Representation and Heuristics in GP

Cezary Z. Janikow
UMSL
Roadmap

- GP Search Space
- Heuristics
  - Weak and Strong
  - Local and Global
  - First-order and Higher order
  - Examples
- Efficiency and Effectiveness
- Learning Heuristics
  - Learning in GP
  - Off-line and On-line
GP Search Space

- Representation vs. Solution Space
- Spanning the Space
- Constraining the Space
GP Search Space

• Best mappings
  • One-to-one, onto

• But – real life?
GP Search Space

• Real life
  • Sufficiency enlarges Function/Terminal sets, exponentially enlarging Representation Space
  • Closure further enlarges the space
• Result:
  • Redundancy with many-to-one mapping
GP Search Space

- Handling redundancy
- Over-represent better solutions
- Locality of genospace vs. phenospace
- Ok, but how?
  - Maybe it is better to search smaller space with more focused local search?
  - This in effect reduces the redundancy
  - Good focus “over-represents” better solutions
Spanning GP Search Space

- **2-D space**
  - **Tree structures** constrained by size limits and function arity
  - **Tree instances** of specific structures constrained by domain sizes

\[ y = \frac{x + \sin(a)}{2} \]
Constraining GP Search Space

- **Tree structures**
  - Hard to accomplish directly w/o instantiations
  - Indirect by adjusting possible instantiations

- **Tree instances**
  - **Strong** constraints
    - Prohibit some instantiations (labelings)
    - Structure-preserving cross, STGP, CGP, CFG-GP, etc.
  - **Weak** constraints
    - Favor some instantiations over others
    - CGP, Probabilistic Tree Grammars, EADs
Heuristics in GP

- Principles
- Strong vs. Weak
- Local vs. Global vs. Complete Models
- First-order vs. Higher-order
- Examples
Heuristics

• Rules of thumb that can improve outcome
  • Requirements per specs generally NOT considered heuristics
  • I want roots with 1 head vs. I prefer robots with 1 head
Strong vs. Weak

- Strong
- Weak

\[
\begin{align*}
\text{Strong} & \quad \text{Weak} \\
\frac{\sin x + y + 4}{?} &
\end{align*}
\]
Local vs. Global vs. Models

- Local
- Global
- Complete domain models (EDA)
- Local and global can be strong or weak

\[
\frac{x + \sin y}{a + 4}
\]
First-Order vs. Higher-Order

- **First-order**
  - Can be efficiently processed in CGP

- **Higher-order**
  - Can be efficiently processed in CGP with ADFs (but easy only for strong constraints)

- Both local and global
Heuristics and Learning in GP

- GP learns about the search space using selection
  - Discovers more promising regions
  - But does not learn how to search the regions more effectively

- Heuristics can be used to search local subspaces more effectively w/o sacrifice in efficiency
- Heuristics can be learned, allowing GP to learn how to perform local searches more efficiently/effectively
GP converges to regions but does not learn to improve local search (except in crossover by limiting the material in the population)
GP with Heuristics: Strong

GP converges to regions and performs pruned uniform local search
GP with Heuristics: Weak and Strong

GP converges to regions and performs more guided local search

Learning of the heuristics will improve further improve the search

\[ P^i \rightarrow \text{Pruned non-uniform distribution} \rightarrow P^{i+1} \]

- Mutation/Crossover
- Reproduction
Examples

- First-order local/global strong heuristics in SantaFe
  - Compare vs. weak heuristics
- Higher-order strong with first-order weak heuristics in Mult-11
- Global higher-order weak heuristics in Mult-11
First-order local/global strong heuristics in SantaFe

**Terminals**
- turn *left, right, move* action

**Functions**
- *if-food-ahead*
  - test the position directly ahead for food, and if true perform the first action, otherwise perform the second action
- *progn2, progn3*
  - take two and three arguments, respectively, and execute them sequentially.
Reducing Function Set: Basics, Quality
Reducing Function Set: Basics, Efficiency

Figure 2. Efficiency for reduced sets
Reducing Function Set: Combined, Quality

Figure 3. Learning for reduced sets
Reducing Function Set: Combined, Efficiency

Figure 4. Efficiency for reduced sets
Constraining Root and Local Structure: Basics, Quality

Figure 5. Learning for basic structural
Constraining Root and Local Structure: Basics, Efficiency

Figure 6 Efficiency for basic structural
Constraining Root and Local Structure: Combined, Quality

Figure 7. Learning for combined
Constraining Root and Local Structure: Combined, Efficiency

Figure 8. Efficiency for combined
Combined Function Set and Structural Heuristics: Quality

Figure 9. Learning for combinations of
Combined Function Set and Structural Heuristics: Efficiency

Figure 10. Efficiency for combinations of
More Combined Heuristics: Quality

Figure 11. Learning for more combined
More Combined Heuristics: Quality

Figure 12. Efficiency for more combined
Best Heuristics by Inspection: Quality (vs. components)

Figure 13. Learning with CJM heuristic.
Figure 14. Efficiency with CJM
Best Heuristics Summary: Quality

Figure 15. Learning summaries.
Figure 16. Efficiency summaries.
Testing Slightly Different Trails: Same Basic Primitives

Figure 17. Learning for slightly different
Testing Different Trails: Similar Basic Primitives

Figure 18. Learning on substantially
Learning Weak Heuristics with ACGP

Figure 19. Learning curve in ACGP (off-
Comparing Learned Weak and User-Discovered Strong Heuristics

![Graph comparing average fitness over generations for different heuristics]

Figure 20. Comparing our heuristics
First-Order Weak Heuristics in Mult-11

- **Terminals**
  - 3 addresses, 8 data

- **Functions**
  - if(_,_,_)
  - and(_,_), or(_,_), not(_)

First-Order Strong Heuristics in Mult-11

- Strong heuristics: medium and heavy

![Graph showing quality over generations for different heuristics.](image)
First-Order Weak Heuristics in Mult-11

Weak heuristics

![Graph showing the evolution of average fitness over generations for different iterations. The x-axis represents generation, and the y-axis represents average fitness. The graph includes lines for iteration 1, iteration 2, iteration 3, iteration 5, and iteration 10.]
Higher-Order Strong with First-Order Weak Heuristics in Mult-11

- Using Higher-order ADFs to learn strong local heuristics
- Complete ADFs used in First-order weak heuristics
Higher-Order Strong with First-Order Weak Heuristics in Mult-11
Global Higher-Order Weak Heuristics in Multi-11

- Global higher-order weak heuristics
  - 2-level only
  - Left-to-right probabilistic network
  - Discovered if(address,if,if)

- First-order local weak heuristics
  - Discovered if(address,_,_) if(_,if/data,_) if(_,_,if/data)
Global Higher-Order Weak Heuristics in Mult-11

![Graph showing average fitness over generations for different iterations.](image)
Efficiency of Processing

- Processing efficiency
- Combined effectiveness
- Measured only for first-order strong local and global heuristics
  - Processed by CGP
  - Other processing mostly hard-coded code at the moment
Processing in GP

GP Processing and Heuristics
- Setup (negligible)
- Mutation (negligible)
- Crossover (negligible)
- Reproduction (not affected)
- Evaluation (not affected)
Mutation

Heuristics efficiency tree-size independent
Depends on proper label set
Mutation

Measured in mutation-only runs (setup+evaluation subtracted)
Mutation

Per-node time virtually identical on every generation regardless of heuristics and regardless of GP of CGP
- Most work on memory not mutation itself
- Thus, negligible impact on efficiency
Crossover

Heuristics efficiency tree-size dependent (one more traversal)

Also depends on proper label set
Crossover

- Measured in crossover-only runs (setup+evaluation subtracted)
Crossover

Per-node time virtually identical on every generation regardless of heuristics and regardless of GP of CGP

- Despite extra traversal
- Most work on memory not crossover itself
- Thus, negligible impact on efficiency
Efficiency and Effectiveness

- Processing first-order local/global weak and strong constraints has negligible impact on efficiency.
- How about effectiveness of the search?
  - GP can be more effective but only given good constraints.
(Good)Heuristics

Effectiveness

- Faster processing per generation
  - Smaller trees thus faster evaluation and faster memory operations

- Fewer generation
  - Smaller effective search space
  - More informed local search
Faster Generations

Santa Fee trail – uncompleted runs

![Graph showing average tree size over generations for different conditions](image)
Faster Generations

Santa Fee trail – all runs

![Graph showing average tree size over generations for different algorithms: GP, CGP no heuristics, CGP medium heuristics, CGP heavy constraints.](image)
Fewer Generations

![Bar chart showing generations to solve for different conditions: GP, CGP no heuristics, CGP medium heuristics, and CGP heavy constraints. The chart indicates that CGP heavy constraints require the fewest generations to solve.]
Fewer Generations

(Uncompleted assumed 100)

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Generations to solve (avg of 10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GP</td>
<td>80</td>
</tr>
<tr>
<td>CGP no heuristics</td>
<td>70</td>
</tr>
<tr>
<td>CGP medium heuristics</td>
<td>10</td>
</tr>
<tr>
<td>CGP heavy constraints</td>
<td>10</td>
</tr>
</tbody>
</table>
Off-Line (Sporadic) vs. On-Line
Learning of Heuristics

Heuristics can be learned by observing statistics in better trees.

Some generations are needed before statistics are reliable.

Thus, system must semi-converge before heuristics can be extracted, then can be fed back.

- Work started by looking at finals trees
- Off-line (sporadic) approach
Off-Line Learning

- Heuristics can still improve overall performance
- Heuristics can be used next time or on “similar” problem
- Heuristics can provide comprehensible knowledge about the domain
- Feed-back loop is followed by re-initializing of the population
Off-Line Learning – Feedback with Regrowing (SantaFe)

Learning weak heuristics

Figure 19. Learning curve in ACGP (off-
Off-Line Learning – Regrowing and Iteration Length (SantaFe)

Learning weak heuristics in different settings
Off-Line Learning – Regrowing and Iteration Length (Mult-11)

what=2 is regrowth
Off-Line Learning – Regrowing and Iteration Length (Mult-11)

what=2 is regrowth
Off-Line Learning – Regrowing and Iteration Length (Mult-11)

Learning weak heuristics in different settings
Off-Line Learning – Mult-11

Example heuristics if(_,*,*) combined
On-Line Learning

Clearly short iterations reduce or prevent learning

- Regrowing is generally helpful yet it destroys history in short iterations and works only in long iterations
- On-line approach is to use tournament selection between newly reproduced and the regrown populations
On-Line Learning
On-Line Learning (Mult-11)
On-Line Learning (SantaFe)
Summary

Heuristics can be useful
- Reduce the search space and make local search more focused
- Can be processed efficiently
- Can improve effectiveness
- Even simple local/global first-order strong/weak heuristics are useful
- More complex heuristics can further improve effectiveness but efficiency is uncertain
Summary

Heuristics can be learned in GP

- Off-line learning offers better heuristics but may not improve much the current run
- On-line learning is more challenging and needs to be further improved