

Analysis of Ancestry in Genetic Programming with a Graph Database

Nicholas Freitag McPhee
with help from
David Donatucci and M. Kirbie Dramdahl

Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA

1 Oct 2014
Hampshire College CI-Lab

The Big Picture

- Genetic programming (GP) demonstrably works.
- Difficult to determine how this process works.
- Databases allow examination of the internal interactions of a run.
- Graph databases more efficient for this task than relational DBs.
- Analysis may allow us to improve GP tools.

Outline

- 1 Genetic Programming
- 2 Graph Database
- 3 Results
- 4 Conclusions

Outline

- 1 Genetic Programming
 - Transformations and genealogical history
- 2 Graph Database
- 3 Results
- 4 Conclusions

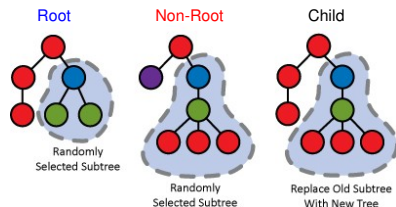
Transformations and genealogical history

Crossover sexual reproduction
(**root** and **non-root**)

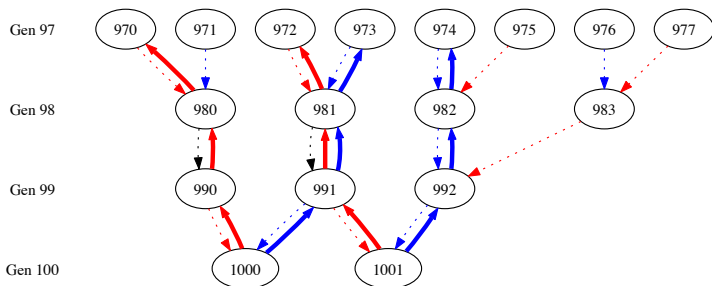
Mutation subtrees altered

Reproduction asexual reproduction

Elitism reproduction based on fitness



geneticprogramming.us



Outline

1 Genetic Programming

2 Graph Database

- Neo4j
- Cypher
- DB setup
- Cypher examples

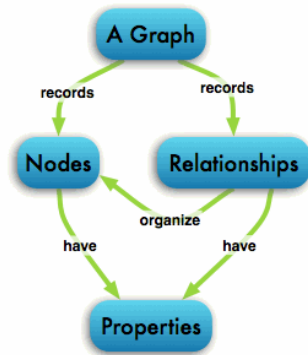
3 Results

4 Conclusions

Neo4j

Neo4j is a graph database.

- Relatively new tool
 - Initial release 2007
 - V1.0 in 2010, V2.0 last December
- Information is stored using a graph consisting of nodes and relationships
- Efficient recursive queries compared with traditional databases



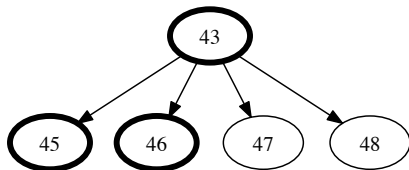
Neo4j <http://goo.gl/nzRWSV>

Cypher

Neo4j's query language is Cypher.

Fundamental elements of
Cypher queries:

- START
- MATCH
- WHERE
- RETURN



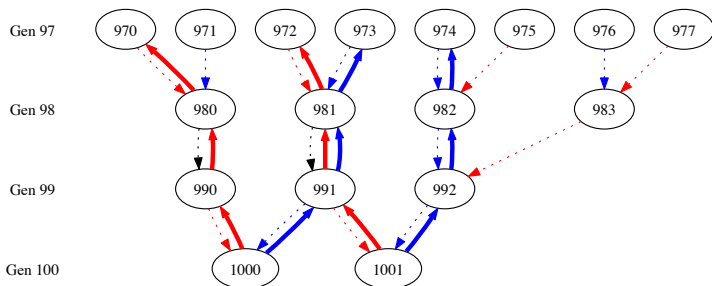
```
START parent=node(43)  
MATCH (parent)-[:PARENTOF]->(child)  
WHERE id(child) < 47  
RETURN parent, child;
```


DB setup

We store:

- Individuals as nodes + data, e.g., program, fitness, generation
- XO, mutation, replication events as edges between nodes

Different edge types indicate relationship types, e.g., *ROOT_PARENT* edge from 991 to 1000 says 991 is the root-parent of 1000.



Finding the “winning” fitness and individual

Finding the minimal fitness across all individuals (nodes) in the DB:

```
start n=node(*)
return min(n.penalizedFitness);
```

Finding one (arbitrary) “winning” individual:

```
start n=node(*)
return n
order by n.penalizedFitness
limit 1;
```

These are not unlike queries in relational databases.

Ancestry of an individual (hard for relational DBs)

Find all the root ancestors of individual 99,000:

```
start n=node(99000)
match ps =
  (n) <- [r:ELITISM|PARENTOF|ROOT_XOOF|MUTANTOF*] - (p)
return distinct ps;
```

Find the fitness of all the root ancestors of individual 99,000:

```
start n=node(99000)
match ps =
  (n) <- [:ELITISM|PARENTOF|ROOT_XOOF|MUTANTOF*] - (p)
where p.generation=1
return extract(x in nodes(ps) | x.fitness);
```

Outline

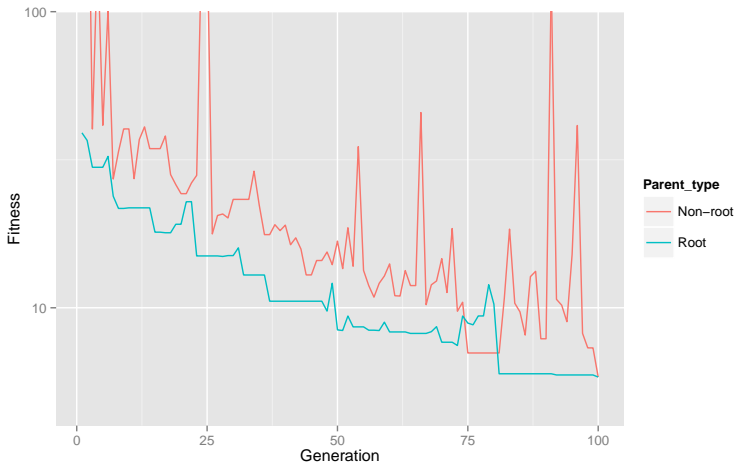
- 1 Genetic Programming
- 2 Graph Database
- 3 Results**
 - A few questions
 - Fitness Over Time
 - Improved Transformations
 - Common Ancestor(s)
- 4 Conclusions

A few questions

- 1 What does the fitness of the “winning” individual’s ancestry line look like over time?
- 2 How often does mutation improve fitness? Also, how often does crossover improve fitness, where the root parent is more fit than the non-root parent, and vice versa?
- 3 Does a group of individuals have a common root parent ancestor and what is the latest generation where such an ancestor occurs?
- 4 How many individuals in the initial generation have any root parent descendants in the final generation?

Fitness Over Time

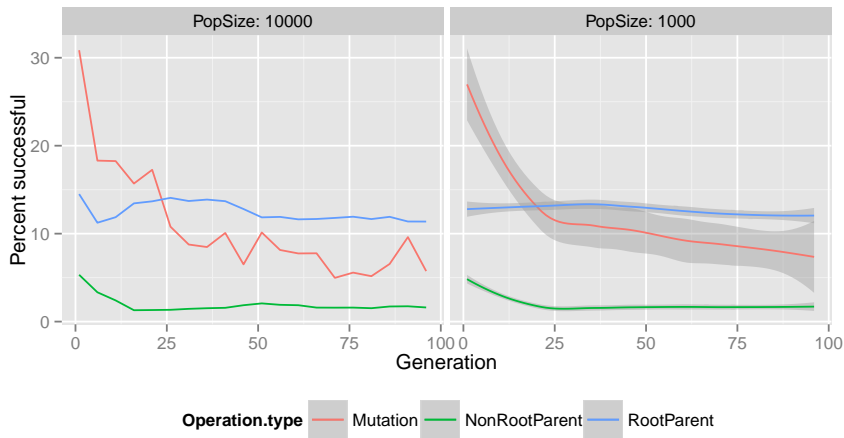
What does the fitness of the “winning” individual’s ancestry line look like over time?



Percentage of Improved Transformations

How often do mutation and crossover improve fitness?

Results for one 10,000 individual run and three 1,000 individual runs

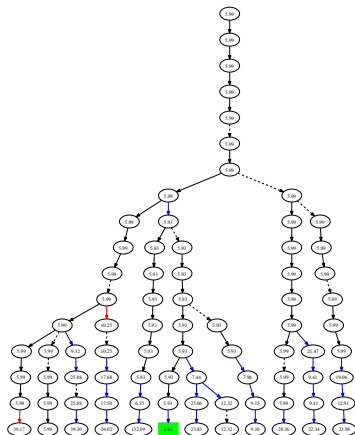


Common Ancestor

How far back to a common root parent ancestor? How many initial generation individuals have descendants in the final generation?

In run w/ 10K individuals for 100 gens:

- Final pop consists of 2 clades
- One clade *strongly* dominates (only 24 left in small clade)
- Each clade root-descends from single individual in initial population



Outline

- 1 Genetic Programming
- 2 Graph Database
- 3 Results
- 4 Conclusions**

Conclusions

Conclusions

- We can collect & extract useful data about evolutionary dynamics
- More detailed analysis that moves beyond statistical summaries
- Suggests interesting avenues to explore
 - Importance of having root parent be more fit
 - Changing role of mutation over the life of a run
 - Potential for clades/quasi-species and their impact

Future Work

- Enforcing the root parent to have better fitness in XO
- Exploring equivalents in other EC systems (e.g., Push)

Thanks!

Thank you for your time and attention!

Contact:

- `mcphee@morris.umn.edu`
- Twitter **@NicMcPhee**
- Paper: `https://github.com/NicMcPhee/MICS-2014-GraphDB-EC`

Big thanks to David Donatucci and Kirbie Dramdahl for their many contributions to this work.

Questions?

References

Run Configurations

Target Function $\sin(x)$

Variables x (range 0.0 to 6.2, incremented by steps of 0.1)

Constants range between -5.0 and 5.0

Operations addition (+), subtraction (-), multiplication (*),
protected division (/)

Generation Number 100

Population Size Per Gen 1,000 (3 runs) and 10,000 (1 run)

Transform Percentages crossover (90%), mutation (1%), reproduction (9%)

Elitism best 1%

Fitness absolute error between target function and
individual function