CSE 527

CAST: a clustering method with a graph-theoretic basis

Larry Ruzzo

Talks this week

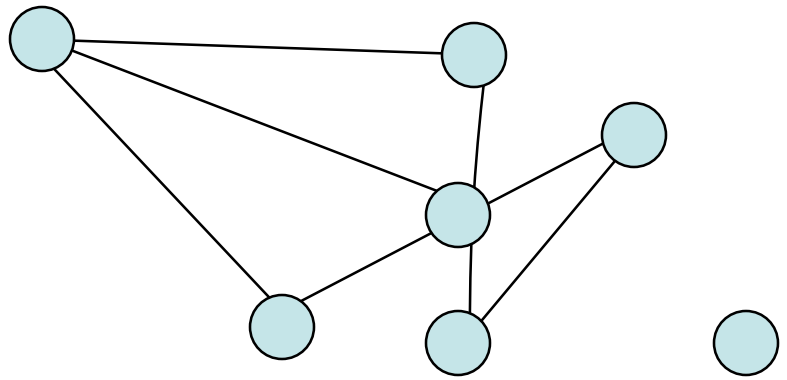
- Today - Dr. Terry Hwa, Professor of Physics, UC San Diego "Complex Transcriptional Logics From Simple Molecular Interactions" 3:30, Hitchcock 132
- Fri - Me, "Improved Gene Selection for Classification of Microarrays" 3:30 Loew 102

More Reading

- Paper on quantitative or computational analysis of microarrays (clustering, normalization, differential expression,…)
- Again, send me a very short comment on it, say by Monday

Graphs

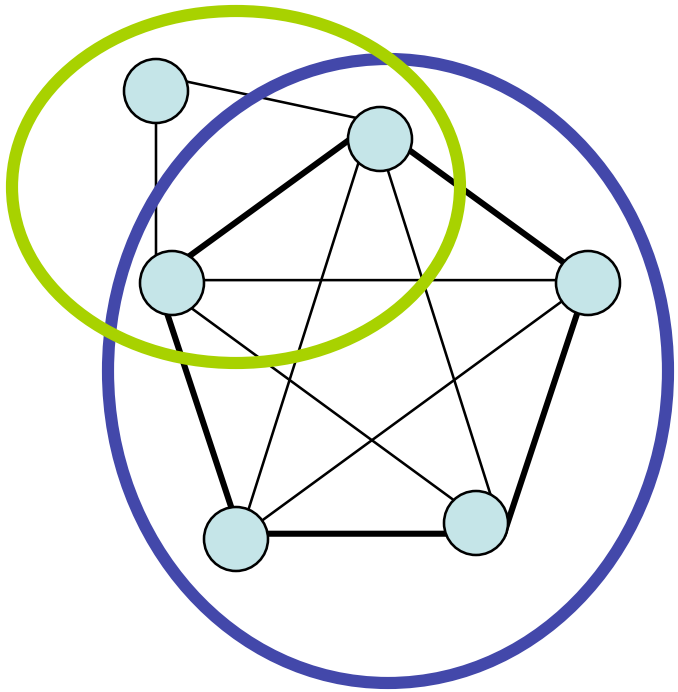
- Vertices
- Edges
- G = (V, E)



- Why? Model simple pairwise relations, e.g.
  - Vertices = genes
  - Edges = “similar” pairs of genes

Cliques

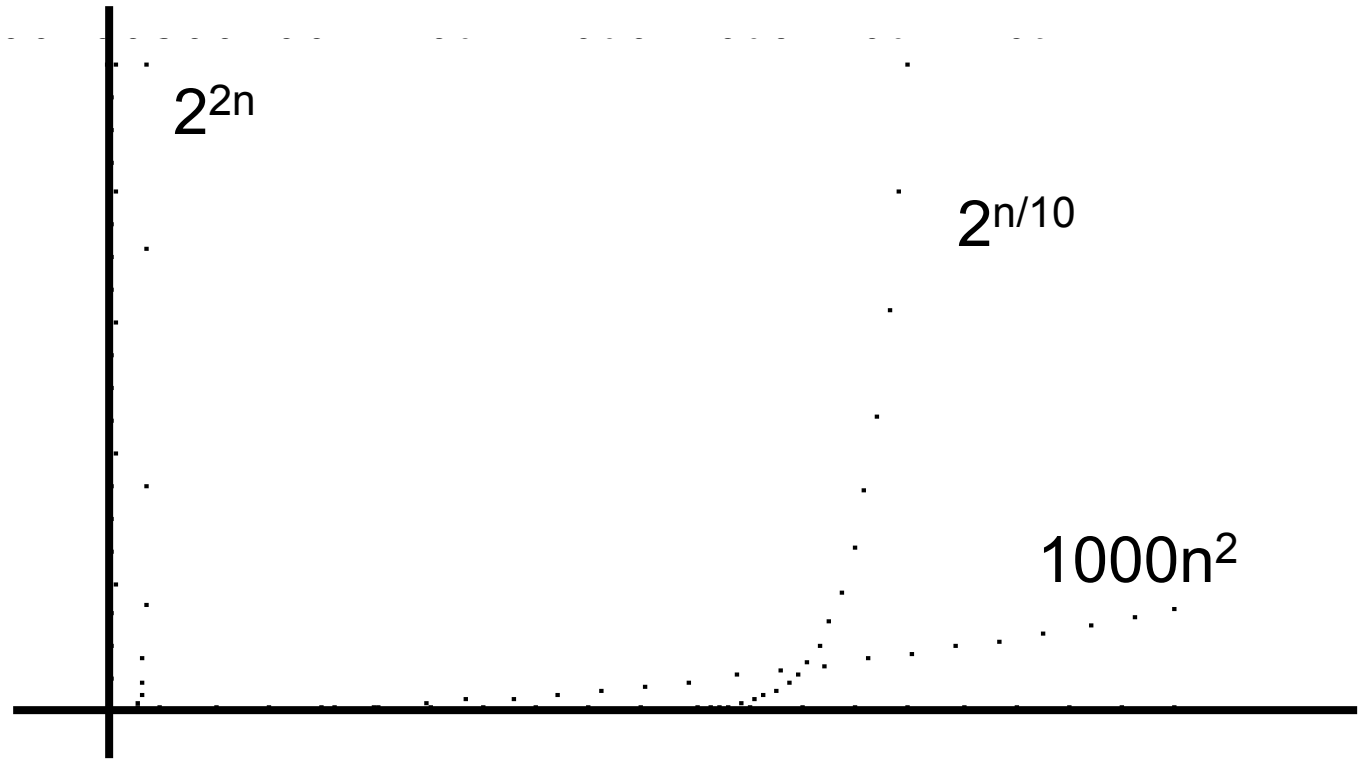
- K-clique is a set of k vertices, each of which is connected (directly) to all the others.
- Why? A “cluster” - all “similar” to each other



Clustering -> Finding Cliques

- 2- cliques -- just edges, easy $\binom{n}{2}$ 4950
- 3-cliques -- triangles $\binom{n}{3}$ 161700
- 10-cliques -- hmmm… $\binom{n}{10}$ 1.7×10^{13}
- General -- $\binom{n}{k} \geq \left(\frac{n}{k}\right)^k$ gets big fast…

Polynomial vs Exponential Growth



Asymptotic Analysis

- How does run time grow as a function of problem size?

  n^2 or $100\ n^2 + 100\ n + 100$ vs 2^{2n}

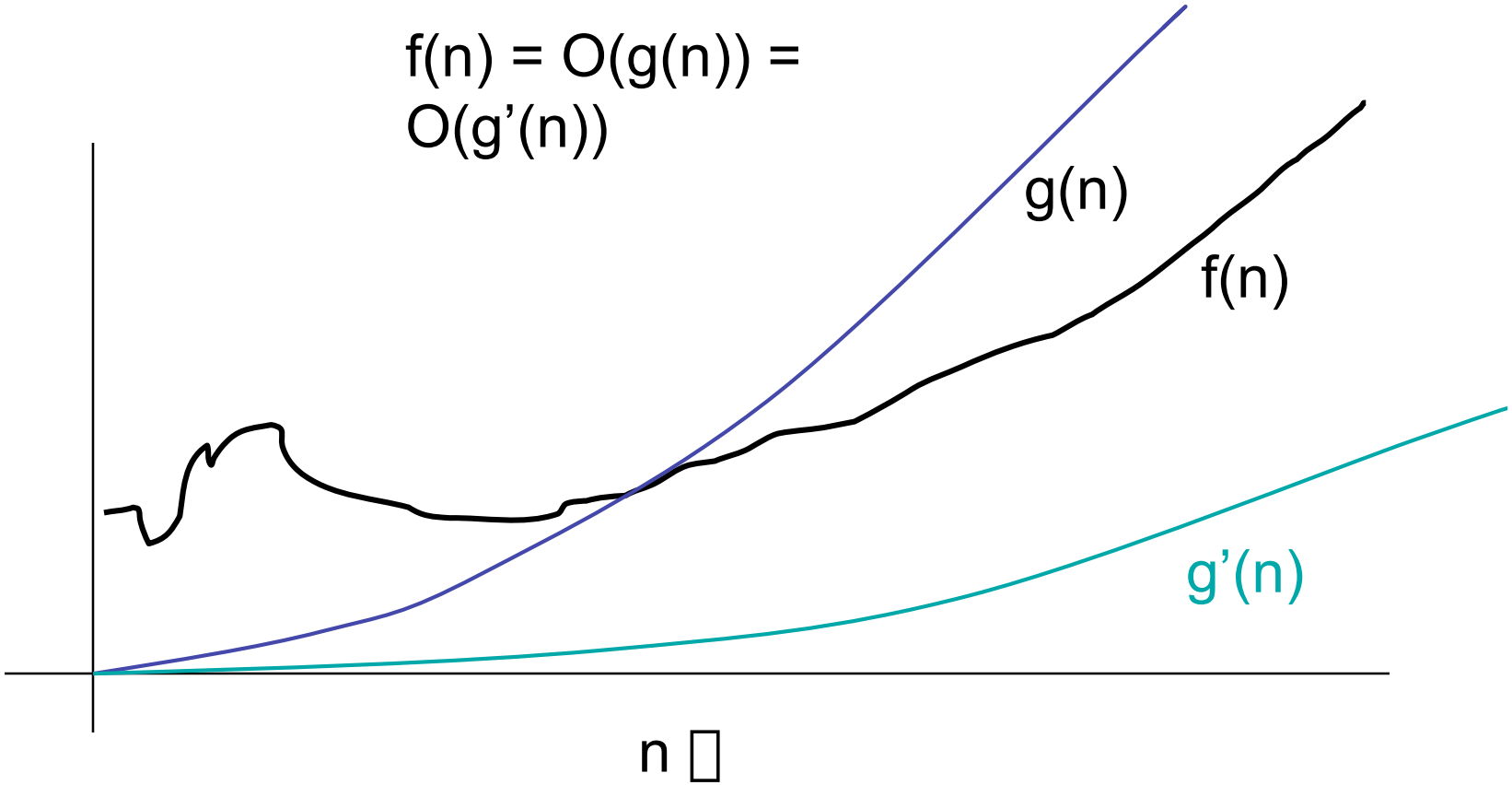
- **Defn:** f(n) = O(g(n)) iff there is a constant c s.t. |f(n)| ≤ cg(n) for all sufficiently large n.

  $100\ n^2 + 100\ n + 100 = O(n^2)$ [e.g. c = 101]

  $n^2 = O(2^{2n})$

  2^{2n} is *not* $O(n^2)$

Big-O Example



Utility of Asymptotics

- “All things being equal,” smaller asymptotic growth rate is better
- All things are never equal
- Even so, big-O bounds often let you quickly pick most promising candidates among competing algorithms
- Poly time algorithms often practical; non-poly algorithms seldom are.

2nd problem: Noise

- Given graph H which is a collection of (large) cliques, *corrupt* each edge/nonedge with probability $\eta < 1/2$
- Call result G
- Problem: Find (approximate) H given G as input

Simplified problem

- Given graph H which is a collection of (large) *disjoint* cliques, corrupt each edge/nonedge with probability $\delta < 1/2$
- Call result G
- Problem: Find (approximate) H given G as input
- Success if $|H' \oplus G| \leq |H \oplus G|$

Notes

- Without noise, simplified problem is very easy - any edge in clique leads to rest
- Noise destroys that
- But - not totally

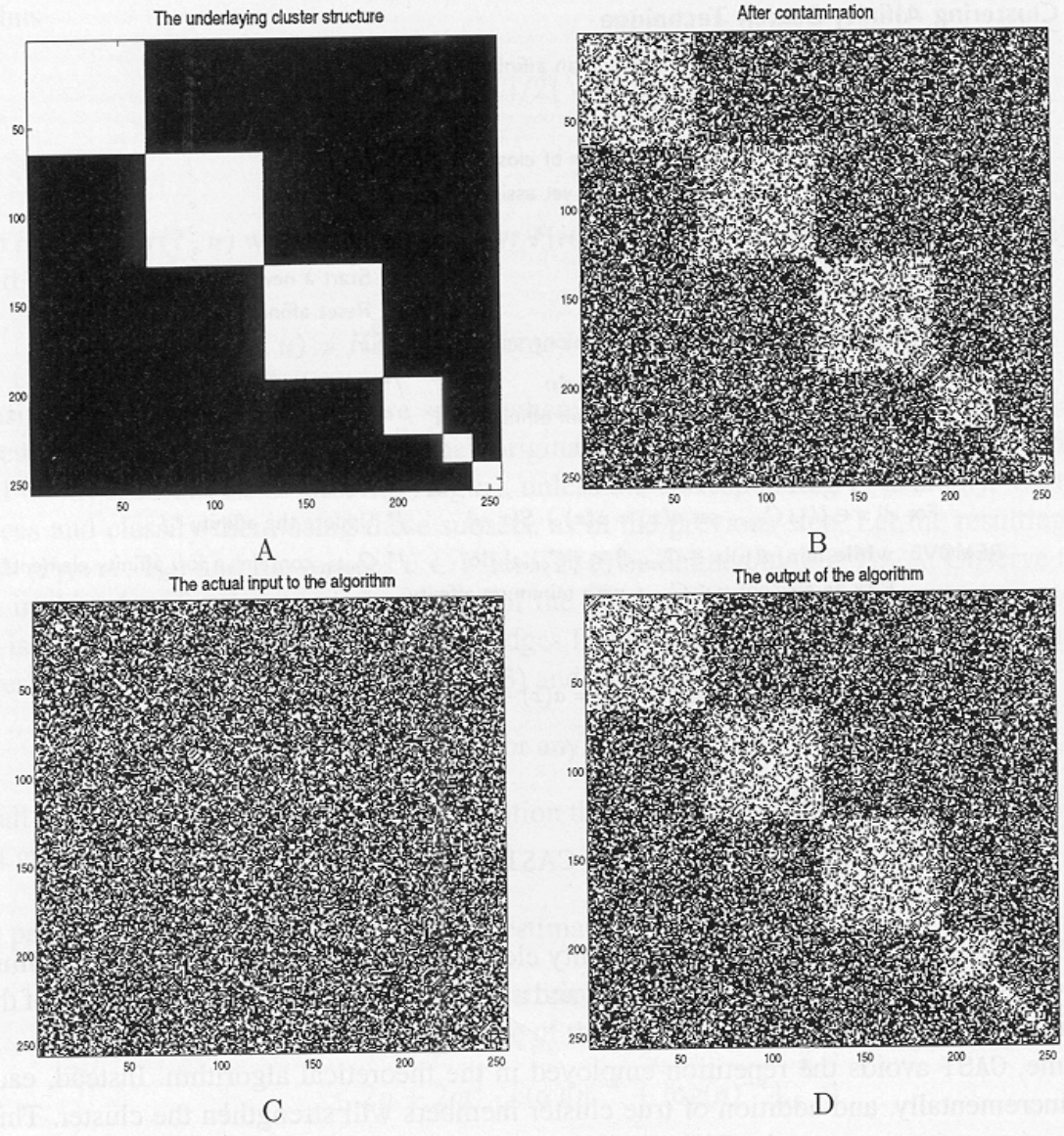


FIG. 3. A visual representation of the simulation process. Cluster structure is $(1/4, 1/4, 1/4, 1/8, 1/16, 1/16), n = 256$. (**A**) The adjacency matrix of the original clique graph before the introduction of errors. Position (i, j) is black iff i and j belong to the same cluster. (**B**) The same matrix after the introduction of errors. Note that the cluster structure is still visible for all but the smallest clusters. (**C**) Same as B but entries are randomly permuted. This is actual input to the algorithm. The challenge is to reconstruct B, and hence the clusters, from C. (**D**) Matrix C reordered according to the solution produced by the algorithm. With the exception of perhaps the smallest clusters, the essential cluster structure is reconstructed.

Ben-Dor, et al., JCB 1999

Main Result

For all alpha < 1/2, epsilon>0, delta > 0 there is an algorithm A and constant c (depending on alpha, epsilon and delta) such that for all clique graphs H with disjoint cliques of minimum size at least epsilon*n, A successfully recovers H' from the alpha-corrupted version G of H with probability > 1-delta, running in time $O(n^2(\log n)^c)$

Fine print

- For “reasonable” choices of parameters, like epsilon = 10%, alpha = .25, the analysis says c < about 600:

$$n^2(\log n)^{600}$$

an unpleasant function whenever log n ≥ 2 …

More Fine Print

- Many model assumptions are very simplistic:
  - "similarity" is all-or-none
  - Disjoint cliques
  - Independent errors
  - Adding/deleting edges equi-probable
  - Known error rate
  - …

Nevertheless

- Interesting, since analysis is probably pessimistic, and intuitions are valuable, even if you never implement exactly this algorithm

Key idea

- Suppose I know k elements of one clique v1, v2, …, vk (a “core”)
- Given another vertex x, is it in same clique?
  - If so, neighbor of k(1-alpha) core members
  - If not, neighbor of k(alpha) core members
- Alpha < .5 < (1-alpha), so join if neighbor of more than half of core. Probability of failure declines exponentially with k.

How do you find a core?

- Brute force: try all subsets of size O(log n)
- Too slow; more subtle - try subsets of size O(log n / log log n) to classify a sample of size O(log n)

Practical Heuristic

- Copen := the unassigned vertex of max average affinity
- Repeat until no change
  - U := unassigned vertex of max affinity to Copen; if > thresh, add
  - If none, v := vertex in Copen of min avg affinity; if < thresh, remove
- Close this cluster & restart

Final pass

- Repeat until no change (or iteration limit):
  - Move each element to the cluster to which it has max affinity

Overall

- Simple to implement
- Reasonably fast in practice
- Gave good results in many tests

Reference

A. Ben-Dor, R. Shamir, Z. Yakhini, “Clustering Gene Expression Patterns” Journal of Computational Biology, v 6 # 3/4 (1999) pp 281-297