2011

Factoring Banded Permutations and Bounds on the Density of Vertex Identifying Codes on the Infinite Snub Hexagonal Grid

Chase A. Albert
College of William and Mary

Factoring Banded Permutations

&

Bounds on the Density of Vertex Identifying Codes on the Infinite Snub Hexagonal Grid

Chase Albert

Abstract

A permutation may be characterized as b-banded when it moves no element more than b places. Every permutation may be factored into 1-banded permutations. We prove that an upper bound on the number of tridiagonal factors necessary is $2b - 1$, verifying a conjecture of Gilbert Strang.

A vertex identifying code of a graph is a subset D of the graph's vertices with the property that for every pair of vertices v_1 and v_2, $N[v_1] \cap D$ and $N[v_2] \cap D$ are distinct and nonempty, where $N[v]$ is the set of all vertices adjacent to v, including v. We compute an upper bound of $1/3$ and a strict lower bound of $3/10$ for the minimum density of a vertex identifying code on the infinite snub hexagonal grid.

Contents

Acknowledgments

Thank you professors Chi-Kwong Li and Gexin Yu for your support and giving me the opportunity to research. Thanks to CSUMS for funding this research. Thank you Professors Virginia Torczon, Ryan Vinroot, and Gexin Yu for reviewing this thesis in committee. Your efforts are appreciated.

Chapter 1

Bounds on the Number of Tridiagonal Factors of a Banded Permutation

1.1 Introduction

Definition 1.1.1. *A b-banded permutation is a bijection π from $[n] = \{1, 2, \ldots, n\}$ to $[n]$, such that for all i from 1 to n, $|\pi(i) - i| \leq b$.*

We write a permutation π as $(\pi(1), \ldots, \pi(n))$. It may also be written as a permutation matrix $A = [a_{i\,j}]$, where $a_{i,j}$ is 1 if $j = \pi(i)$ and 0 otherwise.

Bandedness has direct interpretations in many representations of permutations. In the canonical matrix representation, a b-banded permutation has ones in only the 0th through bth diagonals, where we take the 0th diagonal

to be the main diagonal. The following is the matrix representation of the 3-banded permutation $(4, 5, 6, 1, 2, 3)$:

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}.$$

Permutations may be factored into products of other permutations. Of particular interest here will be 1-banded permutations, which swap disjoint pairs of adjacent elements. One may ask how few 1-banded factors are needed. The following theorem is the main consideration of this chapter, which was first offered as a conjecture by Gilbert Strang [7].

Theorem 1.1.2. *A b-banded permutation may always be factored into* $2b-1$ 1*-banded permutations.*

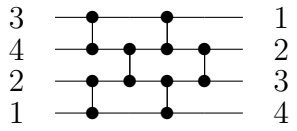
1.2 Sorting Networks

Definition 1.2.1. *A comparison network is a sequence of steps, such that on each step the network compares disjoint pairs of elements of a sequence, and swaps the elements if they are out of order.*

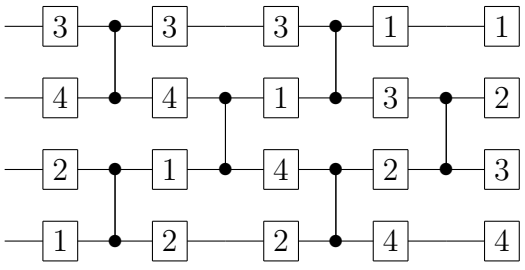
Definition 1.2.2. *A sorting network is a comparison network which sorts whatever sequence it is applied to.*

Sorting networks are discussed at length in [6]. The importance of sorting networks for proving Theorem 1.1.2 is that if we consider just the pairs of elements on a step which are swapped, together they help us construct a factorization of the permutation.

Here is an example of a sorting network. Consider the elements of the permutation $(3, 4, 2, 1)$, listed on the left of the network, as all traveling rightwards along the "wires". Where two wires are connected by a vertical bar, the elements currently on those wires are compared and, if they are out of order, swapped. On the right are the elements in order after sorting, arranged in the identity permutation.



If we watch the elements travel along the wires, this diagram becomes:



Sorting a permutation is exactly the problem of calculating its inverse. Each step of a sorting network can be seen as a factor of this inverse, if

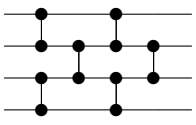
we view only the swaps that it actually performs. Inverses of products of 1-banded permutations may be calculated by simply reversing the order of the factors. Thus from the above sorting network we have that:

$$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Definition 1.2.3. *An n-line odd-even comparison network on every odd step compares the pairs of elements $(1,2),(3,4),\ldots,(2k-1,2k)$ for all $2k \leq n$. On every even step it compares the pairs of elements $(2,3),(4,5),\ldots,(2k,2k+1)$ for all $2k+1 \leq n$.*

An n-line even-odd comparison network in the odd steps compares the pairs of elements $(2,3),(4,5),\ldots,(2k,2k+1)$ for all $2k+1 \leq n$, and on the even steps compares the pairs of elements $(1,2),(3,4),\ldots,(2k-1,2k)$ for all $2k \leq n$.

The example network above, repeated below, is a 4-line odd-even *sorting* network.



1.3 Construction

Instead of proving Theorem 1.1.2, we prove the following stronger result about blocks of permutations, where block is defined below in Definition 1.3.2.

Theorem 1.3.1. *A block of a b-banded permutation can be sorted in at most* $2b-1$ *odd-even steps or* $2b-1$ *even-odd steps.*

1.3.1 Partitioning the Permutation into Blocks

A block of a permutation is a subpermutation that is a diagonal block in the matrix form of the permutation. More formally,

Definition 1.3.2. *A block of a permutation* $\pi : [n] \to [n]$ *is a subpermutation* $\sigma : \{i, i+1, \ldots, k\} \to \{i, i+1, \ldots, k\}$*, such that there is no* $j < i$ *where* $\pi(j) > k$*.*

As stated above, this is the same as splitting the matrix into block-diagonal form. For example,

$$\begin{pmatrix} 4, & 5, & 6, & 1, & 2, & 3, & 7, & 11, & 12, & 10, & 8, & 9 \end{pmatrix}$$

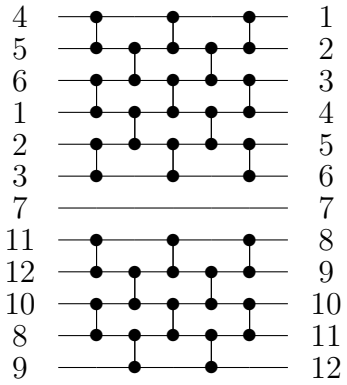
can be partitioned (equivalently) as:

$$\left(\begin{array}{cccccc|c|cccccc} 4, & 5, & 6, & 1, & 2, & 3 & 7 & 11, & 12, & 10, & 8, & 9 \end{array}\right)$$

or

$$\left(\begin{array}{cccccc|c|cccccc}
0&0&0&1&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&0&0&0&0&0&0&0&0\\
\hline
0&0&0&0&0&0&1&0&0&0&0&0\\
\hline
0&0&0&0&0&0&0&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1\\
0&0&0&0&0&0&0&0&0&1&0&0\\
0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0
\end{array}\right).$$

It is well known that the blocks of block-diagonal matrices may be factored independently. Were we to continue the application of Theorem 1.3.1 on this permutation, we would obtain the following comparison network:



1.3.2 Odd-Even or Even-Odd is Sufficient

Assume that we are sorting a block $p = (i_1, \ldots, i_n)$ of a b-banded permutation. If p is 1-banded, then it may be sorted (in $2 \cdot 1 - 1 = 1$ step) by simply swapping its two elements. Assuming that Theorem 1.1.2 holds for up to $(b-1)$-banded permutations, thus if p is less than b-banded we are done.

If p is b-banded, we will use one of the two simple sorting networks described in Definition 1.2.3. In [6, §5.3.4], Knuth uses the following property to show that a network of at most $|p|$ steps is necessary.

Theorem 1.3.3 (The Zero-One Principle)**.** *An n-line comparison network is a sorting network if and only if it sorts all length n sequences of zeros and ones.*

The following lemma and proof can be extracted from Theorem 1.3.3 and its proof as it appears in [6].

Lemma 1.3.4. *A comparison network sorts some permutation $(i_1, \ldots, i_n)$ if and only if, for all k, it sorts the sequences $(f_k(i_1), \ldots, f_k(i_n))$, where*

$$f_k(i) = \begin{cases} 0 & i < k \\ 1 & i \geq k \end{cases}$$

Proof. The forward direction is the case, as if the network sorts the permutation then all elements greater than or equal to any k must be moved the right of k, elements less than k must be moved to the left. The reverse direc-

tion is also true. Assume the network transforms the permutation $(i_1, \ldots, i_n)$ into some permutation for which two elements i_x and i_y remain out of order. Then $(f_y(i_1) \ldots, f_y(i_n))$ describes a binary sequence which is not sorted by the network. ■

In simple English Lemma 1.3.4 is just: a comparison network sorts a permutation only if it moves all of the smaller elements to the left of all of the larger elements, for every notion of "smaller" and "larger". This applies to comparison networks, and not general sorting methods, because a comparison network is fixed independently of the data, whereas a general sorting algorithm is free to make different comparisons as the data changes.

Using the permutation $(4, 2, 1, 7, 5, 3, 9, 8, 6)$ as an example, applying consecutive f_k's to its elements gives us:

$$f_0 \to \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$f_1 \to \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$f_2 \to \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

$$f_3 \to \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$f_4 \to \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$f_5 \to \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$$f_6 \to \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$$f_7 \to \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

$$f_8 \to \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$f_9 \to \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Lemma 1.3.5. *For a b-banded permutation, application of any f_k (from Lemma 1.3.4) yields a binary sequence structured as*

$$(0, \ldots, 0, \langle \textit{mixed section} \rangle, 1, \ldots, 1)$$

where the "mixed section" is composed of out of order elements and has a length at most $2b$.

Proof. Given some k, all elements $f_k(i_j) = 0$, where $j < k - b$. If one of these elements is 1, then the bandwidth cannot be b.
Likewise, all elements $f_k(i_j)$, where $j \geq k + b$ are 1.
So the length of the mixed section centered between $f_k(i_{k-1})$ and $f_k(i_k)$ is at most $2b$. ■

We can ignore any mixed sections or "windows" of length less than $2b$, as an odd-even or even-odd network of depth $2b - 1$ would be sufficient to sort any of them. As a sidenote, at this point we have proven that an odd-even or even-odd network of $2b$ steps will sort any b-banded permutation (not just

a block).

Lemma 1.3.6. *The following construction factors any mixed section (generated by an application of some f_k to a b-banded permutation) into at most $2b-1$ tridiagonal permutation matrices.*

We start from the identity binary sequence and work backwards:

$$(0,\dots,0,1,\dots,1) \to (f_k(i_{k-b})\dots,f_k(i_{k-1}),f_k(i_k),\dots,f_k(i_n)).$$

1. *Move the bth zero of the identity, via subsequent transpositions, to the rightmost zero's position in the f_k'd permutation.*

2. *Now move the $(b-1)$st zero the the next rightmost zero's position, starting from the second step.*

3. *For the $(b-k)$th zero, the rule is wait k steps, then apply rightward transpositions until that zero has reached the position of the $(b-k)$th zero in the permutation. This assumes that every other zero has moved at least one space, which must be the case, as there is a one in the first position.*

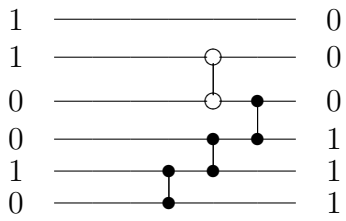
Proof. Once all of the zeros are in the correct place, the ones have to be. Additionally, no zero passes another in this construction, as the "next rightmost" zero is at least 1 space to the left of the previous zero. No zero moves further than b steps. That is, the bth zero moves exactly b steps; if the $(b-k)$th zero moves no further than position $2b-k$, the $(b-k-1)$st zero

can move no further than position $(2b - k - 1)$. Thus the entire network is at most depth $2b - 1$, as the first zero is in position $b - (b - 1)$ and can move at most b steps.

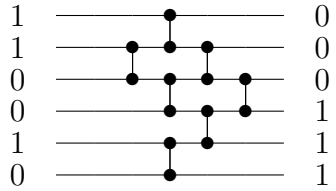
But, if the first zero moves to position $b + 1$, then the second zero must have moved to position $b + 2$, for it must move to the right of the first zero but can move no further than its bandwidth. The same applies to every other zero. This sequence, $(1, \ldots, 1, 0, \ldots, 0)$, exactly describes the length $2b$ sequence in which every element has moved its bandwidth, locally: $(b+1, b+2, \ldots, b+b, 1, 2, \ldots, b-1)$. This piece of the permutation necessarily forms a block (in the block diagonal representation of the permutation), and so would be separated out and sorted alone in $2b - 1$ steps.

In every other case, where the first zero moves at most $b - 1$ steps, the depth of the network required by the window is at most $2b-2$, and is covered by a (locally) odd-even network of depth $2b-1$ or a (locally) even-odd network of depth $2b - 2$. Since any block of the permutation can be sorted in at most $2b - 1$ steps, the entire permutation can be sorted in $\leq 2b - 1$ steps. ■

An example of binary sequence factorization is given below. This binary sequence could have been generated by the permutation sequence $(4, 5, 3, 2, 6, 3)$, but that is not the only one.



The next zero can be started immediately after the first, in the marked position above. It will not go past the first zero. The the full minimal comparison network for this particular sequence follows. Note that if we attempted to sort our example permutation $(4, 5, 3, 2, 6, 3)$ with this network, we would get the out of order sequence $(1, 3, 2, 4, 5, 6)$.



Chapter 2

Vertex Identifying Codes on the Infinite Snub Hexagon Grid

2.1 Introduction

Definition 2.1.1. *A vertex identifying code on a graph G, is a subset D of $V(G)$ which satisfies:*

1. *For any $v \in V(G)$, the intersection of the closed neighborhood $N[v]$ and D is nonempty.*

2. *For any vertices $v, w \in V(G)$, where $v \neq w$, $N[v] \cap D \neq N[w] \cap D$..*

An intuitive description might be that we label only some vertices of a graph, and can identify any vertex by calling off the labels of itself and its neighbors, ignoring order.

As a simple example, if we have the 3-path:

where ◯ is taken to be unlabeled, the left and right vertices are in the code D. Then each vertex can be identified by only its labeled neighbors. The leftmost vertex has $\{1\}$, the middle $\{1, 2\}$, and the rightmost $\{2\}$.

Not all graphs have a vertex identifying code. If some two vertices of a graph have the same closed neighborhood then no code can be assigned, as those two vertices will always be identified as the same. Only when the closed neighborhoods of every vertex are all distinct can a vertex identifying code be constructed. Then $V(G)$, at least, is a vertex identifying code for G. We then turn our attention to finding codes of minimal size, or in the infinite case, of minimal density, defined below.

Definition 2.1.2. *The density of a vertex identifying code D on graph G, $\sigma(D, G)$ is the ratio of vertices in the code to vertices in the graph. $v \in V(G)$.*

$$\sigma(D, G) = \limsup_{k \to \infty} \frac{\left| \bigcup_{i=0}^{k} N_i[v] \cap D \right|}{\left| \bigcup_{i=0}^{k} N_i[v] \right|}$$

Then the minimum density of a vertex identifying code for G may be defined as follows:

$$\sigma_0(G) = \min_{D \subset G} \sigma(D, G)$$

Karpovsky et al. [5], who introduced the problem of finding vertex identifying codes, they consider codes for the infinite triangular, square, and hexagonal grids; they prove that the minimum density for a code on the infinite triangular grid is $1/4$. Cohen et al. [3] later provided an upper bound of $7/10$ for the density of a code on the infinite square grid, this bound was shown to be exact in [1]. The latest work on the infinite hexagonal grid places the minimum density between $5/12$ and $3/7$ [4].

We focus on finding bounds on the minimum density of codes on the infinite snub hexagonal grid, as shown (with a code) in Figure 2.1. As our graph is 5-regular, we immediately know from [3] that $2/7$ is a lower bound on the density. In this chapter we will show that the minimum density for a code on the infinite snub hexagonal grid is between $3/10$ and $1/3$.

In the following sections, we will give a construction for the upper bound of $1/3$, use the discharging method to prove a lower bound of $3/10$, provide a method of automating the search for a lower bound via integer linear programming, and finally we will use a computer search based on this model to show that the our provided lower bound of $3/10$ is strict.

2.2 An Upper Bound

An upper bound was found for this code via construction. This construction has an interesting property we will use later to prove the strictness of our lower bound. The code is such that each of the triangles of vertices which

touch three hexagons have exactly one vertex in the code (Figure 2.1), thus its density is $1/3$. Note that in this construction no two vertices in the code are adjacent.

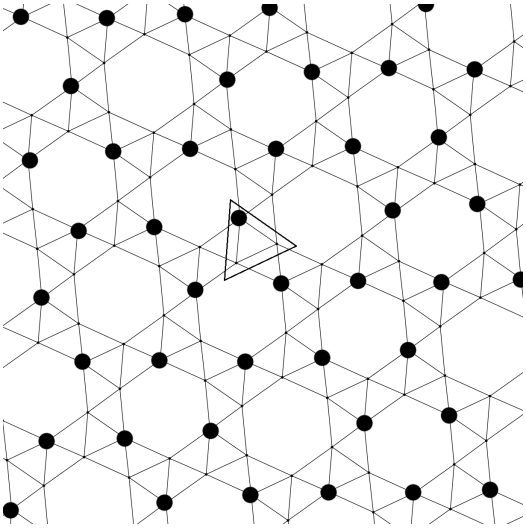


Figure 2.1: A vertex identifying code on the infinite snub hexagonal grid.

2.3 A Lower Bound

We prove our lower bound with the discharging method. Consider each vertex in the code as possessing 1 "charge", and each vertex not in the code as possessing no charge. If we can always redistribute this charge among all the vertices such that they each have at least τ charge, then by the Definition 2.1.2, τ is a lower bound on the density of the graph.

In the the discharging method we design a discharging rule to determine how the charge is redistributed. We then verify that in every case, after

application of this rule, each vertex has at least τ charge. In our proof we use the following discharging rule:

Rule 1: If a vertex v is not in D and has k neighbors in D, then v receives $\frac{3}{10k}$ charge from each of those neighbors.

Now we consider the final charge of each vertex.

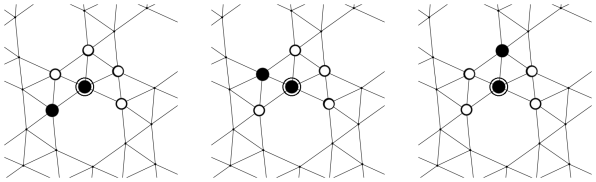
If a vertex v is not in the code, then it has $k \geq 1$ neighbors in the code and receives $k \cdot \frac{3}{10k} = \frac{3}{10}$ charge.

If v is in the code we consider several cases.

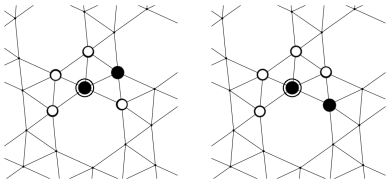
If v has no neighbors in the code. Then each vertex adjacent to v must have at least one more neighbor in the code (or its code would be just $\{v\}$, but that is v's code). Of the five vertices in the open neighborhood $N[v]\backslash\{v\}$, only one vertex has that all its neighbors are either members of $N[v]$ or in the neighborhood of one of the other vertices in $N[v]$. This vertex must have a neighbor in D which is also the neighbor of another vertex in $N[v]$. Then at least one vertex in $N[v]$ is adjacent to 3 vertices in D. Then v gives at most $1/10$ charge to one neighbor, and gives at most $3/20$ charge to four neighbors. Therefore the final charge on v is:

$$1 - \frac{1}{10} - 4 \cdot \frac{3}{20} = \frac{3}{10}$$

If v has one adjacent vertex in the code there are several configurations to consider. Here I diagram them all. Full circles are in the code, hollow circles are outside the code. Each of these diagrams is explored for the worst



(a) $3/10$ (b) $6/19$ (c) $6/19$



(d) $12/37$ (e) $3/10$

Figure 2.2: In each diagram we have at least $3/10$ charge remaining on v.

case charge remaining on v, in every case note that this is at least $3/10$.

If v has more than one adjacent vertex in the code then it keeps at least $\frac{9}{25} > \frac{3}{10}$ charge.

So $\frac{3}{10}$ is a lower bound on the density of a code for the snub hexagon grid.

2.4 Automated Proof

We can format the discharging method as an integer linear program, and perform a computer search for possible lower bounds on the minimum density.

We are given a regular graph G, some subset of vertices A, and a function $\mathcal{N} : V \to \mathcal{P}(V)$. Our discharging rule is that each vertex will receive an equal fraction of charge from every vertex in $\mathcal{N}(v)$.

Our variables (all binary variables) are as follows:

1. For all v in G, x_v is 1 if v is in the code and 0 if it is outside the code.

2. $p_{v\,i}$ is 1 if v has i neighbors in the code, 0 if it does not.

3. $p_{v\,w\,i}$ is 1 only if v is in the code and w has i neighbors.

We must write linear constraints that force these variables to accept these values.

First we constrain the variables x_v such that all of G is coded for.

1. For every v in G, $D \cap N[v] \neq \emptyset$. Then the sum of all the variables in the neighborhood of v must be at least 1, since at least one of them must be in D.

$$\sum_{w \in N[v]} x_w \geq 1$$

2. For all pairs of vertices u and v in G, we have that $D \cap N[u] \neq D \cap N[v]$, that is, $D \cap N[u]$ and $D \cap N[v]$ differ by at least one element.

$$\sum_{w \in N[u] \,\Delta\, N[v]} x_w \geq 1$$

We will running this linear program for each possible size of $A \cap D$, so we add a parameter k and the constraint:

$$\sum_{v \in A} x_v = k.$$

We constrain $p_{v\,i}$ to be the population boolean for the discharging area

$\mathcal{N}(v)$ of every vertex v. We observe that:

$$\sum_{w \in \mathcal{N}(v)} x_w + \sum_{i=0}^{|\mathcal{N}(v)|} (|\mathcal{N}(v)| + 1 - i) \cdot p_{v\,i} + x_v + f_v + f'_v + f''_v + \cdots = |\mathcal{N}(v)| + 1,$$

$$x_v + \sum_{i=0}^{|\mathcal{N}(v)|} p_{v\,i} = 1,$$

where the f_v variables are filler (they have no direct interpretation as part of the model) which allow for the sum to be an equality in the case that v is in the code. They have the additional constraint that $x_v \geq f_v \geq f'_v \geq f''_v \geq \ldots$. We see that if v is in the code (in which case we do not care how many neighbors it has) any $p_{v\,i}$ where $i < |\mathcal{N}(v)|$ may be 1, and several filler may be. If v is not in the code, then no f_v may be 1, $x_v = 0$. If v has k neighbors in the code, then an additional $(|\mathcal{N}(v)| + 1 - k)$ is needed to satisfy the equality, and this is the coefficient of only $p_{v\,i}$. No two smaller $p_{v\,i}$ may be 1 together.

We constrain $p_{v\,w\,i}$ to be $x_v \cdot p_{v\,i}$.

Now we use these variables to obtain a linear objective function which may be minimized such that we get an expression from which we can derive τ. We let k run from 1 to $|A|$, over the course of several runs. We attempt to minimize the charge coming into A so we know the worst case for a given

configuration of vertices A.

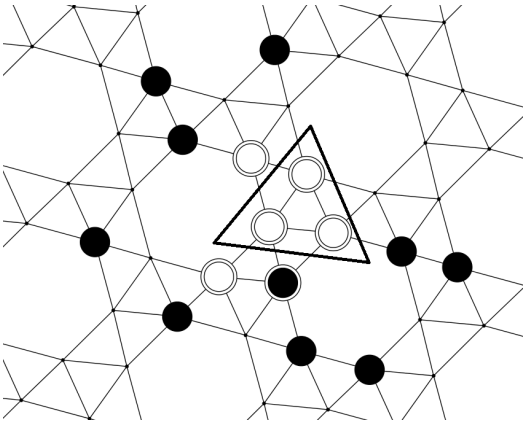
$$\min \sum_{v \in A} \overline{x_v} - \sum_{v \in A} \left(\sum_{w \in \mathcal{N}(v) \setminus \{v\}} \left(\sum_{i=1}^{|\mathcal{N}(v)|} \frac{1}{i} \right) \right)$$

The value of the objective function at the minimum is $-k/\tau_k$. The upper bound on the minimum density, τ, for this area A and rule function $\mathcal{N}$, is then the minimum of these τ_k for all k from 1 to $|A|$.

2.5 The Lower Bound is Strict

We used a computer implementation of the previous section to confirm the $3/10$ lower bound and to perform an additional discharging (of radius 2) on the double-circled area in Figure 2.3. The triangled area was held empty.

Figure 2.3: A construction of minimal τ.



The minimum density diagrammed in this figure is $15/49$, slightly above $3/10$. Since we know that if there are no empty triangles the minimum possible

density is $1/3$, and we now know that if we include even a single empty triangle the density must at least locally be $15/49$, and everywhere else must be at least $3/10$, no construction of density $3/10$ can exist.

Chapter 3

Future Research

Although it is not fully discussed in this paper, the method shown in Lemma 1.3.6 transposition sorts binary sequences optimally. It is reasonable to think we could scan along for the window which takes the longest to sort and line our odd-even network up with that, but it's not always the case. If there are two such windows, they don't always line up. Often it is possible to merge the networks over the two windows, but I would like to have a complete classification for when this is possible.

The entirety of Chapter 1 can be viewed as a special case of the routing number problem [8]. In that problem the allowable swaps are pairs of positions, described as edges of a graph. The graph for our problem is a path—only adjacent elements may be swapped. It's perhaps easy to imagine being additionally able to swap the first and last element (this graph would be a cycle), or being able to swap any odd element with any even element,

&c. Our solution for transposition networks seems to have little relevance for the more general problem.

Several extensions are evident for Chapter 2. Certainly the automated part of the work can handle completely general finite graphs and general infinite repeating graphs, however we were limited in our ability to experiment with the more computationally expensive discharging rules. Implementing this method for use on a greater amount of computation resources than was available would have been ideal.

On a general note, the discharging method in its current form is inadequate for proving very tight lower bounds, as many good constructions for upper bounds use only several repeating elements, and it's expected that codes of minimal density will as well.

I conjecture that the problem of finding a code of a particular density for a particular infinite grid is undecidable. It bears some superficial resemblance to the Wang tile problem ([2][9], the problem of tiling the infinite plane given a set of four-sided puzzle pieces is equivalent to determining if a corresponding Turing machine halts), and does appear to be computationally very difficult.

Bibliography

[1] Y. Ben-Haim, S. Litsyn, Exact minimum density of codes identifying vertices in the square grid, *SIAM J. Discrete Math.*, 19 (2005), no. 1, 69-82.

[2] R. Berger, The undecidability of the domino problem, *Memoirs Amer. Math. Soc.* 66 (1966).

[3] G. Cohen, S. Gravier, I. Honkala, A. Lobstein, M. Mollard, C. Payan, G. Z´mor, Improved Identifying Codes for the Grid, Comments on R19, *Electronic Journal of Combinatorics*, Vol. 6 (1999).

[4] A. Cukierman, G. Yu, New Bounds on the Minimum Density of a Vertex Identifying Code for the Infinite Hexagonal Grid, *Submitted* (2011).

[5] M.G. Karpovsky, K. Chakrabarty, L.B. Levitin, On a new class of codes for iden- tifying vertices in graphs, *IEEE Trans. Inform. Theory* 44 (1998) 599-611.

[6] D. E. Knuth, Art of Computer Programming, Volume 3: Sorting and Searching (2nd Edition), *Addison-Wesley* (1998).

[7] G. Strang, Fast Transforms: Banded Matrices with Banded Inverses, *Proc. Natl. Acad. Sciences*, Vol. 107 (2010), 12413-12416.

[8] W. Li, L. Lu, and Y. Yang, Routing Numbers of Cycles, Complete Bipartite Graphs, and Hypercubes, *SIAM J. Discrete Math.* 24 (2010), no. 4, 1482–1494.

[9] H. Wang, Proving theorems by pattern recognition—II, *Bell System Tech. Journal* 40 (1961).