Computing the Permanent

David Judd
daj2105@columbia.edu
December 11, 2007

The permanent

$$per(A) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)}$$

where S_n is the set of all permutations of the numbers 1, 2, ..., *n*.

- If A is a 0/1 adjacency matrix representing an *n* x *n* bipartite graph, per(A) is the number of perfect matchings in the graph.
- Exactly computing the permanent, even of 0/1 matrix, is in general #P-complete, and so infeasible.

An approximation algorithm

- Based on a Markov chain which generates perfect matchings almost uniformly at random
- Two stage
- 1) Compute weights which make Markov chain results almost uniform
- 2) Compute permanent using Markov chain with near-ideal weights
- Discovered by Jerrum, Sinclair, & Vigoda (2004)

The Markov chain

- States are perfect matchings or 'near-perfect' matchings, which have exactly 2 unmatched nodes or 'holes'
- Define 'activity' $\lambda(u,v)$ for each edge, weight $w(u,v)$ for each possible pair of holes, and activity $\Lambda(m)$ for each matching m, where...
- $\Lambda(m)$ is the product of the activities of all edges in m, times $w(u,v)$ if m is missing nodes u and v
- So, at each step, pick a random edge, if it's in the matching remove it, otherwise add it
- But only actually move from state m to the new state m' with probability $\Lambda(m') / \Lambda(m)$

Computing the weights

- By simulated annealing
- Activities $\lambda(u,v)$ start uniformly, so that weights are easy to calculate, at $\lambda(u,v) = \max(\mathbf{A})$, and decrease to $\mathbf{A}(u,v)$, which we
- Process is slow so that weights which are close to ideal for activities at step t remain close for activities at step $t+1$
- Each weight $w(u,v)$ is updated at each step by the ratio of perfect matchings to matchings with holes at u and v in a sample from the Markov chain

Weights to permanent

- We know that at initialization the sum $\Lambda(\Omega)$ of $\Lambda(m)$ over all matchings m in the Markov chain state space Ω is $(n^2 + 1)n!(A_{max}/A_{min})^n$
- We know that at termination the sum $\Lambda^*(\Omega)$ is approximately $(n^2 + 1)\Lambda^*(P)$ where P is the set of all perfect matchings
- We can estimate the ratio $\Lambda_{t+1}(\Omega) / \Lambda_t(\Omega)$ with a sampling from the Markov chain and the weights from simulated annealing
- So, we can estimate $|P|$ as a product of ratios

Complexity

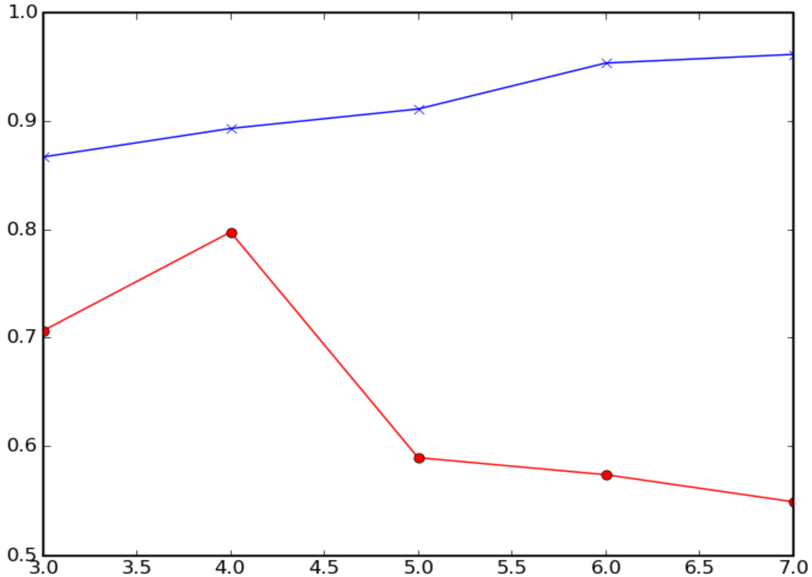
- Upper bounds from Bezakova, Stefancovik, Vazirani & Vigoda (2005)
- Markov chain running time = $O(n^4 \log n)$
- Sample sizes = $O(n^2 \log n)$ or $O(n \log n)$ in different stages
- Phases of simulated annealing = $\Theta(n \log^2 n)$
- Total, neglecting ε, $O(n^7 \log^4 n)$
- But none of these bounds is tight...

Estimating the constants

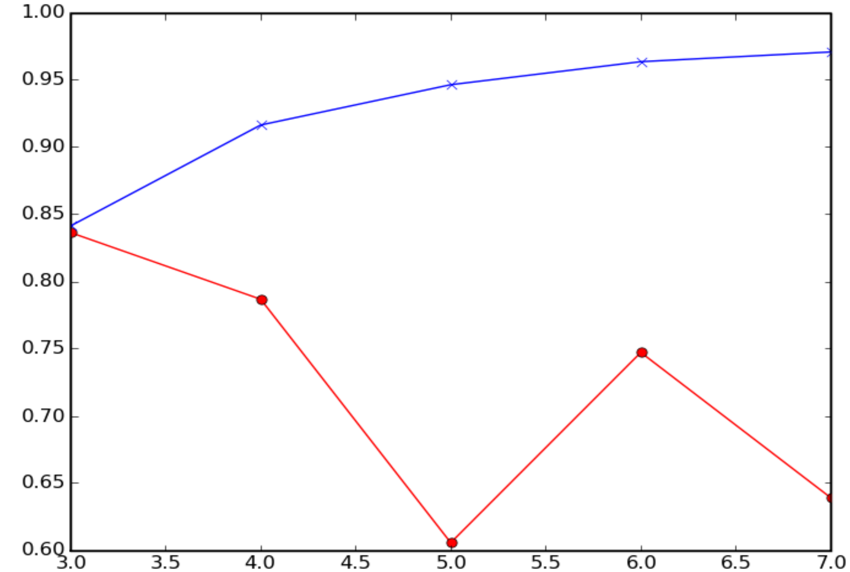
- This 'JSV' algorithm is *slow* – each step of the Markov chain takes constant time, but several logical & floating point operations – and on my laptop, any permanent feasibly computed by JSV can be found exactly and faster
- So, for varying exponents and constants, can calculate the root-mean-square error of JSV, and determine the values required for accuracy
- Need constants for the Markov chain, and for the sizes of samples taken at 3 separate places in the algorithm...

Markov chain constants

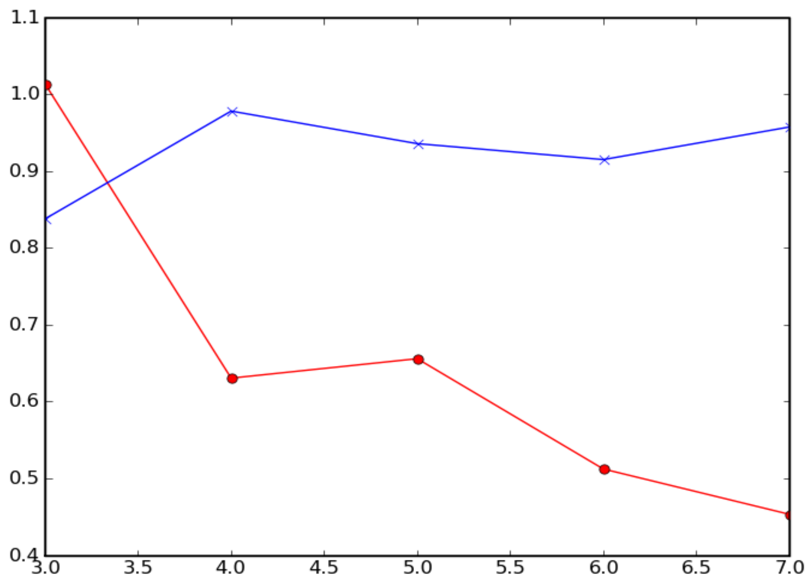
scaled RMSE (red) and correlation (blue) versus *n*, over 30 runs



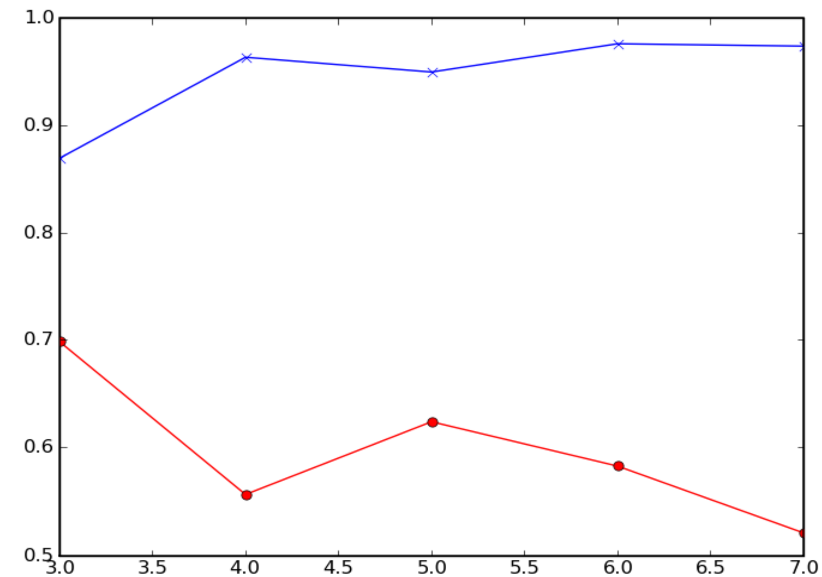
T = O(n)



T = O(n log n)



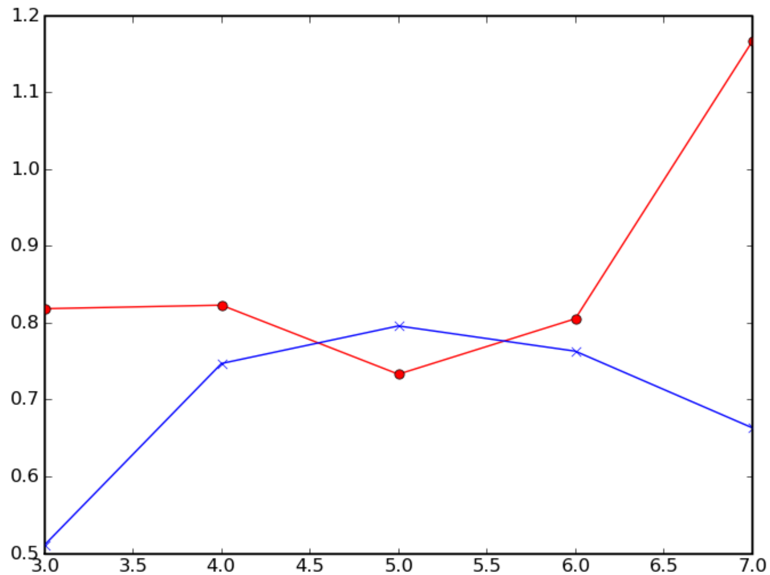
T = O(n^2)



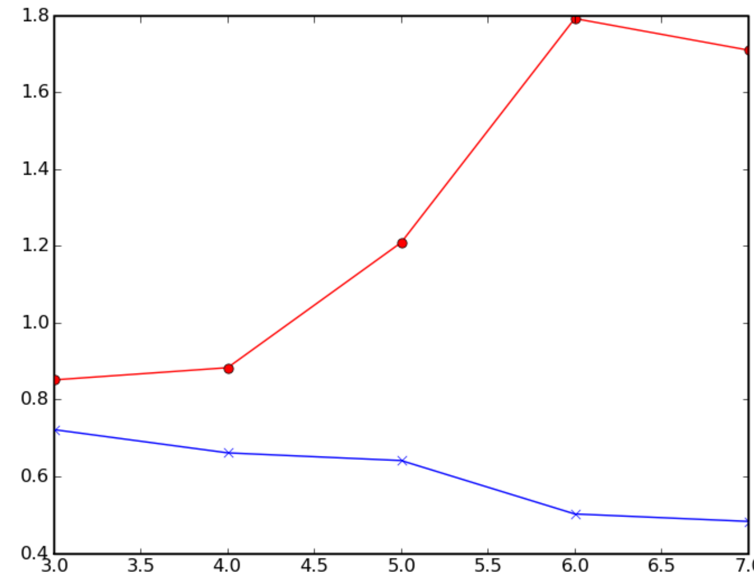
T = O(n^2 log n)

Simulated annealing sample constants

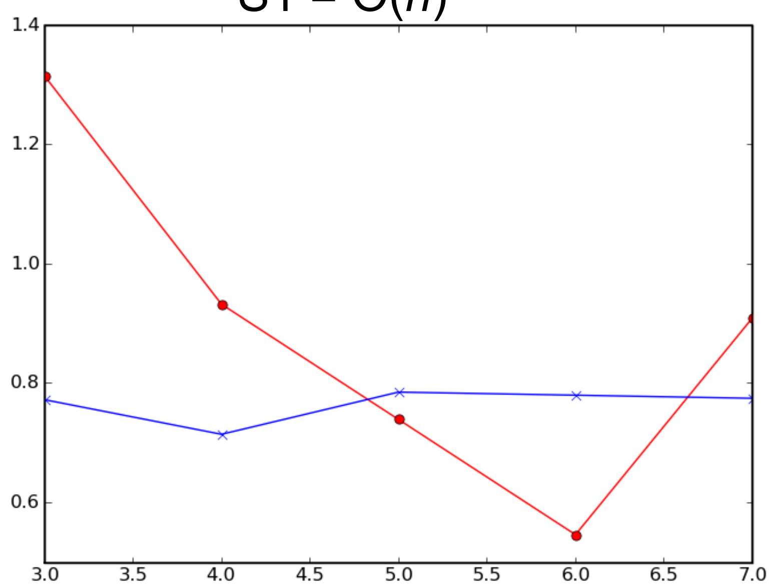
scaled RMSE (red) and correlation (blue) versus *n*, over 30 runs



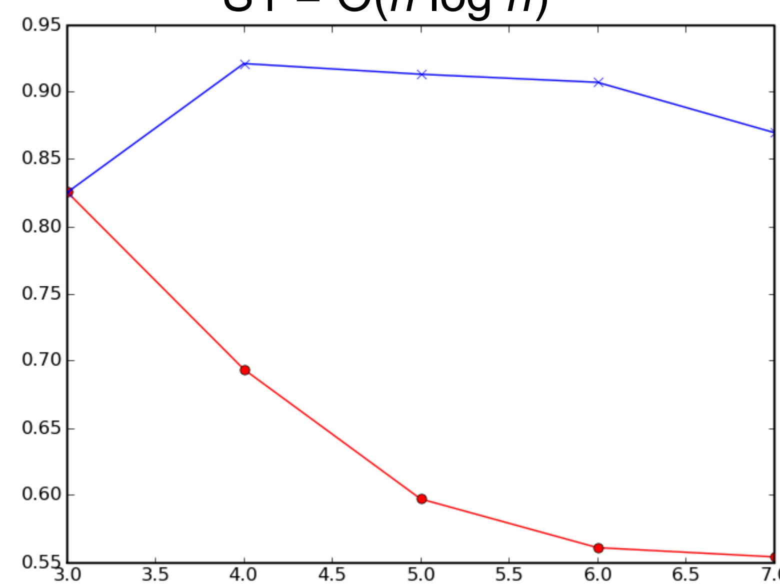
S1 = O(n)



S1 = O($n \log n$)



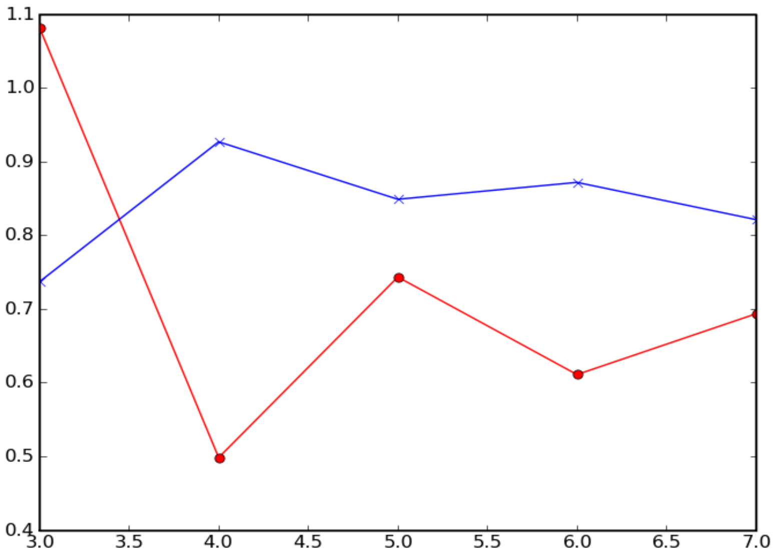
S1 = O(n^2)



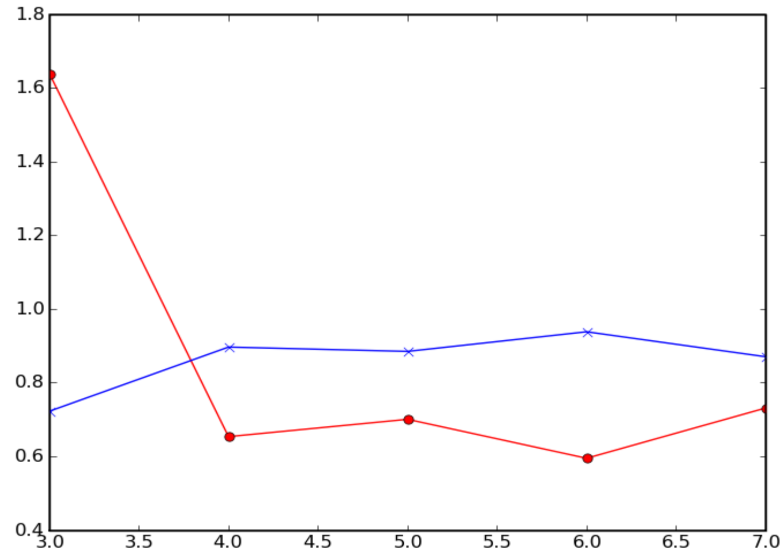
S1 = O($n^2 \log n$)

Product initialization sample constants

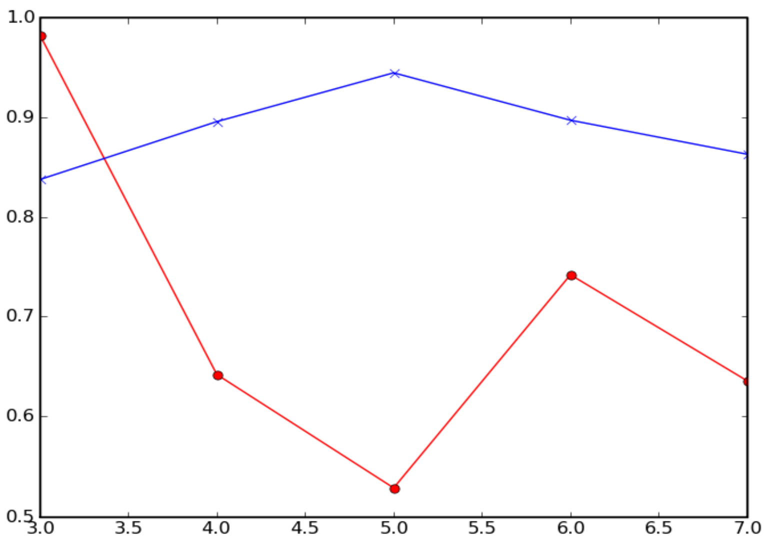
scaled RMSE (red) and correlation (blue) versus *n*, over 30 runs



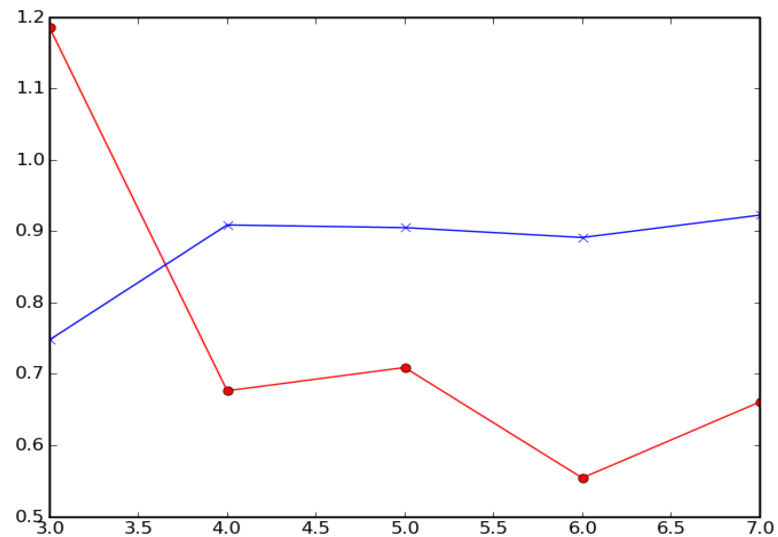
S2 = O(*n*)



S2 = O(*n* log *n*)



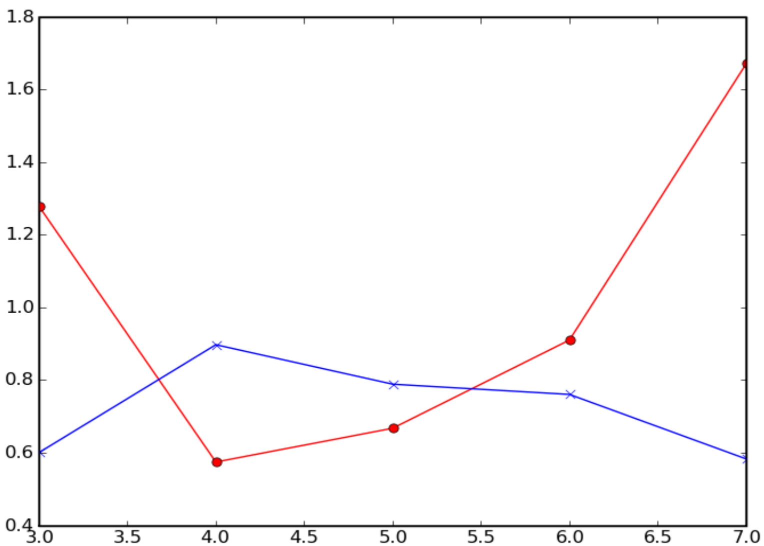
S2 = O(n^2)



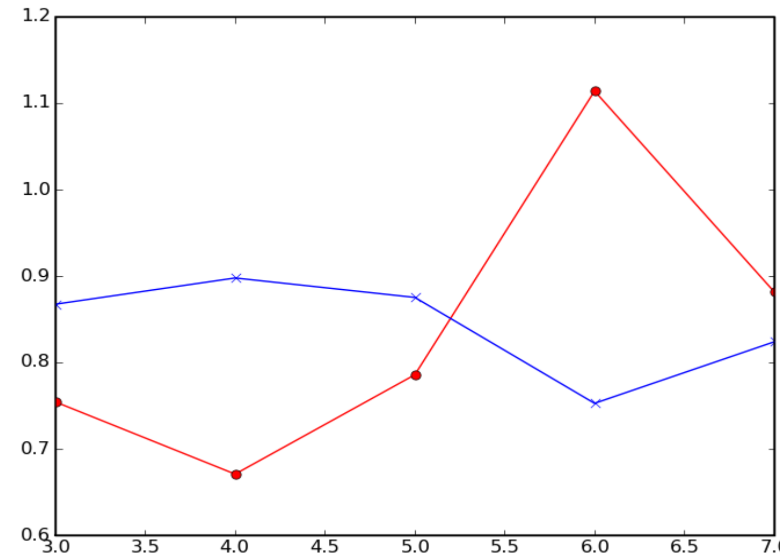
S2 = O(n^2 log *n*)

Product update sample constants

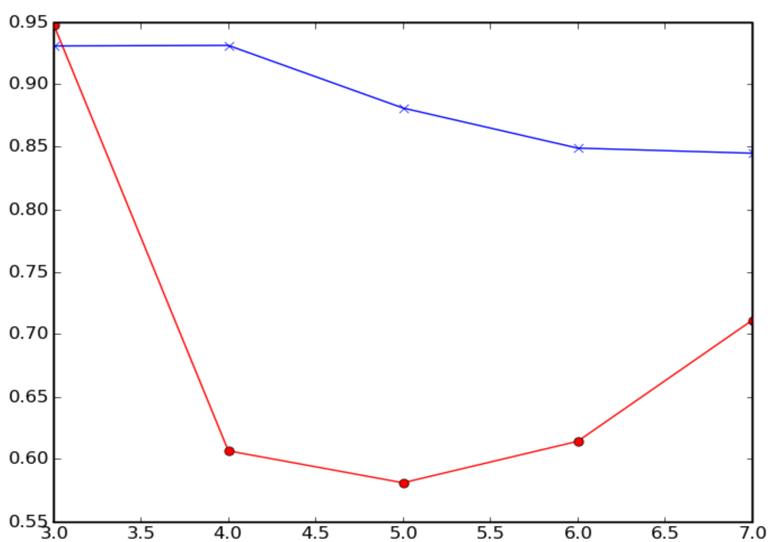
scaled RMSE (red) and correlation (blue) versus *n*, over 30 runs



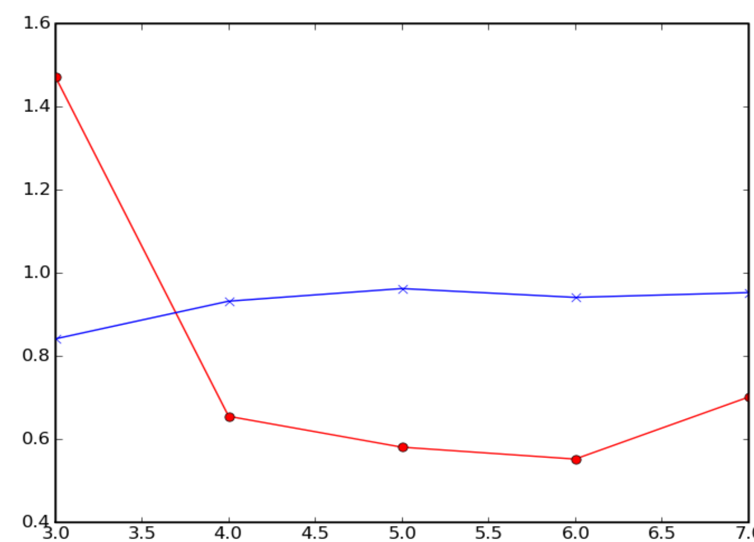
S3 = O(1)



S3 = O(log *n*)



S3 = O(*n*)



S3 = O(*n* log *n*)

Best estimated values

- Steps in the Markov chain: O (n^2)
- Matchings per sample during simulated annealing: O $(n^2 \log n)$
- Matchings per sample initializing the permanent as a product of ratios: O $(n^2 \log n)$
- Matchings per sample for each ratio in updating the permanent as a product: O $(n \log n)$
- Overall algorithm running time: O $(n^5 \log^3(n))$
- Sample sizes cannot be reduced, but Markov chain running time can