Lectures 10 (part), 11

Uriel Feige

March 31, June 7, 2015

1 Perfect graphs and the theta function of Lovasz

1.1 Perfect graphs

Let $\omega(G)$, $\alpha(G)$, $\chi(G)$, $\bar{\chi}(G)$ denote the maximum clique size, maximum independent set size, chromatic number, and clique cover number of graph G, respectively. They are related as follows (here $\bar{G}$ denotes the edge complement of graph G):

$\omega(G) = \alpha(\bar{G})$
$\chi(G) = \bar{\chi}(\bar{G})$
$\chi(G) \geq \omega(G)$
$\chi(G) \cdot \alpha(G) \geq n$

A graph G is *perfect* if $\omega(G') = \chi(G')$ for every vertex induced subgraph G' of G.

We give examples of several families of graphs that are perfect, and note in passing that the complements of these families are also perfect. In other words, also $\alpha(G') = \bar{\chi}(G')$ hold for every vertex induced subgraph.

Bipartite graphs. Every vertex induced subgraph of a bipartite graph is also bipartite. For every bipartite graph G with at least one edge, $\omega(G) = \chi(G) = 2$. Note also that $\alpha(G) = \bar{\chi}(G)$ (where the latter is the minimum number of edges needed to cover all vertices). To see this, consider a maximum matching. As we have already seen, each edge covers one vertex of minimum vertex cover and one in the maximum independent set. The rest of the vertices in the maximum independent set can be covered one by one, either by unmatched edges, or by the vertices themselves (which are cliques of size 1).

Line graphs of bipartite graphs. Let G_l be the line graph of the bipartite graph G. Then $\omega(G_l)$ is the maximum degree in G, and $\chi(G_l)$ is the chromatic index of G, which are known to be equal for bipartite graphs. We also have that $\alpha(G_l)$ is the size of the maximum matching in G, and $\bar{\chi}(G_l)$ is the size of the minimum vertex cover, which again are equal for bipartite graphs.

Interval graphs. We will not elaborate on them, as they are a special case of complements of comparability graphs.

Comparability graphs. Given a partial order on n elements, we treat each element as a vertex, and draw an edge between vertices i and j if either $i \leq j$ or $j \leq i$. Note that an order relation is transitive. A vertex induced subgraph of a comparability graph is a comparability graph. A clique in a comparability graph is a chain in the partial order. An independent set is an antichain. It is easy to show that $\omega(G) = \chi(G)$. This can be proved

by induction on n: remove the maximum element from every maximum length chain; the maximum chain length decreased by one, and the set of all elements removed is an antichain.

Perhaps more interestingly, we have Dilworth's theorem.

Theorem 1 *In every partial order, the cardinality of the largest antichain is equal to the minimum number of chains that cover all elements.*

Proof: Consider an arbitrary maximal (by inclusion) chain C. If every maximum antichain A intersects C then by removing C, the cardinality of the largest antichain decreases by 1, and we can proceed by induction. So let us assume that A and C do not intersect. Maximality of A implies that every element not in A is comparable with some element of A (otherwise A can be extended), and no element not in A has a larger element in A and a smaller element in A (as otherwise these two elements would be comparable, and A would not be an antichain). The smallest element in C is smaller than some element of A and the largest element of C is larger than some element of A (otherwise C can be extended).

Consider all elements except for those that are smaller than some element in A. They can be covered by $|A|$ chains (by induction). The same holds for all elements except those that are larger than some element in A. Using A as the gluing points between these two sets of chains, all elements are covered. □

It follows that the complement of a comparability graph is perfect.

Chordal graphs. A graph is chordal if every cycle of length at least 4 has a chord. These graphs and their complements are perfect.

Lovasz proved the so called *weak perfect graph conjecture*, that the complement of a perfect graph is perfect. Hence for perfect graphs, $\alpha(G') = \bar{\chi}(G')$ for every vertex induced subgraph G' of G.

A chordless odd cycle of length at least five is not a perfect graph, and neither is its complement. The strong perfect graph conjecture of Berge says that every nonperfect graph must contain one of the above as a vertex induced subgraph. The conjecture was proved by Chunovsky and Seymour. Insights from their proof helped in designing a polynomial time algorithm that tests whether a graph is perfect or not.

1.2 A linear program

Let f be a function on graphs with the following "sandwich" property: $\alpha(G) \leq f(G) \leq \bar{\chi}(G)$. If f is computable in polynomial time, it can be used to compute $\omega(G)$, $\alpha(G)$, $\chi(G)$, $\bar{\chi}(G)$ for perfect graphs.

Consider the following primal LP.

maximize $\sum_i x_i$
subject to
$\sum_{i \in C_j} x_i \leq 1$ for every clique C_j,
$x_i \geq 0$ for every vertex i.

Clearly, the above LP gives an upper bound on $\alpha(G)$. Its dual is:

minimize $\sum_j y_j$
subject to
$\sum_{\{j | \text{clique } C_j \text{ contains vertex } i\}} y_j \geq 1$ for every vertex i,
$y_j \geq 0$ for every clique C_j.

Clearly, the dual gives a lower bound on $\bar{\chi}(G)$.

Hence the above LP has the desired sandwich property. Unfortunately, to solve it we seem to need a procedure for maximum weight clique in a vertex weighted graph. This problem is NP-hard. Moreover, it can be shown that for general graphs, finding the optimal value of this LP (the so called *fractional clique cover number*) is NP-hard.

For perfect graphs, both the LP and its dual necessarily have integer optimal solutions. As we shall later see, for perfect graphs the LP can be solved in polynomial time.

1.3 The theta function of Lovasz

The ϑ function of Lovasz ("vartheta" in latex) has the sandwich property, and can be computed with arbitrary precision in polynomial time (via a semidefinite program).

There are many alternative definitions to the ϑ function. Here is a definition of the ϑ function as a minimization problem.

Associate with each vertex a unit vector in R^n. The two vectors associated with vertices that are nonadjacent must have inner product $-1/(k-1)$. Under the above restrictions, minimize k.

Observe that given a vector arrangement with a particular value of k, we can always transform it to a vector arrangement with a larger value of k, by adding another coordinate in which all vectors have the same value, and scaling the vectors to have norm 1.

Theorem 2 *For every graph G, $\alpha(G) \leq \vartheta(G) \leq \bar{\chi}(G)$.*

Proof: Suppose that G can be covered by k disjoint cliques. With each clique associate a unit vector, and make the unit vectors the vertices of a perfect $(k-1)$ dimensional simplex. All inner products are $-1/(k-1)$. This satisfies the constraints, and gives k as an upper bound on the minimization version of $\vartheta(G)$. (Algebraically, the vertices of the simplex can be described as k vectors in R^k. For vector x_i, the ith coordinate is $k-1$ whereas all other coordinates are -1. Hence $x_i^T x_i = (k-1)^2 + (k-1) = k(k-1)$ and $x_i^T x_j = -2(k-1) + k - 2 = -k$. Scaling each vector by $\frac{1}{\sqrt{k(k-1)}}$ gives the desired simplex.)

Suppose that the largest independent set in G is of size k. Associate a unit vector x_i with each vertex v_i of this independent set, and let $x = \sum_{i=1}^{k} x_i$. We have

$$0 \leq x^T x = (\sum x_i^T)(\sum x_i) = k + 2\sum_{i<j} x_i^T x_j \leq k + k(k-1)\max_{i \neq j}[x_i^T x_j]$$

Hence $\max_{i \neq j}[x_i^T y_j] \geq -1/(k-1)$, and k is a lower bound on $\vartheta(G)$. $\square$

Note that $\vartheta(G)$ might be irrational (e.g., for the pentagon we have $\vartheta(C_5) = \sqrt{5}$), and hence we should not expect to compute it exactly.

Theorem 3 *For every graph G and every $\epsilon > 0$, $\vartheta(G)$ can be computed in polynomial time up to a precision of ϵ.*

Proof: To see if $\vartheta(G) \leq k$, set up the following constraints over the vectors x_i.
$x_i^T x_i = 1$,
$x_i^T x_j = -1/(k-1)$ for every $(i,j) \notin E$.

This can equivalently be represented as the following positive semidefinite program (SDP). We associate a variable y_{ij} with every inner product $x_i^T x_j$, and set up the following constraints.

- $y_{ii} = 1$ for every i,
- $y_{ij} = -1/(k-1)$ for every $(i,j) \notin E$,
- The n by n matrix $Y = \{y_{ij}\}$ is symmetric and positive semidefinite (psd).

Observe that the first two sets of constraints just fix the values of the y_{ii} variables and some of the y_{ij} variables to be specific constants, and hence these variables can be replaced by constants. The only constraint that remains is the psd constraint. Slightly relaxing the psd constraint so that the feasible region contains a bounded ball (the relaxation allows slightly negative eigenvalues), one can solve the relaxed SDP using the ellipsoid algorithm. Then one can make the resulting solution matrix psd by adding a small ϵ along the diagonal, and decompose this modified solution matrix Y into $Q^T Q$. The columns of Q, after scaling by $\frac{1}{\sqrt{1-\epsilon}}$ so as to normalize their norm to 1, can serve as the vectors x_i, giving $x_i^T x_j = -\frac{1}{(1+\epsilon)(k-1)}$ for every $(i,j) \notin E$.

Observe that if $\epsilon < \frac{1}{k-1}$, this solution certifies that $\vartheta(G) < k+1$, which for perfect graphs (for which the theta function has integer values) certifies that $\vartheta(G) \leq k$). $\square$

In perfect graphs, one can compute $\alpha(G)$ exactly, by computing $\vartheta(G)$ up to a precision better than $\frac{1}{2}$, and rounding to the nearest integer. The fact that $\alpha(G)$ can be computed exactly for perfect graphs, combined with the fact that every vertex induced subgraph of a perfect graph is perfect as well, allows one to use a reduction from a decision problem to a search problem in order to find an independent set of size $\alpha(G)$ (whenever G is a perfect graph). The basic idea is as follows. For an arbitrary vertex v, use the ϑ function to compare between $\alpha(G)$ and $\alpha(G^{-v})$, where G^{-v} denotes the subgraph induced on all vertices but v. If these two values are equal, then v is not needed for the maximum independent set in G, and it can be discarded permanently from G. If on the other hand $\alpha(G^{-v}) = \alpha(G) - 1$ then v is needed. In this case, remove v and all its neighbors from G, and include v in the output independent set. In any case, one remains with a strict subgraph of G, on which the process can be repeated with a new vertex from this subgraph, until the empty graph is reached.

Note that the above procedure will for every graph either find a maximum independent set (if the output is an independent set of size $\vartheta(G)$, then this independent set is necessarily of maximum cardinality, regardless of whether G if perfect), or certify that the graph in not perfect (if the output is not an independent set of size $\vartheta(G)$, then the graph cannot be perfect).

We shall also need to consider cliques in vertex weighted graphs. Here vertex i has weight w_i, and one wishes to find a clique of maximum weight. In general, algorithms for max clique that work in unweighted graphs work also for weighted graphs, and this applies to the ϑ function as well. A generic reduction (that can be used when we can afford tiny modifications to the weight of cliques) is as follows. Scale all weights so that the maximum weight vertex was weight 1. Pick some sufficiently small ϵ (such as $\epsilon = 1/n^2$), and round all vertex weights to the nearest multiple of ϵ. Now multiply all weights by $1/\epsilon$, and they become integer. Replace every vertex i of integer weight $w_i > 1$ by a clique of size w_i

(every vertex in the clique will have weight 1) connected to all neighbors of vertex i. We obtain an unweighted graph with a polynomial number of vertices, and a straightforward correspondence between weighted cliques in the original graph and cliques in the new graph. It is important for our application that if the original graph was perfect, then so is the resulting graph. And indeed, this is the case, for the following reason. Every cycle of length at least 5 that includes two representatives of the same original vertex i must have a chord. The same applies also to the complement of the graph.

To find a minimum clique-cover (a clique-cover is a set of cliques that cover all vertices) of a perfect graph, our plan is to find what we shall call a *blocking clique* – this is a clique that intersects all maximum independent sets. Once we find a blocking clique, we can remove it, and repeat. The total number of cliques used will equal $\alpha(G)$, as required in perfect graphs. Note that the maximum clique is not guaranteed to be a blocking clique, hence our ability to find maximum cliques using the theta function does not suffice by itself. We now explain a way for finding a blocking clique.

Consider the primal LP of Section 1.2. For a perfect graph G, the value of the LP was seen to be $\alpha(G)$. The value of $\alpha(G)$ may be assumed to be known (by computing the theta function). Pick a small $\epsilon > 0$ (inverse polynomial in n), and replace the objective function of the LP by a constraint $\sum_i x_i = \alpha(G) + \epsilon$. This new LP is nearly feasible, but actually infeasible because $\epsilon > 0$. Nevertheless, try to find a feasible solution using the ellipsoid algorithm. At each iteration one needs to see if there is a violated clique constraint, and this can be done by finding a clique of approximately maximum weight with the help of the theta function. (It suffices for the following arguments that the weight approximates the maximum weight up to a multiplicative factor of $1 - O(\frac{\epsilon}{n})$.) When the ellipsoid algorithm ends (declaring that the LP is not feasible), consider the list of clique constraints that were used. The LP is infeasible also with respect to these cliques alone. Hence call them the *essential* cliques, and consider a modified primal LP that has only the clique constraints that correspond to the essential cliques. The value of the modified primal LP is at least $\alpha(G)$ and at most $\alpha(G) + \epsilon$, and it has polynomially many constraints. Now go to the dual program of the modified primal. The essential cliques are the basic variables of the dual, and there are only polynomially many of them. Solve the dual, to obtain a "fractional" clique-cover. By LP duality, the value of the dual is equal of that of the primal, which is at most $\alpha(G) + \epsilon$. Take the clique with largest fractional weight. If ϵ is sufficiently small (some inverse polynomial in n), then the clique's weight would be larger than ϵ. Hence it must intersect every maximum size independent set (namely, it is a blocking clique, as desired), because the other cliques did not contribute enough fractional weight to cover all vertices of the maximum independent set.