

Automated Discharging Arguments for Density Problems in Grids (Extended Abstract*)

Derrick Stolee
Department of Computer Science
Department of Mathematics
Iowa State University
dstolee@iastate.edu

July 8, 2015

Abstract

Discharging arguments demonstrate a connection between local structure and global averages. This makes it an effective tool for proving lower bounds on the density of special sets in infinite grids. However, the minimum density of an identifying code in the hexagonal grid remains open, with an upper bound of $\frac{3}{7} \approx 0.428571$ and a lower bound of $\frac{5}{12} \approx 0.416666$. We present a new framework for producing discharging arguments using an algorithm. This algorithm replaces the lengthy case analysis of human-written discharging arguments with a linear program that produces the *best possible* lower bound using the specified set of discharging rules. We use this framework to present a lower bound of $\frac{23}{55} \approx 0.41818$ on the density of an identifying code in the hexagonal grid, and also find several sharp lower bounds for variations on identifying codes in the hexagonal, square, triangular, and pentagonal grids. We also present a new method to find matching upper bounds.

*Pages 1-10 of this PDF contain the extended abstract, with some figures, tables, and proofs appearing in appendices (pp. 13-16).

1 Introduction

The discharging method is a well-known technique in discrete mathematics, especially due to its use in the computer-assisted proofs of the Four Color Theorem [1,2,30]. Since that first incredible achievement, almost all other discharging proofs have been verified manually. Applications of discharging include coloring planar graphs [6], density problems in grids [7,8], and structural problems on circulant graphs [11]. Despite its wide use, producing an effective discharging argument is very challenging and the proofs become complicated by a lengthy case analysis. We present an algorithmic method for producing discharging arguments and apply this method to prove lower bounds on the density of sets in infinite grids.

A *plane grid* is an infinite graph embedded in the plane. We will consider four plane grids: the hexagonal, the square, triangular, and pentagonal grids, as shown in Figure 1. These grids model the structure of a wireless sensor network where the nodes are placed in a rigid lattice, as would be typical for use in a field for drought monitoring [9]. Due to the low cost of wireless sensor nodes, these networks can be so large that the boundary of the network is a small portion of the entire network, so using an infinite grid is an effective way to approximate the network. Karpovsky, Chakrabarty, and Levitin [21] considered the problem of detecting faults in such a network and defined an *identifying code* to be a set X of vertices in the grid where the intersection of X with the closed neighborhood of each vertex is distinct. If $N(v)$ is the set of vertices adjacent to v , then the closed neighborhood $N[v]$ is the set $N(v) \cup \{v\}$. Thus an identifying code X in a grid G satisfies $N[v] \cap X \neq \emptyset$ for all $v \in V(G)$ and $N[v] \cap X \neq N[u] \cap X$ for all distinct vertices $u, v \in V(G)$.

An identifying code exists in a graph G if and only if there are no *twins* (distinct vertices u, v where $N[u] = N[v]$) since using the entire vertex set can identify all vertices. The interesting problem is to determine the *smallest* identifying code in G , to minimize the cost of placing fault-detection devices on the nodes representing elements of the code.

In an infinite grid, any identifying code will be infinite, so we need a notion of density instead of size. For a vertex v , let $B_r(v)$ be the set of vertices within distance r of v in G . Given a vertex $v \in V(G)$, The *density* of a set $X \subseteq V(G)$, denoted $\delta(X; v)$, is defined as a limit of the proportion of elements of X in the ball of radius r , as r grows: $\delta(X; v) = \limsup_{r \rightarrow \infty} \frac{|B_r(v) \cap X|}{|B_r(v)|}$. In the grids we consider, the vertex v is irrelevant and we use $\delta(X)$ in place of $\delta(X; v)$.

Cohen *et al.* [5] demonstrated an identifying code in the hexagonal grid of density $\frac{3}{7}$, and Cukierman and Yu [8] found several other constructions of this density. However, lower bounds on the optimal density have not reached this upper bound, despite several attempts [5,7,8,21], with the most recent lower bound of $\frac{5}{12}$ by Cukierman and Yu [8].

Theorem 1. *If X is an identifying code in the hexagonal grid, then $\delta(X) \geq \frac{23}{55} = 0.41\overline{8}$.*

We prove Theorem 1 using a new computer-automated method for constructing discharging arguments.

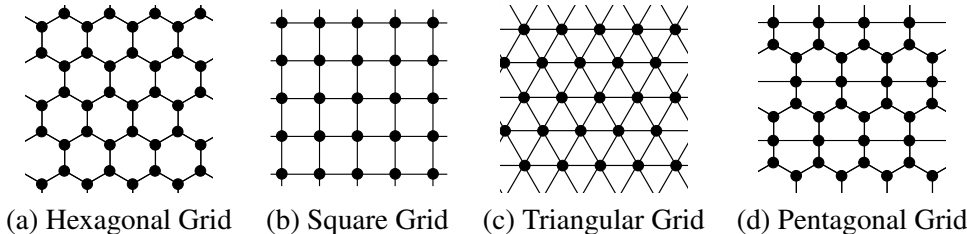


Figure 1: Examples of plane grids.

Due to the generality of the method and the modular nature of the implementation, the method also demonstrates lower bounds for other grids and for other variants on identifying codes. Thus, several sharp lower bounds are proven for these variants in the hexagonal, square, and triangular grids, which were major theorems of previous work [3, 16, 21, 22, 31, 33].

Our main contribution is the development of this new computational approach to producing discharging arguments. In Section 3, we discuss the structure of our discharging arguments. A recent survey by Cranston and West [6] includes a more general perspective, but focuses mostly on coloring planar graphs. Briefly, discharging arguments all feature three main steps: (1) assigning initial charge, (2) distributing charge according to certain discharging rules, and (3) verifying that all objects have a certain amount of charge. The most novel contribution of this new framework is that steps (2) and (3) are done in opposite order. By using a combinatorial generation algorithm, we enumerate every possible way that the discharging rules can interact on a single object and create a linear program based on those interactions. Every feasible solution corresponds to a correct discharging argument, and an optimal solution provides the largest lower bound possible using that set of rules. This pairing of combinatorial generation and linear programming is similar to the use of generating and solving a semidefinite program in Razborov’s flag algebra method [27], which has gained significant attention in recent years (see Razborov’s survey [28]).

The purpose of this extended abstract is to demonstrate how a custom computer algorithm can produce a discharging argument using minimal human interaction. We name our method *ADAGE* for “Automated Discharging Arguments using GEneration,” and a specific proof using the framework is an *adage*. While an adage is a short saying that conveys a general truth, an adage proof has a very short description while the computer handles the significant case analysis. In Section 2 we discuss the important properties of the plane grids in more detail, followed by a definition of a *configuration* and *forbidden configuration*. In Section 3 we set up the discharging argument, demonstrate how it is linked to the density of a set X , and discuss the structure of discharging rules. The most crucial step is discussed in Section 4, where a linear program is built to satisfy the assertions of the discharging argument. Section 5 defines several variations on an identifying code, and the results on these variations are listed in Table 1. In Section 6, we use the resulting discharging arguments to search for sets whose density matches the lower bound presented by the discharging argument. Using this technique, we find matching examples for some problems. Appendices ?? and B list several options for possible discharging rules in the three grids and the lower bounds demonstrated by the adage proofs using those rules.

2 Grids, Density, and Configurations

We shall use standard graph theory terminology (see West [36]) to treat a grid G as an infinite plane graph with vertex set $V(G)$, edge set $E(G)$, and face set $F(G)$.

The grids G have an automorphism group as graphs, but we will restrict the automorphisms to be affine linear maps on the plane with determinant 1. Specifically, we allow rigid motions that are translations and rotations, but do not allow reflection (as such maps would have determinant -1). We make this restriction based on intuition since previous discharging arguments in grids have made use of distinguishing between clockwise and counter-clockwise arrangements, which would be lost if we allowed reflection. Under these automorphisms, all four grids are face-transitive, and all but the pentagonal grid are vertex-transitive. If we did not allow rotation, then the hexagonal grid would not be vertex-transitive and the triangular and pentagonal grids would not be face-transitive.

For a vertex $v \in V(G)$ and an integer $r \geq 0$, the *ball of radius r around v* is the set of vertices within distance r of v and is denoted by $B_r(v)$. The *faces within distance r of v* is the set of faces incident to vertices in $B_{r-1}(v)$ and is denoted by $F_r(v)$. For a face $f \in F(G)$, define $B_r(f)$ to be the vertices v where $f \in B_r(v)$ and $F_r(f)$ to be the faces incident to vertices in $B_{r-1}(f)$.

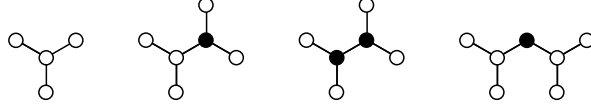


Figure 2: Forbidden configurations for identifying codes in the hexagonal grid. Black vertices are elements of S_1 and white vertices are not elements of S_0 .

A grid is *amenable* if the maximum degree of a vertex in G is finite, the maximum length of a face in G is finite, and $\lim_{r \rightarrow \infty} \frac{|B_{r+d}(v) \setminus B_r(v)|}{|B_r(v)|} = 0$ for all finite values $d \geq 0$. Amenable grids have the property that the boundary of a ball is a vanishing proportion of the volume of the ball as the radius of the ball grows. Using this basic property, it is not difficult to make the following observation.

Observation 2. *Let G be an amenable plane grid. Then, for $u, v \in V(G)$, $\lim_{r \rightarrow \infty} \frac{|B_r(u) \cap B_r(v)|}{|B_r(u) \cup B_r(v)|} = 1$, and hence for any set $X \subseteq V(G)$,*

$$\delta(X, u) = \limsup_{r \rightarrow \infty} \frac{|B_r(u) \cap X|}{|B_r(u)|} = \limsup_{r \rightarrow \infty} \frac{|B_r(v) \cap X|}{|B_r(v)|} = \delta(X, v).$$

By this observation, we can define one vertex in $V(G)$ as the “zero vertex,” denoted by v_0 , and define the *density* of a set $X \subseteq V(G)$ to be the limit $\delta(X) = \limsup_{r \rightarrow \infty} \frac{|B_r(v_0) \cap X|}{|B_r(v_0)|}$. We also define one face in $F(G)$ as the “zero face” denoted by f_0 .

A *configuration* is a tuple (V, S_0, S_1, F) where V is a finite set of vertices in $V(G)$, S_0 and S_1 are disjoint subsets of V , and F is a finite set of faces in $F(G)$. We call S_1 the *elements* and S_0 the *nonelements*. While we require that $S_0 \cap S_1 = \emptyset$, we do not require that $S_0 \cup S_1 = V$. Vertices in $V \setminus (S_0 \cup S_1)$ are considered *undetermined vertices*. Frequently, we will denote a configuration by C and refer to the entries of the tuple (V, S_0, S_1, F) by $V(C)$, $S_0(C)$, $S_1(C)$, and $F(C)$. The automorphism group of G naturally acts on configurations to produce a notion of isomorphism between configurations.

Such a configuration C represents a finite induced subgraph of the grid G and its planar dual G^* , as well as some information about X on that induced subgraph. Specifically, a configuration C is *embedded* in X if $S_1(C) \subseteq X$, and $S_0(C) \subseteq V(G) \setminus X$. Further, C is *embeddable* in X if there exists a configuration C' isomorphic to C such that C' is embedded in X .

Many families of subsets of $V(G)$, such as identifying codes, can be defined in terms of *forbidden configurations*. Given a collection $\mathcal{F} = \{C_1, \dots, C_k\}$ of configurations, the family $\text{forb}(\mathcal{F})$ consists of sets $X \subseteq V(G)$ where for every $C_i \in \mathcal{F}$, the configuration C_i is not embeddable in the set X . For a single configuration C , we use $\text{forb}(C)$ to denote $\text{forb}(\{C\})$.

For example, a *dominating set* is a set $X \subseteq V(G)$ such that $N[v] \cap X \neq \emptyset$ for all vertices $v \in V(G)$. If $v \in V(G)$ and C is the configuration with $V(C) = S_0(C) = N[v]$, then $\text{forb}(C)$ is the family of dominating sets in G . Observe that for the family \mathcal{F} of configurations in Figure 2, $\text{forb}(\mathcal{F})$ is the family of identifying codes in the hexagonal grid.

For a family $\mathcal{F} = \{C_1, \dots, C_k\}$ of forbidden configurations and a configuration C where $S_0(C) = S_1(C) = \emptyset$, we say that a configuration C' is an \mathcal{F} -realization of C if $V(C') = V(C)$, $S_0(C') \cup S_1(C') = V(C')$, and C' does not contain any configuration $C_i \in \mathcal{F}$. It is not difficult to generate all \mathcal{F} -realizations of a configuration C up to isomorphism using standard techniques. If $\mathcal{F} = \emptyset$, then there are $2^{|V(C)|}$ realizations of C , and possibly fewer when $\mathcal{F} \neq \emptyset$.

We will use this method of generating \mathcal{F} -realizations of a configuration C to examine all cases of how an embedding of C in the grid G can intersect a set X . But first, we must discuss the structure of our discharging argument.

3 Charge Assignment and Discharging Rules

For our discharging argument, we will consider vertices and faces of G to be *chargeable objects* in that we will associate them with a numerical value, called a *charge*. We assign a charge function $\mu: V(G) \rightarrow \mathbb{R}$ on the vertices of G and a charge function $\nu: F(G) \rightarrow \mathbb{R}$ on the faces of G . These functions are based on the positions of the elements in a set $X \subseteq V(G)$: for every vertex $v \in V(G)$, $\mu(v) = \begin{cases} 1 & v \in X, \\ 0 & v \notin X \end{cases}$, and for every face $f \in F(G)$, let $\nu(f) = 0$. Observe that $\sum_{v \in B_r(v_0)} \mu(v) + \sum_{f \in F_r(v_0)} \nu(f) = |X \cap B_r(v_0)|$ and hence

$$\delta(X) = \limsup_{r \rightarrow \infty} \frac{|X \cap B_r(v_0)|}{|B_r(v_0)|} = \limsup_{r \rightarrow \infty} \frac{\sum_{v \in B_r(v_0)} \mu(v) + \sum_{f \in F_r(v_0)} \nu(f)}{|B_r(v_0)|}.$$

A *discharging function* is a function $D_X: (V(G) \cup F(G)) \times (V(G) \cup F(G)) \rightarrow \mathbb{R}$ where $D_X(x, y) = -D_X(y, x)$ for all $x, y \in V(G) \cup F(G)$. Specifically, we can say that for two chargeable objects $x, y \in V(G) \cup F(G)$, the value $D_X(x, y)$ is the amount of charge to *exchange from x to y* . Given a discharging function D_X , we define the resulting charge functions $\mu': V(G) \rightarrow \mathbb{R}$ and $\nu': F(G) \rightarrow \mathbb{R}$ as follows:

$$\begin{aligned} \mu'(v) &= \mu(v) + \sum_{u \in V(G)} D_X(u, v) + \sum_{g \in F(G)} D_X(g, v). \\ \nu'(f) &= \nu(f) + \sum_{u \in V(G)} D_X(u, f) + \sum_{g \in F(G)} D_X(g, f). \end{aligned}$$

For values $c, d > 0$, we say that D_X is (c, d) -*local* if $|D_X(x, y)| \leq c$ always, and $D_X(x, y) = 0$ whenever the distance between x and y in G exceeds d . If a discharging function D_X is (c, d) -local and the grid is amenable, then the change in the total charge within the ball is negligible compared to the total charge of the ball.

Our main assertion for a “good” discharging function is that the resulting charge functions satisfy $\mu'(v) \geq w$ and $\nu'(f) \geq 0$ for all vertices $v \in V(G)$ and faces $f \in F(G)$. Roughly, this means that the initial charge on the vertices was “spread out” evenly so that every vertex has charge at least w , and the faces did not contribute any positive charge to the vertices and instead were simply “messengers” of charge. In the hexagonal grid, passing charge between vertices and faces can be particularly effective, since a face is incident to three antipodal pairs of vertices.

We make this assertion of a good discharging function concrete in the following theorem.

Theorem 3. *Let G be an amenable grid. Let $X \subseteq V(G)$, $c, d, w \geq 0$, and let D_X be a (c, d) -local discharging function. Define the charge functions μ, μ', ν, ν' by the discharging process using X and D_X . If $\mu'(v) \geq w$ for all $v \in V(G)$ and $\nu'(f) \geq 0$ for all $f \in F(G)$, then $\delta(X) \geq w$.*

This theorem follows from basic limit computations. A proof is included in the appendix.

Theorem 3 demonstrates that (c, d) -local discharging functions provide a way to bound the density of a set X . We consider $\mu'(v) \geq w$ and $\nu'(f) \geq 0$ to be *postconditions* for the discharging process (where the initial charge values form *preconditions*). However, as defined, the function D_X depends on the entire (possibly infinite) set X . This is not an effective strategy for us to prove anything about a discharging function. In order to build *effective* discharging functions, we will assemble one using *discharging rules*.

Informally, a discharging rule is a way to examine the local situation around a chargeable object, and then decide to exchange a certain amount of charge among nearby chargeable objects. Such a rule could, for instance, consider which elements in $B_2(v)$ are in X , and use that information to exchange charge between v and the vertices adjacent to v , or between v and the faces incident to v . If the amount of charge exchanged

depends only on the isomorphism class of the configuration $(B_2(v), B_2(v) \setminus X, B_2(v) \cap X, F_1(v))$, then this rule has a finite description, even though it is applied an infinite number of times. When several discharging rules are applied simultaneously, the discharging function D_X is defined by collecting all of the charge exchanges from all instances of the discharging rules.

Formally, a *discharging rule* is a tuple $R = (C, z, y_1, \dots, y_t, \sigma)$ where C is a configuration, z, y_1, \dots, y_t are chargeable objects in C , and σ is a function $\sigma : \{0, 1\}^{V(C)} \times \{y_1, \dots, y_t\} \rightarrow \mathbb{R}$. We will consider the first parameter of σ to be the incidence vector corresponding to the set $X \cap V(C)$. Thus $\sigma(X \cap V(C), y_i)$ determines how much charge to exchange from y_i to z , given the realization of C . For an embedding π of C into G , the rule considers $\pi^{-1}(X) \cap V(C)$ and the function σ defines that some amount of charge is exchanged from each $\pi(y_i)$ to $\pi(z)$. Thus, the rule defines a discharging function D_X^R as

$$D_X^R(a, b) = \sum_{\substack{\pi : \pi(z) = b \\ i : \pi(y_i) = a}} \sigma(\pi^{-1}(X) \cap V(C), y_i) - \sum_{\substack{\pi : \pi(z) = a \\ i : \pi(y_i) = b}} \sigma(\pi^{-1}(X) \cap V(C), y_i).$$

The above definition states that the amount of charge sent from a to b is the combination of the charge sent from a to b via all embeddings of the rule where $\pi(z) = b$ and $\pi(y_i) = a$ for some i , minus the charge sent from b to a via all embeddings of the rule where $\pi(z) = a$ and $\pi(y_i) = b$ for some i .

If R_1, \dots, R_m is a list of rules, then the discharging function D_X resulting from using these rules simultaneously is defined as $D_X(a, b) = \sum_{i=1}^m D_X^{R_i}(a, b)$.

We can very quickly describe the configuration C and chargeable objects z, y_1, \dots, y_t of a discharging rule. The function σ is more complicated, and in fact we do not specify it at all. For each possible element of the domain of σ , we create a variable. In the next section, we will describe how to create a linear program to assign value to these variables, thereby completely defining the discharging rules.

We now describe a few discharging rules that can be defined for any grid. The following list carefully defines each rule, but these rules can be simply described for the hexagonal grid by Figure 3.

- $D_i R_j$: Let z be a vertex and y a vertex with $d(z, y) = i$. Consider the configuration $B_j(z) \cap B_j(y)$. Use the information from $X \cap B_j(z) \cap B_j(y)$ to specify how much charge is exchanged from y to z .
- V_i : Let z be a vertex, and consider the configuration on $B_i(z)$ and $F_1(z)$ and let $\{y_1, \dots, y_t\} = F_1(z)$. Use the information from $X \cap B_i(z)$ to specify how much charge is exchanged from each y_j to z .
- N : Let z be a vertex and consider the configuration C on $F_1(z)$ that contains all vertices incident to the faces in $F_1(z)$. Use the information from $X \cap C$ to specify how much charge is exchanged from each incident face to z . In the hexagonal grid, N is larger than V_2 but smaller than V_3 .
- N^+ : Let z be a vertex and f an incident face, and consider the configuration C on $F_1(z)$ that contains all vertices incident to the faces in $F_1(z)$ and all vertices within distance two of f . Use the information from $X \cap C$ to specify how much charge is exchanged from f to z .

It is possible to define an infinite number of discharging rules. Note that some rules are inherently *more complex* than others, so a partial ordering can be defined on the rules. For instance, $V_1 \subset V_2 \subset N \subset V_3$, so N is at least as effective as V_2 . We call attention to a few features of the discharging rules that should be balanced carefully in order to create the most effective rules.

Scope of Information. The larger the configuration used for the discharging rule, the more information is known about the local environment of the chargeable object receiving charge. However, as the configuration C grows, the number of realizations of the rule grows approximately as $2^{|V(C)|}$, with some loss for symmetry and for avoiding forbidden configurations.

Range of Exchange. Depending on the distance between z and the y_i 's, charge can be exchanged across

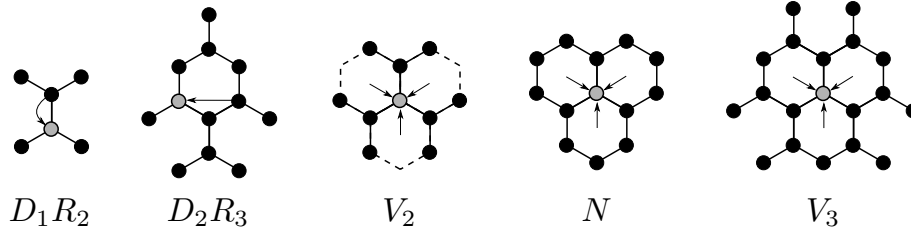


Figure 3: Examples of Rules in the Hexagonal Grid.

several distances. It may be beneficial to allow for charge to move longer distances, especially if it is possible to have large regions in G where X is much less dense than in other areas.

Dependence. For nearby chargeable objects, the configurations for different discharging rules overlap. Thus, some information is shared between the chargeable objects and that information can be used to assign value to the rule.

With these ideas in mind, we will describe a few extra features that can be used to customize our discharging arguments. These features exist to mimic patterns in human-created discharging arguments. When appropriate, we will “lose” information in order to reduce the number of cases we check.

See Figure 7 for the rule S (for “simple”). The rule S considers a vertex v_0 and its three neighbors v_1, v_2, v_3 and exchanges charge from v_1 to v_0 . The information used to describe the amount is listed by a set of *keys*, where each key is a linear combination of variables x_0, \dots, x_3 where x_i has value 1 if and only if v_i is an element. The “value” of a discharging rule is described by the tuple of key values preceded by a label for the rule; the description used for the simple rule is $s(x_0, x_1, x_2 + x_3)$. Thus, if $x_2 + x_3 = 1$, we “forget” which vertex is an element and which vertex is not. We also use two *kernel inequalities*. These inequalities are checked in order, and if they are satisfied we ignore any key values that do not use variables from that inequality and use “*” to signify an unknown quantity. For example, when $x_0 \geq 1$, the only value to consider is $s(1, *, *)$; this essentially implies that we do not want elements to receive charge from neighbors. Also when $x_0 + x_1 \leq 0$, the only value to consider is $s(0, 0, *)$; this essentially implies that we do not want to exchange charge between nonelements. There are five distinct ways to complete this rule, and these are listed in Figure 7 along with values that will imply a lower bound of $\frac{2}{5}$ on the density of an identifying code in the hexagonal grid.

By carefully designing keys and kernels for rules, we can greatly reduce the number of cases to check. Specifically, kernels are particularly helpful to restrict a rule when it sees a configuration that is more dense than expected in a sparse set.

This step of creating a discharging argument is the step that requires the most amount of creativity and human intervention. Creating interesting and effective rules is really where the proof author has most control, and this step is absolutely crucial in determining whether a discharging proof will provide a strong lower bound. The strength of the rules must be balanced with the computational challenge of verifying their correctness, which is the topic of the next section.

4 The Linear Program

The most difficult part of assigning value to discharging rules is verifying that objects of low charge receive enough charge to meet the goal charge while guaranteeing that objects of high charge do not lose so much charge they drop below the goal charge. In the contrapositive, it must be impossible to construct a configuration around a chargeable object where every discharging rule is evaluated and the resulting charge violates the postconditions. Thus, we shall create a configuration C around each chargeable object (up to isomor-

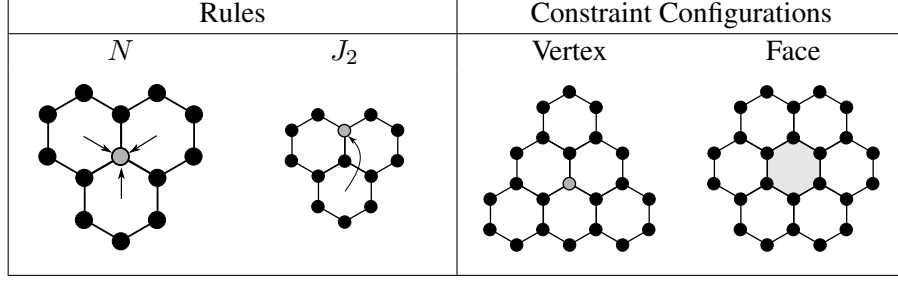


Figure 4: The rules N and J_2 and the resulting constraint configurations.

phism) such that C contains the shape of each rule that can exchange charge to or from that object, and then generate each \mathcal{F} -realization of C . Every such realization determines which realizations of the discharging rules to use, and these values are combined to form a constraint in a linear program.

Recall that our postconditions for vertices and faces are implied by the following inequalities:

$$\begin{aligned}\mu'(v) &= \mu(v) + \sum_{u \in V(G)} D_X(u, v) + \sum_{g \in F(G)} D_X(g, v) \geq w \\ \nu'(f) &= \nu(f) + \sum_{u \in V(G)} D_X(u, f) + \sum_{g \in F(G)} D_X(g, f) \geq 0\end{aligned}$$

Since we are using a finite list of finite-sized discharging rules, this inequality will in fact use a finite number of nonzero terms. Also, the amount of exchanged charge depends on a finite-sized local region about each chargeable object. Given a grid G and a list R_1, \dots, R_m of discharging rules, the *constraint configuration* about a chargeable object x is defined as the set of faces and vertices that appear in an embedding $\pi(C)$ of a configuration $C = C(R_j)$ such that $\pi(z(R_j)) = x$ or $\pi(y_i(R_j)) = x$ for some i . Observe that the constraint configurations about two chargeable objects, x and x' , are isomorphic if and only if x and x' are in orbit within G .

For example, consider the rules N and J_2 in the hexagonal grid as shown in Figure 4. Since the hexagonal grid is vertex- and face-transitive, we only need to consider the constraint configurations for v_0 and f_0 . About v_0 , there are three embeddings of N and three embeddings of J_2 such that the vertices $z(N)$ and $z(J_2)$ are mapped to v_0 . Together, these embeddings form a constraint configuration about v_0 consisting of all faces in $F_2(v_0)$, and all vertices incident to a face in $F_2(v_0)$. About f_0 , there are 18 embeddings of N and six embeddings of J_2 such that one of the faces $y_1(N), y_2(N), y_3(N)$ or the face $y_1(J_2)$ are mapped to f_0 . Together, these embeddings form a constraint configuration about f_0 consisting of all faces in $F_1(f_0)$ and all vertices incident to a face in $F_1(f_0)$. These constraint configurations are shown in Figure 4.

Given a constraint configuration, the way the discharging rules assign value to the charge exchange is determined exactly by the way that X intersects the vertices of this configuration. Therefore, we generate all \mathcal{F} -realizations of the constraint configuration. Given an \mathcal{F} -realization C' of our constraint configuration C , we add a constraint to our linear program.

Suppose we are using the rules R_1, \dots, R_k . If C is a constraint configuration centered on a vertex v and C' is an \mathcal{F} -realization of C , then we enforce that $\mu'(v) \geq w$ after the discharging process is complete by adding the constraint

$$\mu(v) + \sum_{j=1}^k \sum_{\pi: \pi(z(R_j))=v} \sum_i \sigma_j(S_1(C'), y_i) - \sum_{j=1}^k \sum_{i, \pi: \pi(y_i(R_j))=v} \sigma_j(S_1(C'), y_i) \geq w$$

to the linear program. If C is a constraint configuration centered on a face f and C' is an \mathcal{F} -realization of C , then we enforce that $\nu'(f) \geq 0$ after the discharging process is complete by adding the constraint

$$\nu(f) + \sum_{j=1}^k \sum_{\pi: \pi(z(R_j))=f} \sum_i \sigma_j(S_1(C'), y_i) - \sum_{j=1}^k \sum_{i, \pi: \pi(y_i(R_j))=f} \sigma_j(S_1(C'), y_i) \geq 0$$

to the linear program.

Observe that whenever these constraints are satisfied, the discharging argument demonstrates a lower bound of $\delta(X) \geq w$ for any $X \in \text{forb}(\mathcal{F})$. In order to produce the largest lower bound, use $\max w$ as the optimization function of the linear program.

Thus, we have a complete description of an adage proof. In summary, the three main steps are: (1) Define a set of rules R_1, \dots, R_k , and generate all \mathcal{F} -realizations C' of their configurations, mapping the values $\sigma_j(S_1(C'), y_i)$ to a list of variables; (2) For each constraint configuration (up to isomorphism), generate all \mathcal{F} -realizations and add the resulting constraint to the linear program; (3) Solve the linear program to determine the values $\sigma_j(S_1(C'), y_i)$ and the lower bound w .

These steps were implemented and executed for several sets of rules, which are shown in Appendix A. All software and data are available online¹. The following theorem implies Theorem 1.

Theorem 4. *Let X be an identifying code in the hexagonal grid. The adage proof using rule N demonstrates $\delta(X) \geq \frac{23}{55} \approx 0.41818$.*

This bound of $\frac{23}{55}$ seems to be a “natural” barrier, and not just an artifact of the rule N . The rule N exchanges charge between vertices and incident faces, using the faces to aggregate and balance the charge. We have a set of rules that exchange charge only among vertices and find the same bound of $\frac{23}{55}$.

The ADAGE framework as described is not tied to any specific grid or family of forbidden configurations \mathcal{F} . In the next section, we discuss variations on identifying codes and summarize the adage proofs of sharp lower bounds for those variations.

5 Variations

Due to the modular development of the ADAGE framework for grids, the components for the grid and the forbidden configurations can be interchanged. This allows for adage proofs to be constructed for the hexagonal, square, triangular, and pentagonal grids. More planar grids could be implemented and used, including those that are not vertex- or face-transitive, such as the hexagon-triangle grid.

There are several variations of an identifying code, each with its own application to fault-detection in computer networks. A set $X \subset V(G)$ matches these variations if the following constraints are satisfied:

- Dominating Set: $N[v] \cap X \neq \emptyset$ for all $v \in V(G)$.
- Identifying Code: $N[v] \cap X \neq \emptyset$ and $(N[v] \cap X) \neq (N[u] \cap X)$ for all distinct $u, v \in V(G)$.
- Strong Identifying Code: $N[v] \cap X \neq \emptyset$ and $\{N[v] \cap X, N(v) \cap X\} \cap \{N[u] \cap X, N(u) \cap X\} = \emptyset$ for all distinct $u, v \in V(G)$ (see [14, 18]).
- Locating-Dominating Code: $N(v) \cap X \neq \emptyset$ for $v \notin X$, and $N(v) \cap X \neq N(u) \cap X$ for all distinct $u, v \in V(G) \setminus X$ (see [4, 13, 16, 32]).
- Open-Locating-Dominating (OLD) Code: $N(v) \cap X \neq \emptyset$ and $N(u) \cap X \neq N(v) \cap X$ for all distinct $u, v \in V(G)$ (see [22, 31]).
- Neighbor-Identifying Code: $N[v] \cap X \neq \emptyset$ and $N[u] \cap X \neq N[v] \cap X$ for all $uv \in E(G)$.

¹See <http://www.math.iastate.edu/dstolee/r/adage.htm> or <http://github.com/derrickstolee/ADAGE/> for all software and data.

All of these variations are implemented in the current version of ADAGE on grids. Several collections of discharging rules were used to find adage proofs of lower bounds on these variations, and the results can be found in Table 1. We believe this is the first use of neighbor-identifying codes; our motivation is to demonstrate that we can find sharp lower bounds without known results to guide our search. Below, we summarize adage proofs of previous sharp results with attribution to the first authors to find such bounds. See Appendix B for lower bounds demonstrated by other rule sets.

Theorem 5 (Ben-Haim and Litsyn [3]). *Let X be an identifying code in the square grid. The adage proof using the rule V_2 demonstrates $\delta(X) \geq \frac{7}{20}$.*

Theorem 6 (Karpovsky, Chakrabarty, and Levitin [21]). *Let X be an identifying code in the triangular grid. The adage proof using the rule V_1 demonstrates $\delta(X) \geq \frac{1}{4}$.*

Theorem 7 (Honkala [13]). *Let X be a locating-dominating code in the hexagonal grid. The adage proof using the rule V_2 demonstrates $\delta(X) \geq \frac{1}{3}$.*

Theorem 8 (Slater [33]). *Let X be a locating-dominating code in the square grid. The adage proof using the rule V_2 demonstrates $\delta(X) \geq \frac{3}{10}$.*

Theorem 9 (Seo and Slater [31]). *Let X be an open-locating dominating code in the hexagonal grid. The adage proof using the rule V_2 demonstrates $\delta(X) \geq \frac{1}{2}$.*

Theorem 10 (Seo and Slater [31]). *Let X be an open-locating dominating code in the square grid. The adage proof using the rules D_1R_2 and D_2R_2 demonstrates $\delta(X) \geq \frac{2}{5}$.*

Theorem 11 (Kincaid, Oldham, and Yu [22]). *Let X be an open-locating dominating code in the triangular grid. The adage proof using the rule D_1R_2 demonstrates $\delta(X) \geq \frac{4}{13}$.*

Observe that among all variations on all three grids, the only variations that failed to find a sharp lower bound were identifying codes on the hexagonal grid, and strong identifying codes on all three grids. Likely, the strong identifying codes are more challenging because a strong identifying code is both an identifying code and an open-locating dominating code, so the optimal density is highest among all of these variations. Also, there are more forbidden configurations and this leads to fewer realizations of the discharging rules (and hence fewer variables in the linear program).

There are also variations on identifying codes that are robust against edge changes [12, 15, 17, 24, 33], or identify all sets of vertices of size at most ℓ [10, 23, 25], or consider balls of larger radius [19–21, 26, 29, 34, 35]. These variations are good candidates for future implementation.

6 Upper Bounds

Previous upper bounds on the minimum densities of codes in grids have used ad-hoc methods. For instance, Cukierman and Yu [8] selected a tiling of the hexagonal grid and then used an integer program to minimize the density of a periodic set using that tiling. While these methods have been effective, they rely significantly on human selection of a good tiling scheme. We use our discharging arguments to find upper bounds.

Suppose a discharging argument is optimal with lower bound w and there exists an \mathcal{F} -free set X with $\delta(X) = w$. Consider applying the discharging argument to X , fix some $\varepsilon > 0$, let V_+ be the set of vertices v with final charge $\mu'(v) \geq w + \varepsilon$, and let F_+ be the set of faces f with final charge $\nu'(f) \geq \varepsilon$. Observe that since $\delta(X) = w$, the density of V_+ in G (and F_+ in the dual of G) is zero. Thus, for every radius $r > 0$ there exists at least one vertex $v \in V(G)$ where $B_r(v) \cap V_+ = \emptyset$ and $F_r(v) \cap F_+ = \emptyset$.

This observation allows for a new method for searching for upper bounds. Start by selecting a reasonably-sized region of G , such as $B_r(0) \cup F_r(0)$ for some $r > 0$. Then, search over all \mathcal{F} -realizations of this region

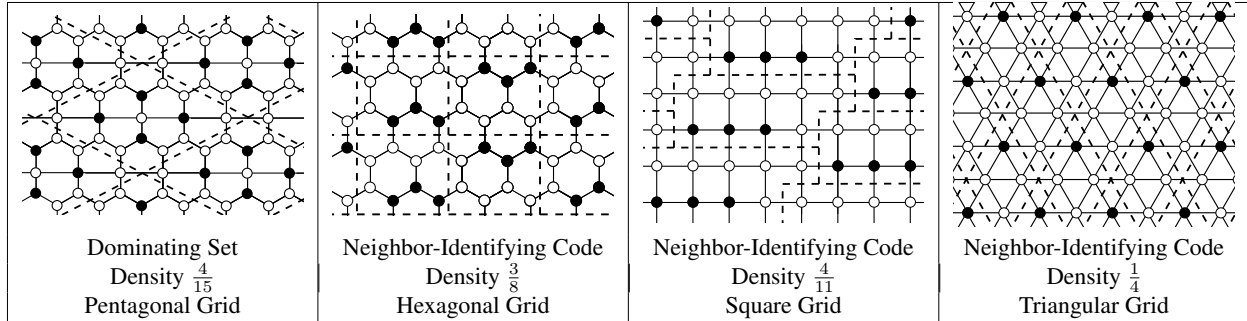


Figure 5: Some periodic tilings of optimum density.

using backtrack search for a realization that satisfies $\mu'(v) < w + \varepsilon$ and $\nu'(f) < \varepsilon$ for all vertices v and faces f ; the search can backtrack whenever all discharging rules are complete at a vertex or face and the final charge is too large. If no such realization is found, then the lower bound is *sharp* and $\delta(X) > w$ for all \mathcal{F} -free sets X . If such a realization is found, then we can apply automorphisms of G to attempt to cover the grid. Two automorphisms σ_1 and σ_2 generate a subgroup Γ . We will *accept* the automorphisms σ_1, σ_2 and the realization Y if all Γ -orbits contain at least one element of $B_r(0) \cup F_r(0)$ and there is no Γ -orbit that contains an element of Y and an element of $B_r(0) \setminus Y$. Using this algorithm, we determine several upper bounds that match the lower bounds given by the adage proofs. The optimal sets are shown in Figure 5.

7 Conclusions and Future Work

This first application of the ADAGE framework is successful in showing alternative proofs of existing sharp bounds [3, 16, 21, 22, 31, 33], and surpassing the human-written proofs of lower bounds on identifying codes in the hexagonal grid [5, 7, 8, 21]. The computer-automated portions of the method replace lengthy case analysis and can be more detailed than something within the reach of a human prover. However, the simple description of the discharging rules can perhaps lead to a deeper understanding of the structure and success of a discharging argument. By automating the process of assigning value to the discharging rules, a proof author can focus more on the creative process of designing rules. Thus, the most important part is to balance the strength of the rules against the size of the constraint configurations.

We have implemented several grid operations that allow us to test the discharging methods on the grids formed by almost all of the uniform convex tilings². Basic rules such as V_i or $D_i R_j$ can be generated automatically, but more advanced rules that include keys and kernels must be designed by hand and tested.

All of the lower bounds presented in this paper are approximate, up to rounding error in floating point arithmetic and the numeric instability of the linear programming solver. To be completely accurate, we will implement a process to round all values to fractions and test all constraints using exact arithmetic. For smaller instances, we can solve using simplex and round to fractions easily with no error. Multiple issues arise here, though, as we use the barrier method instead of simplex due to significant time savings (our programs are very degenerate) but barrier solutions are typically in the middle of a face of the polytope and this leads to worse fractional representations. This tradeoff is non-trivial.

Acknowledgements

Thanks to Michael Ferrara, Stephen G. Hartke, Bernard Lidický, Ryan R. Martin, and Paul S. Wenger for several very helpful discussions about the discharging method and identifying codes.

²See https://en.wikipedia.org/wiki/List_of_convex_uniform_tilings.

References

- [1] K. Appel, W. Haken, Every planar map is four colorable. Part I: Discharging. *Illinois Journal of Mathematics*, 21(3):429–490, 1977.
- [2] K. Appel, W. Haken, J. Koch, Every planar map is four colorable. Part II: Reducibility. *Illinois Journal of Mathematics*, 21(3):491–567, 1977.
- [3] Y. Ben-Haim and S. Litsyn. Exact minimum density of codes identifying vertices in the square grid. *SIAM Journal on Discrete Mathematics*, 19(1):69–82, 2005.
- [4] J. Cáceres, C. Hernando, M. Mora, I. M. Pelayo, and M. L. Puertas. Locating–dominating codes: Bounds and extremal cardinalities. *Applied Mathematics and Computation*, 220:38–45, 2013.
- [5] G. D. Cohen, I. Honkala, A. Lobstein, and G. Zémor. Bounds for codes identifying vertices in the hexagonal grid. *SIAM Journal on Discrete Mathematics*, 13(4):492–504, 2000.
- [6] D. W. Cranston and D. B. West. A guide to the discharging method. *arXiv preprint arXiv:1306.4434*, 2013.
- [7] D. W. Cranston and G. Yu. A new lower bound on the density of vertex identifying codes for the infinite hexagonal grid. *The Electronic Journal of Combinatorics*, 16(1):R113, 2009.
- [8] A. Cukierman and G. Yu. New bounds on the minimum density of an identifying code for the infinite hexagonal grid. *Discrete Applied Mathematics*, 161(18):2910–2924, 2013.
- [9] X. Dong, M. C. Vuran, and S. Irmak. Autonomous precision agriculture through integration of wireless underground sensor networks with center pivot irrigation systems. *Ad Hoc Networks*, 11(7):1975–1987, 2013.
- [10] F. Foucaud, T. Laihonen, and A. Parreau. An improved lower bound for $(1, \leq 2)$ -identifying codes in the king grid. *arXiv preprint arXiv:1111.2477*, 2011.
- [11] S. G. Hartke and D. Stolee. Uniquely K_r -saturated graphs. *The Electronic Journal of Combinatorics*, 19(4):P6, 2012.
- [12] I. Honkala. An optimal edge-robust identifying code in the triangular lattice. *Annals of Combinatorics*, 8(3):303–323, 2004.
- [13] I. Honkala. An optimal locating-dominating set in the infinite triangular grid. *Discrete Mathematics*, 306(21):2670–2681, 2006.
- [14] I. Honkala. An optimal strongly identifying code in the infinite triangular grid. *The Electronic Journal of Combinatorics*, 17(1):R91, 2010.
- [15] I. Honkala, M. G. Karpovsky, and L. B. Levitin. On robust and dynamic identifying codes. *IEEE Transactions on Information Theory*, 52(2):599–612, 2006.
- [16] I. Honkala and T. Laihonen. On locating–dominating sets in infinite grids. *European Journal of Combinatorics*, 27(2):218–227, 2006.
- [17] I. Honkala and T. Laihonen. On identifying codes that are robust against edge changes. *Information and Computation*, 205(7):1078–1095, 2007.
- [18] I. Honkala, T. Laihonen, and S. Ranto. On strongly identifying codes. *Discrete Mathematics*, 254(1):191–205, 2002.
- [19] V. Junnila. New lower bound for 2-identifying code in the square grid. *Discrete Applied Mathematics*, 161(13):2042–2051, 2013.
- [20] V. Junnila and T. Laihonen. Optimal lower bound for 2-identifying codes in the hexagonal grid. *The Electronic Journal of Combinatorics*, 19(2):P38, 2012.
- [21] M. G. Karpovsky, K. Chakrabarty, and L. B. Levitin. On a new class of codes for identifying vertices in graphs. *IEEE Transactions on Information Theory*, 44(2):599–611, 1998.

- [22] R. Kincaid, A. Oldham, and G. Yu. Optimal open-locating-dominating sets in infinite triangular grids. *Discrete Applied Mathematics*, 2015.
- [23] T. Laihonen. On optimal edge-robust and vertex-robust $(1, \leq \ell)$ -identifying codes. *SIAM Journal on Discrete Mathematics*, 18(4):825–834, 2005.
- [24] T. Laihonen. On robust identification in the square and king grids. *Discrete Applied Mathematics*, 154(17):2499–2510, 2006.
- [25] T. Laihonen. Optimal t -edge-robust r -identifying codes in the king lattice. *Graphs and Combinatorics*, 22(4):487–496, 2006.
- [26] R. Martin and B. Stanton. Lower bounds for identifying codes in some infinite grids. *The Electronic Journal of Combinatorics*, 17(R122):1, 2010.
- [27] A. A. Razborov. Flag algebras. *The Journal of Symbolic Logic*, 72(04):1239–1282, 2007.
- [28] A. A. Razborov. Flag algebras: an interim report. In *The Mathematics of Paul Erdős II*, pages 207–232. Springer, 2013.
- [29] D. L. Roberts and F. S. Roberts. Locating sensors in paths and cycles: The case of 2-identifying codes. *European Journal of Combinatorics*, 29(1):72–82, 2008.
- [30] N. Robertson, D. Sanders, P. Seymour, and R. Thomas. The four-colour theorem. *Journal of Combinatorial Theory, Series B*, 70(1):2–44, 1997.
- [31] S. J. Seo and P. J. Slater. Open neighborhood locating-dominating sets. *Australasian J. Combin*, 46:109–120, 2010.
- [32] P. J. Slater. Locating dominating sets and locating-dominating sets. In *Graph Theory, Combinatorics and Applications: Proceedings of the Seventh Quadrennial International Conference on the Theory and Applications of Graphs*, volume 2, pages 1073–1079, 1995.
- [33] P. J. Slater. Fault-tolerant locating-dominating sets. *Discrete Mathematics*, 249(1):179–189, 2002.
- [34] B. Stanton. Improved bounds for r -identifying codes of the hex grid. *SIAM Journal on Discrete Mathematics*, 25(1):159–169, 2011.
- [35] J. Ville and T. Laihonen. Optimality of a 2-identifying code in the hexagonal grid. In *WCC 2011-Workshop on coding and cryptography*, pages 47–56, 2011.
- [36] D. B. West. *Introduction to graph theory*, volume 2. Prentice Hall Upper Saddle River, 2001.

A Discharging Rules

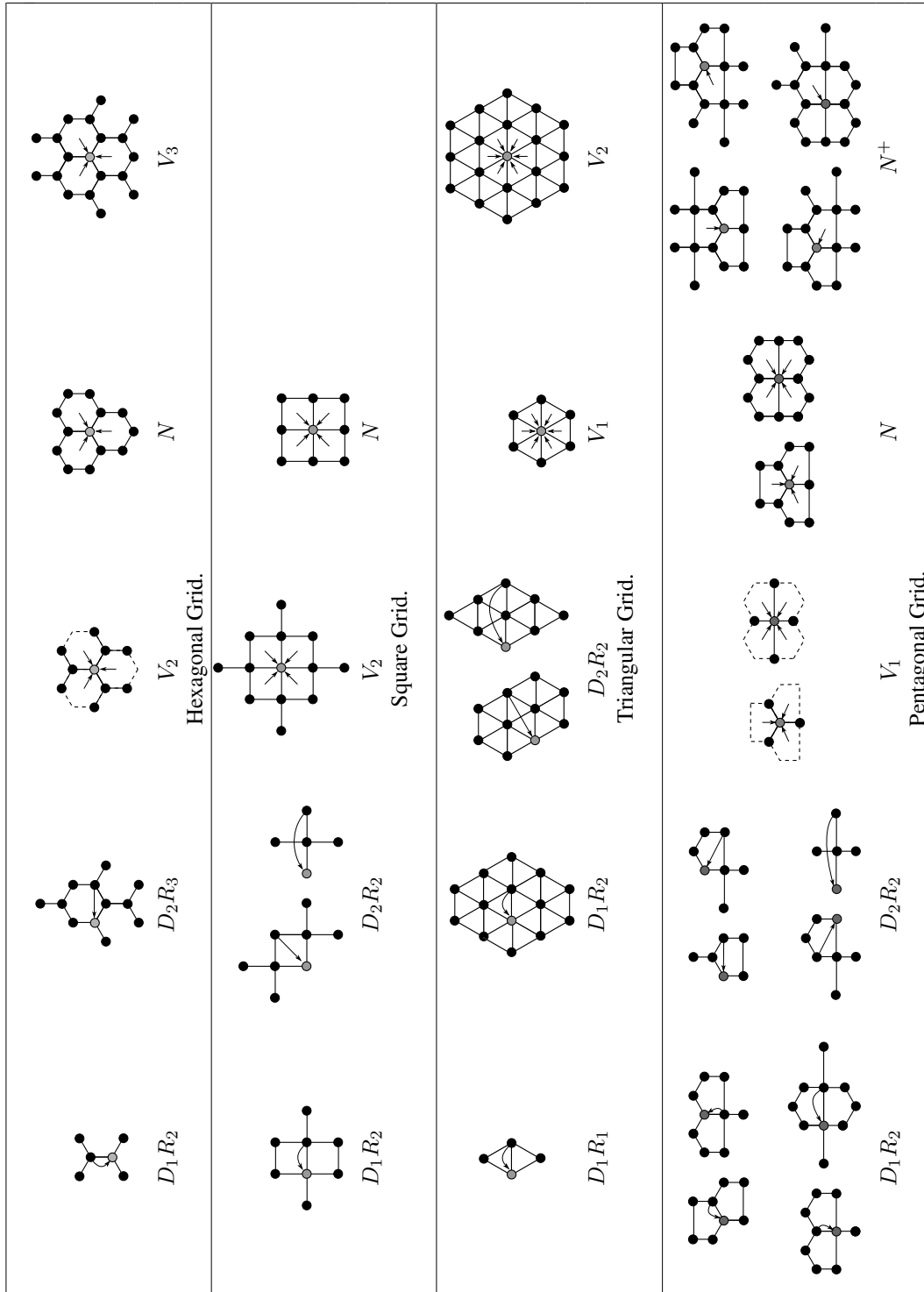


Figure 6: Examples of rules in the four grids.

B Bounds on Codes Using ADAGE

<i>Set Type</i>	Hexagonal Grid		Square Grid	
Dominating Set	V_1	$\frac{1}{4} \approx 0.250000^*$	V_1	$\frac{1}{5} \approx 0.200000^*$
Identifying Code	N	$\frac{23}{55} \approx 0.418182^\dagger$	D_1R_2, D_2R_2	$\frac{7}{20} \approx 0.350000^*$
	<i>Upper</i> [5]:	$\frac{3}{7} \approx 0.428571^\ddagger$	<i>Upper</i> [3]:	$\frac{7}{20} \approx 0.350000^*$
Strong Identifying Code	D_1R_2, D_2R_2	$\frac{8}{17} \approx 0.470588$	D_1R_2, D_2R_2	$\frac{7}{18} \approx 0.388889$
	<i>Upper</i> [14]:	$\frac{1}{2} \approx 0.500000^*$	<i>Upper</i> [14]:	$\frac{2}{5} \approx 0.400000^*$
Locating-Dominating Code	V_2	$\frac{1}{3} \approx 0.333333^*$	V_2	$\frac{3}{10} \approx 0.300000^*$
	<i>Upper</i> [16]:	$\frac{1}{3} \approx 0.333333^*$	<i>Upper</i> [33]:	$\frac{3}{10} \approx 0.300000^*$
OLD Code	D_1R_2, D_2R_2	$\frac{1}{2} \approx 0.500000^*$	D_1R_2, D_2R_2	$\frac{2}{5} \approx 0.400000^*$
	<i>Upper</i> [31]:	$\frac{1}{2} \approx 0.500000^*$	<i>Upper</i> [31]:	$\frac{2}{5} \approx 0.400000^*$
Neighbor-Identifying Code	V_1	$\frac{3}{8} \approx 0.375000^*$	V_1	$\frac{3}{11} \approx 0.272727^*$
	<i>Upper</i> :	$\frac{3}{8} \approx 0.375000^*$	<i>Upper</i> :	$\frac{3}{11} \approx 0.272727^*$
<i>Set Type</i>	Triangular Grid		Pentagonal Grid	
Dominating Set	V_1	$\frac{1}{7} \approx 0.142857^*$	V_1	$\frac{4}{15} \approx 0.266666^*$
Identifying Code	D_1R_2	$\frac{1}{4} \approx 0.250000^*$	N^+	$\frac{5}{13} \approx 0.384615$
	<i>Upper</i> [21]:	$\frac{1}{4} \approx 0.250000^*$		
Strong Identifying Code	D_1R_2	$\frac{4}{13} \approx 0.307692$	V_2	$\frac{5}{12} \approx 0.416666$
	<i>Upper</i> [14]:	$\frac{6}{19} \approx 0.315789^*$		
Locating-Dominating Code	V_2	$\frac{12}{53} \approx 0.226415$	V_2	$\frac{22}{69} \approx 0.318841$
	<i>Upper</i> [13]:	$\frac{13}{57} \approx 0.228070^*$		
OLD Code	D_1R_2	$\frac{4}{13} \approx 0.307692^*$	D_1R_2, D_2R_2	$\frac{4}{9} \approx 0.444444$
	<i>Upper</i> [22]:	$\frac{4}{13} \approx 0.307692^*$		
Neighbor-Identifying Code	V_1	$\frac{1}{4} \approx 0.250000^*$	D_1R_2, D_2R_2	$\frac{1}{3} \approx 0.333333$
	<i>Upper</i> :	$\frac{1}{4} \approx 0.250000^*$		

* Bound given is optimal lower bound on density.

† Bound given is current-best lower bound on density, but may not be optimal.

‡ Bound given is current-best upper bound on density, but may not be optimal.

Table 1: Density lower bounds for various set types in various grids, using various discharging rules.

C Proof of Theorem 3

Proof of Theorem 3. By Observation 2, we can select the zero vertex v_0 and zero face f_0 such that $|B_d(v_0)| = \max\{|B_d(v)| : v \in V(G)\}$ and $|F_d(f_0)| = \max\{|F_d(f)| : f \in F(G)\}$.

Recall that by the definitions of μ and ν ,

$$\delta(X) = \limsup_{r \rightarrow \infty} \frac{|X \cap B_r(v_0)|}{|B_r(v_0)|} = \limsup_{r \rightarrow \infty} \frac{\sum_{v \in B_r(v_0)} \mu(v) + \sum_{f \in F_r(v_0)} \nu(f)}{|B_r(v_0)|} \quad (1)$$

By hypothesis,

$$\limsup_{r \rightarrow \infty} \frac{\sum_{v \in B_r(v_0)} \mu'(v) + \sum_{f \in F_r(v_0)} \nu'(f)}{|B_r(v_0)|} \geq \limsup_{r \rightarrow \infty} \frac{\sum_{v \in B_r(v_0)} w + \sum_{f \in F_r(v_0)} 0}{|B_r(v_0)|} = w. \quad (2)$$

Our goal is to prove that the limit at the end of (1) and the limit at the beginning of (2) are equal, thereby showing that $\delta(X) \geq w$.

Using the definition of μ' and ν' and the fact that D_X is (c, d) -local, we find that the absolute difference $\left| \sum_{v \in B_r(v_0)} [\mu'(v) - \mu(v)] + \sum_{f \in F_r(v_0)} [\nu'(f) - \nu(f)] \right|$ is equal to the magnitude of the charge that D_X exchanges across the boundaries of $B_r(v_0)$ and $F_r(v_0)$:

$$\begin{aligned} & \left| \sum_{v \in B_r(v_0)} \left[\sum_{u \notin B_r(v_0)} D_X(u, v) + \sum_{g \notin F_r(v_0)} D_X(g, v) \right] + \sum_{f \in F_r(v_0)} \left[\sum_{u \notin B_r(v_0)} D_X(u, v) + \sum_{g \notin F_r(v_0)} D_X(g, f) \right] \right| \\ & \leq \sum_{u \in B_{r+d}(v_0) \setminus B_r(v_0)} \left[\sum_{v \in B_d(u)} |D_X(u, v)| + \sum_{f \in F_d(u)} |D_X(u, f)| \right] \\ & \quad + \sum_{g \in F_{r+d}(v_0) \setminus F_r(v_0)} \left[\sum_{v \in B_d(g)} |D_X(g, v)| + \sum_{f \in F_d(g)} |D_X(g, f)| \right] \\ & \leq |B_{r+d}(v_0) \setminus B_r(v_0)| \cdot [|B_d(v_0)| + |F_d(v_0)|] \cdot c + |F_{r+d}(v_0) \setminus F_r(v_0)| \cdot [|B_d(f_0)| + |F_d(f_0)|] \cdot c \\ & \leq c' \cdot [|B_{r+d}(v_0) \setminus B_r(v_0)| + |F_{r+d}(v_0) \setminus F_r(v_0)|], \end{aligned}$$

where $c' = c[|B_d(v_0)| + |F_d(f_0)|]$. Since G is amenable and has finite maximum degree $\Delta(G)$,

$$\limsup_{r \rightarrow \infty} \frac{c' |B_{r+d}(v_0) \setminus B_r(v_0)| + |F_{r+d}(v_0) \setminus F_r(v_0)|}{|B_r(v_0)|} \leq \limsup_{r \rightarrow \infty} \frac{c'(\Delta(G) + 1) |B_{r+d}(v_0) \setminus B_r(v_0)|}{|B_r(v_0)|} = 0,$$

and therefore

$$\limsup_{r \rightarrow \infty} \frac{\left| \sum_{v \in B_r(v_0)} [\mu'(v) - \mu(v)] + \sum_{f \in F_r(v_0)} [\nu'(f) - \nu(f)] \right|}{|B_r(v_0)|} = 0,$$

proving the claim. \square

D A Full Linear Program Example

Recall the definition of the *simple* rule S on the hexagonal grid (represented here as Figure 7).

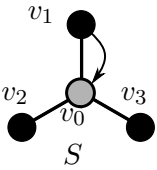
 <p style="text-align: center;">Rule</p>	$\frac{s(x_0, x_1, x_2 + x_3)}{s(1, *, *) = 0}$ $s(0, 0, *) = 0$ $s(0, 1, 0) = \frac{2}{5}$ $s(0, 1, 1) = \frac{1}{5}$ $s(0, 1, 2) = \frac{2}{15}$ <p style="text-align: center;">Keys and Realizations</p>	$x_0 \geq 1$ $x_0 + x_1 \leq 1$ <p style="text-align: center;">Kernels</p>
---	---	--

Figure 7: The rule S with its keys and kernels.

The linear constraints for the adage proof that $\delta(X) \geq \frac{2}{5}$ for an identifying code X are given below.

$$\begin{array}{rcll}
 1 & - & 1 s(0, 1, 2) & + & 1 s(1, *, *) & \geq & w \\
 1 & - & 1 s(0, 1, 1) & + & 1 s(1, *, *) & \geq & w \\
 1 & - & 1 s(0, 1, 0) & + & 1 s(1, *, *) & \geq & w \\
 1 & - & 2 s(0, 1, 2) & + & 2 s(1, *, *) & \geq & w \\
 1 & - & 1 s(0, 1, 1) & - & 1 s(0, 1, 2) & + & 2 s(1, *, *) \geq w \\
 1 & - & 1 s(0, 1, 0) & - & 1 s(0, 1, 2) & + & 2 s(1, *, *) \geq w \\
 1 & - & 2 s(0, 1, 1) & + & 2 s(1, *, *) & \geq & w \\
 1 & - & 1 s(0, 1, 0) & - & 1 s(0, 1, 1) & + & 2 s(1, *, *) \geq w \\
 1 & - & 3 s(0, 1, 2) & + & 3 s(1, *, *) & \geq & w \\
 1 & - & 1 s(0, 1, 1) & - & 2 s(0, 1, 2) & + & 3 s(1, *, *) \geq w \\
 1 & - & 2 s(0, 1, 1) & - & 1 s(0, 1, 2) & + & 3 s(1, *, *) \geq w \\
 1 & - & 3 s(0, 1, 1) & + & 3 s(1, *, *) & \geq & w \\
 0 & + & 3 s(0, 1, 2) & - & 3 s(1, *, *) & \geq & w \\
 0 & + & 2 s(0, 1, 1) & - & 2 s(1, *, *) & \geq & w \\
 0 & + & 1 s(0, 1, 0) & - & 1 s(1, *, *) & \geq & w
 \end{array}$$

Observe the realization $s(0, 0, *)$ does not appear in the linear program, since it always has a coefficient of zero.