Towards Scalable Simulation Of Complex Network Routing Policies

Andrew Stone, Michelle Strout, Steven DiBenedetto, and Daniel Massey

Colorado State University

December 5, 2009

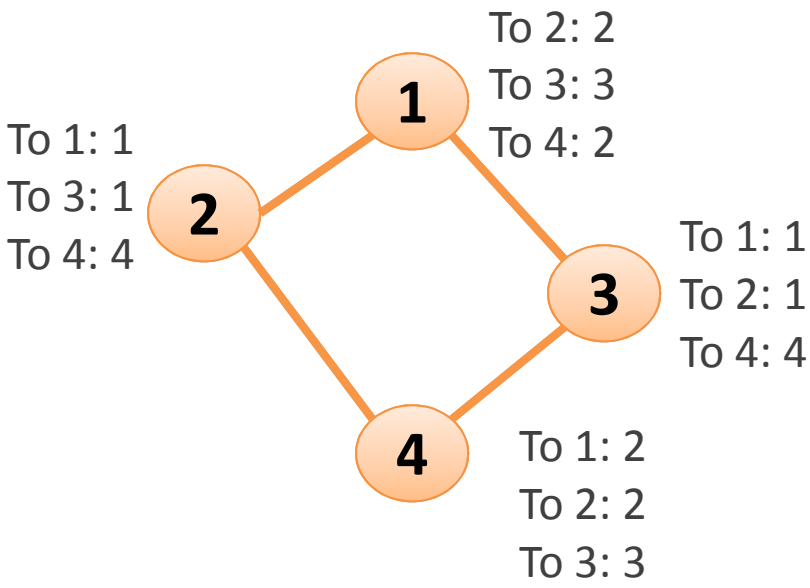
FRACTAL Workshop

The Problem

People who create/research networks are interested in the **routing problem**.

Particularly they want to know:

- how **policy** affects routing tables
- how **topology** affects routing tables



Especially interested on large scale networks (10k – 100k nodes)

Policies today are more complicated than they used to be.

No longer just shortest path. ISPs consider economic factors.

Solution: Create a routing simulator that can solve what routing tables converge to when there are different **policies** and **topologies**

We've created such a Simulator: Mr. Sim

About Metarouting

Metarouting is a technique to formalize routing policies

There is an analogy to be drawn between **data-flow analysis** and **metarouting** (MR has something similar to a meet and transfer function)

Data-flow analysis has lattice theoretic frameworks
Metarouting has routing algebras

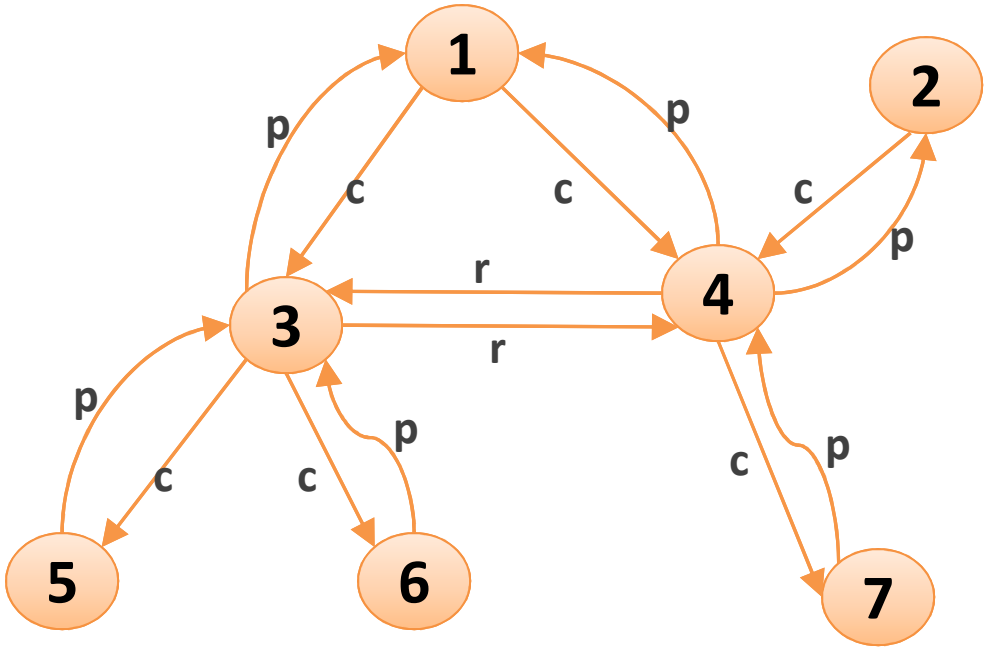
For data-flow analysis people have built implementation frameworks
In Mr. Sim we attempt to do the same for metarouting.

What I'll talk about:

what a routing simulation looks like
how **routing algebras** are formalized
how our **implementation framework** is used to specify policies

How a topology is modeled

Nodes = autonomous systems
Edges = connections
Edge labels describe the connection and influence policy



Examples of connections

c: customer connection
r: peer connection
p: provider connection

Propagating routes

A route is: (destination, next-hop, signature)

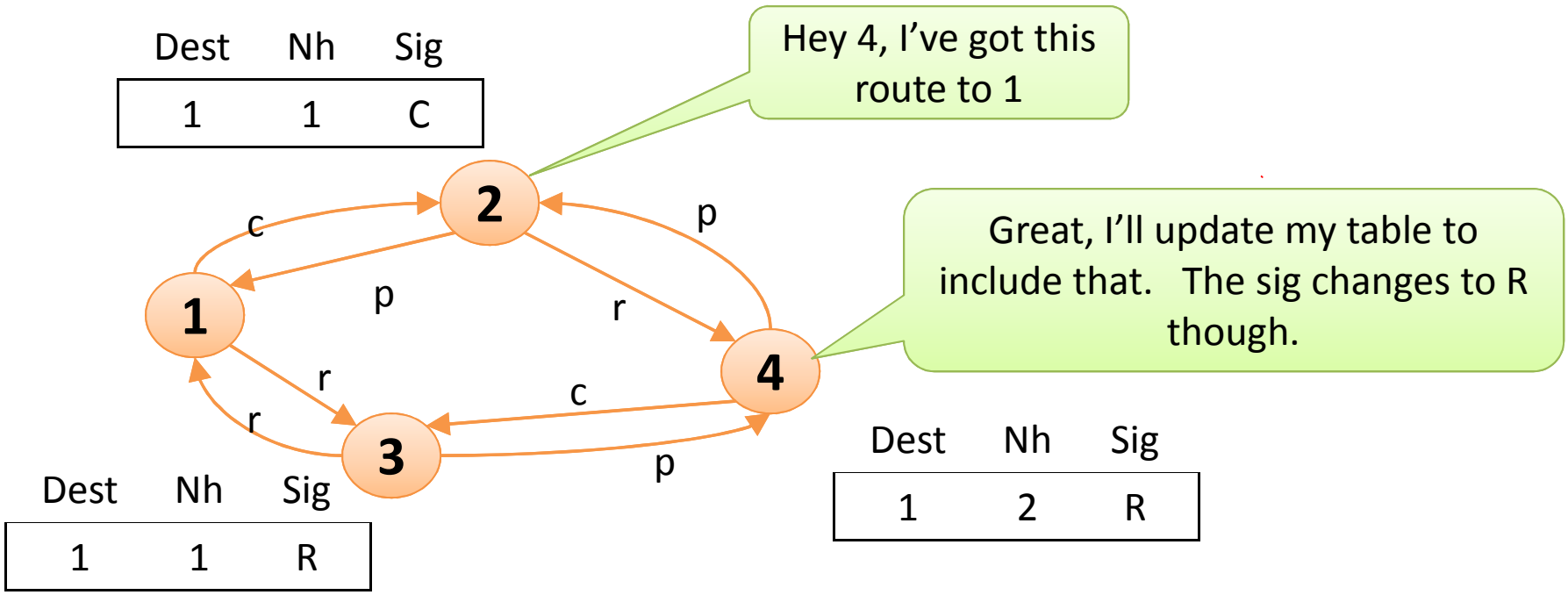
Σ = {C, R, P} L = {c, r, p}

Signature determines how preferred a route is

$\preceq$ C < R < P

smaller is more preferred.

Nodes propagate routes to neighbors. Neighbors update their routing tables based on the propagated information.



Propagating routes

A route is: (destination, next-hop, signature)

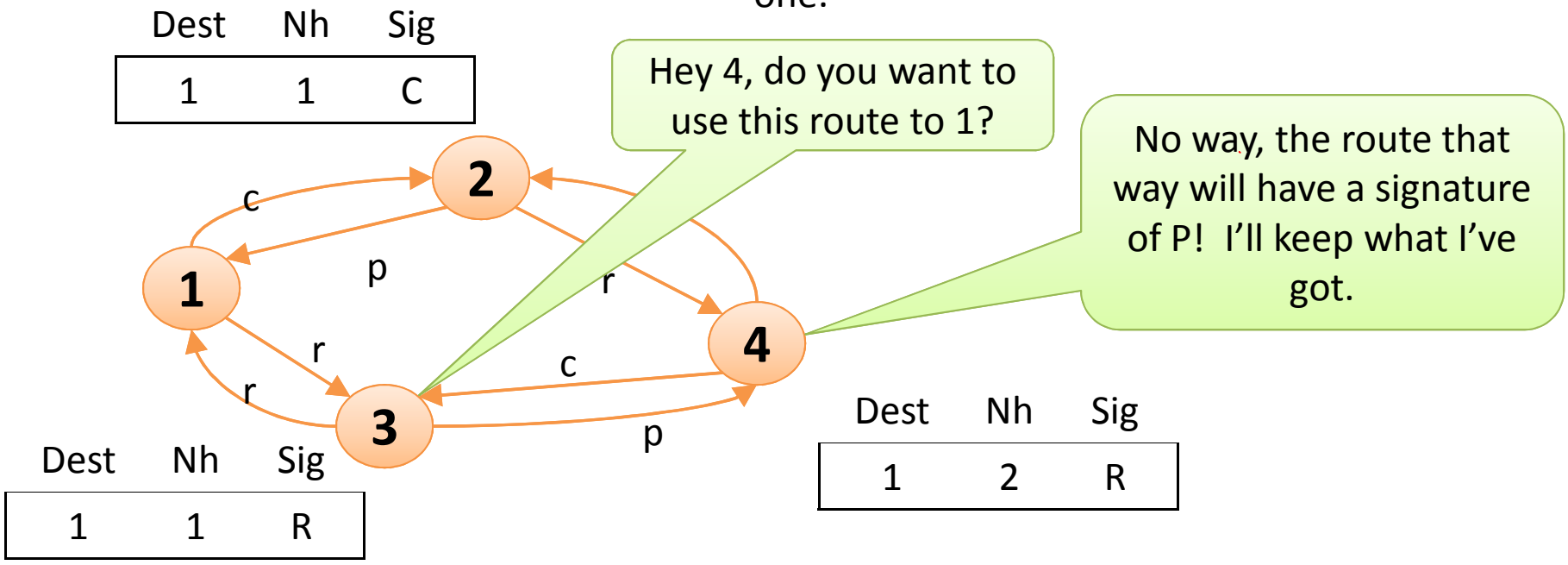
Σ = {C, R, P} L = {c, r, p}

Signature determines how preferred a route is

$\preceq$ C < R < P

smaller is more preferred.

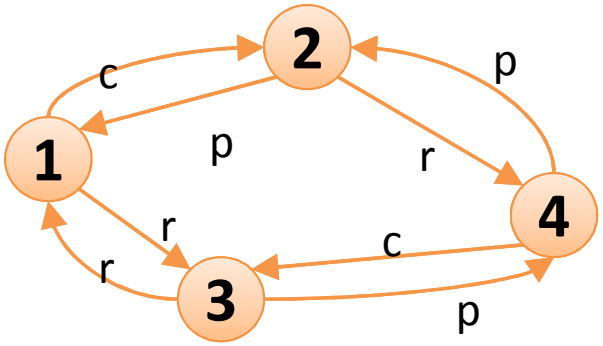
When a route is propagated it is compared against any existing routes to the same destination. The table is only updated if the new route is more preferred than the current one.



A simple policy

old signature

label	C	R	P	ϕ
c	C	ϕ	ϕ	ϕ
r	R	R	ϕ	ϕ
p	P	P	P	ϕ



In our example:
Routes are assigned some signature (C, R, P, or ϕ). Edges have some label (c, r, or p).

When propagating a new route, the signature is updated using the label operator.

The label operator is a function:
(signature) X (label) -> (signature)

Routing Algebras

$$(\Sigma, \preceq, L, \oplus, \mathcal{O})$$

Σ Set of possible signatures

$\oplus$ Label operator

$\preceq$ Preference relation (partial ordering of sigs)

$\mathcal{O}$ Originator set (possible signatures initial routes can have)

L Set of possible labels

Routing algebras can be composed. It's been shown that if certain properties of the algebra hold convergence is guaranteed. Specifically that the operator is strictly monotonic.

Mr. Sim Classes:

Mr. Sim model's routing algebras in a C++ framework. Users override these classes to define various properties of the algebra.

The framework includes the following abstract classes:

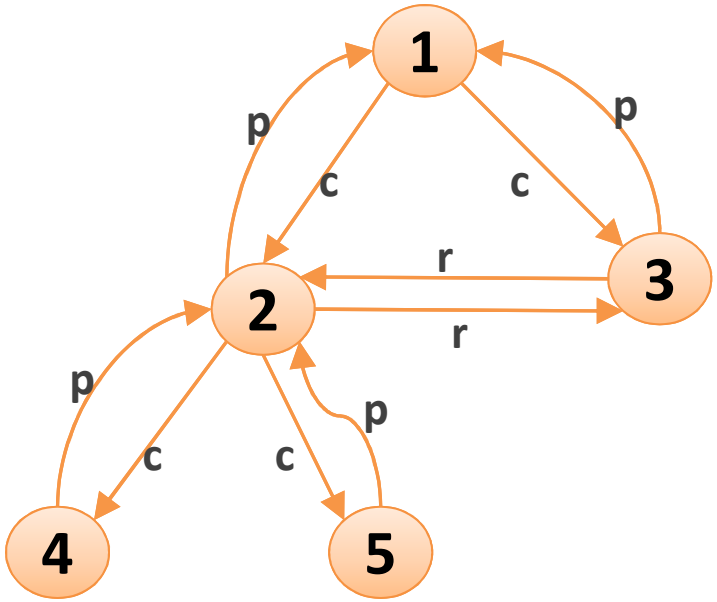
Signature
Label
PreferenceRelation
LabelOperator
NetworkReader

Algebra

Algebra is passed a PreferenceRelation, LabelOperator, and NetworkReader

How topologies are input to Mr. Sim

Map files list the edges in the network graph and the policies (labels) that are on those edges.



SimpleNet.map

```
1 2 c
2 1 p
1 3 c
3 1 p
2 3 r
3 2 r
2 4 c
4 2 p
2 5 c
5 2 p
```

Simulation Segmentation

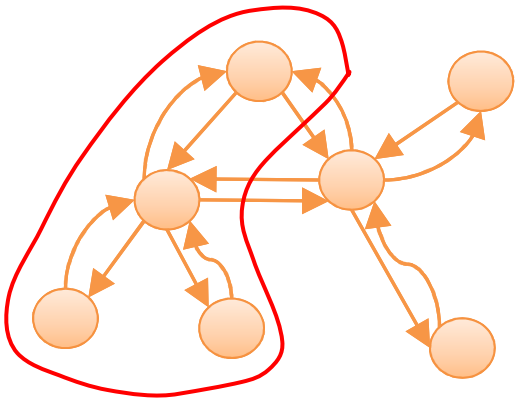
network	memory usage
1k nodes	.8 GB
10k nodes	6.88 GB
30k nodes	could not test
internet (33k nodes)	could not test

Doing all all-routes simulation on an internet-scale graph uses an enormous amount of memory. O(n^2) me requirement for tables.

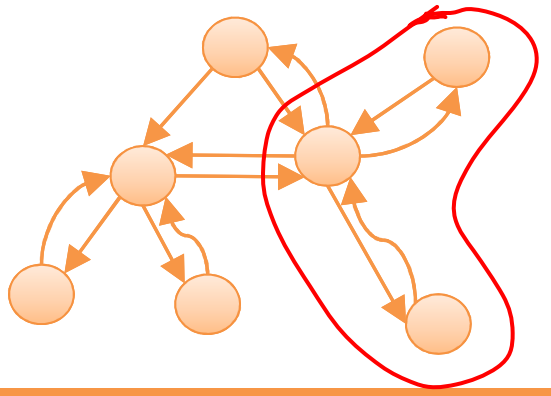
Swapping kills performance, so this is not a feasible way to solve the routing problem.

Due to how we model the routing problem it is possible to do a set of simulations that each only solve the problem for a set of routes.

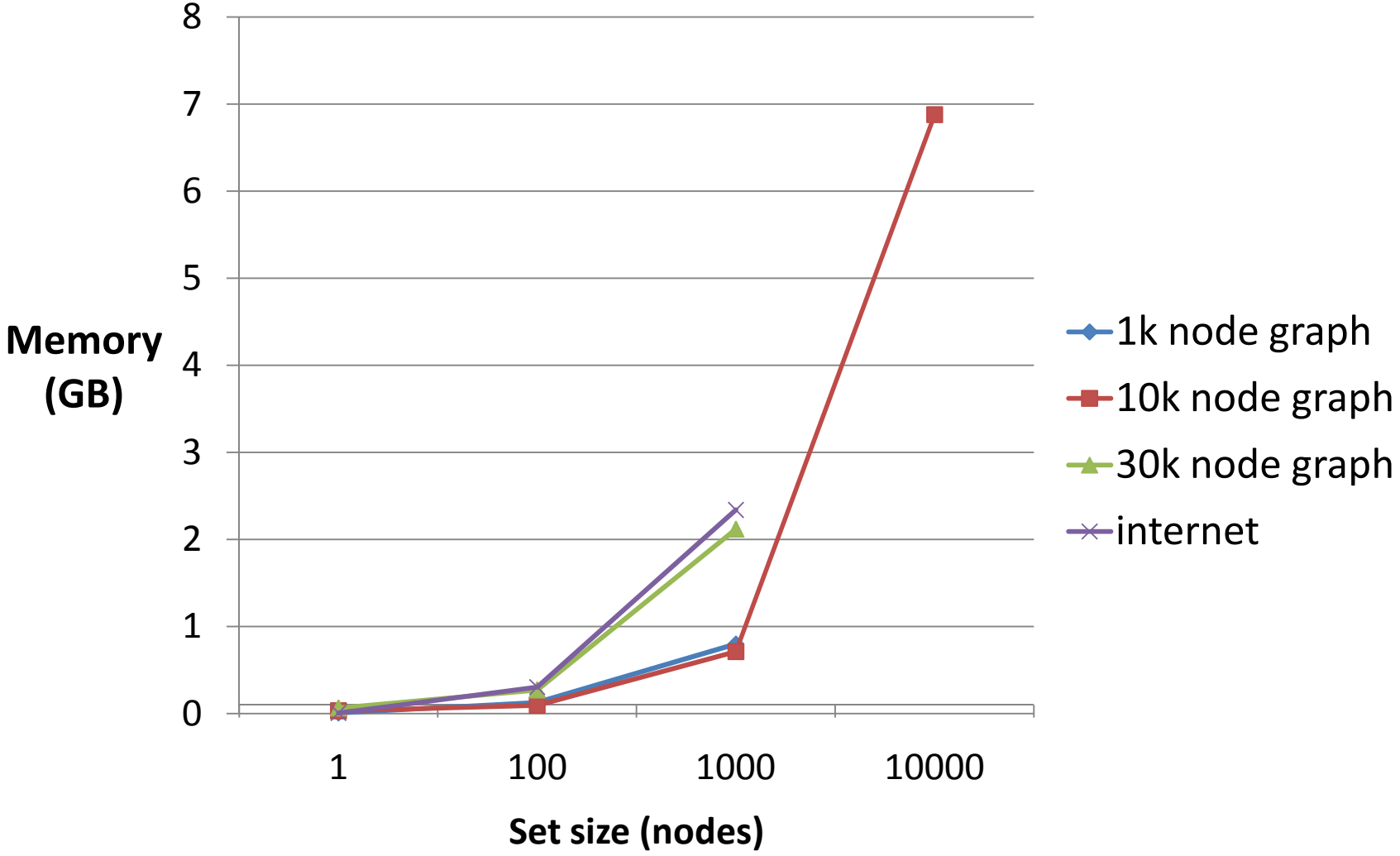
First just find routes to these nodes:



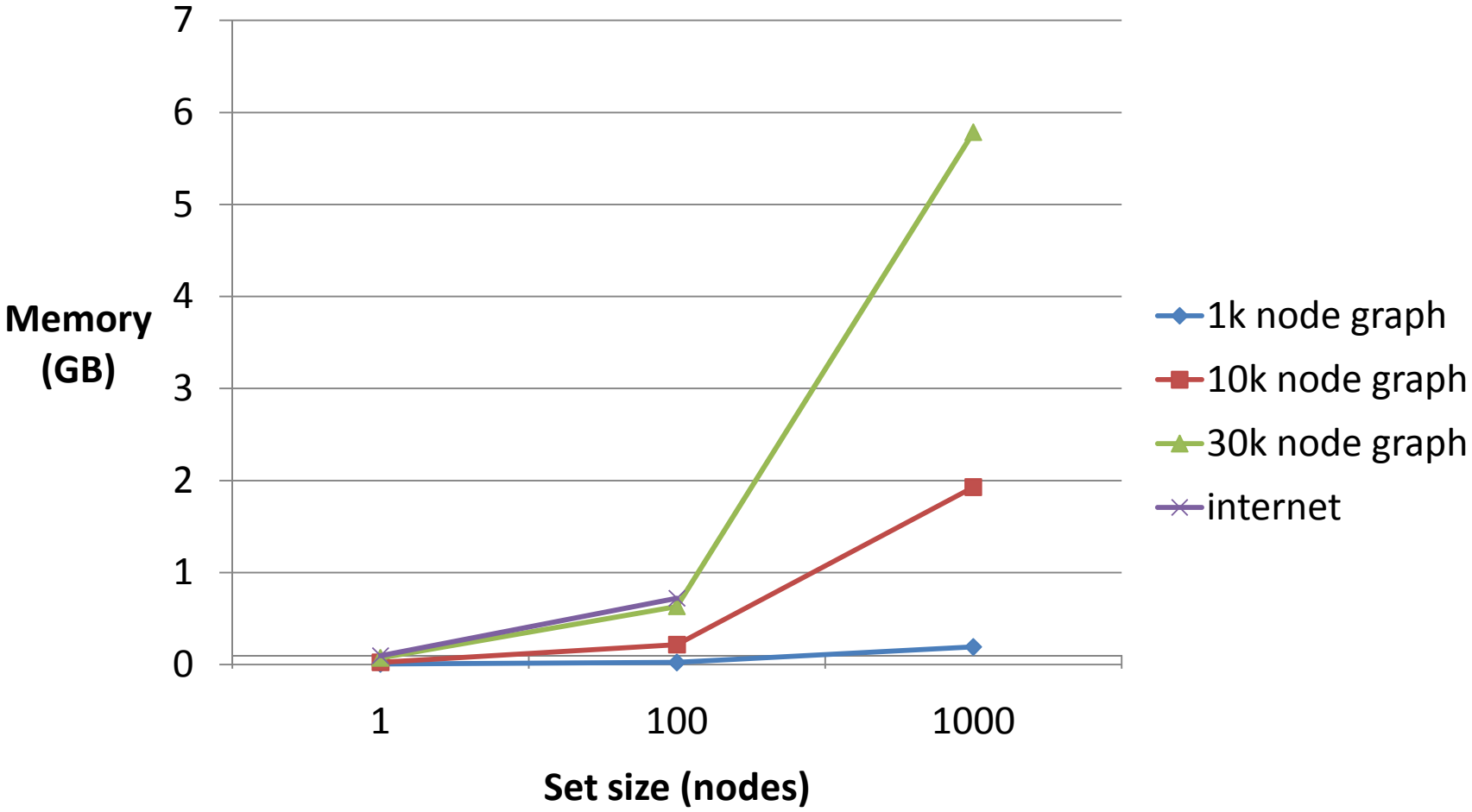
Then find routes to these nodes:



Segmented simulation: memory usage, simple policy

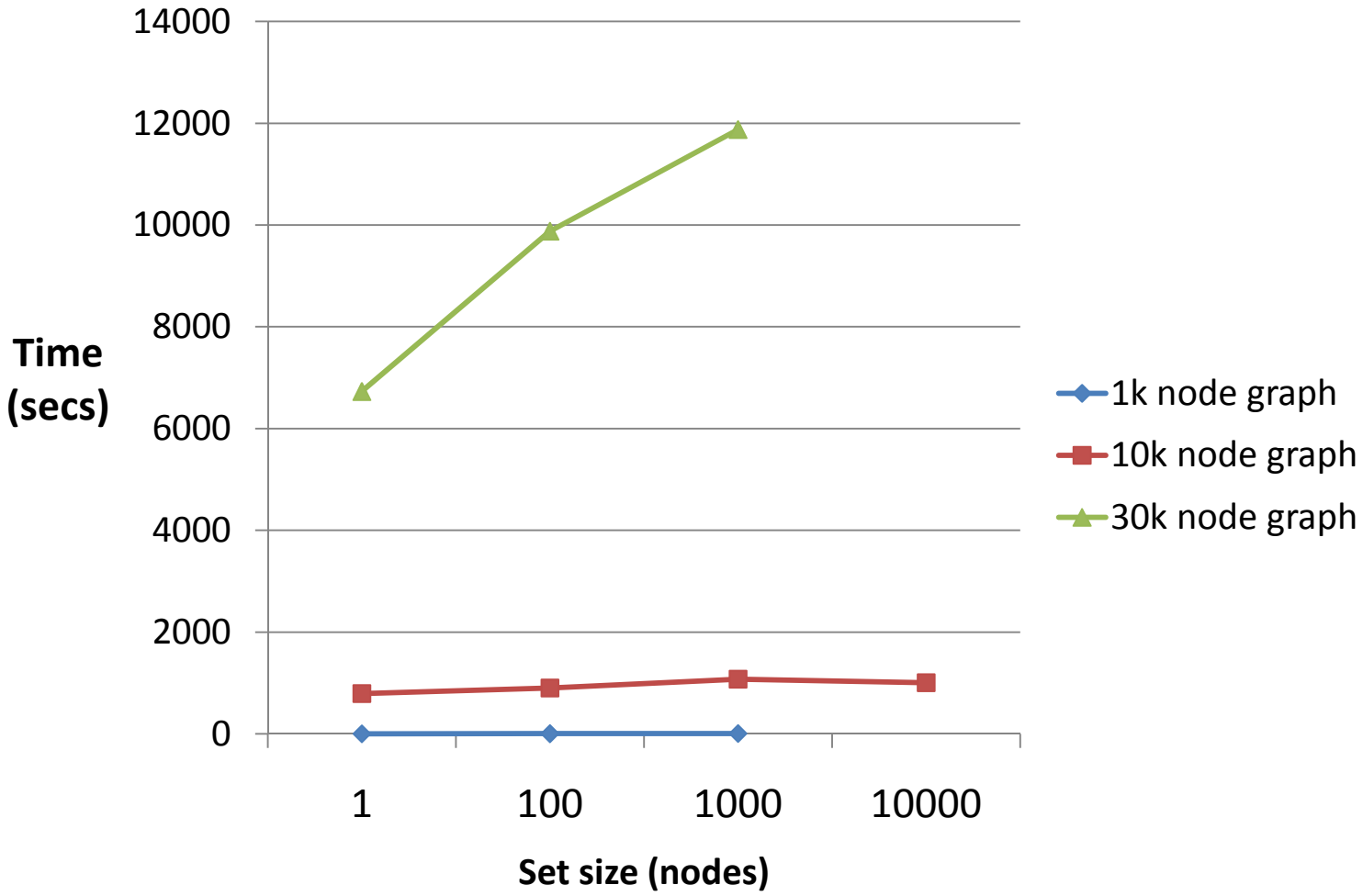


Segmented simulation: memory usage, realistic policy

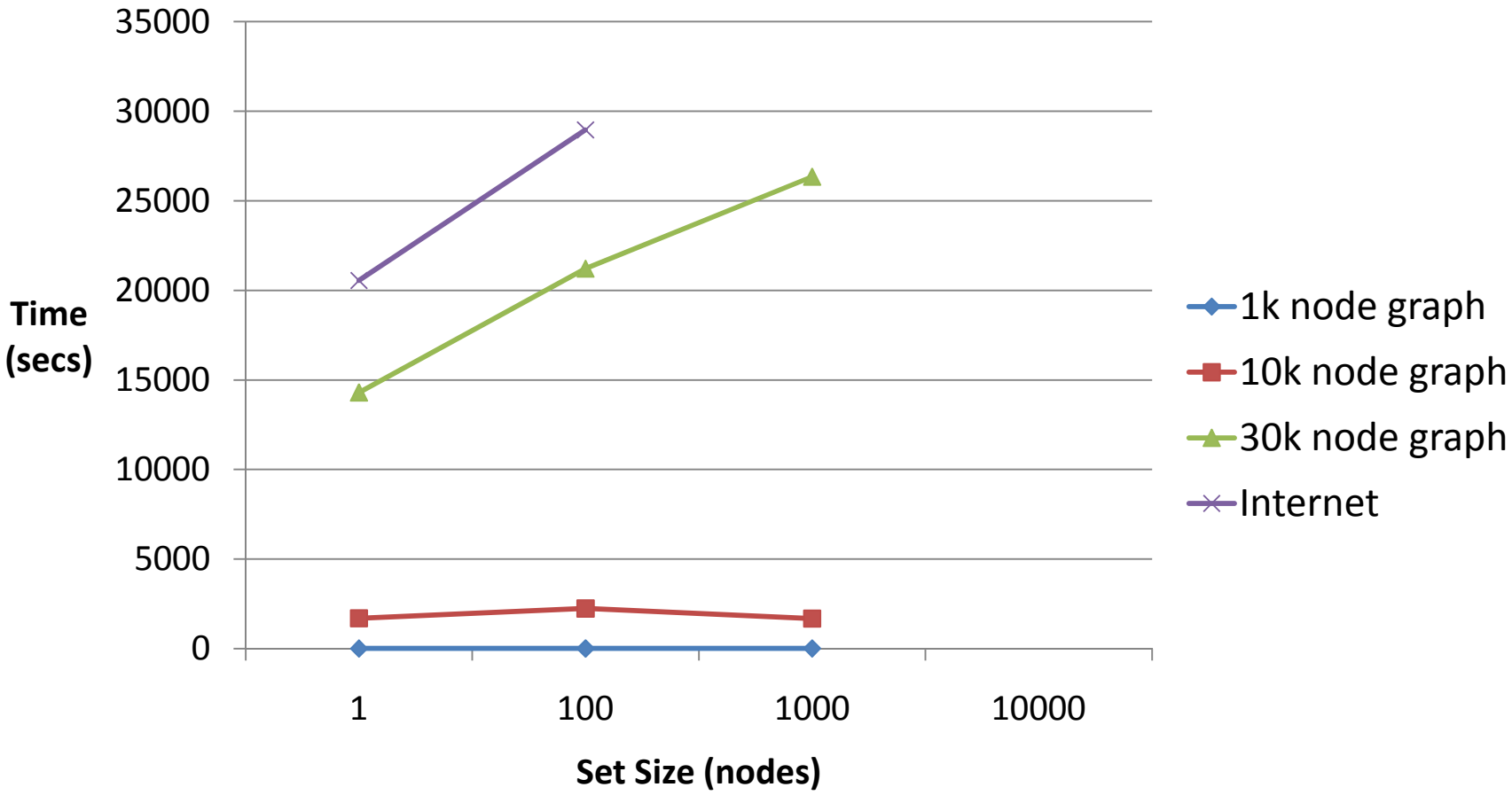


Segmented simulation: execution time

Simple policy



Segmented simulation: execution time realistic policy



Current Status

What we have:

A C++ library framework for writing routing algebras
A way to do lexical compositions of routing algebras
A simulator that can perform the simulation in a segmented fashion

Future work

adding parallelization and studying how we can improve performance further.

Our Goal:

To make it feasible to do an overnight simulation of an large scale (10k-100k) graph with a handful of machines.

We have this for a simple policy, we need to look at more complex ones.