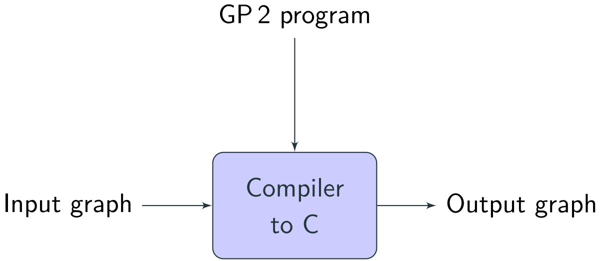
Probabilistic Graph Programming for Randomized and Evolutionary Algorithms

Timothy Atkinson Detlef Plump Susan Stepney

University of York

Introduction to GP 2

Graph Programming Language GP 2



- Experimental language for graphs.
- Rule-based visual manipulation of graphs.
- Computationally complete.
- Non-deterministic.

An Example: Transitive Closure

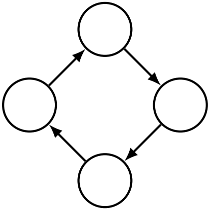
A graph is *transitive* if for every directed path $v_1 \rightsquigarrow v_2$ where $v_1 \neq v_2$ there is an edge $v_1 \to v_2$.

Main $:=$ link!

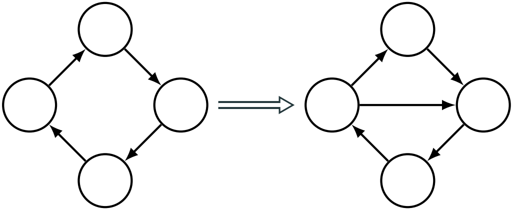
```
link(a,b,c,d,e:list)
```


```
where not edge(1, 3)
```

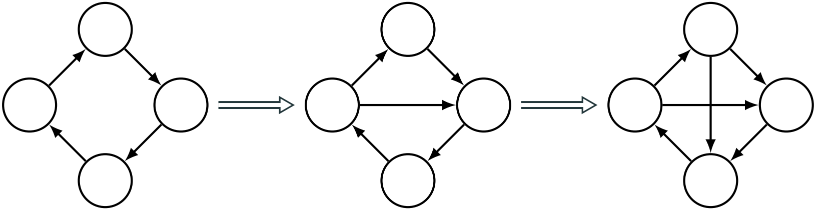
An Example: Transitive Closure



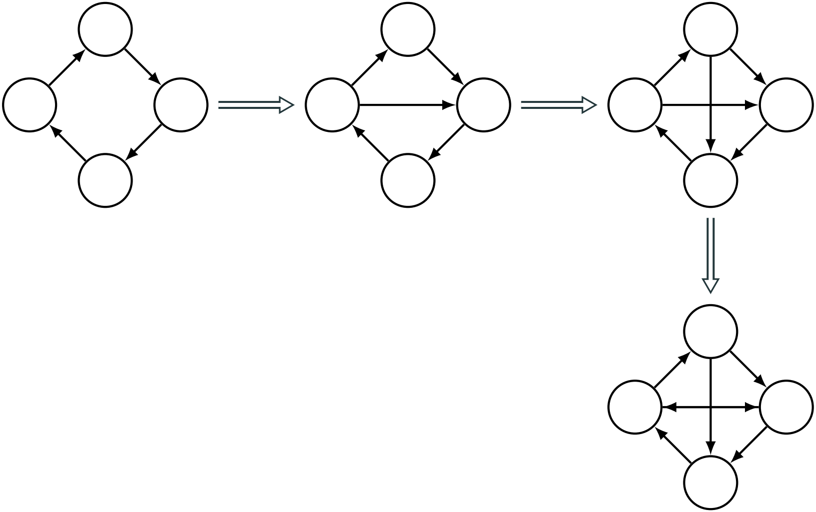
An Example: Transitive Closure



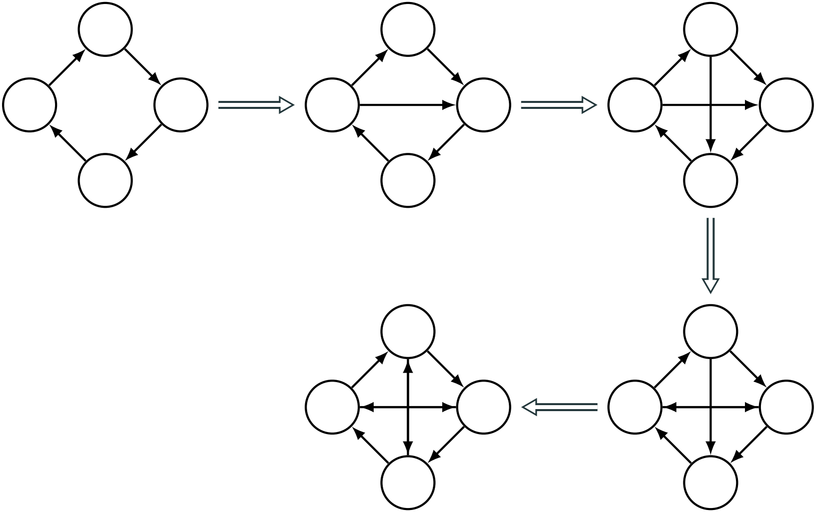
An Example: Transitive Closure



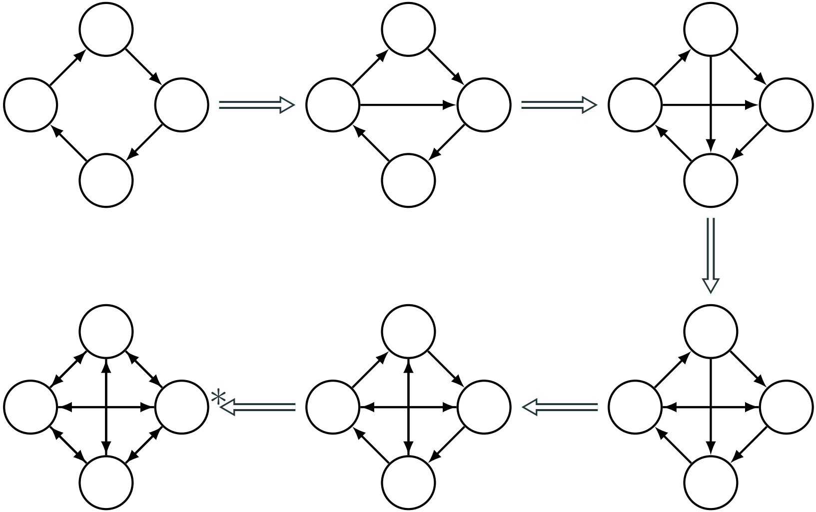
An Example: Transitive Closure



An Example: Transitive Closure



An Example: Transitive Closure



P-GP 2:
A Probabilistic Extension of GP 2

Non-determinism in GP 2

- Calling rule-set $\mathcal{R}$ on graph G is non-deterministic in GP 2:

$$G \Rightarrow_{\mathcal{R}} H_i \in \{H \mid G \Rightarrow_{r_i,g} H \text{ and } r_i \in \mathcal{R}\}$$

- We extend GP 2's syntax to allow a programmer to specify a probability distribution over these outcomes.
- This allows a programmer to specify randomized algorithms, a powerful concept used in broader computer science (see [1]).

Syntax for Rule Calls

A rule-set is executed probabilistically by calling it within square brackets:

$$[r_1, r_2 \ldots r_n]$$

A rule-set called using conventional curly brackets is executed 'non-deterministically' as in GP 2:

$$\{r_1, r_2 \ldots r_n\}$$

Our extension P-GP 2 is conservative; all existing GP 2 programs are valid and will be executed as before.

Specifying Probabilities I

A rule-set $\mathcal{R}$ called on host graph G is executed by:

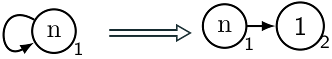
1. Probabilistically pick a rule $r_i \in \mathcal{R}$ according to a weighted distribution.
2. Probabilistically pick a match g for rule r_i according to a uniform distribution.
3. Execute (r_i, g) on G:

$$G \Rightarrow_{r_i, g} H$$

Specifying Probabilities II

We choose a rule first, using a weighted distribution, from the set of rules with valid matches; $\mathcal{R}^G$.Each rule r_i has an associated real-valued positive weight given by $w(r_i)$ - specified in square brackets after the rule declaration:

```
grow_loop(n:int) [3.0]
```



Then the probability of choosing r_i from $\mathcal{R}^G$ is

$$\frac{w(r_i)}{\sum_{r_x \in \mathcal{R}^G} w(r_x)}$$

Specifying Probabilities III

Once rule r_i has been chosen, we choose a match for r_i with uniform probability from the set of valid matches G^{r_i}. Some match g is chosen with probability

$$\frac{1}{|G^{r_i}|}$$

Yielding an overall definition of the probability distribution $P_{G^{\mathcal{R}}}$ over the set of all possible rule-match pairs $G^{\mathcal{R}}$:

$$P_{G^{\mathcal{R}}}(r_i, g) = \frac{w(r_i)}{\sum_{r_x \in \mathcal{R}^G} w(r_x)} \times \frac{1}{|G^{r_i}|}$$

Related Work

Other approaches look at *modeling* e.g. Probabilistic GTS (discrete) [2] and Stochastic GTS (continuous) [3]. These are single graph transformation systems with probability distributions over outcomes.

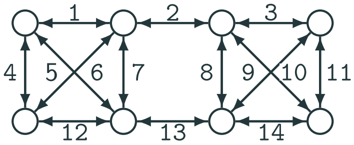
We look at *programming*; sequential graph transformation systems that *algorithmically* transform a graph.

Applications I:
Karger's Algorithm

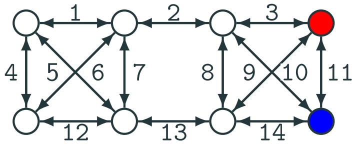
Karger's Algorithm & Minimum Cuts

- Karger's algorithm [4] is a probabilistic algorithm for computing the **minimum cut** of a graph with a known lower bound probability of success.
- The minimum cut of a graph is a minimal set of edges to remove from a graph to produce two disconnected sub-graphs.
- General idea is to repeatedly contract (merge) adjacent nodes until only 2 nodes remain.

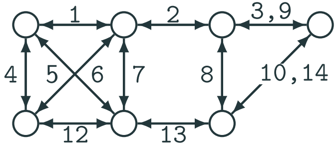
Karger's Algorithm



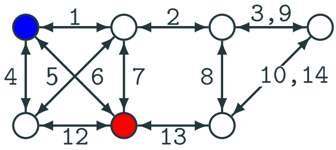
Karger's Algorithm



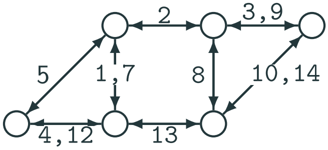
Karger's Algorithm



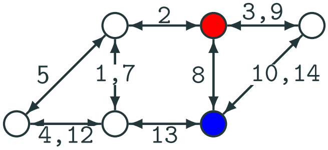
Karger's Algorithm



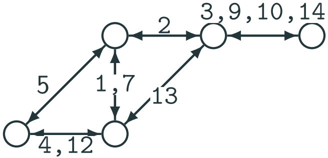
Karger's Algorithm



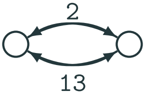
Karger's Algorithm



Karger's Algorithm



Karger's Algorithm



Karger's Guarantee

Consider a global minimum cut of c edges of graph G with n nodes and e edges:

- The minimum degree of G must be at least c, and therefore $e \geq \frac{n.c}{2}$.
- The probability of contracting some edge in the minimum cut is therefore
$$\frac{c}{e} \leq \frac{c}{\frac{n.c}{2}} = \frac{2}{n}$$
- The probability of producing the minimum cut (by *never* contracting some edge in the minimum cut) is bounded by:

$$p_n \geq \prod_{i=3}^{n} 1 - \frac{2}{i} = \frac{2}{n(n-1)}$$

Karger's Algorithm in P-GP 2

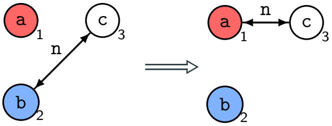
Main := (three_node; [pick_pair]; delete_edge!; redirect!; cleanup)!

three_node(a,b,c:list)

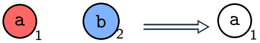
pick_pair(a,b:list; n:int)

delete_edge(a,b:list; n:int)

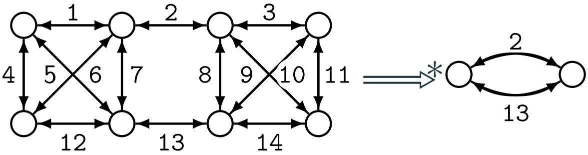
redirect(a,b,c:list; n:int)



cleanup(a,b:list)



Karger: An Example



$$p_n \geq \frac{2}{8.(8-1)} = \frac{1}{28}$$

Applications II:
$G(n,p)$ Model for Random Graphs

The $G(n, p)$ Model

The $G(n, p)$ [5] model randomly generates graphs (V, E, s, t) such that:

- $|V| = n$.
- For each pair in $V \times V$, an edge exists with probability p.

Sampling the $G(n, p)$ model in P-GP 2

`Main := (pick_edge; [keep_edge, delete_edge])!; unmark_edge!`

`pick_edge(a,b,c:list)`

`unmark_edge(a,b,c:list)`

`keep_edge(a,b,c:list) [p]`

`delete_edge(a,b,c:list) [1.0 - p]`

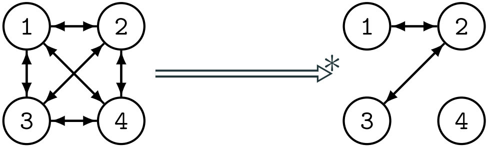
Expects, as input, a fully connected graph with n nodes.

$G(n,p)$: an Example

- A graph with M edges occurs with probability

$$p^M(1-p)^{\binom{n}{2}-M}$$

- $G(n, 0.4)$, with probability 0.0207:

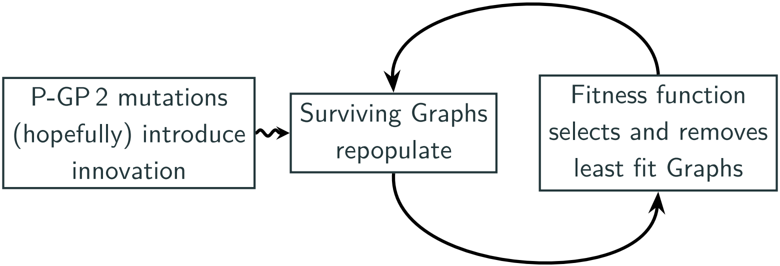


Applications III: Evolving Graphs by Graph Programming

Why Evolve Graphs?

Graphs are ubiquitous:
(Neural/Bayesian) Networks, (Quantum) Circuits, Syntax Trees etc.

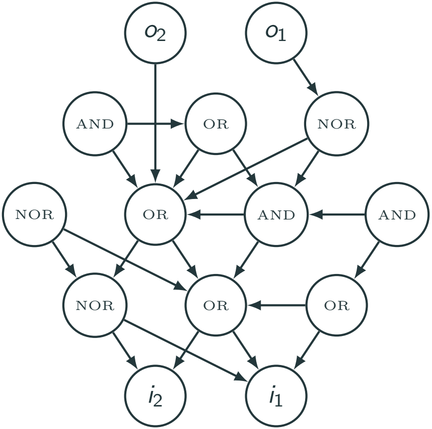
Evolutionary Algorithms iteratively explore poorly understood domains.



In this work we focus on digital circuit benchmarks.

EGGP

An Evolutionary Algorithm for learning graphs:



$$o_2 = (i_2 \downarrow i_1) \vee (i_2 \vee i_1)$$

Edge Mutations in P-GP 2

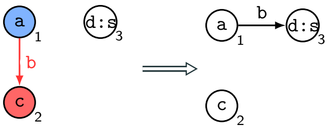
Main := try ([pick_edge]; mark_output!; [mutate_edge]; unmark!)

pick_edge(a,b,c:list)

mark_output(a,b,c:list)

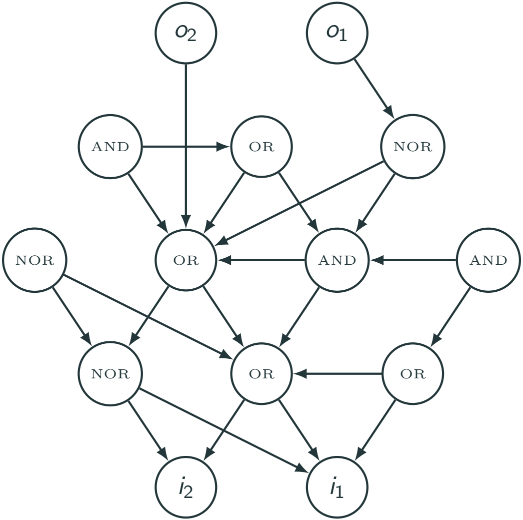
unmark(a:list)

mutate_edge(a,b,c,d:list; s:string)

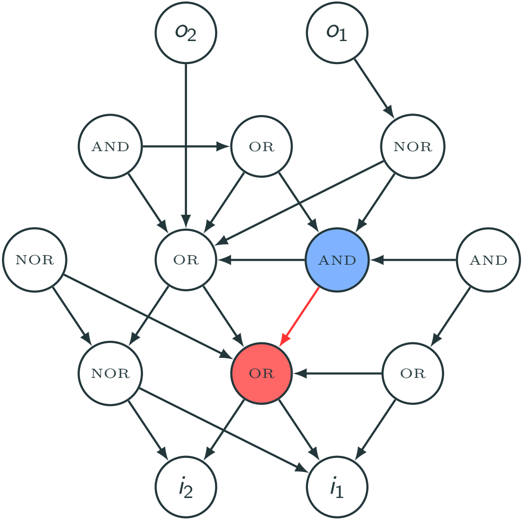


where s != "OUT"

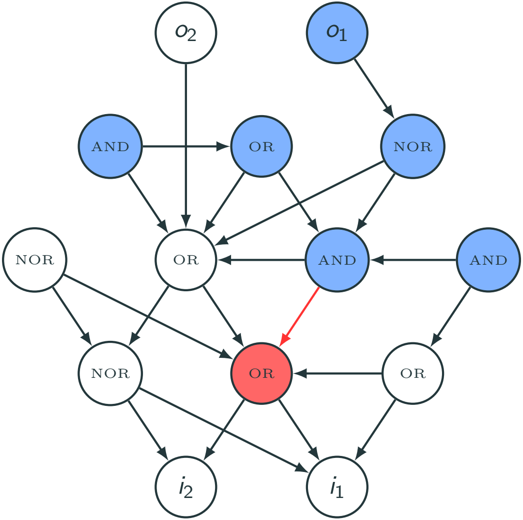
Edge Mutations



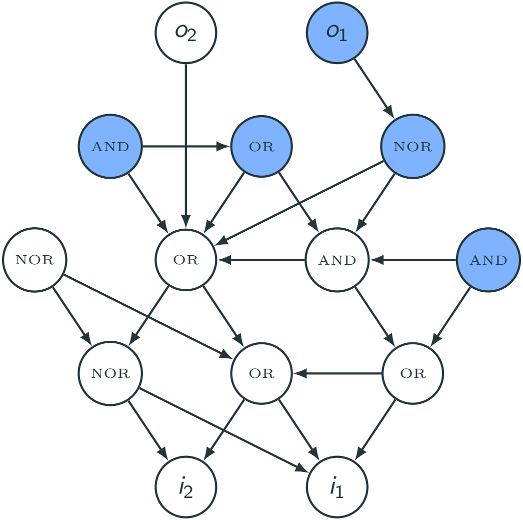
Edge Mutations: [pick_edge]



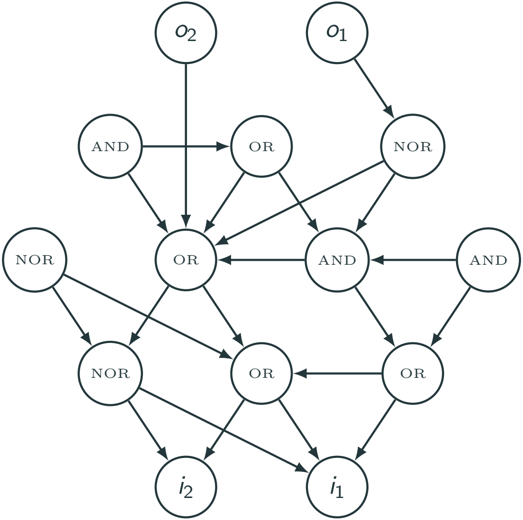
Edge Mutations: `mark_output!`



Edge Mutations: [mutate_edge]



Edge Mutations: `unmark!`



EGGP vs. CGP

CGP is a standard algorithm for evolving directed acyclic graphs, and was originally designed for evolving circuits [6].

	EGGP	**CGP**		
Problem	Median Evaluations	Median Evaluations	p	A
5-Bit Odd Parity	38,790	96,372	10^{-18}	**0.86**
6-Bit Odd Parity	68,032	502,335	10^{-31}	**0.97**
7-Bit Odd Parity	158,852	1,722,377	10^{-33}	**0.99**
8-Bit Odd Parity	315,810	7,617,310	10^{-34}	**0.99**

Results from Digital Circuit benchmarks for CGP and EGGP. The p value is from the two-tailed Mann-Whitney U test. Where $p < 0.05$, the effect size from the Vargha-Delaney A test is shown; large effect sizes $(A > 0.71)$ are shown in **bold**.

Conclusion

Conclusion

Contributions:

- Extended GP 2 to allow probabilistic rule-call execution.
- Implemented 3 distinct & previously impossible probabilistic graph programs using P-GP 2.

Future Work:

- What other randomized algorithms can we now implement?
- What other randomized algorithms *can't* we implement?
- Investigate efficiency of matching strategies e.g. incremental pattern matching.

Thank you!

P-GP 2:
https://github.com/UoYCS-plasma/P-GP2
EGGP:
https://github.com/UoYCS-plasma/EGGP

- Rajeev Motwani and Prabhakar Raghavan.
  ***Randomized Algorithms*.**
  Cambridge University Press, 1995.
- Christian Krause and Holger Giese.
  Probabilistic graph transformation systems.
  In *Proc. International Conference on Graph Transformation (ICGT 2012)*, volume 7562, pages 311–325. Springer, 2012.
- Reiko Heckel, Georgios Lajios, and Sebastian Menge.
  Stochastic graph transformation systems.
  Fundamenta Informaticae, 74(1):63–84, 2006.

David R. Karger.
Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm.
In *Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1993)*, pages 21–30. Society for Industrial and Applied Mathematics, 1993.

E. N. Gilbert.
Random graphs.
The Annals of Mathematical Statistics, 30(4):1141–1144, 1959.

Julian F. Miller, editor.
***Cartesian Genetic Programming*.**
Springer, 2011.