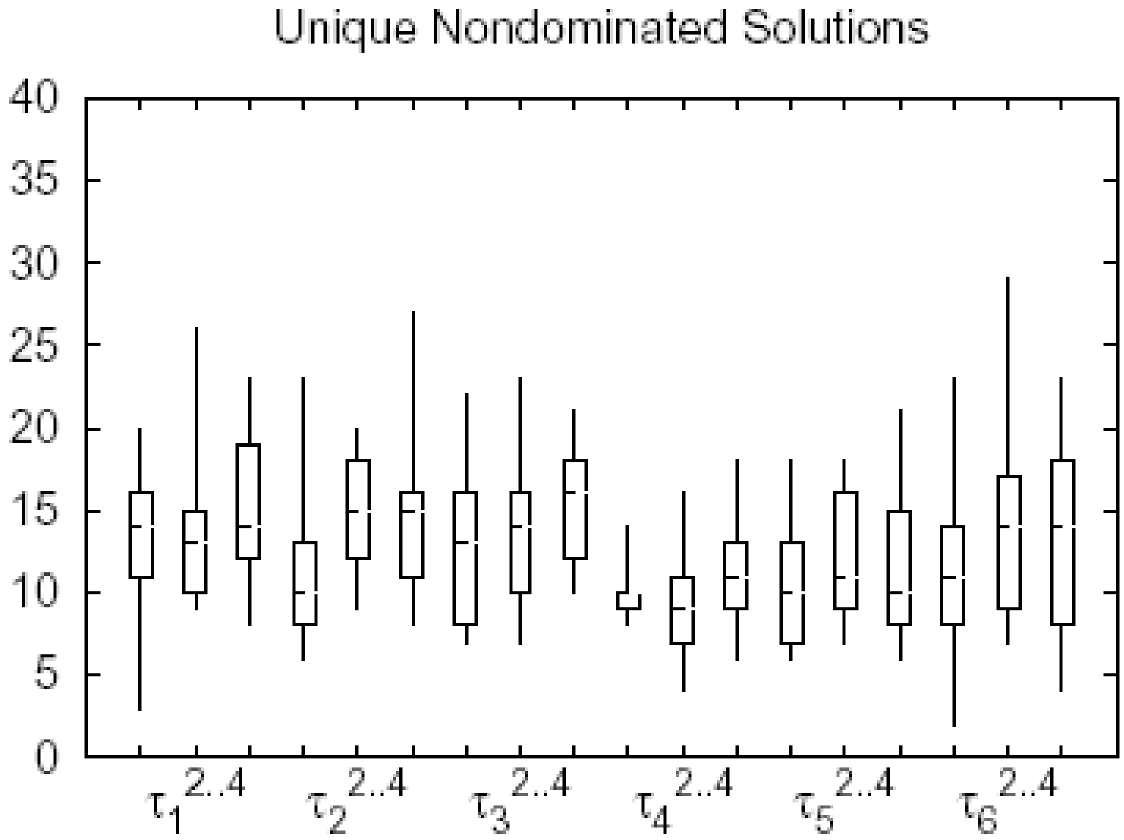




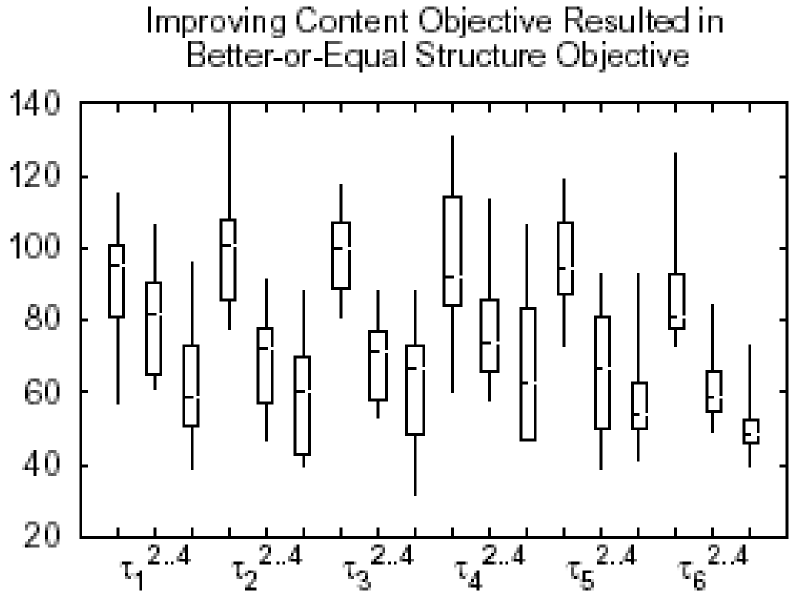
Scaling Effects



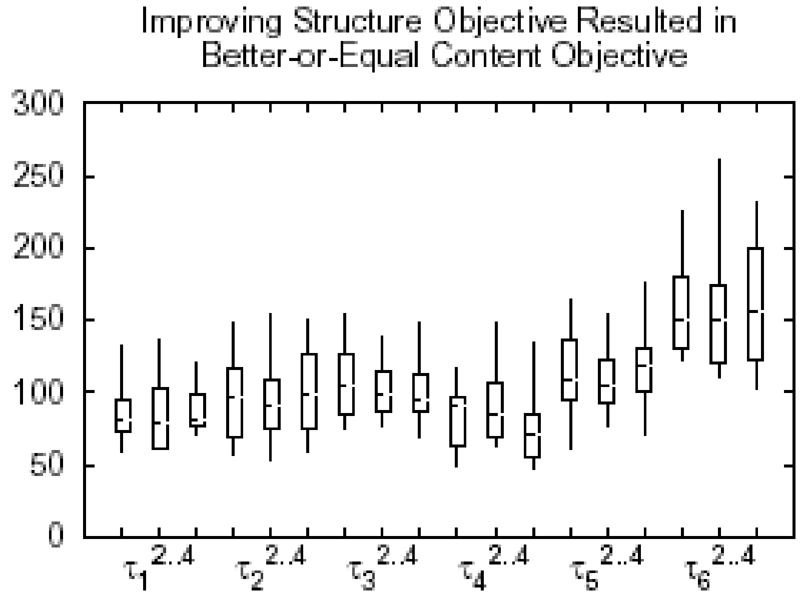
Nondominated Solutions



Conflicts

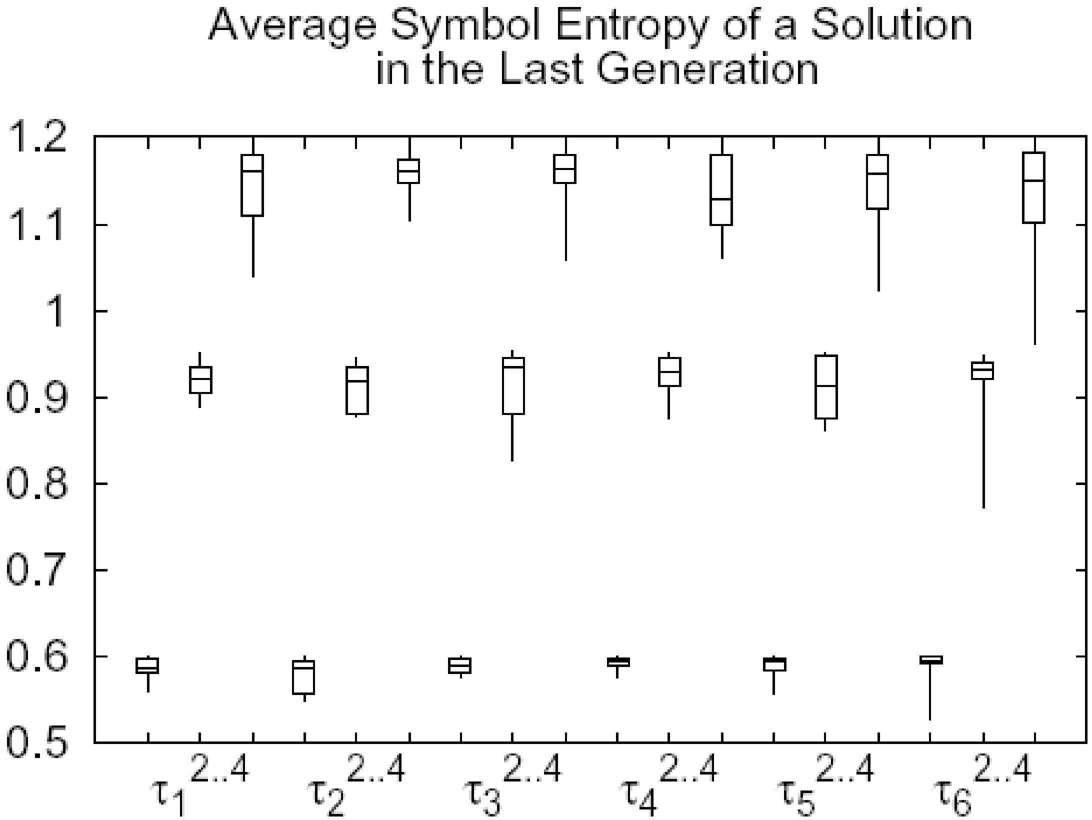


Improving Content Objective Resulted in
Better-or-Equal Structure Objective
140
120
100
80
60
40
20
$\tau_1^{2..4}$ $\tau_2^{2..4}$ $\tau_3^{2..4}$ $\tau_4^{2..4}$ $\tau_5^{2..4}$ $\tau_6^{2..4}$



Improving Structure Objective Resulted in
Better-or-Equal Content Objective
300
250
200
150
100
50
0
$\tau_1^{2..4}$ $\tau_2^{2..4}$ $\tau_3^{2..4}$ $\tau_4^{2..4}$ $\tau_5^{2..4}$ $\tau_6^{2..4}$

Symbol Entropy

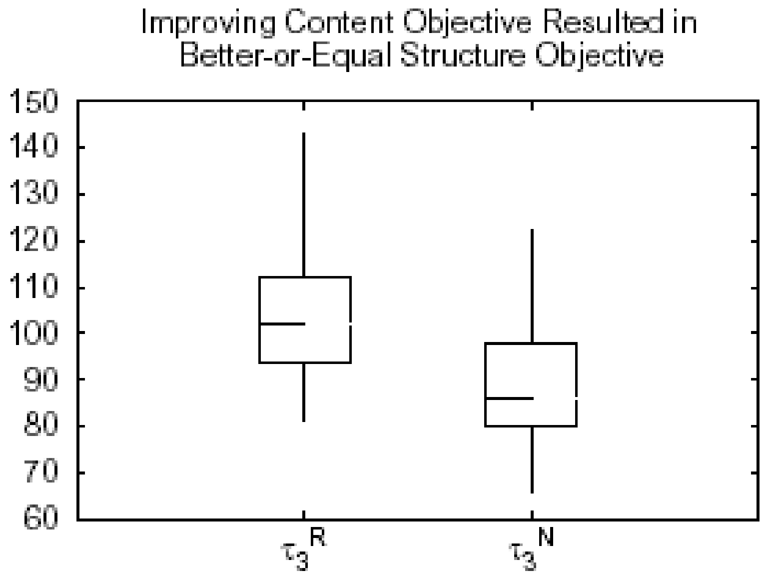


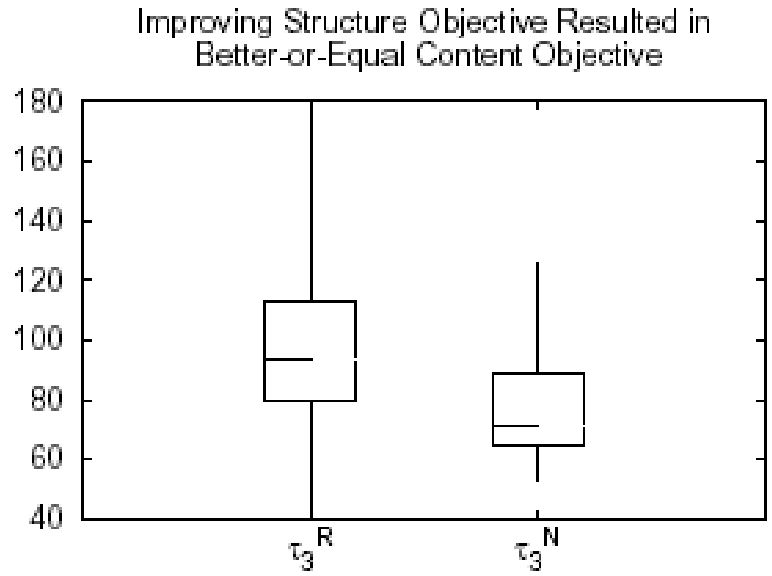
2.408

1.431

0.602

Easy Instances





Other Work

- Tree-String problem used in:
  - study of various GP algorithm features
  - study of operator-based distance (simple version)
  - conceptual problems for automated design of systems self-assembly
- Online algorithms and code
- Mappings to other domains

Conclusions

- Presenting the Tree-String problem
- Showed examples of instantiation for GP
- Demonstrated its use for GP difficulty

- Comments/Discussion/Questions:

smg@cs.nott.ac.uk