Fast generation of spatially embedded random networks

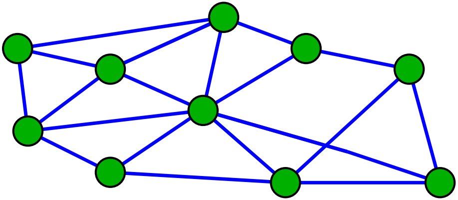
Eric Parsonage and Matthew Roughan
eric@eparsonage.com
matthew.roughan@adelaide.edu.au
http://www.maths.adelaide.edu.au/matthew.roughan/ with
Jono Tuke

UoA

July 18, 2015

Random Graphs

- Graph: $G(N, E)$
  - N = set of nodes (vertices)
  - E = set of edges (links)



- Motivation
  - simulations to test new network protocols
  - models for structured connections in an epidemic
  - ...
- Canonical example: Gilbert-Erdös-Rényi (GER) [1, 2]
  - two cases:
    - ★ $G(n, e)$: put e edges on random node pairs (n nodes)
    - ★ $G(n, p)$: put edge between each node pair with probability p

SERNs

Spatially Embedded Random Networks

- GER is too simple
  - many ways to generalise
- One approach is a SERN
  - generate random points in some metric space
  - generate links between node pairs independently with probability p_{ij}

$$p_{i,j} = f\Big(d(n_i, n_k)\Big)$$

  - NB: links are not independent, because of distance dependencies
- Motivation:
  - real actors are often in some space
  - often some "cost" to a link that depends on distance
    - ★ e.g., computer network, you have to run a cable
    - ★ e.g., epidemic, spread of infection requires transport of vector

SERN variations

- Many choices for metric space and point generation
  - typically points uniformly distributed over a unit square
  - many obvious generalisations of space and measure
- Many choices for distance functions
  common cases:
  - Random Plane Networks [3]:

$$f(d) = I(d \leq r)$$

  - Waxman [4]:

$$f(d) = qe^{-sd}$$

Simulation

- Uses for these graphs often require simulation
  - for testing protocols
  - in estimation, e.g., ABC
- Often (in the past)
  - simulation toolkits couldn't handle huge networks
  - we didn't have large-scale data anyway

but neither of these features holds anymore

- I want to be able to generate graphs
  - with thousands to millions or even billions of nodes
  - I want to generate large numbers of them
- Most existing graph generation toolkits (for cases I deal with) use $O(n^2)$ algorithms
  - usually in time
  - sometimes also in memory

but most real graphs are sparse $O(e) \ll O(n^2)$

GER

The history of the Gilbert-Erdös-Rényi (GER) is illustrative

- Almost all code for generating GERs
  - $O(n^2)$ Bernoulli trials [5, 6, 7]

```
Input: n, q, s                      // parameters of the graph
Output: E = set of edges
for i = 1..n do
    for j = i+1..n do
        calculate d_ij
        calculate p_ij = q exp(-s d_ij)
        generate r ~ U[0,1]
        if r <= p_ij then
            add (i,j) to E
        end
    end
end
```

Algorithm 1: Naive Waxman generation

- In 2005 Batagelj and Brandes [8] came up with an $O(e)$ algorithm
- Only two sets of software (I can find) use this: NetworkX and igraph

None have better than $O(n^2)$ for a SERN [9]

Batagelj and Brandes algorithm

Their approach is based on the following insight

- Think of the possible edges in a list
  - order doesn't matter
- The actual edges are selected (notionally) by Bernoulli trials
  - we can instead just do geometric jumps between edges
- Just requires the idea of homogeneous memoryless renewal process

But it doesn't work for a SERN because not all links are equal

- we might be able to transform, but
- we don't want to even calculate all of the distances!

Fast Waxman 1

We can apply the same idea as follows

$$p_{ij} = qe^{-sd_{ij}} \leq q$$

Hence, the GER random graph $G(n, q)$ provides an "upper bound" graph

- that suggests an algorithm

```
Input: n, q, s                                   // parameters of the graph
Output: E = set of edges
Construct a GER(n,q) graph G_1(N, E_1) using geometric jumps
forall the (i, j) ∈ E_1 do
    calculate d_ij
    calculate p_ij = exp(−s d_ij)
    generate r ~ U[0, 1]
    if r ≤ p_ij then
        add (i, j) to E
    end
end
```

Algorithm 2: q-jumping

How good is it?

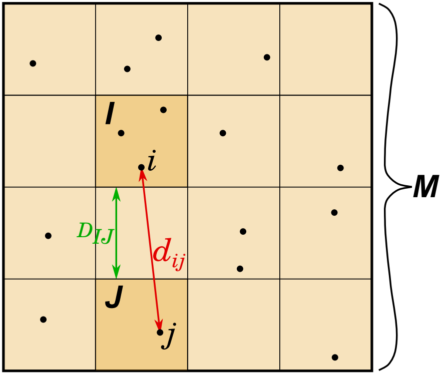
- Algorithm complexity is $O(e_1)$ where e_1 is edges in the $GER(n,q)$
  - efficiency depends on how close e is to e_1

$$\begin{aligned} \mathbb{E}[e_1] &= n\bar{k}/2 \\ \mathbb{E}[e] &= n\bar{k}\tilde{G}(s)/2 \end{aligned}$$

    - ★ $\bar{k}$ is average node degree
    - ★ $\tilde{G}(s)$ is Laplace transform of PDF of the *line-picking* problem
  - so we have an $O(e)$ algorithm, but how close to optimal optimal?
- Efficiency depends on $\tilde{G}(s)$
  - $\tilde{G}(0) = 1$
  - $\tilde{G}(s) \to 0$ for large s
  - efficiency is its good for small s
  - but for large s we have $\mathbb{E}[e_1] = \mathbb{E}[e]/\tilde{G}(s)$

What can we do for large s

Consider breaking the region into M^2 "buckets", e.g.,



We can put a lower bound $D_{IJ} \leq d_{ij}$ on the distance between nodes i and j in buckets I and J, respectively.

Fast Waxman 2

- GER skipping algorithm didn't depend on the order of the potential edges, or even that we generated them all at once
- Group potential edges into bucket-pairs (I, J)
- Perform skipping to create

$$GER(n_{IJ}, q \exp(-sD_{IJ}))$$

  upper-bound subgraph for each bucket pair
- Calculate the exact distance, and filter with probability

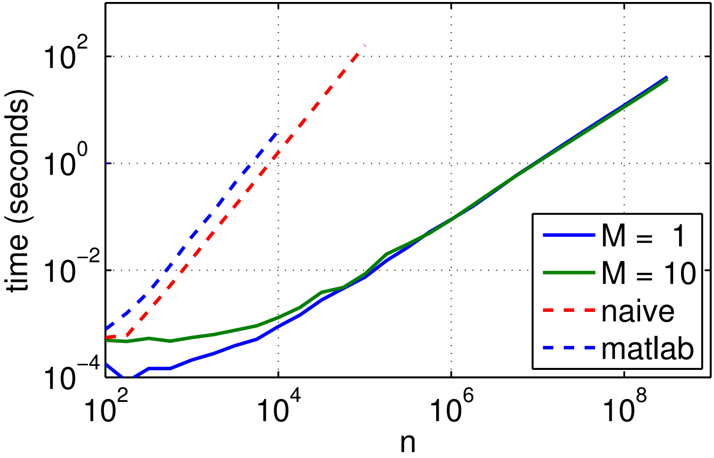
$$p_{ij} = \exp\big(- s(d_{ij} - D_{IJ})\big)$$

- Put all the edges back together

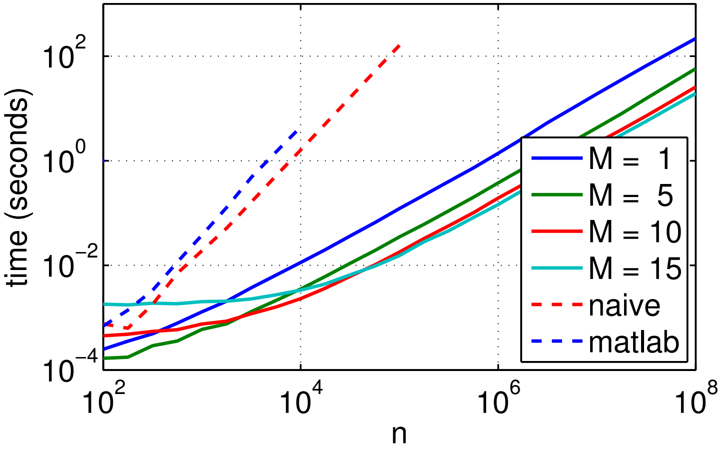
Coding

- This isn't quite trivial
  - the time to create a link in this code isn't much longer than the time to access the relevant memory
  - buckets can't be calculated on the fly
  - can't sort the points into buckets (sorting $O(n \log n)$)
  - controlling the memory allocated has to be done carefully
- The algorithm parallelises
  - only other similar example on GER [10]
  - we have a multi-thread implementation
  - its hard to avoid blocking, so speedup limited

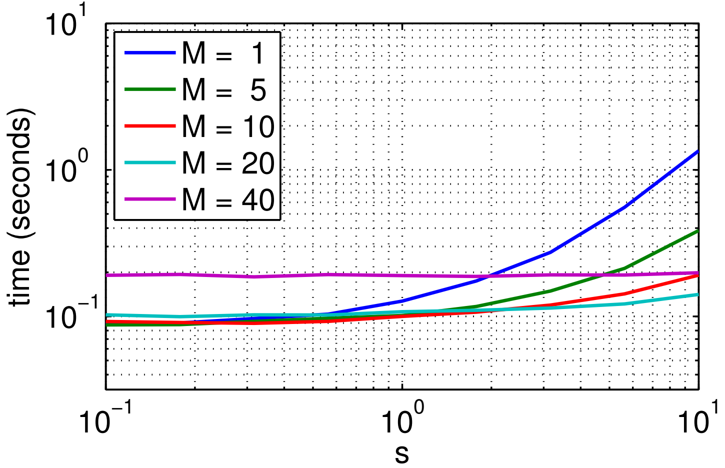
Results: small $s = 0.1$, fixed $\bar{k}$



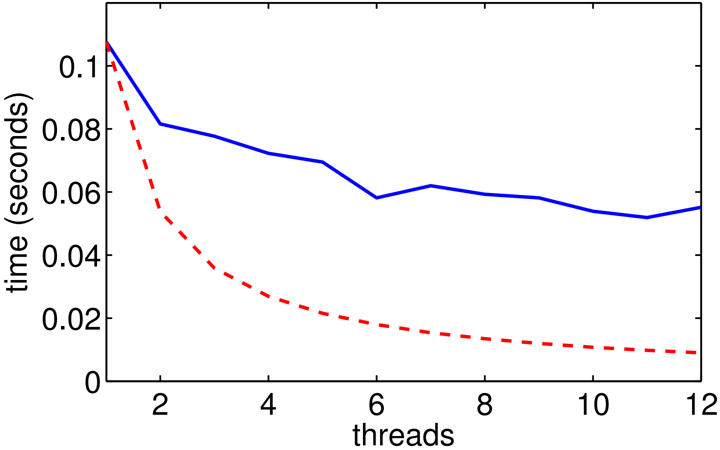
Results: large $s = 10$, fixed $\bar{k}$



Results: fixed $n = 1,000,000$



Results: fixed $n = 1,000,000$



Conclusion

- Random graphs
  - current generation techniques often naive
  - we can do better
- SERNs
  - showed how to do Waxman
  - not to hard to see how to generalise to many other cases
- There are some problems
  - what about non-convex regions
  - what about non-monotonic distance functions

E. Gilbert, "Random graphs," *Annals of Mathematical Statistics*, vol. 30, pp. 1441–1144, 1959.

P. Erdös and A. Rényi, "On the evolution of random graphs," *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, vol. 5, pp. 17–61, 1960.

E. N. Gilbert, "Random plane networks," *Journal of the Society for Industrial and Applied Mathematics*, vol. 9, no. 4, pp. 533–543, 1961.

B. Waxman, "Routing of multipoint connections," *IEEE J. Select. Areas Commun.*, vol. 6, no. 9, pp. 1617–1622, 1988.

E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A quantitative comparison of graph-based models for Internet topology," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 770–783, 1997.

M.-A. Weisser and J. Tomasik, "aSHIIP: autonomous generator of random Internet-like topologies with inter-domain hierarchy," in *18th IEEE Symposium on Modeling Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'10)*, 2010.

D. Magoni, "nem: A software for network topology analysis and modeling," in *Proceedings of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems*, MASCOTS '02, (Washington, DC, USA), IEEE Computer Society, 2002.

V. Batagelj and U. Brandes, "Efficient generation of large random networks," *Phys. Rev. E*, vol. 71, p. 036113, Mar 2005.

J. Lothian, S. Powers, B. D. Sullivan, M. Baker, J. Schrock, and S. W. Poole, "Synthetic graph generation for data-intensive HPC benchmarking: Background and framework," Tech. Rep. ORNL/TM-2013/339, Oak Ridge National Laboratory, October 2013.

S. Nobari, X. Lu, P. Karras, and S. Bressan, "Fast random graph generation," in *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, (New York, NY, USA), pp. 331–342, ACM, 2011.