OPTIMIZED DISTRIBUTED 2D TRANSFORMS FOR IRREGULARLY SAMPLED SENSOR NETWORK GRIDS USING WAVELET LIFTING

Godwin Shen and Antonio Ortega

Signal and Image Processing Institute
University of Southern California
godwinsh@usc.edu, ortega@sipi.usc.edu

ABSTRACT

We address the design and optimization of an energy-efficient lifting-based 2D transform for wireless sensor networks with irregular spatial sampling. The 2D transform is designed to allow for unidirectional computation found in existing path-wise transforms, thereby eliminating costly backward transmissions often required by existing 2D transforms, while simultaneously achieving greater data decorrelation than those path-wise transforms. We also propose a framework for optimizing the 2D transform via an extension of standard dynamic programming (DP) algorithms, where a selection is made among alternative coding schemes (e.g., different number of levels in the wavelet decomposition). A recursive DP formulation is provided and an algorithm is given that finds the minimum cost coding scheme assignment for our proposed 2D transform.

Index Terms— Data Compression, Dynamic Programming, Wavelet Transforms, Wireless Sensor Networks

1. INTRODUCTION

In recent years, low-cost, densely deployed Wireless Sensor Networks (WSN) have been studied for applications such as instrumentation and actuation of standard household systems or for monitoring of complex environments [1]. Because these are typically battery powered devices, it is important to find power-efficient techniques for data gathering and transmission.

In this paper, we focus on in-network compression techniques, where data collected at individual nodes is aggregated and compressed as it "flows" towards a fusion center or sink, along a series of pre-established paths. These approaches exploit spatial correlations in the data in order to reduce the overall number of bits needed to represent a snapshot of information sensed in the network. Obviously, the goal is to achieve *overall* power savings. Thus, these systems have to be evaluated by considering not only the "final" transmission costs (which depend on the number of bits needed to represent compressed data) but also the cost involved in generating a compressed representation (e.g., additional "local" data transmissions needed for compression or the cost of computation at the nodes.)

Techniques for in-network data compression include the distributed KLT [2], wavelet based methods [3, 4, 5, 6, 7, 8, 9], and networked Slepian-Wolf coding [10]. These techniques may all involve some "local" communication overhead, i.e., neighboring nodes have to exchange information so as to be able to compute the distributed transform. For example in the distributed KLT or wavelet-based methods each transform coefficient is computed based on information from multiple nodes and so the nodes need to exchange data (e.g., a finely quantized version of their measurements) before the actual transform coefficients can be computed. Note that even networked Slepian-Wolf coding techniques [10] would require some communication overhead, since the correlation structure of the sensor data is assumed to be known, which would at least require some initialization and periodic updates (where nodes would exchange measurements to estimate the correlation.)

We consider, in the context of wavelet-based approaches [3, 4, 9], a fundamental trade-off, namely, that of deciding the number of levels of decomposition to be used: more levels of decomposition leads to potential improvements in compression performance, but at the cost of additional local communication (i.e., more levels leads to longer spatial filters, so that input data from more nodes is needed to generate a single output.) Wavelet transforms are widely used to compress regularly sampled 2D images and thus their application to compress sensor data, which can be seen as samples of a 2D field, is a natural extension. For example, an early proposal by Servetto [6] was based on the assumption that sensors are located in a regular grid. But, as illustrated in [7], these regular grid assumptions will not hold in general.

This has motivated researchers to investigage transforms that operate with irregular 2D node placements. Wagner *et al* [7] propose to apply a lifting transform along a tesselation of the nodes. While this 2D transform can operate over arbitrary node deployments, a major drawback of this method is that it does not directly consider the cost involved in computing the transform, in particular the abovementioned local communication costs. More specifically, this method requires backward data transmissions that flow away from the sink, so that compression performance is good in terms of overall rate, but the transmission costs can be high. As an alternative, there have been proposals to use 1D wavelet transforms along the routing paths in the sensor network, in a way that explicitly considers the cost of computing a transform and transporting transformed data to the sink [3, 9]. These methods compute the transform in a unidirectional manner, i.e., the transform is computed as data flows towards the sink, thereby eliminating the overhead introduced in [7, 8], and, again unlike [7, 8], they also provide techniques for optimizing the number of levels of decomposition in order to exploit the tradeoff between local communications and overall compression rates. The main drawback of the approaches in [3, 9] is that they are essentially 1D transforms and, consequently, do not exploit the 2D spatial correlation that exists in general 2D networks. For example, in [3], 1D transforms are performed and simple ad hoc techniques are used to combine the information obtained from two or more merging paths along a route to the sink, so that the overall signal representation is not critically sampled [1].

This work was supported in part by NASA under grant AIST-05-0081.

[1] The transform can be made critically sampled by simply forcing the rout-

In this paper, we seek to combine the main advantages of these existing methods. As in [7, 8], we define a critically sampled 2D transform by extending the techniques in [3]. These new 2D transforms are computed in a unidirectional manner, eliminating the backward transmissions required in [7, 8], and can be developed on arbitrary routing trees, facilitating joint decisions on compression and routing. We also develop an algorithm that can choose the optimal coding scheme assignment for such a 2D transform in order to exploit the tradeoff seen in [3, 9]. To the best of our knowledge, no technique has been developed that can optimize encoding for a unidirectional 2D transform for a given routing tree in an irregular 2D node deployment. Path-wise approaches were suggested in [3] but are essentially 1D optimizations that do not directly extend to our 2D transform.

The remainder of this paper is organized as follows. Section 2 presents the design and unidirectional computation of our proposed transform. Section 3 gives a general overview of our proposed optimization framework. Section 4 provides experimental results to compare our proposed method against those in [3, 7]. Section 5 concludes the paper.

2. UNIDIRECTIONAL 2D TRANSFORMS USING LIFTING

Our goal in this section is to design a distributed 2D transform using lifting that is critically sampled to avoid the overhead in [3] and computable in a unidirectional manner in order to lower transmission costs. Consider a sensor field with N nodes, where data x_n is captured by node $n \in \mathcal{I} = \{1, 2, \ldots, N\}$. Let T be a tree representing data routing through the network, with the root of the tree corresponding to the sink (indexed by $N+1$) , and each leaf node representing the first sensor in a given path towards the sink. Let $\mathcal{C}_n$ and ρ_n denote the set of children and the parent of sensor n in T, respectively. We shall enumerate the children of n as $c_{n,m} = \mathcal{C}_n(m)$, where $m \in K_n = \{1, 2, \ldots, M_n\}$ and $M_n = |\mathcal{C}_n|$ is the number of children of node n. Let depth(n) be the depth of node n in T, with depth$(N+1) = 0$.

2.1. Lifting Transform Design

A lifting transform [11] can be performed once we define disjoint sets of prediction and update nodes at each level of decomposition j, denoted $\mathcal{P}_j$ and $\mathcal{U}_j$, respectively. Denote $\mathbf{p}_{n,j}$ and $\mathbf{u}_{m,j}$ the prediction and update operators at nodes $n \in \mathcal{P}_j$ and $m \in \mathcal{U}_j$, respectively. Then following the standard lifting techniques for $j = 1$, we will have that the detail $d_m (m \in \mathcal{P}_1)$ and smooth, $s_n (n \in \mathcal{U}_1)$, coefficients can be computed as:

$$d_m = x_m + \sum_{k \in \mathcal{U}_1} \mathbf{p}_{m,1}(k) x_k \quad \text{and} \quad s_n = x_n + \sum_{k \in \mathcal{P}_1} \mathbf{u}_{m,1}(k) d_k,$$

where $\mathbf{p}_{n,1}(n) = \mathbf{u}_{m,1}(m) = 1$.

For a 1-level transform, we split the nodes into prediction and update sets according to their depth (i.e., the set of prediction nodes will include all nodes of odd depth, with update nodes being those of even depth) with respect to the root of the tree (the sink) which has depth zero. Thus all one-hop neighbors of an even (odd) node in T will be odd (even) nodes. In order facilitate computation of the transform as data is being transported along T, we choose localized operators so that non-zero weights are assigned only to one-hop neighbors (in T) of a given node, i.e., $\mathbf{p}_{n,1}(k) = 0$ and $\mathbf{u}_{n,1}(k) = 0$ for $k \notin \mathcal{C}_{n,1} \cup \{\rho_{n,1}\}$. Note that, in general, nodes that are close to each other in T will also be close in physical distance (so that local transport costs are kept low) and will also tend to produce data that is more correlated than nodes further away in the tree. Thus, restricting the transform to the one-hop neighbors is also reasonable in terms of exploiting signal correlation for compression.

Figure 1 gives an example of the tree used to split nodes for 2-levels. By extension to j-levels, T_j will consist of nodes of even depth in T_{j-1} with an edge between two nodes in T_j only if they are 2-hops apart in T_{j-1}. Then we can apply the same split method for each T_j. Using this construction, we denote the children and parent nodes of $n \in T_j$ as $\mathcal{C}_{n,j}$ and $\rho_{n,j}$. Note that j-levels can be used only if $\lfloor \max(\text{depth})/2^j \rfloor \geq 1$. Also note that only T is used for routing, all other trees are for splitting purposes only.

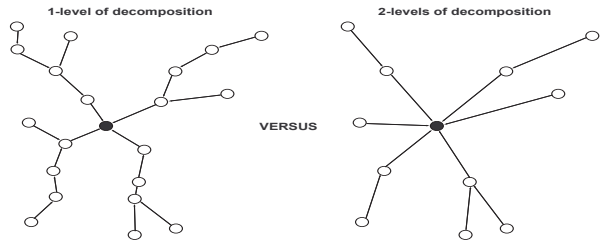

Fig. 1. Trees used for splitting. Black center node is the sink

To facilitate the extension of the algorithms in [3] to our setting, we propose the following filters for j-levels. For a node $n \in \mathcal{P}_j$, we generate a prediction by averaging data in neighboring nodes, i.e., $\mathbf{p}_{n,j}(m) = -\frac{1}{|\mathcal{C}_{n,j}|+1}$ for each $m \in \mathcal{C}_{n,j} \cup \{\rho_{n,j}\}$. For a node $m \in \mathcal{U}_j$, we similarly perform smoothing by using the detail coefficients of its neighbors, i.e., $\mathbf{u}_{m,j}(k) = \frac{1}{2(|\mathcal{C}_{m,j}|+1)}$ for each $k \in \mathcal{C}_{m,j} \cup \{\rho_{m,j}\}$. Note that for 1D paths, these filters simplify to the 5/3 CDF lifting filters from which our intuitive design is derived. Our transform can also be extended to use filter design methods similar to those in [7] so that filters are adapted to relative node positions. This may be a topic for future work.

2.2. Unidirectional Transform Computation

At j-levels of decomposition, the lifting transform in Section 2.1 is computed by first applying the prediction filter on even nodes (in T_j) to generate each $d_{m,j}$, then applying the update filter on the detail coefficients to generate each $s_{n,j}$. Also notice that $s_{n,0} = x_n$ for all $n \in \mathcal{I}$. So for every $m \in \mathcal{P}_j$:

$$d_{m,j} = s_{m,j-1} + \sum_{k \in \mathcal{C}_{m,j}} \mathbf{p}_{m,j}(k) s_{k,j-1} + \mathbf{p}_{m,j}(\rho_{m,j}) s_{\rho_{m,j},j-1} \tag{1}$$

and given every $d_{m,j}$, for each $n \in \mathcal{U}_j$ we have:

$$s_{n,j} = s_{n,j-1} + \sum_{m \in \mathcal{C}_{n,j}} \mathbf{u}_{n,j}(m) d_{m,j} + \mathbf{u}_{n,j}(\rho_{n,j}) d_{\rho_{n,j},j}. \tag{2}$$

This transform is invertible by construction (as all lifting structures) and is also critically sampled, since there is only one wavelet coefficient per node, thereby eliminating the inefficiency of the path merging technique in [3]. Defining the transform based on neighborhood relations on the routing tree facilitates a unidirectional implementation. Note that in (1) and (2) we explicitly separated terms corresponding to children and parent nodes. Clearly, a unidirectional approach would be such that a given node uses only data from its

ing to avoid path merges, but that leads to worse overall performance [3]

children, so there are no transmissions away from the sink (from a parent to a child). As shown in [3] for the 1D case and a 1-level decomposition, this can be achieved by allowing node n to compute a "partial" wavelet coefficient using its own data and data from its children. This partial coefficient is then forwarded to its parent ρ_n where the computation is finalized.

Let $d_p(m)$ and $s_p(n)$ be the partial coefficient for each odd and even node respectively. Assuming one level of decomposition, a generalized partial coefficient algorithm works as follows. For each $m \in \mathcal{P}$, $d_p(m) = x_m + \sum_{n \in \mathcal{C}_m} \mathbf{p}_m(n) x_n$, where the data x_n for each child n is recovered from the partial $s_p(n)$ as $x_n = [s_p(n) - \sum_{\bar{m} \in \mathcal{C}_n} \mathbf{u}_n(\bar{m}) d_{\bar{m}}]$. This is necessary since each child $n \in \mathcal{C}_m$ will forward the partial $s_p(n)$ to m instead of the raw data x_n (which requires more bits). For each $n \in \mathcal{U}$, do the following in this order. For all $m \in \mathcal{C}_n$ complete previous partials i) $d_m = d_p(m) + \mathbf{p}_m(n) x_n$ and ii) for each $\bar{n} \in \mathcal{C}_m$, $s_{\bar{n}} = s_p(\bar{n}) + \mathbf{u}_{\bar{n}}(m) d_m$. Then, generate $s_p(n) = x_n + \sum_{m \in \mathcal{C}_n} \mathbf{u}_n(m) d_m$.

This algorithm provides transform computation as data flows to the sink by computing coefficients over a small number of hops rather than at a single node. This eliminates backward transmissions, though it does incur some overhead since a few additional bits are allocated to each partial to mitigate the effects of quantization as in [5]. But from the partial coefficient computations above, it is apparent that these few bits are only carried 1-hop for odd partials and 2-hops for even partials, beyond which the coefficient is full and those few bits are dropped. This is clearly less costly than backward transmissions of full data/coefficients.

3. UNIDIRECTIONAL 2D TRANSFORM OPTIMIZATION

As discussed earlier, a mixture of coding schemes throughout the network may be more energy-efficient than just one scheme, resulting in a tradeoff between more local transmission cost for lower final transport cost [3, 9, 10]. In [4], a Dynamic Programming (DP) framework was developed that finds optimal coding scheme assignments in a path-wise manner. This guarantees optimality per path but not necessarily 2D optimality. In any case, the optimization formulation and algorithm in [4] does not directly extend to overlapping 1D paths so we must reformulate the problem for our transform.

Let $S = \{1, 2, \ldots, J\}$ index the set of coding schemes. For our transform, each $j \in S$ corresponds to a number of levels of decomposition. Let $\mathcal{J}_j(n)$ be the optimal cost to arrive at coding scheme j at node n from the children of n. Let $t^n_{i,j}$ be the cost to transition from coding scheme i at node n to scheme j at ρ_n. Each $t^n_{i,j}$ term captures each of the individual costs (i.e., transition and computation costs) detailed in [5]. We can formulate our forward DP problem as follows. For every node $n \in \mathcal{I}$ and for every $j \in S$:

$$\mathcal{J}_j(n) = \min_{\{i_m \in S : m \in K_n\}} \left\{ \sum_{m=1}^{M_n} t^{c_{n,m}}_{i_m,j} + \mathcal{J}_{i_m}(c_{n,m}) \right\}$$

The optimal costs must be found in a sequential manner and this can be done as in Algorithm 1. For every node n, Algorithm 1 also stores the optimal levels of the children of n that correspond to each $\mathcal{J}_j(n)$ as $\mathbf{t}(n, j) = (i^*_1, i^*_2, \ldots, i^*_{M_n})$. Given the optimal costs, define $\mathcal{I}_1 = \{m \in \mathcal{I} : \text{depth}(m) = 1\}$. For every $n \in \mathcal{I}_1$ compute $j^*_n = \arg\min_{j \in S}[\mathcal{J}_j(n)]$. Each j^*_n gives the optimal scheme for each depth one node and can therefore be used in conjuction with the $\mathbf{t}(m, j)$ vectors of the descendants of n to assign the optimal scheme to those descendants, all of which provide the optimal network.

Algorithm 1 Compute Optimal Costs

```
for k = max(depth) : −1 : 1 do
    I_k = {m ∈ I : depth(m) = k}
    for each n ∈ I_k do
        for each j ∈ S_n do
            Compute J_j(n) and t(n, j) = (i*_1, i*_2, ..., i*_{M_n})
        end for
    end for
end for
```

4. RESULTS AND DISCUSSION

For our performance evaluations, we compare against the path-wise transform with heuristic path merging strategy in [3] and the 2D transform in [7]. The transform in [7] splits nodes along a Delaunay tesselation, so their resulting transform neighbors are not always neighbors along a good routing tree T (as in our transform). This tesselation invokes a particular routing tree and an order in that tree, but that order may not be efficient since some backward transmissions will be needed. This is not an issue for our transform since data is always transmitted along a good routing tree T towards the sink. In any case, for that in [7], once a node receives all of its neighbors data/coefficients it computes its transform coefficient, quantizes it, and transmits it along the shortest path routing tree to the sink.

We consider two different network topologies that exhibit very different performance. The input data was generated using a second order AR model with poles and zeros chosen to provide a relatively smooth field. Figure 2 shows the optimal network topology for 200 nodes randomly deployed throughout a 600×600 grid and the associated performance curves. Figure 3 shows the same for a 2-cluster network with 100 nodes. For our transform, we only consider transform and routing along a shortest path routing tree.

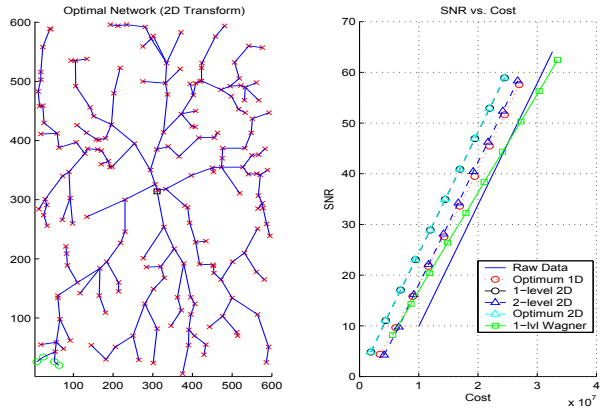

Fig. 2. Energy consumption comparison shown on the right. Optimal levels of decomposition for a uniform network shown on the left. Red x's denote 1-level nodes and green circles denote 2-level nodes.

For both networks, our optimal transform is significantly better than those in [3, 7]. As expected, our optimal transform is also superior to using either our 1-level or 2-level transform throughout the network. The performance of the 1-level transform is comparable to that of the optimal transform, mainly because so few nodes use 2-levels.

While the transform in [7] exploits 2D spatial correlation more effectively than ours, the cost is still higher. Their transform cost

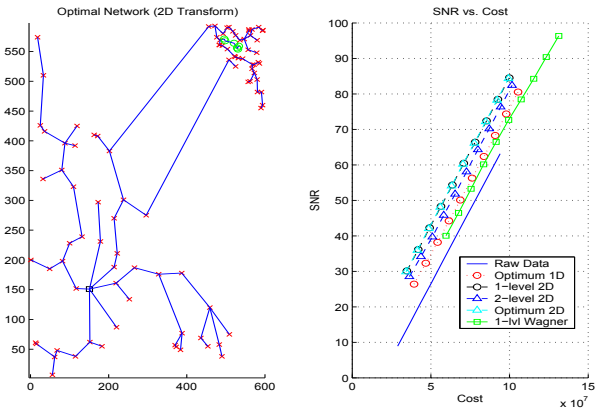

Fig. 3. Energy consumption comparison shown on the right. Optimal levels of decomposition for a uniform network shown on the left. Red x's denote 1-level nodes and green circles denote 2-level nodes.

is even worse than the raw data cost in some cases, a result corroborated in [8] (a follow-up of [7]). This is not as prominent in the 2-cluster network since the nodes in the top cluster are very close together, thereby increasing overall coding efficiency. Backward transmissions are also reduced since nodes in the top cluster do not have nodes in the bottom cluster as transform neighbors. This is not the case for the uniform network since nodes are not as tightly spaced.

In both networks, our transform is superior to those in [3, 7] since the unidirectional computation eliminates costly backward transmissions without sacrificing critical sampling and 2D data decorrelation. The same is true if we consider the average cost of each network for a fixed distortion. Figure 4 shows the cost averaged over 30 different uniform and 2-cluster networks for each number of nodes. Clearly, all considered versions of our transform outperform that of [7] in both cases. The optimal transform is also always the best.

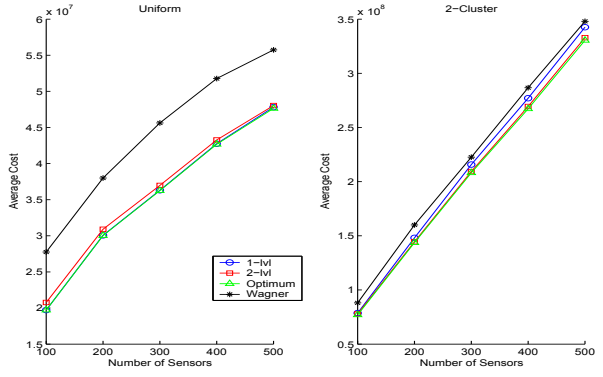

Fig. 4. Average Cost Comparison.

5. CONCLUSIONS

We have developed a 2D transform that can be computed in a unidirectional manner along an arbitrary routing tree. This allows the transform to exploit 2D spatial correlation beyond existing path-wise transforms without incurring the overhead of more general 2D transforms. The transform is also flexible enough to accomodate arbitrary 2D node deployments while maintaining simplicity of design and implementation. An optimization framework was also proposed that exploits the tradeoff between higher local cost for more complex coding in exchange for lower final transport cost. Simulation results have established the superiority of our proposed method over existing methods in terms of transform computation cost and coefficient transport cost. These improvements are mainly the result of the unidirectional computation of our 2D transform, which strongly suggests the need to permit unidirectional transform computation when designing distributed transforms for sensor networks so that the amount of savings gained from greater data correlation is not offset by excessively high local communication costs.

6. REFERENCES

[1] C. Chong and S. P. Kumar, "Sensor networks: Evolution, opportunities, and challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, August 2003.

[2] M. Gastpar, P. Dragotti, and M. Vetterli, "The distributed Karhunen-Loève transform," in *Proceedings of the 2002 International Workshop on Multimedia Signal Processing*, St. Thomas, US Virgin Islands, December 2002.

[3] A. Ciancio, S. Pattem, A. Ortega, and B. Krishnamachari, "Energy-efficient data representation and routing for wireless sensor networks based on a distributed wavelet compression algorithm," in *IPSN '06: Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*, New York, NY, USA, 2006, pp. 309–316, ACM Press.

[4] A. Ciancio and A. Ortega, "A dynamic programming approach to distortion-energy optimization for distributed wavelet compression with applications to data gathering inwireless sensor networks," in *Proc. of 2006 IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, ICASSP'06*, 2006.

[5] A. Ciancio, *Distributed Wavelet Compression Algorithms for Wireless Sensor Networks*, Ph.D. thesis, University of Southern California, 2006.

[6] S. D. Servetto, "Distributed signal processing algorithms for the sensor broadcast problem," *Conf. on Information Sciences and Systems, The Johns Hopkins University*, March 2003.

[7] R. Wagner, H. Choi, R. Baraniuk, and V. Delouille, "Distributed wavelet transform for irregular sensor network grids," in *IEEE Stat. Sig. Proc. Workshop (SSP)*, July 2005.

[8] R. Wagner, R. Baraniuk, S. Du, D.B. Johnson, and A. Cohen, "An architecture for distributed wavelet analysis and processing in sensor networks," in *IPSN '06: Proc. of the Fifth International Conference on Information Processing in Sensor Networks*, New York, NY, USA, 2006, pp. 243–250, ACM Press.

[9] J. Acimovic, B. Beferull-Lozano, and R. Cristescu, "Adaptive distributed algorithms for power-efficient data gathering in sensor networks," *Intl. Conf. on Wireless Networks, Comm. and Mobile Computing*, vol. 2, pp. 946–951, June 2005.

[10] R. Cristescu, B. Beferull-Lozano, and M. Vetterli, "Networked Slepian-Wolf: Theory and algorithms," *1st European Workshop on Sensor Networks EWSN 2004*, 2004, Berlin, Germany.

[11] W. Sweldens, "The lifting scheme: A construction of second generation wavelets," Technical report 1995:6, Industrial Mathematics Initiative, Department of Mathematics, University of South Carolina, (ftp://ftp.math.sc.edu/pub/imi_95/imi95_6.ps), 1995.