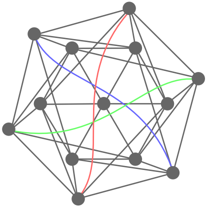
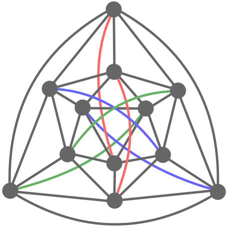
Computational Combinatorics and the search for Uniquely K_r-Saturated Graphs

Stephen G. Hartke

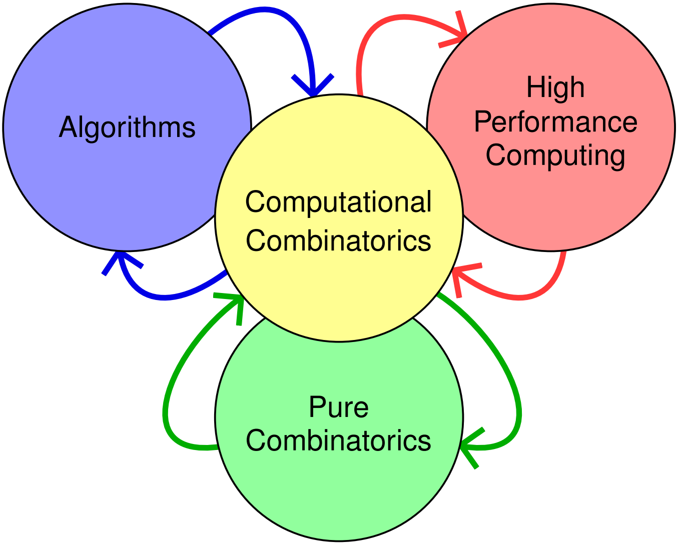
University of Nebraska–Lincoln, USA
`hartke@math.unl.edu`
`http://www.math.unl.edu/~shartke2/`





Joint work with Derrick Stolee, Iowa State University.

What is Computational Combinatorics?



What is Computational Combinatorics?

Using a combination of

- **pure mathematics**,
- **algorithms**, and
- **computational resources**

to solve problems in pure combinatorics by

- **providing evidence** for conjectures,
- finding **examples** and **counterexamples**, and
- **discovering and proving theorems**.

The Goal

Determine if certain **combinatorial objects** exist with given **structural** or **extremal** properties.

The Goal

Determine if certain **combinatorial objects** exist with given **structural** or **extremal** properties.

Examples:

1. Is there a **projective plane** of order 10?
2. When do **strongly regular graphs** exist?
3. How many **Steiner triple systems** are there of order 19?

The Goal

Determine if certain **combinatorial objects** exist with given **structural** or **extremal** properties.

Examples:

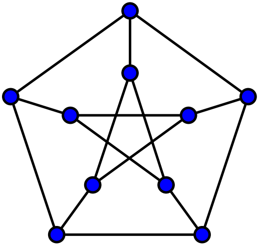
1. Is there a **projective plane** of order 10?
   (Lam, Thiel, Swiercz, 1989)

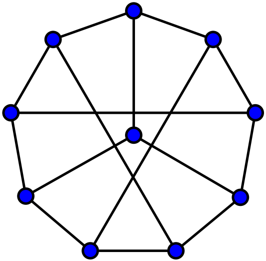
2. When do **strongly regular graphs** exist?
   (Spence 2000, Coolsaet, Degraer, Spence 2006, many others)

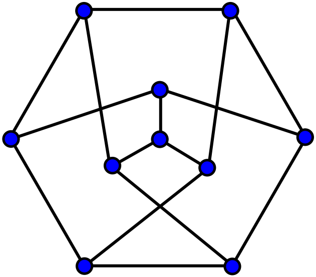
3. How many **Steiner triple systems** are there of order 19?
   (Kaski, Östergård, 2004)

Combinatorial Object: Graphs

A **graph** G of **order** n is composed of a set $V(G)$ of n vertices and a set $E(G)$ of edges, where the edges are unordered pairs of vertices.

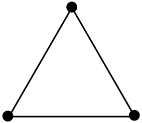




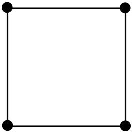


Combinatorial Object: Graphs

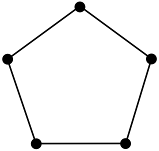
Cycles C_k



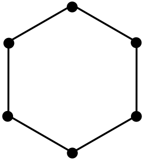
C_3



C_4



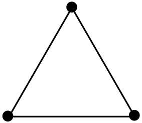
C_5



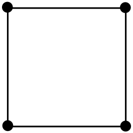
C_6

Combinatorial Object: Graphs

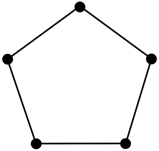
Cycles C_k



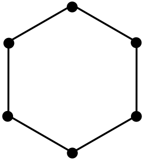
C_3



C_4

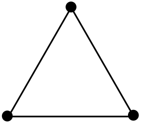


C_5

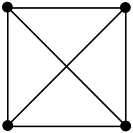


C_6

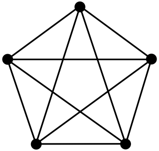
Complete Graphs K_r (cliques)



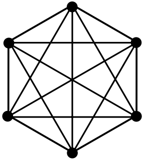
K_3



K_4



K_5



K_6

Main Technique: Combinatorial Search

Goal: Determine if certain combinatorial objects exist with given structural or extremal properties.

Idea: Build objects *piece-by-piece* from *base examples* to enumerate all desired examples of a given order.

Main Technique: Combinatorial Search

Goal: Determine if certain combinatorial objects exist with given structural or extremal properties.

Idea: Build objects *piece-by-piece* from *base examples* to enumerate all desired examples of a given order.

The computer performs a long, detailed case analysis.

Main Technique: Combinatorial Search

Goal: Determine if certain combinatorial objects exist with given structural or extremal properties.

Idea: Build objects *piece-by-piece* from *base examples* to enumerate all desired examples of a given order.

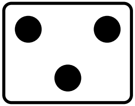
The computer performs a long, detailed case analysis.

Our job is to efficiently design the case analysis, using **algorithms**:

- Combinatorial Generation
- Combinatorial Optimization
- Graph Algorithms

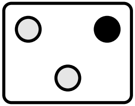
Example: Generating Graphs by Edges

We can build graphs starting at $\overline{K_n}$ by adding edges.



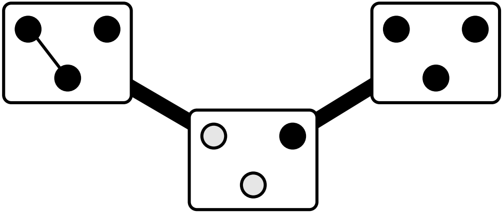
Example: Generating Graphs by Edges

We can build graphs starting at $\overline{K_n}$ by adding edges.



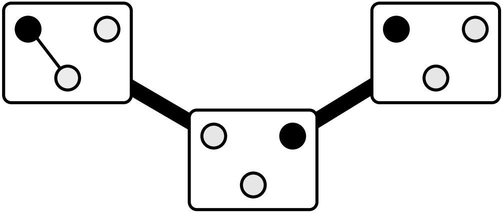
Example: Generating Graphs by Edges

We can build graphs starting at $\overline{K_n}$ by adding edges.



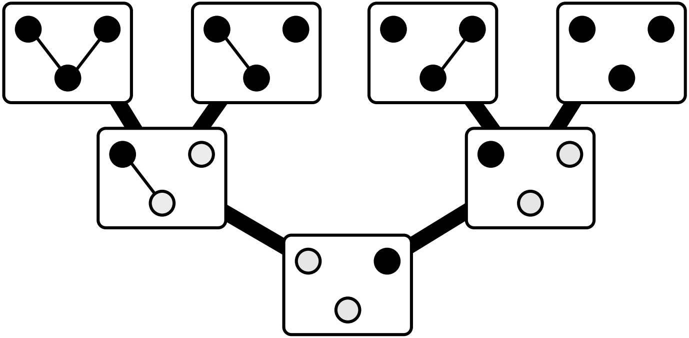
Example: Generating Graphs by Edges

We can build graphs starting at $\overline{K_n}$ by adding edges.



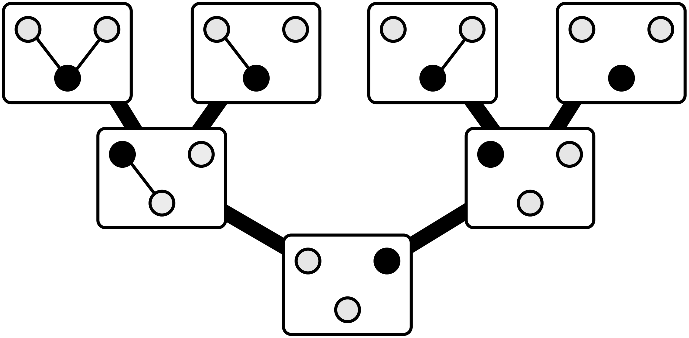
Example: Generating Graphs by Edges

We can build graphs starting at $\overline{K_n}$ by adding edges.



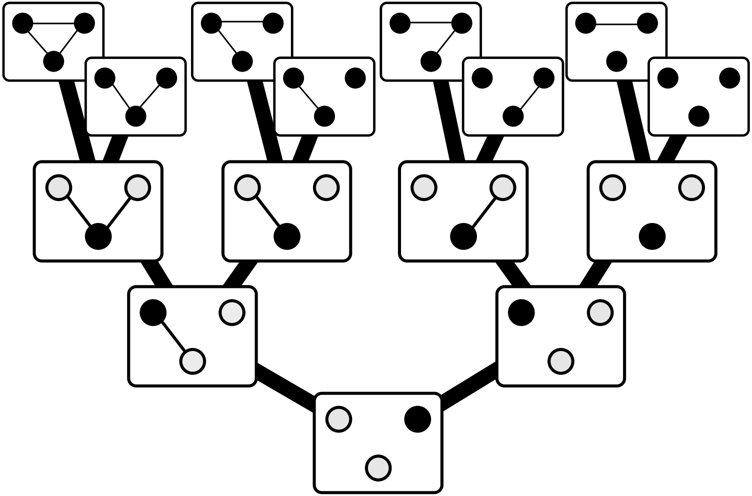
Example: Generating Graphs by Edges

We can build graphs starting at $\overline{K_n}$ by adding edges.



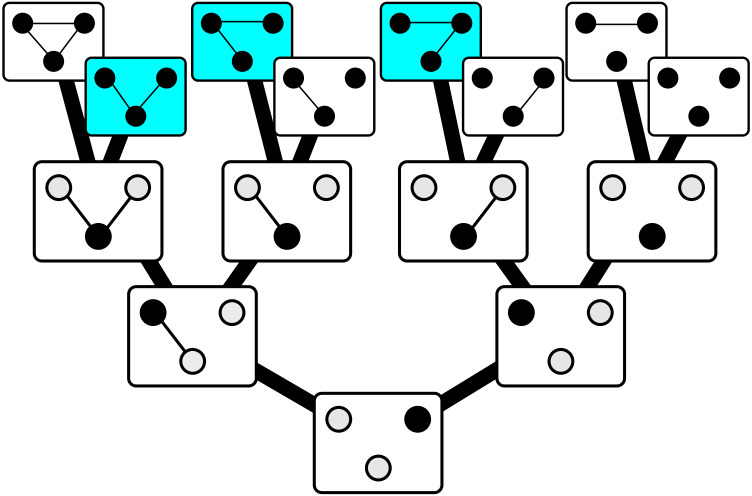
Example: Generating Graphs by Edges

We can build graphs starting at $\overline{K_n}$ by adding edges.



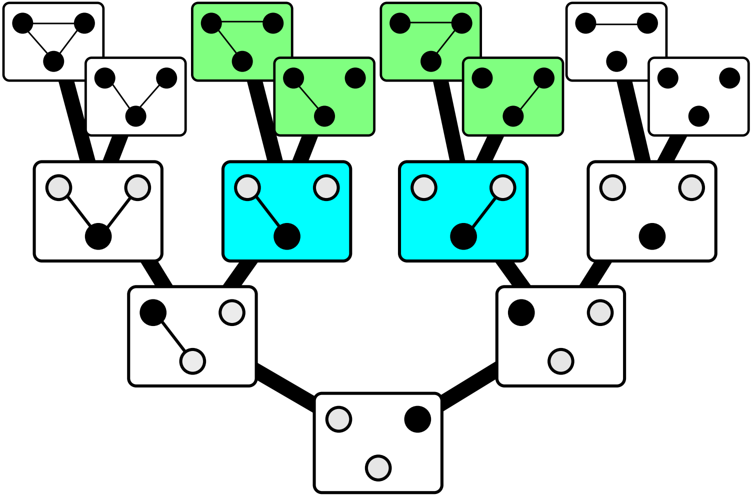
Example: Generating Graphs by Edges

We can build graphs starting at $\overline{K_n}$ by adding edges.

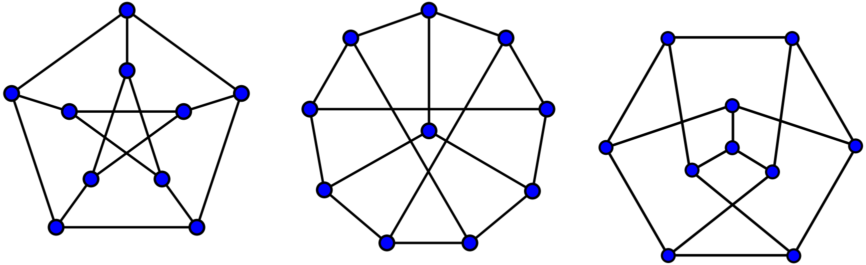


Example: Generating Graphs by Edges

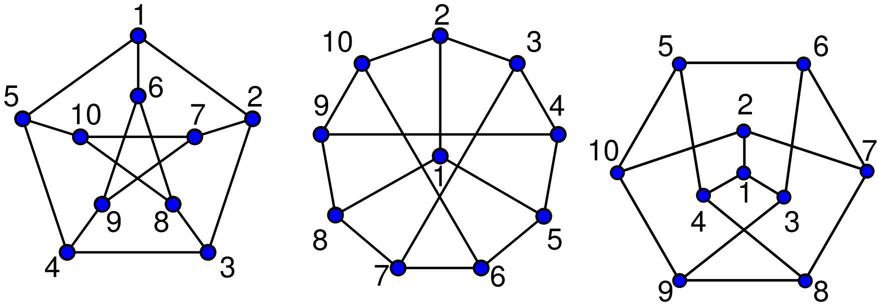
We can build graphs starting at $\overline{K_n}$ by adding edges.



Example: Generating Graphs by Edges

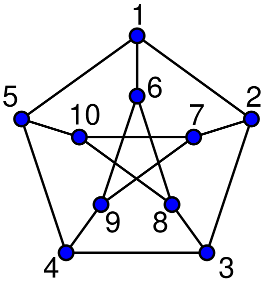


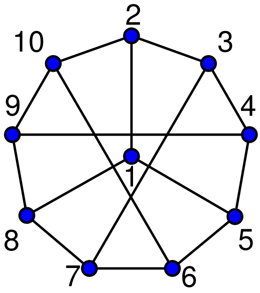
Example: Generating Graphs by Edges

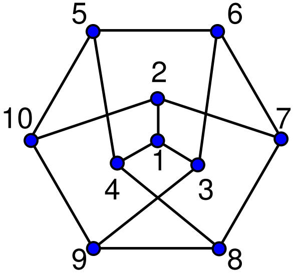


Example: Generating Graphs by Edges

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 $E(G_2)$.







Labeled Versus Unlabeled Objects

A **labeled** graph has a linear ordering on the vertices.

An **unlabeled** graph represents an isomorphism class of graphs.

Labeled Versus Unlabeled Objects

A **labeled** graph has a linear ordering on the vertices.

An **unlabeled** graph represents an isomorphism class of graphs.

Most interesting graph properties are **invariant under isomorphism**.

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 $2^{\Omega(n^2)}$.

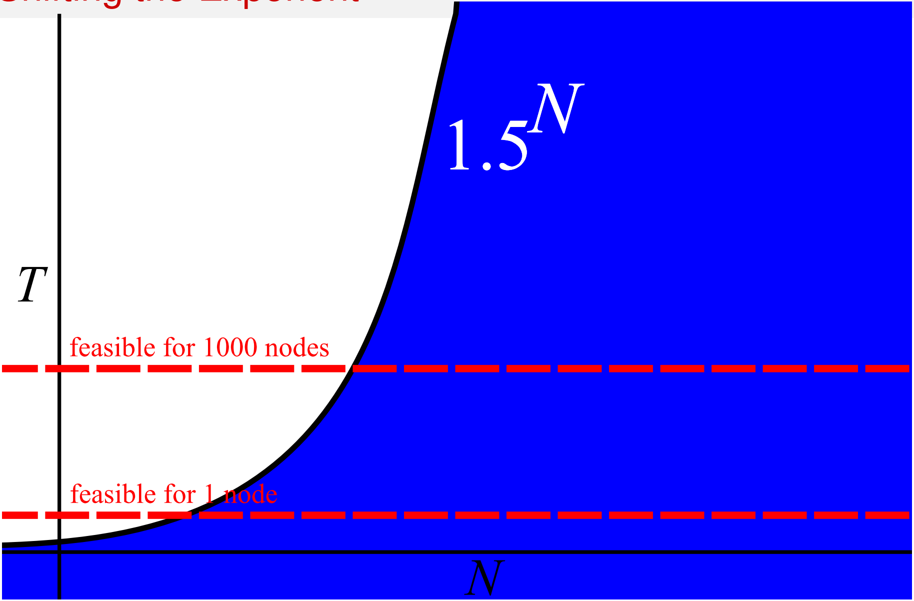
n	Unlabeled connected graphs of order *n*
6	85
7	509
8	4,060
9	41,301
10	510,489
11	7,319,447
12	**117,940,535**
13	2,094,480,864
14	40,497,138,011
15	845,480,228,069

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

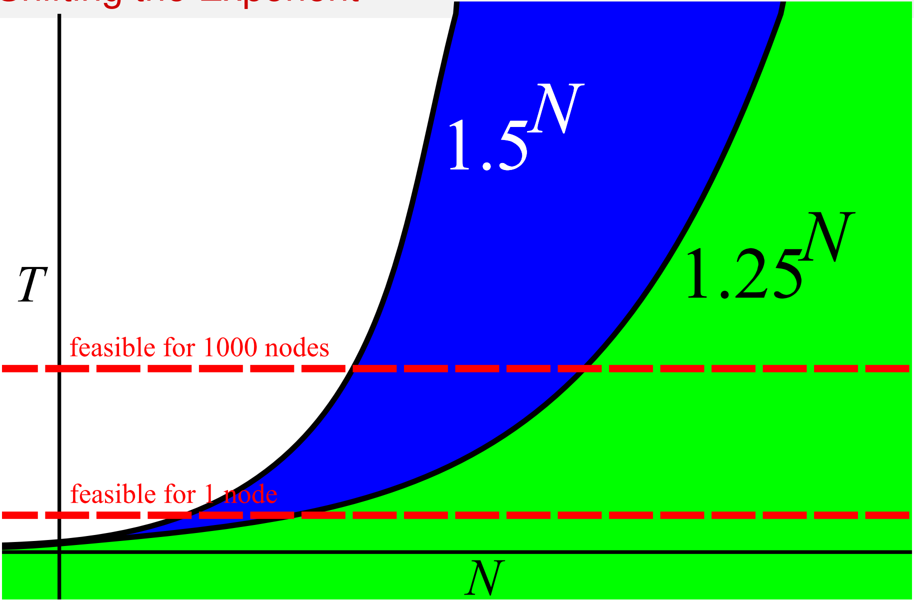
n	Unlabeled connected graphs of order *n*
6	85
7	509
8	4,060
9	41,301
10	510,489
11	7,319,447
12	117,940,535
13	2,094,480,864
14	**40,497,138,011**
15	845,480,228,069

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

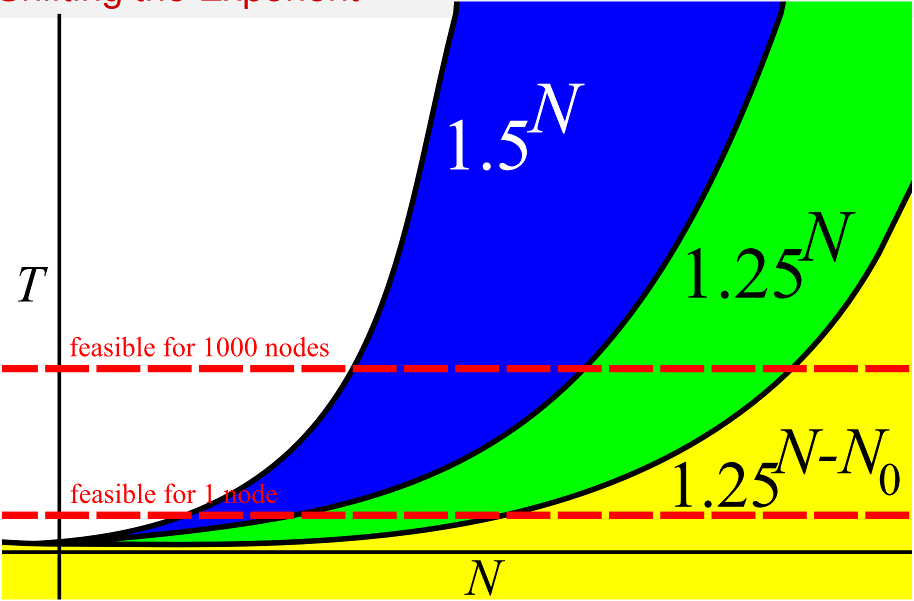
Shifting the Exponent



Shifting the Exponent

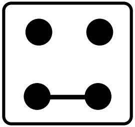


Shifting the Exponent

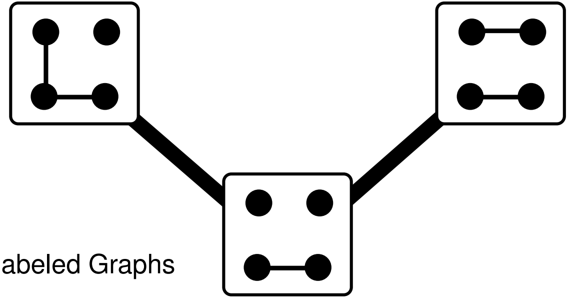


Example: Generating Graphs by Edges

Unlabeled Graphs

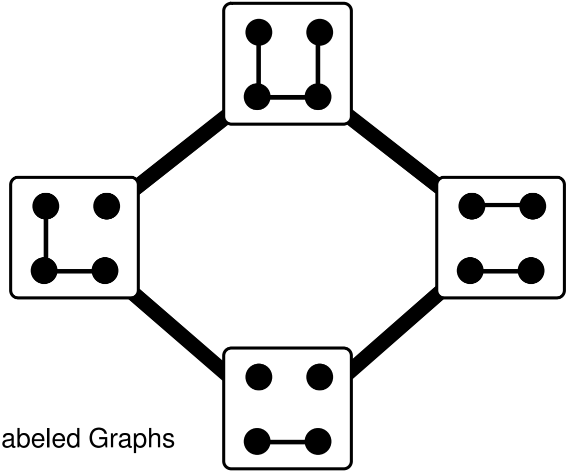


Example: Generating Graphs by Edges



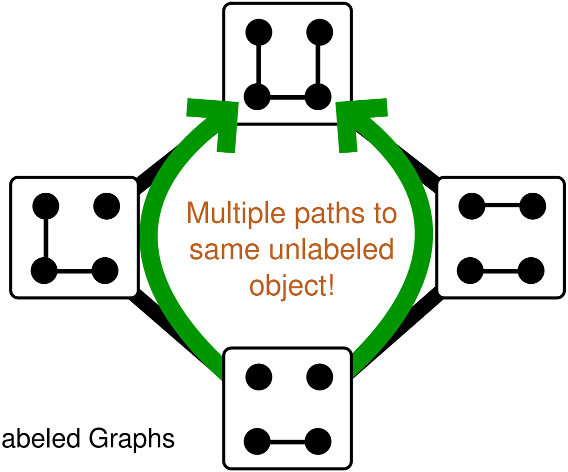
Unlabeled Graphs

Example: Generating Graphs by Edges



Unlabeled Graphs

Example: Generating Graphs by Edges



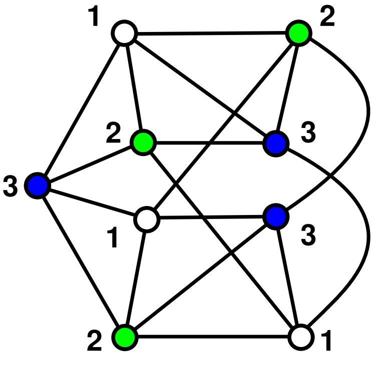
Multiple paths to same unlabeled object!

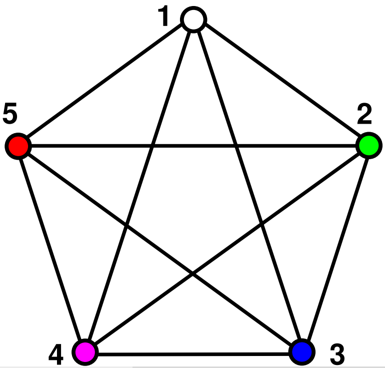
Unlabeled Graphs

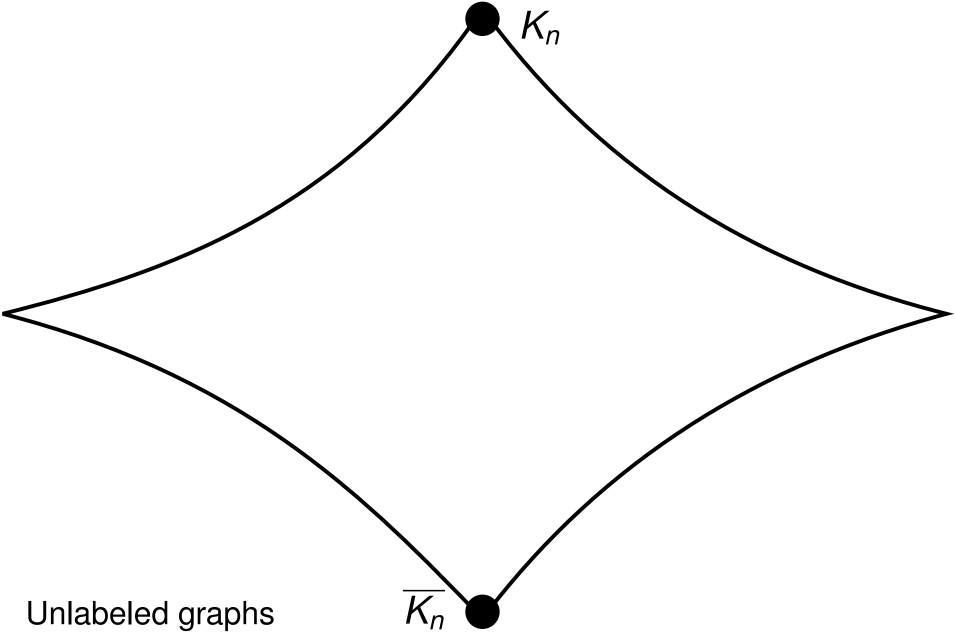
Toy Example

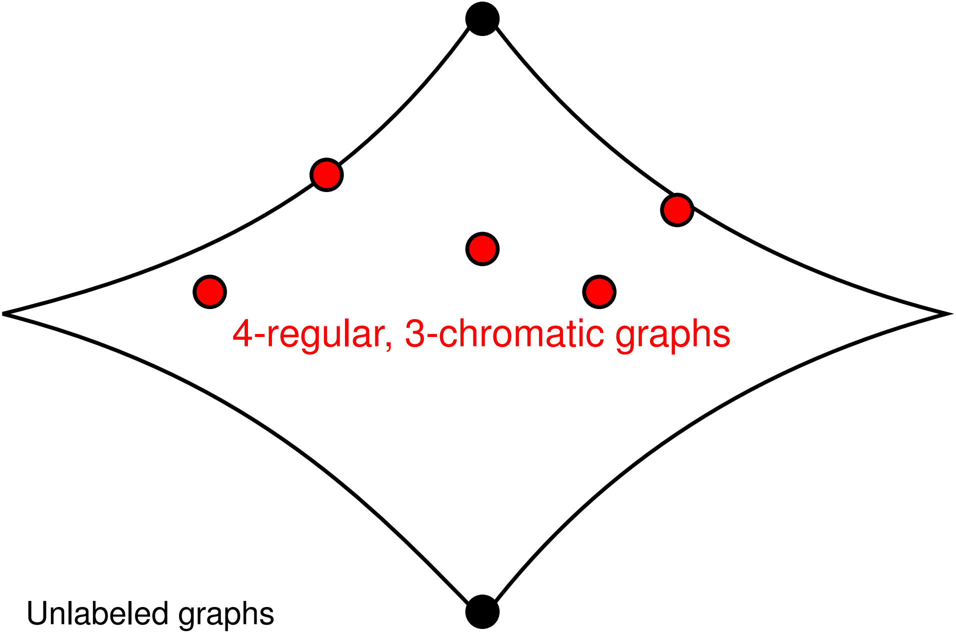
Suppose we are searching for graphs which are:

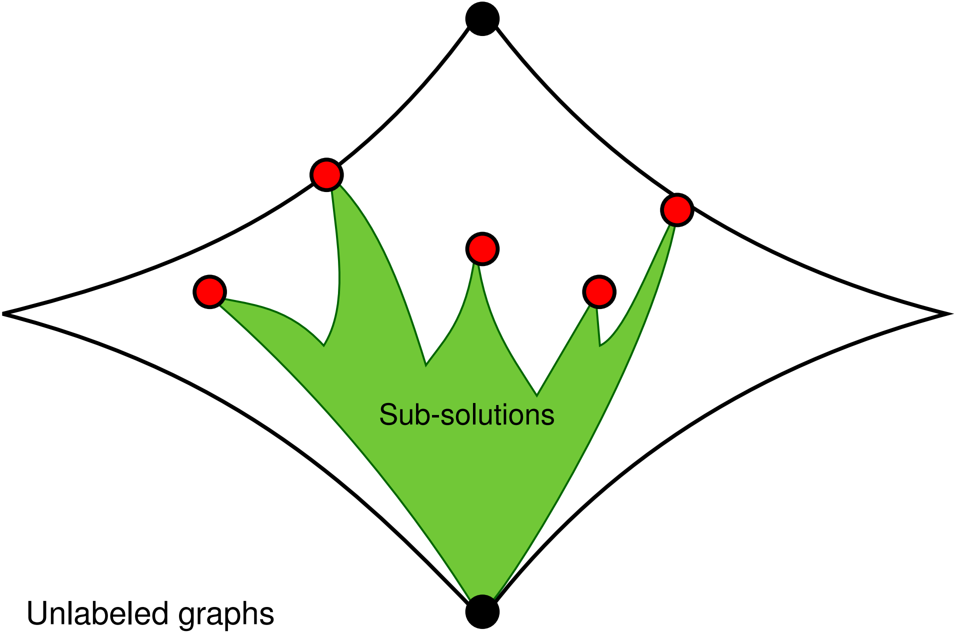
1. **4-regular**: All vertices have 4 incident edges.
2. **3-colorable**: The vertices can be colored with three colors so that no edge is monochromatic.

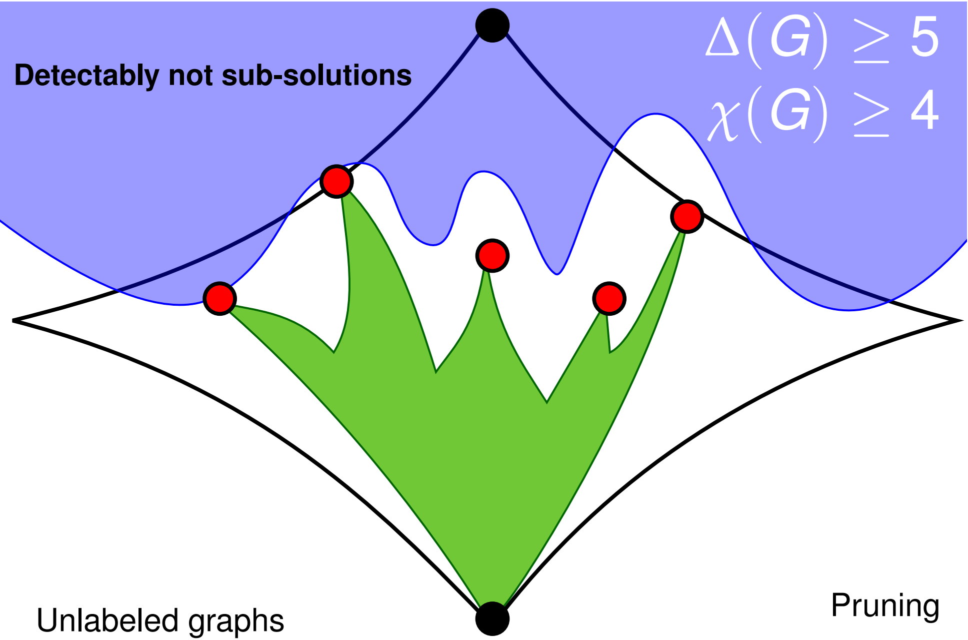


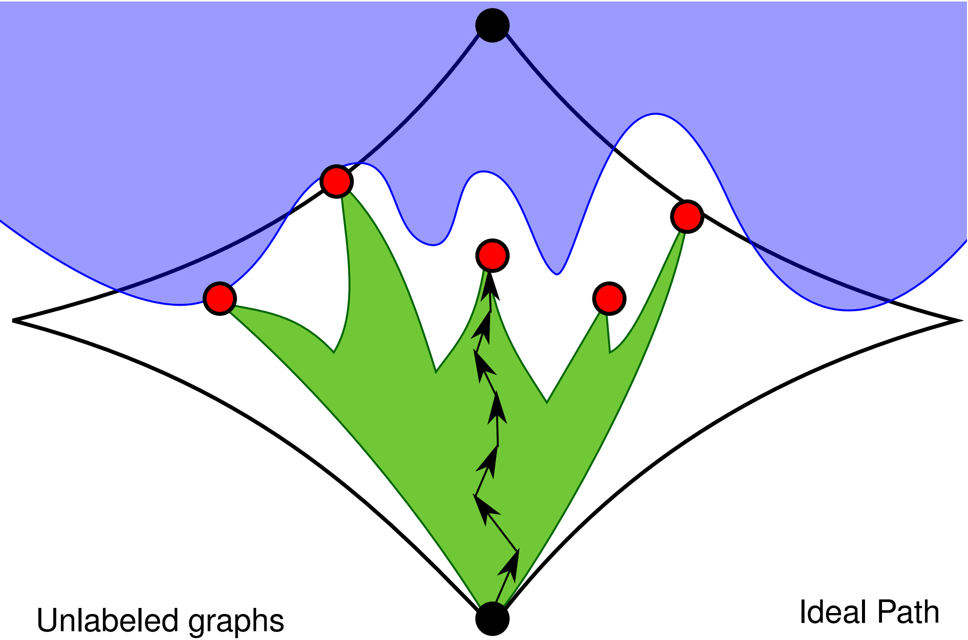


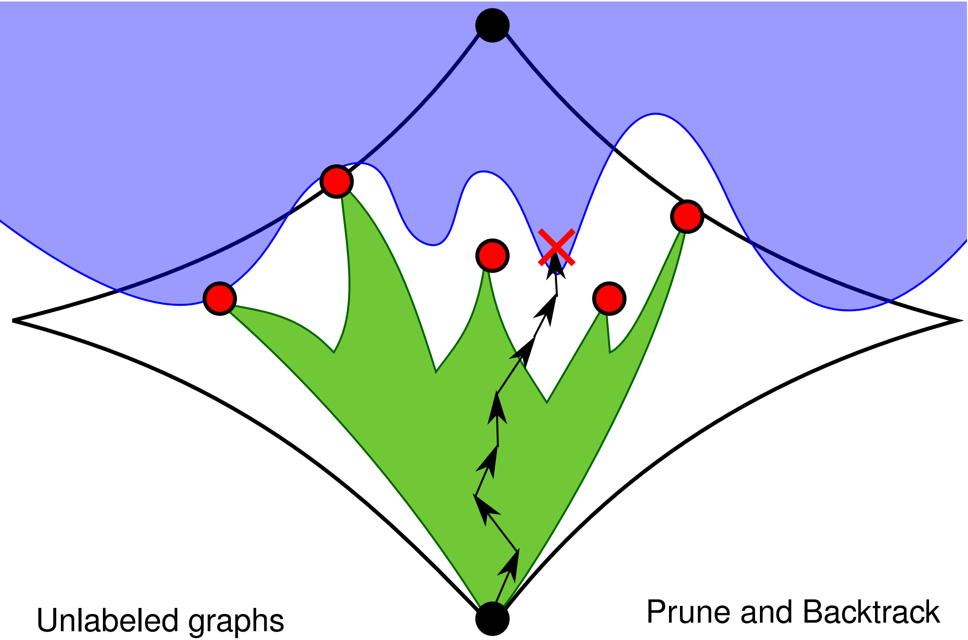

K_n
$\overline{K_n}$
Unlabeled graphs


4-regular, 3-chromatic graphs
Unlabeled graphs

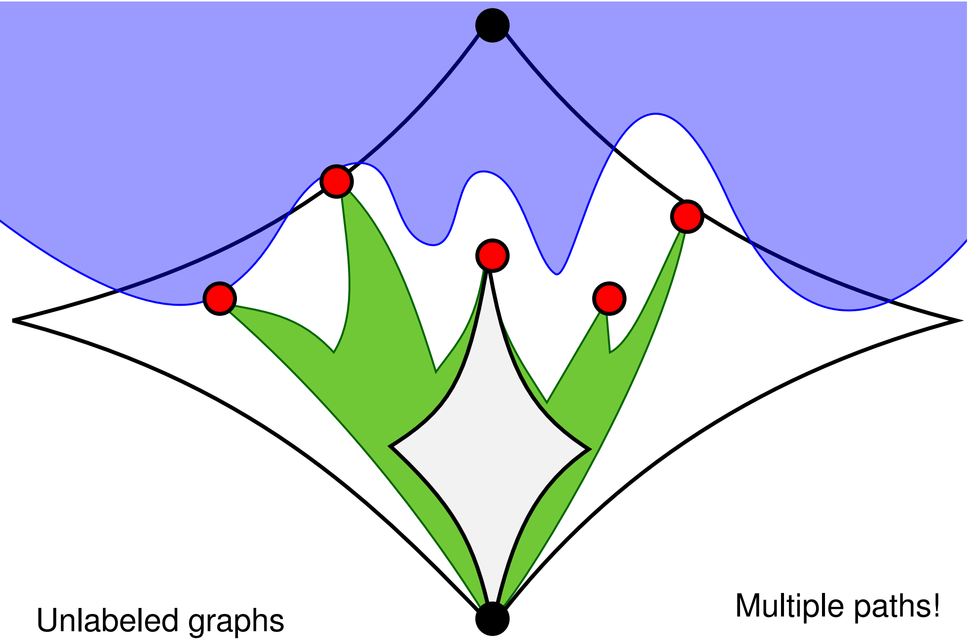

Sub-solutions
Unlabeled graphs


Detectably not sub-solutions
$\Delta(G) \geq 5$
$\chi(G) \geq 4$
Unlabeled graphs
Pruning


Unlabeled graphs
Ideal Path

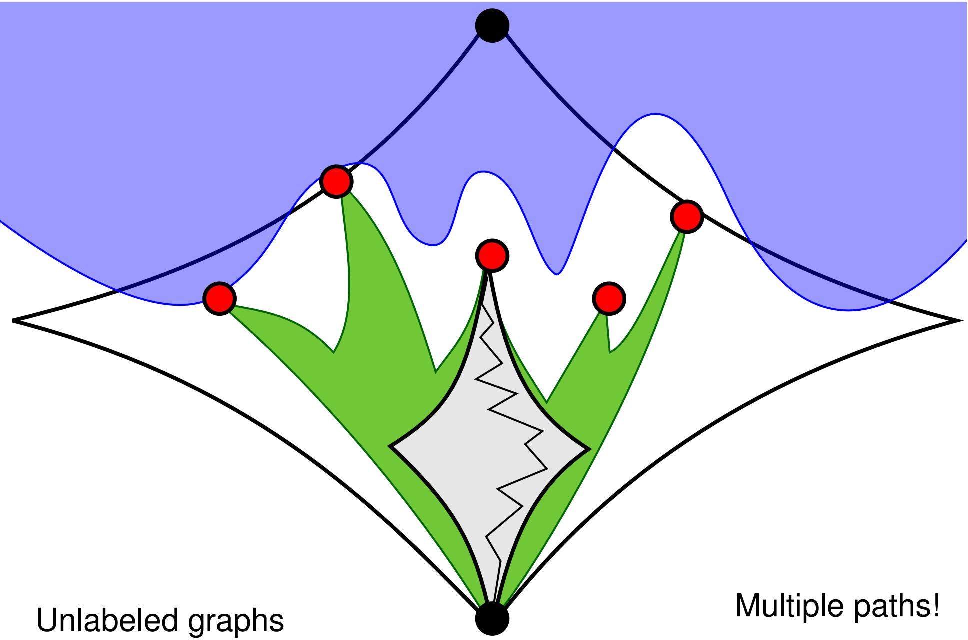


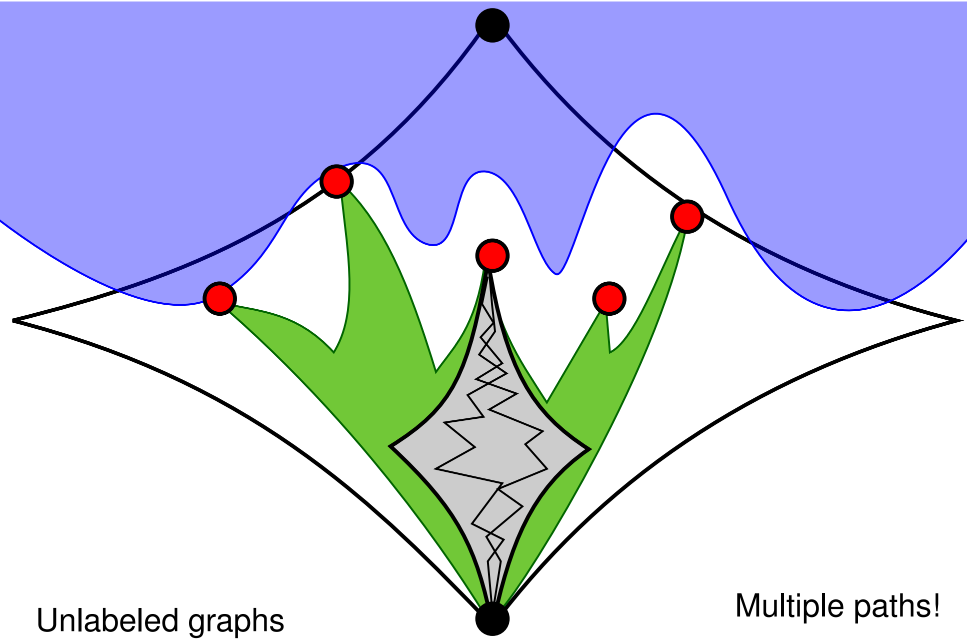
Unlabeled graphs
Prune and Backtrack



Unlabeled graphs

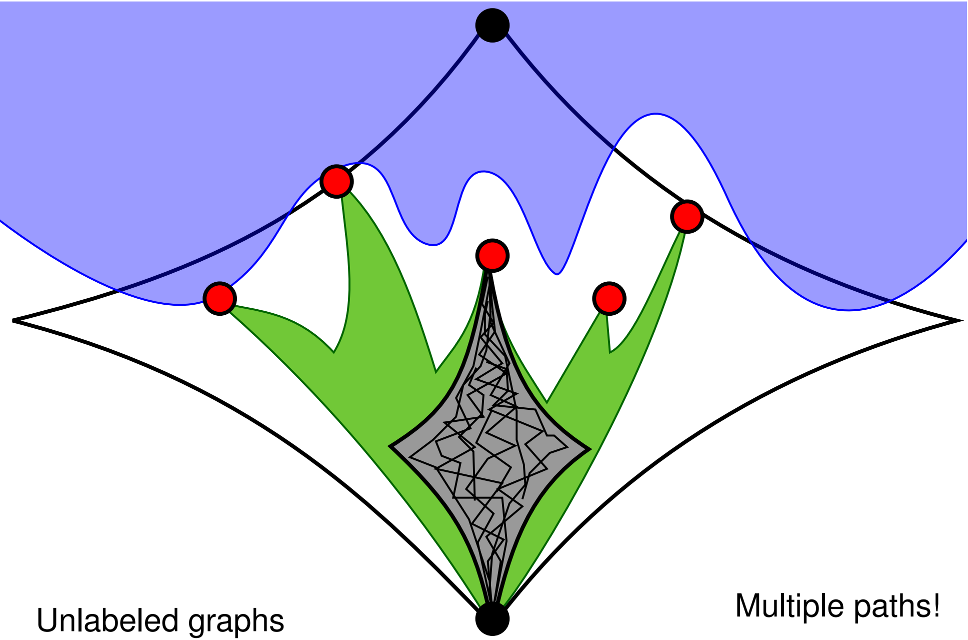
Multiple paths!


Unlabeled graphs
Multiple paths!



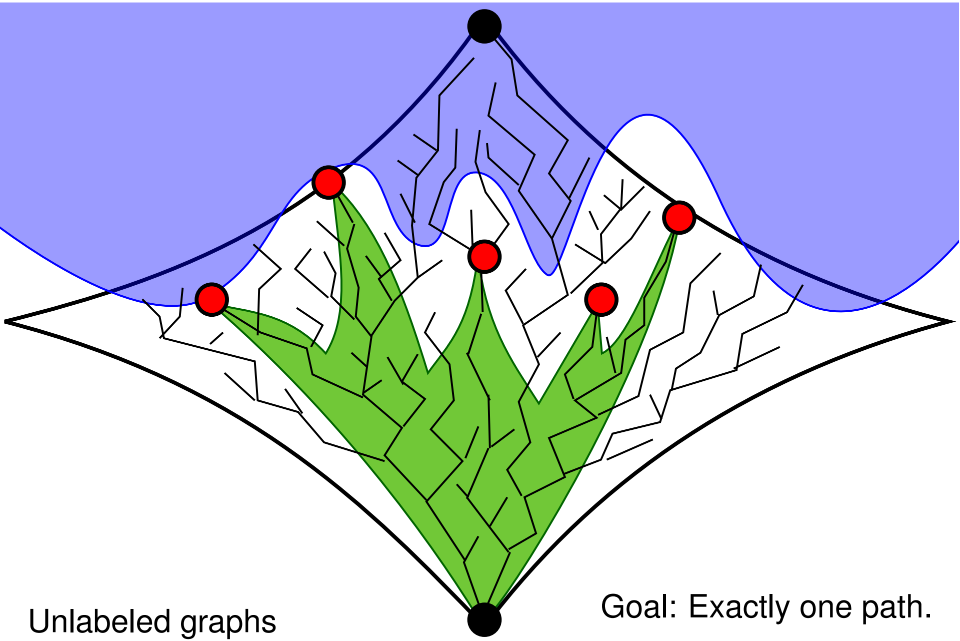
Unlabeled graphs

Multiple paths!



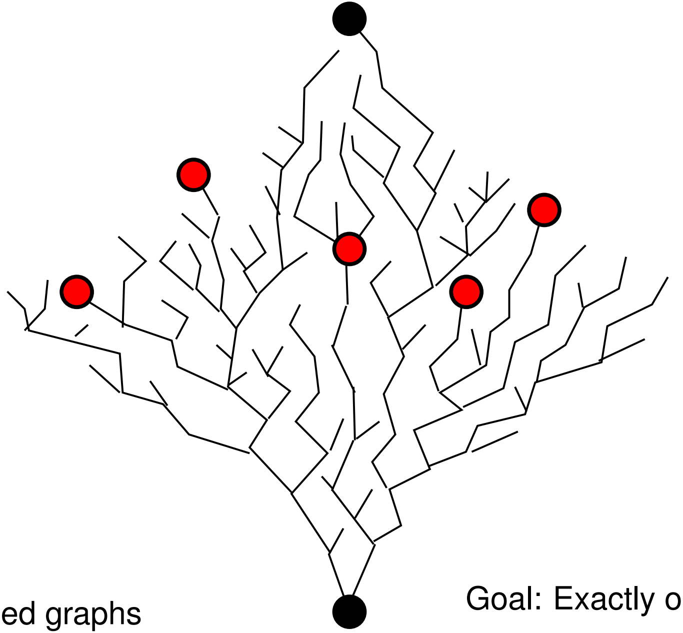
Unlabeled graphs

Multiple paths!



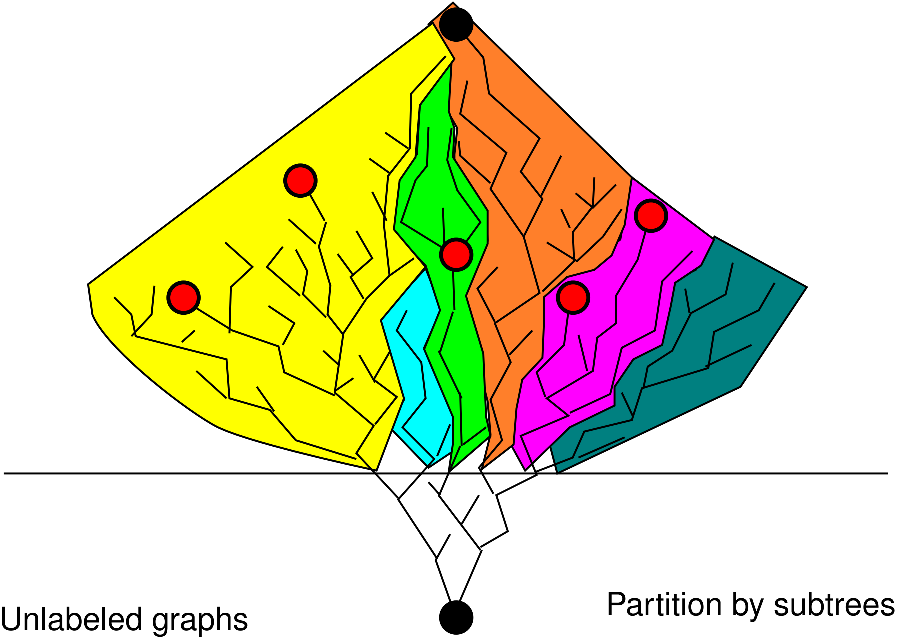
Unlabeled graphs

Goal: Exactly one path.



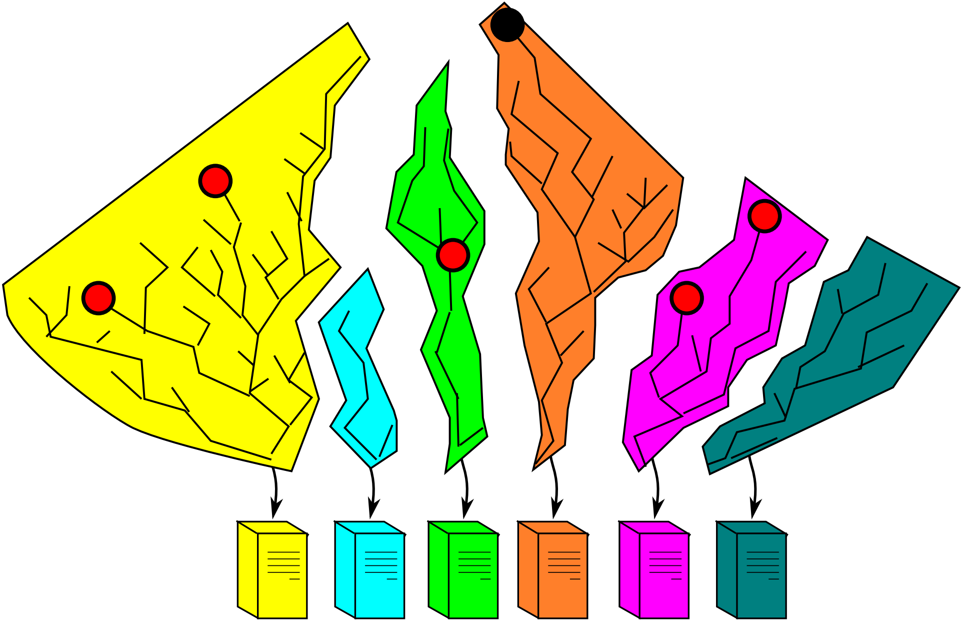
Unlabeled graphs

Goal: Exactly one path.



Unlabeled graphs

Partition by subtrees.



Parallelize!

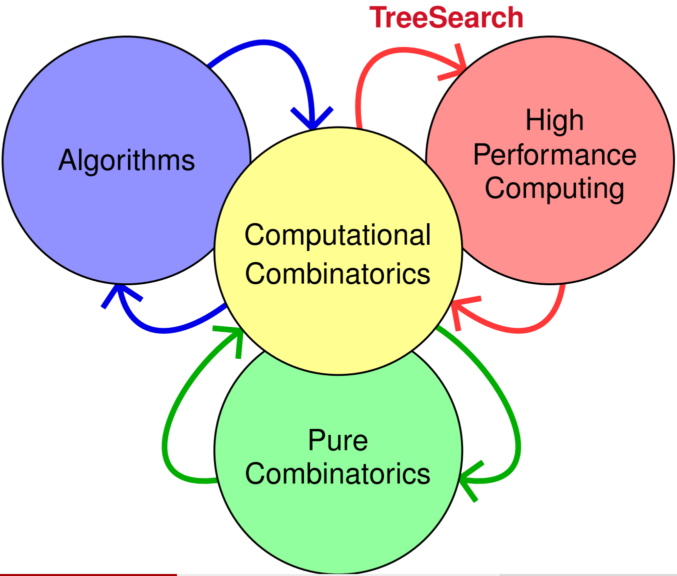
Implementation

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

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

Open Science Grid

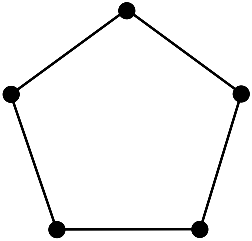
Computational Combinatorics

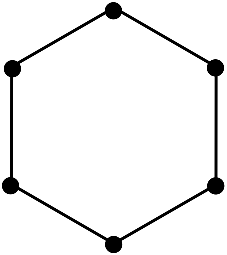


H-Saturated Graphs

Definition A graph G is ***H*-saturated** if

- G does not contain H as a subgraph. (***H*-free**)
- For every $e \in E(\overline{G})$, $G + e$ contains H as a subgraph.


5-cycle

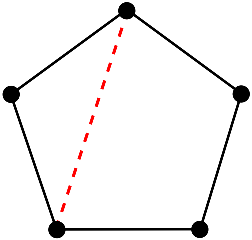

6-cycle

Example: $H = K_3$ where K_r is the **complete graph** on r vertices.

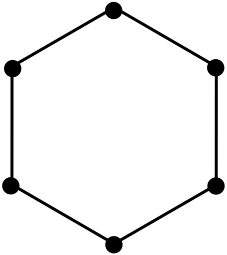
H-Saturated Graphs

Definition A graph G is ***H*-saturated** if

- G does not contain H as a subgraph. (***H*-free**)
- For every $e \in E(\overline{G})$, $G + e$ contains H as a subgraph.



5-cycle
is K_3-saturated



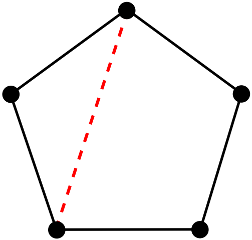
6-cycle

Example: $H = K_3$ where K_r is the **complete graph** on r vertices.

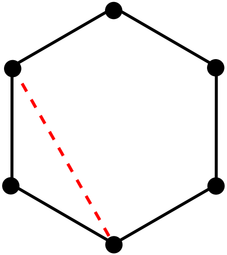
H-Saturated Graphs

Definition A graph G is ***H*-saturated** if

- G does not contain H as a subgraph. (***H*-free**)
- For every $e \in E(\overline{G})$, $G + e$ contains H as a subgraph.



5-cycle
is K_3-saturated



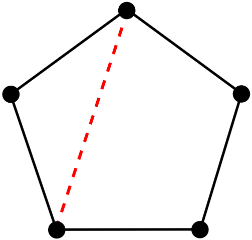
6-cycle

Example: $H = K_3$ where K_r is the **complete graph** on r vertices.

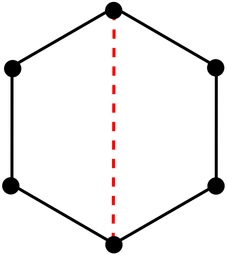
H-Saturated Graphs

Definition A graph G is ***H*-saturated** if

- G does not contain H as a subgraph. (***H*-free**)
- For every $e \in E(\overline{G})$, $G + e$ contains H as a subgraph.



5-cycle
is K_3-saturated



6-cycle
is **not** K_3-saturated

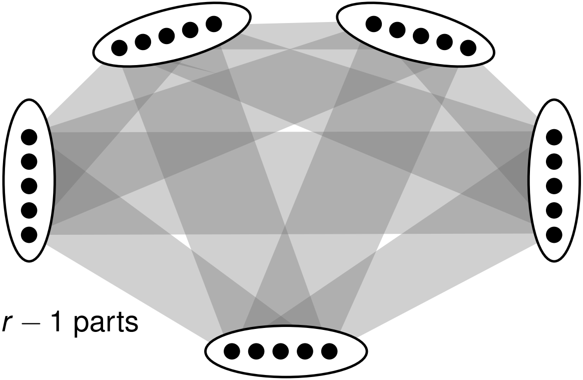
Example: $H = K_3$ where K_r is the **complete graph** on r vertices.

Turán's Theorem

Theorem (Turán, 1941) Let $r \geq 3$. If G is K_r-saturated on n vertices, then G has **at most** $\left(1 - \frac{1}{r-1}\right) \frac{n^2}{2}$ edges (asymptotically).

Turán's Theorem

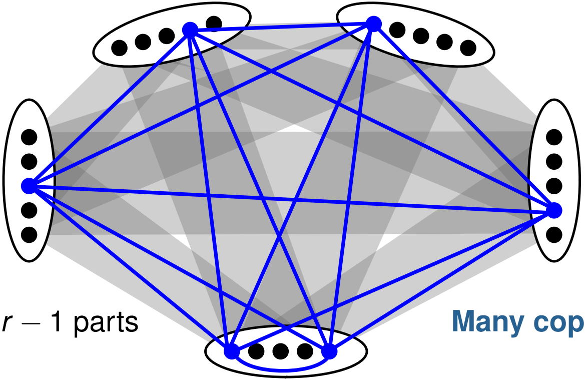
Theorem (Turán, 1941) Let $r \geq 3$. If G is K_r-saturated on n vertices, then G has **at most** $\left(1 - \frac{1}{r-1}\right)\frac{n^2}{2}$ edges (asymptotically).



$r - 1$ parts

Turán's Theorem

Theorem (Turán, 1941) Let $r \geq 3$. If G is K_r-saturated on n vertices, then G has **at most** $\left(1 - \frac{1}{r-1}\right)\frac{n^2}{2}$ edges (asymptotically).



$r - 1$ parts

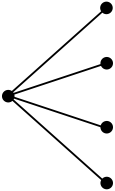
Many copies of K_r!

Erdős, Hajnal, and Moon

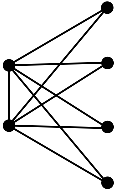
Theorem (Erdős, Hajnal, Moon, 1964) Let $r \geq 3$. If G is K_r-saturated on n vertices, then G has **at least** $\binom{r-2}{2} + (r-2)(n-r+2)$ edges.

Erdős, Hajnal, and Moon

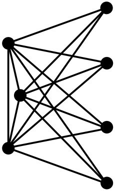
Theorem (Erdős, Hajnal, Moon, 1964) Let $r \geq 3$. If G is K_r-saturated on n vertices, then G has **at least** $\binom{r-2}{2} + (r-2)(n-r+2)$ edges.



1-book



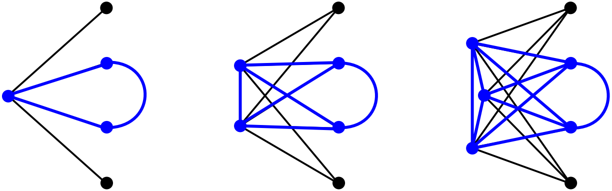
2-book



3-book

Erdős, Hajnal, and Moon

Theorem (Erdős, Hajnal, Moon, 1964) Let $r \geq 3$. If G is K_r-saturated on n vertices, then G has **at least** $\binom{r-2}{2} + (r-2)(n-r+2)$ edges.



1-book

2-book

3-book

Exactly one copy of K_r!

Uniquely H-Saturated Graphs

The Turán graph has **many** copies of K_r when an edge is added.

The books have **exactly one** copy of K_r when an edge is added.

Uniquely H-Saturated Graphs

The Turán graph has **many** copies of K_r when an edge is added.

The books have **exactly one** copy of K_r when an edge is added.

Definition A graph G is **uniquely *H*-saturated** if G does not contain H as a subgraph and for every edge $e \in \overline{G}$ admits **exactly one** copy of H in $G + e$.

Uniquely C_k-Saturated Graphs

Lemma (Cooper, Lenz, LeSaulnier, Wenger, West, 2011)

The uniquely C_3-saturated graphs are either **stars** or **Moore graphs** of diameter 2 and girth 5.

Uniquely C_k-Saturated Graphs

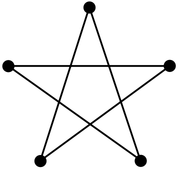
Lemma (Cooper, Lenz, LeSaulnier, Wenger, West, 2011)
The uniquely C_3-saturated graphs are either **stars** or **Moore graphs** of diameter 2 and girth 5.

Theorem (Hoffman, Singleton, 1964) There are a **finite number** of Moore graphs of diameter 2 and girth 5.

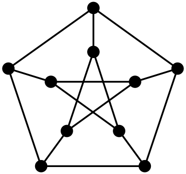
Uniquely C_k-Saturated Graphs

Lemma (Cooper, Lenz, LeSaulnier, Wenger, West, 2011)
The uniquely C_3-saturated graphs are either **stars** or **Moore graphs** of diameter 2 and girth 5.

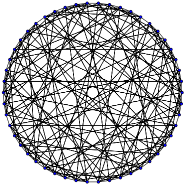
Theorem (Hoffman, Singleton, 1964) There are a **finite number** of Moore graphs of diameter 2 and girth 5.



C_5



Petersen



Hoffman–
Singleton

?

57-Regular
Order 3250

Uniquely C_k-Saturated Graphs

Theorem (Cooper, Lenz, LeSaulnier, Wenger, West, 2011)

There are a finite number of uniquely C_4-saturated graphs.

Uniquely C_k-Saturated Graphs

Theorem (Cooper, Lenz, LeSaulnier, Wenger, West, 2011)
There are a finite number of uniquely C_4-saturated graphs.

Theorem (Wenger, 2010)
The only uniquely C_5-saturated graphs are **friendship graphs**.

Uniquely C_k-Saturated Graphs

Theorem (Cooper, Lenz, LeSaulnier, Wenger, West, 2011)
There are a finite number of uniquely C_4-saturated graphs.

Theorem (Wenger, 2010)
The only uniquely C_5-saturated graphs are **friendship graphs**.

Theorem (Wenger, 2010)
For $k \in \{6, 7, 8\}$, no uniquely C_k-saturated graph exists.

Conjecture (Wenger, 2010)
For $k \geq 9$, no uniquely C_k-saturated graph exists.

Uniquely K_r-Saturated Graphs

We consider the case where $H = K_r$ (an ***r*-clique**) for $r \geq 4$.

Uniquely K_r-Saturated Graphs

We consider the case where $H = K_r$ (an ***r*-clique**) for $r \geq 4$.

($K_3 \cong C_3$)

Uniquely K_r-Saturated Graphs

We consider the case where $H = K_r$ (an ***r*-clique**) for $r \geq 4$.

($K_3 \cong C_3$)

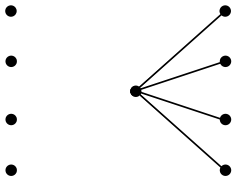
Definition A graph G is **uniquely K_r-saturated** if G does not contain an ***r*-clique** and for every edge $e \in \overline{G}$ there is **exactly one *r*-clique** in $G + e$.

Dominating Vertices

Adding a dominating vertex to a uniquely K_r-saturated graph creates a uniquely K_{r+1}-saturated graph.

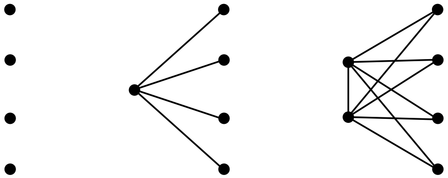
Dominating Vertices

Adding a dominating vertex to a uniquely K_r-saturated graph creates a uniquely K_{r+1}-saturated graph.



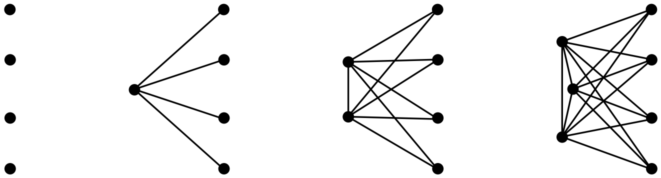
Dominating Vertices

Adding a dominating vertex to a uniquely K_r-saturated graph creates a uniquely K_{r+1}-saturated graph.



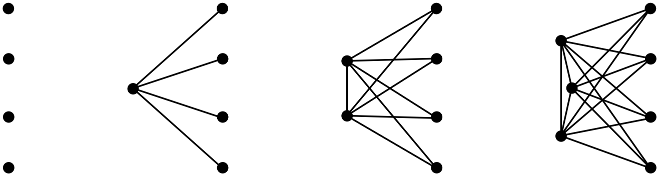
Dominating Vertices

Adding a dominating vertex to a uniquely K_r-saturated graph creates a uniquely K_{r+1}-saturated graph.



Dominating Vertices

Adding a dominating vertex to a uniquely K_r-saturated graph creates a uniquely K_{r+1}-saturated graph.



Call uniquely K_r-saturated graphs without a dominating vertex

***r*-primitive**.

r-Primitive Graphs

A uniquely K_r-saturated graph with no dominating vertex is r**-primitive**.

r-Primitive Graphs

A uniquely K_r-saturated graph with no dominating vertex is **r-primitive**.

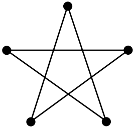
2**-primitive** graphs are **empty graphs**.

r-Primitive Graphs

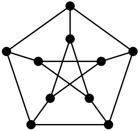
A uniquely K_r-saturated graph with no dominating vertex is ***r*-primitive**.

2**-primitive** graphs are **empty graphs**.

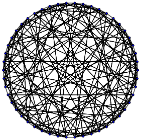
3**-primitive** graphs are **Moore graphs** of diameter 2 and girth 5.



C_5



Petersen



Hoffman–Singleton

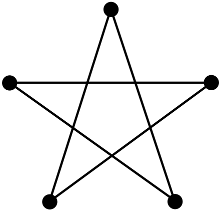
?

57-Regular
Order 3250

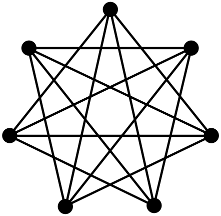
r-Primitive Graphs

A uniquely K_r-saturated graph with no dominating vertex is ***r*-primitive**.

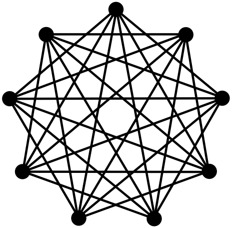
For $r \geq 1$, $\overline{C_{2r-1}}$ is r-primitive.



$\overline{C_5}$



$\overline{C_7}$



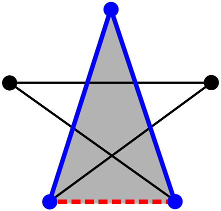
$\overline{C_9}$

(Collins, Cooper, Kay, Wenger, 2010)

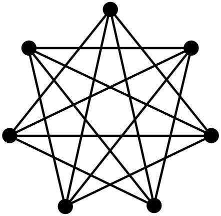
r-Primitive Graphs

A uniquely K_r-saturated graph with no dominating vertex is ***r*-primitive**.

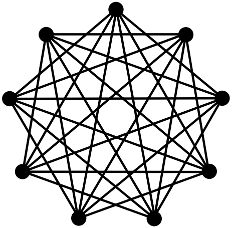
For $r \geq 1$, $\overline{C_{2r-1}}$ is r-primitive.



$\overline{C_5}$



$\overline{C_7}$



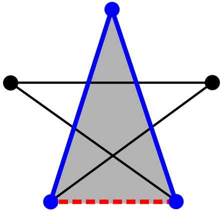
$\overline{C_9}$

(Collins, Cooper, Kay, Wenger, 2010)

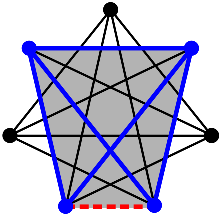
r-Primitive Graphs

A uniquely K_r-saturated graph with no dominating vertex is **r-primitive**.

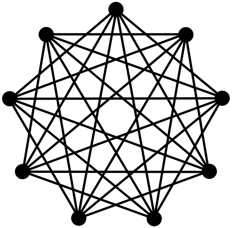
For $r \geq 1$, $\overline{C_{2r-1}}$ is r-primitive.



$\overline{C_5}$



$\overline{C_7}$



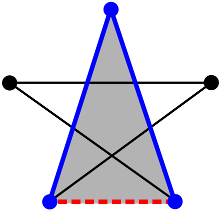
$\overline{C_9}$

(Collins, Cooper, Kay, Wenger, 2010)

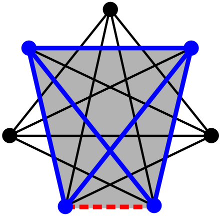
r-Primitive Graphs

A uniquely K_r-saturated graph with no dominating vertex is r**-primitive**.

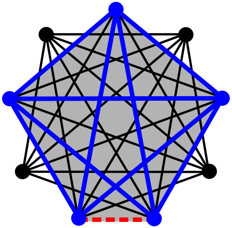
For $r \geq 1$, $\overline{C_{2r-1}}$ is r-primitive.



$\overline{C_5}$



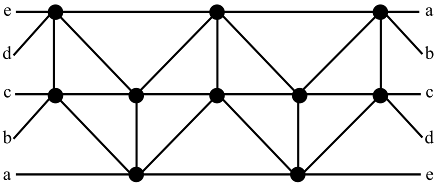
$\overline{C_7}$



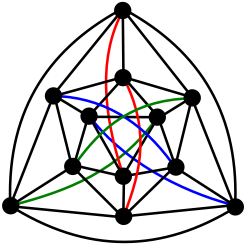
$\overline{C_9}$

(Collins, Cooper, Kay, Wenger, 2010)

Uniquely K_4-Saturated Graphs



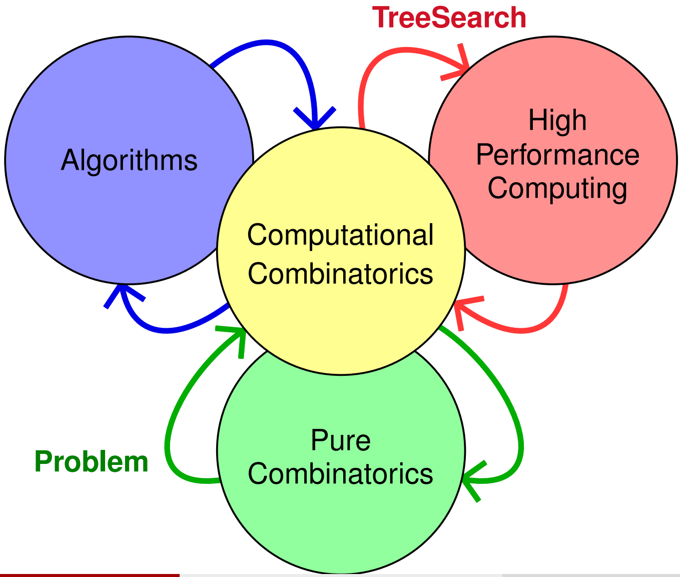
10 vertices



12 vertices

Previously known 4-primitive graphs (Collins, Cooper, Kay, 2010)

Computational Combinatorics



TreeSearch

Algorithms

High Performance Computing

Computational Combinatorics

Pure Combinatorics

Problem

The Problem

Goal: Characterize uniquely K_r-saturated graphs.

First Step: Reduce to r-primitive graphs.

The Problem

Goal: Characterize uniquely K_r-saturated graphs.

First Step: Reduce to r-primitive graphs.

1. Fix $r \geq 3$. Are there a **finite number** of r-primitive graphs?

The Problem

Goal: Characterize uniquely K_r-saturated graphs.

First Step: Reduce to r-primitive graphs.

1. Fix $r \geq 3$. Are there a **finite number** of r-primitive graphs?

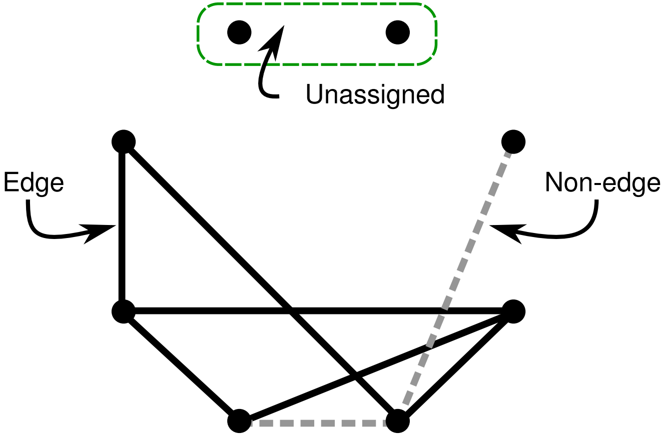
2. Is every r-primitive graph **regular**?

Edges and Non-Edges

Non-edges are crucial to the structure of r-primitive graphs.

Edges and Non-Edges

Non-edges are crucial to the structure of r-primitive graphs.



Tricolored graph

Edges, Non-Edges, and Variables

Fix a vertex set $\{v_1, v_2, \ldots, v_n\}$.

For $i, j \in \{1, \ldots, n\}$, let

$$x_{i,j} = \begin{cases} 1 & v_i v_j \in E(G) \\ 0 & v_i v_j \notin E(G) \\ * & v_i v_j \text{ unassigned} \end{cases}.$$

A vector $\mathbf{x} = (x_{i,j} : i, j \in \{1, \ldots, n\})$ is a **variable assignment**.

Symmetries of the System

The constraints

- There is no r-clique in G.
- Every non-edge e of G has exactly one r-clique in $G + e$.

are independent of vertex labels.

Symmetries of the System

The constraints

- ○ There is no r-clique in G.
- ○ Every non-edge e of G has exactly one r-clique in $G + e$.

are independent of vertex labels.

Automorphisms of the tricolored graph define **orbits** on variables $x_{i,j}$.

Orbital Branching

Orbital branching reduces the number of isomorphic duplicates.
(Ostrowski, Linderoth, Rossi, Smriglio, 2007)

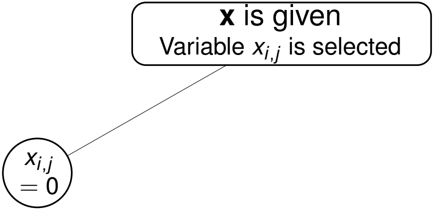
Generalizes **branch-and-bound** strategy from Integer Programming.

Branch-and-Bound

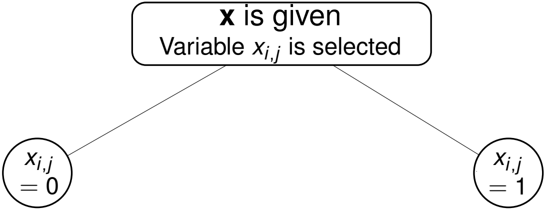
$\mathbf{x}$ is given

Variable $x_{i,j}$ is selected

Branch-and-Bound



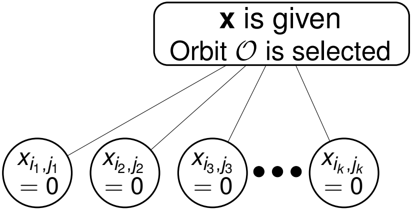
Branch-and-Bound



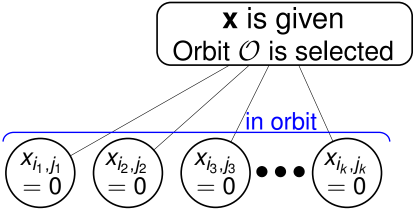
Orbital Branching

x is given
Orbit $\mathcal{O}$ is selected

Orbital Branching



Orbital Branching



x is given
Orbit $\mathcal{O}$ is selected

in orbit

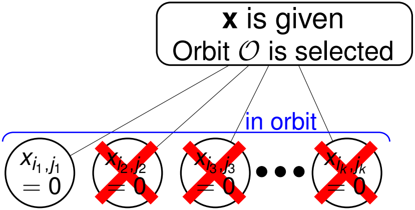
$x_{i_1,j_1} = 0$

$x_{i_2,j_2} = 0$

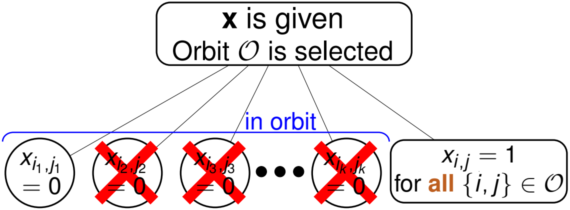
$x_{i_3,j_3} = 0$

$x_{i_k,j_k} = 0$

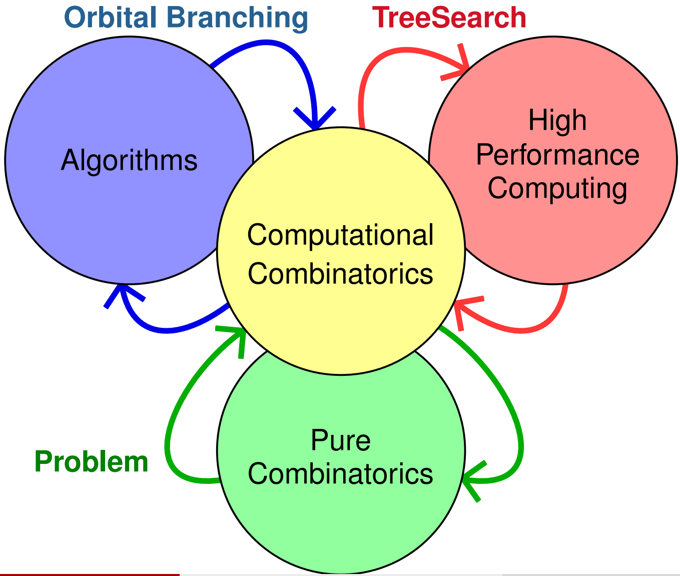
Orbital Branching

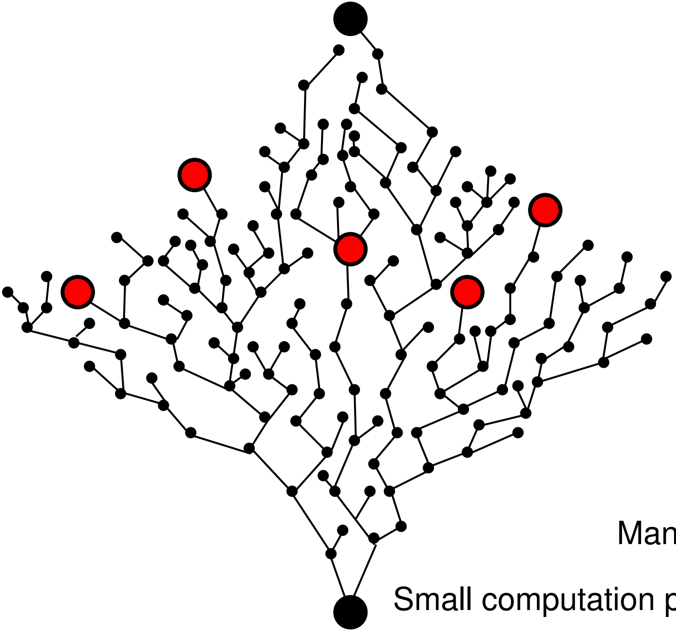


Orbital Branching



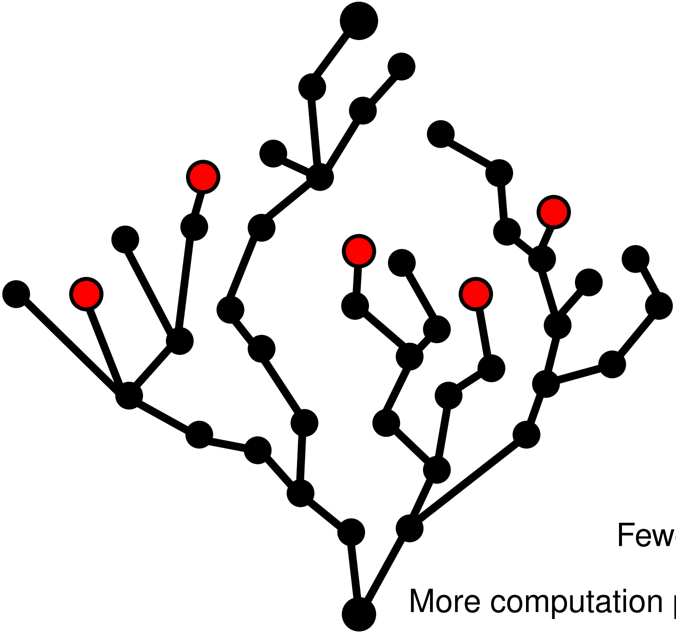
Computational Combinatorics





Many nodes.

Small computation per node.



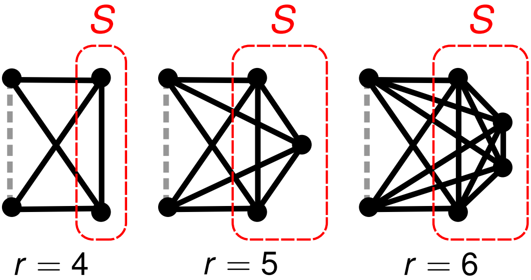
Fewer nodes.

More computation per node.

K_r-Completions

For every non-edge we add, we add a K_r-**completion**:

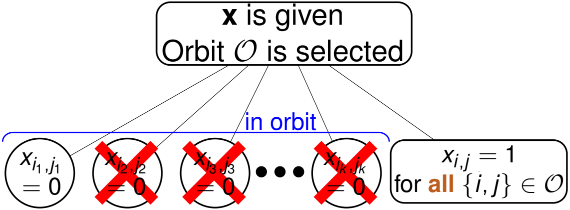
$x_{i,j} = 0$ **if and only if** there exists a set $S \subset [n]$, $|S| = r - 2$, so that $x_{i,a} = x_{j,a} = x_{a,b} = 1$ for all $a, b \in S$.



Orbital Branching with K_r-Completions

$\mathbf{x}$ is given
Orbit $\mathcal{O}$ is selected

Orbital Branching with K_r-Completions



x is given

Orbit $\mathcal{O}$ is selected

in orbit

$x_{i_1,j_1} = 0$

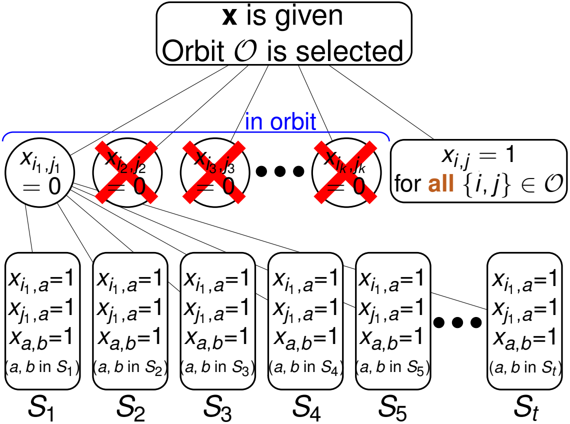
$x_{i_2,j_2} = 0$

$x_{i_3,j_3} = 0$

$x_{i_k,j_k} = 0$

$x_{i,j} = 1$ for **all** $\{i,j\} \in \mathcal{O}$

Orbital Branching with K_r-Completions



x is given

Orbit $\mathcal{O}$ is selected

in orbit

$x_{i_1,j_1} = 0$

$x_{i_2,j_2} = 0$

$x_{i_3,j_3} = 0$

$x_{i_k,j_k} = 0$

$x_{i,j} = 1$ for **all** $\{i,j\} \in \mathcal{O}$

$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_1) S_1

$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_2) S_2

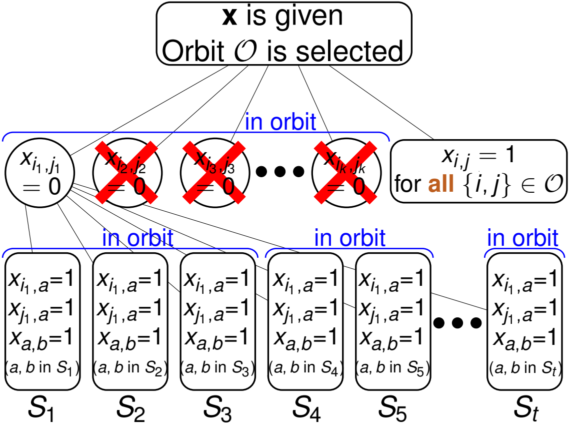
$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_3) S_3

$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_4) S_4

$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_5) S_5

$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_t) S_t

Orbital Branching with K_r-Completions



$\mathbf{x}$ is given

Orbit $\mathcal{O}$ is selected

in orbit

$x_{i_1,j_1} = 0$

$\cancel{x_{i_2,j_2} = 0}$

$\cancel{x_{i_3,j_3} = 0}$

$\cancel{x_{i_k,j_k} = 0}$

$x_{i,j} = 1$ for **all** $\{i,j\} \in \mathcal{O}$

in orbit

in orbit

in orbit

$x_{i_1,a}=1$ $x_{j_1,a}=1$ $x_{a,b}=1$ (a, b in S_1) — S_1

$x_{i_1,a}=1$ $x_{j_1,a}=1$ $x_{a,b}=1$ (a, b in S_2) — S_2

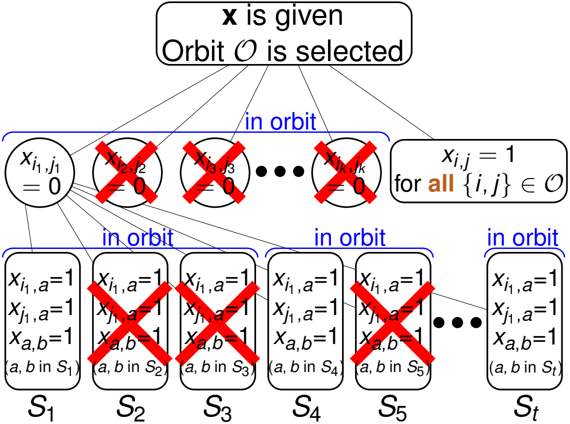
$x_{i_1,a}=1$ $x_{j_1,a}=1$ $x_{a,b}=1$ (a, b in S_3) — S_3

$x_{i_1,a}=1$ $x_{j_1,a}=1$ $x_{a,b}=1$ (a, b in S_4) — S_4

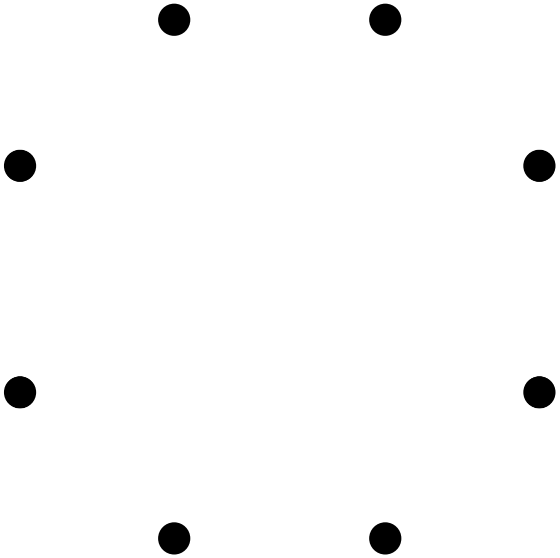
$x_{i_1,a}=1$ $x_{j_1,a}=1$ $x_{a,b}=1$ (a, b in S_5) — S_5

$x_{i_1,a}=1$ $x_{j_1,a}=1$ $x_{a,b}=1$ (a, b in S_t) — S_t

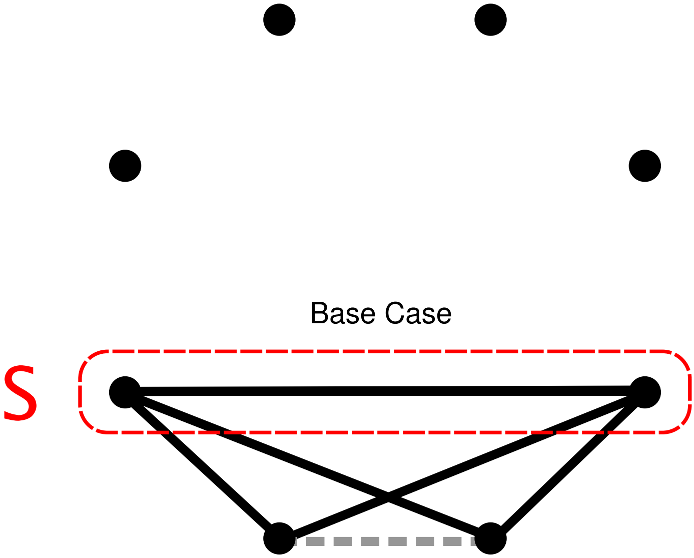
Orbital Branching with K_r-Completions


x is given
Orbit $\mathcal{O}$ is selected
in orbit
$x_{i_1,j_1} = 0$
$x_{i_2,j_2} = 0$
$x_{i_3,j_3} = 0$
$x_{i_k,j_k} = 0$
$x_{i,j} = 1$
for **all** $\{i,j\} \in \mathcal{O}$
in orbit
in orbit
in orbit
$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_1)
$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_2)
$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_3)
$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_4)
$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_5)
$x_{i_1,a}$=1 $x_{j_1,a}$=1 $x_{a,b}$=1 (a, b in S_t)
S_1
S_2
S_3
S_4
S_5
S_t

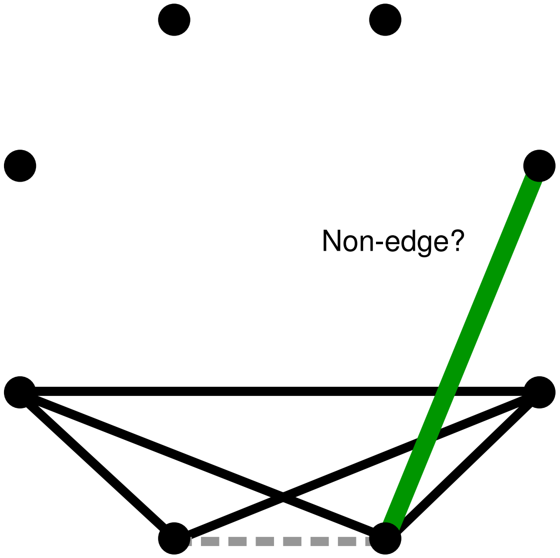
Orbital Branching with K_r-Completions



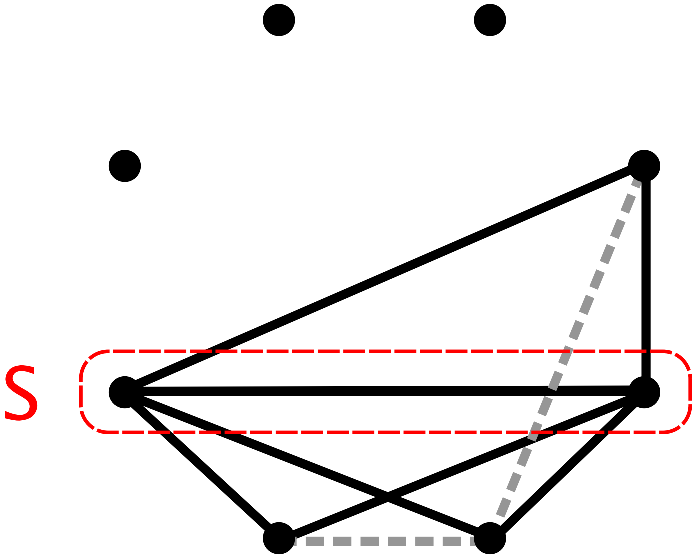
Orbital Branching with K_r-Completions



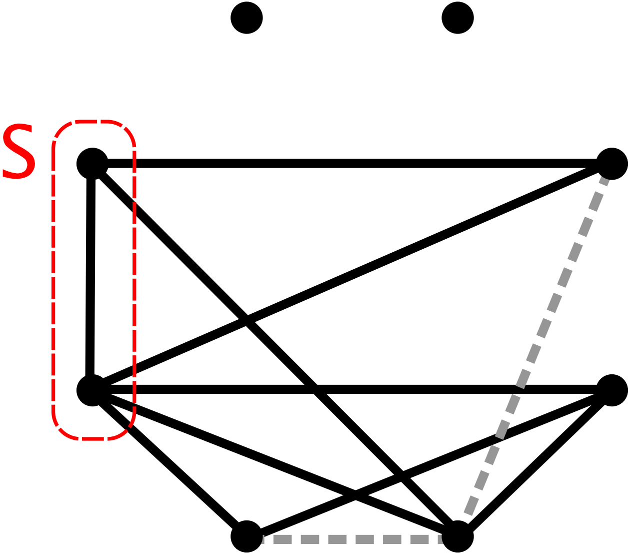
Orbital Branching with K_r-Completions



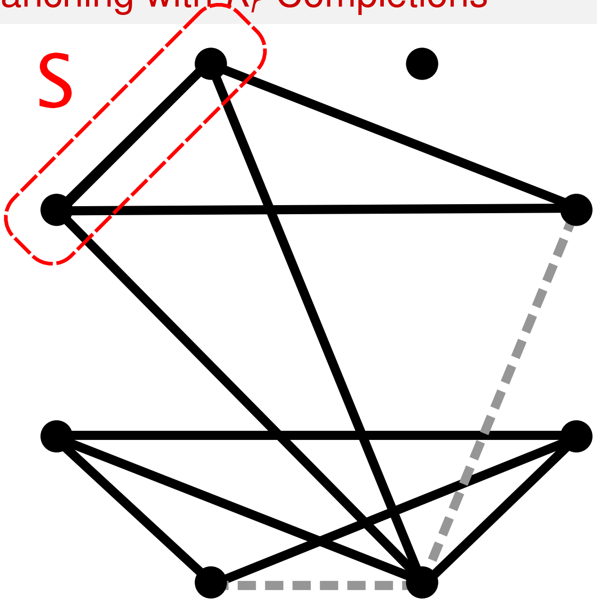
Orbital Branching with K_r-Completions



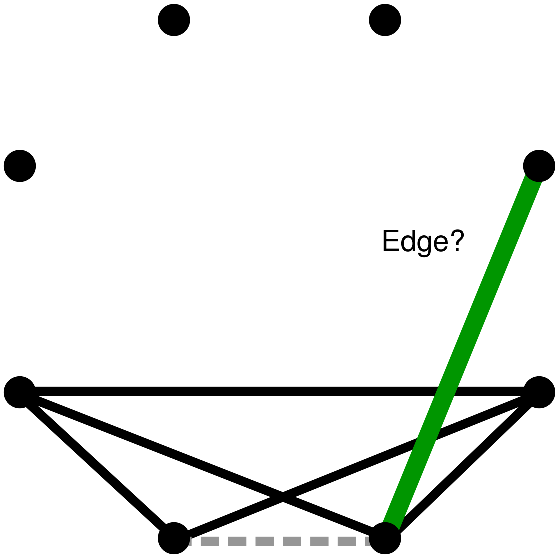
Orbital Branching with K_r-Completions



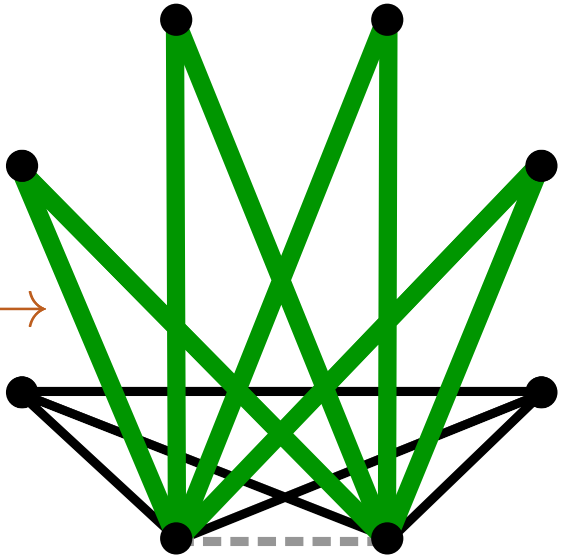
Orbital Branching with K_r-Completions



Orbital Branching with K_r-Completions

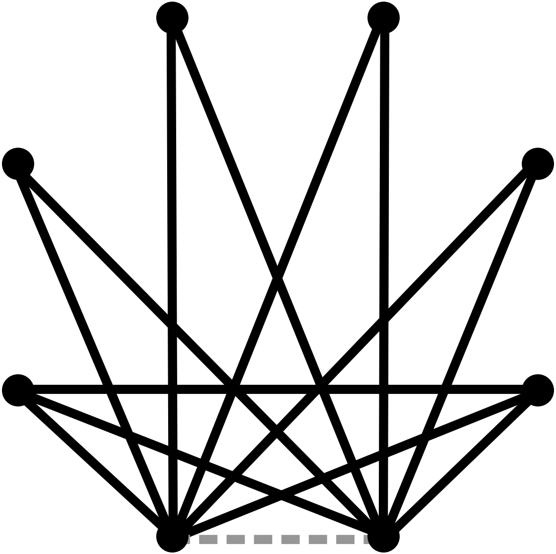


Orbital Branching with K_r-Completions

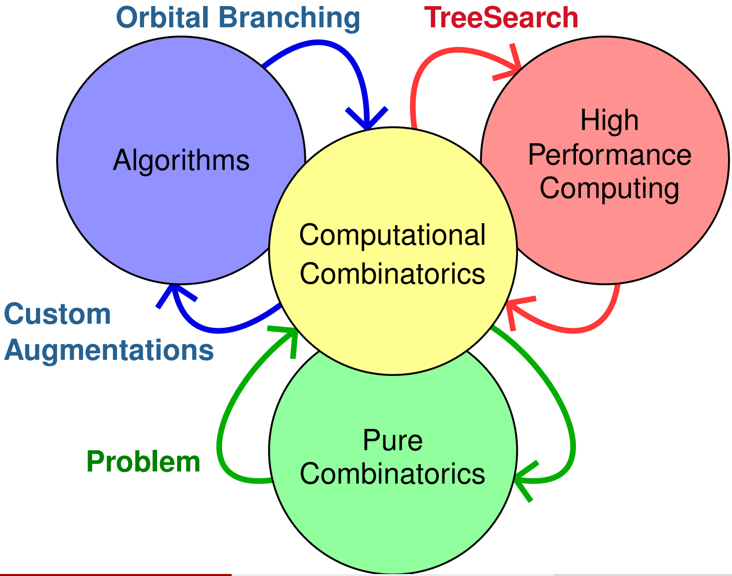


in orbit →

Orbital Branching with K_r-Completions



Computational Combinatorics



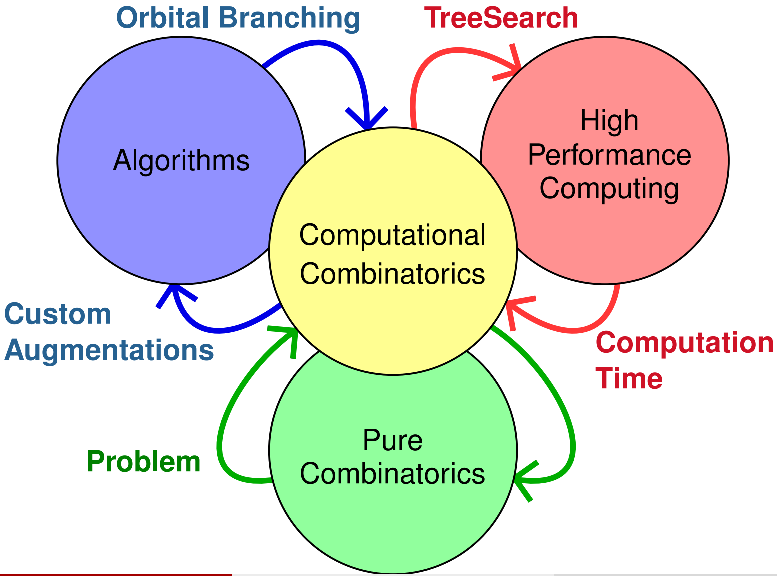
Exhaustive Search Times

n	$r = 4$	$r = 5$	$r = 6$	$r = 7$	$r = 8$
10	0.10 s	0.37 s	0.13 s	0.01 s	0.01 s
11	0.68 s	5.25 s	1.91 s	0.28 s	0.09 s
12	4.58 s	1.60 m	25.39 s	1.97 s	1.12 s
13	34.66 s	34.54 m	6.53 m	59.94 s	20.03 s
14	4.93 m	10.39 h	5.13 h	20.66 m	2.71 m
15	40.59 m	23.49 d	10.08 d	12.28 h	1.22 h
16	6.34 h	1.58 y	1.74 y	34.53 d	1.88 d
17	3.44 d			8.76 y	115.69 d
18	53.01 d				
19	2.01 y				
20	45.11 y				

Total CPU times using Open Science Grid.

($\approx 8.83 \times 10^{18}$ connected graphs of order 20)

Computational Combinatorics



⟵ clique size ⟶

⟵ vertices ⟶

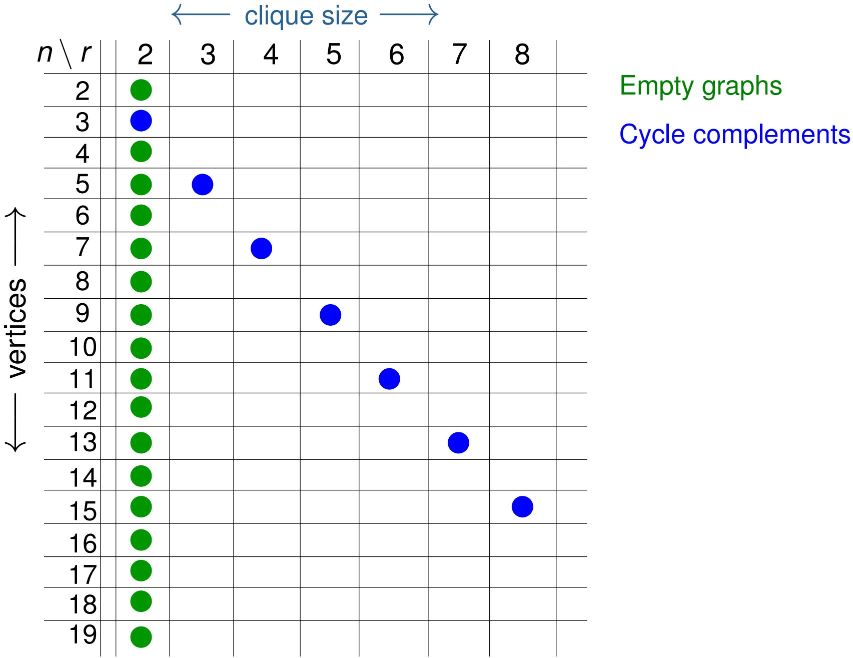
$n \setminus r$		2	3	4	5	6	7	8	
2									
3									
4									
5									
6									
7									
8									
9									
10									
11									
12									
13									
14									
15									
16									
17									
18									
19									

⟵ clique size ⟶

⟵ vertices ⟶

$n \setminus r$	2	3	4	5	6	7	8
2	●						
3	●						
4	●						
5	●						
6	●						
7	●						
8	●						
9	●						
10	●						
11	●						
12	●						
13	●						
14	●						
15	●						
16	●						
17	●						
18	●						
19	●						

Empty graphs



$n \setminus r$	2	3	4	5	6	7	8
2	🟢						
3	🔵						
4	🟢						
5	🟢	🔵					
6	🟢						
7	🟢		🔵				
8	🟢						
9	🟢			🔵			
10	🟢						
11	🟢				🔵		
12	🟢						
13	🟢					🔵	
14	🟢						
15	🟢						🔵
16	🟢						
17	🟢						
18	🟢						
19	🟢						

🟢 Empty graphs

🔵 Cycle complements

⟵ clique size ⟶

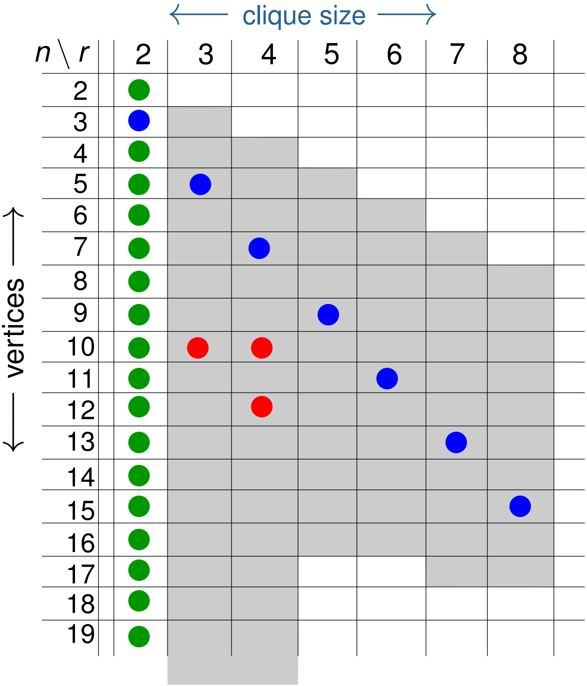
⟵ vertices ⟶

$n \setminus r$	2	3	4	5	6	7	8
2	🟢						
3	🔵						
4	🟢						
5	🟢	🔵					
6	🟢						
7	🟢		🔵				
8	🟢						
9	🟢			🔵			
10	🟢	🔴	🔴				
11	🟢				🔵		
12	🟢		🔴				
13	🟢					🔵	
14	🟢						
15	🟢						🔵
16	🟢						
17	🟢						
18	🟢						
19	🟢						

Empty graphs

Cycle complements

Old examples

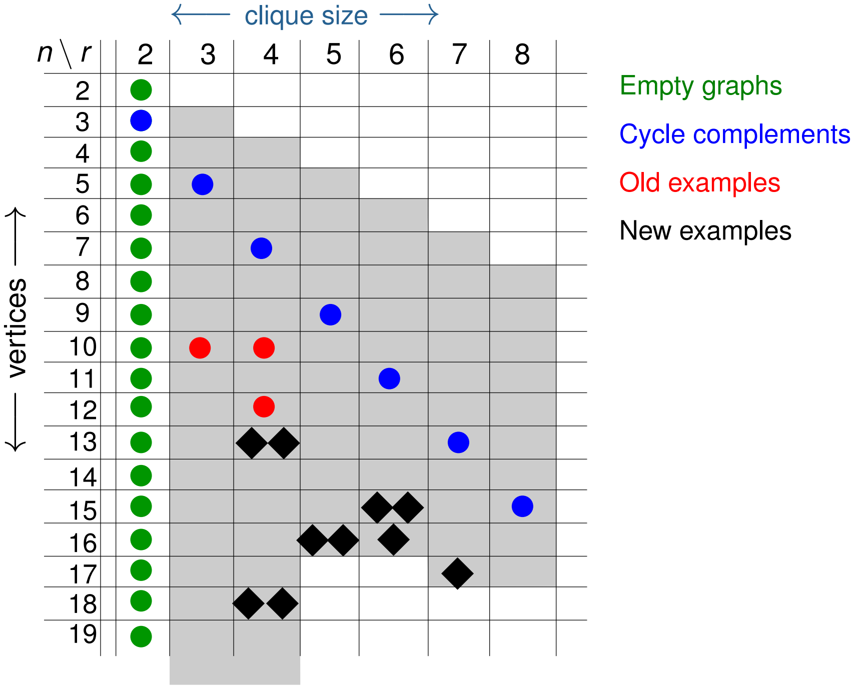


n \ r	2	3	4	5	6	7	8
2	🟢						
3	🔵	▨					
4	🟢	▨	▨				
5	🟢	▨ 🔵	▨	▨			
6	🟢	▨	▨	▨	▨		
7	🟢	▨	▨ 🔵	▨	▨	▨	
8	🟢	▨	▨	▨	▨	▨	▨
9	🟢	▨	▨	▨ 🔵	▨	▨	▨
10	🟢	▨ 🔴	▨ 🔴	▨	▨	▨	▨
11	🟢	▨	▨	▨	▨ 🔵	▨	▨
12	🟢	▨	▨ 🔴	▨	▨	▨	▨
13	🟢	▨	▨	▨	▨	▨ 🔵	▨
14	🟢	▨	▨	▨	▨	▨	▨
15	🟢	▨	▨	▨	▨	▨	▨ 🔵
16	🟢	▨	▨	▨	▨	▨	▨
17	🟢	▨	▨			▨	▨
18	🟢	▨	▨				
19	🟢	▨	▨				

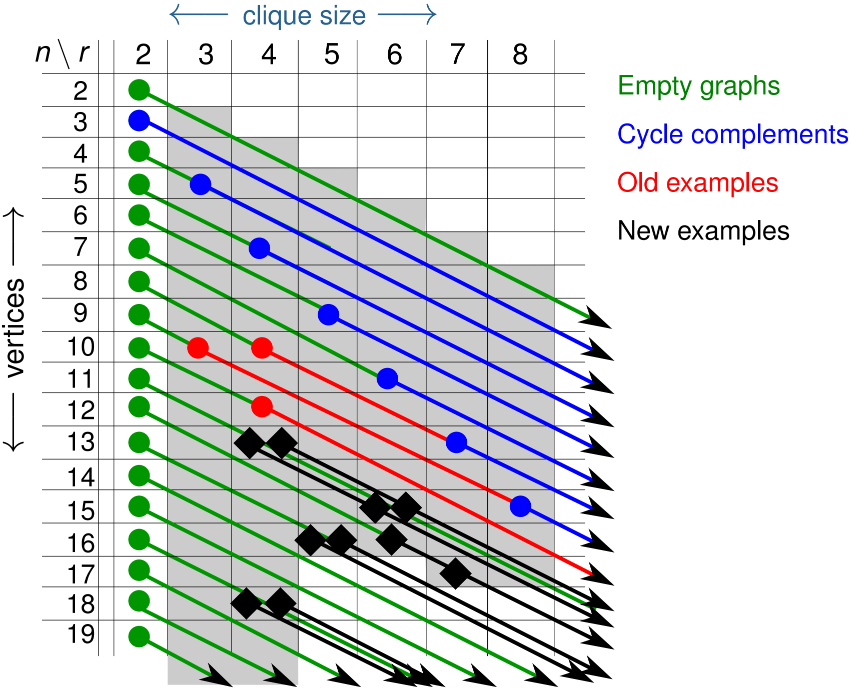
Empty graphs

Cycle complements

Old examples

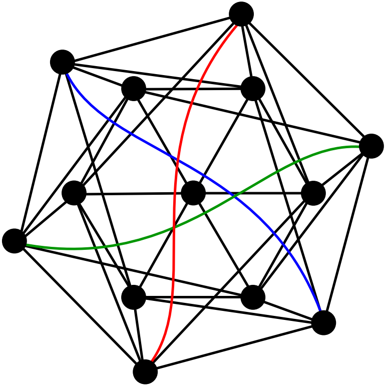


$n \setminus r$	2	3	4	5	6	7	8
2	🟢						
3	🔵						
4	🟢						
5	🟢	🔵					
6	🟢						
7	🟢		🔵				
8	🟢						
9	🟢			🔵			
10	🟢	🔴	🔴				
11	🟢				🔵		
12	🟢		🔴				
13	🟢		◆◆			🔵	
14	🟢						
15	🟢				◆◆		🔵
16	🟢			◆◆	◆		
17	🟢					◆	
18	🟢		◆◆				
19	🟢						

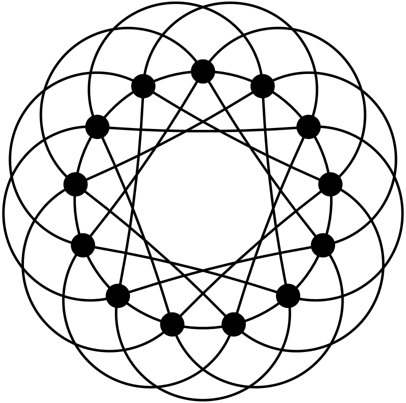

clique size
n \ r
2
3
4
5
6
7
8
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
vertices
Empty graphs
Cycle complements
Old examples
New examples

4-Primitive Graphs

$n = 13$



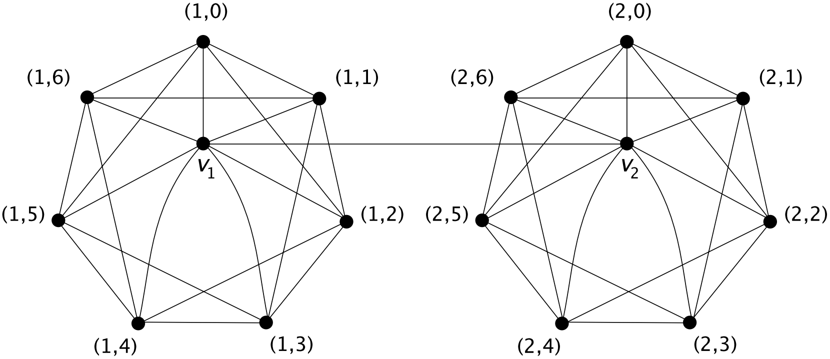
$G_{13}^{(A)}$



Paley(13)

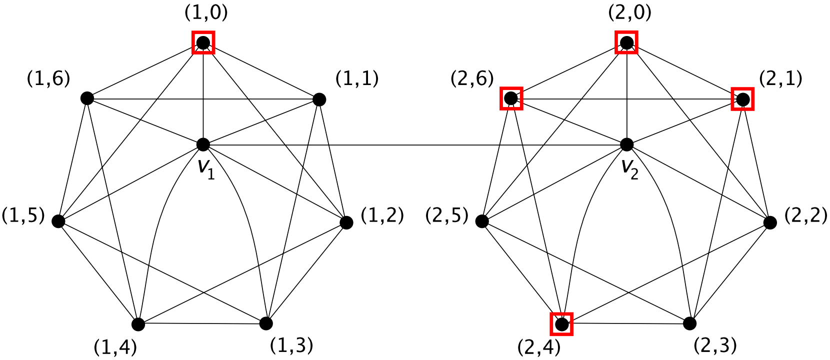
5-Primitive Graph

$n = 16 : G_{16}^{(A)}$



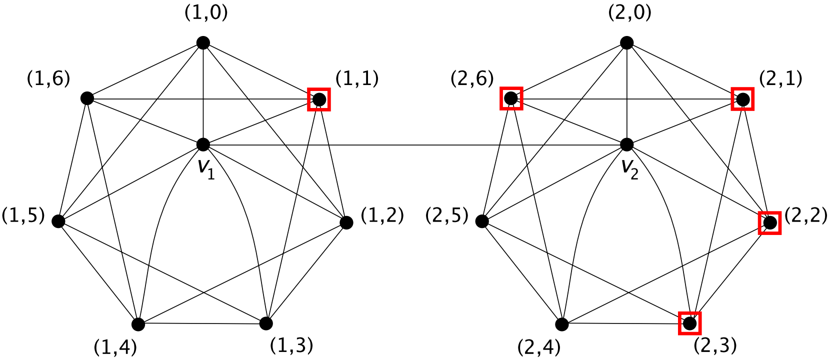
5-Primitive Graph

$n = 16 : G_{16}^{(A)}$



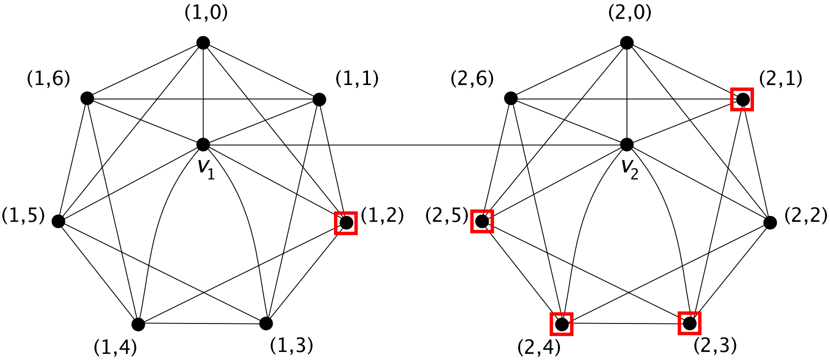
5-Primitive Graph

$n = 16 : G_{16}^{(A)}$



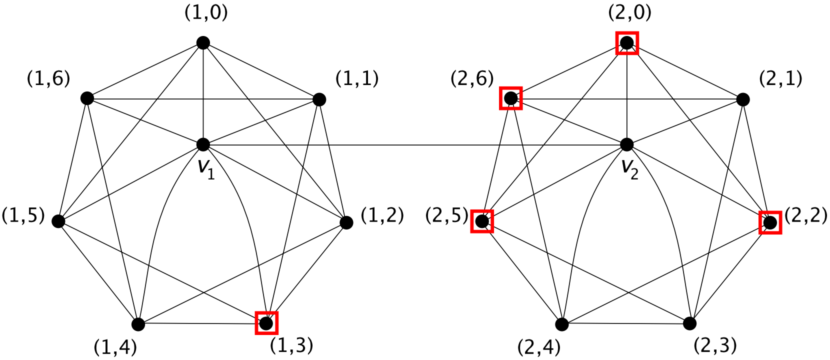
5-Primitive Graph

$n = 16 : G_{16}^{(A)}$



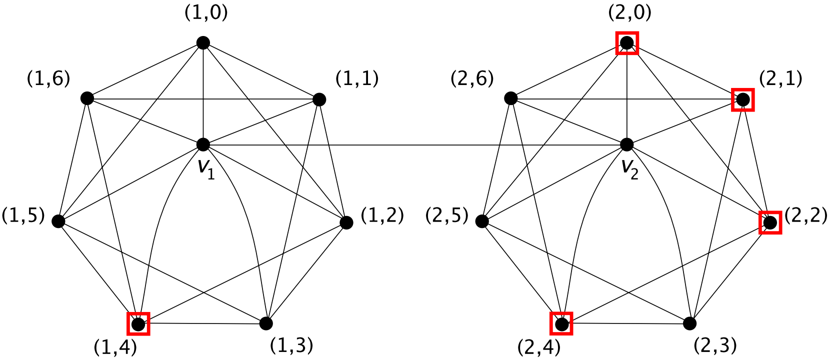
5-Primitive Graph

$n = 16 : G_{16}^{(A)}$



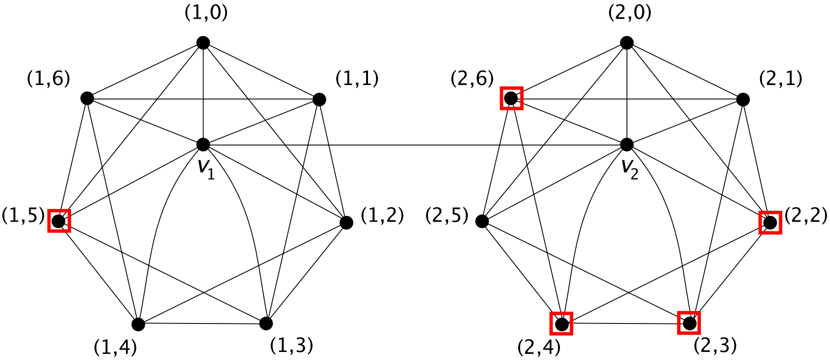
5-Primitive Graph

$n = 16 : G_{16}^{(A)}$



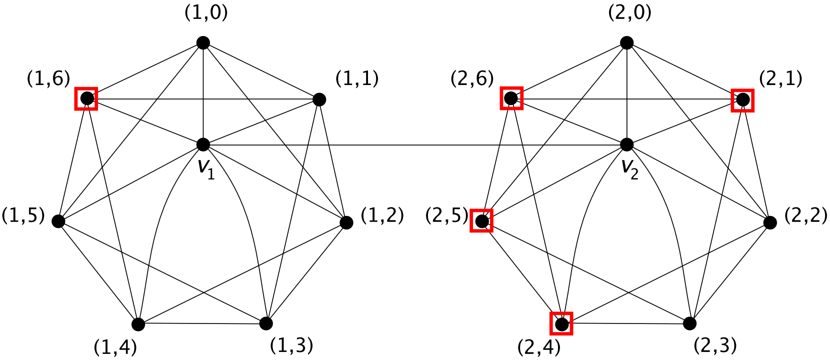
5-Primitive Graph

$n = 16 : G_{16}^{(A)}$



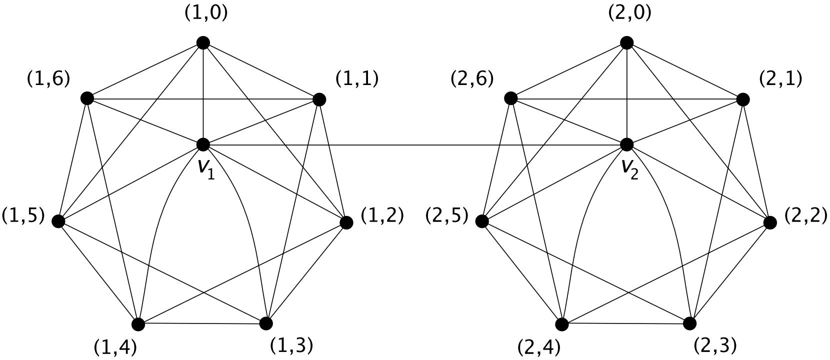
5-Primitive Graph

$n = 16 : G_{16}^{(A)}$



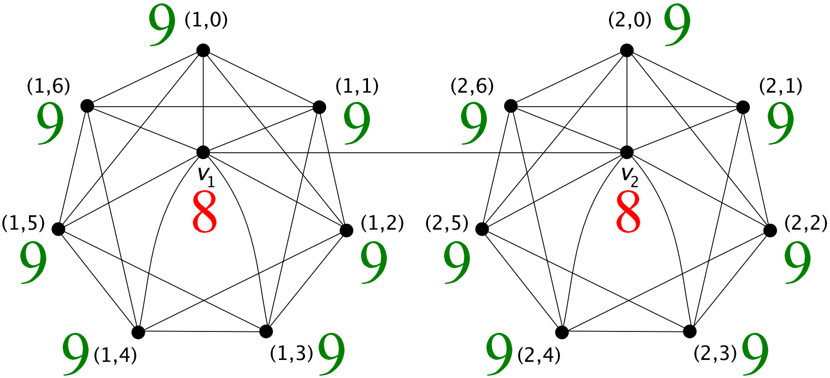
5-Primitive Graph

$n = 16 : G_{16}^{(A)}$



5-Primitive Graph

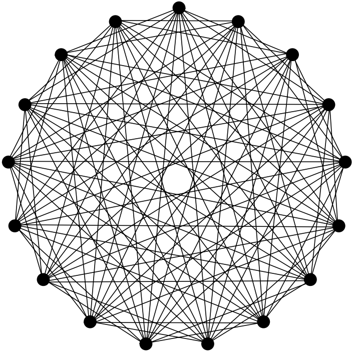
$n = 16 : G_{16}^{(A)}$



Not all *r*-primitive graphs are regular!

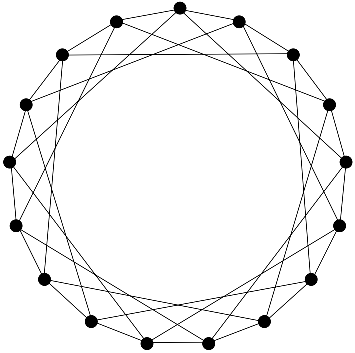
7-Primitive Graph

$n = 17 : G_{17}^{(A)}$



7-Primitive Graph

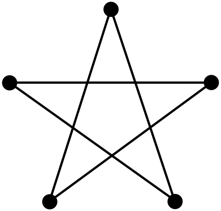
$n = 17 : G_{17}^{(A)}$



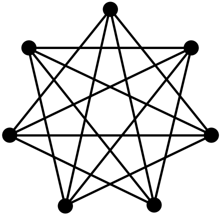
The **Cayley complement** $\overline{C}(\mathbb{Z}_n, S)$ has vertex set $\{0, 1, \ldots, n-1\}$ and an edge *ij* if and only if $|i-j| \pmod{n} \notin S$.

The **Cayley complement** $\overline{C}(\mathbb{Z}_n, S)$ has vertex set $\{0, 1, \ldots, n-1\}$ and an edge *ij* if and only if $|i-j| \pmod{n} \notin S$.

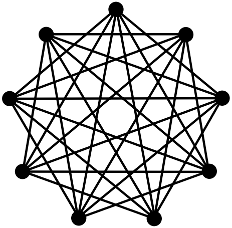
For $r \geq 1$, $\overline{C}(\mathbb{Z}_{2r-1}, \{1\}) \cong \overline{C_{2r-1}}$ is *r*-primitive.



$\overline{C_5}$



$\overline{C_7}$



$\overline{C_9}$

Searching for r-Primitive Cayley Complements

To search for Cayley complements $\overline{C}(\mathbb{Z}_n, S)$ with $|S| = g$:

1. Select a generator set $S = \{a_1 = 1 < a_2 < a_3 < \cdots < a_g\} \subseteq \mathbb{Z}$.

Searching for *r*-Primitive Cayley Complements

To search for Cayley complements $\overline{C}(\mathbb{Z}_n, S)$ with $|S| = g$:

1. Select a generator set $S = \{a_1 = 1 < a_2 < a_3 < \cdots < a_g\} \subseteq \mathbb{Z}$.
2. Select an integer $n > 2a_g$.

Searching for r-Primitive Cayley Complements

To search for Cayley complements $\overline{C}(\mathbb{Z}_n, S)$ with $|S| = g$:

1. Select a generator set $S = \{a_1 = 1 < a_2 < a_3 < \cdots < a_g\} \subseteq \mathbb{Z}$.
2. Select an integer $n > 2a_g$.
3. Compute $r = \omega(\overline{C}(\mathbb{Z}_n, S)) + 1$.

Searching for r-Primitive Cayley Complements

To search for Cayley complements $\overline{C}(\mathbb{Z}_n, S)$ with $|S| = g$:

1. Select a generator set $S = \{a_1 = 1 < a_2 < a_3 < \cdots < a_g\} \subseteq \mathbb{Z}$.

2. Select an integer $n > 2a_g$.

3. Compute $r = \omega(\overline{C}(\mathbb{Z}_n, S)) + 1$.

4. Check if $\overline{C}(\mathbb{Z}_n, S) + \{0, a_i\}$ has a unique r-clique for all $a_i \in S$.

Searching for r-Primitive Cayley Complements

To search for Cayley complements $\overline{C}(\mathbb{Z}_n, S)$ with $|S| = g$:

1. Select a generator set $S = \{a_1 = 1 < a_2 < a_3 < \cdots < a_g\} \subseteq \mathbb{Z}$.
2. Select an integer $n > 2a_g$.
3. Compute $r = \omega(\overline{C}(\mathbb{Z}_n, S)) + 1$.
4. Check if $\overline{C}(\mathbb{Z}_n, S) + \{0, a_i\}$ has a unique r-clique for all $a_i \in S$.

Used Niskanen and Östergård's *cliquer* software to compute $\omega(G)$.

Two or Three Generators

S	r	n
$\{1,4\}$	7	17
$\{1,6\}$	16	37
$\{1,8\}$	29	65
$\{1,10\}$	46	101
$\{1,12\}$	67	145

$g = 2$

S	r	n
$\{1,5,6\}$	9	31
$\{1,8,9\}$	22	73
$\{1,11,12\}$	41	133
$\{1,14,15\}$	66	211
$\{1,17,18\}$	97	307

$g = 3$

Infinite Families

Conjecture (Hartke, Stolee, 2012) Let $t \geq 1$,

$$n = 4t^2 + 1, \qquad \text{and} \qquad r = 2t^2 - t + 1.$$

The Cayley complement $\overline{C}(\mathbb{Z}_n, \{1, 2t\})$ is r-primitive.

Conjecture (Hartke, Stolee, 2012) Let $t \geq 1$,

$$n = 9t^2 - 3t + 1 \qquad \text{and} \qquad r = 3t^2 - 2t + 1.$$

The Cayley complement $\overline{C}(\mathbb{Z}_n, \{1, 3t-1, 3t\})$ is r-primitive.

Infinite Families

Theorem (Hartke, Stolee, 2012) Let $t \geq 1$,

$$n = 4t^2 + 1, \qquad \text{and} \qquad r = 2t^2 - t + 1.$$

The Cayley complement $\overline{C}(\mathbb{Z}_n, \{1, 2t\})$ is r-primitive.

Proof uses **counting** method.

Conjecture (Hartke, Stolee, 2012) Let $t \geq 1$,

$$n = 9t^2 - 3t + 1 \qquad \text{and} \qquad r = 3t^2 - 2t + 1.$$

The Cayley complement $\overline{C}(\mathbb{Z}_n, \{1, 3t-1, 3t\})$ is r-primitive.

Infinite Families

Theorem (Hartke, Stolee, 2012) Let $t \geq 1$,

$$n = 4t^2 + 1, \qquad \text{and} \qquad r = 2t^2 - t + 1.$$

The Cayley complement $\overline{C}(\mathbb{Z}_n, \{1, 2t\})$ is r-primitive.

Proof uses **counting** method.

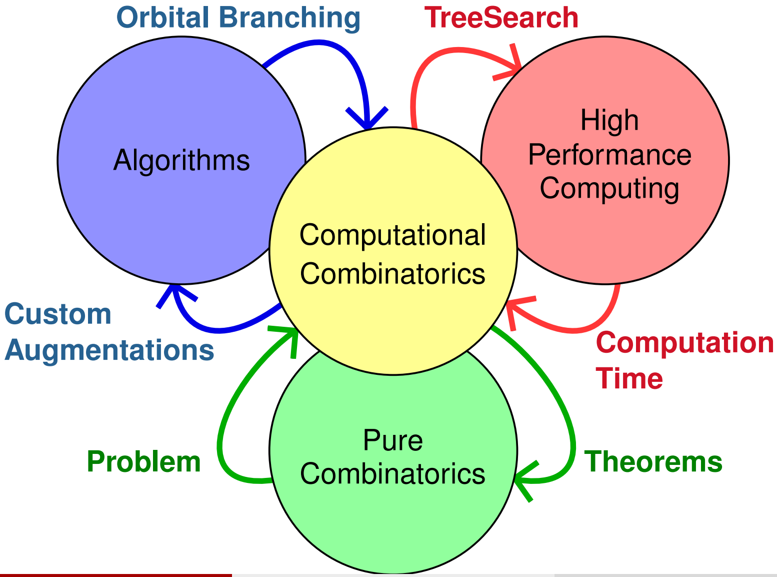
Theorem (Hartke, Stolee, 2012) Let $t \geq 1$,

$$n = 9t^2 - 3t + 1 \qquad \text{and} \qquad r = 3t^2 - 2t + 1.$$

The Cayley complement $\overline{C}(\mathbb{Z}_n, \{1, 3t-1, 3t\})$ is r-primitive.

Proof uses **discharging** method.

Computational Combinatorics



What Next?

Technique-specific

1. **Orbital Branching:** Formalize custom augmentations for arbitrary constraint systems. Apply to problems like strongly regular graphs.

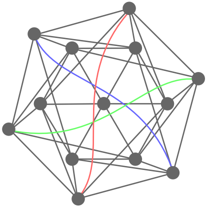
2. **Discharging:** Automate process so computer can discover and write proofs.

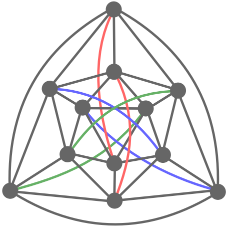
3. **More Techniques:** Find, Adapt, or Develop.

Computational Combinatorics and the search for Uniquely K_r-Saturated Graphs

Stephen G. Hartke

University of Nebraska–Lincoln, USA
hartke@math.unl.edu
http://www.math.unl.edu/∼shartke2/





Joint work with Stephen G. Hartke, appeared in *Electronic Journal of Combinatorics*, 2012.