International Meeting on High-Dimensional Data-Driven Science (HD^3-2015) IOP Publishing
Journal of Physics: Conference Series **699** (2016) 012015 doi:10.1088/1742-6596/699/1/012015

Performance of a community detection algorithm based on semidefinite programming

Federico Ricci-Tersenghi
Dipartimento di Fisica, INFN–Sezione di Roma1 and CNR–Nanotec, Università La Sapienza, Piazzale Aldo Moro 5, I-00185 Roma, Italy.
E-mail: `federico.ricci@uniroma1.it`

Adel Javanmard
USC Marshall School of Business, University of Southern California

Andrea Montanari
Department of Electrical Engineering and Department of Statistics, Stanford University

Abstract. The problem of detecting communities in a graph is maybe one the most studied inference problems, given its simplicity and widespread diffusion among several disciplines. A very common benchmark for this problem is the stochastic block model or planted partition problem, where a phase transition takes place in the detection of the planted partition by changing the signal-to-noise ratio. Optimal algorithms for the detection exist which are based on spectral methods, but we show these are extremely sensible to slight modification in the generative model. Recently Javanmard, Montanari and Ricci-Tersenghi [1] have used statistical physics arguments, and numerical simulations to show that finding communities in the stochastic block model via semidefinite programming is quasi optimal. Further, the resulting semidefinite relaxation can be solved efficiently, and is very robust with respect to changes in the generative model. In this paper we study in detail several practical aspects of this new algorithm based on semidefinite programming for the detection of the planted partition. The algorithm turns out to be very fast, allowing the solution of problems with $O(10^5)$ variables in few second on a laptop computer.

1. Introduction and model definition

When dealing with a high-dimensional dataset one often looks for hidden structures, that may be representative of the signal one is trying to extract from the noisy dataset. In the community detection problem, we are asked to find the most significative clustering of the vertices of a graph, such that intra-cluster connections are much more abundant/sparse in the assortative/disassortative case with respect to inter-clusters connections (see Ref. [2] for a review). A common benchmark in this class is provided by the the so-called planted partition problem or stochastic block model (SBM) [3], defined as follows: an undirected graph $G = (V, E)$ is given, where $V = \{1, \dots, n\}$ is the vertex set and E is the edge set. Vertices are divided in q equal size groups V_k, with $k = 1, \dots, q$, such that $V = \cup_{k=1}^{q} V_k$, $|V_k| = n/q$ and $V_i \cap V_j = \emptyset$ if

$i \neq j$. The function $\kappa(i)$ returns the cluster vertex i belongs to. Conditioned on the partition $\{V_k\}_{k=1,\dots,q}$, edges are drawn independent with

$$\mathbb{P}\Big[(i,j) \in E \Big| \{V_k\}\Big] = \begin{cases} c_{\text{in}}/n & \text{if } \kappa(i) = \kappa(j)\,, \\ c_{\text{out}}/n & \text{if } \kappa(i) \neq \kappa(j)\,. \end{cases}$$

The resulting random graph is sparse and has a mean degree equal to $c = [c_{\text{in}} + (q-1)c_{\text{out}}]/q$. We will be mainly concerned with the assortative case, where $c_{\text{in}} > c_{\text{out}}$ holds strictly. The goal is to detect the partition $\{V_k\}$ given the graph G.

The Bayesian approach [4] predicts the existence of a threshold at

$$c_{\text{in}} - c_{\text{out}} = q\sqrt{c}\,,$$

for $q \leq 4$, separating a phase where detecting the partition is impossible from a phase where the detection can be achieved, using the best possible algorithms. We defined the signal-to-noise ratio as

$$\lambda = \frac{c_{\text{in}} - c_{\text{out}}}{q\sqrt{c}}\,,$$

such that the phase transition for $q \leq 4$ takes place at $\lambda_c = 1$. We will mainly be interested in the $q = 2$ case, where the existence of the threshold at $\lambda_c = 1$ has been proved rigorously [5, 6]. In this case the planted partition can be conveniently coded in a vector $\boldsymbol{x}_0 \in \{+1,-1\}^n$, and, calling $\hat{\boldsymbol{x}}(G)$ the estimate of the partition in graph G obtained by any inference procedure, we can quantify the success of the detection algorithm computing the overlap with respect to the planted partition (i.e. the absolute value of the normalized scalar product)

$$Q = \frac{1}{n}|\langle \hat{\boldsymbol{x}}(G), \boldsymbol{x}_0 \rangle|\,.$$

Eventually we can consider also its average value over the ensemble of random graphs, $\mathbb{E}[Q]$. For the SBM Ref. [7] proposed a belief-propagation message-passing algorithm to find the Bayes optimal estimator, which has indeed a non-zero overlap with the planted partition as soon as $\lambda > \lambda_c = 1$.

Spectral methods based on the Laplacian (unnormalized or normalized) are known to be sub-optimal, since they have a threshold which is strictly larger than the optimal one [8], i.e. $\lambda_c^{\text{Lap}} = \sqrt{c/(c-1)} > 1$. However, a new spectral method based on the non-backtracking matrix introduced in Ref. [9] achieves optimality in the detection of the planted partition in the SBM, at the cost of computing the complex spectrum of a non symmetric matrix. Later, such a spectral method has been strongly simplified by showing its similarity to the computation of the spectrum of the so-called Bethe Hessian matrix [10], which is a $n \times n$ symmetric matrix defined as

$$H(r) = (r^2 - 1)\mathbb{1} - rA - D\,,$$

where A is the adjacency matrix, $A_{ij} = A_{ji} = \mathbb{I}[(ij) \in E]$, and D is a diagonal matrix with entries equal to the vertex degrees d_i. In the assortative SBM with $q = 2$, the planted partition is detected by computing the negative eigenvalues of $H(\sqrt{c})$ [10] and the best estimator $\hat{\boldsymbol{x}}^{\text{BH}}(G)$ turns out to be given by the vector of signs of the components of the eigenvector corresponding to the second largest (in absolute value) eigenvalue.

2. Spectral methods versus optimization methods

Although the partition detection based on the Bethe Hessian is optimal for the SBM, it turns out to be not very robust if the generative model departs even slightly from the random graph

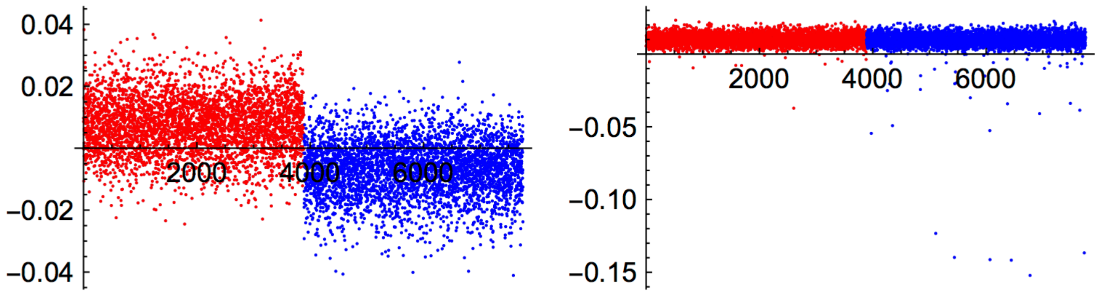

Figure 1. Components of the Bethe Hessian estimator for the planted partition in a graph of 7 755 vertices and 13 348 edges generated according to the SBM and then reduced to its 2-core (left). In the right plot we have used the same graph, where 2 cliques of sizes 3 and 5 has been added. A difference of only 13 edges not generated according to the SBM makes the spectral method completely useless for the detection.

ensemble defined above. Given that the underlying assumption in the SBM is that the graph is locally tree-like, the addition of few cliques may drastically deteriorate the performance of the algorithm based on the Bethe Hessian. Just to give you an idea of how fragile may be the Bethe Hessian based estimator $\hat{\boldsymbol{x}}^{\mathrm{BH}}(G)$ we show in Fig. 1 its components for 2 almost identical graphs. The graph in the left plot is a typical graph from the SBM ensemble with $q = 2$, $c = 3$, $\lambda = 1.2$ and $n = 10^4$ (actually we focus on the 2-core of the graph, for reasons explained below, that has 7 755 vertices and 13 348 edges in this case): the resulting overlap with the planted partition is $Q = 0.59$, in agreement with previous studies [10]. The graph in the right plot is exactly the same graph in the left plot with the addition of *only two cliques* of sizes 3 and 5 (that is just 13 more edges!), but the ability to infer the planted partition from the spectrum of the Bethe Hessian is completely lost, as a consequence of the eigenvector localization. Indeed the resulting overlap is $Q = 0.01$.

Robustness is an important requirement, since no generative model is exact in applications. To this aim, converting the inference problem to an optimization problem is welcome. Obviously the result will depend on the objective function to maximize. One of the most used objective functions for detecting communities in real world graphs is the modularity [11, 12] defined as

$$\sum_{g=1}^{q} \sum_{i,j\in g} \left(A_{ij} - \frac{d_i d_j}{2m} \right) ,$$

where the index g runs over the clusters of vertices in the partition. Modularity measures the excess of intra-cluster connections with respect to a random assignment. Maximizing the modularity is not an easy job, especially because one can find several phase transition in the space of partitions [13, 14], which are likely to affect the the maximum likelihood problem.

In the case of $q = 2$ groups of equal size, the problem of detecting the graph partition is equivalent to the minimum bisection problem, and the objective function to be minimized is just the cut size. Consequently the maximum-likelihood estimator is given by

$$\hat{\boldsymbol{x}}^{\mathrm{ML}}(G) = \operatorname*{argmax}_{\boldsymbol{x}\in\{+1,-1\}^n} \left(\sum_{(ij)\in E} x_i x_j \middle| \sum_i x_i = 0 \right) . \tag{1}$$

However computing the estimator in (1) is a hard problem (NP-complete and non polynomial time approximable in the worst case [15]), because the function to be maximized may have several local maxima, that trap the optimization algorithms.

In order to approximate this problem with a more tractable version it is customary to relax it as semidefinite programming (SDP) over the set of $n \times n$ real and symmetric matrices $\boldsymbol{C}$ [16]

$$\text{maximize} \sum_{i,j} A_{ij} C_{ij} \qquad \text{subject to } \boldsymbol{C} \succeq 0\,,\ C_{ii} = 1\,,\ \sum_j C_{ij} = 0\ \forall i\,. \tag{2}$$

The positive semidefinite condition, $\boldsymbol{C} \succeq 0$, requires all the eigenvalues of $\boldsymbol{C}$ to be non-negative and makes the feasible set convex, thus ensuring the existence of a tractable maximizer of (2).

The maximum-likelihood problem (1) is recovered from the formulation in (2) by enforcing the matrix $\boldsymbol{C}$ to be of rank 1: $C_{ij} = x_i x_j$. So, in general, willing to solve the non-convex problem in (2) with rank 1 matrices, one may search for a solution to the convex problem with generic rank n matrices, and then project back this solution to the space of rank 1 matrices.

A convenient way to represent a $n \times n$ real and symmetric positive semidefinite matrix of rank m is to consider it as a correlation matrix between n real vectors of m components each:

$$C_{ij} = \underline{x}_i \cdot \underline{x}_j\,, \quad \text{with } \underline{x}_i \in \mathbb{R}^m\,,\ \|\underline{x}_i\|^2 = \underline{x}_i \cdot \underline{x}_i = 1\,. \tag{3}$$

3. Our community detection algorithm and it performances

In order to solve problem (2) over the set of rank m matrices we have recently proposed the following procedure [1]. First of all search for a configuration of the n unit-length m-components vectors $\underline{x}_i \in \mathbb{R}^m$, $\|\underline{x}_i\| = 1$, optimizing the following objective function

$$\text{maximize} \sum_{(ij)\in E} \underline{x}_i \cdot \underline{x}_j\,, \qquad \text{subject to} \sum_i \underline{x}_i = \underline{0}\,. \tag{4}$$

Let us call $\underline{\boldsymbol{x}}^* = \{\underline{x}_1^*, \ldots, \underline{x}_n^*\}$ the maximizer. To project back the maximizer to a vector of n reals, we first compute the matrix $\Sigma \in \mathbb{R}^{m\times m}$ measuring the correlations among the m components of the maximizer averaged over the entire graph

$$\Sigma_{jk} = \frac{1}{n} \sum_{i=1}^{n} (\underline{x}_i^*)_j (\underline{x}_i^*)_k\,, \tag{5}$$

whose principal component we call $\underline{v}_1$, and then we project each $\underline{x}_i$ over $\underline{v}_1$. Thus the rank m SDP estimator $\hat{\boldsymbol{x}}^{\text{SDP}}(G)$ has components

$$\hat{x}_i^{\text{SDP}}(G) = \text{sign}(\underline{x}_i \cdot \underline{v}_1)\,. \tag{6}$$

As before we measure the success of our detection algorithm by computing the overlap with respect to the planted partition

$$Q^{\text{SDP}} = \frac{1}{n} |\langle \hat{\boldsymbol{x}}^{\text{SDP}}, \boldsymbol{x}_0 \rangle|\,.$$

In Ref. [1] we have shown that the above algorithm, in the $m \to \infty$ limit, is optimal for synchronization problems defined on dense graphs and almost optimal for solving the SBM. By 'almost optimal' we mean that the overlap Q^{SDP} shows a phase transition at λ_c^{SDP} slightly larger than the optimal $\lambda_c = 1$: for example for $c = 3$ we have $\lambda_c^{\text{SDP}} \simeq 1.017$, with $Q^{\text{SDP}} > 0$ for $\lambda > \lambda_c^{\text{SDP}}$. Just for comparison, we remind that for $c = 3$ the spectral methods based on the Laplacian are unable to detect the planted partition as long as $\lambda < \sqrt{3/2} \simeq 1.22$.

The algorithm we use to find the maximizer in (4) is nothing but a zero temperature dynamics for a model with m-component spins $\underline{x}_i$ placed on the vertices of the graph G, with the addition of an external field that self adapt in order to keep the global magnetization null. In practice at each step of the algorithm we update spins in a random order aligning each spin to its local field

$$\underline{x}_i^{(t+1)} = \frac{\sum_{j:(ij)\in E} \underline{x}_j^{(t)} - \underline{M}(t)}{\| \sum_{j:(ij)\in E} \underline{x}_j^{(t)} - \underline{M}(t)\|} ,$$

with the global magnetization being $\underline{M}(t) = \sum_i \underline{x}_i$. We always check that the global magnetization becomes very small at large times, $\lim_{t\to\infty} \underline{M}(t) = \underline{M}(\infty) \ll 1$. As a stopping criterion we check the largest variation in a spin during the last step, $\Delta_{\max} = \max_i \|\underline{x}_i^{(t+1)} - \underline{x}_i^{(t)}\|$, and we stop when $\Delta_{\max} < \varepsilon$, with $\varepsilon = 10^{-3}$ or 10^{-4} (the specific value is rather irrelevant, being the results independent of ε as long as $\varepsilon \lesssim 10^{-3}$). We call t_{conv} the number of steps required to meet the stopping criterion.

The main parameter in this algorithm is the number of components m, and we expect the behavior of the algorithm to depend strongly on m at least close to the critical point $\lambda_c = 1$. Indeed for $m = 1$ the maximization of the likelihood is a NP-hard problem. We expect a greedy dynamics, as the one we are using, to get stuck in some local maxima, while for $m \to \infty$ the problem is convex and thus the greedy dynamics should be able to reach the maximum.

Given that each step of the dynamics requires $O(m\,n)$ operations, an interesting aspect to study is the minimal value of m that allows the algorithm to get close to the $m = \infty$ maximizer, such that the quasi optimal solution of Ref. [1] can then be obtained through the projection in (6). On general grounds, since the SDP in (2) has n constraints, one can show [17] that for $m \geq 2\sqrt{n}$ the objective function in (4) has no local maxima, which are not global maxima. However, we expect $\sqrt{n}$ to be a loose upper bound for the optimal value of m on random instances. Indeed, for $\lambda = 0$ the model we are studying is close to an m-components spin glass on a random graph, whose ground state (i.e. the maximizer) has a number of non-zero components (i.e. the rank) growing roughly as n^{μ}, with $1/3 \lesssim \mu < 2/5$ depending on the mean degree c [18]. From the recent work [19] we known that the maximum of the objective function on rank m solutions deviates for the corresponding $m = \infty$ value by no more than $O(1/m)$ terms. We will see that actually the convergence on the SDP estimator $\hat{\boldsymbol{x}}^{\text{SDP}}$ is much faster.

In Ref. [1] we have studied the effect of changing m in solving the SBM close to the critical value $\lambda_c = 1$. The kind of results we got are illustrated in Fig. 2 for graphs generated according to the SBM with $c = 5$ and $\lambda = 1$. We plot the histogram of the convergence times, t_{conv}, and we see that running the algorithm with $m = 20$ and $m = 40$ leads essentially to the same dynamics and the same convergence times for $n = 1000$ and $n = 2000$. On the contrary for $n = 8000$ convergence times with $m = 20$ are sensibly larger and much more disperse than with $m = 40$ (notice we are plotting $\log(t_{\text{conv}})$). Our conclusion is that the objective function we are climbing with the greedy dynamics is less smooth for $m = 20$ than for $m = 40$.

Although the dynamics get slower (i.e. t_{conv} increases) reducing the value of m, because the objective function to be maximized gets rougher, each single step of the algorithm becomes much more economic, and so it is worth asking how much the maximizer reached by running the algorithm with a "too small" value of m is informative for the goal of detecting communities.

The graphs that we use are generated according to the SBM with $c = 3$, and subsequently they are pruned by eliminating recursively dangling ends. Equivalently we reduce the graph to its 2-core, and this has the advantage of reducing the noise level, without changing the complexity of the problem. Indeed any removed tree joins the core on a single vertex and the optimal solution is to assign all tree vertices to the same cluster the core vertex belongs to.

In order to study the robustness of our approach, we add cliques according to the following rule: for each vertex, with probability p, we add a clique among all its neighbour vertices. This superimposed structure is well justified since in many real world networks, especially social networks, structures of this kind are very common (two friends of a person tend to be also friends). But from the point of view of the SBM this is an additional perturbation which may

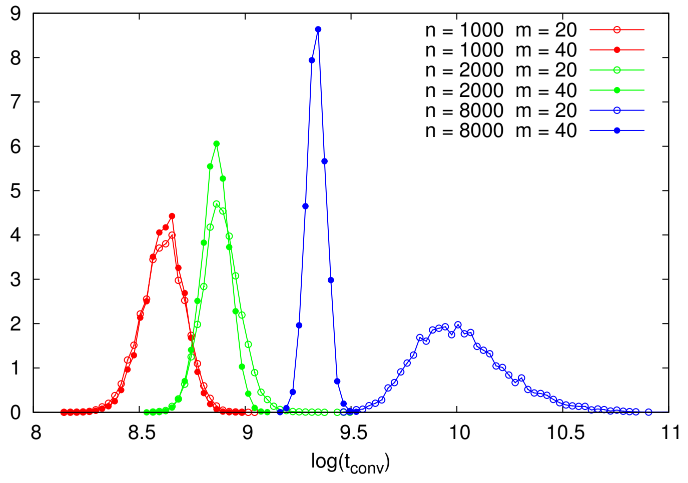

Figure 2. Distribution of convergence times for our greedy algorithm measured on many graphs generated according to the SBM with $c = 5$ and $\lambda = 1$. The typical running time grows mildly with the problem size n, but if m is not large enough the convergence time becomes much larger and sample to sample fluctuations much more severe.

degrade the performance of detecting algorithms. Indeed this is what we have shown preliminary in Fig. 1 where the graph of the right plot has been generated with a very small $p = 10^{-4}$ and still the effect on the Bethe Hessian algorithm is dramatic.

In order to get statistics on a single graph, we run the algorithm with 100 different *clones*. Each clone starts from a different random initial condition and evolve according to the greedy algorithm illustrated above. Remind that the algorithm, although greedy, is not deterministic, because variables are updated in a random order. At all the times when we take measurements we compute the SDP estimator $\hat{\boldsymbol{x}}^{\text{SDP}}$ and the corresponding overlap Q^{SDP} for each of the 100 clones.

In Fig. 3 we show the robustness of our algorithm with respect to the addition of cliques. We consider a graph generated initially with $n = 40\,000$ nodes at $\lambda = 1.1$: the resulting core has 30 597 vertices, 53 161 edges for $p = 0$ and 54 734 edges for $p = 0.01$. In Fig. 3 we plot Q^{SDP} as a function of the number of iteration of our algorithm, for $m = 16$ and $m = 64$. The horizontal lines report the value of the overlap q achieved by the Bethe Hessian algorithm, which is very close to the Bayes optimal value, according to Ref. [10]. We observe that for $p = 0$ our algorithm reaches a higher value of the overlap with respect to the Bethe Hessian result, both for $m = 16$ and $m = 64$. More importantly the addition of cliques ($p = 0.01$ case) deteriorates very little the performances of our algorithm, while the spectral method return a random guess with $Q \simeq 0$. By virtue of robustness of our algorithm, hereafter we are going to study only the $p = 0$ case.

In Fig. 4 we report the results for Q^{SDP} obtained on the same graph as in Fig. 3 with many more values of m. Error bars represent the standard deviation over the 100 clones. We observe that, while for $m = 1, 2$ the algorithm is unable to detect any significant partition, already for $m = 3$ the signal is substantial and eventually for $m \geq 8$ the algorithm achieves the optimal value for Q^{SDP}. Being the graph size quite large (30 597 vertices and 53 161 edges) is surprising that our algorithm works so nicely also for values of m as small as $m = 8$.

Let us focus on values $m \geq 8$ and let us compute the real running time, that is the wall clock time our algorithm takes to maximize the objective function. This time is not exactly proportional to $m\,n$ because all the operations among the m-components vectors (e.g. sums and scalar products) can be highly optimized by modern compilers, and so we expect the real

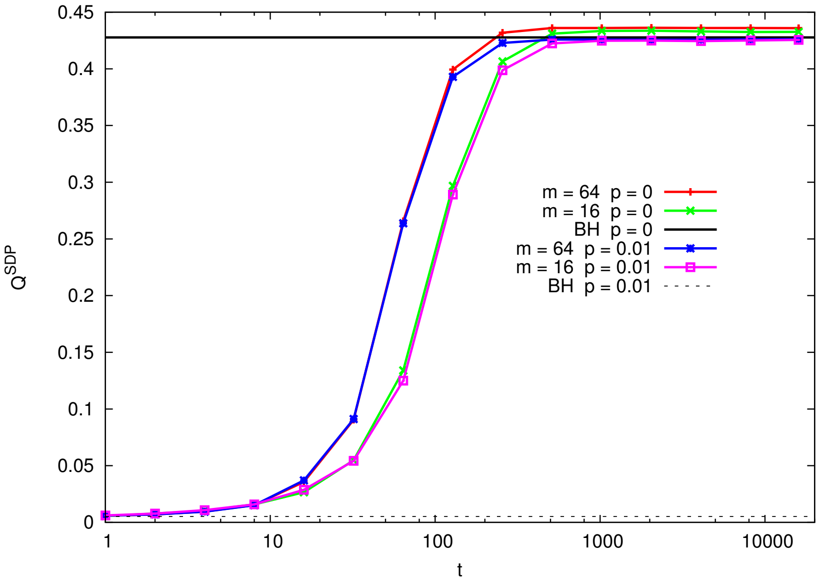

Figure 3. Overlap with the planted partition obtained from the configurations at time t in our algorithm. We report the average over the 100 clones. The comparison with the overlap achieved by the Bethe Hessian algorithm is favorable, especially when a small fraction of cliques is added, and the prediction from the Bethe Hessian is equivalent to a random guess. The algorithm run with $m = 64$ and $m = 16$ requires different running times, but achieves essentially the same inference accuracy.

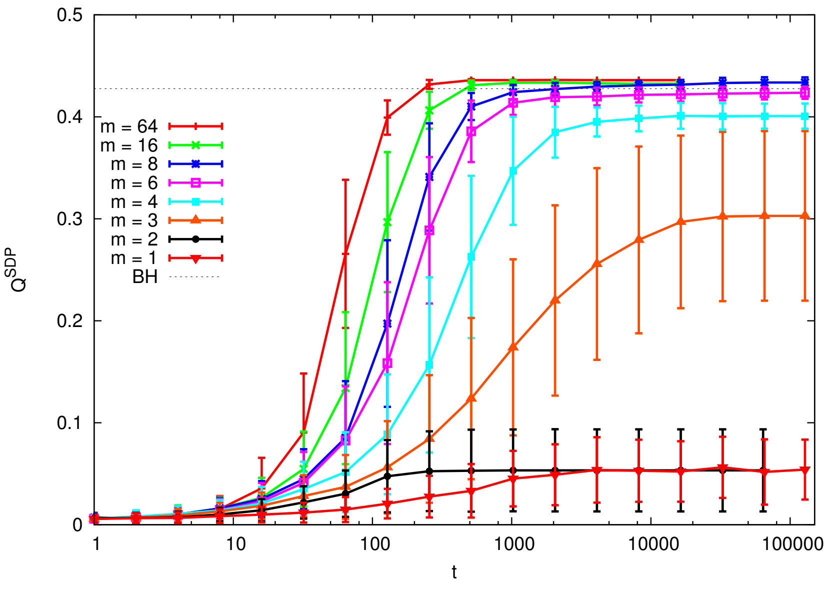

Figure 4. Dependence on m of the inferred overlap $Q^{\rm SDP}$ for $\lambda = 1.1$. Error bars represent standard deviation in the population of 100 clones.

running time to grow sublinearly with m. In Fig. 5 we report the growth of $Q^{\rm SDP}$ for 2 graphs of initial size $n = 10^5$ and $\lambda = 1.1$ (left) and $\lambda = 1.2$ (right); the sizes of their two cores are 77 250 nodes, 132 578 edges, and 77 370 nodes, 132 890 edges respectively. Error bars represent again the standard deviation over the 100 clones, while the CPU time has been measured for the

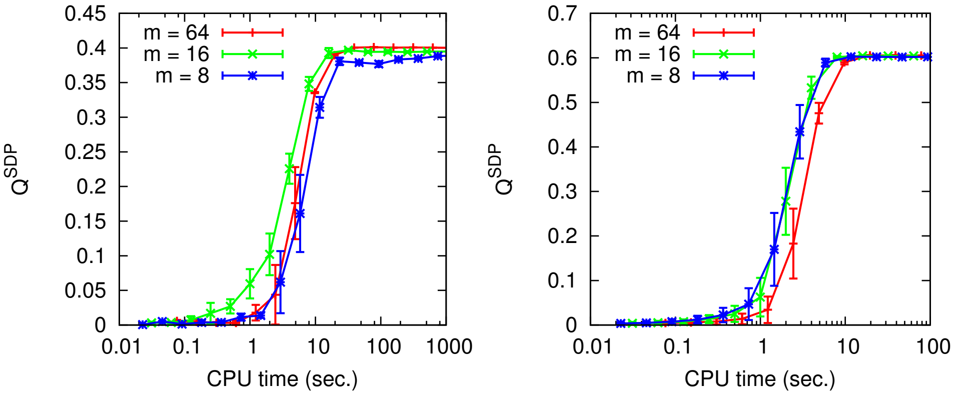

Figure 5. Inferred overlap by our SDP-based greedy algorithm as a function of the real wall clock time. The two problems have been generated according to the SBM with $n = 10^5$, $c = 3$ and $\lambda = 1.1$ (left) and $\lambda = 1.2$ (right). Times have been measured in seconds on a personal laptop with a 2 GHz Intel Core i7 processor. Dependence on m in running times is very weak.

simulation of a single clone on a personal laptop with a 2 GHz Intel Core i7 processor. Looking at Fig. 5 we notice that the differences in the real running time by varying the value of m are not very significant, and so maybe working with a moderate value of m is preferred, especially because the actual running time is very small: few seconds on a laptop to solve a problem with about 10^5 variables!

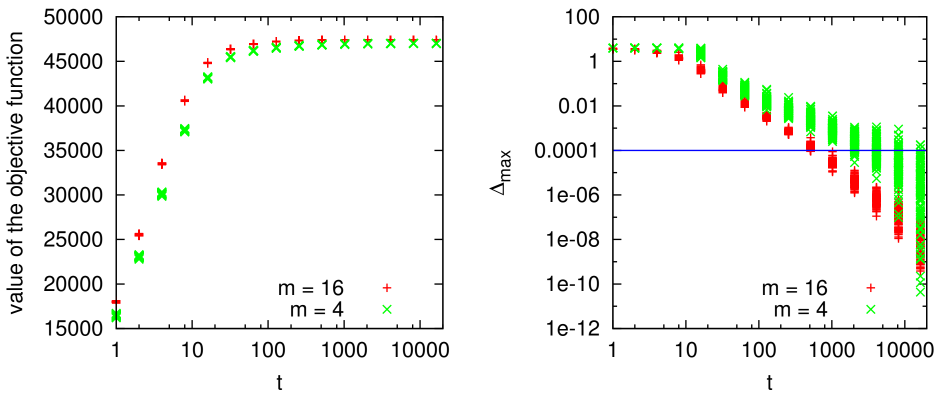

Figure 6. The objective function and the maximum variation in a variable during the last step, $\Delta_{\max}$, as a function of the number of algorithm steps. At each time we plot 100 points corresponding to different clones (although in the left panel they are hardly visible, because almost perfectly superimposed). The problem has size $n = 40\,000$ and $\lambda = 1.1$ (and is the same studied in Fig. 7).

In Figs. 3, 4 and 5 we have always considered the overlap with the planted partition; however in practice such an overlap is unknown. The only information available to us are the value of the objective function, $\Delta_{\max}$ and the distances between clones (if we run more than one clone). The first two quantities are shown in Fig. 6 for a graph generated with $n = 40\,000$ and $\lambda = 1.1$ (at each time we report the 100 measurements taken in different clones). From the previous analysis we known that $m = 16$ may be a good value, while $m = 4$ is definitely too small (see green and cyan lines in Fig. 4). However looking at the data in Fig. 6 we do not see large differences in the objective function and only the fluctuations in t_{conv}, given by the time where $\Delta_{\max} = 10^{-4}$

(blue horizontal line in the right panel), are suggesting $m = 4$ may be too small.

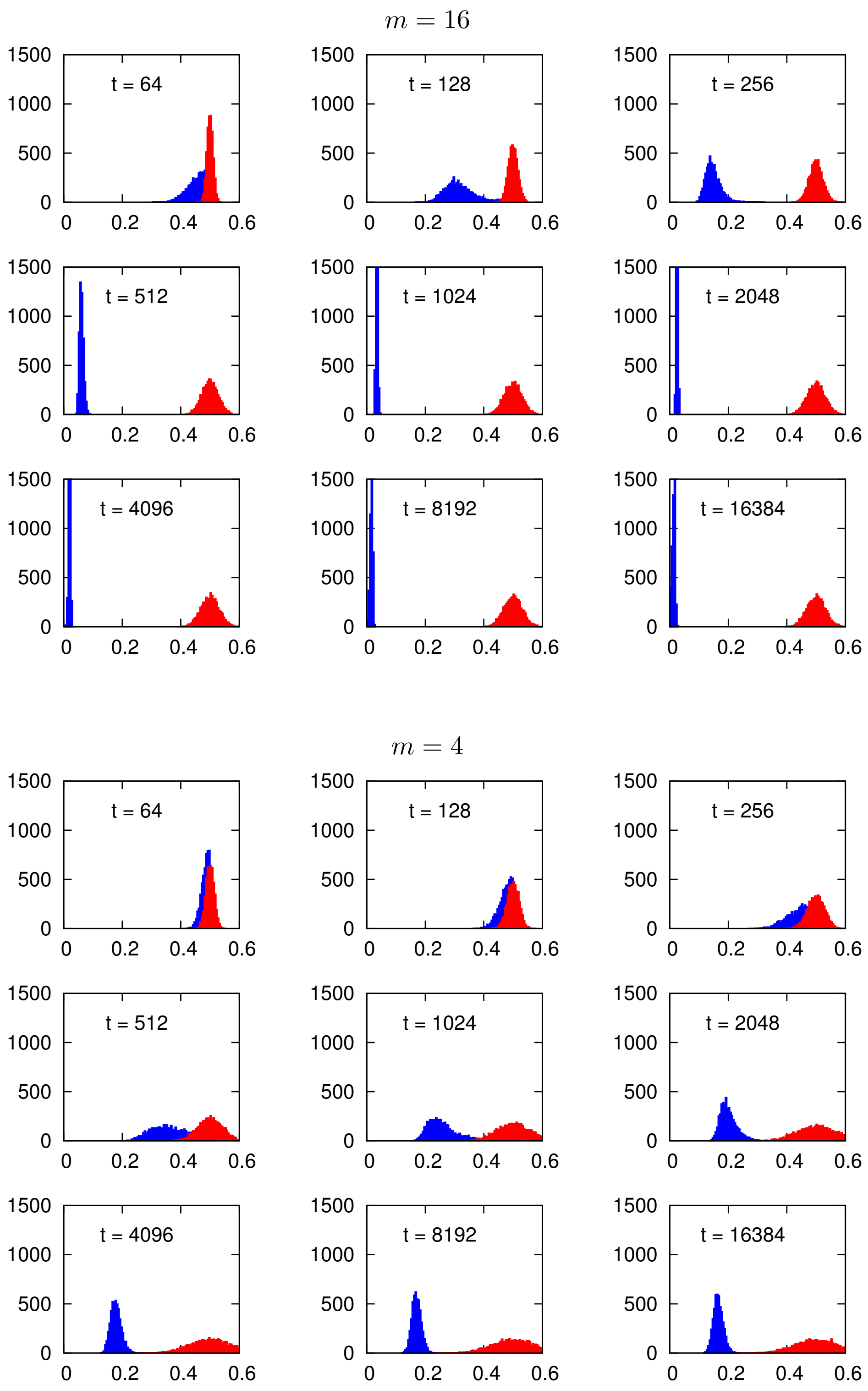

Figure 7. Histograms of the 4950 distances among the 100 clones before (red) and after (blue) the eliminating the rotational symmetry. Each panel is for a different algorithm time; upper panels are for $m = 16$ and lower ones are for $m = 4$.

Given that any clone eventually gets stuck in a maxima (either local or global), we would like to understand how likely the clones have reached the global maximum, without the need to run the algorithm with a larger value of m (that would make convergence to the global maximum easier). To this end we measure the distances between any pair of clones, by measuring

$$d = \frac{1}{2}\left(1 - \frac{1}{n}\sum_{i=1}^{n} \underline{x}_i^{(1)} \cdot \underline{x}_i^{(2)}\right) ,$$

where $\underline{\boldsymbol{x}}^{(1)}$ and $\underline{\boldsymbol{x}}^{(2)}$ are their configurations. The idea is that if all the clones have reached the global maximum they are all very close by, while if the objective function is too rough and they get stuck in local maxima, then their distances remain strictly positive. There is one caveat in computing straightforwardly the distance between the clones: since the objective function is invariant under a rotation $R : \mathbb{R}^m \to \mathbb{R}^m$ applied to all the n vectors and since the clones start from random configurations, we expect the clones to remain as orthogonal as possible, with a value of the distance $d \simeq 1/2$. In Fig. 7 we plot in red the histograms of the 4950 distances between the 100 clones, measured during the running of the algorithm with $m = 16$ (above) and $m = 4$ (below). Indeed, we see that the peak of the distributions remain close to $d = 1/2$ and provide no information at all about the real closeness of the clones. In order to get a meaningful information we should break the rotational invariance, by rotating the configuration of each clone and minimize their distances. We have done this in the following way: for each clone, we have computed the empirical correlation function Σ defined in (5), the corresponding principal component $\underline{v}_1$ and we have rotated the configuration such that $\underline{v}_1 \parallel (1, 0, \dots, 0)$ and all distances are smaller than 1/2. The resulting distances are shown in Fig. 7 with the blue histograms: it is now clear that running the algorithm with $m = 16$ makes all the clones converge to the same optimal configuration, while in the $m = 4$ case the clones get stuck in local maxima and their distances converge to strictly positive values.

4. Conclusions and perspectives

We studied how the algorithm based on SDP for community detection introduced in Ref. [1] works on very large graphs generated according to the SBM. We have shown the algorithm is very robust with respect to variations in the generative model: adding a good number of cliques only slightly degrades the algorithm performances (in contrast with spectral methods whose accuracy drops dramatically). The present work demonstrates that surprisingly small values of m may be sufficient for detecting the partition even in very large problems. Running few clones (even just 2 clones) allows one to understand whether the global maximum of the objective function is achieved. Consequently one can choose whether to rerun the algorithm with a larger value for m.

There are many possible extensions of our work. We just mention the most straightforward ones. We have made some preliminary runs of the algorithm with $q = 3$ and observed that 3 clusters of equal (or almost equal) size can be detected with a very good accuracy. However a systematic study of the performances of our algorithm for $q > 2$ is still missing. Given that for $q > 4$ the SBM undergoes a first order phase transition, it would be very interesting to study how the SDP relaxation in general and our algorithm in particular perform in detecting communities in that case. We think our algorithm can be also adapted to detect communities of different sizes by modifying the constraint of zero global magnetization; this modification is relevant to study real world problems.

Acknowledgments

A.J. and A.M. were partially supported by NSF grants CCF-1319979 and DMS-1106627 and the AFOSR grant FA9550-13-1-0036.

References

[1] Javanmard A, Montanari A and Ricci-Tersenghi F 2015 *Submitted*
[2] Fortunato S 2010 *Physics Reports* **486** 75–174
[3] Holland P W, Laskey K B and Leinhardt S 1983 *Social networks* **5** 109–137
[4] Decelle A, Krzakala F, Moore C and Zdeborová L 2011 *Physical Review E* **84** 066106
[5] Mossel E, Neeman J and Sly A 2013 *arXiv preprint arXiv:1311.4115*
[6] Massoulié L 2014 *Proceedings of the 46th Annual ACM Symposium on Theory of Computing* (ACM) 694–703
[7] Decelle A, Krzakala F, Moore C and Zdeborová L 2011 *Physical Review Letters* **107** 065701
[8] Kawamoto T and Kabashima Y 2015 *Phys. Rev. E* **91**(6) 062803
[9] Krzakala F, Moore C, Mossel E, Neeman J, Sly A, Zdeborová L and Zhang P 2013 *Proceedings of the National Academy of Sciences* **110** 20935–20940
[10] Saade A, Krzakala F and Zdeborová L 2014 *Advances in Neural Information Processing Systems* 406–414
[11] Clauset A, Newman M E and Moore C 2004 *Physical review E* **70** 066111
[12] Newman M E 2006 *Proceedings of the National Academy of Sciences* **103** 8577–8582
[13] Zhang P and Moore C 2014 *Proceedings of the National Academy of Sciences* **111** 18144–18149
[14] Schülke C and Ricci-Tersenghi F 2015 *Physical Review E* **92** 042804
[15] Arora S, Berger E, Hazan E, Kindler G and Safra M 2005 *Foundations of Computer Science, 2005. FOCS 2005. 46th Annual IEEE Symposium on* (IEEE) 206–215
[16] Goemans M X and Williamson D P 1995 *Journal of the ACM (JACM)* **42** 1115–1145
[17] Burer S and Monteiro R D 2003 *Mathematical Programming* **95** 329–357
[18] Braun A and Aspelmeier T 2006 *Physical Review B* **74** 144205
[19] Montanari A and Sen S 2015 *arXiv preprint arXiv:1504.05910*