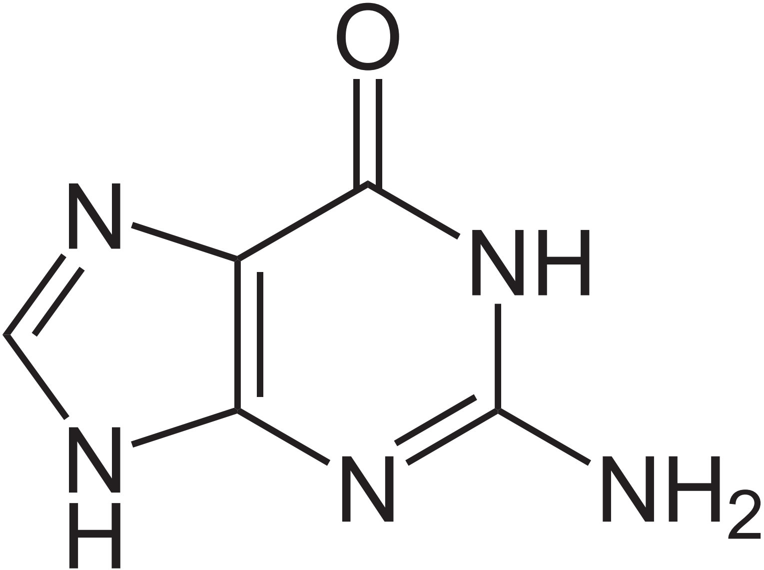
Combinatorial Generation in the Presence of Symmetry

Derrick Stolee

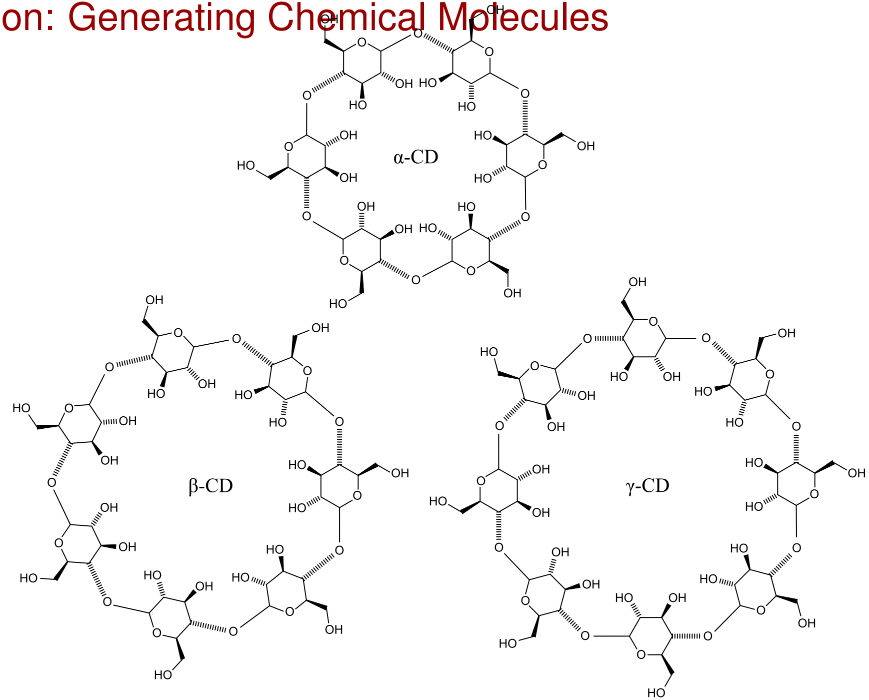
Iowa State University
dstolee@iastate.edu
http://www.math.iastate.edu/dstolee/

December 2, 2013
ISU MECS Seminar

Application: Generating Chemical Molecules

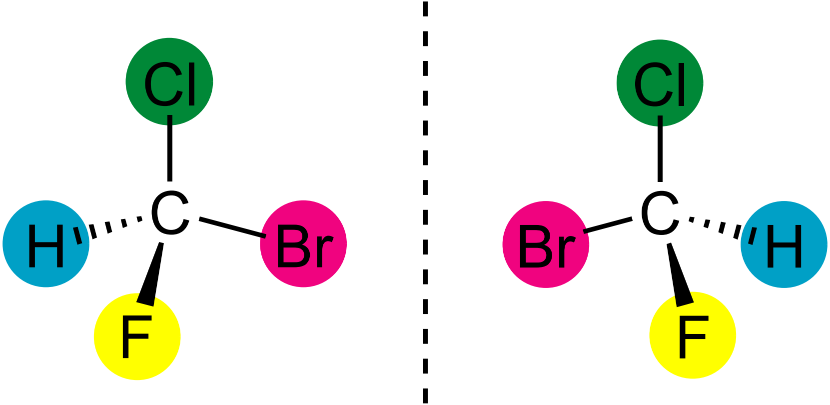


Application: Generating Chemical Molecules

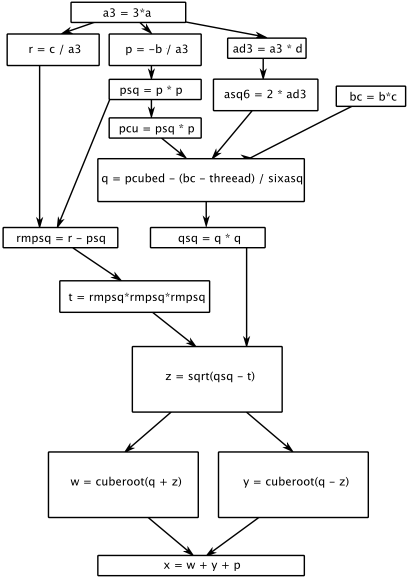


Application: Generating Chemical Molecules

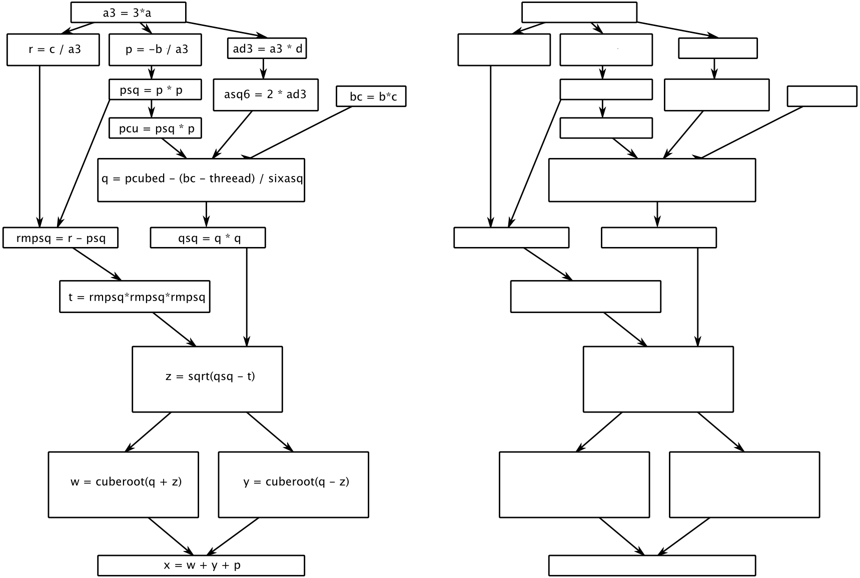
Chirality?



Application: Compiling Software



Application: Compiling Software



Exponential Behavior is Unavoidable

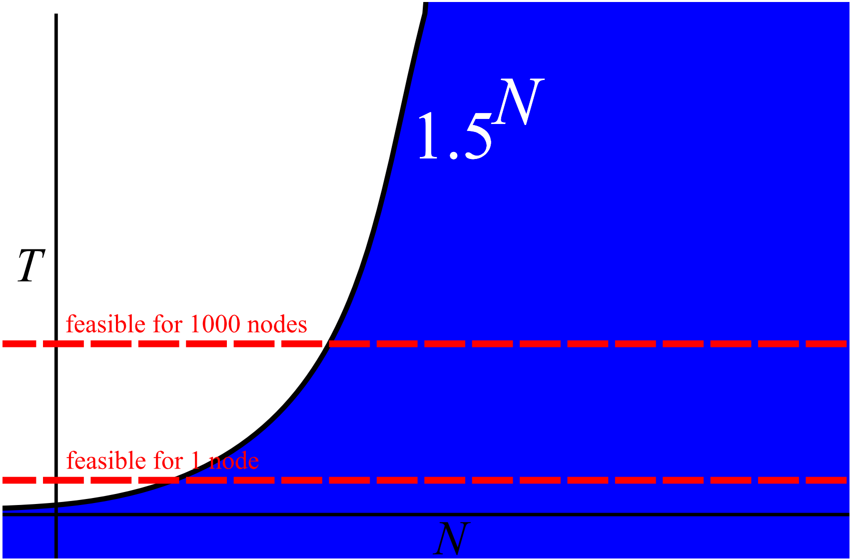
When dealing with NP-hard problems (or **worse!**), exponential behavior is unavoidable.

Exponential Behavior is Unavoidable

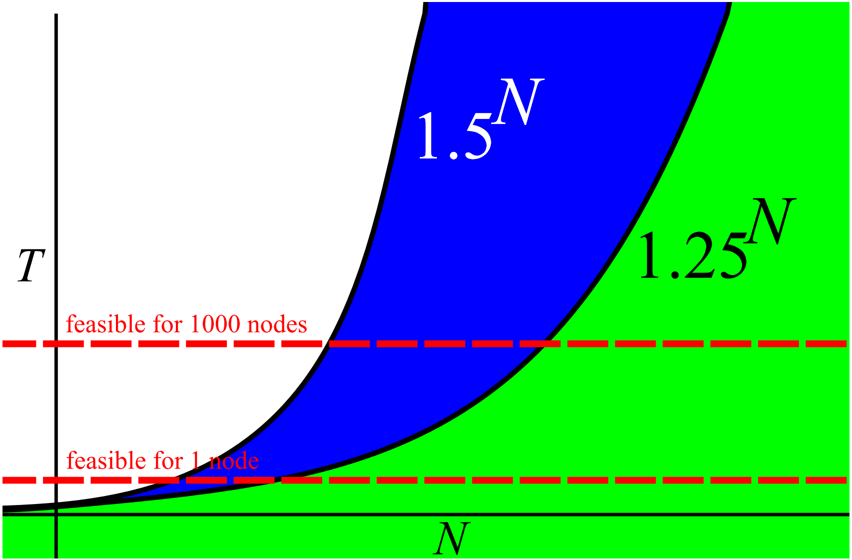
When dealing with NP-hard problems (or **worse!**), exponential behavior is unavoidable.

All we can do is **delay** or **diminish** that exponential behavior.

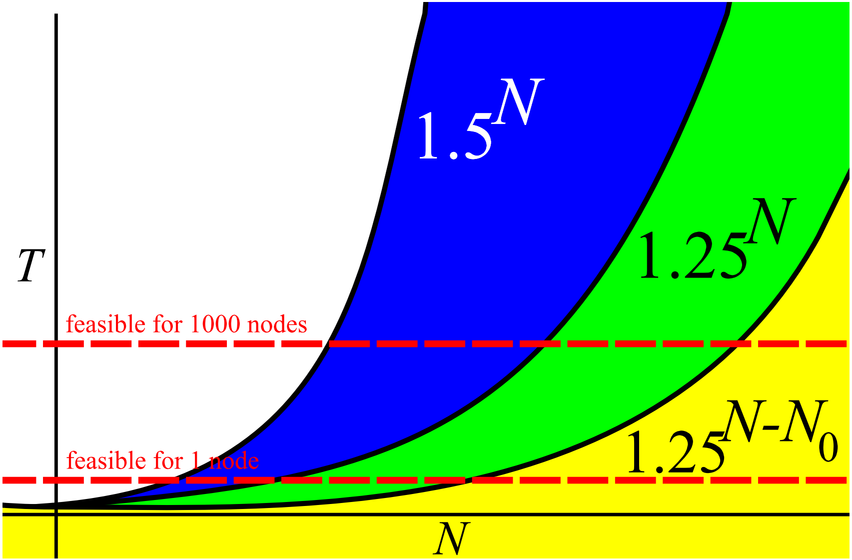
Shifting the Exponent



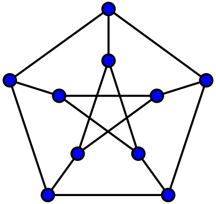
Shifting the Exponent

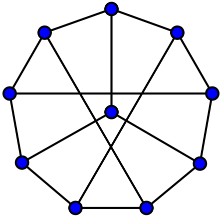


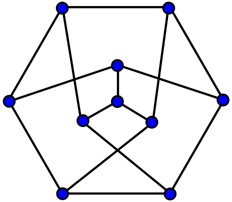
Shifting the Exponent



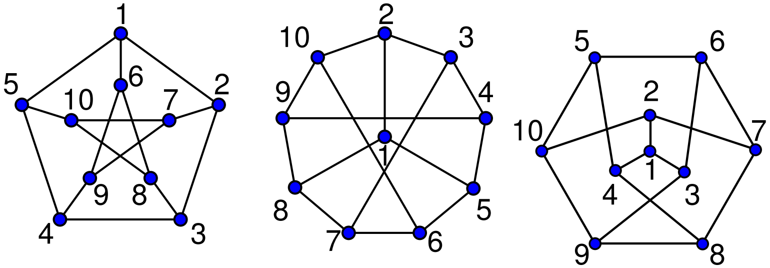
Graphs



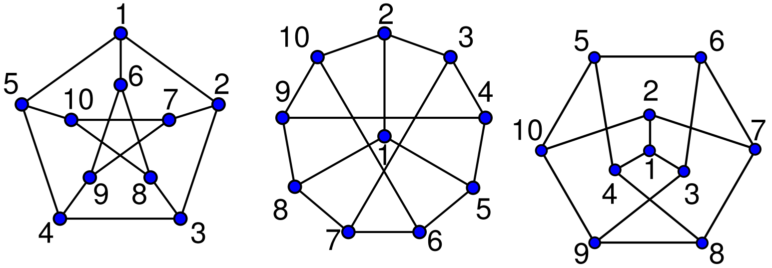




Graphs

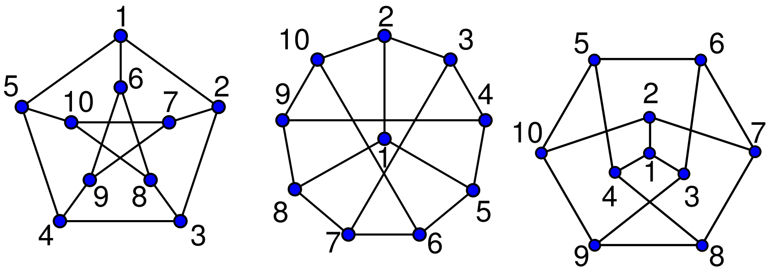


Graphs



An **isomorphism** between G_1 and G_2 is a bijection from $V(G_1)$ to $V(G_2)$ that induces a bijection from $E(G_1)$ to $E(G_2)$.

Graphs

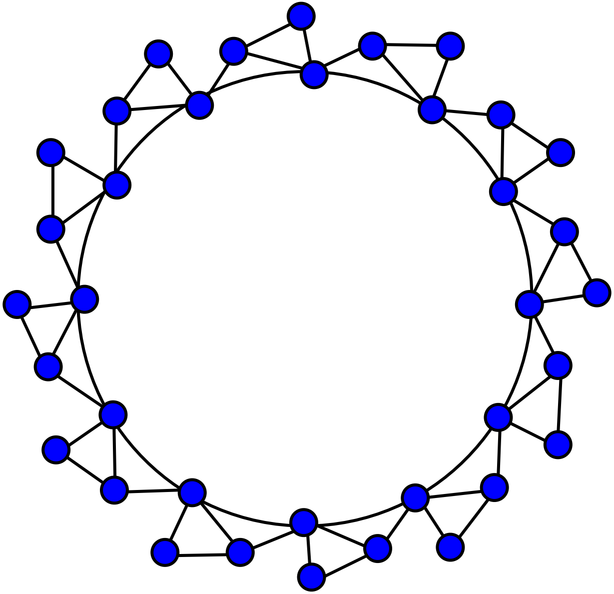


An **isomorphism** between G_1 and G_2 is a bijection from $V(G_1)$ to $V(G_2)$ that induces a bijection from $E(G_1)$ to $E(G_2)$.

An **automorphism** of G is a bijection from $V(G)$ to $V(G)$ that induces a bijection from $E(G)$ to $E(G)$.

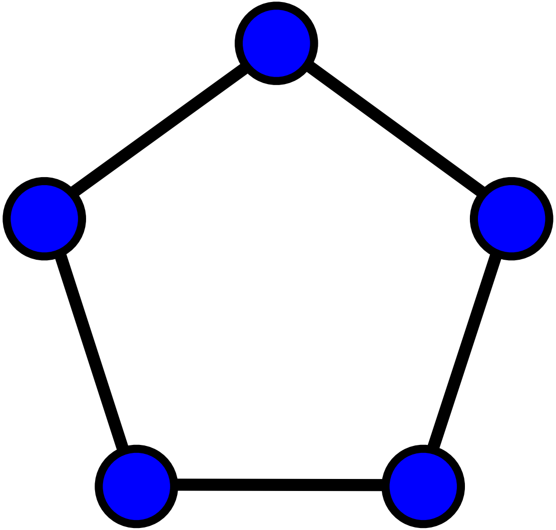
Graphs: Automorphisms

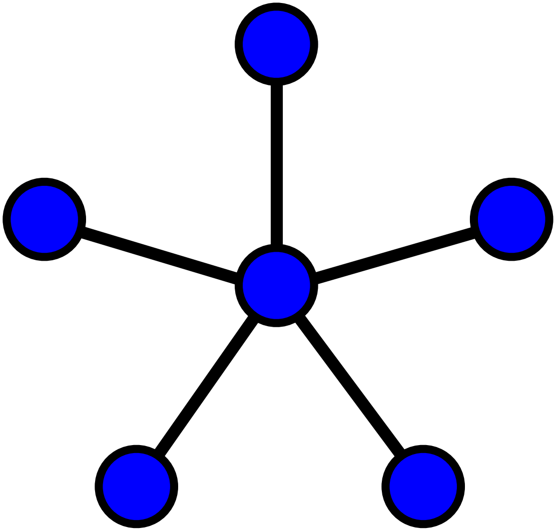
The set of **automorphisms** form a **group**.

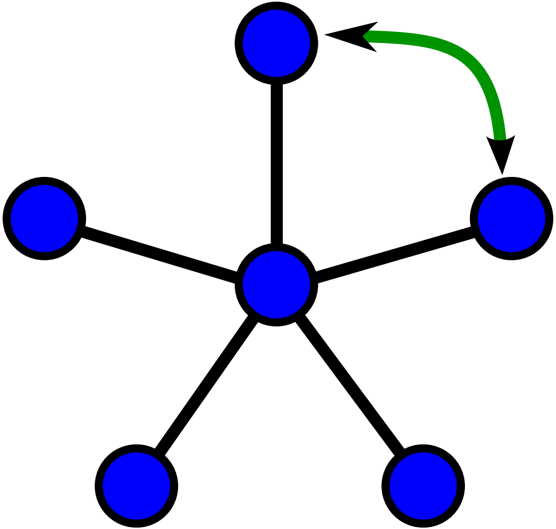


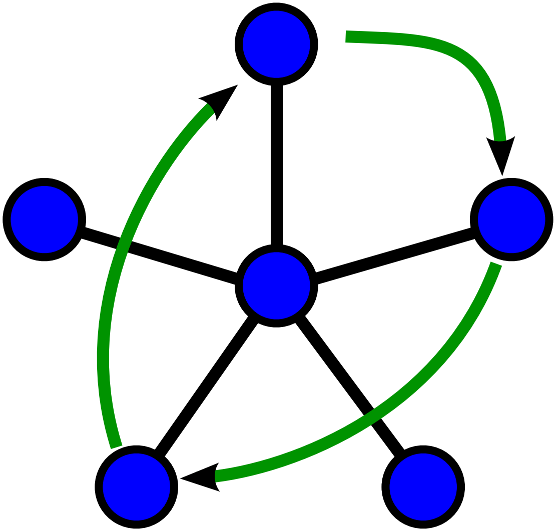
Graphs: Automorphisms

The set of **automorphisms** form a **group**.





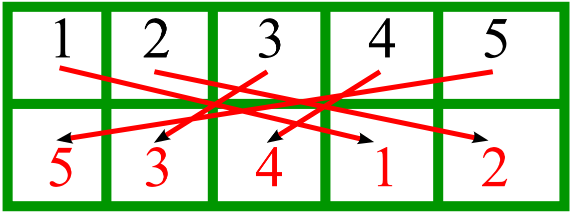




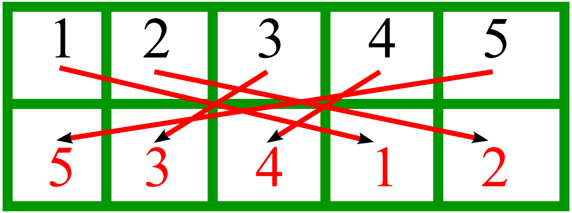
Permutations

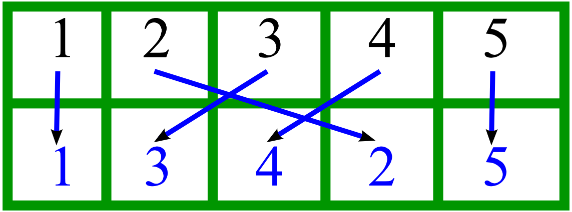
1	2	3	4	5
5	3	4	1	2

Permutations

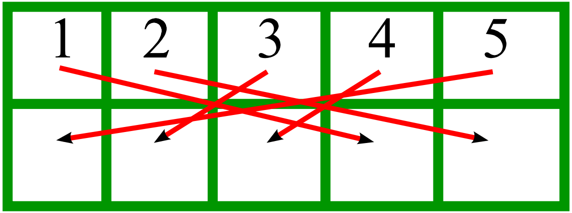


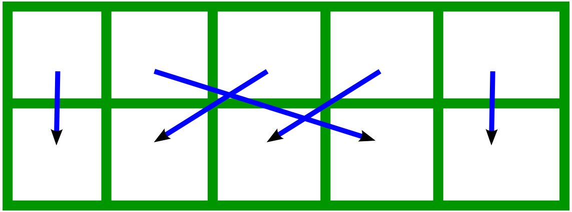
Permutations



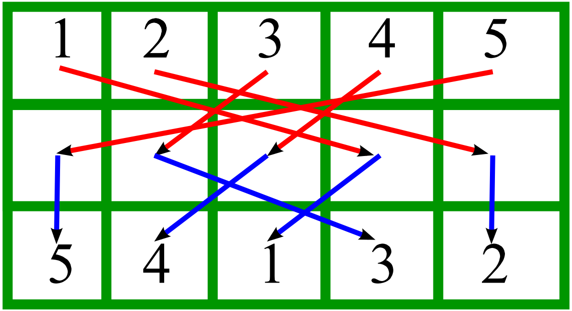


Permutations



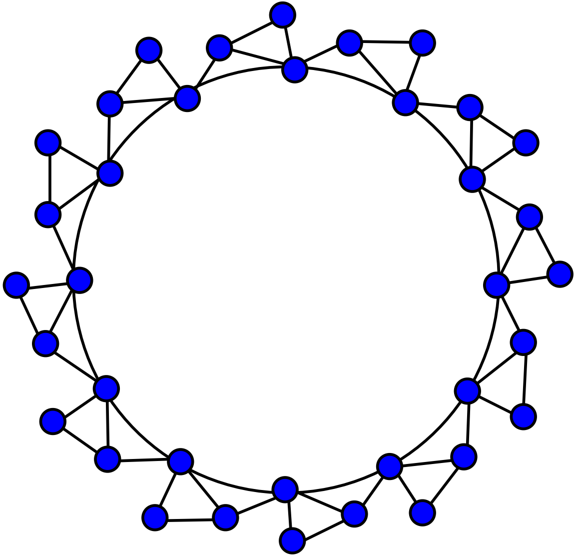


Permutations



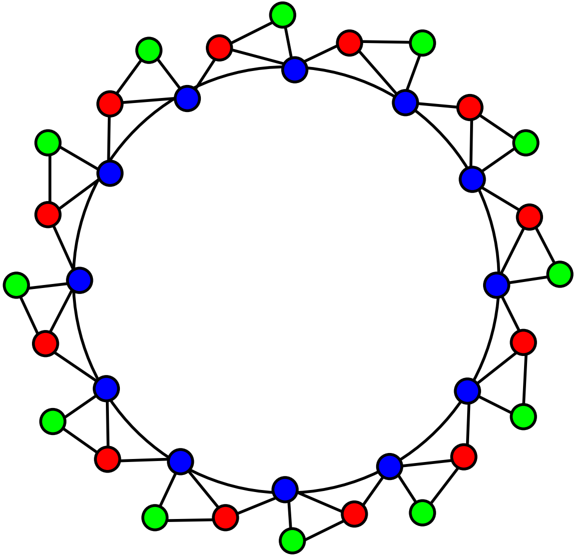
Graphs: Orbits

An **orbit** is a maximal set of objects such that every object is sent to every other object by some automorphism.



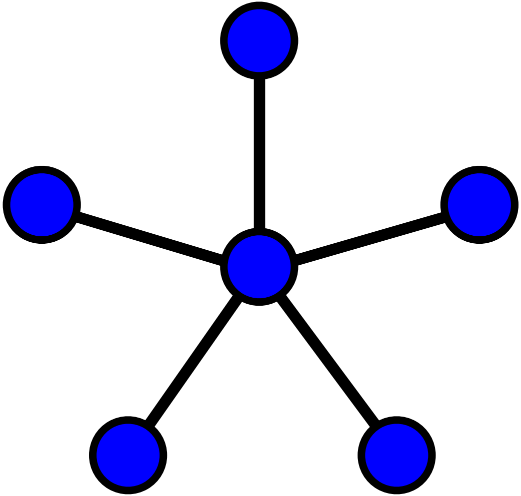
Graphs: Orbits

An **orbit** is a maximal set of objects such that every object is sent to every other object by some automorphism.



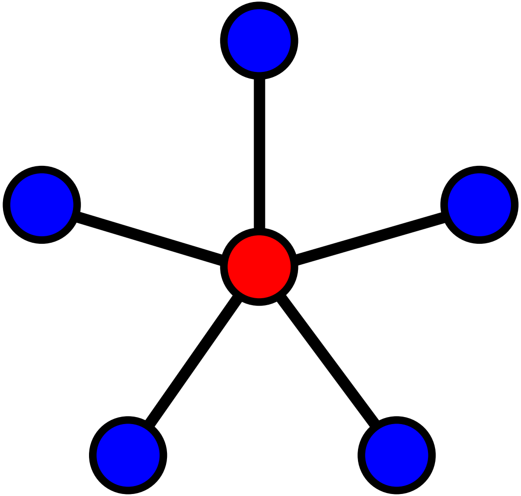
Graphs: Orbits

An **orbit** is a maximal set of objects such that every object is sent to every other object by some automorphism.



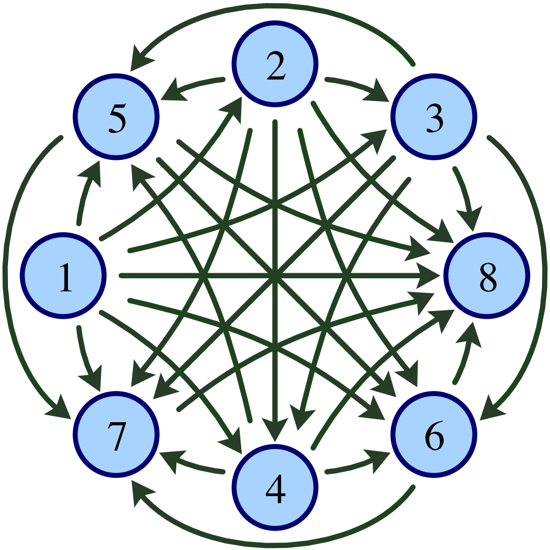
Graphs: Orbits

An **orbit** is a maximal set of objects such that every object is sent to every other object by some automorphism.



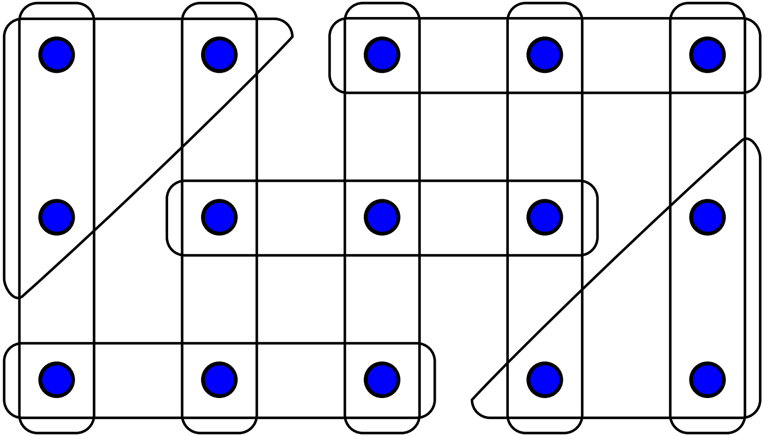
Other Objects

Directed Graphs



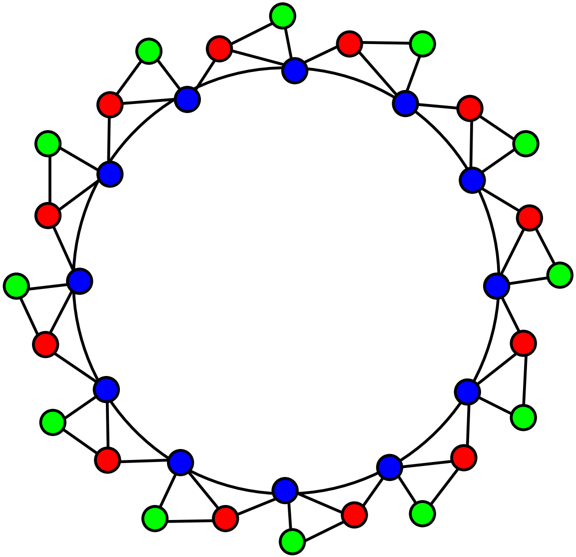
Other Objects

Hypergraphs



Other Objects

Colored (Partitioned) Graphs



Other Objects

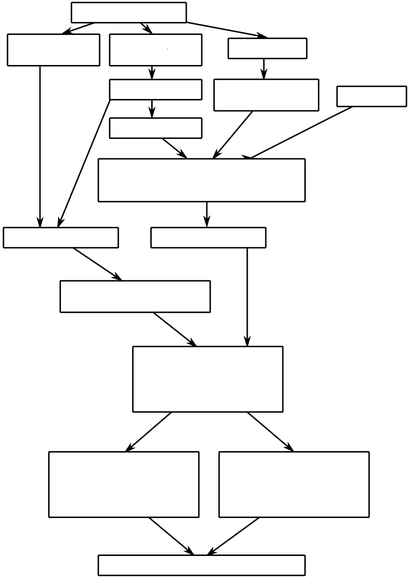
Latin Squares

A	B	P	C	O	D	N	E	M	F	L	G	K	H	J	I
B	C	A	D	P	E	O	F	N	G	M	H	L	I	K	J
C	D	B	E	A	F	P	G	O	H	N	I	M	J	L	K
D	E	C	F	B	G	A	H	P	I	O	J	N	K	M	L
E	F	D	G	C	H	B	I	A	J	P	K	O	L	N	M
F	G	E	H	D	I	C	J	B	K	A	L	P	M	O	N
G	H	F	I	E	J	D	K	C	L	B	M	A	N	P	O
H	I	G	J	F	K	E	L	D	M	C	N	B	O	A	P
I	J	H	K	G	L	F	M	E	N	D	O	C	P	B	A
J	K	I	L	H	M	G	N	F	O	E	P	D	A	C	B
K	L	J	M	I	N	H	O	G	P	F	A	E	B	D	C
L	M	K	N	J	O	I	P	H	A	G	B	F	C	E	D
M	N	L	O	K	P	J	A	I	B	H	C	G	D	F	E
N	O	M	P	L	A	K	B	J	C	I	D	H	E	G	F
O	P	N	A	M	B	L	C	K	D	J	E	I	F	H	G
P	A	O	B	N	C	M	D	L	E	K	F	J	G	I	H

This 16×16 *latin square assists in the construction of a Williams Design.*

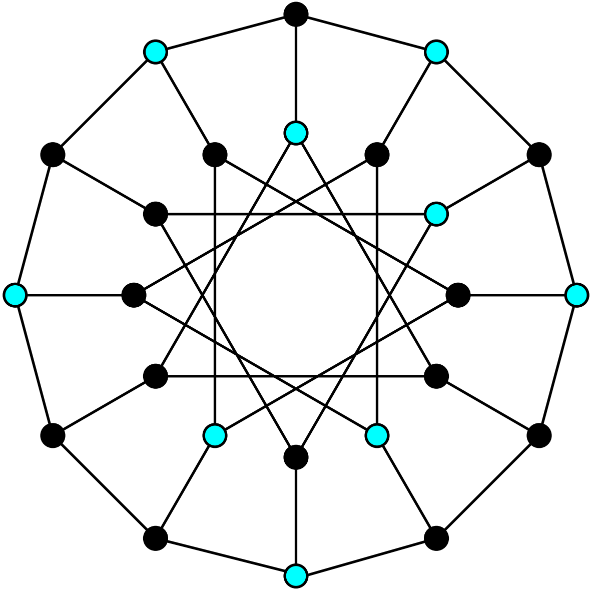
Other Objects

Partially-Ordered Sets



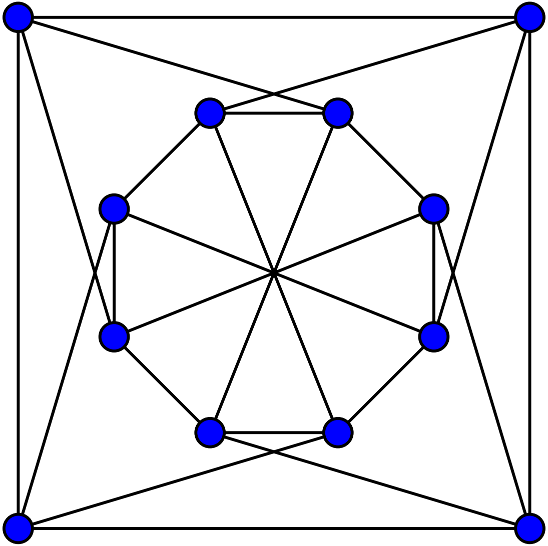
Subobjects

Independent Sets



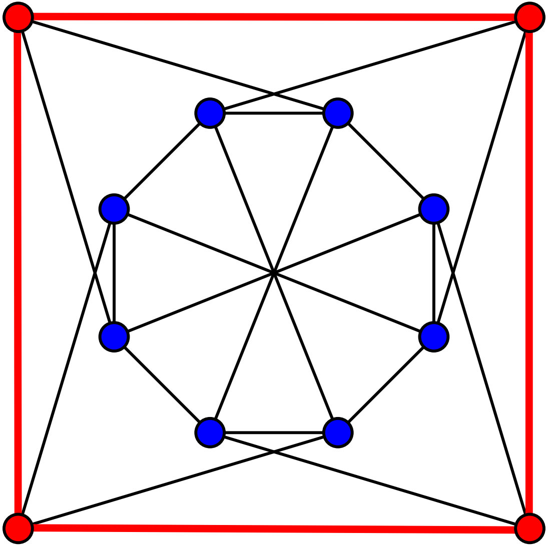
Subobjects

(Induced) Subgraphs



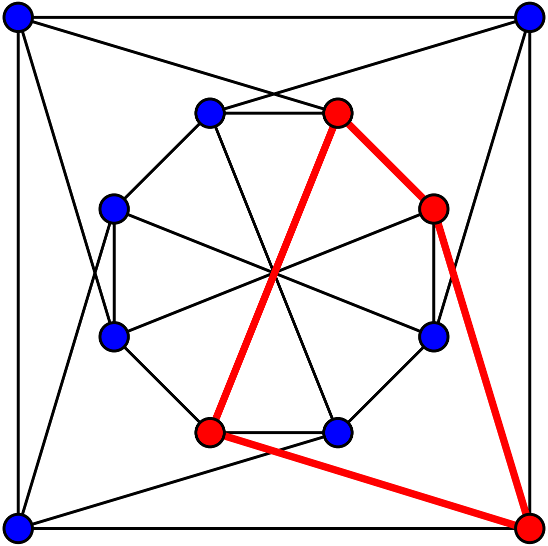
Subobjects

(Induced) Subgraphs



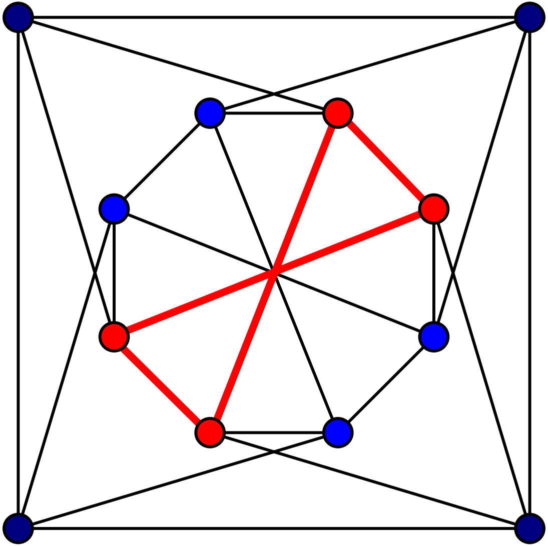
Subobjects

(Induced) Subgraphs



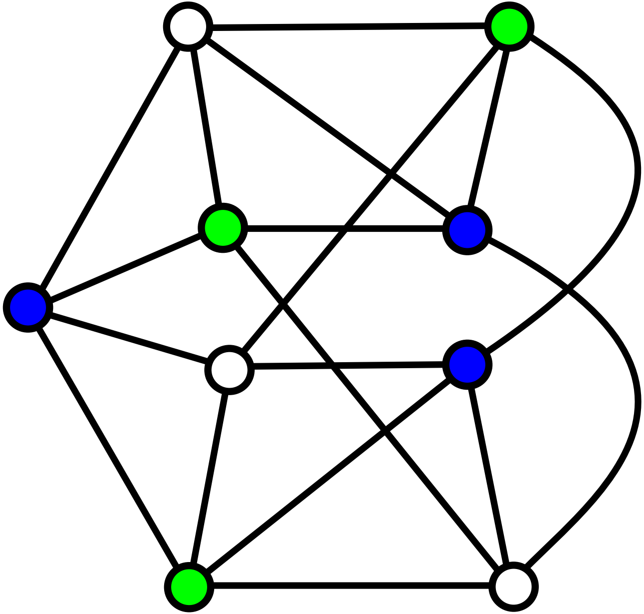
Subobjects

(Induced) Subgraphs



Subobjects

(Proper) Colorings



Generation Algorithms

Goal: Generate all *unlabeled* objects that satisfy the constraints.

Symmetry Breaking

1. Reduces isomorphic duplicates.
2. Does not allow for dynamic symmetry updates.
3. Removes symmetry, then uses standard symmetry-unaware algorithms.

Orbital Branching

1. Reduces isomorphic duplicates.
2. Allows for dynamic symmetry updates.
3. Branching method can be customized to the given problem.
4. Integrates well with branch-and-bound methods and constraint propagation.

(Ostrowski talked about this, also my CS Colloquium)

Canonical Deletion

1. Eliminates isomorphic duplicates*.
2. Allows for dynamic symmetry updates.
3. Augmentation method can be customized to the given problem.
4. Does not integrate well with branch-and-bound methods or constraint propagation.

Brendan McKay, Isomorph-free exhaustive generation, *J. of Algorithms* (1997).

Canonical Deletion

1. Build objects piece-by-piece.
2. Define a *canonical construction path* to every unlabeled object.
3. Only follow paths that agree with the canonical construction path.

Example: Generating Graphs by Vertex Additions

Let's generate all graphs of order n by adding vertices one-by-one.

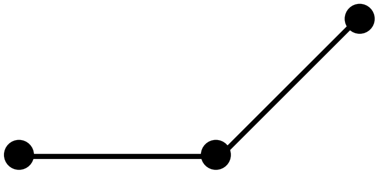
Augmentation: Add a vertex adjacent to a set $S \subset V(G)$.

Deletion: Select a vertex $v \in V(G)$ to delete, $G' = G - v$.

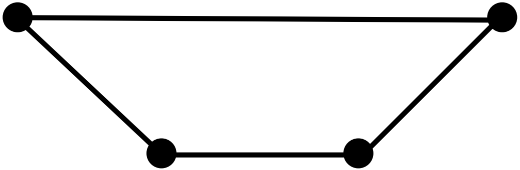
Generating with Vertex Augmentations

Generating with Vertex Augmentations

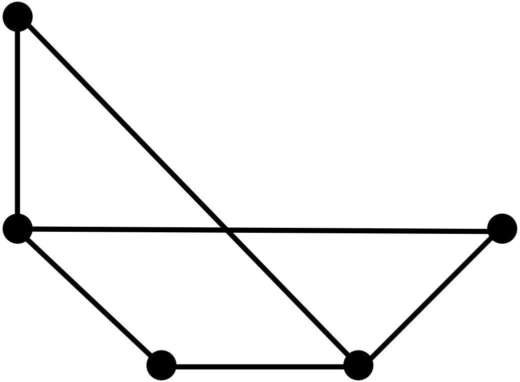
Generating with Vertex Augmentations



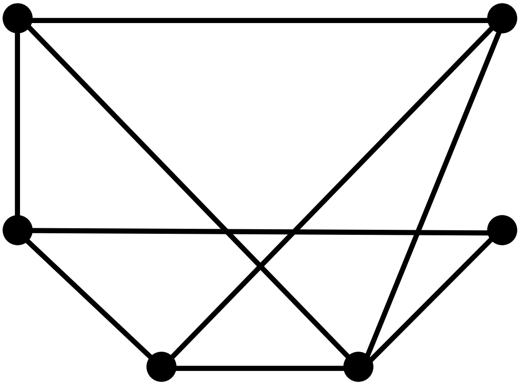
Generating with Vertex Augmentations



Generating with Vertex Augmentations



Generating with Vertex Augmentations



Example: Generating Graphs by Vertex Additions

Let's generate all graphs of order n by adding vertices one-by-one.

Augmentation: Add a vertex adjacent to a set $S \subset V(G)$.

IMPORTANT: Only one augmentation per orbit!

Deletion: Select a vertex $v \in V(G)$ to delete, $G' = G - v$.

Canonical Labeling

A **canonical labeling** takes a labeled graph G

Canonical Labeling

A **canonical labeling** takes a labeled graph G and applies labels $\sigma_G(v)$ to each $v \in V(G)$

Canonical Labeling

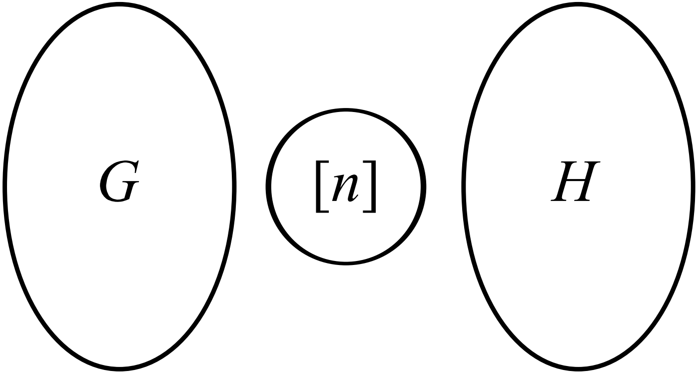
A **canonical labeling** takes a labeled graph G and applies labels $\sigma_G(v)$ to each $v \in V(G)$ so that any $H \cong G$ with labels $\sigma_H(v)$

Canonical Labeling

A **canonical labeling** takes a labeled graph G and applies labels $\sigma_G(v)$ to each $v \in V(G)$ so that any $H \cong G$ with labels $\sigma_H(v)$ has an isomorphism $\sigma_H^{-1}(\sigma_G(v))$ from G to H.

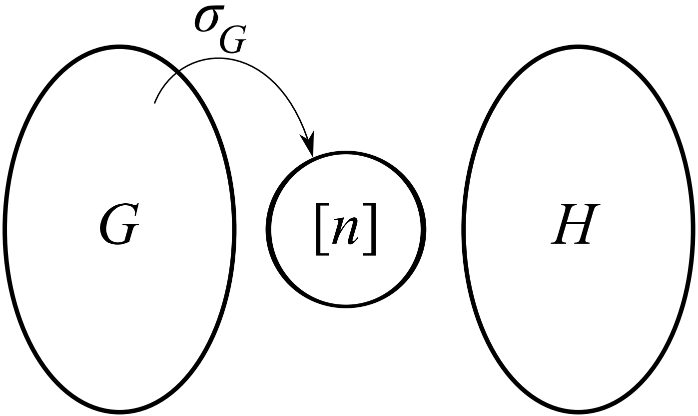
Canonical Labeling

A **canonical labeling** takes a labeled graph G and applies labels $\sigma_G(v)$ to each $v \in V(G)$ so that any $H \cong G$ with labels $\sigma_H(v)$ has an isomorphism $\sigma_H^{-1}(\sigma_G(v))$ from G to H.



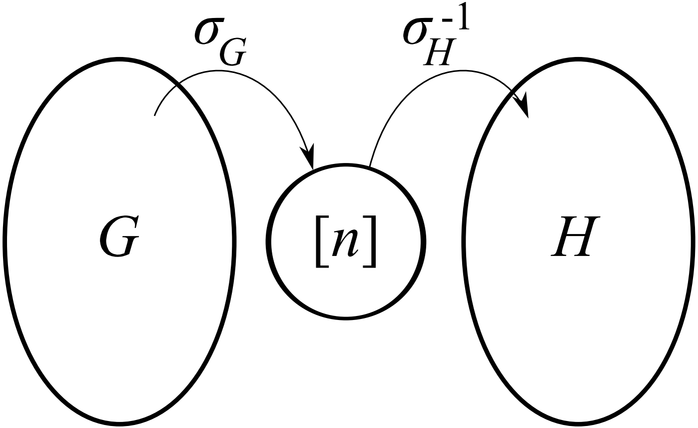
Canonical Labeling

A **canonical labeling** takes a labeled graph G and applies labels $\sigma_G(v)$ to each $v \in V(G)$ so that any $H \cong G$ with labels $\sigma_H(v)$ has an isomorphism $\sigma_H^{-1}(\sigma_G(v))$ from G to H.



Canonical Labeling

A **canonical labeling** takes a labeled graph G and applies labels $\sigma_G(v)$ to each $v \in V(G)$ so that any $H \cong G$ with labels $\sigma_H(v)$ has an isomorphism $\sigma_H^{-1}(\sigma_G(v))$ from G to H.



Canonical labels can be computed by McKay's `nauty` software.

Canonical Deletion by Filtering

Let $S = V(G)$. Filter S until $|S| = 1$ by the following conditions:

Canonical Deletion by Filtering

Let $S = V(G)$. Filter S until $|S| = 1$ by the following conditions:

1. Remove cut vertices from S.

Canonical Deletion by Filtering

Let $S = V(G)$. Filter S until $|S| = 1$ by the following conditions:

1. Remove cut vertices from S.
2. Let $d = \min\{\deg(v) : v \in S\}$. Set

$$S \leftarrow \{v \in S : \deg(v) = d\}.$$

Canonical Deletion by Filtering

Let $S = V(G)$. Filter S until $|S| = 1$ by the following conditions:

1. Remove cut vertices from S.
2. Let $d = \min\{\deg(v) : v \in S\}$. Set
$$S \leftarrow \{v \in S : \deg(v) = d\}.$$
3. (Include other, more complicated invariants here.)

Canonical Deletion by Filtering

Let $S = V(G)$. Filter S until $|S| = 1$ by the following conditions:

1. Remove cut vertices from S.
2. Let $d = \min\{\deg(v) : v \in S\}$. Set
$$S \leftarrow \{v \in S : \deg(v) = d\}.$$
3. (Include other, more complicated invariants here.)
4. Compute a canonical labeling ℓ, and set
$$v = \operatorname{argmin}_{v \in S} \ell(v).$$

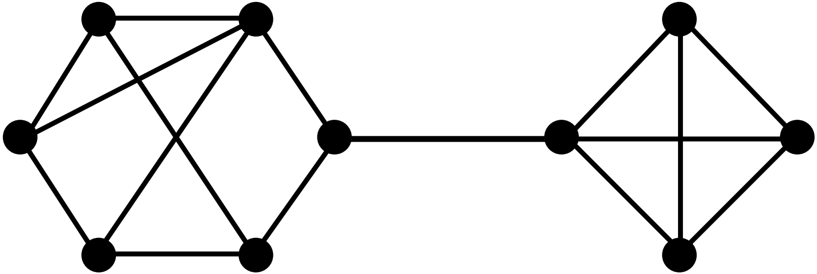
Canonical Deletion by Filtering

Let $S = V(G)$. Filter S until $|S| = 1$ by the following conditions:

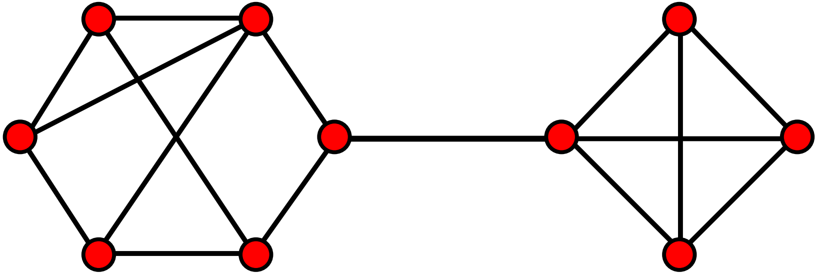
1. Remove cut vertices from S.
2. Let $d = \min\{\deg(v) : v \in S\}$. Set
$$S \leftarrow \{v \in S : \deg(v) = d\}.$$
3. (Include other, more complicated invariants here.)
4. Compute a canonical labeling ℓ, and set
$$v = \operatorname{argmin}_{v \in S} \ell(v).$$

The vertex v is the canonical deletion.

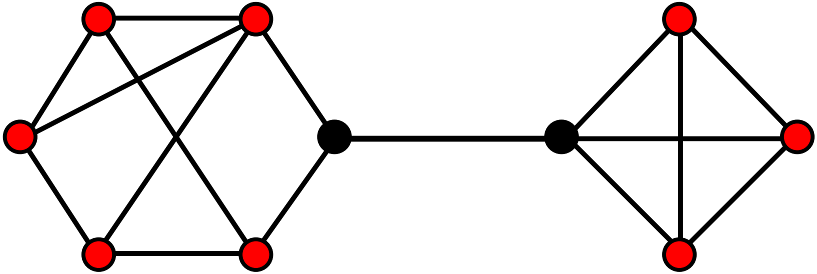
Canonical Vertex Deletions



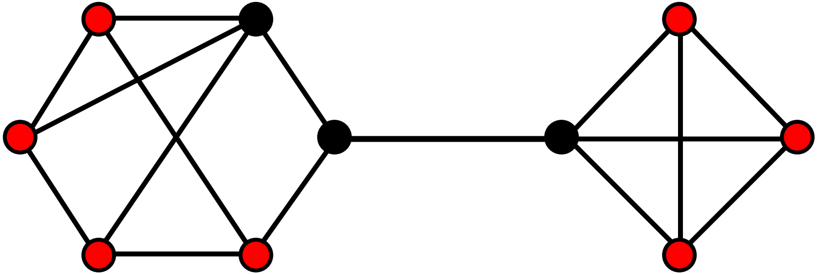
Canonical Vertex Deletions



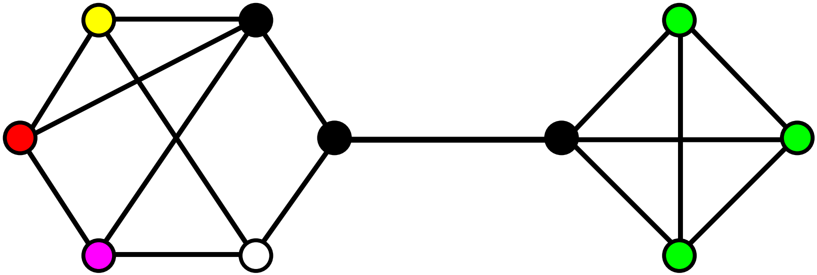
Canonical Vertex Deletions



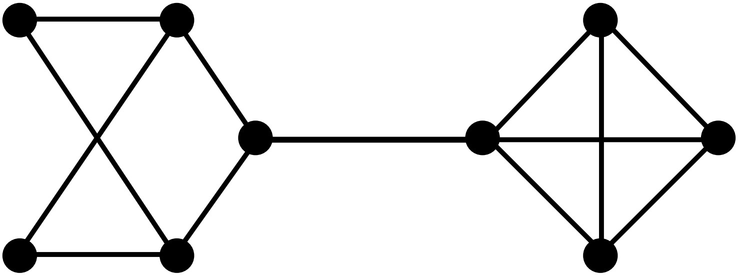
Canonical Vertex Deletions



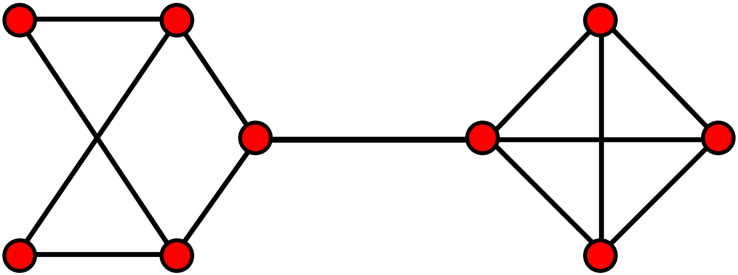
Canonical Vertex Deletions



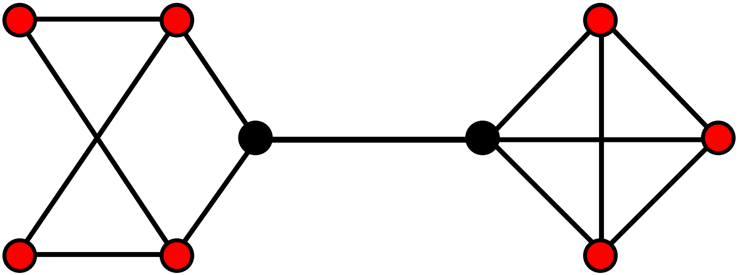
Canonical Vertex Deletions



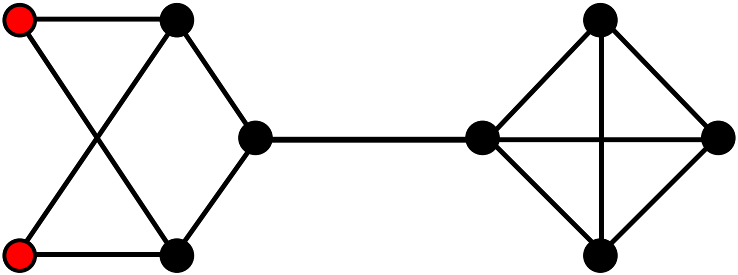
Canonical Vertex Deletions



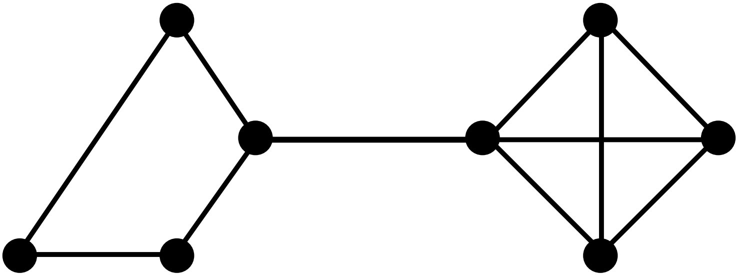
Canonical Vertex Deletions



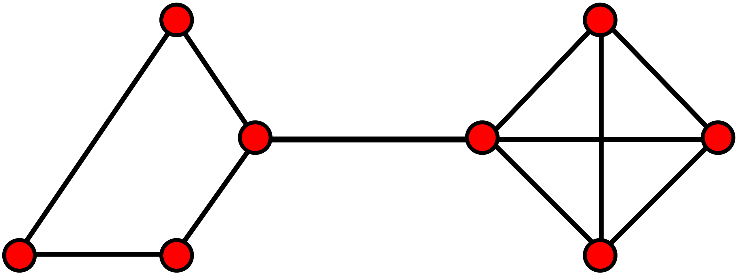
Canonical Vertex Deletions



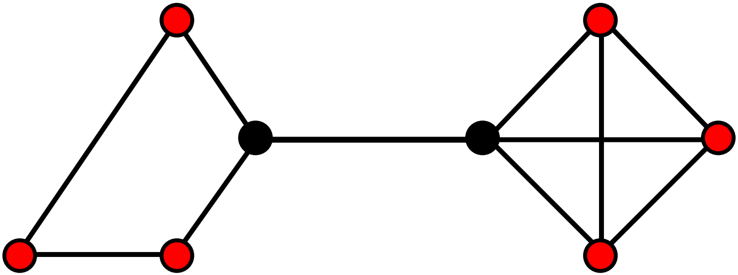
Canonical Vertex Deletions



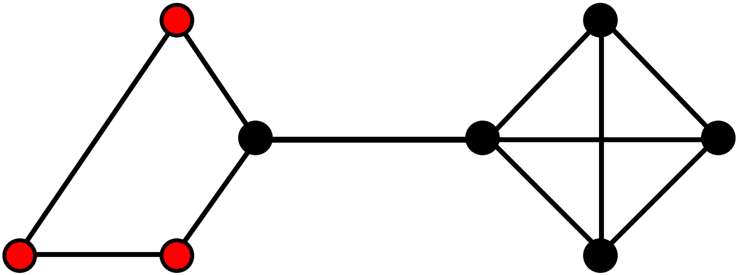
Canonical Vertex Deletions



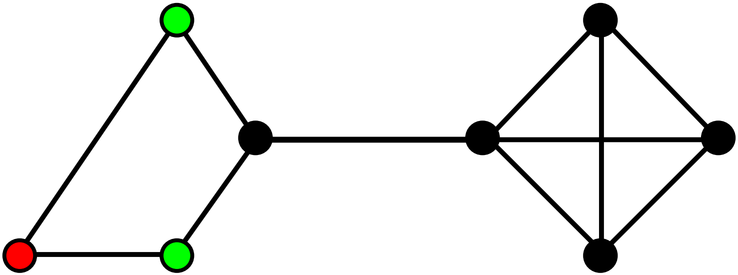
Canonical Vertex Deletions



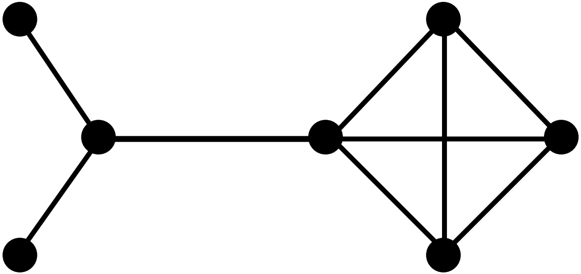
Canonical Vertex Deletions



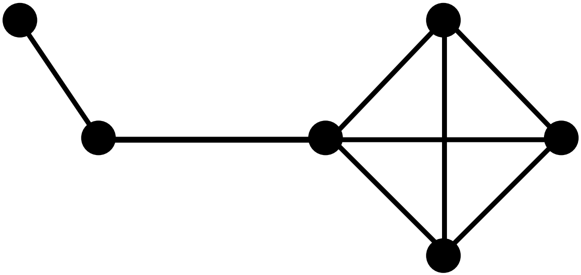
Canonical Vertex Deletions



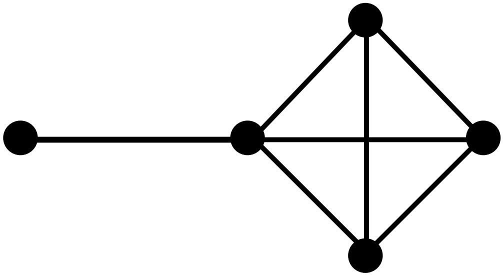
Canonical Vertex Deletions



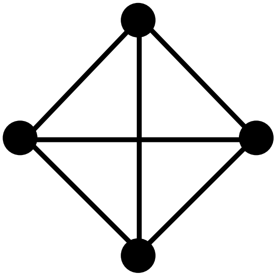
Canonical Vertex Deletions



Canonical Vertex Deletions



Canonical Vertex Deletions



Using Deletion To Minimize Augmentations

By thinking of our filtering mechanism for the canonical deletion, we can avoid making augmentations that will not be canonical deletions:

Using Deletion To Minimize Augmentations

By thinking of our filtering mechanism for the canonical deletion, we can avoid making augmentations that will not be canonical deletions:

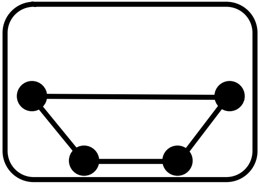
1. If minimizing degree, do not add anything of degree more than $\delta(G) + 1$.

Using Deletion To Minimize Augmentations

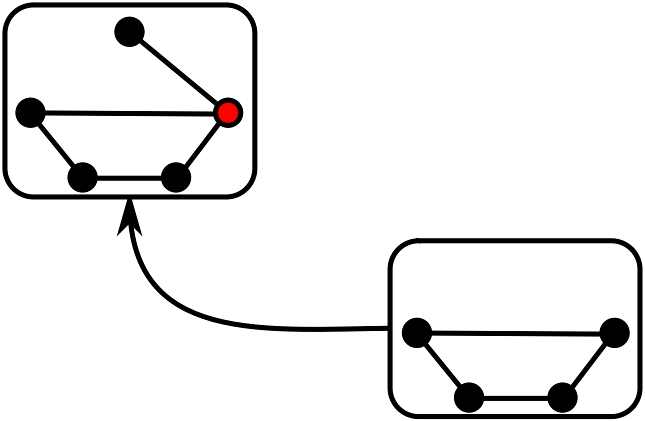
By thinking of our filtering mechanism for the canonical deletion, we can avoid making augmentations that will not be canonical deletions:

1. If minimizing degree, do not add anything of degree more than $\delta(G) + 1$.
2. If not deleting cut-vertices, everything has degree at least one.

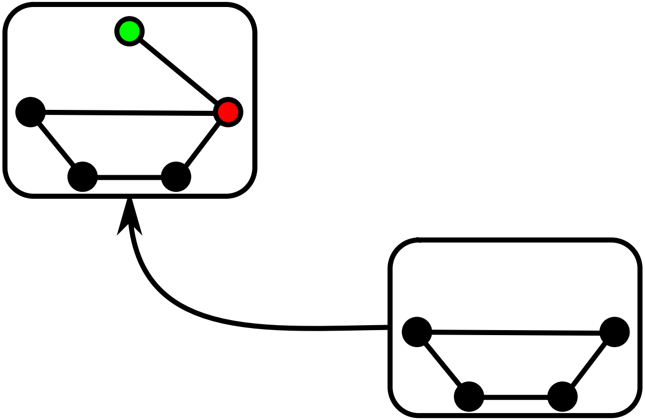
Canonical Vertex Deletions



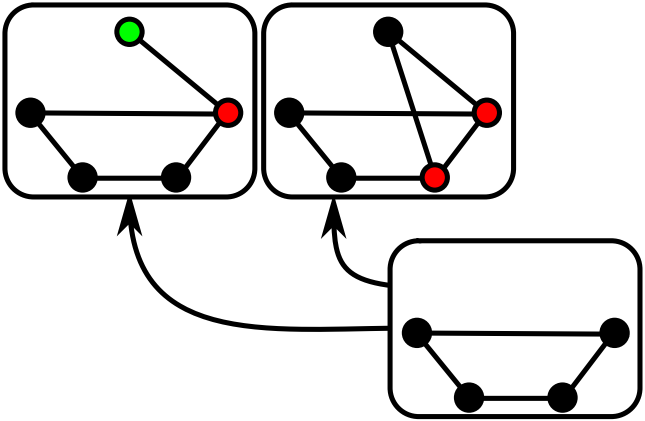
Canonical Vertex Deletions



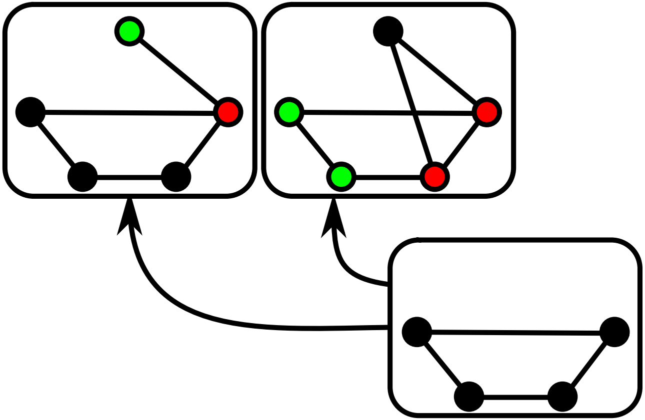
Canonical Vertex Deletions



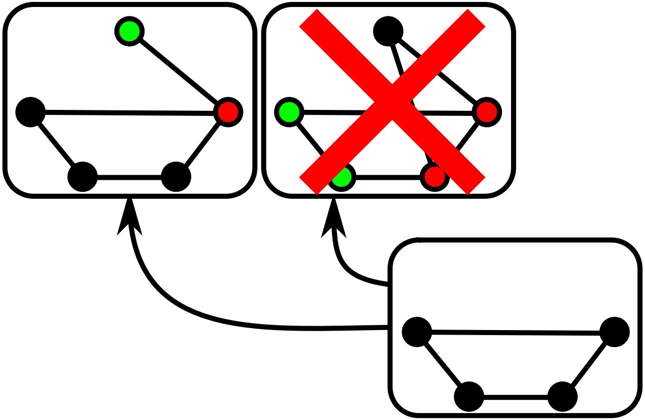
Canonical Vertex Deletions



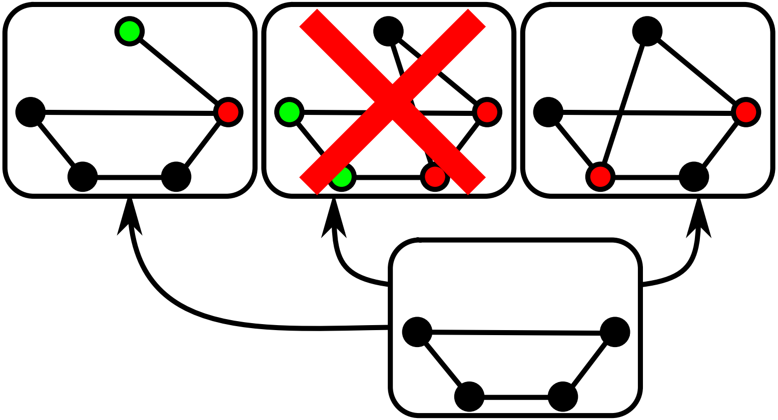
Canonical Vertex Deletions



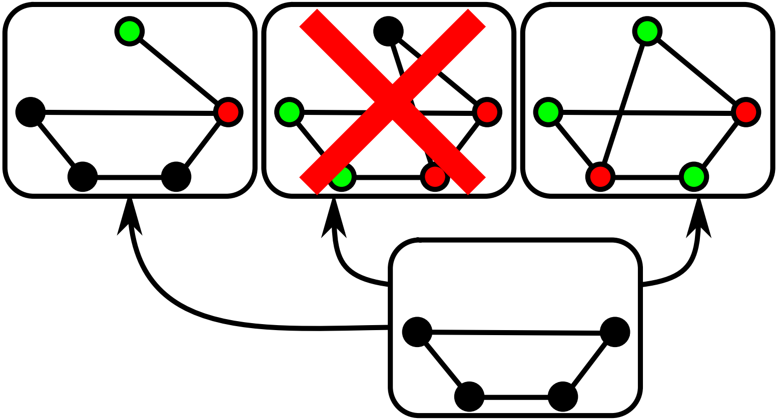
Canonical Vertex Deletions



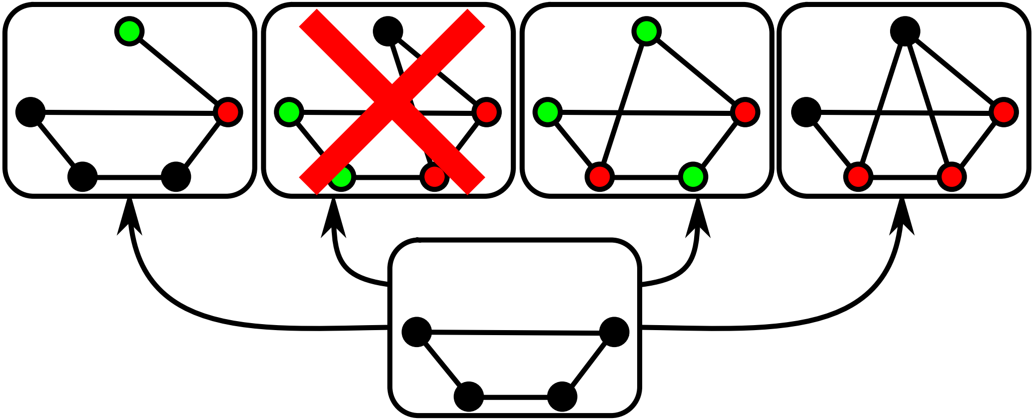
Canonical Vertex Deletions



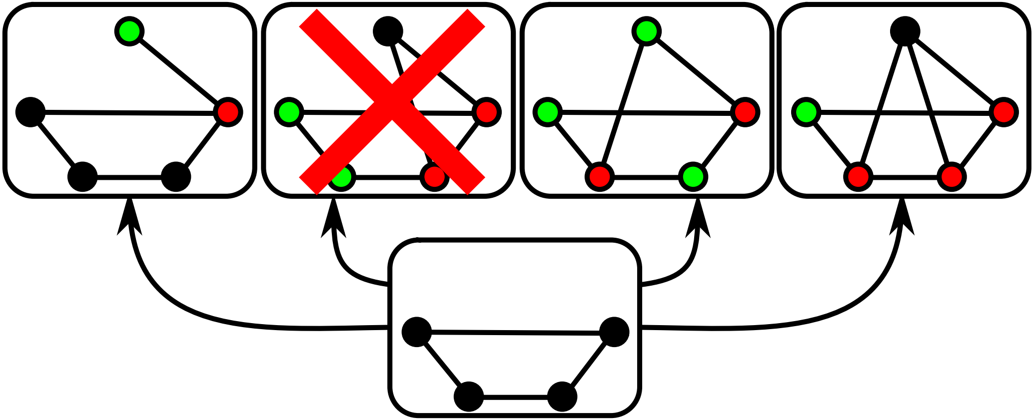
Canonical Vertex Deletions



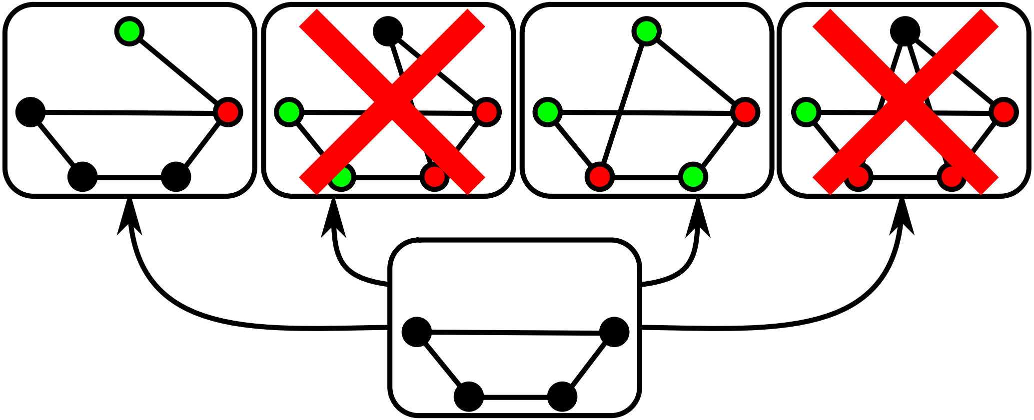
Canonical Vertex Deletions



Canonical Vertex Deletions



Canonical Vertex Deletions



Effectiveness of Canonical Deletion

Every unlabeled object is **expanded** exactly once.

Effectiveness of Canonical Deletion

Every unlabeled object is **expanded** exactly once.

Every unlabeled object is **reached** at most *once per possible deletion*.

Effectiveness of Canonical Deletion

Every unlabeled graph is **expanded** exactly once.

Every unlabeled graph is **reached** at most *n times*.

Effectiveness of Canonical Deletion

Every unlabeled graph is **expanded** exactly once.

Every unlabeled graph is **reached** at most *n times*.

Most unlabeled graphs have $n!$ different labelings.

Effectiveness of Canonical Deletion

Every unlabeled graph is **expanded** exactly once.

Every unlabeled graph is **reached** at most *n times*.

Most unlabeled graphs have $n!$ different labelings.

So the resulting computation time is about

$$\sum_{n=1}^{N} 2^{\binom{n}{2}} \cdot \frac{nf(n)}{n!} \approx 2^{N^2 - N \log N}$$

where $f(n)$ is the average time to compute canonical labels and automorphisms.

n	Labeled graphs of order n
6	32,768
7	2,097,152
8	268,435,456
9	68,719,476,736
10	35,184,372,088,832
11	36,028,797,018,963,968
12	73,786,976,294,838,206,464
13	302,231,454,903,657,293,676,544
14	2,475,880,078,570,760,549,798,248,448
15	40,564,819,207,303,340,847,894,502,572,032

$$2^{\binom{n}{2}} \approx 2^{\theta(n^2)}$$

n	Unlabeled connected graphs of order n
6	85
7	509
8	4,060
9	41,301
10	510,489
11	7,319,447
12	117,940,535
13	2,094,480,864
14	40,497,138,011
15	845,480,228,069

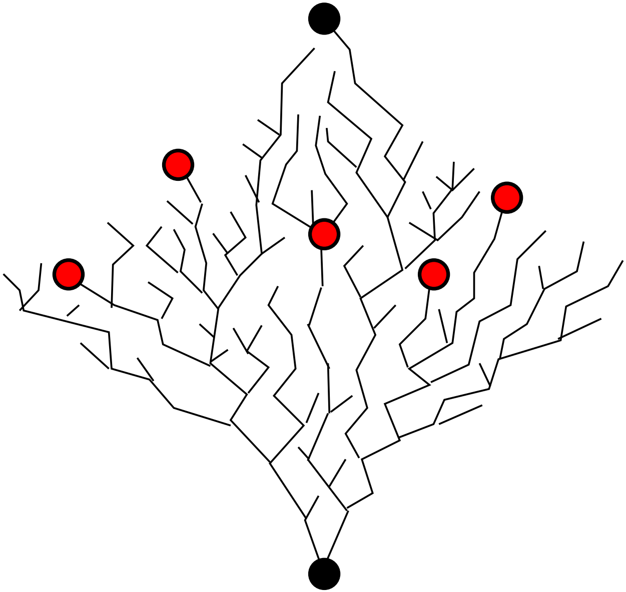
OEIS Sequence A002851 Grows $2^{\Omega(n^2)}$.

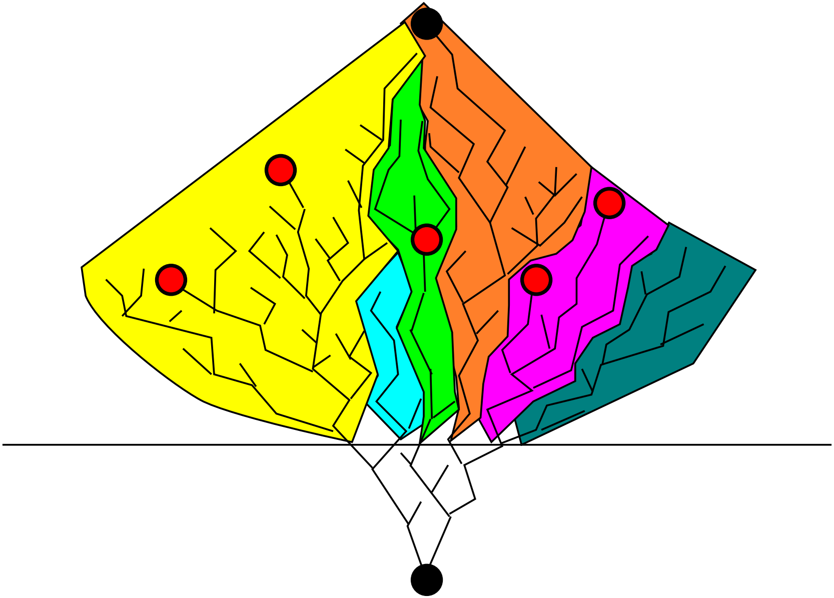
n	Unlabeled connected graphs of order n
6	85
7	509
8	4,060
9	41,301
10	510,489
11	7,319,447
12	**117,940,535**
13	2,094,480,864
14	40,497,138,011
15	845,480,228,069

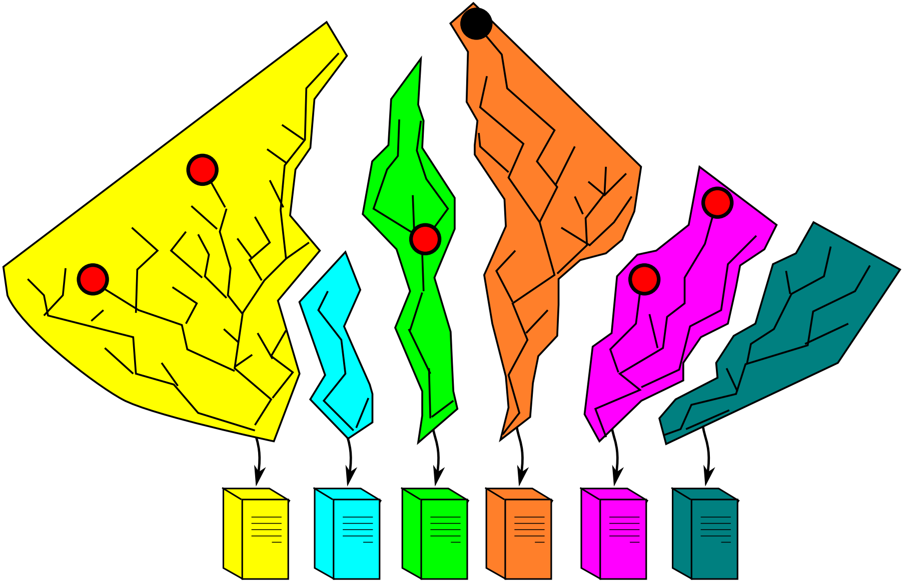
Requires about **1 day** of CPU Time.

n	Unlabeled connected graphs of order n
6	85
7	509
8	4,060
9	41,301
10	510,489
11	7,319,447
12	117,940,535
13	2,094,480,864
14	**40,497,138,011**
15	845,480,228,069

Requires over **1 year** of CPU Time.







Implementation

My **TreeSearch** library enables parallelization in the Condor scheduler.

Executes on the **Open Science Grid**, a collection of supercomputers around the country.

Open Science Grid

Research Problem

Q: How can we integrate **constraint propagation** with canonical deletion?

Next Week: Generating Graphs with p Perfect Matchings

Let $f(n, p)$ be the **maximum number of edges** in a graph of order n with **exactly p perfect matchings**.

Next Week: Generating Graphs with p Perfect Matchings

Let $f(n, p)$ be the **maximum number of edges** in a graph of order n with **exactly p perfect matchings**.

We determine this value and characterize all graphs achieving this bound for all n (for small p).

Next Week: Generating Graphs with p Perfect Matchings

Let $f(n,p)$ be the **maximum number of edges** in a graph of order n with **exactly p perfect matchings**.

We determine this value and characterize all graphs achieving this bound for all n (for small p).

Requires building a canonical deletion that has the number of perfect matchings be **monotonic!**

To learn more...

- B. D. McKay. Isomorph-free exhaustive generation.
- B. D. McKay. Small graphs are reconstructible.
- F. Margot. Pruning by isomorphism in branch-and-cut.
- B. D. McKay, A. Meynert. Small latin squares, quasigroups, and loops.
- G. Brinkmann, B. D. McKay. Posets on up to 16 points.
- P. Kaski, P. R. J. Östergard. The Steiner triple systems of order 19.
- D. Stolee. Isomorph-free generation of 2-connected graphs with applications.
- D. Stolee. Generating p-extremal graphs.

Combinatorial Generation in the Presence of Symmetry

Derrick Stolee

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

December 2, 2013
ISU MECS Seminar