Combinatorics, Probability and Computing (2012) 21, 643–660.
doi:10.1017/S0963548312000193

Are Stable Instances Easy?

YONATAN BILU[1†] and NATHAN LINIAL[2]

[1]Mobileye Vision Technologies Ltd, 13 Hartom Street, PO Box 45157, Jerusalem, 91450 Israel
(e-mail: yonatan.bilu@mobileye.com)
[2]Institute of Computer Science, Hebrew University, Jerusalem 91904, Israel
(e-mail: nati@cs.huji.ac.il)

Received 28 June 2009; revised 28 March 2012

We introduce the notion of a stable instance for a discrete optimization problem, and argue that in many practical situations only sufficiently stable instances are of interest. The question then arises whether stable instances of NP-hard problems are easier to solve, and in particular, whether there exist algorithms that solve in polynomial time all sufficiently stable instances of some NP-hard problem. The paper focuses on the Max-Cut problem, for which we show that this is indeed the case.

AMS 2010 *Mathematics subject classification*: Primary 68Q17
Secondary 05C50

1. Introduction

Computational complexity theory as we know it today is concerned mostly with worst-case analysis of computational problems. For example, we say that a problem is NP-hard if the existence of an algorithm that correctly decides *every* instance of the problem implies that SAT can be decided in a polynomially equivalent time complexity. However, the study of decision and optimization problems is motivated not merely by theoretical considerations. Much of our interest in such problems arises because they formalize certain real-world tasks. From this perspective, we are not interested in *all* problem instances, but only in those which can actually occur in reality.

This is often the case with clustering problems, which are ubiquitous in most fields of engineering, experimental and applied science. Any concrete formulation of the clustering problem is likely to be NP-hard. However, this does not preclude the possibility that the problem can be solved efficiently in practice. In fact, in numerous application areas, large-scale clustering problems are solved on a regular basis. Moreover, we are usually

† This research is supported by grants from the Israel–US Binational Science Foundation and the Israel Science Foundation.

only interested in instances where the data are actually made up of fairly well-defined clusters – the instances where solving the problem is interesting from the practical perspective.

Put differently, the usual way for proving that clustering is NP-hard is by a reduction to, say, SAT. This reduction entails the construction of instances for the clustering problem, such that the existence of an algorithm that can solve all of them efficiently implies the existence of an algorithm that efficiently solves SAT. However, it may well be the case that all these instances are clearly artificial, and solving them is of no practical interest.

As a concrete example, consider the problem of clustering protein sequences into families. Out of the enormous space of all possible sequences, only a tiny fraction is encountered in nature, and it is only these (or slight modifications thereof) that we actually care about.

Our case in point is the Max-Cut problem, which can be thought of as a clustering into two clusters. It is well known that this problem is NP-complete, and so it is believed that there is no algorithm that solves it on *all* graphs, in polynomial time. In this work we strive to identify properties of instances of the Max-Cut problem (*i.e.*, of weighted graphs), which capture the notion that the input has a well-defined structure with respect to Max-Cut (*i.e.*, the maximal cut 'stands out' among all possible cuts). Our goal is to show that Max-Cut can be solved efficiently on inputs that have such properties.

Considerations in a similar spirit have led to the development of *smoothed analysis*, initiated in [19] (see [20] for some of the exciting developments in that area). The similarity has two main facets: (i) both lines of research attempt to investigate the computational complexity of problems from a non-worst-case perspective, (ii) both are investigations of the *geometry* of the instance space of the problem under consideration. The goal is to discover interesting parts of this space in which the instances have complexity lower than the worst case. Viewed from this geometric perspective, the set-up that we study here is very different from what is done in the theory of smoothed analysis. There one shows that the hard instances form a discrete and isolated subset of the input space. Consequently, for every instance of the problem, a small random perturbation is very likely to have low computational complexity. In the problems that we study here the situation is radically different. The 'interesting' instances (*stable* instances, as we shall call them) are very rare. Indeed, it is not hard to show that under reasonable models of random instances, the probability that a random instance is stable tends to zero as the problem size grows. What we wish to accomplish is to efficiently solve *all* instances within this subspace. We claim that this tiny set is interesting because it includes all realistic clustering problems.

Balcan, Blum and Gupta [2] say that an instance of a problem has the (c, ϵ)-property if every c-approximation of the optimal solution is ϵ-close to it. They show that a close-to-optimal solution can be found efficiently for such instances. (Here two solutions are 'close' if they differ by at most an ϵ fraction of the vertices.)

Ostrovsky, Rabani, Schulman and Swamy [16] investigate k-clustering problems with the property that the 'correct' number of clusters is indeed k: the optimal value for k clusters is at most ϵ that attained for $k-1$ clusters (it is a minimization problem). They give a polynomial-time approximation scheme for such instances (see also [1]).

These two papers can be seen as having motivation and goals similar to ours: to view certain instances as 'interesting', and show that clustering can be efficiently found – or at least approximated – for such instances.

An optimization problem is specified by a function $f : \mathcal{I}_n \times \mathcal{P}_n \to \mathbb{R}$, where $\mathcal{I}_n$ is the collection of all instances of *size n* (in our case all $n \times n$ symmetric non-negative matrices). Also, $\mathcal{P}_n$ is the set of all possible solutions (here, partitions of $[n]$ into two parts), and $f(I, P)$ is the value of solution P to instance I (here, the weight of the cut induced by P for input matrix I). The optimization problem seeks to maximize $f(I, P)$ over $P \in \mathcal{P}_n$ for a given $I \in \mathcal{I}_n$. We investigate the properties of f for the Max-Cut problem, whereas the other papers mentioned above consider the properties of $f(I^*, P)$ for a *given* instance $I^* \in \mathcal{I}_n$.

The notion of *stability* is central to our work. This is a concrete way to formalize the intuition that the only instances of interest are those for which small perturbations in the data (which may reflect some measurement errors, for example) do not change the optimal partition of the graph.

Definition. Let W be an $n \times n$ symmetric, non-negative matrix. A γ-*perturbation* of W, for $\gamma \geqslant 1$, is an $n \times n$ matrix W' such that $\forall i, j = 1, \ldots, n$, $W_{i,j} \leqslant W'_{i,j} \leqslant \gamma \cdot W_{i,j}$.

Let $(S, [n]\backslash S)$ be a maximal cut of W, *i.e.*, a partition that maximizes $\sum_{i \in S, j \notin S} W_{i,j}$. The instance W (of the Max-Cut problem) is said to be γ-stable if, for every γ-perturbation W' of W, $(S, [n]\backslash S)$ is the unique maximal cut of W'.

However, perhaps this definition is not sufficient. Consider two bipartite graphs which are joined together by a single edge. The resulting graph *is* γ-stable for all γ, but the alignment of the two bipartite graphs with respect to one another completely depends on the adjoining edge. Hence, to better capture our intuition of what it means for a solution to be stable, it is reasonable to demand that in addition to stability the graph contains no small cuts. We show that the combination of both these properties indeed allows efficient solution of Max-Cut (Example 4.2).

In Section 3 we present an algorithm that solves in polynomial time γ-stable instances of Max-Cut: (i) on simple graphs of minimal degree δ, when $\gamma > \frac{2n}{\delta}$, and (ii) on weighted graphs of maximal degree Δ when $\gamma > \sqrt{\Delta n}$. In Section 4 we explore several spectral conditions which make Max-Cut amenable on stable instances. This involves analysing the *spectral partitioning* heuristic for Max-Cut. In particular, we show that Max-Cut can be solved efficiently on (locally) stable graphs. In Section 5.1 we show how to deduce an improved approximation bound for the Goemans–Williamson algorithm on stable instances, and that Max-Cut is easy in a certain random model for such instances.

Finally, while we claim that all interesting clustering instances are 'stable' (in an intuitive sense), the formalization of stability itself suggested here, and the results obtained, are only a first step in that direction. In particular, it is of great interest to study more permissive notions of stability, where a small perturbation can slightly modify the optimal solution. There are also other natural ways to capture the concept of stability. Similar considerations can be applied to many other optimization problems (some are discussed in [3]). These possibilities are mostly left for future investigations.

2. Preliminaries

2.1. Notation

Throughout the paper we denote the vertex set of the graph G under discussion by $[n]$. A vector $v \in \mathbb{R}^n$ induces the partition of $[n]$ into the sets $(\{i : v_i > 0\}, \{i : v_i \leqslant 0\})$. Viewing it as a partition of G's vertex set, we call it the *cut induced by* v in G.

The *indicator vector* of a partition $(S, \bar{S})$ of $[n]$ (or a cut in G) is the vector $v \in \{-1, 1\}^n$, with $v_i = 1$ if and only if $i \in S$.

For a weighted graph G, we denote the indicator vector of its maximal cut by mc^*. We generally assume that this cut is unique, otherwise mc^* is an indicator of some maximal cut.

For a subset $A \subset [n]$, we denote $\bar{A} = [n] \backslash A$.

For two disjoint subsets of vertices in the graph, A, B, we denote by $E(A, B)$ the set of edges going between them, and $w(A, B) = \sum_{(i,j) \in E(A,B)} W_{i,j}$. With a slight abuse of notation, we denote $w(i) = \sum_j W_{i,j}$. Finally, for a set of edges $F \subset E$, denote $w(F) = \sum_{(i,j) \in F} W_{i,j}$.

We switch freely between talking about the graph and about its associated weight matrix. Given a symmetric non-negative $n \times n$ matrix W with zero diagonal (as input to the Max-Cut problem), we define its *support* as a graph $G = (V, E)$ with vertex set $V = [n]$ where $(i, j) \in E$ if and only if $w_{ij} > 0$.

2.2. Properties and equivalent definitions

A useful way to think of γ-stability is as a game between two (computationally unbounded) players, Measure and Noise. Given a graph G, Measure chooses a cut $(S, \bar{S})$. Noise then multiplies weights of his choice by factors between 1 and γ, obtaining a graph G' (over the same vertex and edge sets, but with possibly different weights). He then chooses a different cut, $(T, \bar{T})$. Noise wins if in G' $w(T, \bar{T}) > w(S, \bar{S})$. Otherwise, Measure wins. A graph is γ-stable if Measure has a winning strategy.

Observe that the players' strategy is clear: Measure chooses the maximal cut, and Noise, without loss of generality, multiplies by γ the weights of the edges in $E(T, \bar{T}) \backslash E(S, \bar{S})$. Multiplying weights of other edges either does not change $w(T, \bar{T}) - w(S, \bar{S})$, or decreases it. Hence, we arrive at an equivalent definition for γ-stability.

Proposition 2.1. *Let $\gamma \geqslant 1$. A weighted graph G graph with maximal cut $(S, \bar{S})$ is γ-stable (with respect to Max-Cut) if, for every vertex set $T \neq S, \bar{S}$,*

$$w(E(S, \bar{S}) \backslash E(T, \bar{T})) > \gamma \cdot w(E(T, \bar{T}) \backslash E(S, \bar{S})).$$

This view of stability suggests how γ-stable graphs can be generated. Let G' be a γ'-stable graph. Multiplying the weights of all the edges in the maximum cut by $\frac{\gamma}{\gamma'}$ yields a γ-stable graph G. Moreover, it is not hard to see that all γ-stable (weighted) graphs can be obtained this way, in the following sense. For every γ-stable graph, dividing all edges in the maximum cut by γ yields an underlying graph with the same maximum cut partition.

One pleasing aspect of γ-stability is that it is *oblivious to scale*: multiplying all weights in a graph by a constant factor does not change its stability. This can be readily seen

from Proposition 2.1. It may seem natural to define γ two-way stability as robustness to perturbation by a multiplicative factor between $\frac{1}{\gamma}$ and γ (so-called two-way perturbation). But obliviousness to scale easily implies that a graph is γ two-way stable if and only if it is γ^2-stable.

It is also natural to consider a solution as 'interesting' if it stands out among all the alternatives. Let $(S, \bar{S})$ be a maximal cut in a graph G, and consider an alternative cut, $(T, \bar{T})$. Consider the set $E(S, \bar{S}) \Delta E(T, \bar{T})$ of those edges on which the two cuts 'disagree'. We seek to measure the difference between the cuts $(S, \bar{S})$ and $(T, \bar{T})$ relative to the size of $w(E(S, \bar{S}) \Delta E(T, \bar{T}))$. So say that $(S, \bar{S})$ is α *edge distinct* (with $\alpha > 0$) if, for any $T \subset V$,

$$w(S, \bar{S}) - w(T, \bar{T}) > \alpha \cdot w(E(S, \bar{S}) \Delta E(T, \bar{T})).$$

Now, denote $W_T = w(E(T, \bar{T}) \backslash E(S, \bar{S}))$ and $W_S = w(E(S, \bar{S}) \backslash E(T, \bar{T}))$. If G is α edge distinct then

$$\begin{aligned} W_S - W_T &= w(S, \bar{S}) - w(T, \bar{T}) \\ &> \alpha \cdot w(E(S, \bar{S}) \Delta E(T, \bar{T})) \\ &= \alpha \cdot (W_S + W_T). \end{aligned}$$

Hence, $W_S \geqslant \frac{1+\alpha}{1-\alpha} W_T$, and by Proposition 2.1 G is $\frac{1+\alpha}{1-\alpha}$-stable. Similarly, if G is γ-stable, then it is $\frac{\gamma-1}{\gamma+1}$ edge distinct.

We do not discuss distinctness directly here: the interested reader is referred to [3].

2.3. Variations on a theme

We shall also be interested in a weaker version of stability, which will prove useful for some of the results below.

Definition. Let W be an instance of the Max-Cut problem and let $(S, \bar{S})$ be its optimal partition. We say that W is γ-*locally stable* if for all $v \in S$

$$\gamma \cdot \sum_{u \in S} W_{u,v} < \sum_{u \in \bar{S}} W_{u,v},$$

and for all $v \in \bar{S}$

$$\gamma \cdot \sum_{u \in \bar{S}} W_{u,v} < \sum_{u \in S} W_{u,v}.$$

Observe that every γ-stable graph is also γ-locally stable. This follows from Proposition 2.1, with T differing from S at a single vertex (v).

Note also that the definition depends on the cut $(S, \bar{S})$, and so one can talk about locally stable cuts. That is, a locally stable instance is one where the Max-Cut is locally stable.

It is essentially known that Max-Cut is NP-hard even when restricted to γ-locally stable instances (for γ at most exponential in the size of the input) [17].[1] In fact, one can impose local stability without altering the overall stability. Let G be a graph with weighted adjacency matrix W. Let D be a diagonal matrix with $D_{i,i} = w(i)$. Let $G^\times$ be a graph on $V \times \{0,1\}$, with weighted adjacency matrix

$$G^\times = \begin{pmatrix} W & \tau D \\ \tau D & W \end{pmatrix},$$

for some $\tau \geqslant 1$.

It is not hard to see that the maximal cut in $G^\times$ consists of two copies of that in G. Specifically, $(S, \bar{S})$ is a maximal cut in G if and only if $(S \times \{0\} \cup \bar{S} \times \{1\}, S \times \{1\} \cup \bar{S} \times \{0\})$ is a maximal cut in $G^\times$. It is also not hard to see that G is γ-stable, if and only if $G^\times$ is, and that $G^\times$ is at least 2τ-locally stable.

Finally, the definition of stability arguably falls short of our intuitive concept of a stable instance. Consider a graph composed of two complete bipartite graphs, with a single edge connecting one component to the other. This graph is bipartite, and hence infinitely stable, yet this infinite stability hinges on this one edge.

Hence we would like to consider the following variation, where we ask that the optimal partitioning is stable not only with respect to multiplicative perturbation, but also an additive one. For this to make sense, the additive perturbation should not change the total weight of edges occurring at a vertex by too much (otherwise such stability is not possible: one could add a heavy edge between two vertices on the same side, forcing them apart).

Definition. An instance G of Max-Cut, with (weighted) adjacency matrix A, is (γ, ϵ)-stable (resp. locally stable) if:

(1) it is γ-stable (locally stable),
(2) the Max-Cut of A and $A + B$ is the same, when $\forall i, j,\ B_{i,j} \geqslant 0$ and $\forall i,\ \sum_j B_{i,j} \leqslant 2\epsilon \sum_j A_{i,j}$.

3. Combinatorial approach

One approach to solving a Max-Cut problem is to identify a pair of vertices which must to be on the same side of the optimal cut (*e.g.*, in a simple graph, two vertices with the same neighbourhood). Two such vertices can be safely merged into a single vertex, keeping multiple edges. If this can be repeated until a bipartite graph is obtained, then the problem is solved.

[1] The NP-completeness of Max-Cut can be shown by a reduction from 3-not-all-equal SAT. Construct a graph over the formula's literals, and for every 3-clause define three edges (a triangle) connecting the clause's literals. It is not hard to see that the formula is satisfiable if and only if the graph's Max-Cut value is twice the number of clauses. It is also not hard to see that if this is indeed the case, the cut is 2-locally stable. Furthermore, by adding edges between a literal and its negation, the structure of the Max-Cut does not change, and local stability increases.

Observe that if G is a γ-stable graph, and i, j are two vertices on the same side of the maximal cut, then the graph G', obtained from G by merging i and j into a single vertex i', is γ-stable as well. Indeed, any γ-perturbation of G' induces a γ-perturbation of G over the same edges. If as a result of this perturbation the maximal cut changes in G', then this new cut is also maximal in the similarly perturbed G, since it contains the same edges (in contradiction to G being γ-stable).

This observation implicitly guides the first algorithm presented below. In it we identify pairs of vertices which are on opposite sides of the maximal cut. By continuing to do so, we grow ever larger connected bipartite subgraphs until they all connect. In the second algorithm we explicitly merge together vertices on the same side for as long as we know how to, and then, once we have a much smaller graph, use the first algorithm.

3.1. An efficient algorithm for *n*-stable instances

We start by describing an algorithm that solves the Max-Cut problem on (weighted) graphs of maximal degree Δ which are $\sqrt{\Delta n}$-stable. The idea is to iteratively identify sets of edges which belong to the maximal cut. When they form a connected spanning bipartite graph, the maximal cut is found.

FindMaxCut(G) (G is a weighted graph)

1 Initialize $T = (V(G), \emptyset)$. Throughout the algorithm T will be a bipartite subgraph of G.

2 While T is not connected, do:

(a) Let $C_1, \ldots, C_t$ be the connected components of T. Each of them is a bipartite graph, with vertex bipartition $V(C_i) = (L_i, R_i)$.

(b) Let C_{i^*} be a component with the least number of vertices. For each $j = 1, \ldots, t$, $j \neq i^*$, let $E_j^0 = E(L_{i^*}, L_j) \cup E(R_{i^*}, R_j)$ and $E_j^1 = E(L_{i^*}, R_j) \cup E(R_{i^*}, L_j)$. Let j^* and c^* be such that the weight of $E_{j^*}^{c^*}$ is the largest among all E_j^c.

(c) Add the edges of $E_{j^*}^{c^*}$ to T.

3 Output the cut defined by the two sides of T.

Theorem 3.1. *The algorithm above runs in polynomial time and solves the Max-Cut problem for every γ-stable instance, provided that $\gamma > \min\{\sqrt{\Delta n}, \frac{n}{2}\}$. Here an instance is an n-vertex graph of maximal degree Δ.*

Proof. Let $(S, \bar{S})$ be the maximal cut. We maintain that throughout the algorithm, S *separates* each connected component $C_i = (L_i, R_i)$. Namely, either $L_i \subset S$, $R_i \subset \bar{S}$ or $R_i \subset S$, $L_i \subset \bar{S}$.

This clearly holds at the outset. If it holds at termination, the algorithm works correctly. So consider the first iteration when this does not hold. Let C_{i^*} be a smallest connected component at this stage, and denote $k = |C_{i^*}|$. Up to this point our assumption holds, so say $L_{i^*} \subset S$ and $R_{i^*} \cap S = \emptyset$. Let j^* and c^* be those chosen as in step 2(b). Since this is the point where the algorithm errs, $E_{j^*}^{c^*}$ is added to T, yet $E_{j^*}^{c^*} \cap E(S, \bar{S}) = \emptyset$.

Now consider the γ-perturbation of the graph obtained by multiplying the edges in $E_{j^*}^{c^*}$ by γ. If the original graph is γ-stable, the maximal cut of the perturbed graph is $(S, \bar{S})$ as well. Consider the cut obtained by flipping the sides of L_{i^*} and R_{i^*}. That is, denote $Z = S \backslash L_{i^*} \cup R_{i^*}$, and consider the cut $(Z, \bar{Z})$.

The cut $(Z, \bar{Z})$ contains the edges $E_{j^*}^{c_{j^*}}$, which $(S, \bar{S})$ does not. For each $j \neq j^*$, let c_j be such that $E_j^{c_j}$ is in the cut $(S, \bar{S})$ (we will be interested only in non-empty subsets). In the extreme case, all these edges are not in the cut $(Z, \bar{Z})$. Observe that all other edges in $E(S, \bar{S})$ are also in $E(Z, \bar{Z})$.

Define $J = \{j \neq i : E_j^{c_j} \neq \emptyset\}$. Since the weight of $(Z, \bar{Z})$, even in the perturbed graph, is smaller than that of $(S, \bar{S})$, we have

$$\gamma \cdot w(E_{j^*}^{c_{j^*}}) < \sum_{j \in J} w(E_j^{c_j}).$$

(The left-hand side is a lower bound on what we gain when we switch from S to Z, and the right-hand side is an upper bound on the loss.) Recall that $E_{j^*}^{c_{j^*}}$ was chosen to be the set of edges with the largest total weight. Hence,

$$\sum_{j \in J} w(E_j^{c_j}) \leqslant |J| w(E_{j^*}^{c_{j^*}}),$$

and so $\gamma < |J|$. Clearly, $|J| \leqslant \min\{\frac{n}{k}, k\Delta\}$, and so

$$\gamma^2 < \frac{n}{k} k\Delta = n\Delta.$$

This is a contradiction to the assumption that the input is $\sqrt{n\Delta}$-stable.

Since in particular $\gamma > \Delta$, local stability implies that for each vertex we know that the heaviest edge emanating from it is in the optimal cut (if there are several edges of maximal weight then all of them are). Hence, it is safe to start with T having these edges as its edge set. Thus, $k \geqslant 2$, implying that $\frac{n}{2}$ stability is sufficient. □

The concept of stability clearly applies to other combinatorial optimization problems. Similarly, the algorithm above can be adjusted to solve highly stable instances of other problems. For example, a similar algorithm finds the optimal solution to (weighted) $\sqrt{n\Delta}$-stable instances of the Multi-Way Cut problem, and Δ-stable instances of the Vertex Cover problem (where again n is the number of vertices in the graph, and Δ the maximal degree) [3].

3.2. An efficient algorithm for simple graphs of high minimal degree

A complementary approach is useful when the graph is unweighted, and of high minimal degree. Suppose a γ-stable graph, for some big (but bounded) γ has minimal degree $n/2$. Then by local stability each side in the maximal cut must be of size nearly $n/2$, and the neighbourhoods of any two vertices on the same side have most of their vertices in common. Thus we can easily cluster together the vertices into the two sides of the maximal cut. Even when the minimal degree is lower, we can use the same scheme to obtain several clusters of vertices which are certain to be on the same side, and then use the algorithm from the previous subsection to find the maximal cut.

Theorem 3.2. *There is an algorithm that solves in polynomial time every instance of unweighted Max-Cut that is γ-stable for every $\gamma \geqslant \frac{4n}{\delta}$. Here an instance is an n-vertex graph $G = (V, E)$ of minimal degree δ. Furthermore, if $\delta = \Omega\left(\frac{n}{\log n}\right)$, then γ-local stability suffices.*

It clearly suffices to consider $\gamma = \frac{4n}{\delta}$. Let $N_i \subset V$ be the neighbour set of vertex i and $d_i = |N_i|$. Define H to be a graph on V with i, j adjacent if $|N_i \cap N_j| > \frac{d_i + d_j}{\gamma+1}$. Since G is in particular γ-locally stable, every vertex i has at most $\frac{d_i}{\gamma+1}$ of its neighbours on its own side of the maximal cut. Hence, the vertices of each connected component of H must be on the same side of the maximal cut.

Let c be the number of C_i, the connected components of H, and let $U \subset V$ be a set of c vertices, with exactly one vertex from each of these connected components. Let the degrees of the vertices in U be $d_{i_1} \leqslant d_{i_2} \leqslant \cdots \leqslant d_{i_c}$. For any $u, v \in U$ we have that $|N_u \cap N_v| \leqslant \frac{d_u + d_v}{\gamma+1}$. We claim that $c < \frac{\gamma+1}{2}$. If this is not the case, let us apply the inclusion–exclusion formula and conclude

$$\begin{aligned}\left|\bigcup_1^{(\gamma+1)/2} N_i\right| &\geqslant \sum_{j=1}^{(\gamma+1)/2}\left(d_{i_j} - \sum_{k=1}^{j-1}\frac{d_{i_j} + d_{i_k}}{\gamma+1}\right)\\ &= \sum_{j=1}^{(\gamma+1)/2} d_{i_j}\left(1 - \frac{1}{\gamma+1}\sum_{k=1}^{j-1}\left(1 + \frac{d_{i_k}}{d_{i_j}}\right)\right)\\ &\geqslant \sum_{j=1}^{(\gamma+1)/2} d_{i_j}\left(1 - \frac{2(j-1)}{\gamma+1}\right)\end{aligned}$$

since, by assumption $d_{i_k} \leqslant d_{i_j}$ for $k < j$. Also, $d_{i_j} \geqslant \delta$ for all j, and clearly $|\bigcup_1^{(\gamma+1)/2} N_i| < n$. Therefore,

$$\begin{aligned}n > \left|\bigcup_1^{(\gamma+1)/2} N_i\right| &\geqslant \delta \sum_{j=1}^{(\gamma+1)/2}\left(1 - \frac{2(j-1)}{\gamma+1}\right)\\ &= \delta\left(\frac{\gamma+1}{2} - \frac{2(\gamma+1)(\gamma-1)}{8(\gamma+1)}\right) \geqslant \frac{\gamma\delta}{4},\end{aligned}$$

a contradiction, which implies $c < \frac{\gamma+1}{2}$.

Now consider the graph G' obtained from G by contracting all vertices in each C_i into a single vertex, keeping multiple edges. By our previous observation, G' has the same Max-Cut value as G. Consequently, as discussed at the beginning of this section, the graph G' is γ-stable. It follows that G' is a weighted graph whose stability exceeds half its number of vertices. By Theorem 3.1, the optimal cut in G' (and hence in G) can be found in polynomial time, as claimed.

It is also worth mentioning that if $\delta = \Omega\left(\frac{n}{\log n}\right)$ then γ is $O(\log n)$, and we can find the maximal cut in G' by going over all cuts. Moreover, in this case it suffices to assume that G is γ-locally stable. □

4. A spectral approach

4.1. Definitions

Spectral partitioning is a general name for a number of heuristic methods for various graph partitioning problems, which are popular in several application areas. The common theme is to consider an appropriate eigenvector of a possibly weighted adjacency matrix of the graph in question, and partition the vertices according to the corresponding entries. Why is it at least conceivable that such an approach should yield a good solution for Max-Cut? The Max-Cut problem can clearly be formulated as

$$\min_{y\in\{-1,1\}^n} \sum_{(i,j)\in E} W_{i,j} y_i y_j.$$

The Goemans–Williamson algorithm [10] works by solving a semi-definite programming relaxation of the problem. In other words, where, as above, we multiply the matrix W by a rank 1 positive semi-definite matrix, in the semi-definite programming relaxation we multiply it by a positive semi-definite matrix of rank (at most) n. Let us consider instead the relaxation of the condition $y \in \{-1,1\}^n$, to $y \in \mathbb{R}^n, ||y||^2 = n$. The resulting problem is well known. By the variational characterization of eigenvalues, this relaxation amounts to finding the eigenvector corresponding to the least eigenvalue of W. Let u be such a vector. This suggests a *spectral partitioning of* W that is the partition of $[n]$ induced by u.

We also consider what we call *extended spectral partitioning*. Let D be a diagonal matrix. Think of $W + D$ as the weighted adjacency matrix of a graph, with loops added. Such loops do not change the weight of any cut, so that regardless of what D we choose, a cut is maximal in W if and only if it is maximal in $W + D$. Furthermore, it is not hard to see that W is γ-stable if and only if $W + D$ is. Our approach is first to find a 'good' D and then take the spectral partitioning of $W + D$ as the maximal cut. These observations suggest the following question: Is it true that for every γ-stable instance W with γ large enough there exists a diagonal D for which extended spectral partitioning solves Max-Cut? If so, can such a D be found efficiently? Below we present certain sufficient conditions for these statements.

4.2. Spectral partitions of stable instances

The input to the Max-Cut problem is a symmetric non-negative $n \times n$ matrix W with zero diagonal. The *support* of W is a graph $G = (V, E)$ with vertex set $V = [n]$, where $(i, j) \in E$ if and only if $w_{ij} > 0$.

Lemma 4.1. *Let W be a γ-stable instance of Max-Cut with support $G = (V, E)$. Let D be a diagonal matrix, and u an eigenvector corresponding to the least eigenvalue of $W + D$. If*

$$\gamma \geqslant \frac{\max_{(i,j)\in E} |u_i u_j|}{\min_{(i,j)\in E} |u_i u_j|},$$

then the spectral partitioning induced by $W + D$ yields the maximal cut.

Proof. As noted above, for any diagonal matrix D, the problems of finding a maximal cut $W + D$ and in W are equivalent. Normalize u so that $\min_{(i,j)\in E} |u_i \cdot u_j| = 1$. (If u has

any 0 coordinates, the statement of the lemma is meaningless.) Let D' be the diagonal matrix $D'_{i,i} = D_{i,i} \cdot u_i^2$. Let W' be the matrix $W'_{i,j} = W_{i,j} \cdot |u_i u_j|$. Observe that W' is a γ-perturbation of W, hence the maximal cut in W' (and in $W' + D'$), is the same as in W. In other words, mc^* is a vector that minimizes the expression

$$\min_{x \in \{-1,1\}^n} x(W' + D')x.$$

Also, the vector u minimizes the expression

$$\min_{y \in \mathbb{R}^n} \left(\sum_{i,j} W_{i,j} y_i y_j + \sum_i D_{i,i} y_i^2 \right) / \|y\|^2.$$

Think of u as being revealed in two steps. First, the absolute value of each coordinate is revealed, and then, in the second step, its sign. Thus, in the second step we are looking for a sign vector x that minimizes the expression

$$\left(\sum_{i,j} W_{i,j} \cdot |u_i| x_i \cdot |u_j| x_j + \sum_i D_{i,i} u_i^2 \right) / \|u\|^2.$$

Clearly, mc^* is such a vector. Since the input is stable, the optimal cut is unique, and so mc^* and $-mc^*$ are the only such vectors. Hence, the partition they induce is the same as that induced by u. □

Note. A more careful analysis shows a somewhat stronger result. It suffices that

$$\gamma \geqslant \frac{\max_{(i,j) \in E \,:\, u_i u_j < 0} -u_i u_j}{\min_{(i,j) \in E \,:\, u_i u_j \geqslant 0} u_i u_j}.$$

For details, see [3].

4.3. A sufficient condition for extended spectral partitioning

Lemma 4.2. *Let W be an instance of Max-Cut, and let D be the diagonal matrix $D_{i,i} = -mc_i^* \sum_j W_{i,j} mc_j^*$. If $W + D$ is positive semi-definite, then extended spectral partitioning solves Max-Cut for W efficiently.*

Proof. First, note that without loss of generality the solution to Max-Cut is unique. Assume all entries of W are non-negative integers (this is merely scaling). Enumerate the edges, and increase the weight of the ith edge by 2^{-i}. Observe that in this new graph the solution is unique (it is, in fact, encoded by the digits after the decimal point of the optimal value), and that it is also an optimal solution for W. In particular, W is γ-stable for some $\gamma > 1$.

Now, it is easy to see that the vector mc^* is in the kernel of $W + D$. Since $W + D$ is positive semi-definite, 0 is its least eigenvalue, and mc^* is an eigenvector of $W + D$ corresponding to the smallest eigenvalue. Hence, the assertion of Lemma 4.1 holds. It remains to show that Max-Cut can be found efficiently.

Observe that $\text{trace}(D) = w(E_{\text{cut}}) - w(E_{\text{notcut}}) = 2 \cdot w(E_{\text{cut}}) - w(E)$, where E_{cut} is the set of edges in the maximal cut, and E_{notcut} is the set of all other edges. Hence, to determine

the value of the Max-Cut, it suffices to compute $m = \text{trace}(D)$. Since $mc^*(W + D)mc^* = 0$, it follows that $mc^*\ W\ mc^* = -m$.

We claim that $m = \min \text{trace}(A)$ over $A \in \mathbb{A}$, where $\mathbb{A}$ is the set of all positive definite matrices A such that $A_{i,j} = W_{i,j}$ for $i \neq j$. (As we discuss in Section 5.1, this is the dual problem of the Goemans–Williamson relaxation [10].)

It follows that the smallest such trace is $\leqslant m$, since $W + D \in \mathbb{A}$. For the reverse inequality note that every $A \in \mathbb{A}$ satisfies $mc^*Amc^* = -m + \text{trace}(A)$. But A is positive semi-definite, so $\text{trace}(A) \geqslant m$, as claimed.

As observed by Delorme and Poljak [7] (and in fact even in [4]), the theory developed by Grötschel, Lovász and Schrijver [11, 12] for the ellipsoid algorithm makes it possible to efficiently solve the above optimization problem.

As noted at the beginning of the proof, without loss of generality, knowing the optimal cut value is equivalent to knowing the actual optimal partitioning. □

If W is a real symmetric matrix under consideration, we denote its eigenvalues by $\lambda_1 \geqslant \cdots \geqslant \lambda_n$. We show next that if the last two eigenvalues are sufficiently small in absolute value, then the assertion in Lemma 4.2 holds. We also recall the notation $w(i) = \sum_j W_{i,j}$. Since $w(i)$ can be viewed as a 'weighted vertex degree', we denote $\min_i\{w(i)\}$ by $\tilde{\delta} = \tilde{\delta}(W)$.

Lemma 4.3. *Let W be a γ-locally stable instance of Max-Cut with spectrum $\lambda_1 \geqslant \cdots \geqslant \lambda_n$, support G and smallest weighted degree $\tilde{\delta}$. Let D be a diagonal matrix with*

$$D_{i,i} = -mc_i^* \sum_j W_{i,j} mc_j^*.$$

If

$$2\tilde{\delta} \cdot \frac{\gamma - 1}{\gamma + 1} + \lambda_n + \lambda_{n-1} > 0,$$

then $W + D$ is positive semi-definite and Max-Cut can be found efficiently on W.

Proof. Let x (resp. y) be a unit eigenvector of $W + D$ corresponding to the smallest (second smallest) eigenvalue of $W + D$. We can and will assume that x and y are orthogonal. Since 0 is an eigenvalue of $W + D$ (with eigenvector mc^*), it follows that $x(W + D)x \leqslant 0$. If we can show that $y(W + D)y > 0$, then the second smallest eigenvalue of $W + D$ is positive, and this matrix is positive semi-definite, as claimed.

By local stability, $D_{i,i} \geqslant \frac{\gamma-1}{\gamma+1}\tilde{\delta}$, so all of D's eigenvalues are at least $\frac{\gamma-1}{\gamma+1}\tilde{\delta}$.

Therefore

$$xWx \leqslant -xDx \leqslant -\frac{\gamma - 1}{\gamma + 1}\tilde{\delta}.$$

By the variational theory of eigenvalues (the Courant–Fischer theorem), since x and y are two orthogonal unit vectors, we have

$$\lambda_n + \lambda_{n-1} \leqslant xWx + yWy.$$

Also,

$$\frac{\gamma-1}{\gamma+1}\tilde{\delta} \leqslant yDy.$$

When we sum the three inequalities it follows that

$$2\frac{\gamma-1}{\gamma+1}\tilde{\delta} + \lambda_n + \lambda_{n-1} \leqslant y(W+D)y.$$

The lemma follows. Lemma 4.2 implies that extended spectral partitioning solves Max-Cut for W. □

Note. Note that we have actually shown that the solution to the semi-definite programming problem is of rank $n-1$ (only mc^* is in the kernel). As this is the dual problem of the Goemans–Williamson formulation, complementary slackness implies that the solution of the Goemans–Williamson semi-definite program is of rank 1. Hence, in this case solving the Goemans–Williamson semi-definite program gives the Max-Cut as well.

A slightly stronger version of this lemma can be proved for the regular case.

Lemma 4.4. *Let W be an instance of Max-Cut with spectrum $\lambda_1 \geqslant \cdots \geqslant \lambda_n$, support G and all weighted degrees $\tilde{\delta}$. Let $c \in \{-1,1\}^n$ be a γ-locally stable cut in G. Let D be a diagonal matrix with $D_{i,i} = -c_i \sum_j W_{i,j}c_j$. If*

$$\gamma > \frac{2\tilde{\delta}}{\lambda_{n-1}+\tilde{\delta}} - 1,$$

then $W+D$ is positive semi-definite, c is the Max-Cut and it can be found efficiently.

Proof. Observe that, just as in the proof above, $D_{i,i} \geqslant \frac{\gamma-1}{\gamma+1}\tilde{\delta}$. Since $D_{i,i} \leqslant \tilde{\delta}$, all eigenvalues of D are in $[\tilde{\delta}(1-\frac{2}{\gamma+1}), \tilde{\delta}]$. As in the proof above, since $(W+D)c = 0$, to prove that $W+D \in PSD_n$ it is enough to show that $\lambda_{n-1}(W+D) > 0$. By a theorem of Weyl,

$$\lambda_i(W+D) \in \lambda_i(W) + \left[\left(1-\frac{2}{\gamma+1}\right)\tilde{\delta}, \tilde{\delta}\right],$$

and so it is enough that

$$\lambda_{n-1}(W) > -\left(1-\frac{2}{\gamma+1}\right)\tilde{\delta},$$

as claimed. □

Corollary 4.5. *Max-Cut can be found efficiently on 3-regular graphs with $\lambda_{n-1} > -1$.*

Proof. This follows from Lemma 4.4 and the observation that 3-regular graphs are always 2-locally stable. □

4.4. Examples of graph families on which Max-Cut can be found efficiently
Lemma 4.3 gives a sufficient condition under which the extended spectral partitioning solves Max-Cut efficiently. In this subsection we identify certain families of graphs for which the assertion in the lemma holds.

Example 4.1. *Let G be a γ-locally stable d-regular simple graph with second eigenvalue λ. Max-Cut can be found efficiently on G if*

$$\gamma > \frac{5d + \lambda}{d - \lambda}.$$

Note that this family of graphs is not empty: for example, it includes all connected d-regular bipartite graphs.

Proof. Let A be the adjacency matrix of G, and let A_{in} be the adjacency matrix of the graph spanned by the edges of the maximal cut. Let $A_{\text{out}} = A - A_{\text{in}}$.

Since G is γ-locally stable, the maximal degree in A_{out}, and hence its spectral radius, is at most $\frac{d}{\gamma+1}$. Therefore, by subtracting A_{out} from A, eigenvalues are shifted by at most this value (this follows by Weyl's theorems on matrix spectra, for example). In other words, the second eigenvalue of A_{in} is at most $\lambda + \frac{d}{\gamma+1}$. Since A_{in} is bipartite, its spectrum is symmetric, and so $|\lambda_{n-1}(A_{\text{in}})| \leqslant \lambda + \frac{d}{\gamma+1}$. Now adding A_{out} to A_{in} again shifts the spectrum by at most $\frac{d}{\gamma+1}$, and so $|\lambda_{n-1}(A)| \leqslant \lambda + \frac{2d}{\gamma+1}$ (and in fact we know that $\lambda_{n-1}(A)$ is negative, by the Perron–Frobenius theorem).

For the condition in Lemma 4.4 to hold, it thus suffices that $-\lambda - \frac{2d}{\gamma+1} > \frac{2d}{\gamma+1} - d$, which is equivalent to that claimed. □

An example of a family of graphs for which the lemma holds is as follows. Start with a d-regular bipartite graph with n vertices on each side, which is a good expander ($\lambda = o(d)$). Then add edges to this graph: $d' = \frac{1}{7}d$ new edges to each vertex. The resulting graph is at least 7-locally stable: for each vertex, at least d of its edges participate in the maximal cut, and at most d' do not. The second largest eigenvalue of the new graph is at most $d' + o(d)$, since to the original adjacency matrix we have added a matrix with spectral radius d' . The new graph is $(d + d')$-regular, hence the lemma holds for this family of graphs.

Example 4.2. *Let $G = (V, E)$ be a γ-locally stable, d-regular graph with Cheeger constant h. Max-Cut can be found efficiently on G if*

$$\gamma > \frac{5 + \sqrt{1 - (h/d)^2}}{1 - \sqrt{1 - (h/d)^2}}.$$

Proof. Recall that the Cheeger constant of a graph is defined by

$$h(G) = \min_{U \subset V \,:\, |U| \leqslant \frac{n}{2}} \frac{|E(U, \bar{U})|}{|U|},$$

and provides an upper bound on G's second eigenvalue (*e.g.*, [15]):

$$\lambda_2(G) \leqslant \sqrt{d^2 - h(G)^2}.$$

By Example 4.1, Max-Cut can be found efficiently on G. □

Example 4.3. *Let $G = (V, E)$ be a (γ, ϵ)-locally stable d-regular graph. Max-Cut can be found efficiently on G if*

$$\gamma > \frac{30}{\epsilon^2}.$$

Proof. We will first need a technical observation. For (γ, ϵ)-local stability to make sense it must be the case that $\epsilon \leqslant \frac{1}{2}$, since clearly adding more than d edges between a vertex and other vertices on the same side of the maximal cut causes the maximal cut to change. So in particular, we assume here that $\gamma > 80$. It is not hard to verify that, by Example 4.2, if $h/d > 0.315$, such a value of γ implies that Max-Cut can be found efficiently on G. So we may assume $h/d < 0.315$

Now, let $(S, \bar{S})$ be the maximal cut in G, and consider the set U (such that $|U| \leqslant \frac{n}{2}$) for which the Cheeger constant $h = h(G)$ is obtained. Denote $U_S = U \cap S$, and similarly $U_{\bar{S}}$, $\bar{U}_S$ and $\bar{U}_{\bar{S}}$. Assume without loss of generality that $|U_S| \geqslant |U_{\bar{S}}|$.

Consider adding $\epsilon d|U|$ edges: $\epsilon d|U_S|$ between U_S and $\bar{U}_S$, and $\epsilon d|U_{\bar{S}}|$ between $U_{\bar{S}}$ and $\bar{U}_{\bar{S}}$.

Assume that this can be done in such a way that the degree of each vertex is increased by at most $2\epsilon d$. Then by the second property of (γ, ϵ)-stability the partitioning of the maximal cut should not change. Hence, it must be the case that $h(G) \cdot |U| \geqslant \epsilon d|U|$ or $h(G) \geqslant \epsilon d$. This, in turn, implies that $\sqrt{1 - (h(G)/d)^2} \leqslant 1 - \frac{1}{2}\epsilon^2$, and by Example 4.2, if γ is as stated above then Max-Cut can be found efficiently on G.

It remains to show that it is possible to add $\epsilon d|U|$ edges under the aforementioned restrictions. Let us count the number of edges between U_S and $U_{\bar{S}}$. At most $h|U|$ of the edges meeting U cross over to $\bar{U}$ (by definition of h). At most $\frac{d|U|}{2(\gamma+1)}$ are internal to either U_S or $U_{\bar{S}}$. Hence, there must be at least

$$\frac{1}{2}\left(d|U| - \frac{d|U|}{(\gamma+1)} - h|U|\right)$$

between U_S and $U_{\bar{S}}$. This, in turn, implies that

$$\frac{|U|}{2}\left(1 - \frac{1}{(\gamma+1)} - \frac{h}{d}\right) \leqslant |U_S|, |U_{\bar{S}}| \leqslant \frac{|U|}{2}\left(1 + \frac{1}{(\gamma+1)} + \frac{h}{d}\right).$$

Similarly, and since $|U| \leqslant |\bar{U}|$, we have

$$\frac{|U|}{2}\left(1 - \frac{1}{(\gamma+1)} - \frac{h}{d}\right) \leqslant |\bar{U}_S|.$$

So we have $|U_S|/|\bar{U}_S| \leqslant \frac{1+\alpha}{1-\alpha}$, where we denote $\alpha = \frac{1}{(\gamma+1)} + \frac{h}{d}$. Now, adding $\epsilon d|U_S|$ edges from U_S uniformly to $\bar{U}_S$ increases the degrees in $\bar{U}_S$ by at most $\epsilon\frac{1+\alpha}{1-\alpha}d$. For the restrictions

to hold we need that $\frac{1+\alpha}{1-\alpha} < 2$ or $\alpha < \frac{1}{3}$. But by the technical observation at the beginning, $\gamma \geqslant 80$ and $h/d < 0.315$, and so $\alpha < 0.33$. □

5. Results derived from previous works

5.1. Performance of the Goemans–Williamson approximation algorithm

Let us quickly recall the Goemans and Williamson (GW) approximation algorithm for Max-Cut [10]. We first rephrase the Max-Cut problem as

$$\text{maximize } \frac{1}{2}\sum_{(i,j)\in E} W_{i,j}(1 - y_i y_j)$$
$$\text{over } y \in \{-1,1\}^n.$$

Equivalently, we seek to minimize $\sum_{(i,j)\in E} W_{i,j}Y_{i,j}$ over all $\{-1,1\}$-matrices Y that are positive semi-definite and of rank 1. In the GW algorithm the rank constraint is relaxed, yielding a semi-definite programming problem which can be solved efficiently with approximation guarantee of ~ 0.8786. Moreover, they show that when the weight of the maximal cut is sufficiently big, this guarantee can be improved. Namely, let R ($\geqslant \frac{1}{2}$) be the ratio between the weight of the maximal cut and the total weight of the edges. Let $h(t) = \arccos(1-2t)/\pi$. Then the approximation ratio is at least $h(R)/R$.

By local stability, the contribution of each $v \in V$ to the maximal cut is $\frac{\gamma}{\gamma+1}$ the total weight of the edges incident with it. Summing this over all vertices, we get that the maximal cut weighs at least $R = \frac{\gamma}{\gamma+1}$ of the total weight. Thus, the performance guarantee of the GW algorithm on γ-stable instances is at least $\left(1 - O\left(\frac{1}{\sqrt{\gamma}}\right)\right)$.

Note that for this we only required local stability.

The semi-definite program used in the GW algorithm can be strengthened when the input is γ-stable, by inequalities that express this stability. It is interesting to consider whether these additional constraints can improve the approximation ratio further.

5.2. Spectrally partitioning random graphs

Consider the following model for random weighted graphs. Let P be some probability measure on $[0,\infty)$. Generate a matrix W' (a weighted adjacency matrix), by choosing each entry $W'_{i,j}$, $i < j$, independently from P. Set $W'_{i,j} = W'_{j,i}$ for $i > j$, and $W'_{i,i} = 0$. Let C be the set of edges in the maximal cut of W (for 'reasonable' P, this will be unique w.h.p.). Set $W_{i,j} = \gamma \cdot W'_{i,j}$ for $(i,j) \in C$.

It is easy to see that W is indeed γ-stable, yet for certain probability measures the problem becomes trivial. For example, if P is a distribution on $\{0,1\}$, the maximal cut in W simply consists of all the edges with weight γ.

An even simpler random model is the following. Take n even. Generate an $n \times n$ matrix W' as above. Choose $S \subset [n]$, $|S| = n/2$ uniformly at random. Let C be the set of edges in the cut $(S, \bar{S})$. Set $W_{i,j} = \gamma \cdot W'_{i,j}$ for $(i,j) \in C$. Denote this distribution $\mathbb{G}(n,P,\gamma)$. For an appropriate γ, w.h.p. $(S,\bar{S})$ will be the maximal cut in W. This random model is close to what is sometimes known as the 'planted partition model' [5, 4, 8, 13, 6, 9, 14, 18].

Following work by Boppana [4] on a similar random model (for unweighted graphs), we can deduce that w.h.p. the maximal cut of graphs from this distribution can be found efficiently.

Theorem 5.1. *Let P be a distribution with bounded support, expectation μ and variance σ^2. There exists a polynomial-time algorithm that w.h.p. solves Max-Cut for $G \in \mathbb{G}(n, P, \gamma)$, when*

$$\gamma = 1 + \Omega\left(\sqrt{\frac{\log n}{n}}\right).$$

The theorem follows from Lemma 4.2 and the following one, which is an easy consequence of [4].

Lemma 5.2. *Let P be a distribution with bounded support, expectation μ and variance σ^2. Let $G \in \mathbb{G}(n, P, \gamma)$, and S the subset chosen in the generating G. Let $mc \in \{-1, 1\}^n$ be the indicator vector of the cut $(S, \bar{S})$. Let D be the diagonal matrix defined by $D_{i,i} = -mc_i \sum_j W_{i,j}\, mc_j$. If*

$$\gamma \geqslant 1 + \Omega\left(\sqrt{\frac{\log n}{n}}\right),$$

then w.h.p.:

(1) *mc is the indicator vector of the maximal cut in G.*
(2) *$W + D$ is positive semi-definite.*

6. Conclusion, open problems and acknowledgements

In this work we have shown that stability, supplemented by certain properties of the input instance, allows for an efficient algorithm for Max-Cut. However, if nothing is assumed about the input, we only know that n-stability is sufficient. Can this be improved? Note that $\gamma \geqslant n$ is very far from what happens in the random model, where it is only required that

$$\gamma \geqslant 1 + \Omega\left(\sqrt{\frac{\log n}{n}}\right).$$

A bold conjecture is that there is some constant, γ^*, such that γ^*-stable instances can be solved in polynomial time. By contrast, can it be proved that Max-Cut is NP-hard even for γ-stable instances for some small constant γ?

Our motivation in defining stability and distinctness is to identify natural properties of a solution to an NP-hard problem, which 'make it interesting', and allow it to be found in polynomial time. Stability and distinctness indeed make Max-Cut amenable, but are in no way the only possible properties, and it would be very interesting to suggest others.

We thank Ting-Yu Lin for finding an error in an earlier version of the manuscript. We also thank an anonymous reviewer for pointing us towards [2].

References

[1] Ackerman, M. and Ben David, S. (2009) Clusterability, A theoretical study. *Proc. 12th International Conference on Artificial Intelligence and Statistics*, Vol. 5, pp. 1–8.

[2] Balcan, M. F., Blum, A. and Gupta, A. (2009) Approximate clustering without the approximation. In *SODA '09: Proc. Nineteenth Annual ACM–SIAM Symposium on Discrete Algorithms*, SIAM, pp. 1068–1077.

[3] Bilu, Y. On spectral properties of graphs and their application to clustering. PhD Thesis, available at `http://www.cs.huji.ac.il/~nati/PAPERS/THESIS/bilu.pdf`.

[4] Boppana, R. (1987) Eigenvalues and graph bisection: An average case analysis. In *28th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, pp. 280–285.

[5] Bui, T. N. Chaudhuri, S., Leighton, F. T. and Sipser, M. (1987) Graph bisection algorithms with good average case behavior. *Combinatorica* **7** 171–191.

[6] Condon, A. and Karp, R. M. (2001) Algorithms for graph partitioning on the planted partition model. *Random Struct. Alg.* **18** 116–140.

[7] Delorme, C. and Poljak, S. (1993) Laplacian eigenvalues and the maximum cut problem. *Math. Programming* **62** 557–574.

[8] Dyer, M. E. and Frieze, A. (1986) Fast solution of some random NP-hard problems. In *27th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, pp. 313–321.

[9] Feige, U. and Kilian, J. (2001) Heuristics for semirandom graph problems. *J. Comput. System Sci.* **63** 639–671.

[10] Goemans, M. X. and Williamson, D. P. (1995) Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. Assoc. Comput. Mach.* **42** 1115–1145.

[11] Grötschel, M., Lovász, L. and Schrijver, A. (1981) The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* **1** 169–197.

[12] Grötschel, M., Lovász, L. and Schrijver, A. (1984) Corrigendum to our paper: 'The ellipsoid method and its consequences in combinatorial optimization' [*Combinatorica* **1** (1981) 169–197]. *Combinatorica* **4** 291–295.

[13] Jerrum, M. and Sorkin, G. B. (1998) The Metropolis algorithm for graph bisection. *Discrete Appl. Math.* **82** 155–175.

[14] McSherry, F. (2001) Spectral partitioning of random graphs. In *42nd IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society, pp. 529–537.

[15] Mohar, B. (1989) Isoperimetric numbers of graphs. *J. Combin. Theory Ser. B* **291** 47–274.

[16] Ostrovsky, R., Rabani, Y., Schulman, L. J. and Swamy, C. (2006) The effectiveness of Lloyd-type methods for the k-means problem. In *FOCS '06: Proc. 47th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society, pp. 165–176.

[17] Papadimitriou, C. H. (1994) *Computational Complexity*, Addison-Wesley.

[18] Shamir, R. and Tsur, D. (2002) Improved algorithms for the random cluster graph model. In *Proc. 8th Scandinavian Workshop on Algorithm Theory*, pp. 230–239.

[19] Spielman, D. and Teng, S. H. (2001) Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In *Proc. 33rd Annual ACM Symposium on Theory of Computing*, ACM Press, pp. 296–305.

[20] Vershynin, R. (2006) Beyond Hirsch conjecture: Walks on random polytopes and smoothed complexity of the simplex method. In *FOCS '06: Proc. 47th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society, pp. 133–142.