Semidefinite Programs for the Design of Codes for Delay-Constrained Communication in Networks *

Frédérique E. Oggier
Section de Mathématiques,
Faculté des Sciences - Univ. de Genève.
CH-1211 Genève, Switzerland.

Sergio D. Servetto
Lab. de Communications Audiovisuelles,
Ecole Polytechnique Fédérale de Lausanne.
CH-1015 Lausanne, Switzerland.

http://lcavwww.epfl.ch/ccn/

Abstract

We consider the problem of designing codes for the problem of delay-constrained communication in packet networks. We show how good codes for this problem can be characterized as the solution of a pair of semidefinite programs plus a rank constraint. Using this characterization, we formulate the design problem as one of finding suitable graph embeddings into euclidean space, for a certain family of graphs which represent constraints that these codes must satisfy. In the case of codes in dimension 1, our formulation results in a good approximation algorithm for the classical problem of graph bandwidth, which may be of independent interest.

1 Introduction

1.1 Communication under Delay Constraints

Consider the following data transmission problem. At the location of a transmitter, we are given a point $p \in \mathcal{R}^n$ that needs to be encoded (using not more than R bits/dimension) and sent to a remote receiver. The transmission is to take place over a network which accepts as input packets of size at most $m < nR$ bits. Therefore, our encoding of nR bits is split into $\lceil \frac{nR}{m} \rceil$ packets, and all these packets are sent over the network: when all packets arrive at destination, the receiver reassembles the original length nR encoding, and computes an approximation to the original point p. Under the assumption that every transmitted packet arrives at destination, this is the classical problem of vector quantization. The goal in that case is to select a set of points $\Gamma = \{\gamma_1 ... \gamma_{2^{nR}}\} \subset \mathcal{R}^n$, to then send to the receiver the index of a point γ which satisfies that $||\gamma - p||^2 \leq ||\gamma_k - p||^2$, for all $k = 1...2^{nR}$. The design of good constellations (meaning, of sets Γ for which the average approximation error is small) is the classical vector quantization problem, fairly well understood by now [6].

In real life however, networks *do* lose packets: if while in transit a packet reaches a congested routing node, the router may be unable to service this packet, and drop it. One popular technique to deal with these dropped packets is for the receiver to keep

*Work done at EPFL, as part of F. E. Oggier's *Travail de Diplôme.*

1068-0314/01 $10.00 © 2001 IEEE

One popular technique to deal with these dropped packets is for the receiver to keep track of those that arrived, and send this information as feedback to the transmitter: if after a while the transmitter does not receive an acknowledgement for a packet sent, then it can assume that the packet was lost and retransmit it, and do this periodically until an ack is received.

The problem with retransmission-based protocols is that they may induce unacceptable delays for some applications. In a file download, it is critical that all transmitted bits arrive: in this case, delays are a nuisance to minimize, but it is hard to work around them otherwise. On the other hand, when the goal is to transmit points $p \in \mathcal{R}^n$, the situation is different: (a) even if all packets arrive some distortion is introduced anyway by representing p with its nearest $\gamma \in \Gamma$; and (b) the application may dictate hard upper bounds on acceptable delays, beyond which the data is useless even if all of it arrives (e.g., if p is a portion of an image in a video-conference). The main question that we address in this work is: how should a point p be encoded into nR bits, in a way such that we can get increasingly more accurate approximations of p as more and more packets become available at the receiver?

1.2 Low Bandwidth Sphere Packings

Given a total rate constraint of R bits/dimension, there is a total of 2^{nR} points that can be transmitted. For simplicity of notation, in this paper we will only consider the case in which the information to send is to be split among two equal length packets (the extension to more than two packets, as well as to packets of unequal length, is entirely straightforward). To split nR bits over two *independent* encodings of length $\frac{1}{2}nR$, consider using some subset $\mathcal{L} \subset \{1...\sqrt{2^{nR}}\} \times \{1...\sqrt{2^{nR}}\}$, where each of the coordinates represents a packet. Let π_1, π_2 denote projection operators, and for any $(i,j) \in \mathcal{L}$, define also the sets $\mathcal{L}_i^1 = \{\ell \in \mathcal{L} : \pi_1(\ell) = i\}$ and $\mathcal{L}_j^2 = \{\ell \in \mathcal{L} : \pi_2(\ell) = j\}$. The main question that needs to be addressed in the design of the sought codes is: for a given set $\mathcal{L}$, to which $p_{ij} \in \mathcal{R}^n$ should each $(i,j) \in \mathcal{L}$ be mapped, in order to minimize the diameter (i.e., the approximation error) of the sets $\mathcal{L}_i^1$ and $\mathcal{L}_j^2$, and subject to the normalizing constraint that $||p-p'|| \geq 1$, for all mapped points $p \neq p'$?[1] The tradeoffs involved in different choices of $\mathcal{L}$ are illustrated in Fig. 1.

First of all, we need to understand what are the tradeoffs involved in the different choices of $\mathcal{L}$. At one extreme, suppose first that $\mathcal{L} = \{1...\sqrt{2^{nR}}\} \times \{1...\sqrt{2^{nR}}\}$. In this case, we have that $|\mathcal{L}_i^1| = |\mathcal{L}_j^2| = \sqrt{2^{nR}}$. The volume of an n-dimensional sphere of radius ρ is $B_n\rho^n = \pi^{\frac{n}{2}}/\frac{n}{2}!\rho^n$ (asuming n even). Therefore, the volume of the smallest sphere containing a sphere of radius $1/2$ –enough to guarantee the normalizing constraint– centered around each point in $\mathcal{L}_i^1$ or $\mathcal{L}_j^2$ is at least $\sqrt{2^{nR}}B_n\frac{1}{2^n}$, and so its radius must be at least $\frac{1}{2}\sqrt{2^R}$, which in turns implies that $d(\mathcal{L}_i^1) = d(\mathcal{L}_j^2) \geq \sqrt{2^R} - o(1)$ (the $o(1)$ term accounts for some negligible boundary effects ignored in this estimate of the diameter). At the other extreme, suppose now that $\mathcal{L} = \{(1,1),(2,2),...,(\sqrt{2^{nR}},\sqrt{2^{nR}})\}$. In this case, it is clear that $d(\mathcal{L}_i^1) = d(\mathcal{L}_j^2) = 0$, irrespective of the transmission rate R. Therefore, we see that for our first choice

[1] Diameter of $P = d(P) = \sup_{p,p' \in P} ||p - p'||^2$.

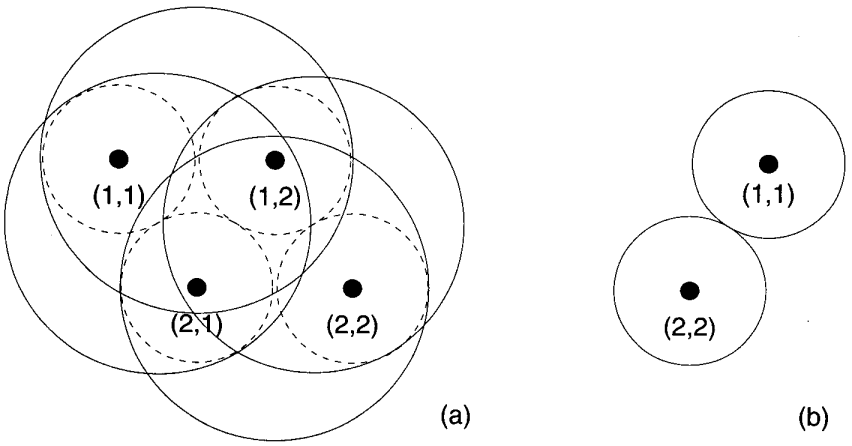

Figure 1: On the different choices for $\mathcal{L}$. With 4 points to send (2 bits), we create two packets of length 1 bit. Which one-bits should be put in each packet? In (a) we send all possible pairs of bits: this leads to packing more information in a message (1 of 4 points is sent), but the uncertainty due to losses is higher, as reflected by the size of circles containing all points with equal projections (i.e., that could be confounded in the presence of losses). With (b) we send less information (only 1 of 2 points), but losses are less costly. Our goal is to decide, for a fixed and given set $\mathcal{L}$ (which in practice would be obtained as a function of network conditions), which points in $\mathcal{R}^n$ to let each pair in $\mathcal{L}$ represent, trying to keep the points that could be confounded as close as possible to each other.

of $\mathcal{L}$, we can transmit a total of 2^{nR} different points, but with high uncertainty if one of the components is lost, or else we could choose to send less information (as little as $\sqrt{2^{nR}}$ points), in which case the uncertainty under failures is much lower. We see then that the role of $\mathcal{L}$ is essentially that of separating the information to be transmitted into independent components, and of trading off amount of information sent for robustness in the presence of loss of information. As such, we regard $\mathcal{L}$ as a parameter of our code: for a fixed and given $\mathcal{L}$, we want to find the best set of points in $\mathcal{R}^n$ to transmit using $\mathcal{L}$. In this paper, we show how this problem can be formulated as a pair of semidefinite programs.

1.3 Semidefinite Programming

In *semidefinite programming* the goal is to optimize a linear function of a matrix, subject to linear constraints, and subject to a constraint on the matrix being positive semidefinite. The standard primal formulation is given by:

$$\begin{aligned} \min \quad & C \bullet X \\ \text{s.t.} \quad & A_i \bullet X = b_i, \quad i = 1, ..., m \\ & X \succcurlyeq 0 \end{aligned}$$

where C, A_i's and X are $n \times n$ matrices, and X is symmetric; the '$\bullet$' operation is the inner product of matrices:

$$A \bullet B \triangleq \sum_{i,j} A_{ij}B_{ij} = \text{trace}A^\top B;$$

and the 'inequality constraint' $\succcurlyeq$ indicates for real symmetric matrices A and B, $A \succcurlyeq B$ (respectively $A \succ B$), whenever $A - B$ is positive semidefinite (respectively positive definite). Semidefinite programs (SDPs) are a generalization of linear programs, in that linear programs are SDPs with the added constraint that the solution matrix X be diagonal.

Let U be any positive semidefinite matrix. It is a classical result from linear algebra that such matrices admit a factorization of the form $U = V^\top V$. If we denote by v_i the columns of V, then we have that $U_{ij} = v_i \cdot v_j$. The usefulness of semidefinite programs for us will be that we will be able to solve optimization problems involving sets of points $v_1 ... v_n$, on which we will be able to specify linear constraints on the inner products among arbitrary pairs of points.

The literature on SDP is extensive, and there is no way we can cover it extensively here. For details, the reader is referred to some excellent tutorial articles and books that cover the subject [1, 7]. Applications of SDP to combinatorial optimization problems of the type that will be studied in this work are surveyed in [1].

1.4 Main Contributions and Organization of the Paper

The problem of packet communication under delay constraints, as stated above, is equivalent to the problem of Multiple Description (MD) source coding, if "channels" in a diversity system are identified with "packets" in a network transmission. In this work, essentially what we do is "turn around" the labeling problem for lattice quantizers considered in [10]. In that work, the authors started with a fixed codebook (the lattice), and then solved the problem of deciding which pairs of labels to assign to each codeword. In this paper, we start with a given set of labels (the set $\mathcal{L}$), and we solve the problem of deciding which codewords (in $\mathbb{R}^n$) to assign to each label. Our main tool for carrying out this plan is a graph structure which describes, in a very simple and intuitive manner, all the constraints that labeling functions for MD codes must satisfy. The construction of that graph is inspired by a series of papers on zero-error information theory, particularly by [8].

Different special cases of the problem of designing MD codes have been studied in the past:

- MD scalar quantization, two channels, equal rates [11].
- MD lattice vector quantization, two channels, equal rates [10].
- MD lattice vector quantization, two channels, unequal rates [4].

And whereas all of these problems are relatively simple variations on the same theme (design of MD codes), the codes obtained in each of these cases appear to be difficult to generalize to other cases. We believe the main contribution of this paper consists

of having obtained a characterization of good codes for the problem of MD coding as the rank-constrained solution of a suitably defined SDP program, and its relaxation into a "pure" SDP formulation. This characterization allows us to accomplish two important goals. First, to present a unified view on all the variations mentioned above. Second, to obtain codes for variations of this problem on which much less is known, such as the case of more than two packets [5]. In our opinion, the problem of MD coding is an intuitively simple one. Unfortunately however, most existing solutions are technically fairly involved. Our characterization of good codes will allow us to focus on the essential aspects that make certain codes better than others, using only elementary geometry and continuous optimization concepts.

The rest of this paper is organized as follows. In Section 2, we develop the above mentioned characterization of good codes. In Section 3 we show how to relax the original formulation into a pure SDP formulation (without rank constraints), and we study properties of this relaxation. In Section 4 we give a summary, and discuss future work.

2 A Mathematical Programming Characterization for Good Codes

Define a graph $G = (V, E)$, where $V = \mathcal{L}$, and $E = E_1 \cup E_2 \subseteq V \times V$, with $E_k = \{(\ell, \ell') : \pi_k(\ell) = \pi_k(\ell')\}$: essentially, the vertices of this graph are the pairs $\mathcal{L}$, and an edge exists between two pairs whenever loss of information leads to confusion among them. The optimal choice of points in $\mathcal{R}^n$ is given by the solution to the pair of programs in Table 1.

$\min C$				$\min b$			
s.t.	$\|v_i - v_j\|^2$	$\geq$	$1,\ \forall i \forall j$	s.t.	$\|v_i - v_j\|^2$	$\geq$	$1,\ \forall i \forall j$
	$\|v_i\|$	$\leq$	$C,\ \forall i$		$\|v_i\|$	$\leq$	$C^*,\ \forall i$
	$\text{rank}\{v_1 ... v_N\}$	$=$	n		$\text{rank}\{v_1 ... v_N\}$	$=$	n
					$\|v_i - v_j\|^2$	$\leq$	$b,\ \forall (i,j) \in E$

Table 1: Exact formulation of the problem of designing MD vector quantizers, in the special case of two balanced descriptions. Here, C^* is the optimal solution for the first program.

The idea is that with the first program in Fig. 1 we find the radius C^* of the smallest sphere containing N spheres of radius 1. In the second program, we use that radius C^* as a constant: subject to having all the spheres densely packed, we want to minimize the bandwidth b of the embedding. Intuitively, the role of the constraints is the following:

- $\|v_i - v_j\|^2 \geq 1$: to avoid degenerate solutions, such as having $v_i = \bar{0}$, for all i.
- $\|v_i\| \leq C$: to obtain dense packings. This is an approximation to the quantizer design problem: in high dimensions, the Voronoi cell of optimal quantizers are

spheres. Furthermore, in low dimensions, in many cases the best known sphere packing coincides with the best known quantizer [3]. This constraint will be seen later to be necessary also in the relaxation of the rank constraint.

- $\|v_i - v_j\|^2 \leq b$, whenever $(i,j) \in E$: to obtain small bandwidth embeddings. As argued before, the bandwidth of the embedding will be directly proportional to the approximation error when information is lost.

- $\text{rank}\{v_1 ... v_N\} = n$: to ensure that the resulting solution set will correspond to points in $\mathbb{R}^n$. This issue is further elaborated on next.

Suppose now that we have an algorithm that solves the programs above, from which we can obtain a matrix $U = (v_i^\top v_j)_{i,j=1}^N$. But what we are interested in is the set of vectors v_i, $i = 1, ..., N$. Then we still need to find the v_i's, knowing all pairs of dot products. What we need in this case is a factorization of U into $U = V^\top V$, where $V = (v_1, ..., v_N)$. From classical linear algebra, we have the following result:

> *Let M be an $n \times n$ symmetric matrix. The two following conditions are equivalent:*
>
> 1. *There exists an $p \times n$ matrix R such that $M = R^\top R$.*
> 2. *rank $M \leq p$ and the eigenvalues of M are all ≥ 0.*

This essentially says that if we want the v_i's to be in $\mathbb{R}^n$, we need rank $V = n$.

Such a factorization is relatively easy to obtain if we consider the case $N = n$: since our SDP constraints (without the rank constraint) will induce full rank solutions, a standard Cholesky factorization will do the trick. The problem is that we are typically interested in finding the solution for $N \gg n$ (for example, $n = 2$ and $N = 100$). But in this case, from a Cholesky factorization we obtain a set of vectors in $\mathbb{R}^{100}$. Hence, we see that a constraint on the rank of a solution is needed indeed.

3 Codes as Solutions of Semidefinite Programs

3.1 The Spreading Constraints

Given the formulation in Table 1, one could try to write a computer program to solve those two optimization problems. Unfortunately, it seems unlikely that an *efficient* algorithm exists for such purpose. This is because we observe that if it were possible to set the rank of a solution to 1 and still solve this problem in polynomial time, then we would obtain a polynomial time algorithm for solving the classical graph bandwidth problem, a problem known to be NP-hard [9]. Therefore, it seems clear that some sort of approximation is needed.

The main difficulty in solving the programs in Table 1 efficiently appears to be caused by the rank constraint. As argued in Section 2, we need to control the rank of a solution matrix U in order to get a solution of the right dimension, so this constraint cannot be eliminated. However, excluding this constraint, all other constraints are

legal constraints for semidefinite programs (SDPs). And SDPs are indeed solvable in polynomial time using, for example, the Ellipsoid method [7].

We claim that if we relax just a little bit the rank requirement, we will still be able to come up with a meaningful solution. The idea will be to replace the rank constraints by a set of *spreading* constraints, whose role is to induce "almost" rank deficient solutions to the SDPs. These spreading constraints will take the form of linear combinations of dot products among the v_i's, and hence solvable using the Ellipsoid method as well [7, Ch. 3]. How this is so is best illustrated with an example, presented in Fig 2.

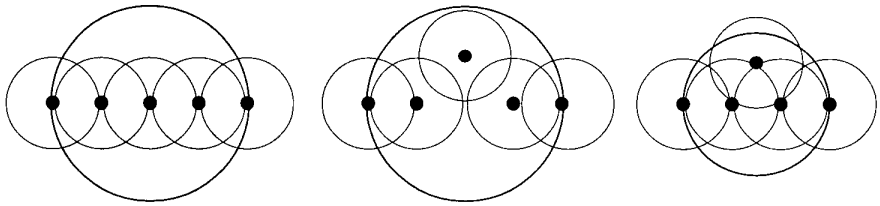

Figure 2: To illustrate how the spreading constraints, coupled with the dense packing property, may be used to induce nearly rank deficient solutions. For simplicity, consider the first program only (no bandwidth constraints yet). A solution set for this program, in 1D, is illustrated in (a). If we move one of these points outside of the desired subspace, as in (b), then C is no longer smallest. And if we rearrange the v_i's so that a new C becomes smallest, as in (c), we notice that the solution points are closer to each other than in (a). The essential idea of the spreading constraints is to disallow configurations of points "too close" to each other.

3.2 Some Intuition: the 1D Case

The requirement of having points "not too close to each other" is formally stated by asking that the sum of distances within sets of points not be small. Then the general form of the constraints is:

$$\sum_{s \in S} \|p - s\|^2 \geq C_S \triangleq \sum_{s \in \Lambda_{|S|}} \|p - s\|^2,$$

where:

- $p \in S \subseteq \mathcal{L}$, $S \neq \emptyset$ (one such constraint for each p and each S).
- Λ is a dense sphere packing in n dimensions, and Λ_k denotes the set of the k shortest vectors in Λ.

Basically, the constants C_S are derived from the distances among points in a *known* dense packing in $\mathbb{R}^n$. To better understand the role played by the C_S's, before

giving their most general form we present their derivation with an example, in the specific case of the 1D integer lattice packing.

Consider 7 points (s_1,...,s_7) at distance one from each other, as in Fig. 3.

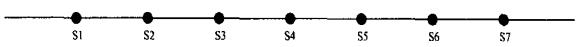

Figure 3: 7 equidistant points $s_1, ..., s_7$, in $\mathbb{Z}$.

Notice that at distance zero from s_4, there is only one point, s_4 itself. At distance one from s_4, there are two points, s_3 and s_5. At distance two from s_4, there are two points, s_2 and s_6 and finally, at distance three, there are two points too, s_1 and s_7. If we consider the squared distance, that means that at squared distance 1, 4 and 9 from s_4 there are two points, at squared distance zero there is one point, and at squared distance 2, 3, 5, 6, 7, 8 there are no points. We can generalize that considering an infinite number of points ($\mathbb{Z}$) instead of seven points, and we obtain the following table, where m is the squared distance from a given point and $N(m)$ is the number of points at squared distance m from the given point. $N(m) = 2$ when m is a squared number and $N(m) = 0$ else (with $N(0) = 1$).

m	0	1	2	3	4	5	6	7	8	9	10	...
$N(m)$	1	2	0	0	2	0	0	0	0	2	0	...

Now, if we compute $\sum_{i=1}^{7} \|s_4 - s_i\|^2$, we obtain

$$\begin{aligned} 3^2 + 2^2 + 1^2 + 0^2 + 1^2 + 2^2 + 3^2 &= 2 \cdot 1^2 + 2 \cdot 2^2 + 2 \cdot 3^2 \\ &= N(1) \cdot 1 + N(4) \cdot 4 + N(9) \cdot 9 \\ &= \sum_{k=0}^{9} k \cdot N(k). \end{aligned}$$

This shows that the expression $\sum_k kN(k)$ computes the sum of the squared distance from one point to other *symmetric* points (s_1,s_2,s_3,s_5,s_6,s_7 are symmetric around s_4): k corresponds to the distance, and $N(k)$ to the number of points.

Suppose now that we have only six points (instead of 7), and that we consider s_3 as the anchor point. We now have two points at distance 1, two points at distance 2, but only one point at distance 3. Then if we compute $\sum_{k=1}^{9} kN(k)$, we include a distance corresponding to a point not in this set; if instead of using k=1...9, we use $k = 1...4$, we exclude the point at distance 3. This is illustrated in Fig 4.

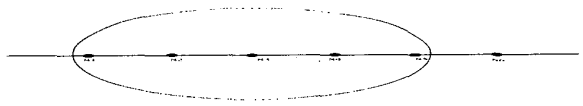

Figure 4: 6 equidistant points $s_1, ..., s_6$, in $\mathbb{Z}$.

In this latter case, the problem is that in the expression $\sum_{k=1}^{9} kN(k)$ we have that k is the correct squared distance, but the number of points at squared distance

9 is not $N(9)$, but smaller instead. How many points are there exactly? Well, there are $1 = 6 - 5 = |S| - \sum_{k=0}^{M-1} N(k)$, where M is the smallest number such that $\sum_{k=0}^{M} N(k) \geq |S|$. Hence, to conclude our example, we have that for seven points $M = 9$, and

$$\begin{aligned}\sum_{i=1}^{6} \|s_3 - s_i\|^2 &= 2^2 + 1^1 + 1^1 + 2^2 + 3^2 = \sum_{k=0}^{8} kN(k) + 3^2 \\ &= \sum_{k=0}^{M-1} kN(k) + M\left(|S| - \sum_{k=0}^{M-1} N(k)\right).\end{aligned}$$

3.3 The General Case

The solution in n dimensions proceeds as follows. Consider a fixed, given packing $\Lambda \subset \mathbb{R}^n$, with theta series

$$\theta(q) = \sum_{\lambda \in \Lambda} q^{\lambda \cdot \lambda} = \sum_{k=0}^{\infty} N(k) q^k.$$

From this packing obtain the sequence $N(k)$ counting the number of points of squared norm k in Λ. Consider also a set $V = \{v_1 ... v_N\} \subset \mathbb{R}^n$. Compute the numbers $C_0 ... C_N$ by letting

$$\begin{aligned}M_i &= \min\left\{m \in \mathbb{N} : \sum_{j=0}^{m} N(j) \geq i\right\} \\ C_i &= \sum_{k=0}^{M_i - 1} kN(k) + M_i\left(i - \sum_{k=0}^{M_i - 1} N(k)\right)\end{aligned}$$

Then, we compute codes for our problem by solving the programs of Table 1, but where the rank constraints are replaced by a set of constraints of the form

$$\sum_{s \in S} \|p - s\|^2 \geq \underbrace{\sum_{k=0}^{M-1} kN(k) + M\left(|S| - \sum_{k=0}^{M-1} N(k)\right)}_{C_{|S|}},$$

for all subsets $\emptyset \subset S \subseteq V$, and for all $p \in S$.

We conclude this section with a remark on the effect of these constraints on the rank of a solution. We observe that the combination of the spreading and the dense packing constraints essentially forces the norm of most points to be that prescribed by the coefficients of the θ series of the lattice considered (except possibly for points in the vicinity of the surface of the smallest sphere that contains them all). Therefore, it seems only natural to ask the following question: does the θ series uniquely determine the underlying lattice Λ? If the answer were positive, we could claim that the solution of our SDPs are small bandwidth labelings of the points in Λ, and hence that the

dimension of our solution set is indeed the sought one. Unfortunately, there are a few examples where inequivalent lattices are known to have identical θ series. However, in all examples we are aware of the inequivalent lattices are all of the same dimension; and furthermore, at least in dimension 2 a lattice is uniquely determined by its θ series [3, pg. 47]. This issue is currently under investigation.

4 Conclusions

In this paper we have presented the first steps of our work on the design of codes for the problem of delay-constrained network communication, based on the use of semidefinite programming principles. We would like to conclude our presentation with a discussion on our current work in this regard.

The main task we are currently working on is that of solving the proposed SDPs, to obtain concrete codes. Obtaining such numerical results requires a fair amount of work, since standard packages for solving SDPs typically require an explicit listing of *all* the constraints, but in our problem the number of constraints is exponential in the number of points of a solution. We are currently developing an implementation of the Ellipsoid method of [7] for solving SDPs, in which instead of describing the feasible polytope of the SDP by explicitly listing all its constraints, that polytope is implicitly described by an algorithm that can answer whether a given point is feasible or not, and if not returns a hyperplane separating the given point from the feasible set; this is how we circumvent the need for listing exponentially many constraints. Future work will deal with the analysis of performance of codes thus obtained.

Acknowledgements – The work of Blum et al. [2], on the graph bandwidth problem, provided much of the motivation for some of these ideas: we are grateful to T. Berger-Wolf and E. M. Reingold for bringing this paper to our attention. We also wish to thank our reviewers, for their thoughtful comments and suggestions. Finally, we also wish to thank M. Vetterli, for his continuous support and encouragement.

References

[1] F. Alizadeh. Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization. *SIAM J. Optim.*, 5(1):13–51, 1995.

[2] A. Blum, G. Konjevod, R. Ravi, and S. Vempala. Semi-Definite Relaxations for Minimum Bandwidth and other Vertex-Ordering Problems. In *Proc. 30th ACM Symp. Theory Comp. (STOC)*, 1998.

[3] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer Verlag, 3rd edition, 1998.

[4] S. N. Diggavi, N. J. A. Sloane, and V. A. Vaishampayan. Design of Asymmetric Multiple Description Lattice Vector Quantizers. In *Proc. IEEE Data Compression Conf. (DCC)*, Snowbird, UT, 2000.

[5] M. Fleming and M. Effros. Generalized Multiple Description Vector Quantization. In *Proc. IEEE Data Compression Conf. (DCC)*, Snowbird, UT, 1999.

[6] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Trans. Inform. Theory*, 44(6):2325–2383, 1998.

[7] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer Verlag, 1988.

[8] L. Lovász. On the Shannon Capacity of a Graph. *IEEE Trans. Inform. Theory*, 25(1):1–7, 1979.

[9] C. Papadimitriou. The NP-Completeness of the Bandwidth Minimization Problem. *Computing*, 16:263–270, 1976.

[10] S. D. Servetto, V. A. Vaishampayan, and N. J. A. Sloane. Multiple Description Lattice Vector Quantization. In *Proc. IEEE Data Compression Conf. (DCC)*, Snowbird, UT, 1999.

[11] V. A. Vaishampayan. Design of Multiple Description Scalar Quantizers. *IEEE Trans. Inform. Theory*, 39(3):821–834, 1993.