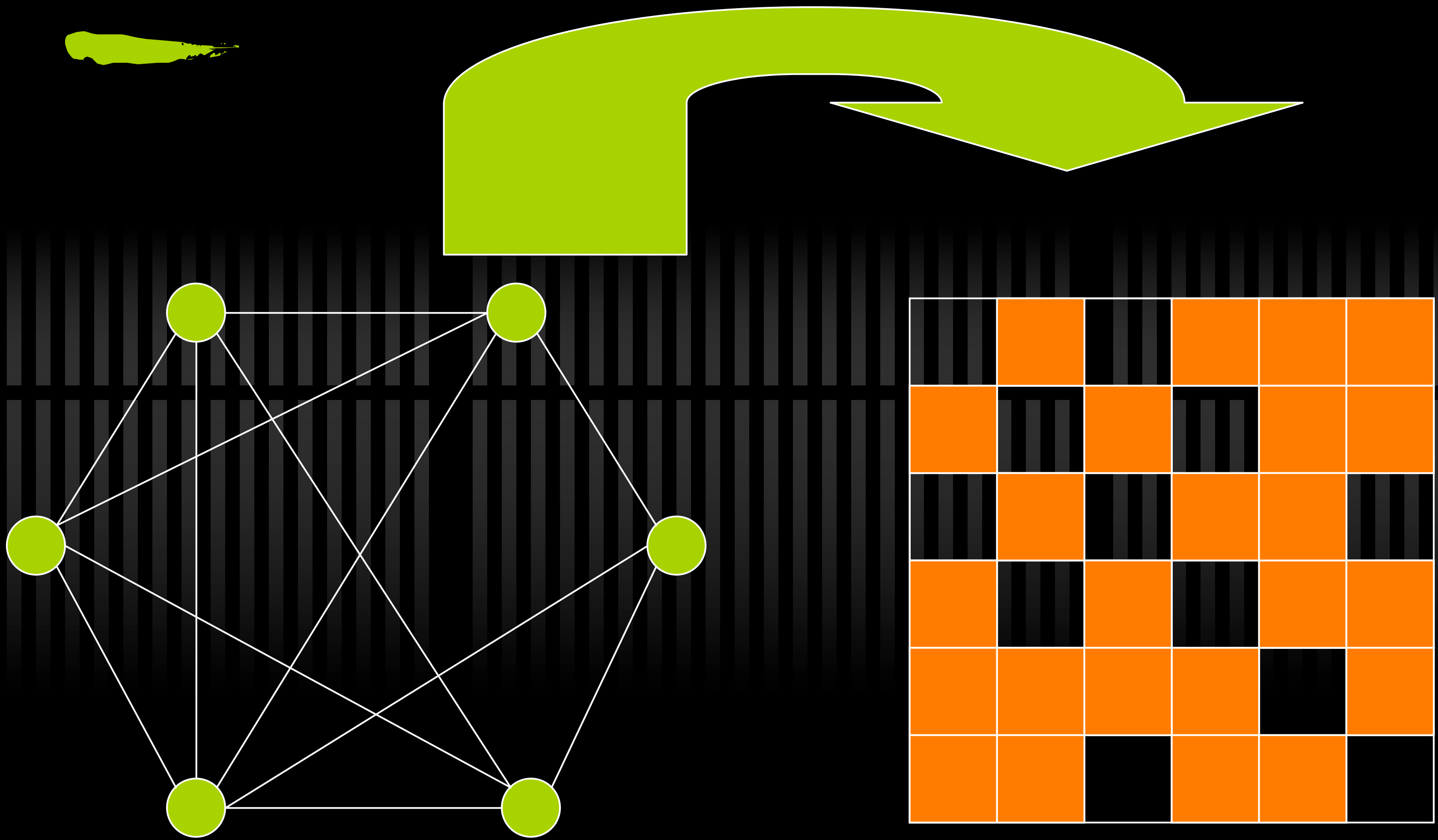
A Parallel Algorithm For Maximal Clique Detection

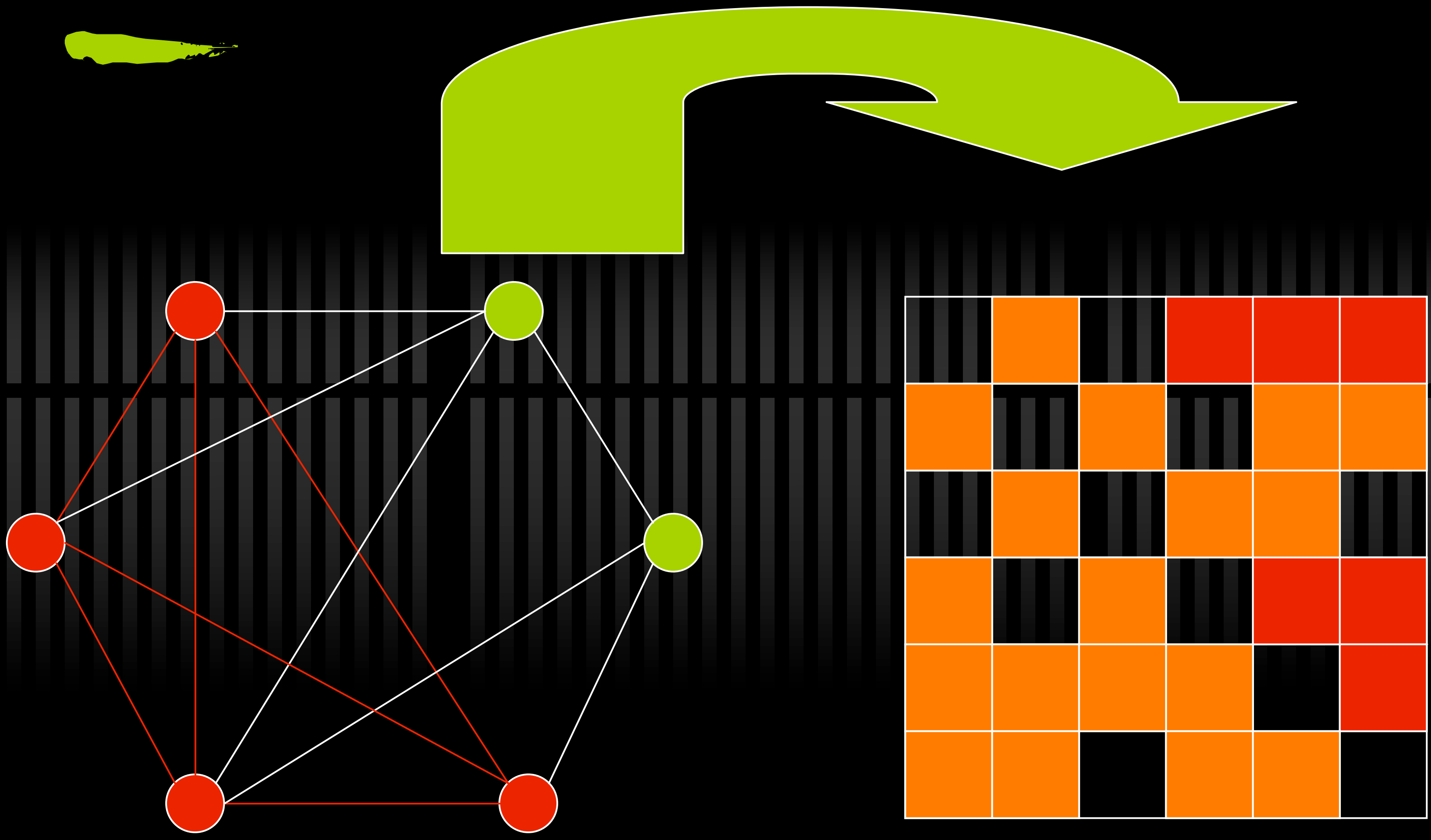
By Steven Kapturowski

CSE 633, Spring 2009

The Problem



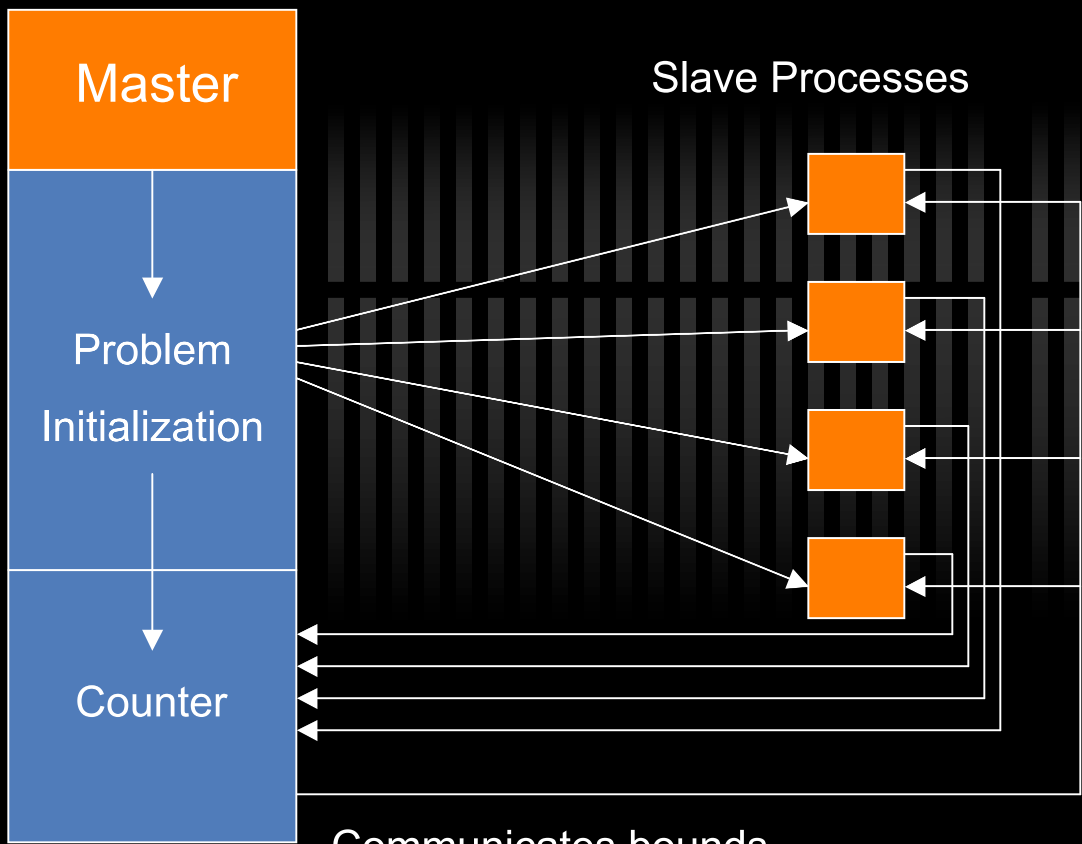
The Problem



The Sequential Solution

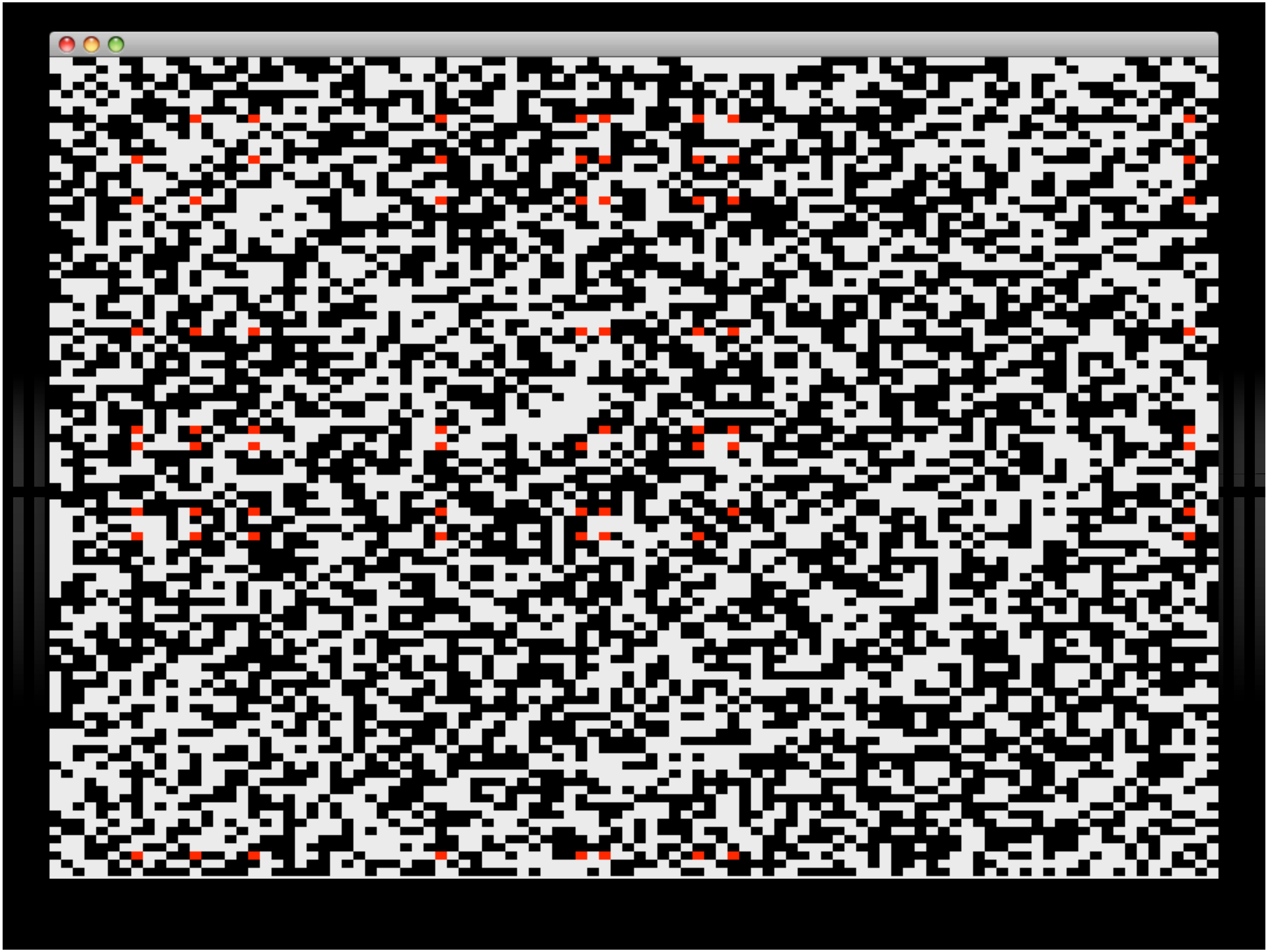
- We enumerate through all graphs of size n = 3 and check for completeness
- For each complete graph check all graphs of size n+1 in which it is included
  - Note: we only need to check nodes greater than the highest value in complete graph!
- Repeat Step 2

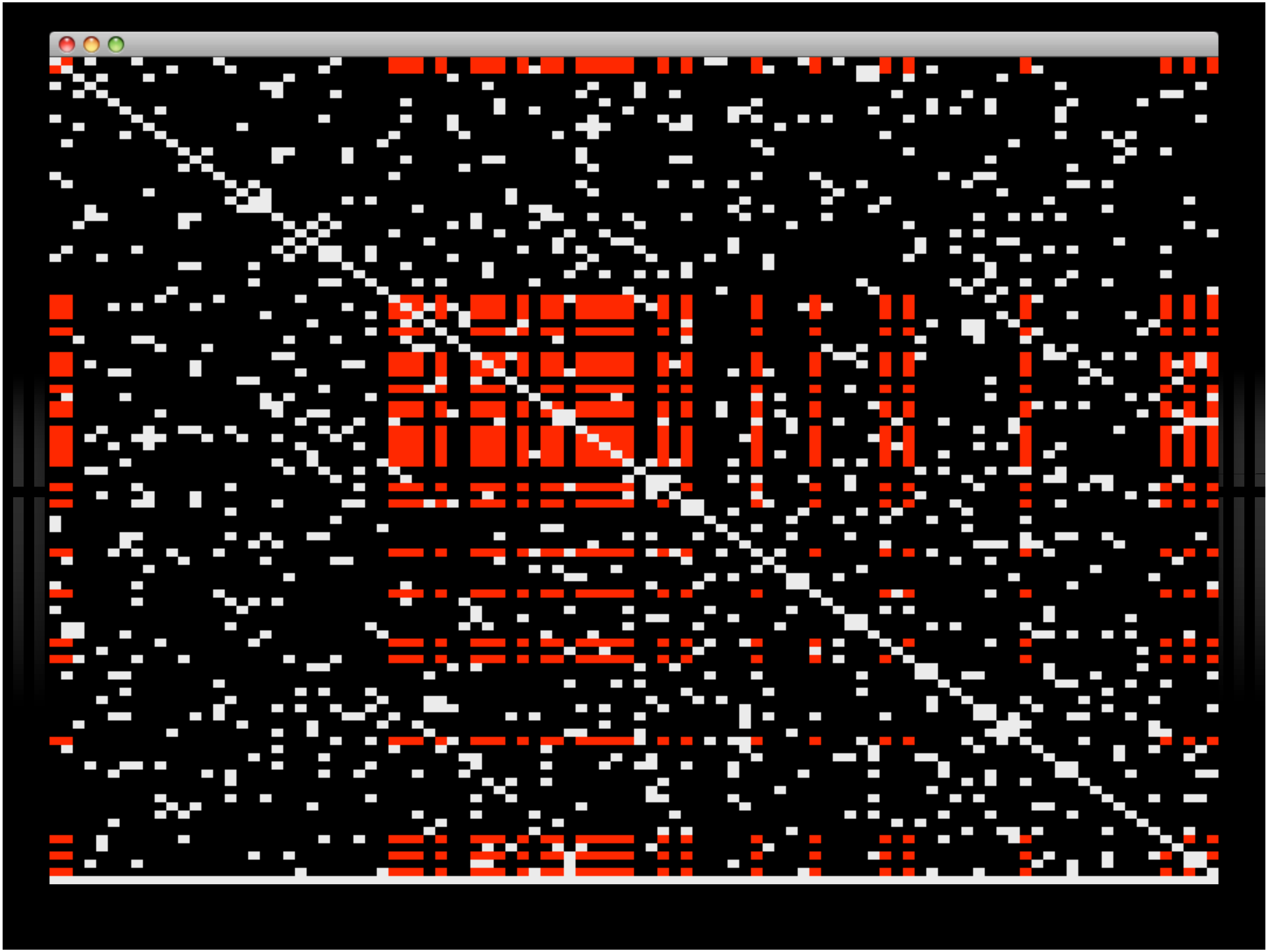
Parallel Solution

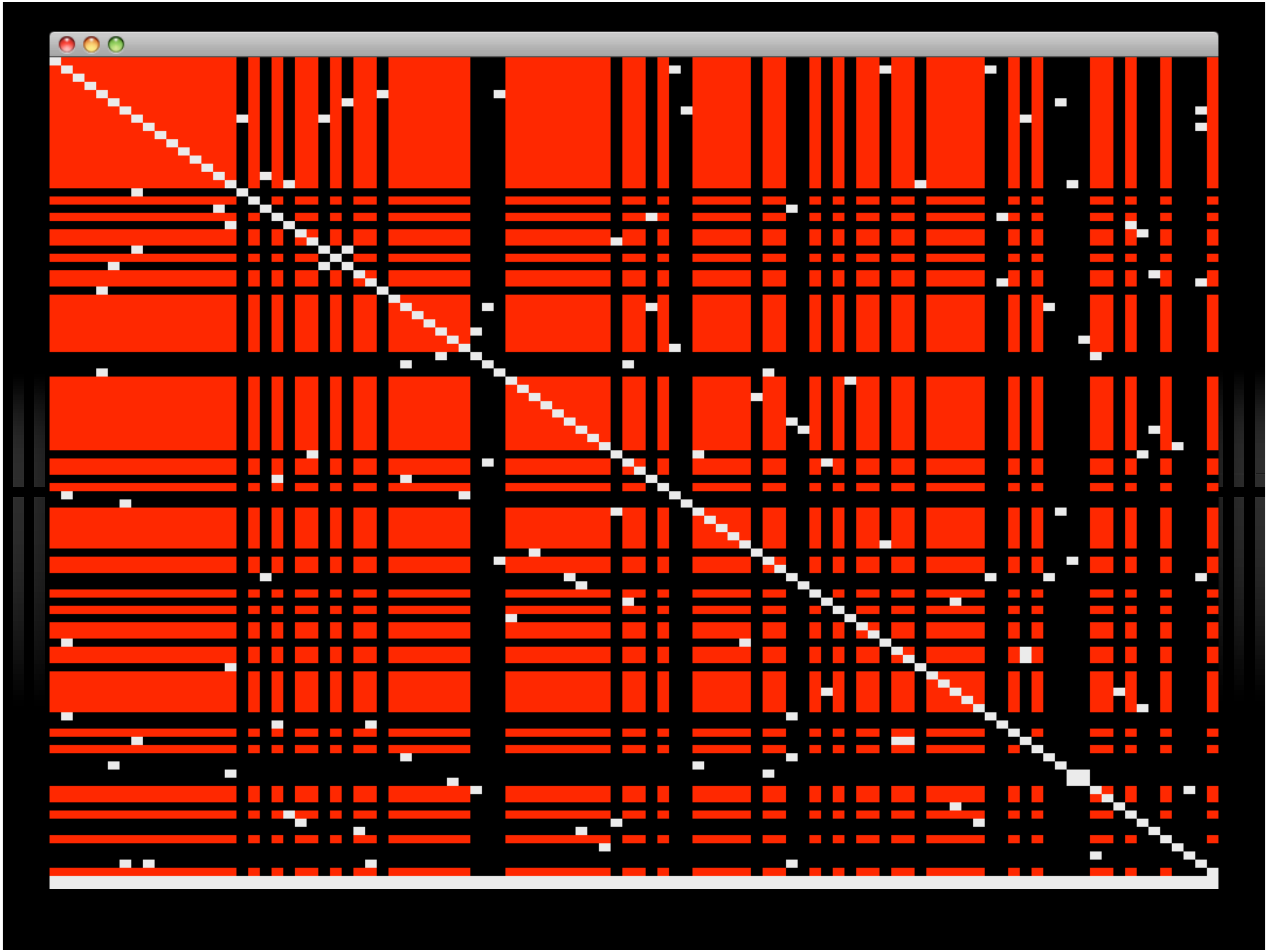


Communicates bounds
At staggered time intervals

Note: we *must* take wide intervals, otherwise the overhead will dominate over any gains

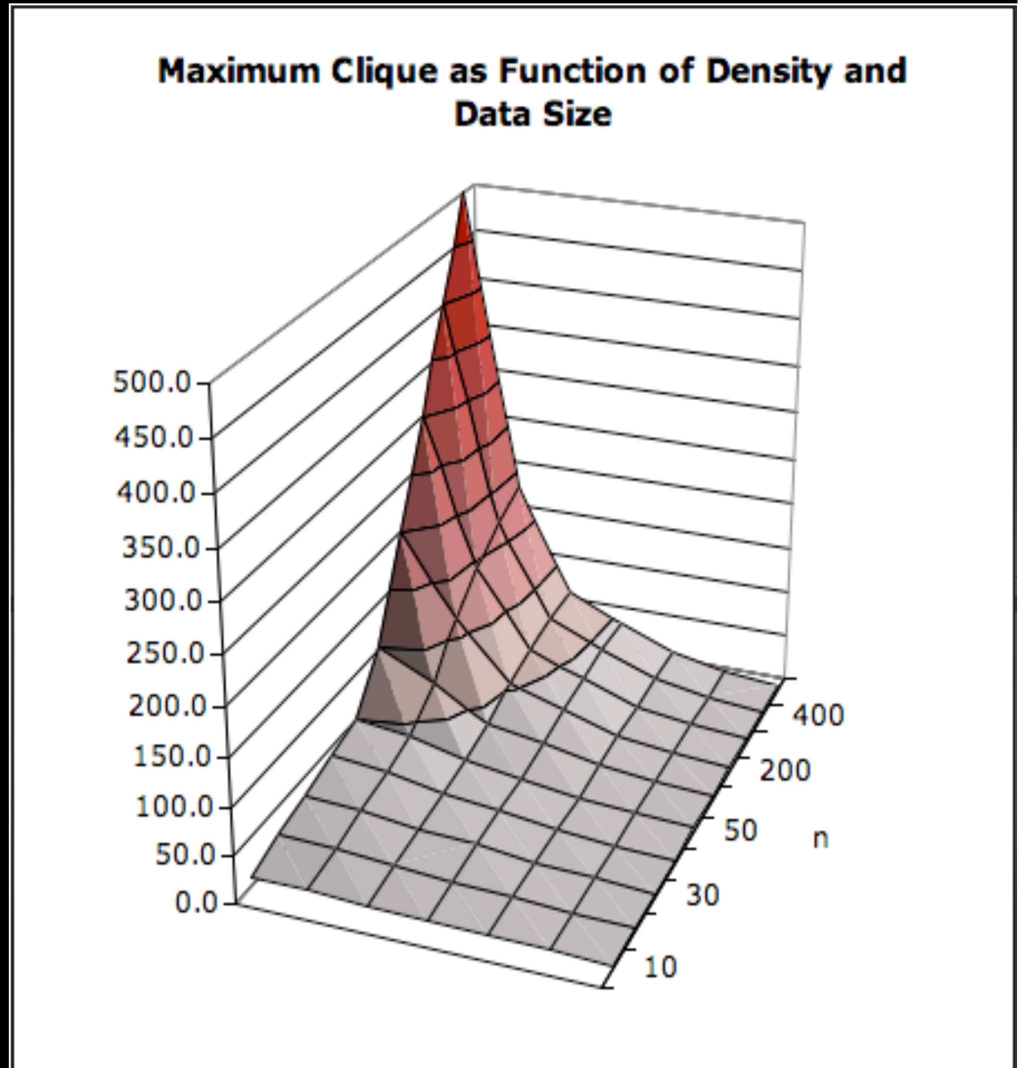




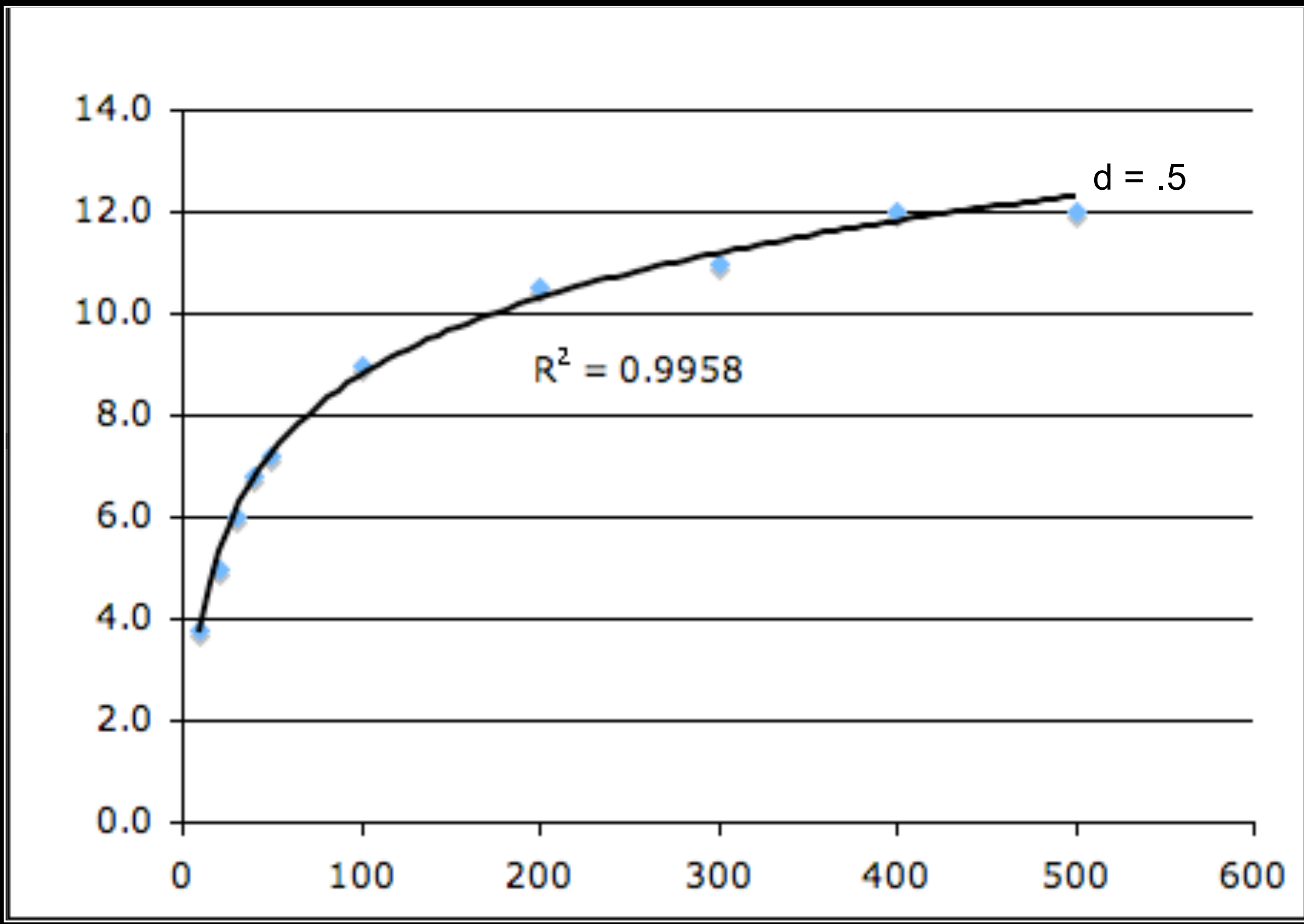


Run Time

- *A naïve approach will search all 2^n permutations*
- But remember, we never search larger graphs than we need to…
- *Must be a function of M(x)*
- *M(x) appears to grow as Ø(log[n]) for wide density range*



Maximum Clique Vs Number of Vertices



Note: via our probabilistic method of construction, a graph of n vertices will have as its expected number of edges:

$E = d(n^2-n)/2$

Where d is the density of the graph

Binomial Coefficients

- Notice that $2^n = \sum_{i=0}^{n} \binom{n}{i}$
- Which we can cut off early…

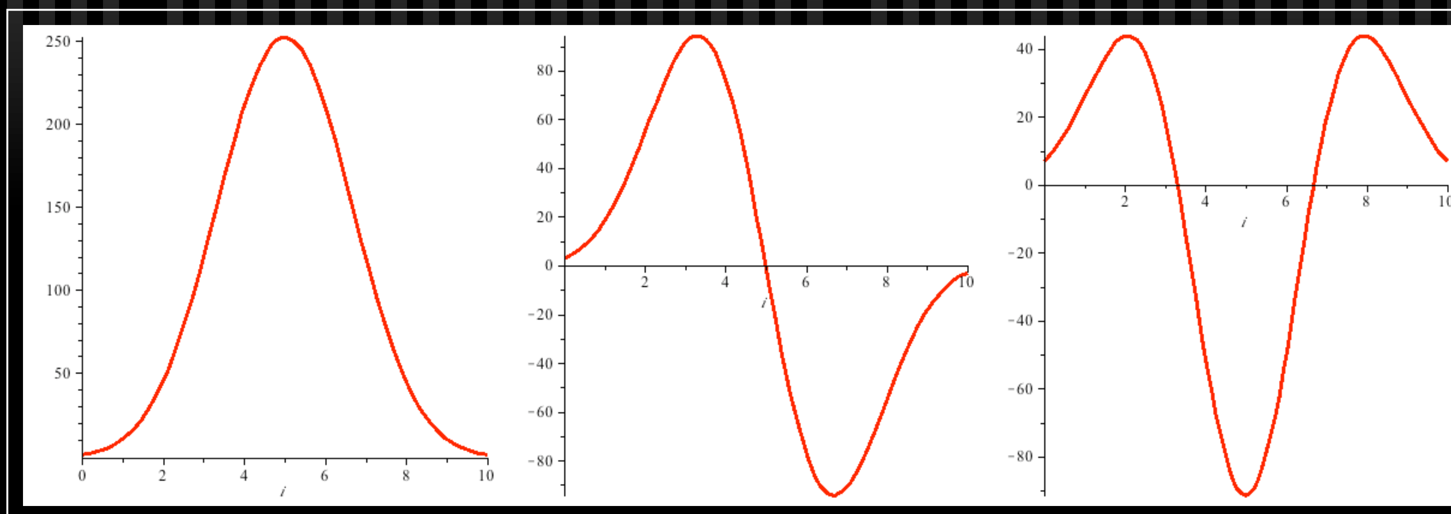
M(n)

$$\sum_{i=3} i \binom{n}{i}$$

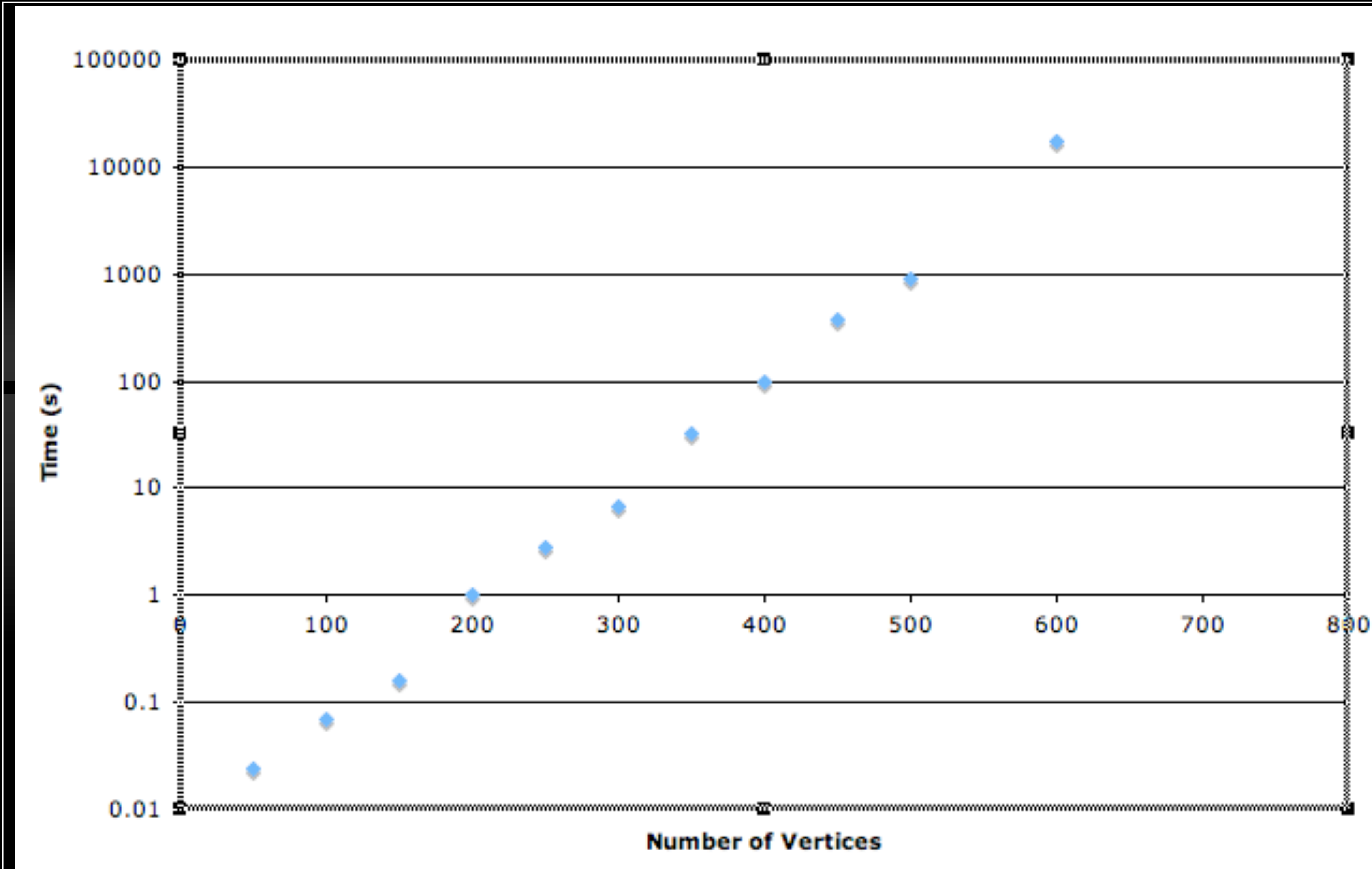
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
1 8 28 56 70 56 28 8 1
1 9 36 84 126 126 84 36…
1 10 45 120 210 252 210…
1 11 55 165 330 462 462...
1 12 66 220 495 792 924…
1 13 78 286 715…
1 14 91 364 1001…
1 15 105 455 1365…
1 16 120 560 1820…

Closing the Gap

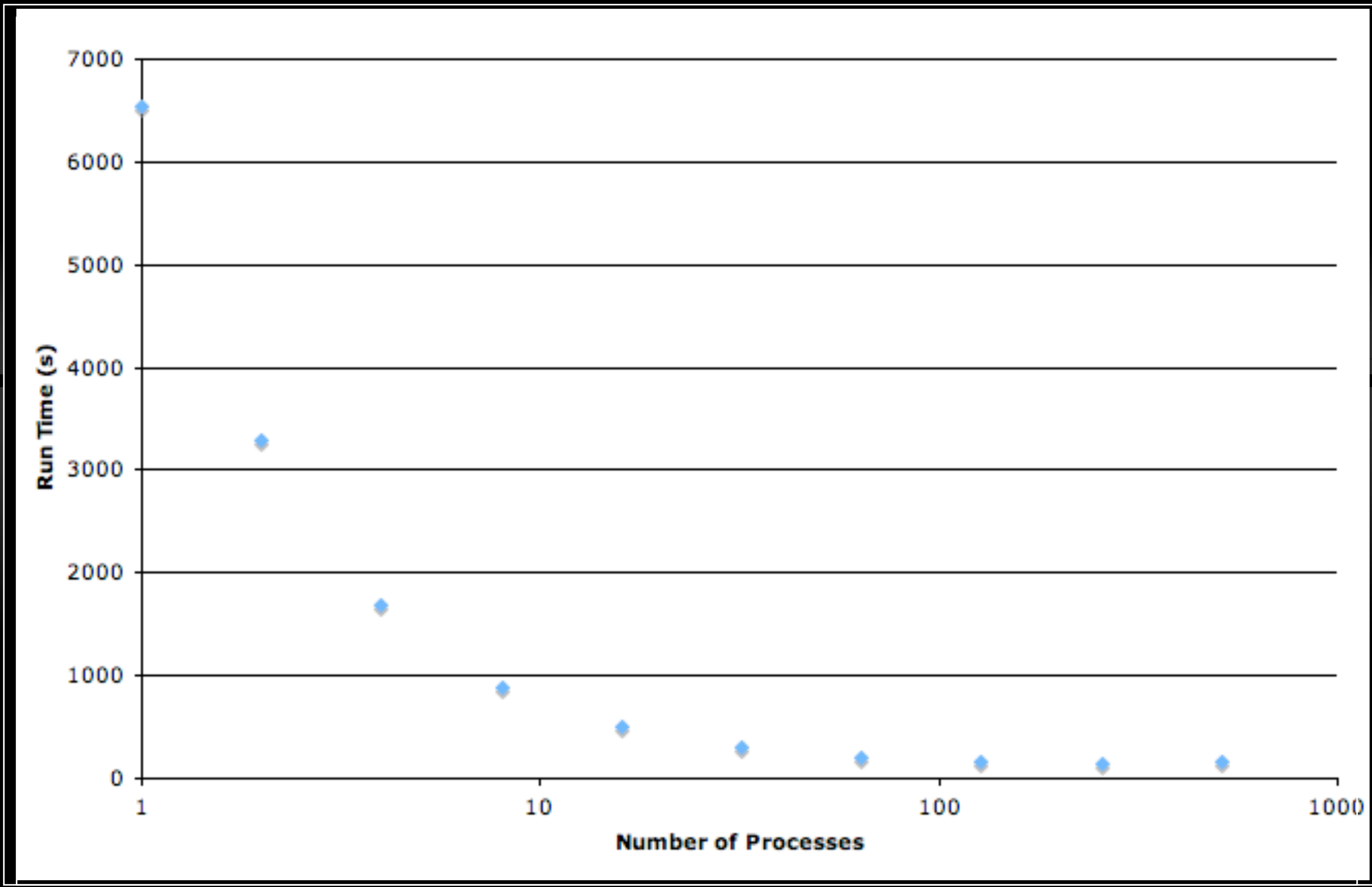
- We want to kill off all but the last term *without* imposing a strict identity on M(n)
- We must guarantee increasing slope over some region and verify true for the bottom end
- We want some domain of x, ≥ 0 such that $\lim_{n \to \infty} \partial_i^2 \binom{n}{i} i \Big|_{i = \frac{xn}{2x+1}} = \infty$



Run Time Vs. Data Size



Run Time Vs. Number of Processes



Approximate Solution

- ✓ For *any* graph size the algorithm *will* find good approximation quickly

(even though it will continue searching for exponential time to *ensure* global quality)

But now for the catch…

- ✓ This is merely an artifact of the homogeneity implicit in our method of construction of the graph
- ✓ In general this will not work
- ✓ But we can recover this characteristic if we order nodes based on edge density