How (Not) to Compute Domatic Partitions of Graphs

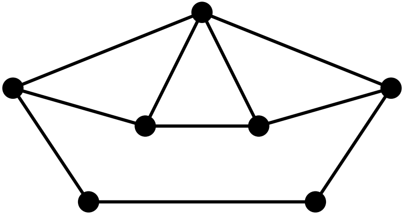
Oscar Levin

University of Northern Colorado

Rocky Mountain Section of the MAA meeting
April 14, 2012

Dominating Sets in Graphs

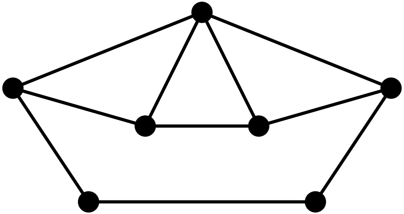
Given a graph, we look for sets of vertices close to everything.



A set is *dominating* if every vertex of G is in, or adjacent to a vertex in, the set.

Dominating Sets in Graphs

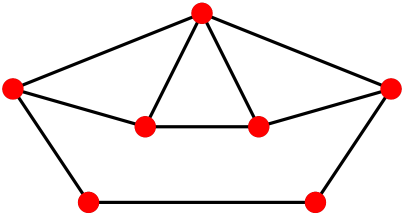
Given a graph, we look for sets of vertices close to everything.



A set is *dominating* if every vertex of G is in, or adjacent to a vertex in, the set.

Dominating Sets in Graphs

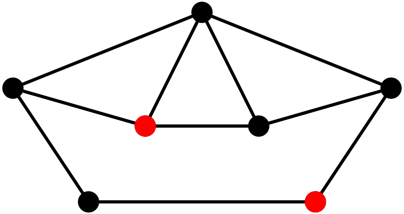
Given a graph, we look for sets of vertices close to everything.



A set is *dominating* if every vertex of G is in, or adjacent to a vertex in, the set.

Dominating Sets in Graphs

Given a graph, we look for sets of vertices close to everything.



A set is *dominating* if every vertex of G is in, or adjacent to a vertex in, the set.

Domatic Partitions

Definition

A *domatic k-partition* of a graph G is a partition of (all) the vertices of G into k (disjoint) dominating sets.

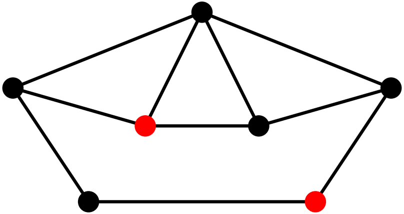
The *domatic number* of a graph is the size of the largest domatic partition.

Domatic Partitions

Definition

A *domatic k-partition* of a graph G is a partition of (all) the vertices of G into k (disjoint) dominating sets.

The *domatic number* of a graph is the size of the largest domatic partition.

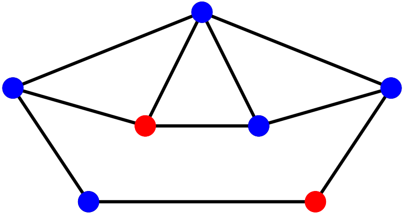


Domatic Partitions

Definition

A *domatic k-partition* of a graph G is a partition of (all) the vertices of G into k (disjoint) dominating sets.

The *domatic number* of a graph is the size of the largest domatic partition.

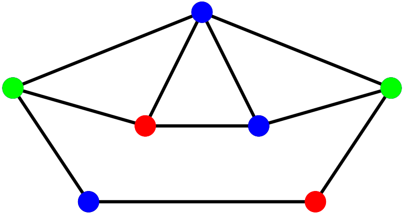


Domatic Partitions

Definition

A *domatic k-partition* of a graph G is a partition of (all) the vertices of G into k (disjoint) dominating sets.

The *domatic number* of a graph is the size of the largest domatic partition.

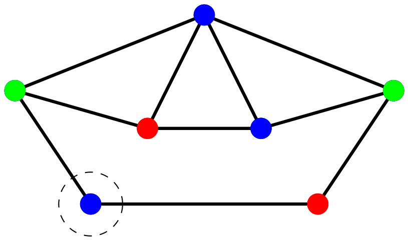


Domatic Partitions

Definition

A *domatic k-partition* of a graph G is a partition of (all) the vertices of G into k (disjoint) dominating sets.

The *domatic number* of a graph is the size of the largest domatic partition.



Main Question

Question

Given a graph, how hard is it to find a domatic partition?

Is there an algorithm which computes a domatic partition for all graphs?

Including infinite graphs?

What does that even mean?

Main Question

Question

Given a graph, how hard is it to find a domatic partition?

Is there an algorithm which computes a domatic partition for all graphs?

Including infinite graphs?

What does that even mean?

Main Question

Question

Given a graph, how hard is it to find a domatic partition?

Is there an algorithm which computes a domatic partition for all graphs?

Including infinite graphs?

What does that even mean?

Main Question

Question

Given a graph, how hard is it to find a domatic partition?

Is there an algorithm which computes a domatic partition for all graphs?

Including infinite graphs?

What does that even mean?

Computable Graphs

Definition

A graph G is computable if the edge relation is computable. That is, there is an algorithm which, when given vertices v_1, v_2, decides $E(v_1, v_2)$.

Question (restated)

Given a computable graph, is there a computable function $\varphi(x)$ which outputs which set in a domatic partition an input vertex belongs to?

Is there a computable *domatic partition?*

Computable Graphs

Definition

A graph G is computable if the edge relation is computable. That is, there is an algorithm which, when given vertices v_1, v_2, decides $E(v_1, v_2)$.

Question (restated)

Given a computable graph, is there a computable function $\varphi(x)$ which outputs which set in a domatic partition an input vertex belongs to?

Is there a computable *domatic partition?*

The Answer for 2-Partitions

Suppose G has a domatic 2-partition (so no isolated vertices).
There is an algorithm which produces a domatic 2-partition.

Vertices: $\{v_0, v_1, v_2, \ldots\}$

Put $v_0 \in A$.

Put $v_n \in B$ iff there is an adjacent vertex $v_k \in A$ (with $k < n$)

A is a dominating set: if $v_n \notin A$ then ...

B is a dominating set: if $v_n \notin B$ then ...

The Answer for 2-Partitions

Suppose G has a domatic 2-partition (so no isolated vertices). There is an algorithm which produces a domatic 2-partition.

Vertices: $\{v_0, v_1, v_2, \ldots\}$

Put $v_0 \in A$.

Put $v_n \in B$ iff there is an adjacent vertex $v_k \in A$ (with $k < n$)

A is a dominating set: if $v_n \notin A$ then ...

B is a dominating set: if $v_n \notin B$ then ...

The Answer for 2-Partitions

Suppose G has a domatic 2-partition (so no isolated vertices). There is an algorithm which produces a domatic 2-partition.

Vertices: $\{v_0, v_1, v_2, \ldots\}$

Put $v_0 \in A$.

Put $v_n \in B$ iff there is an adjacent vertex $v_k \in A$ (with $k < n$)

A is a dominating set: if $v_n \notin A$ then …

B is a dominating set: if $v_n \notin B$ then …

The Answer for 2-Partitions

Suppose G has a domatic 2-partition (so no isolated vertices).
There is an algorithm which produces a domatic 2-partition.

Vertices: $\{v_0, v_1, v_2, \ldots\}$

Put $v_0 \in A$.

Put $v_n \in B$ iff there is an adjacent vertex $v_k \in A$ (with $k < n$)

A is a dominating set: if $v_n \notin A$ then ...

B is a dominating set: if $v_n \notin B$ then ...

The Answer for 2-Partitions

Suppose G has a domatic 2-partition (so no isolated vertices). There is an algorithm which produces a domatic 2-partition.

Vertices: $\{v_0, v_1, v_2, \ldots\}$

Put $v_0 \in A$.

Put $v_n \in B$ iff there is an adjacent vertex $v_k \in A$ (with $k < n$)

A is a dominating set: if $v_n \notin A$ then ...

B is a dominating set: if $v_n \notin B$ then ...

The Answer for 2-Partitions

Suppose G has a domatic 2-partition (so no isolated vertices).
There is an algorithm which produces a domatic 2-partition.

Vertices: $\{v_0, v_1, v_2, \ldots\}$

Put $v_0 \in A$.

Put $v_n \in B$ iff there is an adjacent vertex $v_k \in A$ (with $k < n$)

A is a dominating set: if $v_n \notin A$ then ...

B is a dominating set: if $v_n \notin B$ then ...

The Answer for 2-Partitions

Suppose G has a domatic 2-partition (so no isolated vertices). There is an algorithm which produces a domatic 2-partition.

Vertices: $\{v_0, v_1, v_2, \ldots\}$

Put $v_0 \in A$.

Put $v_n \in B$ iff there is an adjacent vertex $v_k \in A$ (with $k < n$)

A is a dominating set: if $v_n \notin A$ then ...

B is a dominating set: if $v_n \notin B$ then ...

The Answer for 3-Partitions

Proposition

There is a computable graph with domatic number 3 with no computable domatic 3-partition.

To proof this, we diagonalize against all computable functions.

The Answer for 3-Partitions

Proposition

There is a computable graph with domatic number 3 with no computable domatic 3-partition.

To proof this, we diagonalize against all computable functions.

Some Ideas from Computability Theory

There is an effective list of all algorithms:

$$\varphi_0, \varphi_1, \varphi_2, \cdots$$

These can be simulated by a *universal* algorithm

We can run these programs "simultaneously" to see if any look like they compute a domatic 3-partition.

Meanwhile, we build a computable graph with a 3-partition

When some φ_e tries to compute a 3-partition, we thwart it.

Some Ideas from Computability Theory

There is an effective list of all algorithms:

$$\varphi_0, \varphi_1, \varphi_2, \cdots$$

These can be simulated by a *universal* algorithm

We can run these programs "simultaneously" to see if any look like they compute a domatic 3-partition.

Meanwhile, we build a computable graph with a 3-partition

When some φ_e tries to compute a 3-partition, we thwart it.

Some Ideas from Computability Theory

There is an effective list of all algorithms:

$$\varphi_0, \varphi_1, \varphi_2, \dots$$

These can be simulated by a *universal* algorithm

We can run these programs "simultaneously" to see if any look like they compute a domatic 3-partition.

Meanwhile, we build a computable graph with a 3-partition

When some φ_e tries to compute a 3-partition, we thwart it.

Some Ideas from Computability Theory

There is an effective list of all algorithms:

$$\varphi_0, \varphi_1, \varphi_2, \cdots$$

These can be simulated by a *universal* algorithm

We can run these programs "simultaneously" to see if any look like they compute a domatic 3-partition.

Meanwhile, we build a computable graph with a 3-partition

When some φ_e tries to compute a 3-partition, we thwart it.

Some Ideas from Computability Theory

There is an effective list of all algorithms:

$$\varphi_0, \varphi_1, \varphi_2, \cdots$$

These can be simulated by a *universal* algorithm

We can run these programs "simultaneously" to see if any look like they compute a domatic 3-partition.

Meanwhile, we build a computable graph with a 3-partition

When some φ_e tries to compute a 3-partition, we thwart it.

The Construction

G will start with copies of K_4, one for each φ_e.

Build G in stages. At each stage, build a new K_4 and check whether φ_e has halted on its copy of K_4.

If φ_e looks like it computes a 3-partition on its K_4, spring the trap!

The Construction

G will start with copies of K_4, one for each φ_e.

Build G in stages. At each stage, build a new K_4 and check whether φ_e has halted on its copy of K_4.

If φ_e looks like it computes a 3-partition on its K_4, spring the trap!

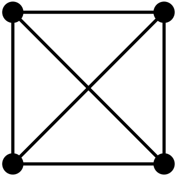
The Construction

G will start with copies of K_4, one for each φ_e.

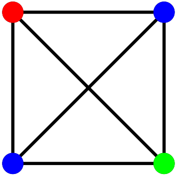
Build G in stages. At each stage, build a new K_4 and check whether φ_e has halted on its copy of K_4.

If φ_e looks like it computes a 3-partition on its K_4, spring the trap!

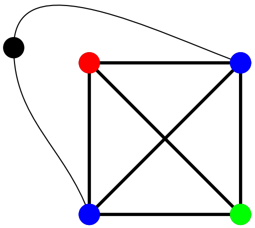
The Trap



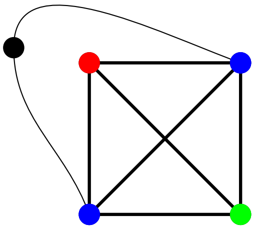
The Trap



The Trap

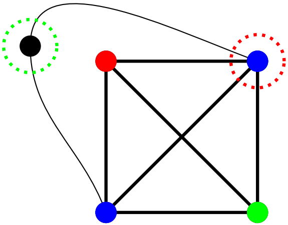


The Trap



The sprung trap still has a 3-partition, but not the one φ_e claims.

The Trap



The sprung trap still has a 3-partition, but not the one φ_e claims.

Worse than that...

Proposition

For any k, there is a computable graph with domatic k-partition but no computable 3-partition.

Even if we know the degrees of all vertices, we still can't do it:

Proposition

There is a highly computable graph with domatic 3-partition but no computable 3-partition.

Worse than that. . .

Proposition

For any k, there is a computable graph with domatic k-partition but no computable 3-partition.

Even if we know the degrees of all vertices, we still can't do it:

Proposition

There is a highly computable graph with domatic 3-partition but no computable 3-partition.

There is no first order information about a graph which helps compute domatic partitions.

Proposition

The question of whether a computable graph has a domatic 3-partition is Σ^1_1-complete.

Basically this says that in order to determine whether a graph has a domatic 3-partition, you must find one.

There is no first order information about a graph which helps compute domatic partitions.

Proposition

The question of whether a computable graph has a domatic 3-partition is Σ^1_1-complete.

Basically this says that in order to determine whether a graph has a domatic 3-partition, you must find one.

The End

Thanks for listening