On the Parallel Approximability of Some Classes of Quadratic Programming

Maria Serna* Fatos Xhafa*

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya, Pau Gargallo 5
08028-Barcelona, Spain
Email: {mjserna, fatos}@goliat.upc.es

Abstract

In this paper we analyze the parallel approximability of two special classes of Quadratic Programming. First, we consider Convex Quadratic Programming . We show that the problem of Approximating Convex Quadratic Programming is P-complete. We also consider two approximation problems related to it, *Solution Approximation* and *Value Approximation* and show both of these cannot be solved in NC, unless P=NC. Secondly, on the positive side, we show that we have an NC Approximation Scheme for those instances of Quadratic Programming that are "smooth" and "positive." Then we show how to extend the result for positive instances of bounded degree Smooth Integer Programming problems. Finally, we formulate several combinatorial problems as positive QP (or positive Integer Programs) in packing/covering form and show that the presented techniques can be used to obtain NC Approximation Schemes for "dense" instances of such problems.

1 Introduction

Quadratic Programming (QP) represents a special class of Nonlinear Programming in which the objective function is quadratic and the constraints are linear. Quadratic Programs arise in a wide variety of applications including regression analysis in statistics, economic models of optimal sales revenues, etc. The sequential complexity of QP is well known. QP, in its general form, is NP-complete. Sahni [18] showed that QP is NP-hard and later on Vavasis [20] proved that QP is in the class NP.

In this paper we study the parallel approximability of two special classes of QP. Recall, the class NC consists of problems that can be solved by a PRAM algorithm whose running time is polylogarithmic in instance size while using a polynomial number of processors. Thus a problem that is shown to be P-complete hardly will belong to NC [5]. For those hard problems it is interesting to analyze whether we can get an approximate algorithm in NC. Problems easy to approximate belong to NCAS, the class of problems that have an NC Approximation Scheme, i.e., a family of NC algorithms $\{A_\varepsilon\}_{\varepsilon\geq 0}$, such that A_ε has approximation ratio $1+\varepsilon$, and whose running time depends arbitrarily on ε. Whereas, non-approximable problems are those that no NC algorithm can compute a

*This research was supported by the ESPRIT BRA Program of the European Community under contract No. 7141, project ALCOM II.

constant approximation unless P=NC (see, e.g., [3]). Usually the last behavior is proved by showing that adequate approximation problems are P-complete. First, we consider *Convex Quadratic Programming* (CQP) that is, the Quadratic Programming whose objective function is convex. CQP is solved sequentially in polynomial time, through the ellipsoid method [6] or the interior point algorithm [7]. We show that the problem CQP is non-approximable. To prove the results we make use of the fact that Approximating Linear Programming is P-complete [19, 13]. We also consider two approximation problems related to CQP. *Solution Approximation* consists in finding a feasible solution to CQP whose norm is close to that of an optimal solution. *Value Approximation* is to find a vector x such that the value of the objective function on it is close to the optimal value, without the condition that x be feasible. Considering all possible values for the error parameter ε, we get two families of problems, one for each type of approximation. We show that the problems of both these families are P-complete, for any value of ε, such proving that CQP is hard to approximate.

Our second special class of QP is *Smooth Quadratic Programming* in which we impose restrictions on the magnitudes of all the coefficients appearing in the instance. We show that, given an instance of Smooth QP, if it is, in addition, *positive*, i.e., all coefficients are non-negative, then we can find an NC Approximation Scheme for the problem restricted to such instances. Recall that Luby and Nisan [10] gave a parallel approximation algorithm for Positive LP – the special case of LP in packing/covering form with all the coefficient of the instance being non-negative. We obtain the result by reducing the instance of QP to an instance of Positive Linear Programming, finding in NC an approximate fractional solution to the obtained program, and then rounding the fractional solution to an integer approximate solution for the original problem. The reduction from QP to LP uses the Random Sampling technique, and we show how to deterministically perform it in NC by using random walks in constant degree expanders. The rounding of the fractional solution involves the Randomized Rounding method of [17], and we show how to use the Lattice Approximation technique of [14] to deterministically find the integer solution in NC. Furthermore, we prove that the NC Approximation Scheme for Smooth QP can be used to obtain NC Approximation Schemes for bounded degree *Smooth Integer Programming*.

We should note, however, that the "positiveness" and "smoothness" conditions are restrictive, in the sense that it is difficult to find natural problems whose QP is at the same time positive and smooth. Fortunately, even with these conditions we are able to cast in our model several important combinatorial problems. These are MAX CUT, MAX DICUT, MAX-2-SAT and MAX-k-SAT. We also propose a new problem, that we call *Maximum Product Arrangement* (MaxPA) of a graph, that is defined analogously as that of Maximum Linear Arrangement. First, we formulate all of them as Positive QP or Positive Integer Programs in the packing/covering form, and then we derive NC Approximation Schemes for their dense instances. Notice that these programs are different from those given in [2]. Interestingly, for the problems on graphs, imposing in their QP programs the smoothness condition is equivalent with that of the graph instance being "dense." Our results for the problems mentioned above match those of the sequential case given in [2].

2 Definitions

In this section we give formal definitions of the problems used through the paper. Let Q be an $n \times n$ positive semidefinite matrix, A an $m \times n$ matrix of range m, $a \in \mathbb{R}$, $b \in \mathbb{R}^m$ and $c \in \mathbb{R}^n$. In all definitions, the entries of (Q,c,A,b) are assumed to be integers, unless otherwise stated.

Linear Programming problem (LP) is

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq b \end{array}$$

Convex Quadratic Programming problem (CQP) is

$$\begin{array}{ll} \text{minimize} & f(x) = 1/2\, x^T Q x + c^T x + a \\ \text{subject to} & Ax \leq b \end{array}$$

Smooth Integer Programming [2]. Let $x = (x_i)_{i=1}^n$. A Smooth degree-l Integer Program has the form

$$\begin{array}{ll} \text{minimize} & p(x_1, \ldots, x_n) \\ \text{subject to} & Wx \leq d \quad \text{with} \quad x_i \in \{0,1\}, \quad 1 \leq i \leq n \end{array} \tag{1}$$

where $p(x_1, \ldots, x_n)$ is a degree-l polynomial in which the coefficient of degree-i term is either 0 or $\Theta(n^{l-i})$.

Once we have a programming problem, any of the types defined above, the natural way to define an approximation problem related to it is to relax the problem and ask for a feasible solution for which the value of the objective function is close to the optimum value. Let x^* be an optimal solution, given $\varepsilon \in (0,1]$, the ε-*approximation* problem consists in finding a feasible solution x such that if f is the objective function then

$$\frac{1}{\varepsilon} f(x^*) \geq f(x) \geq f(x^*).$$

Following [19] we also consider an approximation in the space of solutions, the so called **Solution Approximation Problem** in which we ask for a feasible solution x for which

$$||x^*||_p \geq ||x||_p \geq \varepsilon ||x^*||_p$$

where $||x||_p = (\sum_{i=1}^n x_i^p)^{1/p}$. The last approximation problem considered seeks for an approximation of the optimal value, we call it **Value Approximation Problem**, and consists in finding a rational n-vector x such that

$$\frac{1}{\varepsilon} f(x^*) \geq f(x) \geq f(x^*).$$

For a given problem Π, we will denote by ε-Π (resp. ε-ΠS, ε-ΠV) the corresponding approximation problems defined above. Let ε-LP, ε-LPS and ε-LPV be the corresponding problems to LP, for a given value of ε. The following result was shown in [19].

Theorem 1 *ε-LPS and ε-LPV are P-complete for any $\varepsilon \in (0,1]$.*

3 Approximating Convex Quadratic Programming

In this section we prove that the corresponding ε-CQP, ε-CQPS and ε-CQPV are also P-complete. Let us start by showing that CQP cannot be approximated in NC, unless P=NC.

Theorem 2 *The ε-approximation problem to Convex Quadratic Programming (ε-CQP) is P-complete for any $\varepsilon \in (0,1]$.*

Proof. Firstly, we show that there is an NC reduction from LP to CQP. The reduction goes as follows. Suppose we have an instance of LP:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq b \end{array}$$

We construct an instance of CQP as follows:

$$\begin{array}{ll} \text{minimize} & f(x) = c^T x + q(x) \\ \text{subject to} & Ax \leq b \end{array}$$

where $q(x) = 1/2\, ||Ax - b||^2 = 1/2\, x^T A^T Ax - x^T A^T b + 1/2\, b^T b$. Note that $q(x) \geq 0$ for any x and that $q(x) = 0$ for all x with $Ax = b$. Therefore, $q(x)$ as a quadratic form is positive semidefinite.

It holds that if x^* is an optimal solution to LP, then x^* is also an optimal solution for CQP and vice-versa. For that, first recall that an optimal solution to LP is a vertex of the convex polyhedron of feasible solutions, i.e., a point satisfying $Ax = b$. Further, we notice that the quadratic form $q(x)$ achieves its minimum in solutions x with $Ax = b$ and that its minimum value is zero. Therefore, for solutions x^* with $Ax^* = b$ we have

$$c^T x^* = f(x^*).$$

So any feasible solution x^* to $Ax \leq b$ that minimizes CQP also minimizes LP and vice-versa, furthermore it attains the same optimal value, therefore any approximation to the CQP problem will correspond to an approximation to LP. To conclude the proof just note that the reduction can be computed in NC. Since LP and the problem of approximating it are P-complete, the theorem follows. □

In order to show the non-approximability of ε-CQPS and ε-CQPV, we first consider a restricted version of linear programming denoted LPP, as follows:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^n x_i \\ \text{subject to} & Ax \leq b, \end{array} \tag{2}$$

whose optimal solutions have $x_i = 0$ or 1.

Let ε-LPPS and ε-LPPV be the corresponding approximation problems of LPP. In [19], the non-approximability of LP and of the problems ε-LPS and ε-LPV is shown by reducing Circuit Value Problem to LP. The instance of LP obtained by the reduction is in fact an instance of LPP, therefore both ε-LPPS and ε-LPPV are P-complete.

Based on the non-approximability of ε-LPPS and ε-LPPV, we also show that both Solution and Value Approximation to CQP are P-complete.

Theorem 3 *ε-CQPS and ε-CQPV are P-complete for any $\varepsilon \in (0, 1]$.*

Proof. Let us consider the CQP problem obtained from LPP as using the reduction given in Theorem 2. We can easily see that if x^* is the optimal solution (to both LPP and CQP) then we have

$$(||x^*||_p)^p = ||x^*||_1 = \sum_{i=1}^n x_i^*, \tag{3}$$

and

$$\sum_{i=1}^n x_i^* = f(x^*). \tag{4}$$

Now, from (3), (4) and Theorem 1 our theorem follows. □

4 Smooth Quadratic Programming

In this section we consider a restricted version of Quadratic Programming that we call Smooth Quadratic Programming (Smooth QP). Our definition for Smooth QP comes from the more general definition of [2] for Smooth Integer Programming.

Smooth Quadratic Programming (*Smooth QP*) is

$$\begin{aligned} &\text{minimize} \quad g(x) = \sum a_{ij} x_i x_j + \sum b_i x_i + c \\ &\text{subject to} \quad Wx \le d \quad \text{with} \quad x_i \in \{0,1\}, \quad 1 \le i \le n \end{aligned} \tag{5}$$

where a_{ij}, w_{ij} are $O(1)$, b_i is 0 or $\Theta(n)$, and c is 0 or $\Theta(n^2)$.

We show that, given an instance of Smooth QP, we can approximate it in NC if it is, in addition, positive. An instance of Smooth QP is called positive if all the coefficients of the instance are non-negative. We will show that for Positive Smooth QP we can get in NC an ε-approximation for any (constant) value of ε, thus the problem has an NC Approximation Scheme. As we will see this result can be extended easily for the general case, that is for positive instances of Smooth degree-l Integer Programming. To prove the result, we will make use of the following two lemmas. The first one is a well known technique on how to estimate the sum of n numbers by random sampling (see, e.g., [15]).

Lemma 1 (**Sampling Lemma**)
Let $\{a_i\}_i^n$ be a set of n-numbers, where each a_i is $O(1)$. Let $p = \sum_{i=1}^n a_i$ be their sum. If we pick uniformly at random a subset of $s = O(\log n/\varepsilon^2)$ of them and compute their sum q, then with high probability, i.e., with probability at least $1 - O(1/n)$, we have that $p - \varepsilon n \le qn/s \le p + \varepsilon n$.

The second lemma by Raghavan and Thompson shows how to round a fractional solution of a linear program to an integer solution of the same linear program.

Lemma 2 [17] (**Randomized Rounding**)
Let $x = (x_i)_{i=1}^n$ be a vector with $0 \le x_i \le 1$ that satisfies linear constraints $A_i x = b_i$, where each entry of row A_i is $O(1)$, $1 \le i \le n$. Construct $y = (y_i)_{i=1}^n$ randomly by setting $y_i = 1$ with probability x_i and 0 otherwise. Then, with high probability, $A_i y = b_i + O(\sqrt{n \log n})$.

We prove the following theorem.

Theorem 4 *Given a positive instance of Smooth QP and a fixed ε, we can find in NC a 0,1 assignment to variables x_i such that*

$$g(x_1, x_2, \ldots, x_n) \le g(x^*) + \varepsilon n^2$$

where $g(x)$ is the objective function and $g(x^*)$ is its optimal value.

For Theorem 4, it suffices to prove the following theorem.

Theorem 5 *Suppose there is a 0,1 solution to the following Positive Quadratic System*

$$\begin{aligned} &x^T A x + bx \le c \\ &Wx \le d \end{aligned} \tag{6}$$

where a_{ij}, $w_{ij} = O(1)$, $b_i = \Theta(n)$ and c, d_i are constants. Then for any fixed ε, we can find in NC an assignment of 0,1 values to x_i such that

$$x^T A x + bx \le c + \varepsilon n^2. \tag{7}$$

The proof closely follows a theorem from [2]. Here we observe that under our conditions, the 0,1 solution can be found also in NC. The idea is to reduce the QP instance to an instance of LP. Note that the resulting instance will be positive and thus an approximate algorithm for Positive Linear Programming is applied to find a fractional solution in NC. Then, we round the fractional solution to a 0,1 solution using the Randomized Rounding. Two sources of randomness appear in this scheme, first, the reduction we will describe in the next section uses itself randomization, so we have to show how to achieve it in NC. And, secondly, it appears when we apply the Randomized Rounding.

4.1 Reducing Positive Smooth QP to LP

The reduction is as follows. Suppose $x^{\#}$ is a solution of (6) and let $r^{\#} = x^{\#}A + b$. We can write (6) as

$$\begin{array}{l} x^T A + b \leq r^{\#} \\ r^{\#} x \leq c \\ Wx \leq d \\ 0 \leq x_i \leq 1 \end{array} \tag{8}$$

for which, $x^{\#}$ is also a feasible solution. Let x be its fractional solution. From Lemma 2, by randomly rounding this solution to a 0,1 solution y, we have that with high probability, i.e., probability at least $1 - O(1/n)$,

$$\begin{array}{l} y^T A_i + b_i \leq r_i^{\#} + O(\sqrt{n \log n}) \\ r^{\#} y \leq c + O(n) \cdot O(\sqrt{n \log n}) \\ W_i y \leq d + O(\sqrt{n \log n}). \end{array} \tag{9}$$

Note that the second inequality in (9) holds because $r_i^{\#} = \Theta(n)$. Consequently, we will have

$$\begin{array}{rcl} y^T A y + by & = & (y^T A + b)y \\ & \leq & (r^{\#} + O(\sqrt{n \log n}))y \\ & \leq & c + O(n^{3/2} \log n) \\ & \leq & c + \varepsilon n^2. \end{array} \tag{10}$$

We can write (8) if we knew the values $r_i^{\#}$. Instead, it is shown in [2] that using estimates r_i for them such that

$$|r_i^{\#} - r_i| < \varepsilon n \tag{11}$$

still (9) and (10) hold. To show that such estimates can be found in NC, we first prove that they can be found in RNC and later we prove that they can be found also deterministically in NC. To find $r = (r_i)_{i=1}^n$ in RNC we use the Sampling Lemma to produce $n^{O(1/\varepsilon^2)}$ estimates for $r^{\#}$. We can run in parallel $n^{O(1/\varepsilon^2)}$ positive linear programs and take as estimate that whose linear program has the best outcome. The values r_i are found as follows.

(1). Choose a set S of $k = O(\log n/\varepsilon^2)$ indices at random.

(2). In parallel, for each of $2^k = n^{O(1/\varepsilon^2)}$ possible assignments to variables with indices in S produce an estimate r of $r^{\#}$ by taking

$$r_i = b_i + \frac{n}{k} \sum_{j \in S} a_{ij} s_j, \quad 1 \leq i \leq n$$

where s_j is the value assigned to the j-th variable.

Since the assignments are found exhaustively, in one of the assignments generated above we have $s_j = x_j^*$, that is s_j is the j-th component of the optimal solution. In order to estimate $r_i^{\#}$ we only need to estimate $a_{ij}x_j^*$ since $r_i^{\#} = \sum a_{ij}x_j^* + b_i$ and b_i is a constant. Applying the Sampling Lemma on the set $\{a_{ij}x_j^*\}$, results in estimates that satisfy (11).

4.2 Derandomization

Now we show that both the Sampling Lemma and the Randomized Rounding can be done in NC. For the Sampling Lemma, we can use similar arguments to those of the sequential case [2]. Instead of choosing $k = O(\log n/\varepsilon^2)$ indices independently, it suffices to choose the vertices encountered in a random walk of length k on a constant degree expander. The number of such walks is $n^{O(1/\varepsilon^2)}$, i.e., polynomial in n and therefore it could be handled in NC. So, it remains to clear the following two points: first, how to construct in NC a constant degree expander, and second how is determined (simulated in NC) the random walk. Both these points have been extensively treated in [9, 12, 8, 1], and we give a simple explanation as indicated in [11]. A constant degree d expander on n nodes is an n node d-regular graph in which the number of neighbors of any set of vertices S is larger than some positive constant multiple of cardinality of S (see, e.g., [15, pages 108–112]). It is well known that the adjacency matrix of such an expander is a symmetric matrix whose greatest eigenvalue is d and the second largest one less than d. Such constant degree expanders can be explicitly constructed in NC for any n and d, even with the additional condition that the second greatest eigenvalue λ being less than $d^{9/10}$ [8]. Let $N = 2^r$, where $N = n^{O(1/\varepsilon^2)}$ and let us identify the set $\{0,1\}^r$ with the nodes of a d-regular N-node graph expander G, where d is set to some constant value such that $\lambda \leq d/4$. Then, we choose the nodes $y_1, \ldots, y_N$ as the nodes visited in a random walk of length k, with $k = O(\log n/\varepsilon^2)$, starting from a random node z. The random walk is determined by the starting node z and integers $i_j \in \{1, \ldots, d\}$, for $j \in \{1, \ldots, k\}$ describing which edge to use in the j-th step of the walk. Thus, y_1 is the i_1-th neighbor of z, and for $j \geq 2$, y_j is the i_j-th neighbor of y_{j-1}. It turns out that the number of random bits used in this process is $r + O(k)$, and therefore the whole process can be simulated deterministically in NC [1, 16].

Regarding the Randomized Rounding, this technique, in its general setting, seems to be inherently sequential [14]. However, as it is shown in [14], it is possible to derandomize it for several cases, and in particular, for the *Lattice Approximation* problem. The Lattice Approximation problem takes as input an $m \times h$ matrix C with each entry $c_{ij} \in [0,1]$; a vector $r \in [0,1]^h$ and a vector $d \in \mathbb{R}_+^m$ such that $Cr = d$, and outputs a vector $s \in \{0,1\}^h$ such that the norm $||Cr - d||_\infty$ is "small," where $||x||_\infty = \max_{1 \leq i \leq n} |x_i|$. Now, in our case, we use the Randomized Rounding under special conditions. This makes possible the NC derandomization of our case by applying the same technique as for Lattice Approximation problem. Indeed, we have a matrix A with non-negative entries $a_{ij} = O(1)$, a vector $x \in [0,1]^n$ and a vector $b \in \mathbb{R}_+^n$ such that $Ax = b$. By applying the above technique, we find deterministically in NC a vector solution $y \in \{0,1\}^n$ to the same system which approximately satisfies the constraints achieving a relative error of $O(\delta^{0.5-o(1)}\sqrt{\log n})$, i.e.,

$$(Ay)_i = b_i + (\delta^{0.5-o(1)}\sqrt{\log n})b_i,$$

where δ is an error parameter. Since $b_i = \Theta(n)$, we can choose an appropriate δ to satisfy conditions $(Ay)_i = b_i + O(\sqrt{n \log n})$ as required in Lemma 2.

4.3 Extensions

From the arguments in the previous subsections we have already proved Theorem 4. This result can be easily extended to positive instances of Smooth degree-l Integer Programming, for fixed l. Indeed, in this case we reduce (like in the case of Smooth QP) the degree-l program to a degree-$(l-1)$ program, and successively until a Smooth QP is obtained, whose approximate solution is also an approximate solution for the original problem. Note that the reduction is done in NC in l steps.

Theorem 6 *Given a positive instance of Smooth degree-l IP and a fixed ε, we can find in NC a 0,1 assignment to variables x_i such that*

$$p(x_1, x_2, \ldots, x_n) \leq p(x^*) + \varepsilon n^2 \tag{12}$$

where p is the objective function and $p(x^*)$ is its optimal value.

Notice from the definition of Smooth QP that $p(x^*) \geq \delta n^2$, for some positive constant δ, and therefore (12) is written as

$$p(x_1, x_2, \ldots, x_n) \leq (1 + \varepsilon/\delta) p(x^*)$$

which yields us to an approximation scheme. Furthermore, the overall parallel time is bounded by a polylogarithm in the number of variables n, and thus we get

Theorem 7 *There are NC Approximation Schemes for Positive Smooth QP and for bounded degree Positive Smooth IP.*

5 Applications

Smooth QP's (and more generally, Smooth Integer Programs) are strong enough to represent combinatorial problems. These programs were recently used to obtain Polynomial Time Approximation Schemes for dense instances of several NP-hard problems [2], however in their formulation, the instances of QP or degree-l Integer Programs are not positive. Thus, in order to apply our method we have first to formulate them as Positive instances in packing/covering form. In this section we will show how to do that for the problems MAX CUT, MAX DICUT and MAX-2-SAT. We also show how to extend the result of MAX-2-SAT for the general case of MAX-k-SAT on n variables. Next, we define a new problem, the Maximum Product Arrangement (MaxPA) problem for a graph and show that our technique is applicable also to this problem. Then, it turns out that for MAX CUT, MAX DICUT and MaxPA the condition for the instance to be smooth is equivalent with that of the graph instance being *"dense."* A graph instance of n vertices is considered dense if any vertex in it has degree $\Theta(n)$. While, an instance of MAX-k-SAT with m clauses over n variables is consider to be dense if $m = \Omega(n^k)$.

5.1 MAX CUT, MAX DICUT

Recall that in a MAX CUT instance we are given an undirected graph $G = (V, E)$ and we want to find a partition of its vertices $V = L \cup R$ such that maximizes the number of crossing edges from L to R. We associate to any vertex $i \in V$ two 0/1 variables l_i and r_i

defined as follows: $l_i = 1$ (resp. $r_i = 1$) if $i \in L$ (respectively $i \in R$). Now, let us consider the following program for MAX CUT:

$$\begin{array}{lll} \text{maximize} & \frac{1}{2}\sum_{\{i,j\}\in E}(l_i r_j + l_j r_i) & \\ \text{subject to} & & \\ & l_i + r_i = 1, & \forall i \in V \\ & l_i, r_i \in \{0,1\}. & \end{array} \tag{13}$$

Clearly, any coefficient in (13) is non-negative. In order to apply Theorem 4, we still need to transform (13) to match the requirements of this theorem. More precisely we have to:

(a) transform the restrictions "=" into restrictions "$\leq$." This is necessary because Luby and Nisan algorithm solve a positive LP in the packing/covering form, and

(b) the coefficients of the instance should satisfy the *smoothness* condition.

As for (a), we modify appropriately the objective function (see, e.g., [21, pages 141–143]) resulting in the following program:

$$\begin{array}{lll} \text{maximize} & \frac{1}{2}\sum_{\{i,j\}\in E}(l_i r_j + l_j r_i + l_i + l_j + r_i + r_j) & \\ \text{subject to} & & \\ & l_i + r_i \leq 1, & \forall i \in V, \\ & l_i, r_i \in \{0,1\}. & \end{array} \tag{14}$$

Observe that in (14), the restrictions "=" are relaxed in "$\leq$", yet this doesn't change the optimal solution since $l_i + r_i$ is multiplied by a positive constant in the objective function, and since we maximize this "forces" that restriction to hold with equality. Thus, the optimal solution of both (14) and (13) is the same. Finally, for condition (b), first note that the coefficients of the second degree terms are 1, and those of first degree are d_i, the degree[1] of vertex i. But notice by the definition of Smooth QP that, in the objective function, the sum of first degree terms is $\Theta(n^2)$, therefore in order for (14) to approximate the value of the optimal cut, the sum of second degree terms has to be also of order $\Theta(n^2)$. This is achieved if the number of second degree terms is $\Theta(n^2)$, that means if the graph instance is dense. At this point we have the connection between the density of the graph and smoothness of QP instance. Therefore, we have the following:

Theorem 8 *There is an NCAS for dense instances of MAX CUT.*

MAX DICUT problem, i.e., MAX CUT for directed graphs, can be modeled by a Positive QP similar to (13) where in the objective function we have only second degree terms $l_i r_j$, supposing that the edges of the cut are oriented from the left to the right. Therefore, Theorem 8 holds also for MAX DICUT.

5.2 MAX-2-SAT

In any instance of MAX-2-SAT we are given a set $\{C_j\}_{j=1}^m$ of m clauses over n-variables $x_1, \ldots, x_n$, where each clause has 2-literals, and we want to find a truth assignment to the variables that maximizes the number of satisfied clauses. For any variable x_i, we consider two other 0/1 variables t_i, f_i such that $t_i = 1$ (resp. $f_i = 1$) if the variable $x_i =$ true (resp. $x_i =$ false) and 0 otherwise. Now, we will associate a degree-2 term to any clause such that for any assignment of t_i, f_i that evaluates the term to 1, the corresponding assignment of

[1] In fact we can take instead of d_i any positive coefficient of order $\Theta(n)$.

x_i evaluates the clause to true. The term $q_j(t_1, \ldots, t_n, f_1, \ldots, f_n)$ corresponding to clause C_j is constructed as follows. If x_i appears positively in C_j, then we replace it by f_i and by t_i otherwise, and finally subtract the result from 1. For example, from the clause $\overline{x}_1 \vee x_3$ we construct the term $1 - t_1 f_3$. Therefore, we obtain the following program:

$$\begin{array}{lll} \text{maximize} & \sum_{j=1}^{m} q_j(t_1, \ldots, t_n, f_1, \ldots, f_n) & \\ \text{subject to} & & \\ & t_i + f_i = 1, & 1 \leq i \leq n \\ & t_i, f_i \in \{0, 1\}, & 1 \leq i \leq n. \end{array} \tag{15}$$

This program is not positive since the degree-2 terms appear negatively (like in $1 - t_1 f_3$). By letting, $q_j(t_1, \ldots, t_n, f_1, \ldots, f_n) = 1 - T_j(t_1, \ldots, t_n, f_1, \ldots, f_n)$, we can consider instead of (15) the program with the objective function, the sum of the degree-2 terms T_j and take minimization instead of maximization, and thus (15) is transformed into:

$$\begin{array}{lll} \text{minimize} & \sum_{j=1}^{m} T_j(t_1, \ldots, t_n, f_1, \ldots, f_n) + \sum_{i=1}^{n} d_i(t_i + f_i) & \\ \text{subject to} & & \\ & t_i + f_i \geq 1, & 1 \leq i \leq n \\ & t_i, f_i \in \{0, 1\}, & 1 \leq i \leq n. \end{array} \tag{16}$$

As in the cases before we have relaxed conditions $''=''$ to conditions $''\geq''$ by adding to the objective function $t_i + f_i$ multiplied by a positive constant d_i. In order for (16) to be smooth we must choose $d_i = \Theta(n)$, and an approximation for (16) would yield an approximation also for (15) if we let the number of degree-2 terms be of order $\Omega(n^2)$, i.e., if we let the MAX-2-SAT instance be "dense." Therefore, similarly as above, we have the following theorem:

Theorem 9 *There is an NCAS for dense instances of MAX-2-SAT.*

MAX-k-SAT. This problem can be modeled by a program similar to that of MAX-2-SAT (16), but in this case $T_j(t_1, \ldots, t_n, f_1, \ldots, f_n)$ is a degree-k term. In order for the resulting program to be smooth, we have to choose the coefficients d_i to be of order $d_i = \Theta(n^{k-1})$, and therefore $\sum_{i=1}^{n} d_i(t_i + f_i)$ will be of order $\Theta(n^k)$. Consequently, the objective function of the resulting program would approximate the maximum number of satisfied clauses if the sum $\sum_{j=1}^{m} T_j(t_1, \ldots, t_n, f_1, \ldots, f_n)$ has $\Theta(n^k)$ terms, i.e., if the instance of MAX-k-SAT is dense. Thus, from this observation we have:

Theorem 10 *There is an NCAS for dense instances of MAX-k-SAT.*

In fact the above result also holds for dense instances of a restricted version of the MAX SAT problem on n variables whose clauses have up to $l = polylog(n)$ literals. Indeed, in this case we will have a degree-l Integer Program for which the arguments of Subsection 4.3 also hold, i.e., the reduction to a Smooth QP is done also in NC.

Corollary 1 *There is an NCAS for dense instances of MAX SAT of clauses with n variables and up to $polylog(n)$ literals per clause.*

5.3 Product Arrangement of a Graph

Finally, we propose a problem, which we call Product Arrangement of a graph, defined analogously as that of Linear Arrangement [4].

Definition 1 Maximum Product Arrangement (MaxPA).
Given an undirected graph $G = (V, E)$ and a natural number c, determine whether there exists a one-to-one mapping $f : V \rightarrow \{1, \ldots, |V|\}$ such that

$$\sum_{\{u,v\} \in E} f(u) \cdot f(v) \geq c.$$

We observe that MaxPA can be written as a positive QP. Let us consider the following model:

$$\begin{array}{lll} \text{maximize} & \sum_{\{i,j\} \in E} x_i x_j & \\ \text{subject to} & x_i \neq x_j, & i \neq j \\ & x_i \in \{1, \ldots, n\}, & 1 \leq i \leq n \end{array} \tag{17}$$

In order to express the constraints on variables x_i linearly we introduce 0,1 variables δ_{ik},

$$\delta_{ik} = \begin{cases} 1 & \text{if } x_i = k, \text{ and} \\ 0 & \text{otherwise} \end{cases}$$

for $1 \leq i, \ k \leq n$ and put on them the conditions:

$$\delta_{ik} + \delta_{jk} \leq 1, \quad \forall \ i \neq j, \quad 1 \leq k \leq n.$$

That is, variables x_i and x_j cannot take the same value k and

$$\delta_{i1} + \delta_{i2} + \cdots + \delta_{in} = 1, \quad 1 \leq i \leq n.$$

That is, x_i takes a value in $\{1, \ldots, n\}$. Thus, we can write (17) in terms of δ_{ik} by substituting

$$x_i = 1 \cdot \delta_{i1} + 2 \cdot \delta_{i2} + \cdots + n \cdot \delta_{in}, \quad 1 \leq i \leq n.$$

Therefore, we have

$$\begin{array}{ll} \text{maximize} & n^2 \cdot \sum_{\{i,j\} \in E} \left(\frac{1}{n}\delta_{i1} + \frac{2}{n}\delta_{i2} + \cdots + 1 \cdot \delta_{in}\right)\left(\frac{1}{n}\delta_{j1} + \frac{2}{n}\delta_{j2} + \cdots + 1 \cdot \delta_{jn}\right) \\ \text{subject to} & \\ & \delta_{ik} + \delta_{jk} \leq 1, \quad \forall \ 1 \neq j, \quad 1 \leq k \leq n \\ & \delta_{i1} + \delta_{i2} + \cdots + \delta_{in} = 1, \quad 1 \leq i \leq n \\ & \delta_{ik} \in \{0, 1\}, \quad 1 \leq i \ , k \leq n. \end{array} \tag{18}$$

Clearly, any coefficient in (18) is non-negative and furthermore we transform it appropriately in order to apply Theorem 4, resulting in the following program:

$$\begin{array}{ll} \text{maximize} & \sum_{\{i,j\} \in E} \sum_{k,l=1}^{n} a_{kl} \delta_{ik} \delta_{jl} + \sum_{i=1}^{n} \sum_{k=1}^{n} b_i \delta_{ik} \\ \text{subject to} & \\ & \delta_{ik} + \delta_{jk} \leq 1, \quad \forall \ \{i, j\} \in E, \quad 1 \leq k \leq n \\ & \delta_{i1} + \delta_{i2} + \cdots + \delta_{in} \leq 1, \quad 1 \leq i \leq n \\ & \delta_{ik} \in \{0, 1\}, \quad 1 \leq i \ , k \leq n. \end{array} \tag{19}$$

where $a_{kl} = kl/n^2$ and b_i are positive constants. Following similar arguments as in the previous cases we have:

Theorem 11 *There is an NCAS for Maximum Product Arrangement on dense graphs.*

Open Questions

Find an NCAS for Positive QP or Positive CQP. It would be interesting to find other combinatorial problems which can be modeled as positive Smooth QP or bounded degree Smooth IP. We claim that MaxPA for non-bipartite graphs is NP-complete problem, and we believe that similar arguments to those of [4] can be used to prove it.

Acknowledgments

We thank Josep Díaz for many helpful discussions and suggestions. Many thanks to Raymond Greenlaw for his comments that significantly improved this paper. We are very grateful to an anonymous referee for his constructive criticism and many helpful suggestions.

References

[1] M. Ajtai, J. Komlos, and E. Szemeredi. Deterministic Simulation in LOGSPACE. In *Proceedings 19th ACM Symposium on Theory of Computing*, page 132, 1987.

[2] S. Arora, D. Karger, and M. Karpinski. Polynomial Time Approximation Schemes for Dense Instances of NP-hard Problems. In *Proceedings 27th Annual ACM Symposium on Theory of Computing*, pages 284–293, 1995.

[3] J. Díaz, M. Serna, P. Spirakis and J. Torán. *Paradigms for Fast Parallel Approximability.* Cambridge University Press. In preparation

[4] S. Even and Y. Shiloah. NP-completeness of Several Arrangements Problems. Technical Report No. 43, Dept. of Computer Science, Technion, Haifa, Israel, 1975.

[5] R. Greenlaw, H.J. Hoover, and W.L. Ruzzo. *Limits to Parallel Computation: P–Completeness Theory.* Oxford University Press, 1995.

[6] M. Grötschel, L. Lovász, and A. Schrijver. The Ellipsoid Method and its Consequences in Combinatorial Optimization. *Combinatorica*, 1:169–197, 1981.

[7] A. Karmarkar. A New Polynomial Time Algorithm in Linear Programming. *Proc. 16th Annual ACM Symposium on Theory of Computing*, pages 302–311, 1984.

[8] R. Karp, N. Pippenger, and M. Sipser. Expanders, Randomness, or Time versus Space. In *First Annual Conference on Structure in Complexity Theory*, pages 325–329, 1986.

[9] A. Lubotzky, R. Phillips, and P. Sarnak. Explicit expanders and the Ramanujan graphs. In *Proceedings 20th ACM Symposium on Theory of Computing*, pages 240–246, 1988.

[10] M. Luby and N. Nisan. A Parallel Algorithm for Positive Linear Programming. In *Proceedings 25th ACM Symposium on Theory of Computing*, pages 448–457, 1993.

[11] M. Luby and A. Widgerson. Pairwise Idependence and Derandomization. Technical Report TR-95-035, International Computer Science Institute, 1995.

[12] G. Margulis. Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and superconcentrators. *Problems of Information Transmission*, 24:39–46, 1988.

[13] N. Megiddo. A note on approximate linear programming. *Information Processing Letters*, 42:53, 1992.

[14] R. Motwani, J.S. Naor, and M. Naor. The Probabilistic Method Yields Deterministic Parallel Algorithms. *Journal of Computer and System Sciences*, 49:478–516, 1994.

[15] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[16] N Nisan and D. Zuckerman. More Deterministic Simulation in Logspace. In *Proceedings 25th ACM Symposium on Theory of Computing*, pages 235–244, 1993.

[17] P. Raghavan and C. Thompson. Randomized Rounding: a Technique for Provably Good Algorithms and Algorithmic Proofs. *Combinatorica*, 7:365–374, 1987.

[18] S. Sahni. Computationally Related Problems. *SIAM Journal of Computing*, 3:262–279, 1974.

[19] M.J. Serna. Approximating Linear Programming is log-space complete for P. *Information Processing Letters*, 37:233–236, 1991.

[20] S.A. Vavasis. Quadratic Programming is in NP. *Information Processing Letters*, 36:73–77, 1990.

[21] S. Walukiewicz. *Integer Programming.* Kluwer Academic Publishers, 1990.