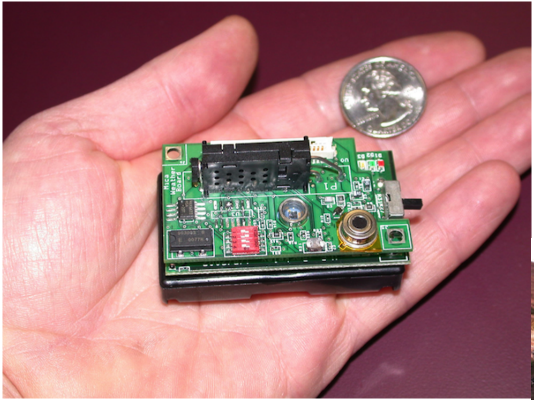
Automated Discharging Arguments for Density Problems in Grids

Derrick Stolee

Iowa State University
dstolee@iastate.edu
http://www.math.iastate.edu/dstolee/

September 2, 2014
Mathematics Department Colloquium

Wireless Sensor Networks



Fault Tolerance

These devices **break!**

Fault Tolerance

These devices **break!**

We want to know which device needs to be repaired after a failure.

Fault Tolerance

These devices **break!**

We want to know which device needs to be repaired after a failure.

We can put detection process on some of the nodes, but that drains power, so we want to put that on the **smallest** number of nodes.

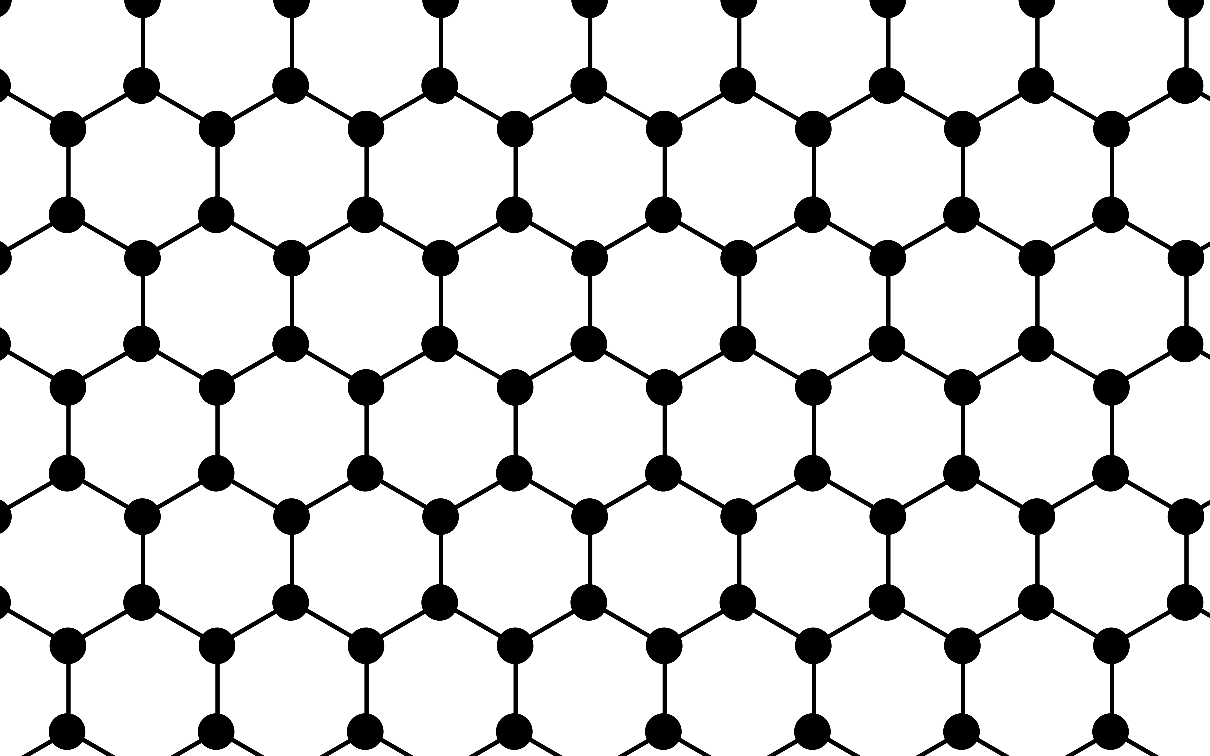
Fault Tolerance

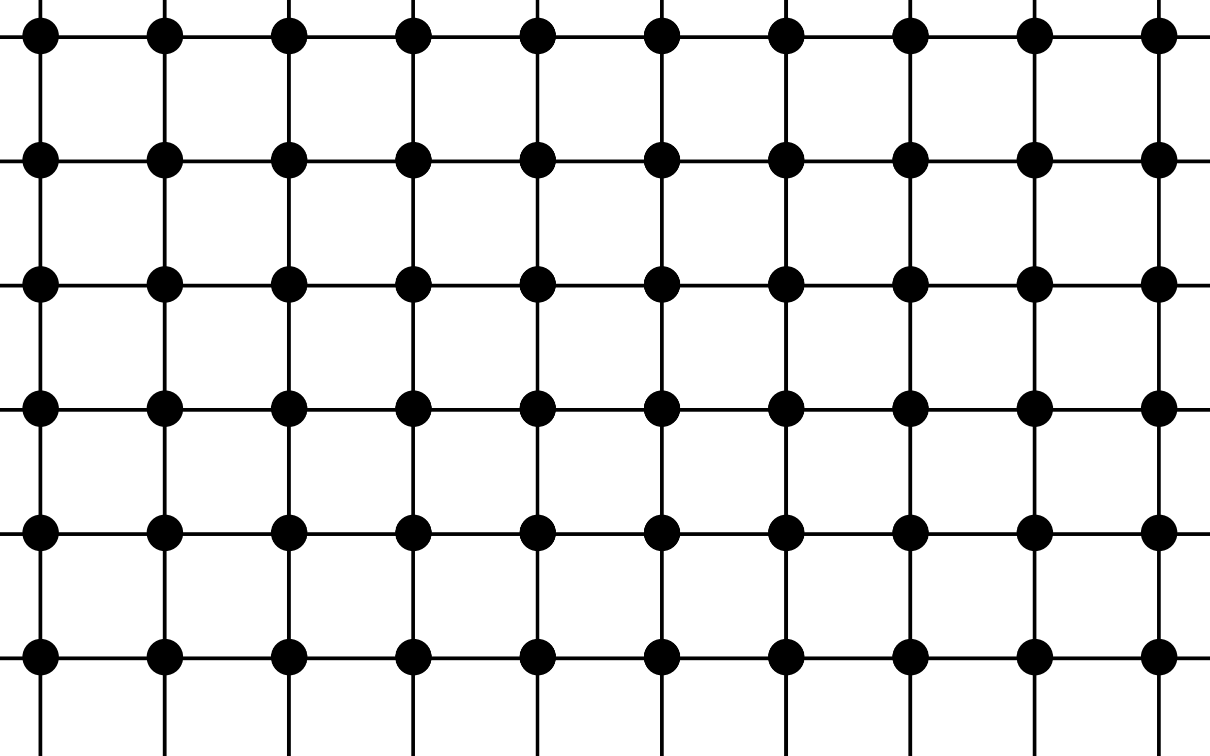
These devices **break!**

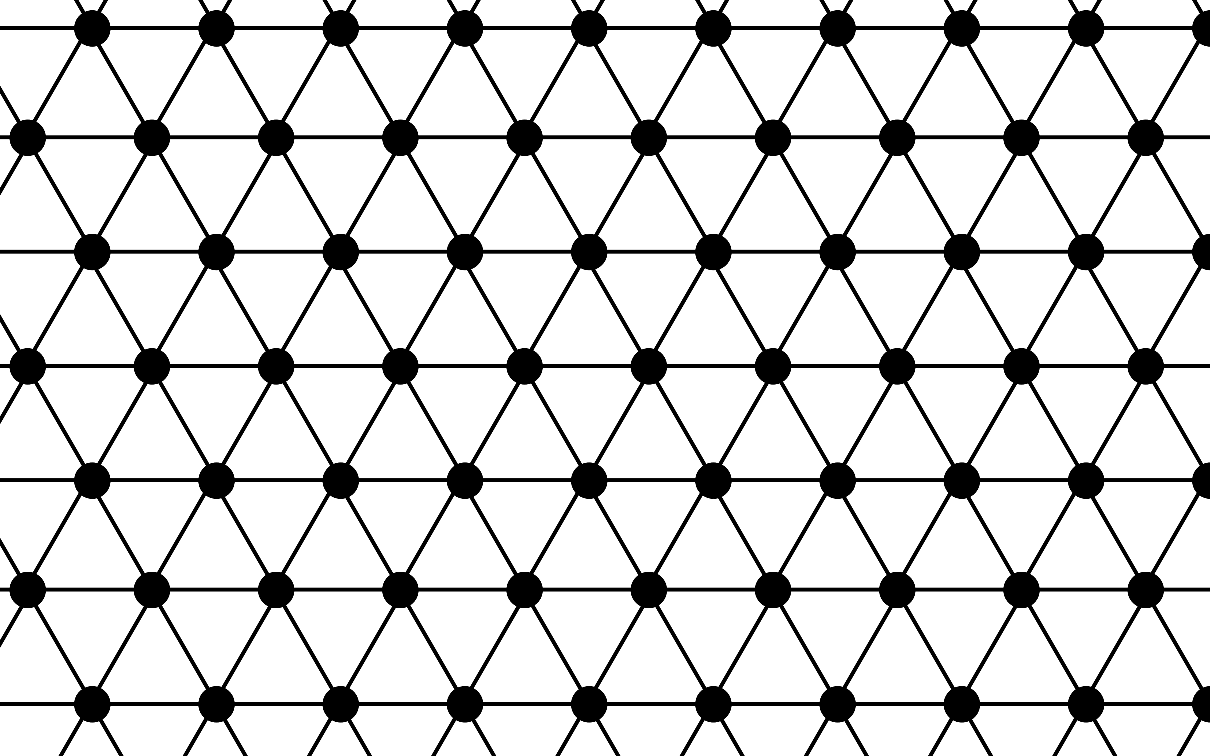
We want to know which device needs to be repaired after a failure.

We can put detection process on some of the nodes, but that drains power, so we want to put that on the **smallest** number of nodes.

Combinatorial Optimization!







Density

These grids are **amenable**:

$$\limsup_{r\to\infty} \frac{|B_{r+d}(v) \setminus B_r(v)|}{|B_r(v)|} = 0,$$

where $B_r(v)$ is the ball of radius r about a vertex v.

Density

These grids are **amenable**:

$$\limsup_{r\to\infty} \frac{|B_{r+d}(v) \setminus B_r(v)|}{|B_r(v)|} = 0,$$

where $B_r(v)$ is the ball of radius r about a vertex v.

This implies two facts:

$$\liminf_{r\to\infty} \frac{|B_r(v) \cap B_r(u)|}{|B_r(v)|} = 1,$$

and

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

for any pair of vertices $u, v \in V(G)$ and any set $X \subseteq V(G)$.

Density

Therefore, we can select an arbitrary vertex $v_0 \in V(G)$ and define the **density** of a set $X \subseteq V(G)$ as

$$\delta(X) = \limsup_{r \to \infty} \frac{|B_r(v_0) \cap X|}{|B_r(v_0)|}.$$

Density

Therefore, we can select an arbitrary vertex $v_0 \in V(G)$ and define the **density** of a set $X \subseteq V(G)$ as

$$\delta(X) = \limsup_{r \to \infty} \frac{|B_r(v_0) \cap X|}{|B_r(v_0)|}.$$

This definition is used for problems where we **minimize** the density.

We would use $\liminf$ for maximizing the density.

Dominating Sets

A set $X \subseteq V(G)$ is a **dominating set** if

- $N[v] \cap X \neq \varnothing$ for all vertices $v \in V(G)$.

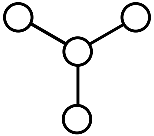
($N[v]$ is the **closed neighborhood** of v: $N[v] = N(v) \cup \{v\}$.)

Dominating Sets

A set $X \subseteq V(G)$ is a **dominating set** if

- $N[v] \cap X \neq \varnothing$ for all vertices $v \in V(G)$.

($N[v]$ is the **closed neighborhood** of v: $N[v] = N(v) \cup \{v\}$.)



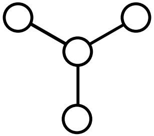
Forbidden Configuration

Dominating Sets

A set $X \subseteq V(G)$ is a **dominating set** if

- $N[v] \cap X \neq \varnothing$ for all vertices $v \in V(G)$.

($N[v]$ is the **closed neighborhood** of v: $N[v] = N(v) \cup \{v\}$.)



Forbidden Configuration

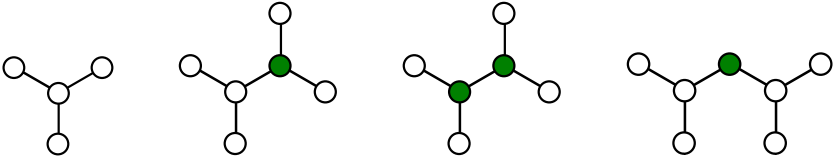
It is not difficult to see that the optimal density of a dominating set in the hexagonal grid is $\frac{1}{4} = 0.250000$.

Identifying Codes

A set $X \subseteq V(G)$ is an **identifying code** if

- $N[v] \cap X \neq \varnothing$ for all vertices $v \in V(G)$, and
- $N[v] \cap X \neq N[u] \cap X$ for all distinct vertices $v, u \in V(G)$.

($N[v]$ is the **closed neighborhood** of v: $N[v] = N(v) \cup \{v\}$.)



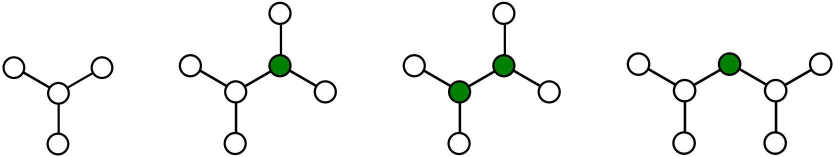
Forbidden Configurations

Identifying Codes

A set $X \subseteq V(G)$ is an **identifying code** if

- $N[v] \cap X \neq \varnothing$ for all vertices $v \in V(G)$, and
- $N[v] \cap X \neq N[u] \cap X$ for all distinct vertices $v, u \in V(G)$.

($N[v]$ is the **closed neighborhood** of v: $N[v] = N(v) \cup \{v\}$.)



Forbidden Configurations

Defined by Karpovsky, Chakrabarty, Levitin in 1998.

Identifying Codes

Alternative Definition

A set $X \subseteq V(G)$ is an **identifying code** if

$$(N[v] \triangle N[u]) \cap X \neq \varnothing$$

for all distinct vertices $v, u \in V(G)$.

Identifying Codes

Alternative Definition

A set $X \subseteq V(G)$ is an **identifying code** if

$$(N[v] \triangle N[u]) \cap X \neq \varnothing$$

for all distinct vertices $v, u \in V(G)$.

So, an identifying code is a specific type of **covering** problem.

Density of Identifying Codes in Grids

Let G be the hexagonal grid, and

$$\delta = \inf\{\delta(X) : X \subset V(G) \text{ is an identifying code}\}.$$

Density of Identifying Codes in Grids

Let G be the hexagonal grid, and

$$\delta = \inf\{\delta(X) : X \subset V(G) \text{ is an identifying code}\}.$$

2000 : Cohen, Honkala, Lobstein, and Zémor: $\delta \leq \frac{3}{7} \approx 0.428571$

Density of Identifying Codes in Grids

Let G be the hexagonal grid, and

$$\delta = \inf\{\delta(X) : X \subset V(G) \text{ is an identifying code}\}.$$

1998 : Karpovsky, Chakrabarty, and Levitin $\delta \geq \frac{2}{5} = 0.400000$

2000 : Cohen, Honkala, Lobstein, and Zémor: $\delta \leq \frac{3}{7} \approx 0.428571$

Density of Identifying Codes in Grids

Let G be the hexagonal grid, and

$$\delta = \inf\{\delta(X) : X \subset V(G) \text{ is an identifying code}\}.$$

1998 : Karpovsky, Chakrabarty, and Levitin $\delta \geq \frac{2}{5} = 0.400000$
2000 : Cohen, Honkala, Lobstein, and Zémor $\delta \geq \frac{16}{39} \approx 0.410256$

2000 : Cohen, Honkala, Lobstein, and Zémor: $\delta \leq \frac{3}{7} \approx 0.428571$

Density of Identifying Codes in Grids

Let G be the hexagonal grid, and

$$\delta = \inf\{\delta(X) : X \subset V(G) \text{ is an identifying code}\}.$$

1998 : Karpovsky, Chakrabarty, and Levitin	$\delta \geq \frac{2}{5} = 0.400000$
2000 : Cohen, Honkala, Lobstein, and Zémor	$\delta \geq \frac{16}{39} \approx 0.410256$
2009 : Cranston and Yu	$\delta \geq \frac{12}{29} \approx 0.413793$

2000 : Cohen, Honkala, Lobstein, and Zémor: $\delta \leq \frac{3}{7} \approx 0.428571$

Density of Identifying Codes in Grids

Let G be the hexagonal grid, and

$$\delta = \inf\{\delta(X) : X \subset V(G) \text{ is an identifying code}\}.$$

1998 : Karpovsky, Chakrabarty, and Levitin	$\delta \geq \frac{2}{5} = 0.400000$
2000 : Cohen, Honkala, Lobstein, and Zémor	$\delta \geq \frac{16}{39} \approx 0.410256$
2009 : Cranston and Yu	$\delta \geq \frac{12}{29} \approx 0.413793$
2013 : Cuickerman and Yu	$\delta \geq \frac{5}{12} \approx 0.416666$

2000 : Cohen, Honkala, Lobstein, and Zémor: $\delta \leq \frac{3}{7} \approx 0.428571$

Density of Identifying Codes in Grids

Let G be the hexagonal grid, and

$$\delta = \inf\{\delta(X) : X \subset V(G) \text{ is an identifying code}\}.$$

1998 : Karpovsky, Chakrabarty, and Levitin	$\delta \geq \frac{2}{5} = 0.400000$
2000 : Cohen, Honkala, Lobstein, and Zémor	$\delta \geq \frac{16}{39} \approx 0.410256$
2009 : Cranston and Yu	$\delta \geq \frac{12}{29} \approx 0.413793$
2013 : Cuickerman and Yu	$\delta \geq \frac{5}{12} \approx 0.416666$
2015^{+}: Stolee	$\delta \geq \frac{23}{55} \approx 0.418181$
2000 : Cohen, Honkala, Lobstein, and Zémor:	$\delta \leq \frac{3}{7} \approx 0.428571$

Discharging Arguments

Discharging demonstrates a connection between **local structure** and **global averages**.

Discharging Arguments

Discharging arguments have a few components:

Discharging Arguments

Discharging arguments have a few components:

Chargeable objects are assigned a numeric, “charge” value.

Discharging Arguments

Discharging arguments have a few components:

Chargeable objects are assigned a numeric, “charge” value.

The total charge is somehow connected to our global average, but is **roughly distributed**.

Discharging Arguments

Discharging arguments have a few components:

Chargeable objects are assigned a numeric, “charge” value.

The total charge is somehow connected to our global average, but is **roughly distributed**.

By **discharging** (or **distributing charge**), we aim to make the charge distributed evenly.

Discharging Arguments

Discharging arguments have a few components:

Chargeable objects are assigned a numeric, “charge” value.

The total charge is somehow connected to our global average, but is **roughly distributed**.

By **discharging** (or **distributing charge**), we aim to make the charge distributed evenly.

If the final charge amount is bounded below by the same value, then we have a bound on the **global average**.

Discharging Arguments

Let X be an identifying code in the hexagonal grid.

Discharging Arguments

Let X be an identifying code in the hexagonal grid.

Define $\mu(v) = \begin{cases} 1 & v \in X \\ 0 & x \notin X \end{cases}$.

Discharging Arguments

Let X be an identifying code in the hexagonal grid.

Define $\mu(v) = \begin{cases} 1 & v \in X \\ 0 & x \notin X \end{cases}$.

$$\delta(X) = \limsup_{r\to\infty} \frac{|B_r(v_0) \cap X|}{|B_r(v_0)|} = \limsup_{r\to\infty} \frac{\sum_{v\in B_r(v_0)} \mu(v)}{|B_r(v_0)|}.$$

Discharging Arguments

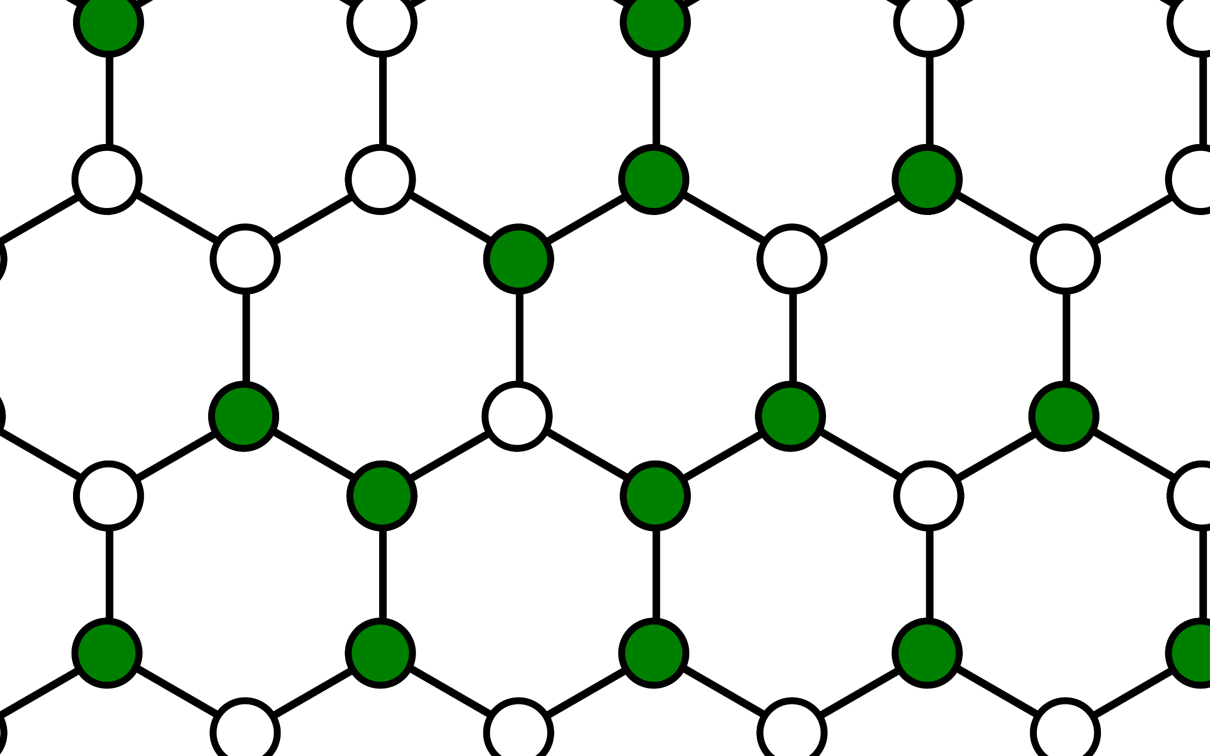
Let X be an identifying code in the hexagonal grid.

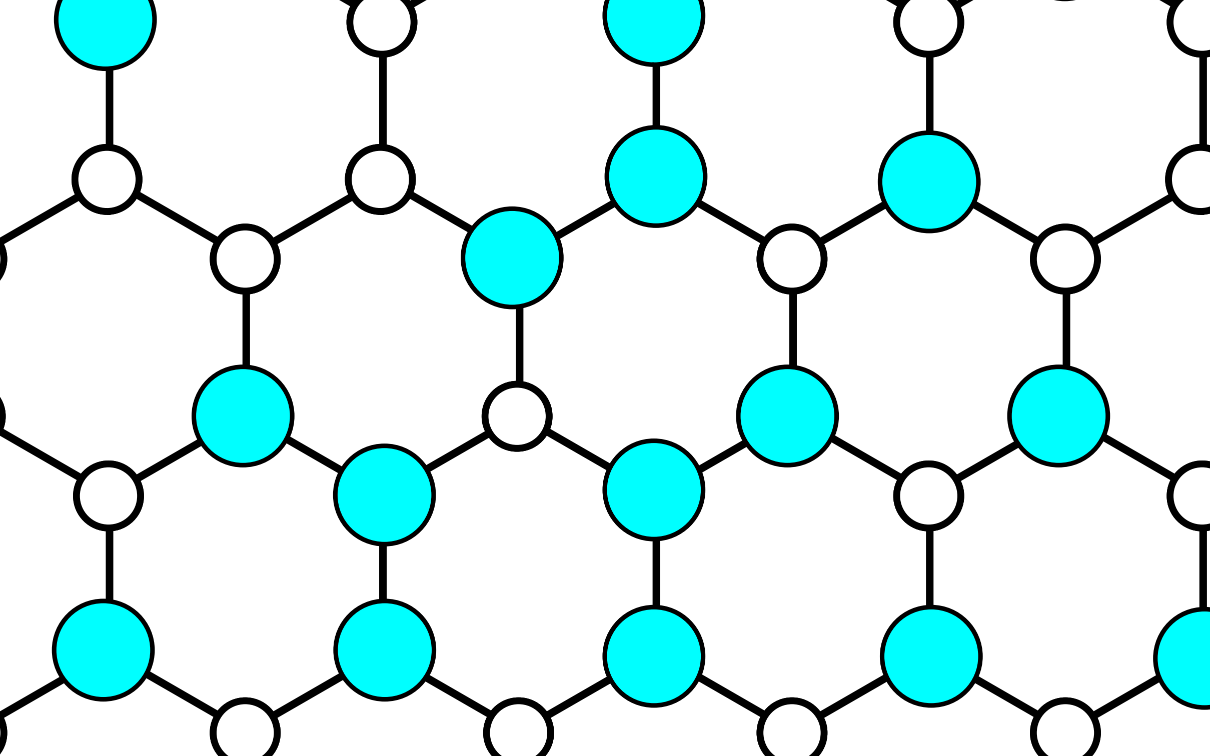
Define $\mu(v) = \begin{cases} 1 & v \in X \\ 0 & x \notin X \end{cases}$.

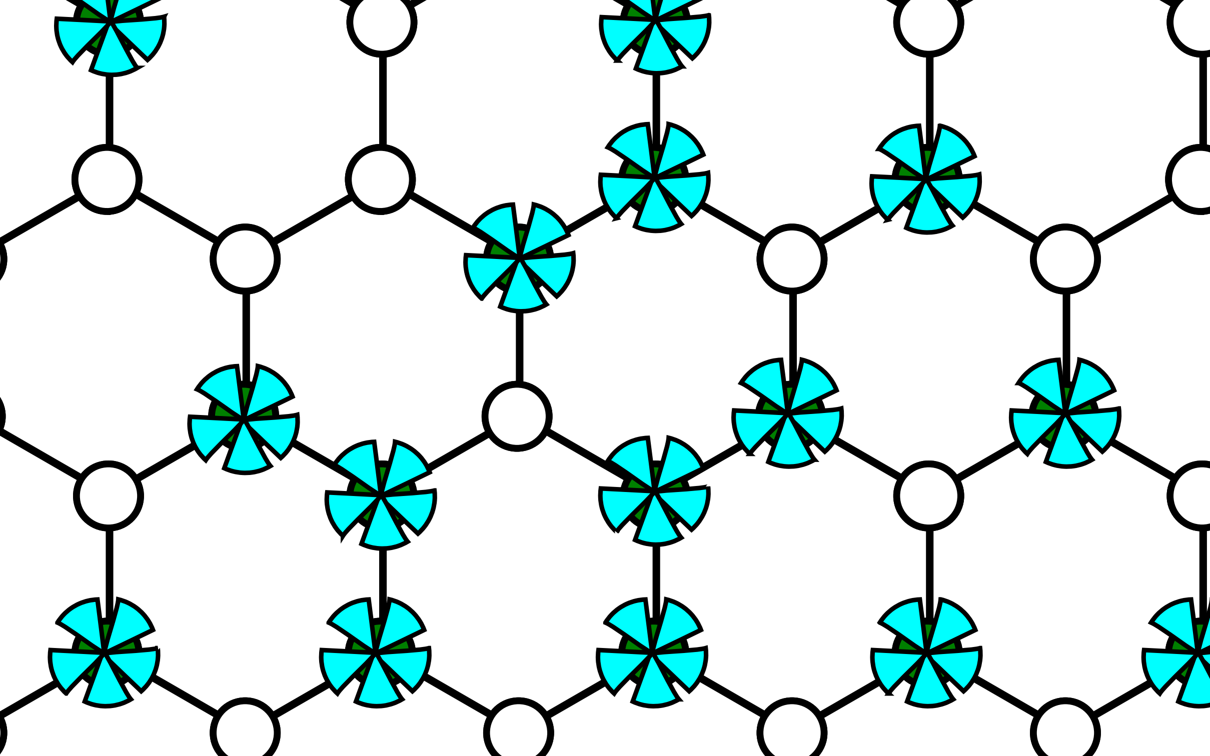
$$\delta(X) = \limsup_{r\to\infty} \frac{|B_r(v_0) \cap X|}{|B_r(v_0)|} = \limsup_{r\to\infty} \frac{\sum_{v\in B_r(v_0)} \mu(v)}{|B_r(v_0)|}.$$

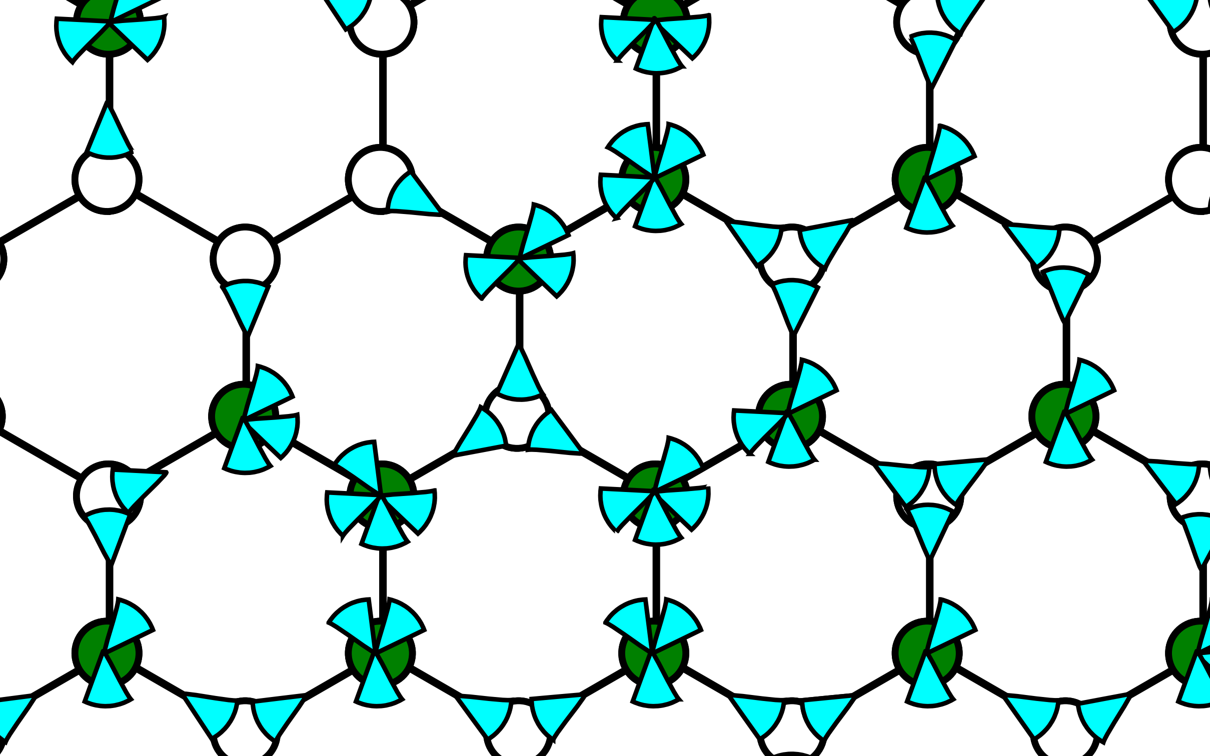
If we **discharge** such that our new charge values $\mu'(v)$ have $\mu'(v) \geq w$ always, then

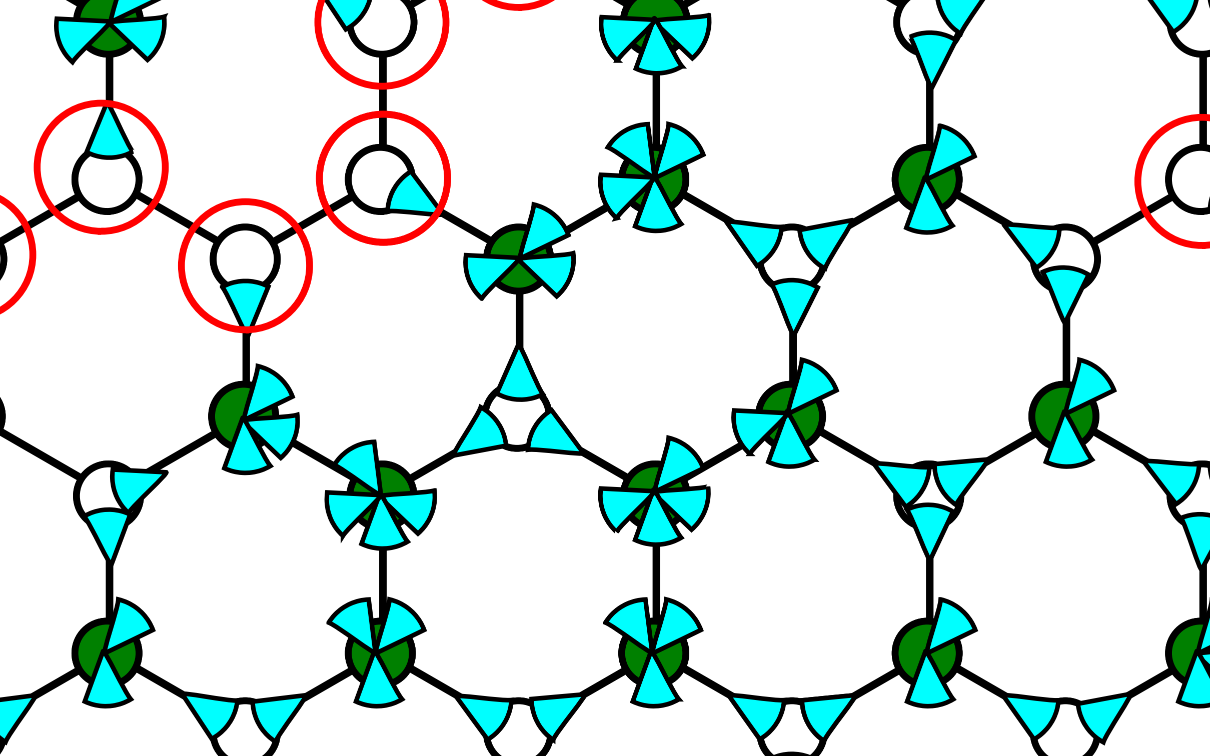
$$\delta(X) = \limsup_{r\to\infty} \frac{\sum_{v\in B_r(v_0)} \mu(v)}{|B_r(v_0)|} = \limsup_{r\to\infty} \frac{\sum_{v\in B_r(v_0)} \mu'(v)}{|B_r(v_0)|} \geq w.$$

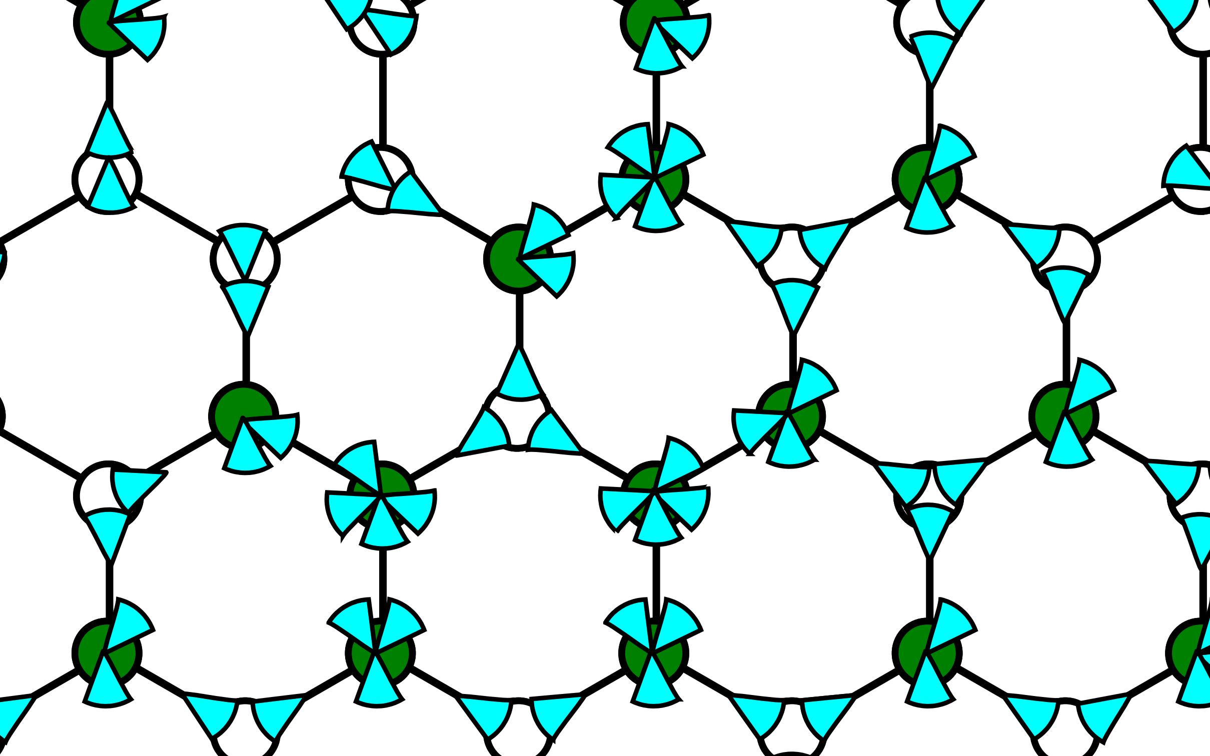






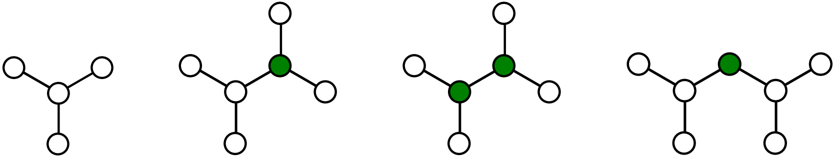






Example Discharging Argument

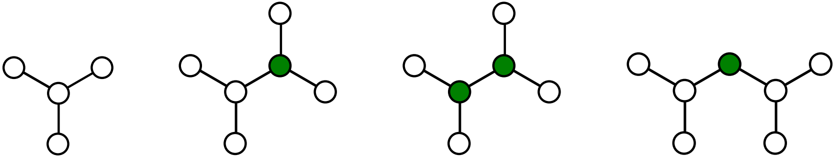
Why did it work?



Forbidden Configurations

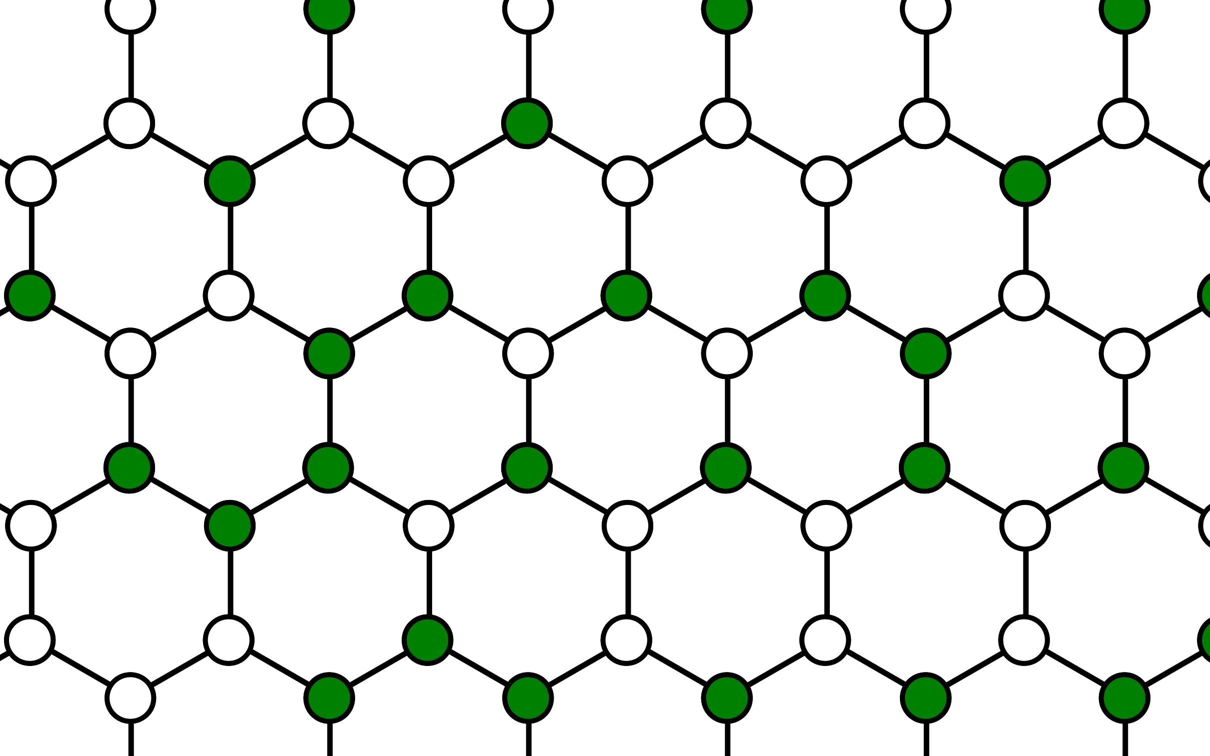
Example Discharging Argument

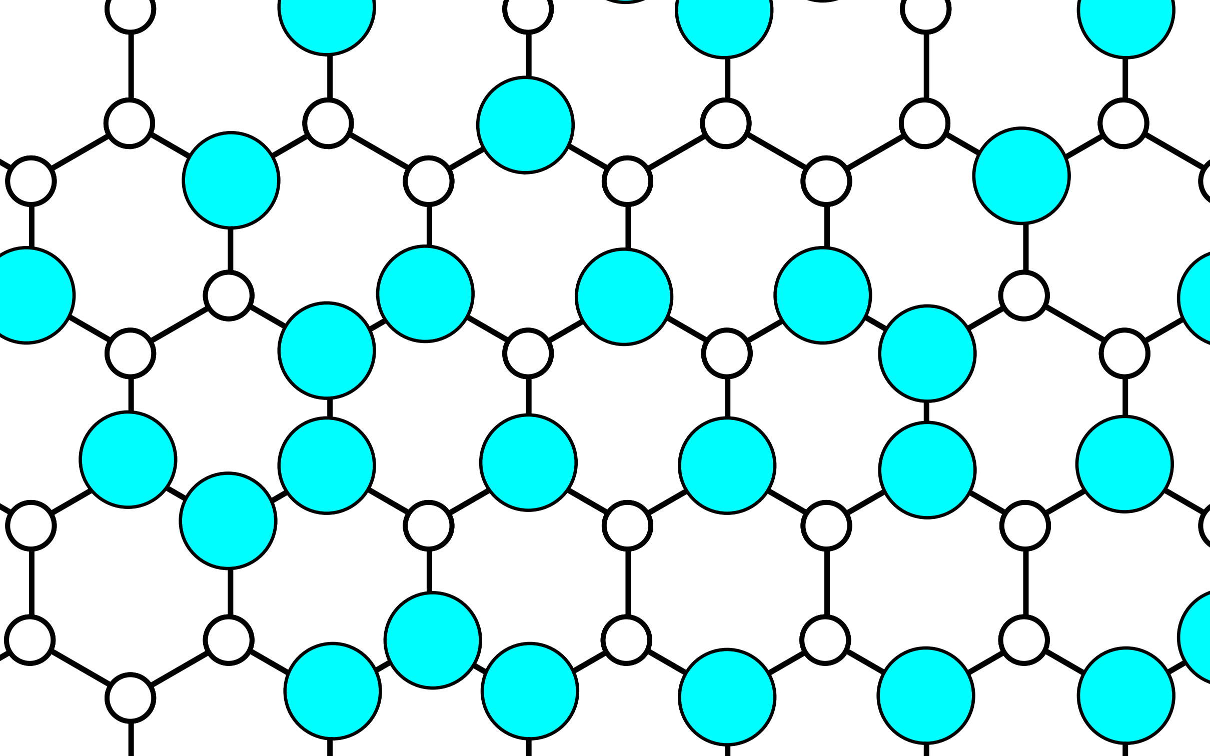
Why did it work?

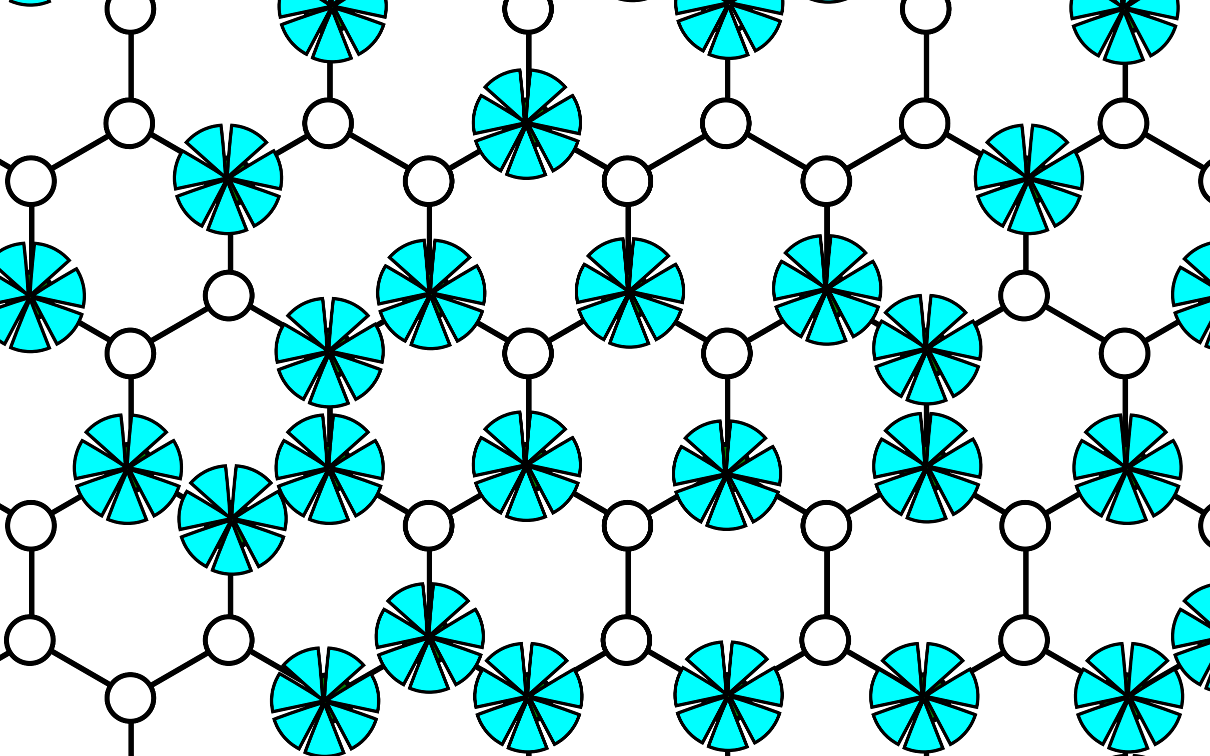


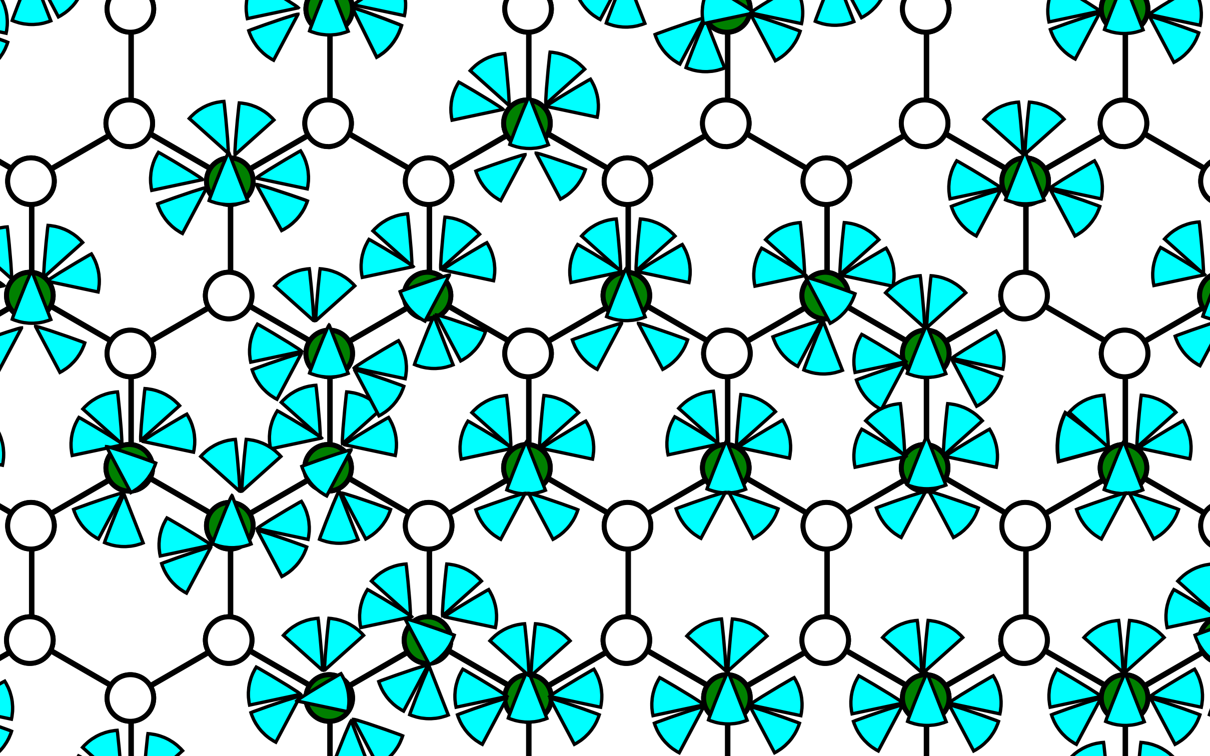
Forbidden Configurations

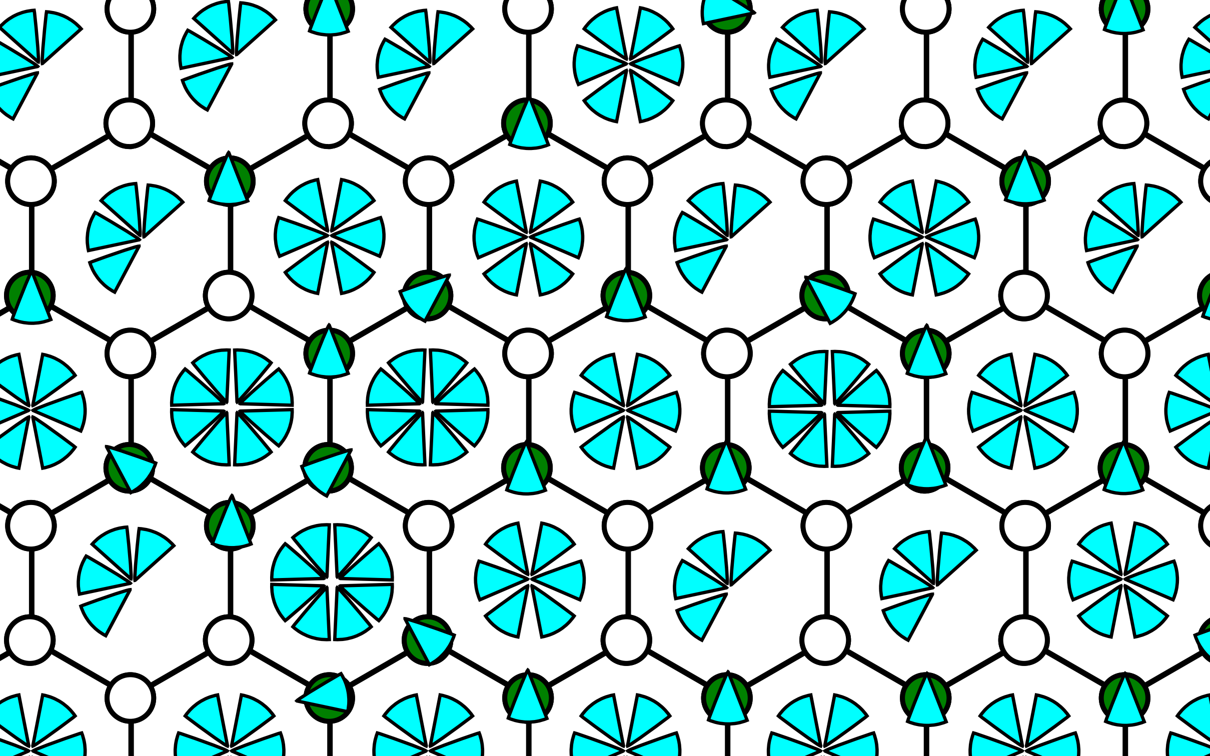
It is also a **sharp** lower bound: $\delta > \frac{2}{5}$ as it is **impossible to construct** a local area where $\mu'(v) = \frac{2}{5}$ for all vertices.

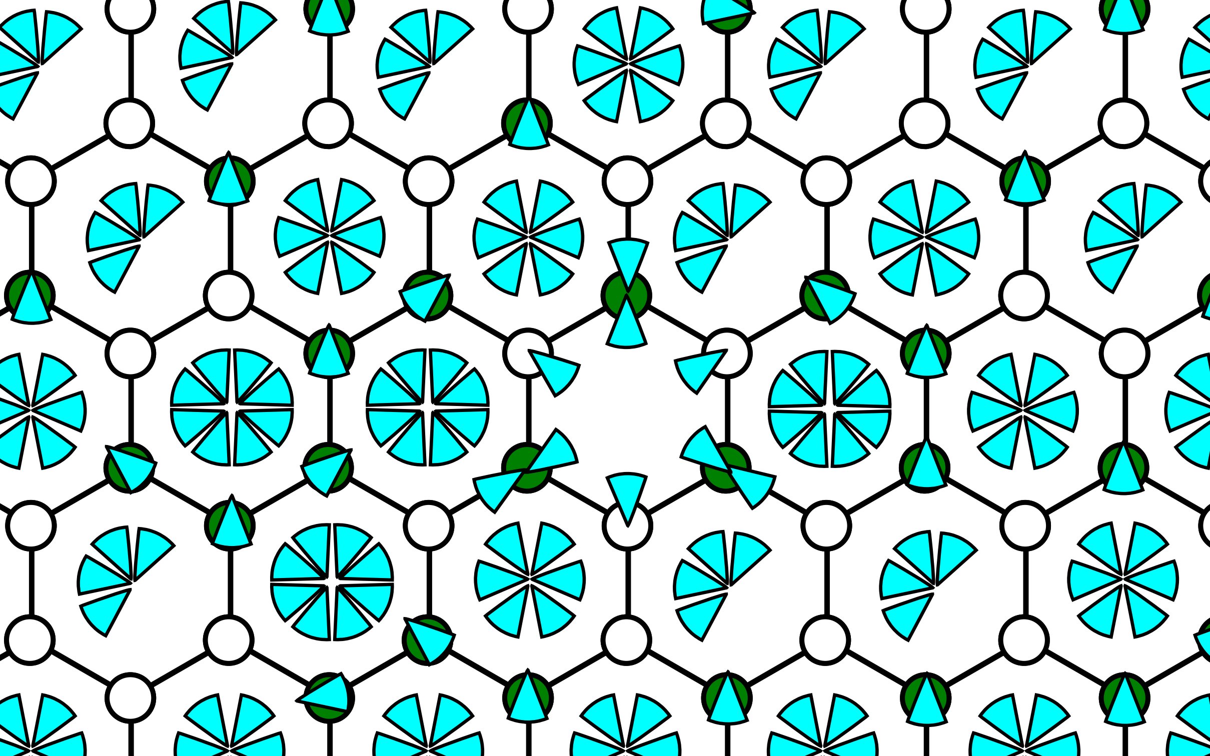


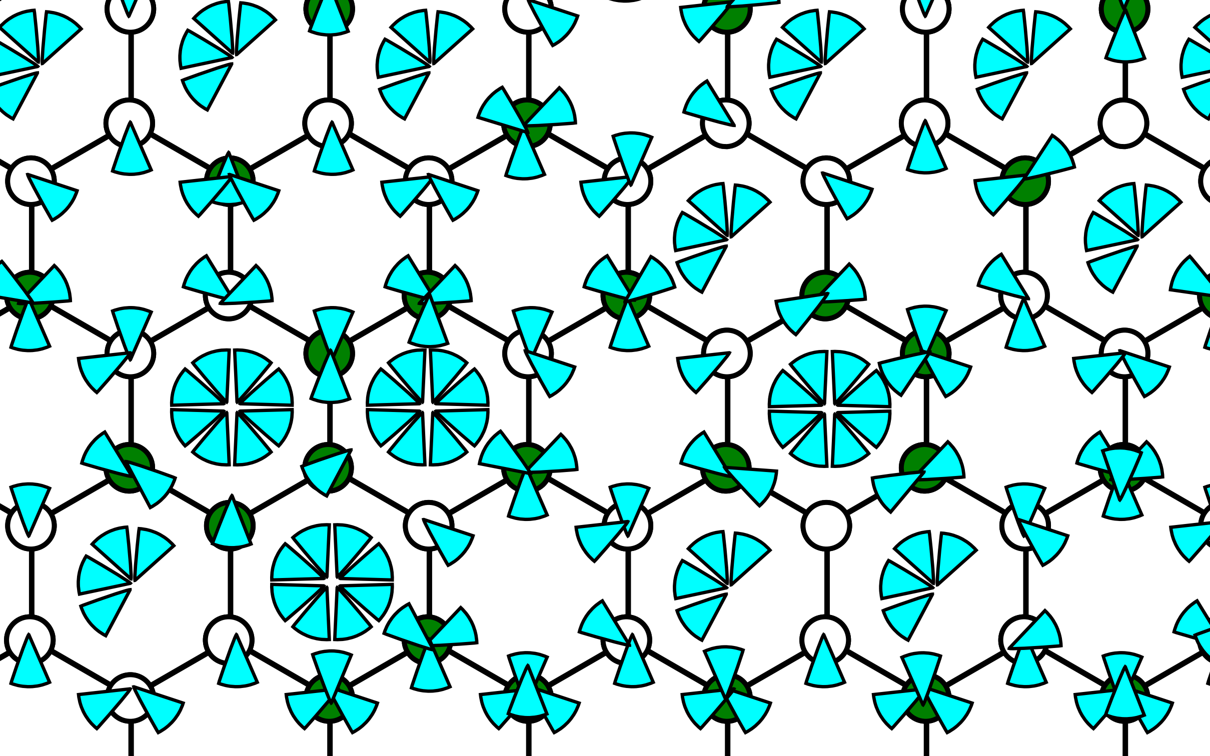


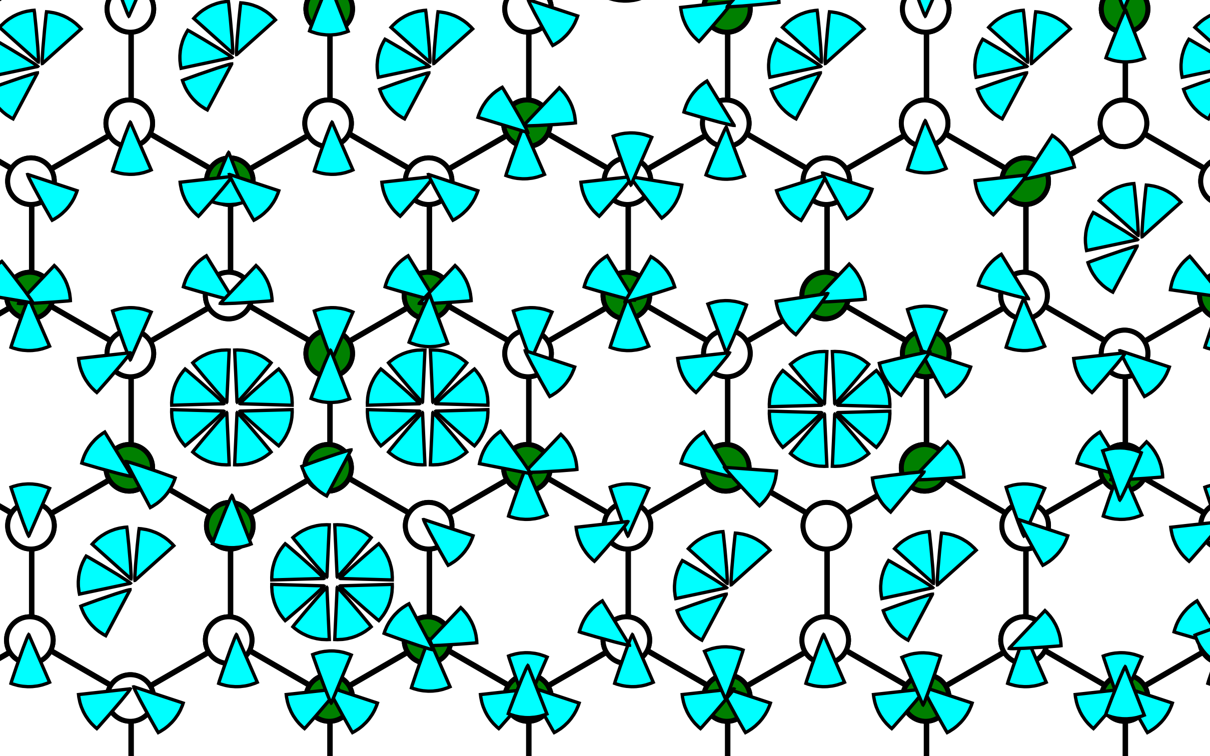


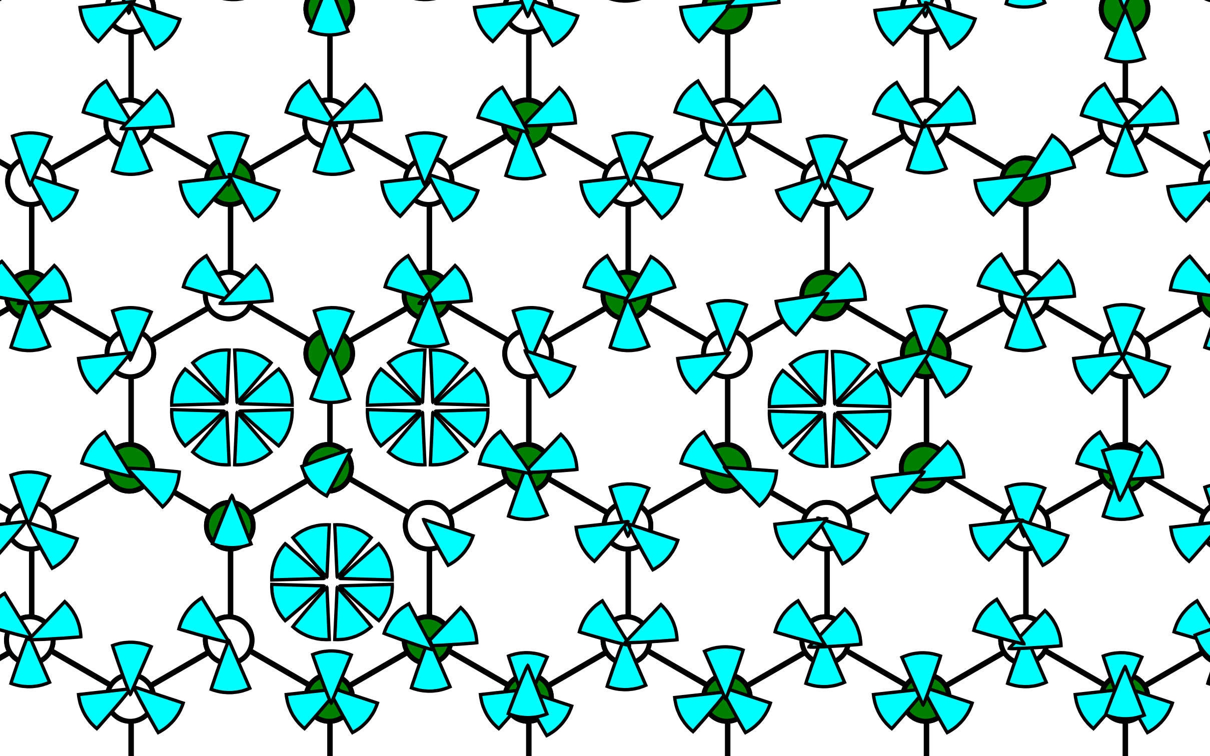


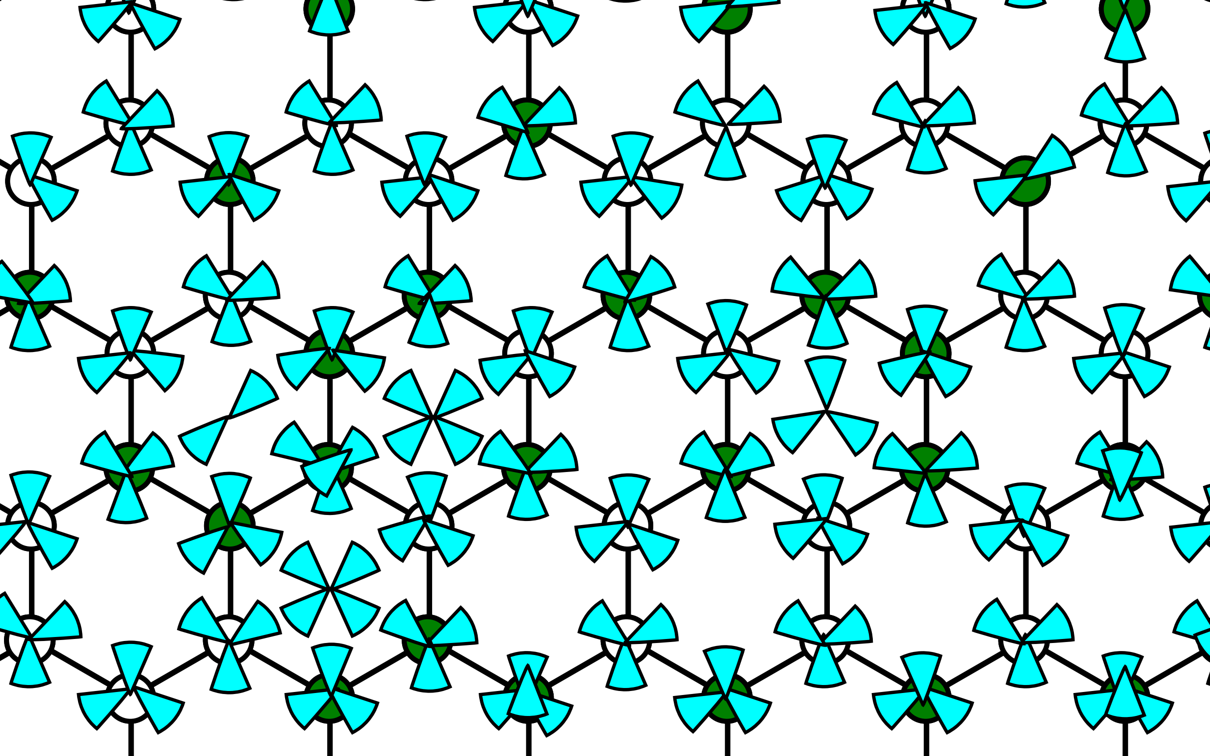


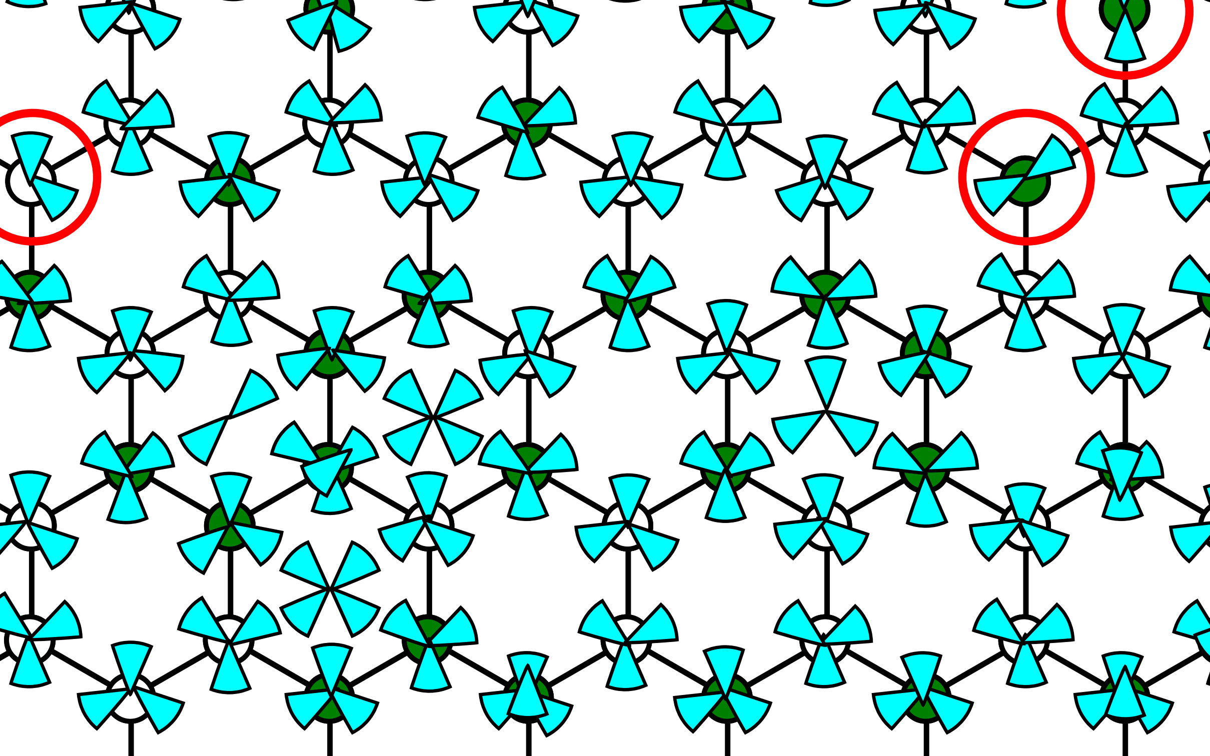


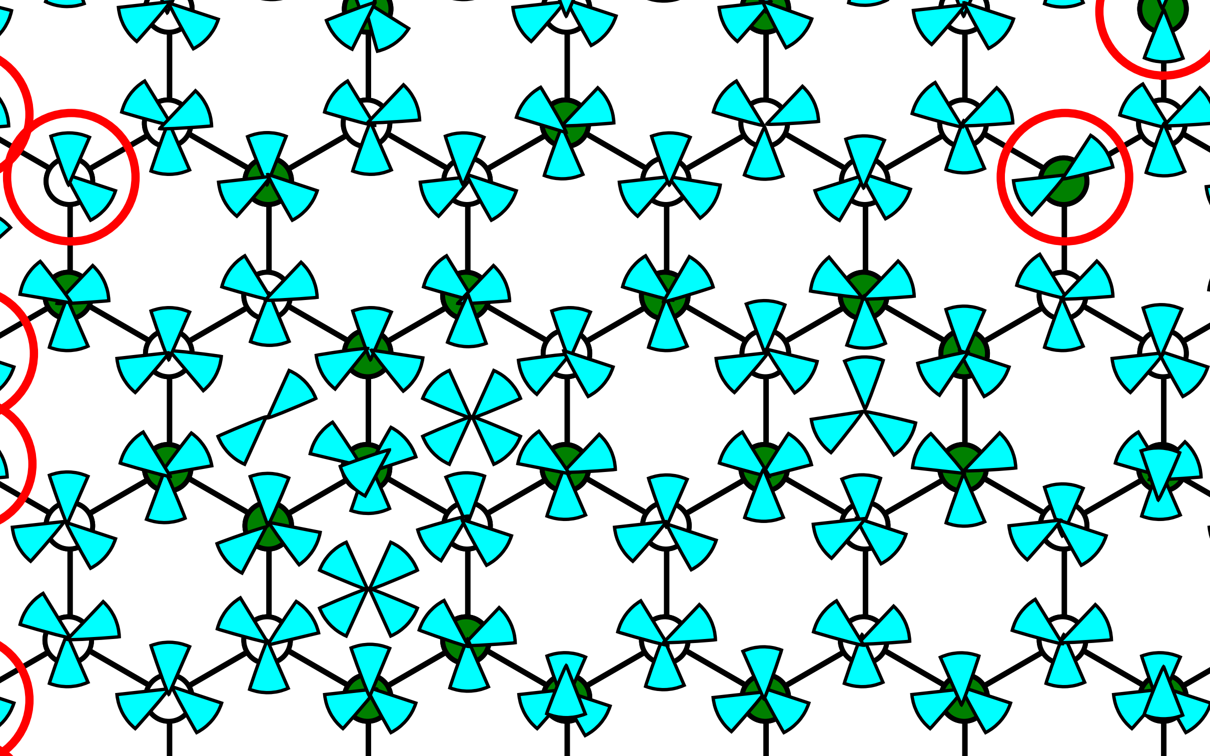


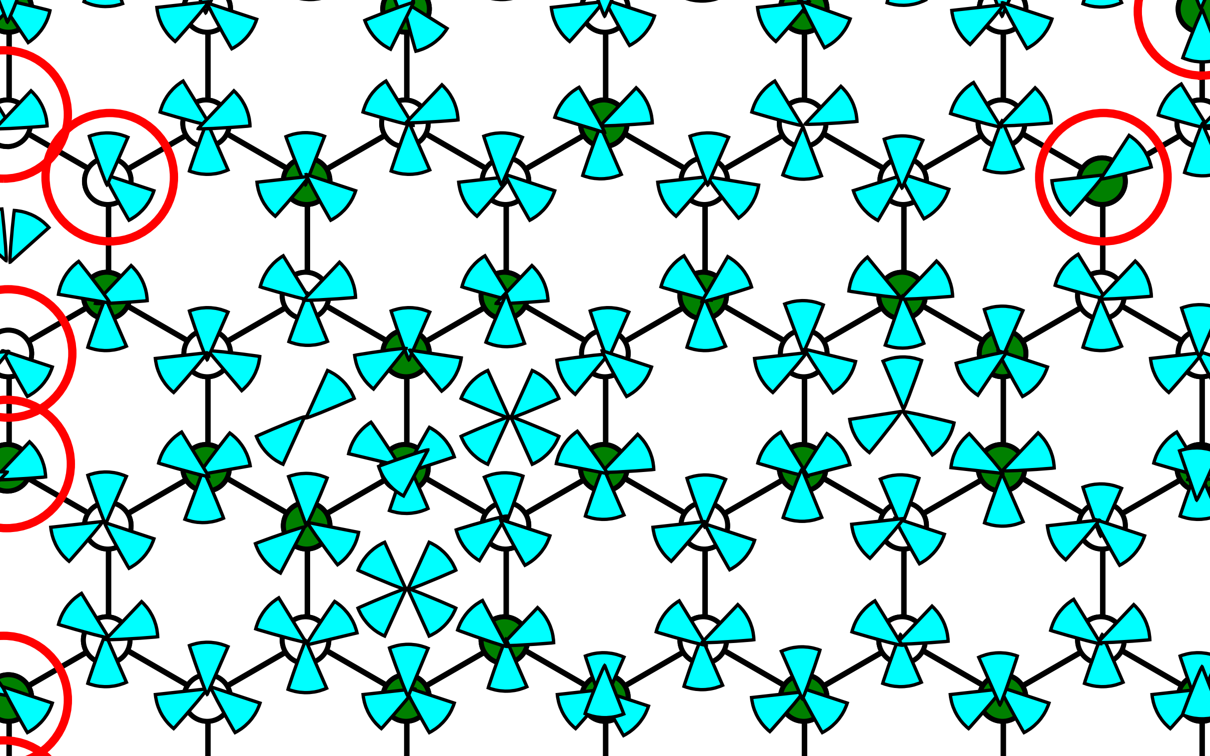


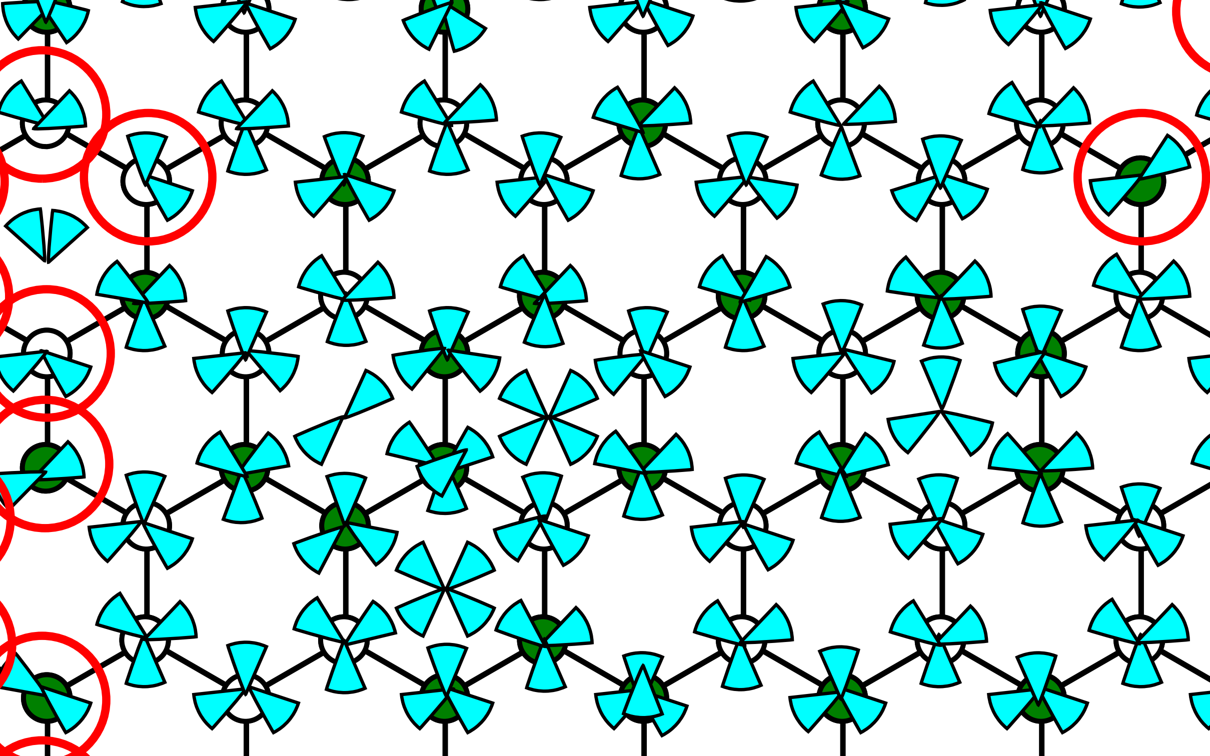


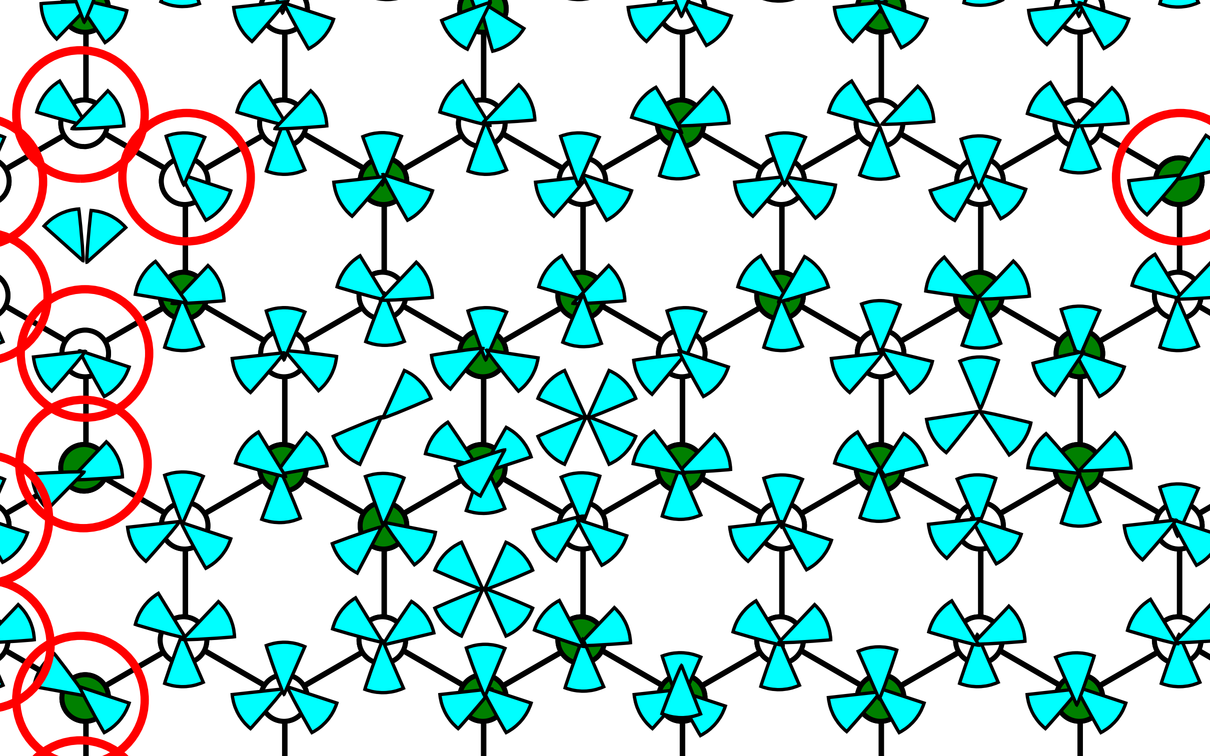


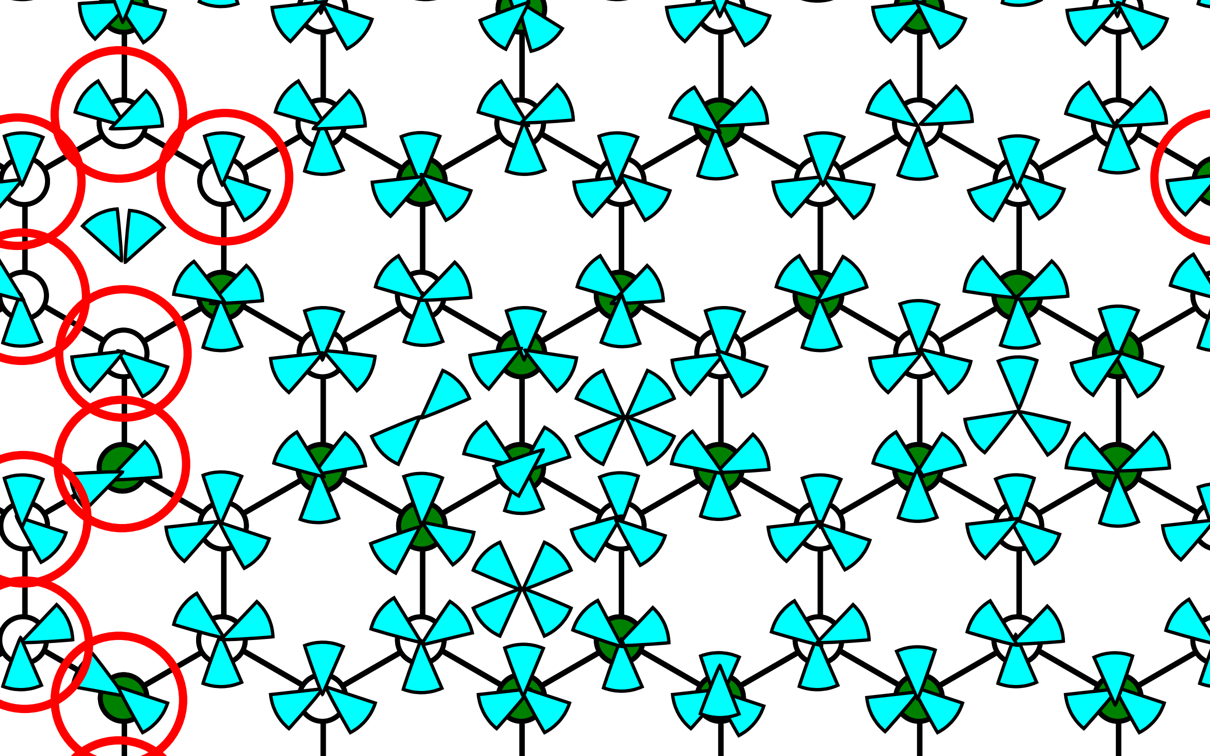












Discharging Arguments

There are a few subtle points:

Discharging Arguments

There are a few subtle points:

We actually have a charge function $\nu(f)$ on the faces: $\nu(f) = 0$.

Discharging Arguments

There are a few subtle points:

We actually have a charge function $\nu(f)$ on the faces: $\nu(f) = 0$.

When we discharge with the faces, we must have that $\nu'(f) \geq 0$ always.

Discharging Arguments

There are a few subtle points:

We actually have a charge function $\nu(f)$ on the faces: $\nu(f) = 0$.

When we discharge with the faces, we must have that $\nu'(f) \geq 0$ always.

The equality

$$\limsup_{r\to\infty} \frac{\sum_{v\in B_r(v_0)} \mu(v)}{|B_r(v_0)|} = \limsup_{r\to\infty} \frac{\sum_{v\in B_r(v_0)} \mu'(v)}{|B_r(v_0)|}$$

holds only when our discharging sends a **bounded amount** of charge a **bounded distance**.

Discharging Arguments

The main difficulty with designing discharging arguments is to balance

- Low-charge objects *receive* enough charge to match the goal value.
- High-charge objects *maintain* enough charge to match the goal value.

We replace the manual “guess-and-check” method with a framework for producing discharging arguments.

We replace the manual “guess-and-check” method with a framework for producing discharging arguments.

Automated **D**ischarging **A**rguments using **GE**neration.

We replace the manual "guess-and-check" method with a framework for producing discharging arguments.

Automated **D**ischarging **A**rguments using **GE**neration.

ADAGE

We replace the manual “guess-and-check” method with a framework for producing discharging arguments.

Automated **D**ischarging **A**rguments using **GE**neration.

ADAGE

A proof using this technique is called an **adage**.

Automated Discharging Arguments

There are three main steps:

Automated Discharging Arguments

There are three main steps:

1. Define the **shape** of the rules.

Automated Discharging Arguments

There are three main steps:

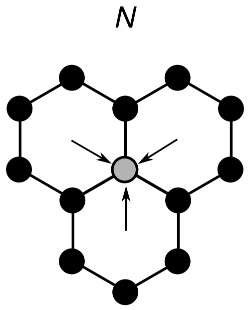
1. Define the **shape** of the rules.
2. Generate **constraints** on the rule values.

Automated Discharging Arguments

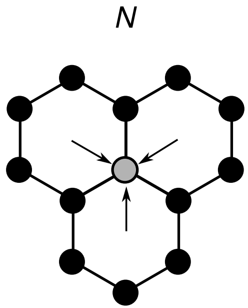
There are three main steps:

1. Define the **shape** of the rules.
2. Generate **constraints** on the rule values.
3. **Optimize** the values.

Generating Rules



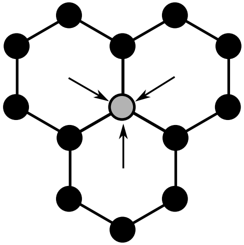
Generating Rules



1758 instances of this rule.

Generating Rules

N

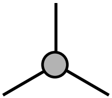


1758 instances of this rule.

We do not assign values to these rules! Only variables!

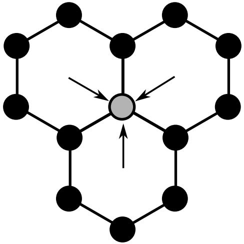
Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



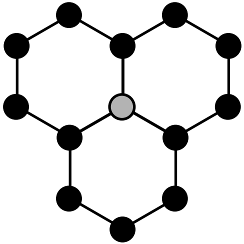
Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



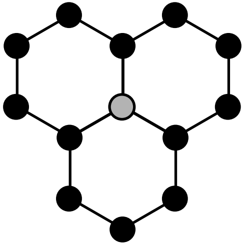
Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.

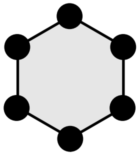


Example constraints:

$$
\begin{array}{ccccccccc}
1 & + & x_{124} & + & x_{125} & + & x_{126} & \geq & w \\
0 & + & 3x_{456} & & & & & \geq & w
\end{array}
$$

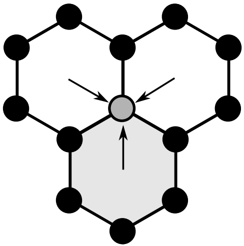
Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



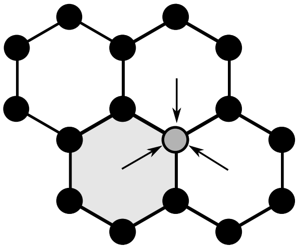
Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



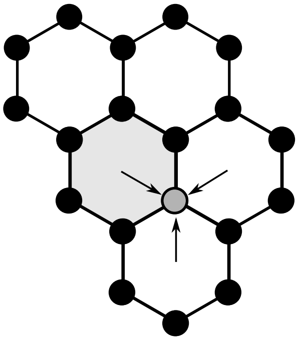
Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



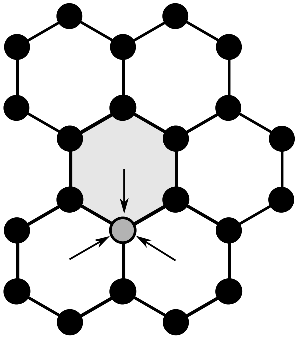
Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



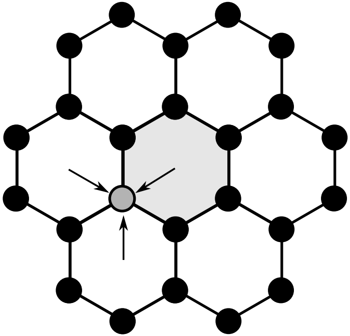
Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



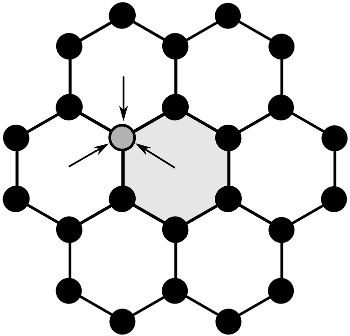
Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



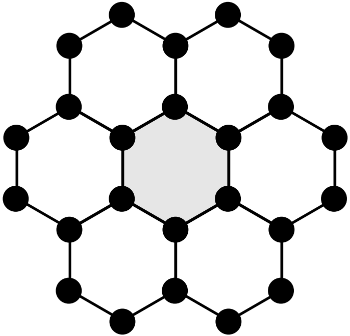
Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



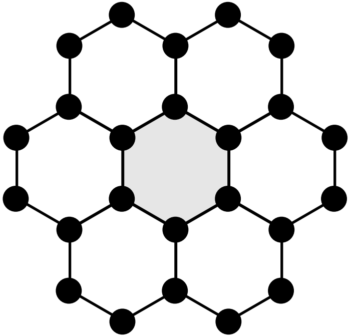
Generating Constraints

Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



Generating Constraints

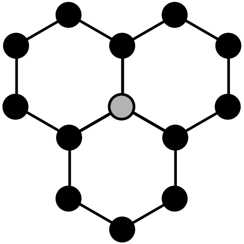
Given a set of rules, we must constrain the values of the rules such that we meet our goal charge values.



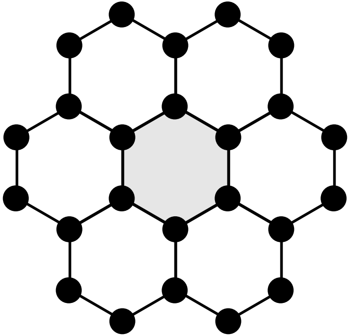
Example constraints:

$$
\begin{aligned}
-x_1 - x_{12} - x_{126} - x_{356} - x_{600} - x_{1563} &\geq 0 \\
- x_5 - 2x_{345} - 3x_{1260} &\geq 0
\end{aligned}
$$

Generating Constraints



Vertex
1,758 Constraints



Face
663,662 Constraints

1,758 Rules with 5,238 Variables.

Solving the Linear Program

To assign value to the rules, we create the following linear program:

$$\begin{array}{lrcll} \max & w & & & \\ & \mu'(v) & \geq & w & \forall v \in V(G) \\ & \nu'(f) & \geq & 0 & \forall f \in F(G) \end{array}$$

Solving the Linear Program

To assign value to the rules, we create the following linear program:

$$\begin{array}{lrcrcl}
\max & & & w & & \\
& \mu(v) & + & \sum_{f \in F(G)} D(f, v) & \geq & w \quad \forall v \in V(G) \\
& \nu(f) & - & \sum_{v \in V(G)} D(f, v) & \geq & 0 \quad \forall f \in F(G)
\end{array}$$

Solving the Linear Program

To assign value to the rules, we create the following linear program:

$$
\begin{array}{lrcl}
\max & & w & \\
& 1 + \sum_{f \in F(G)} D(f, v) & \geq w & \forall v \in X \\
& 0 + \sum_{f \in F(G)} D(f, v) & \geq w & \forall v \in V(G) \setminus X \\
& 0 - \sum_{v \in V(G)} D(f, v) & \geq 0 & \forall f \in F(G)
\end{array}
$$

Solving the Linear Program

To assign value to the rules, we create the following linear program:

$$
\begin{array}{rrrrl}
\max & & w & & \\
\sum_{f \in F(G)} D(f,v) & - & w & \geq & -1 \quad \forall v \in X \\
\sum_{f \in F(G)} D(f,v) & - & w & \geq & 0 \quad \forall v \in V(G) \setminus X \\
-\sum_{v \in V(G)} D(f,v) & & & \geq & 0 \quad \forall f \in F(G) \\
D(f,v), & & w & & \text{free}
\end{array}
$$

Results

Theorem

Let X be an identifying code in the hexagonal grid. The adage proof using rule N demonstrates a lower bound of $\delta(X) \geq \frac{23}{55} = 0.4\overline{18}$.

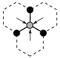
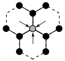
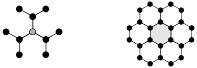
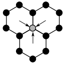
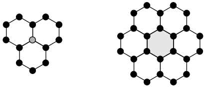
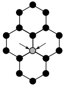
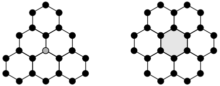
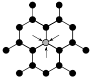
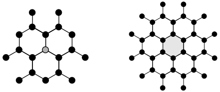
Results

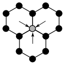
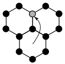
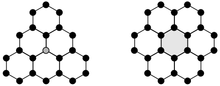
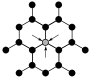
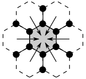
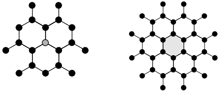
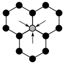
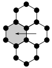
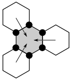
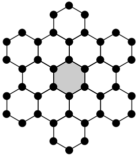
Theorem

Let X be an identifying code in the hexagonal grid. The adage proof using rule N demonstrates a lower bound of $\delta(X) \geq \frac{23}{55} = 0.4\overline{18}$.

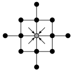
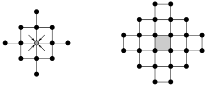
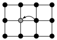
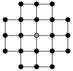
This improves the previous-best lower bound of Cuickerman & Yu ($\frac{5}{12} = 0.41\overline{6}$) but does not match the current-best upper bound ($\frac{3}{7} \approx 0.42857$).

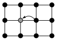
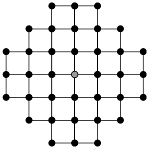
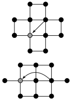
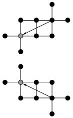
Other Rule Sets in Hexagonal Grid

Rules		Constraint Confgurations
V_1		
V_2		
N		
N^+		
V_3		

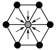
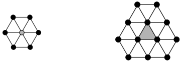
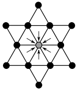
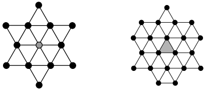
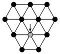
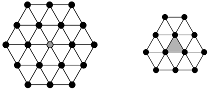
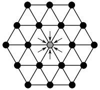
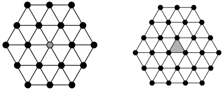
Rules		Constraint Confgurations
N J_2	 	
V_3 F_6	 	
N $F_{1,3}$ Y_{size}	  	

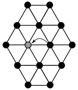
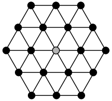
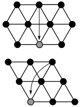
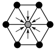
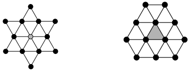
Other Rule Sets in Square Grid

Rules		Constraint Confgurations
V_1		
N		
V_2		
C_1^+		

Rules		Constraint Confgurations
C_1^+		
C_2^+		
C_3^+		

Other Rule Sets in Triangular Grid

Rules		Constraint Confgurations
V_1		
S		
N^+		
V_2		

Rules		Constraint Confgurations
C_1^+		
C_2^+		
V_1		
J_2		

Results for Variations on Identifying Codes

Set Type	**Hexagonal Grid**		**Square Grid**		**Triangular Grid**	
Dominating Set	V_1	$\frac{1}{4} \approx 0.250000^*$	V_1	$\frac{1}{5} \approx 0.200000^*$	V_1	$\frac{1}{7} \approx 0.142857^*$
Identifying Code	N	$\frac{23}{55} \approx 0.418182^\dagger$	V_2	$\frac{7}{20} \approx 0.350000^*$	V_1	$\frac{1}{4} \approx 0.250000^*$
Strong Identifying Code	V_2	$\frac{8}{17} \approx 0.470588$	$C_1 \cup C_2$	$\frac{7}{18} \approx 0.388889$	$C_1^+ \cup C_2^+$	$\frac{4}{13} \approx 0.307692$
Locating-Dominating Code	V_2	$\frac{1}{3} \approx 0.333333^*$	V_2	$\frac{3}{10} \approx 0.300000^*$	$C_1 \cup C_2$	$\frac{12}{53} \approx 0.226415$
Open-Locating-Dominating Code	V_2	$\frac{1}{2} \approx 0.500000^*$	C_1^+	$\frac{2}{5} \approx 0.400000^*$	C_1^+	$\frac{4}{13} \approx 0.307692^*$

Future Work

Future Work

1. More discharging rules, more adage proofs. *Can we do better?*

Future Work

1. More discharging rules, more adage proofs. *Can we do better?*

2. Build a white-box implementation of linear programming. Perhaps use a primal-dual algorithm?

Future Work

1. More discharging rules, more adage proofs. *Can we do better?*

2. Build a white-box implementation of linear programming. Perhaps use a primal-dual algorithm?

3. Extend framework to coloring problems on planar graphs.

Future Work

1. More discharging rules, more adage proofs. *Can we do better?*
2. Build a white-box implementation of linear programming. Perhaps use a primal-dual algorithm?
3. Extend framework to coloring problems on planar graphs.
4. Use discharging as *combinatorial dual* for finite combinatorial optimization problems.

Automated Discharging Arguments for Density Problems in Grids

Derrick Stolee

Iowa State University
dstolee@iastate.edu
http://www.math.iastate.edu/dstolee/

September 2, 2014
Mathematics Department Colloquium