A Sublinear-Time Randomized Parallel Algorithm for the Maximum Clique Problem in Perfect Graphs *

Farid Alizadeh†

Abstract

We will show that Lovasz number of graphs may be computed using interior-point methods. This technique will require $O(\sqrt{|V|})$ iterations, each consisting of matrix operations which have polylog parallel time complexity. In case of perfect graphs Lovasz number equals the size of maximum clique in the graph and thus may be obtained in sublinear parallel time. By using the isolating lemma, we get a Las Vegas randomized parallel algorithm for constructing the maximum clique in perfect graphs.

1 Introduction.

In this work, we will be studying algorithms for computation of maximum cliques and maximum independent sets in perfect graphs. A graph $G = (V, E)$ is *perfect* when, for all of its induced subgraphs G', the size of the maximum clique, $\omega(G')$, is equal to the size of the minimum vertex coloring $\chi(G')$. The celebrated perfect graph theorem of Lovasz [12] indicates that the complements of perfect graphs are also perfect; in other words, for all induced subgraphs G' of G, the size of the largest independent set, $\alpha(G')$, is equal to the size of the minimum clique cover, $\rho(G')$. Grötschel, Lovasz and Schrijver have shown that for perfect graphs one can compute the largest clique and the largest independent set in polynomial time; see [6], [7], [8], and in particular, their elaborate book [9]. Their idea is based on computing an invariant known as the Lovasz number of graphs $\theta(G)$. Lovasz has shown that for all graphs $\omega(G) \leq \theta(G) \leq \chi(G)$. As will be seen in the next section, $\theta(G)$ is defined as the minimum of some convex function derived from the graph. Grötschel, Lovasz and Schrijver use the ellipsoid method for convex programming to establish polynomial time computability of $\theta(G)$. For perfect graphs, since $\omega(G) = \theta(G) = \chi(G)$, it follows that $\omega(G)$ and $\chi(G)$ (and also $\alpha(G)$ and $\rho(G)$) are polynomial time computable. The process of self-reducibility then yields a polynomial time algorithm for construction of maximum cliques and maximum independent sets in these graphs. It should be noticed that in general Lovasz number of a graph is not even a rational number (it is algebraic though), and its polynomial time computability must be qualified by the required number of significant digits.

The ellipsoid method, though of polynomial time complexity, is in practice an inefficient and numerically unstable algorithm. Indeed, Grötschel, Lovasz and Schrijver report that they had difficulty making the algorithm converge for graphs with even 10 or 20 vertices [8]. In this paper, we offer another algorithm for computing $\theta(G)$ which is based on interior point techniques for non-linear programming. The algorithm is numerically more stable. Furthermore, similar to interior point methods for linear programming, the number of iterations used by this algorithm is proportional to the square root of the number of inequality constraints; for Lovasz number of graphs this turns out to be $O^*(\sqrt{|V|})$. (Following Goldberg et al. in [5], $O^*(f(n))$ is synonymous with $O(f(n)\log^k(n))$ for some constant k.) This follows from the general results in the recent work of Nesterov and Nemirovsky [17]. Once we have an oracle which computes $\theta(G)$, and therefore $\omega(G)$ in the case of perfect graphs, we may use a randomized technique based on the isolating lemma of Mulmuley, Vazirani and Vazirani [16] to construct the maximum clique and the maximum independent set in parallel.

Our approach may be viewed somewhat loosely as a generalization of works of Goldberg, Plotkin, Shmoys and Tardos [5] and of Mulmuley, Vazirani and Vazirani [16]. Goldberg et al. use an interior point linear programming technique to compute the maximum matching in bipartite graphs. The particular variant used there is based on the work by Ye [20]. The authors achieve a *deterministic* $O^*(\sqrt{|E|})$ parallel algorithm for the vertex weighted maximum matching problem and–when the weights of the edges are bounded–for the edge weighted maximum matching problem. Also, Karp, Upfal and Widgerson had already discovered a *randomized* polylog time algorithm for the matching problems men-

*This work was supported in part by the Air Force Office of Scientific Research grant AFOSR-87-0127, the National Science Foundation grant DCR-8420935 and the Minnesota Supercomputer Institute.

†Computer Science Department, University of Minnesota, Minneapolis, Mn, 55455; e-mail: alizadeh@cs.umn.edu.

tioned above, [11]. Their work was somewhat simplified by Mulmuley, Vazirani and Vazirani [16] and was shown to be a Las Vegas type randomized algorithm by Karloff [10]. Since by the classical theorem of König, line graphs of bipartite graphs are perfect, the bipartite matching problem is a special case of independent set problem in perfect graphs .

Aside from theoretical issues, the algorithm presented here may be modified to yield practical, that is relatively fast and numerically stable methods for construction of maximum cliques and maximum independent sets in perfect graphs. These issues will be addressed at the end of this work.

2 Definitions and Formulations.

Let $G = (V, E)$ be a simple undirected graph, that is no multiple edges between vertices are allowed, nor is an edge from a vertex to itself, and let $n = |V|$ and $e = |E|$. Let also that each vertex i be assigned a non-negative integer weight w_i, $\mathbf{w} \in R^n$ the weight vector, $\mathbf{w}^{1/2} \in R^n$ a vector whose i^{th} entry is $\sqrt{w_i}$, and $W = (\mathbf{w}^{1/2})(\mathbf{w}^{1/2})^T$ a symmetric rank one $n \times n$ matrix. (Throughout this paper we use boldface lower case letters to name column vectors.) The *weighted clique number* of G, $\omega(G, \mathbf{w})$, is the maximum weight of a clique in G, where the weight of a subgraph is defined as sum of the weights of its vertices. Also, the *weighted chromatic number* of G, $\chi(G, \mathbf{w})$, is the smallest number of colors assigned to vertices of G so that each vertex i has at least w_i colors and no two adjacent vertices share a color. Weighted independence number and weighted clique covering number are defined similarly. Clearly, the unweighted case is the same as when $\mathbf{w} = \mathbf{1}$, where $\mathbf{1}$ is the vector of all ones. In that case we simply write $\omega(G)$, $\chi(G)$, etc. Following Grötschel, Lovasz and Schrijver, we associte to each graph two complementary linear spaces of symmetric matrices:

$$\begin{aligned} \mathcal{M} &= \{X = (x_{ij}) \mid X \text{ is symmetric, and } x_{ij} = 0 \\ &\quad \text{for all } i, j \text{ adjacent in } G \text{ or } i = j\}, \end{aligned}$$

$$\begin{aligned} \mathcal{M}^{\perp} &= \{Y = (y_{ij}) \mid Y \text{ is symmetric, } y_{ij} = 0 \\ &\quad \text{for all } i, j \text{ non adjacent in } G \text{ and } i \neq j\}. \end{aligned}$$

Obviously $\dim(\mathcal{M}^{\perp}) = n + e$. Let $m = \dim(\mathcal{M})$; then $m = \frac{n(n-1)}{2} - e$. We may define an *inner product* on matrices just like the inner product in vectors:

$$A \bullet B := \sum_{ij} A_{ij} B_{ij} = \mathrm{trace}(A^T B).$$

Since we will be dealing with symmetric matrices, we drop the superscript T.

Also, define $\Lambda(X)$ to be the largest eigenvalue of the symmetric matrix X. Λ is a convex, but generally non-smooth function on the set of symmetric matrices. In addition, the notations $A \succ 0$ and $A \succeq 0$ mean A is positive definite and A is positive semi-definite, respectively.

The *weighted Lovasz number* of a graph G is defined as:

$$\theta(G, \mathbf{w}) = \min\{\Lambda(X + W) \mid X \in \mathcal{M}\}. \tag{2.1}$$

Lovasz has shown that for all $\mathbf{w}$:

$$\omega(G, \mathbf{w}) \leq \theta(G, \mathbf{w}) \leq \chi(G, \mathbf{w}). \tag{2.2}$$

For a quick proof see [15]; for a more thorough treatment consult [9]. In case of perfect graphs, equality holds in 2.2, (in fact, equality is a necessary and sufficient condition for perfectness of a graph); in particular, $\theta(G, \mathbf{w})$ is an integer.

There is a dual way of defining $\theta(G, \mathbf{w})$:

$$\begin{aligned} \theta(G, \mathbf{w}) &= \max\{W \bullet Y | Y \in \mathcal{M}^{\perp},\ Y \succeq 0, \\ &\qquad \text{and } \mathrm{trace} Y = 1\}. \end{aligned} \tag{2.3}$$

Notice that the set of positive semi-definite matrices is a convex cone, so is the intersection of this cone with the linear space $\mathcal{M}^{\perp}$. The condition $\mathrm{trace} Y = 1$ makes the feasible set in 2.3 compact. For a proof that 2.1 and 2.3 are equivalent, and for several other definitions of Lovasz number see [13] and [9]. Also, Fletcher [2] and Overton [19] prove more general results, and their treatment is mostly in the context of duality theory in convex programming. In addition, we should mention that Lovasz has shown a "complementary slackness" relation:

THEOREM 2.1. *Y^* is an optimal solution of (2.3) and X^* an optimal solution of (2.1) if and only if*

$$Y^*[\theta(G, \mathbf{w})I - X^* - W] = 0. \tag{2.4}$$

For further details consult [13], [7], [8] and [9]. See also [19] for a convex programming perspective.

It is well-known that $\omega(G, \mathbf{w})$ and $\chi(G, \mathbf{w})$ are NP-hard to compute in general graphs. Yet, as Grötschel, Lovasz, and Schrijver observed $\theta(G, \mathbf{w})$ is polynomial-time computable by virtue of the ellipsoid method for convex programming. However, as in linear programming, one may use interior point techniques to obtain a more practical algorithm which is also suitable for parallelization.

We now formulate the *logarithmic barrier approach* for solving 2.1. Let $\mathbf{x}$ be an m-vector whose entries are those entries of X that are not fixed at 0, $\mathbf{x}'$ be the (m+1)-vector $(z, \mathbf{x}^T)^T$, and

$$A(\mathbf{x}) := X + W.$$

Also, $\Lambda(\mathbf{x}) := \Lambda(A(\mathbf{x}))$. Let $z = \theta(G, \mathbf{w})$. Then treating z as a new variable we may transform the non-smooth optimization problem 2.1 to the following constrained problem:

$$\begin{array}{lll} (2.5) & \min & z \\ & \text{s.t.} & z - \lambda_i(\mathbf{x}) \geq 0 \text{ for } i = 1, \cdots, n, \end{array}$$

where $\lambda_i(\mathbf{x})$ are the eigenvalues of $A(\mathbf{x})$ with some arbitrary but fixed order imposed on them.

Notice that the feasible region of 2.5 is the set of all $\mathbf{x}'$ making the matrix $zI - A(\mathbf{x})$ positive semi-definite. This region is a convex cone. The relative interior of this cone is the set of positive-definite matrices $zI - A(\mathbf{x}) \succ 0$, while its boundary is the set of singular semi-definite matrices of the cone. The idea behind the logarithmic barrier approach is to associate to each point $\mathbf{x}'$ inside the relative interior of the feasible region of 2.5 the function

$$-\sum_{i=1}^{n} \ln[z - \lambda_i(\mathbf{x})].$$

This function is defined only on the positive definite set $zI - A(\mathbf{x}) \succ 0$ and tends to infinity as $\mathbf{x}'$ approaches boundary of the cone (that is when $zI - A(\mathbf{x})$ becomes singular). With this barrier function we may choose a variety of approaches similar to linear programming–such as various potential function reduction methods–to solve 2.5. Here we will follow the classical logarithmic barrier method (as studied in Fiacco and McCormick [1]), with the Newton method applied to the barrier function. Define:

$$\begin{array}{llll} (2.6) & f_\mu(z, \mathbf{x}) & := & z - \mu \sum_{i=1}^{n} \ln[z - \lambda_i(\mathbf{x})] \\ & & = & z - \mu \ln \det[zI - A(\mathbf{x})] \\ & & = & z - \mu \ln P(z, \mathbf{x}), \end{array}$$

where $P(z, \mathbf{x})$ is the characterisitic polynomial of the matrix $A(\mathbf{x})$; it is a degree n polynomial in z with coefficients symmetric multinomials in $\mathbf{x}$. Since the constraint set in 2.5 is a convex cone it follows that f_μ is a convex function, in fact, for $\mu > 0$ it is strictly convex and so has a unique minimum in the interior of the feasible region of 2.5. Call z_μ, $\mathbf{x}_\mu$, $\mathbf{x}'_\mu$ and $A_\mu := A(\mathbf{x}_\mu)$, *$\mu$-optimal* when f_μ attains its minimum at these points. Then the theory of barrier functions indicates that as μ tends to zero, $\mathbf{x}'_\mu$ tends to the optimal solution of 2.5, in particular, z_μ tends to the weighted Lovasz number of G.

3 Optimality conditions and primal and dual paths of optimizers.

In this section we will state the optimality conditions for f_μ. It will be convenient to define the matrix $B(z, \mathbf{x})$:

$$(3.7) \qquad B(z, \mathbf{x}) := [zI - A(\mathbf{x})]^{-1}$$

The function f_μ has $m + 1$ variables, and so, there are $m + 1$ first order conditions as follows:

$$(3.8) \qquad \frac{\partial f_\mu}{\partial z} = 0,$$

$$(3.9) \qquad \frac{\partial f_\mu}{\partial x_{ij}} = 0 \text{ for all } i, j \notin E.$$

But,

$$\begin{array}{llll} & \frac{\partial f_\mu}{\partial z} & = & 1 - \mu \frac{P'(z, \mathbf{x})}{P(z, \mathbf{x})} \\ & & = & 1 - \mu [\frac{1}{z - \lambda_1(\mathbf{x})} + \cdots + \frac{1}{z - \lambda_n(\mathbf{x})}] \\ (3.10) & & = & 1 - \mu \text{trace} B(z, \mathbf{x}). \end{array}$$

Also, by expanding $\det[zI - A(\mathbf{x})]$ along row i, and arranging terms that involve x_{ij}, we will get:

$$\begin{array}{llll} & \frac{\partial f_\mu}{\partial x_{ij}} & = & -\mu \frac{\partial \det[zI - A(\mathbf{x})]/\partial x_{ij}}{\det[zI - A(\mathbf{x})]} \\ & & = & -\mu \frac{-2(-1)^{i+j} \det_{ij}[zI - A(\mathbf{x})]}{\det[zI - A(\mathbf{x})]} \\ (3.11) & & = & 2\mu [B(z, \mathbf{x})]_{ij}, \end{array}$$

where $\det_{ij} A$ is the minor of A after removing row i and column j. Thus z and $\mathbf{x}$ are μ-optimal if and only if:

$$(3.12) \qquad 1 - \mu \text{ trace} B(z, \mathbf{x}) = 0, \text{ and}$$

$$(3.13) \quad 2\mu \, [B(z, \mathbf{x})]_{ij} = 0 \text{ for all } i, j \notin E \text{ and } i \neq j.$$

Some observations are in order. If z_μ, $\mathbf{x}_\mu$, A_μ and $B_\mu := B(z_\mu, \mathbf{x}_\mu)$ are μ-optimal, then optimality conditions 3.13 indicate that $B_\mu \in \mathcal{M}^\perp$. Also, since $z_\mu I - A_\mu$ is positive definite, so is B_μ. Therefore, B_μ satisfies almost all feasibility conditions for the dual problem 2.3, except its trace may not be one. However, from the other optimality condition 3.12 we know that $\text{trace} B_\mu = 1/\mu$. Therefore we get:

THEOREM 3.1. *Let z_μ, $\mathbf{x}_\mu$ and A_μ be μ-optimal for (2.1). Then the matrix μB_μ exists and is in the relative interior of the feasible region of the dual problem 2.3.*

Thus for each primal μ-optimal point $\mathbf{x}'_\mu$, we have a dual feasible point μB_μ. Clearly, z_μ is an upper bound for the Lovasz number $\theta(G, \mathbf{w})$ and $W \bullet (\mu B_\mu)$ is a lower bound. The next theorem gives the size of the duality gap:

THEOREM 3.2. *For μ-optimal solutions we have:*

$$z_\mu - W \bullet (\mu B_\mu) = \mu n. \tag{3.14}$$

Proof. Since $\text{trace}(\mu B_\mu) = 1$ and $\text{trace} B_\mu X_\mu = 0$ (because B_μ and X_μ are complementary), we have:

$$\begin{aligned} & z_\mu - W \bullet (\mu B_\mu) \\ &= z_\mu \text{trace}(\mu B_\mu) - \text{trace}(\mu B_\mu)W - \mu \text{trace} B_\mu X_\mu \\ &= \mu \text{trace}[B_\mu(z_\mu I - X_\mu - W)] \\ &= \mu \text{trace} I \\ &= \mu n. \end{aligned}$$

We may think of theorem 3.2 as a μ version of the "complementary slackness" theorem 2.1. (compare theorem 2.1 with the relation $(\mu B_\mu)(z_\mu I - A_\mu) = \mu I$.)

COROLLARY 3.1. *If the graph G is perfect then for $\mu < 1/n$,*

$$\theta(G, \mathbf{w}) = \lfloor z_\mu \rfloor = \lceil W \bullet (\mu B_\mu) \rceil. \tag{3.15}$$

Proof. Immediate, after noticing that the duality gap is less than 1 and $\theta(G, \mathbf{w})$ is an integer.

Observe that as one changes μ continuously the μ-optimal point $\mathbf{x}'_\mu$ traverses a smooth path through the relative interior of the convex cone $[zI - A(\mathbf{x})] \succeq 0$. The results above indicate that this path induces a path of maximizers μB_μ in the dual feasible region 2.3. Furthermore, as μ tends to zero, the gap between primal objective function, z_μ, and the dual objective function, $W \bullet (\mu B_\mu)$, also tends to zero; in fact, in the case of perfect graphs, as soon as there is only one integer between these values, we may stop and report that integer as the size of the maximum clique.

4 A sketch of the sublinear algorithm.

In this section we will outline an iterative algorithm for solving 2.1 up to ϵ relative accuracy so that the number of iterations is bounded by $O^*(\sqrt{n} \log 1/\epsilon)$. The algorithm will consist of computing the Newton direction for minimizing f_μ. Therefore, first we derive expressions for this direction.

Let $\mathbf{g}_\mu(z, \mathbf{x})$ be the gradient of f_μ; $\mathbf{g}_\mu$ is an $(m+1)$-vector indexed by z and by entries of $\mathbf{x}$, that is by x_{ij} where $i, j \notin E$. We may determine the entries of $\mathbf{g}_\mu$ by 3.10 and 3.11. To find the Newton direction, we need to compute the Hessian of f_μ: $\mathbf{H}_\mu(z, \mathbf{x}) := \nabla^2 f_\mu(z, \mathbf{x})$. $\mathbf{H}_\mu$ is the $(m+1) \times (m+1)$ matrix of the second partial derivatives of f_μ. Let us derive expressions for these partial derivatives. First, observe that taking derivatives with respect to z in the relation $B(z, \mathbf{x})[zI - A(\mathbf{x})] = I$, we get:

$$\frac{\partial B}{\partial z}[zI - A(\mathbf{x})] + B = 0$$

and so,

$$\frac{\partial B}{\partial z} = -B^2. \tag{4.16}$$

Again differentiating $B(z, \mathbf{x})[zI - A(\mathbf{x})] = I$, but this time with respect to x_{ij} we get:

$$\frac{\partial B}{\partial x_{ij}}[zI - A(\mathbf{x})] + B \frac{\partial}{\partial x_{ij}}[zI - A(\mathbf{x})] = 0.$$

But,

$$\frac{\partial}{\partial x_{ij}}[zI - A(\mathbf{x})] = -\mathbf{e}_i \mathbf{e}_j^T - \mathbf{e}_j \mathbf{e}_i^T,$$

where $\mathbf{e}_i$ is the i^{th} unit vector. The last relation is true because $zI - A(\mathbf{x})$ is symmetric and each variable x_{ij} occurs both at entries i, j and j, i. Therefore, we get:

$$\frac{\partial B}{\partial x_{ij}} = B \mathbf{e}_i \mathbf{e}_j^T B + B \mathbf{e}_j \mathbf{e}_i^T B \tag{4.17}$$

Using 3.10 and 4.16 we get

$$\begin{aligned} \frac{\partial^2 f_\mu}{\partial z^2} &= \frac{\partial}{\partial z}(1 - \mu \, \text{trace} B) \\ &= -\mu \, \text{trace} \frac{\partial B}{\partial z} \\ &= \mu \, \text{trace} B^2. \end{aligned} \tag{4.18}$$

Also, using 3.11 and 4.16 we get

$$\begin{aligned} \frac{\partial^2 f_\mu}{\partial z \partial x_{ij}} &= 2\mu \, \mathbf{e}_i^T \frac{\partial B}{\partial z} \mathbf{e}_j \\ &= -2\mu \, [B^2]_{ij}. \end{aligned} \tag{4.19}$$

And using 3.11 and 4.17 we get

$$\begin{aligned} \frac{\partial^2 f_\mu}{\partial x_{ij} \partial x_{kl}} &= \frac{\partial}{\partial x_{kl}}(2\mu \, \mathbf{e}_i^T B \mathbf{e}_j) \\ &= 2\mu \, \mathbf{e}_i^T (\frac{\partial B}{\partial x_{kl}}) \mathbf{e}_j \\ &= 2\mu \, \mathbf{e}_i^T (B \mathbf{e}_k \mathbf{e}_l^T B + B \mathbf{e}_l \mathbf{e}_k^T B) \mathbf{e}_j \\ &= 2\mu \, (B_{ik} B_{jl} + B_{il} B_{jk}) \end{aligned} \tag{4.20}$$

In summary, the Hessian of f_μ is:

$$\mathbf{H}_\mu(z, \mathbf{x}) = \tag{4.21}$$

$$\begin{pmatrix} \mu \text{trace} B^2 & \cdots & -2\mu[B^2]_{kl} & \cdots \\ \vdots & & \vdots & \\ -2\mu[B^2]_{ij} & \cdots & 2\mu \, (B_{ik} B_{jl} + B_{il} B_{jk}) & \cdots \\ \vdots & & \vdots & \end{pmatrix}.$$

Therefore, we get the Newton direction:

$$\mathbf{n}_\mu(z, \mathbf{x}) := \mathbf{H}_\mu^{-1}(z, \mathbf{x}) \mathbf{g}_\mu(z, \mathbf{x}). \tag{4.22}$$

.

We are now ready to state the Newton algorithm as studied by Nesterov and Nemirovsky in [17]. These authors have developed a general theory for polynomial-time methods for convex constrained optimization problems. They have derived sufficient conditions for barrier functions so that when Newton method is applied to them one would get a polynomial time algorithm, with the number of iterations bounded by $O^*(\sqrt{m}\log 1/\epsilon)$ (if the number of inequality constraints is m and we want relative accuracy of ϵ.) They call functions satisfying these sufficient conditions *m-self-concordant barriers*. Furthermore, Nesterov and Nemirovsky show that the barrier function $-\ln\det X$ is n-self-concordant for the cone of symmetric, positive semi-definite $n\times n$ matrices [17]. Here we only give an outline of their algorithm as specialized to our problem. For details refer to the book by Nesterov and Nemirovsky.

We start with an initial $A(\mathbf{x}_0) = X_0 + W$, and z_0 so that $z_0 I - A(\mathbf{x}_0)$ is positive-definite. (For instance, we could choose $X_0 = 0$ and $z_0 = \Lambda(X_0 + W) + 0.5$.) The algorithm is in two stages. In the first stage we want to get to a point sufficiently close to the path of minimizers. Let $\mathbf{b}(z,\mathbf{x})$ be the gradient of the barrier $-\ln\det[zI - A(\mathbf{x})]$, and $\mathbf{H}(z,\mathbf{x})$ its Hessian; both may be computed similar to 3.10, 3.11 and 4.21. Define:

$$f_k^0(z,\mathbf{x}) := (\mathbf{x}' - \mathbf{x}_0')^T\mathbf{b}(z_0,\mathbf{x}_0) - \mu_k \ln\,\det[zI - A(\mathbf{x})]$$

Clearly, the Newton direction for this function is given by

$$(4.23)\mathbf{n}_k^0(z,\mathbf{x}_0) := \mu_k[\mathbf{b}(z_0,\mathbf{x}_0) + \mu_k\mathbf{b}(z,\mathbf{x})]\mathbf{H}^{-1}(z,\mathbf{x}).$$

Set

$$(4.24) \qquad \mu_0 = \exp\left(-\frac{0.136}{0.193+\sqrt{n}}\right)$$

and define

$$c_{k+1} = \sqrt{\mathbf{b}^T(z_k,\mathbf{x}_k)\mathbf{H}^{-1}(z_k,\mathbf{x}_k)\mathbf{b}(z_k,\mathbf{x}_k)}$$

Notice that c_{k+1} is by a constant factor the square root of the amount that f_k^0 is decreased after one iteration of the Newton method. The first stage is as follows:

For $k = 0, 1, 2, \cdots$ repeat:

$$\begin{aligned} \mu_{k+1} &= \mu_k/\mu_0 \\ \mathbf{x}_{k+1}' &= \mathbf{x}_k' + \mathbf{n}_{k+1}^0(z_k,\mathbf{x}_k) \end{aligned}$$

until $c_{k+1} \le 0.150$.

The second stage consists of following the path of minimizers. we take the last $\mathbf{x}_k'$ returned by the first stage as our starting point and rename it to $\mathbf{x}_1'$. also set

$$\mu_1 = \frac{\operatorname{trace} B^2(z_0,\mathbf{x}_0)}{(0.193 - c)}\mu_0,$$

where μ_0 is as defined in 4.24, and c is the last c_{k+1} returned by stage one. Stage two is as follows ($\mathbf{n}_k$ is the same as $\mathbf{n}_{\mu_k}$ and is given by 4.22):

For $k = 1, 2, \cdots$ repeat:

$$\begin{aligned} \mu_{k+1} &= \mu_k\mu_0 \\ \mathbf{x}_{k+1}' &= \mathbf{x}_k' + \mathbf{n}_{k+1}(z_k,\mathbf{x}_k) \end{aligned}$$

until relative accuracy ϵ is achieved.

Basically, The strategy of the two stage algorithm is to reduce μ_k to $(1-\sigma)\mu_k$ where σ is some number between 0 and 1. σ must be sufficiently small so that if we start at point $\mathbf{x}_k'$ and apply one iteration of the Newton method, then the new point would not be too far from the path. At the same time it must be large enough so that we get to the solution fast. It turns out that–as in linear programming–choosing $\mu_0 = 1 - \sigma = 1 - O(\frac{1}{\sqrt{n}})$ will result in sublinear number of iterations. This is the best choice known at the time of writing.

Nesterov and Nemirovsky scrupulously analyze this two stage algorithm and confirm that with the constants chosen as prescribed, we have

Theorem 4.1. *Total number of iterations for the two-stage algorithm presented above is bounded by $O^*(\sqrt{n}\log 1/\epsilon)$.*

This theorem is a specialized version of a general result for the class of self-concordant barrier functions [17].

In case of perfect graphs, since $\theta(G,\mathbf{w}) = \omega(G,\mathbf{w})$ is an integer, we need to get the Lovasz number only up to the closest integer, then rounding will give us the exact answer. Since $\omega(G,\mathbf{w})$ could be as large as $C := \sum_{i=1}^n w_i$, we need to set $\epsilon = 1/2C$. Thus the parallel time complexity of computing $\omega(G,\mathbf{w})$ is $O^*(\sqrt{n}\log C)$. In particular, for unweighted case the parallel time complexity of computing $\omega(G)$ is $O^*(\sqrt{n})$.

5 A Las Vegas randomized scheme for construction of maximum cliques.

So far we have stated how to compute the size of a largest clique (or independent set) in a perfect graph. Suppose that the maximum clique were unique. Then it is a trivial matter to construct it in parallel using the size oracle developed in the previous section. We remove one of the vertices and ask the oracle about the size of the maximum clique in the remaining graph; the removed vertex is in the unique maximum clique if and only if the size returned by the oracle decreases. Doing this for each vertex simultaneously, we construct the maximum clique in parallel .

When we do not have uniqueness, we may use the randomized perturbation scheme of Mulmuley, Vazirani and Vazirani, [16]. First recall their isolating lemma:

LEMMA 5.1. *Let $S = \{x_1, \cdots, x_n\}$ and F a family of subsets of S, that is $F = \{S_1, \cdots, S_N\}$. Further, let elements of S be assigned integer weights chosen uniformly and independently from $[1, 2n]$. Then,*

$$\Pr[\text{There is a unique maximum weight set in } F] \geq \frac{1}{2}.$$

See [16] for proof.

To get an (unweighted) maximum clique in a perfect graph we follow a procedure similar to the one adopted by Mulmuley, Vazirani and Vazirani for constructing the minimum weighted perfect matching in bipartite graphs. The idea is to assign weights to vertices randomly so that with high probability the maximum weighted clique is unique, but at the same time, this clique is among the maximum cliques before assigning weights.

First give a weight of $2n^2$ to each vertex i so that the weight of maximum weighted cliques is at least $2n^2$ more than the next largest clique weight. Now perturb weight of each vertex i by adding integer u_i uniformly and independently chosen from $[1, 2n]$. So now each vertex has weight $\mathbf{w}_i = 2n^2 + u_i$. Notice that if a clique was not maximum before, then it is impossible for it to become maximum after assigning weights. Therefore, the maximum weighted clique is among one of the unweighted maximum cliques. The isolating lemma implies that this clique is unique with a probability at least $1/2$ and we may use the scheme mentioned at the beginning of this section to find it in parallel.

We should mention that this scheme, in fact, results in a Las Vegas type randomized algorithm. No randomization is involved in computing the size oracle, $\omega(G, \mathbf{w})$; only constructing a maximum clique involves probabilistic choices. If the weights generated do not result in a unique maximum weighted clique, the scheme mentioned at the beginning of this section may return a set which is not a maximum clique. This can be checked in parallel and the algorithm will return a message of failure; any set returned by the algorithm will be a genuine maximum clique with no possibility of error.

Finally, there is an easier way of constructing the maximum clique when we know it is unique. We may extract it from a sufficiently close approximation of B^*, the limit of μ-optimal matrix μB_μ as μ tends to 0. Set

$$Y^* := \operatorname{diag}(\mathbf{w}^{1/2}) B^* \operatorname{diag}(\mathbf{w}^{1/2}),$$

and define $f_i = \sum_{j=1}^n Y^*_{ij}$. Then Lovasz in a survey paper shows that the n-vector $\mathbf{u}$ defined as:

$$(5.25) \qquad \mathbf{u}_i = \begin{cases} f_i^2/(\theta(G,\mathbf{w})Y^*_{ii}) & \text{if } Y^*_{ii} \neq 0 \\ 0 & \text{otherwise,} \end{cases}$$

will be the characteristic vector of the maximum clique [14]. (If the maximum clique is not unique, then this vector will lie in the convex hull of the set of maximum cliques.) Therefore, if $k = \theta(G, \mathbf{w})$, then we simply look at the dual solution B_μ and include vertex i in the maximum clique if $(B_\mu)_{ii}$ is among the k largest entries in the diagonal.

6 Practical implementations.

The ideas presented so far may be refined to achieve a practical (sequential or parallel) algorithm for computing maximum cliques and independent sets. If m, the number of variables is relatively small, that is about $O(n)$, then we may use 4.22 to compute the Newton iteration in about $O(n^3)$. If the graph is sparse, that is the number of edges is $O(n)$, we can work on the dual problem (2.3) and get similar results. However, when both G and its complement are relatively dense, that is when the number of edges is approximately equal to the number of missing edges, then the Hessian matrix 4.21 will be an $O(n^2 \times n^2)$ matrix and finding the Newton direction will require $O(n^6)$ operations. In this case, one may use other techniques of mathematical programming. One that we have actually implemented is based on the so-called *limited memory quasi-Newton method* with a scaling as explained in Gill, Murray, and Wright [4]. In this scheme, the amount of work per iteration is reduced from $O(n^3 + m^3)$ to $O(n^3 + m)$ at the expense of somewhat slower convergence than the pure Newton method. We have obtained Lovasz number of graphs of up to 200 vertices and more than 11,000 variables by using this method and the number of iterations we observed were typically a fraction of m (usually they were fairly close to n); the total time spent was up to 5 minutes for the largest graphs on a Cray 2 machine.

It is also possible to dispense with the barrier method altogether and investigate other techniques of mathematical programming. In fact, starting from Fletcher [2], several methods based on sequential quadratic programming have been proposed for general problems with the cone of positive semi-definite matrices as their constraint set, see for example, the papers [3], [18] and [19]. It should be mentioned that these techniques are meant to compute the optimum solution very accurately. However, no computational complexity results are given in these works, and in fact, the worst case complexity of these techniques, like that of the simplex method for linear programming, may be much worse than the observed behavior.

Acknowledgement

I am greatly indebted to Professor Ben Rosen for his support and patience. I learned a great deal about the problem studied here through e-mail conversations with Michael Overton. Also, I would like to thank Dan Boley, Yin Yu Ye, Yurii Nesterov and Av Vadii Nemirovsky for instructive conversations, especially Yin Yu Ye who brought the work of Nesterov and Nemirovsky to my attention.

References

[1] A. Fiaco and G. McCormick, *Nonlinear Programming, Sequential Unconstrained Minimization Techniques*, Research Analysis Corporation, 1968, New edition: SIAM 1990.

[2] R. Fletcher, Semi-Definite Matrix Constraints in Optimization, *SIAM Journal on Control and Optimization*, 23, July 1985.

[3] S. Friedland, J. Nocedal, and M. Overton, The Formulation and Analysis of Numerical Methods for Inverse Eigenvalue Problems, *SIAM Journal on Numerical Analysis*, 24(3), June 1987.

[4] P. Gill, W. Murray, and M. Wright, *Practical Optimization*, Academic Press, 1981.

[5] A. Goldberg, S. Plotkin, D. Shmoys, and E. Tardos, Interior-Point Methods in Parallel Computation (Extended Abstract), Manuscript, April 1989.

[6] M. Grötschel, L. Lovasz, and A. Schrijver, The Ellipsoid Method and its Consequences in Combinatorial Optimization, *Combinatorica*, 1(2) (1981).

[7] ———, Relaxations of Vertex Packing, *Journal of Combinatorial Theory*, Series B 40, 1986.

[8] ———, Polynomial Algorithms for Perfect Graphs, *Annals of Discrete Mathematics* 21(1984).

[9] ———, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, 1988

[10] H. Karloff, A Las Vegas RNC Algorithm for Maximum Matching, *Combinatorica* 6(4) 1986.

[11] R. Karp, E. Upfal, A. Widgerson, Constructing A perfect Matching is in Random NC, *Combinatorica*, 6(1) 1986.

[12] L. Lovasz, Normal Hypergraphs and the Weak Perfect Graph Conjecture, *Discrete Math* 2 (1972).

[13] ———, On the Shannon Capacity of a Graph, *IEEE Transactions on Information Theory*, Vol. IT-25, no. 1, January 1979.

[14] ———, Perfect Graphs, in *selected Topics in Graph Theory 2*, edited by L. Beineke, and R. Wilson. Academic Press, 1983.

[15] ———, *An Algorithmic Theory of Numbers, Graphs and Convexity*, CBMS-NSF50, SIAM, 1986.

[16] K. Mulmuley, U. Vazirani, V. Vazirani, Matching is as Easy as Matrix Inversion, *Combinatorica* 7(1) 1987.

[17] Y. Nesterov, A. Nemirovsky, *Self-Concordant Functions and Polynomial Time Methods in Convex Programming*, expanded edition, Moscow, April 1990.

[18] M. Overton, On minimizing the Maximum Eigenvalue of a Symmetric Matrix, *SIAM Journal on Matrix Analysis and Applications*, vol. 9, no. 2, April 1988.

[19] ———, Large-Scale Optimization of Eigenvalues, Manuscript, NYU Computer Science Department report no. 505, May 1990.

[20] Y. Ye, An $O(n^3L)$ Potential Reduction Algorithm for Linear Programming, Manuscript, The University of Iowa, 1989.