Minimum Variance

Problem description

- Network of N nodes. A given node N_i
  - has a fixed upload capacity, C_i
  - has a variable number of connections, K_i
- Network is an NxN matrix of binary values (i.e. connected or not)
- We have a fitness function, *F(m)*
  - *F* takes a NxN binary matrix
  - and returns the variance in download rates

Brute-force solution

- A brute force solution would test every possible state
- For a given row, R_i
  - Number of combinations is equal to the number of subsets
  - Thus $2^m - 1$, where $m = \min(N, C_i/5)$
- For entire matrix, number of combinations close to 2^{n*n}
- Not feasible to calculate directly

So how about we rephrase this as an optimisation problem?

Optimisation problem

- Attempts to maximise (or minimise) a function, $g(x)$
  - where x is a discrete state of the problem
- Perfect fit for our situation
  - we have well defined state, i.e. the binary matrix
  - and a function to minimise, $F(m)$
- Many techniques to solve or approximate this class of problem
  - Hill climbing
  - Random walks
  - Simulated annealing
  - etc.

Lets look at hill climbing

Hill climbing

- Start with some state
  - doesn't matter which, can be random
- Each cycle, pick a neighbouring state to move to
  - pick neighbouring state with best value of fitness function
- When we can't find a better neighbour, we have found a maximum
  - Unfortunately, it might just be a local maxima
- We can alleviate the problem of local maxima
  - Stochastic hill climbing: pick a random better neighbour
- Still not guaranteed to find an optimal solution
  - But might give us a helpful approximation

I implemented a stochastic hill climber but it always gets stuck in local minima

We need something more reliable
Perhaps simulated annealing?

But wait, what does our simulation do?

Our simulation

- Represents state as a sparse NxN matrix

- Each node attempts to maximise its own download rate
  - by moving to neighbouring state
  - and regularly reevaluating metric
  - basically, hill climbing on smaller scale

- Maximising majority of download rates ~= minimising variance
  - Because large peers unable to maximise their download
  - Therefore, we are optimising to minimise variance
- Unlike a hill climber, we allow backtracking
  - never converges to a fixed solution
  - never gets stuck either

- I believe our simulation is approximating minimum variance