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

- $\Xi$ = structure set, e.g. $\{n_2n_1,l\}$
- $\Psi$ = content set, e.g. $\{+,-,x\}$
- $t$ = instance from $\Xi, \Psi$
- $\alpha(t)$ maps to structure string ($\Xi^*$)
- $\gamma(t)$ maps to content string ($\Psi^*$)
- $\delta(s_1,s_2)$ = longest common substring

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

\[ \delta(\alpha(t_b), \alpha(t_c)) = LCS(\text{nllnnllll}, \text{nllllllll}) = 5 \text{ and} \]

\[ \delta(\gamma(t_b), \gamma(t_c)) = LCS(\text{AAAAABBBBB}, \text{BBBBAAAA}) = 4. \]
Examples

\[ \alpha(t) = nnnnnnnnllnnnnllllnnllllnnllllnnllllnnllllnnllllnnllll
\]

\[ \Psi_1 = \text{AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA} \]

\[ \Psi_2 = \text{BBAABABAABBBBAAABABAABAABBAABAAAAABBBAAABABAABAAA} \]

\[ \Psi_3 = \text{BBACABABBBCBCCABBBBBACABCAACABCAABCBBACACBACBCCCCBACAB} \]

\[ \Psi_4 = \text{CBBBADADBCCABABBBBDCABCBBCDCBBDAADDCCBAACDADCCBCA} \]
## Reduced Conflict Instances

<table>
<thead>
<tr>
<th>Trees</th>
<th>Content</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Tree 1" /></td>
<td>213</td>
<td>n1l1</td>
</tr>
<tr>
<td><img src="image2" alt="Tree 2" /></td>
<td>21435</td>
<td>n1l1</td>
</tr>
<tr>
<td><img src="image3" alt="Tree 3" /></td>
<td>2143657</td>
<td>n1l1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trees</th>
<th>Content</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image4" alt="Tree 1" /></td>
<td>213</td>
<td>n1l1</td>
</tr>
<tr>
<td><img src="image5" alt="Tree 2" /></td>
<td>42513</td>
<td>n1l11</td>
</tr>
<tr>
<td><img src="image6" alt="Tree 3" /></td>
<td>6472513</td>
<td>n1l111</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Trees</th>
<th>Content</th>
<th>Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image7" alt="Tree 1" /></td>
<td>213</td>
<td>n1l1</td>
</tr>
<tr>
<td><img src="image8" alt="Tree 2" /></td>
<td>42513</td>
<td>n1l11</td>
</tr>
<tr>
<td><img src="image9" alt="Tree 3" /></td>
<td>4265713</td>
<td>n1l111</td>
</tr>
</tbody>
</table>

[Diagram of trees and content/structure pairs]
Parity Instances

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.

- $\Psi = \{d_1, \ldots, d_N, F\}$, where $d_1, \ldots, d_N$ 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 $(d_1, \ldots, d_N)$ are placed in positions representing leaf nodes in the tree shape, and all node positions with an $F$ label.
Motivations

• 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
Difficult 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

Graph showing the relationship between Tree Size and Tree Depth, with selected shapes highlighting the increase in size and depth.
Results - Content vs. Structure
Scaling Effects
Conflicts

Improving Content Objective Resulted in Better-or-Equal Structure Objective

Improving Structure Objective Resulted in Better-or-Equal Content Objective
Symbol Entropy

Average Symbol Entropy of a Solution in the Last Generation

2.408
1.431
0.602
Easy Instances

Improving Content Objective Resulted in Better-or-Equal Structure Objective

Improving Structure Objective Resulted in Better-or-Equal Content Objective
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