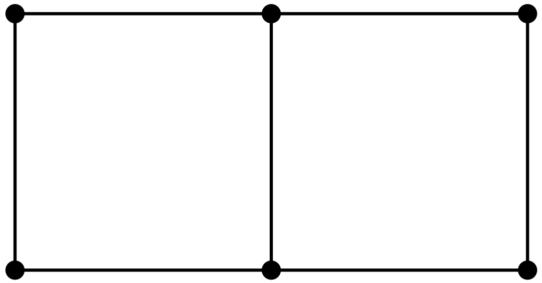
Minimal Dominating Sets in Trees

Counting and Extremal Results (and Enumeration)

Günter Rote
Freie Universität Berlin

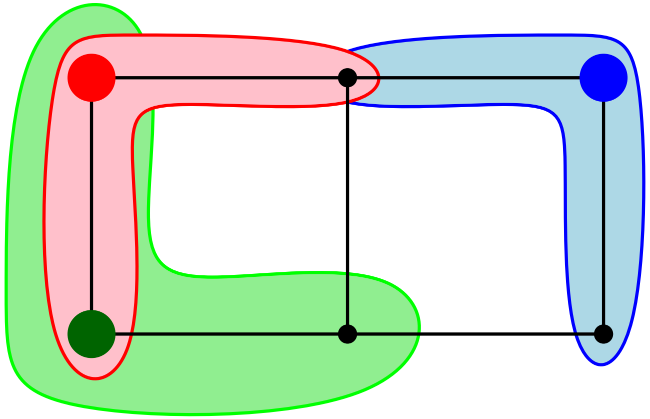


Dominating set D:
Every vertex $v \notin D$ must have a neighbor in D.

Minimal Dominating Sets in Trees

Counting and Extremal Results (and Enumeration)

Günter Rote
Freie Universität Berlin

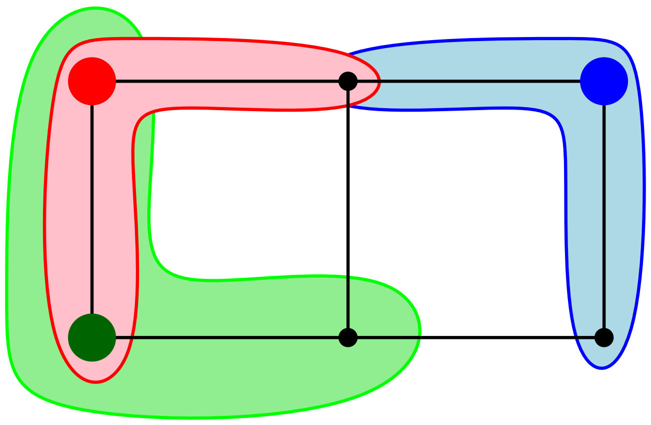


Dominating set D:
Every vertex $v \notin D$ must have a neighbor in D.

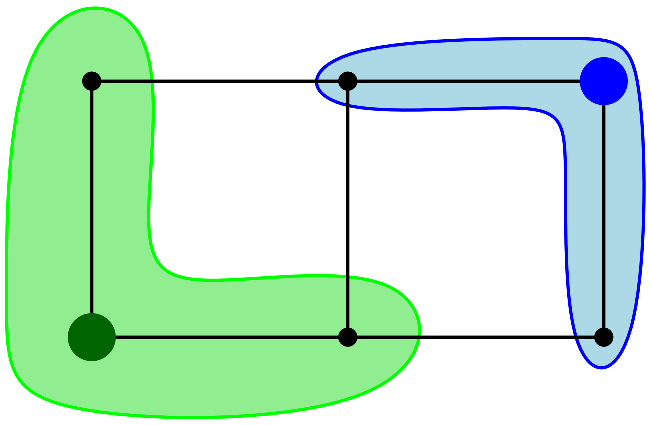
Minimal Dominating Sets in Trees

Counting and Extremal Results (and Enumeration)

Günter Rote
Freie Universität Berlin

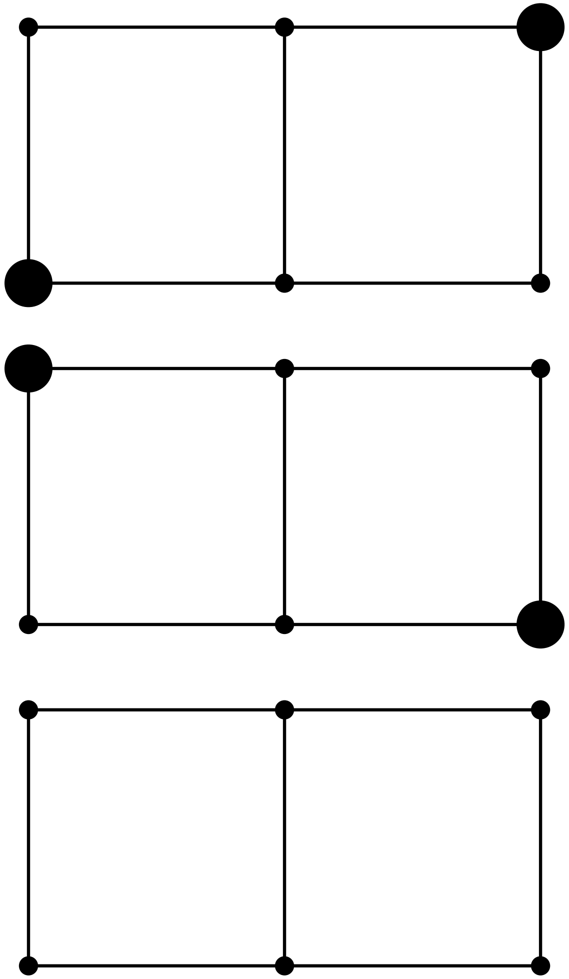


Dominating set D:
Every vertex $v \notin D$ must have a neighbor in D.

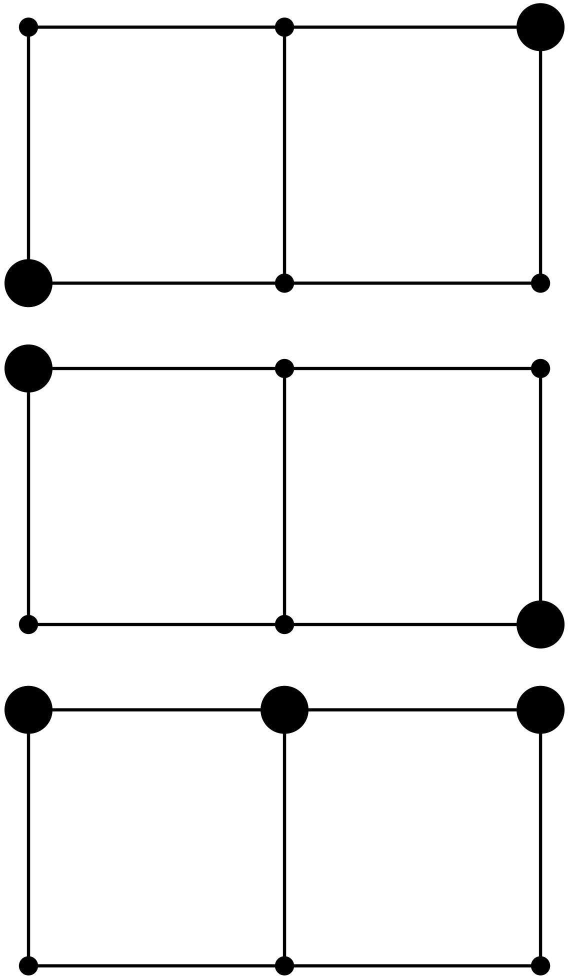


Minimal dominating set D:
No proper subset of D is a dominating set.

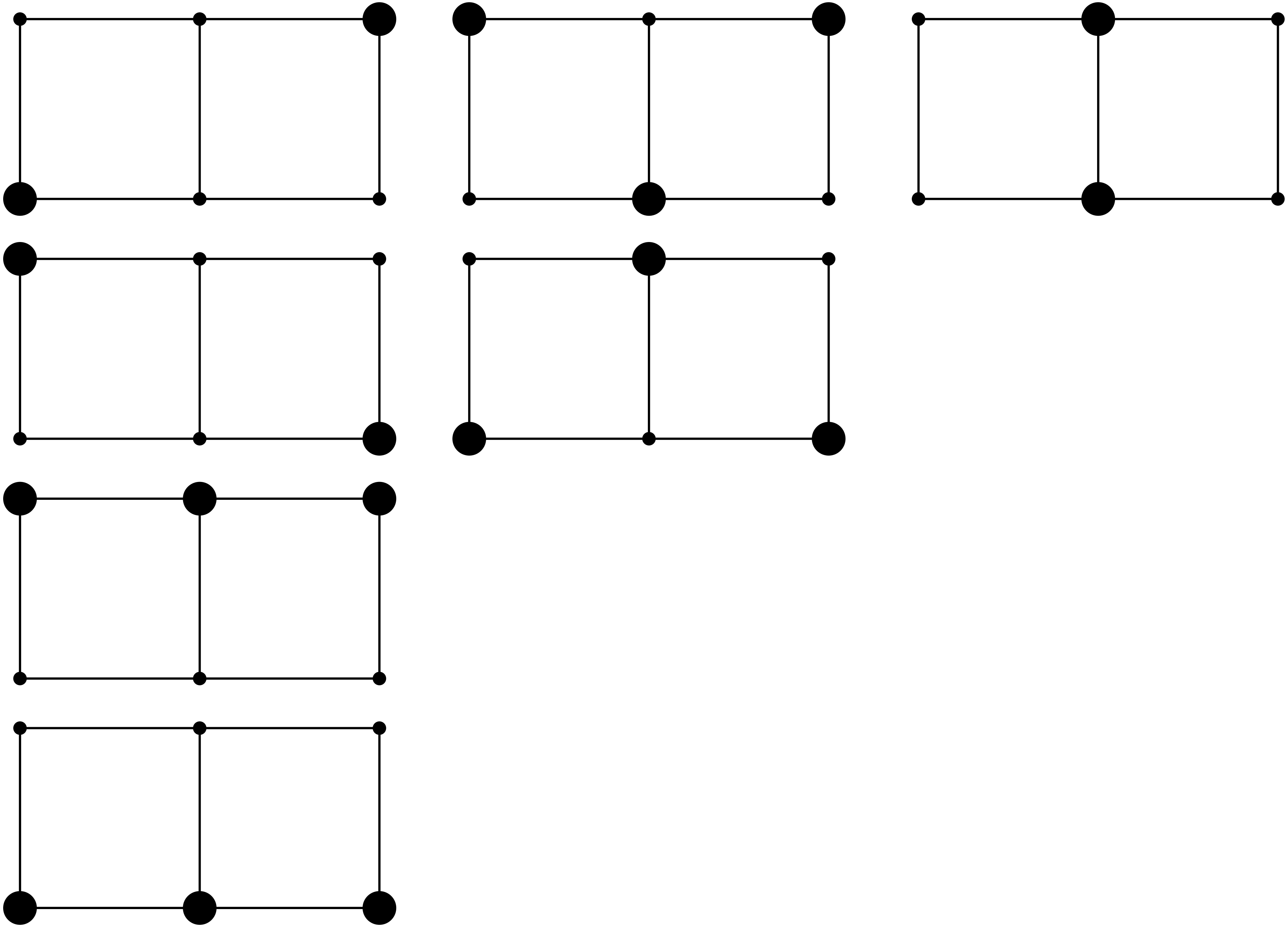
Minimal Dominating Sets



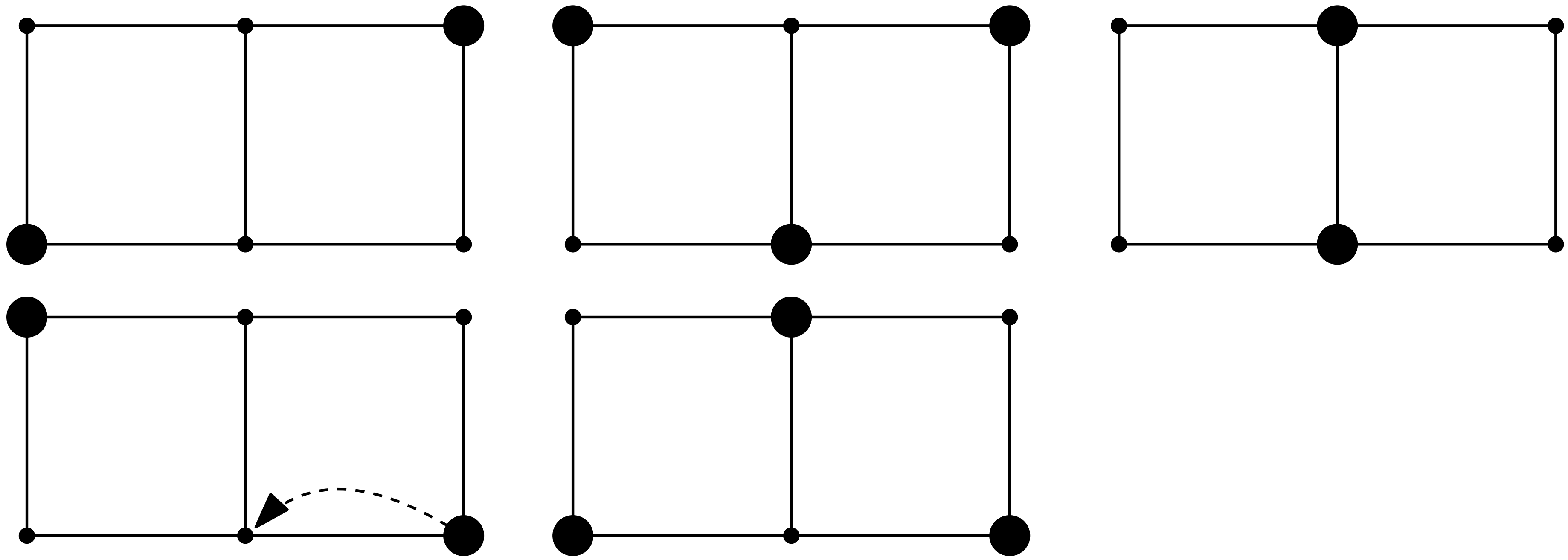
Minimal Dominating Sets

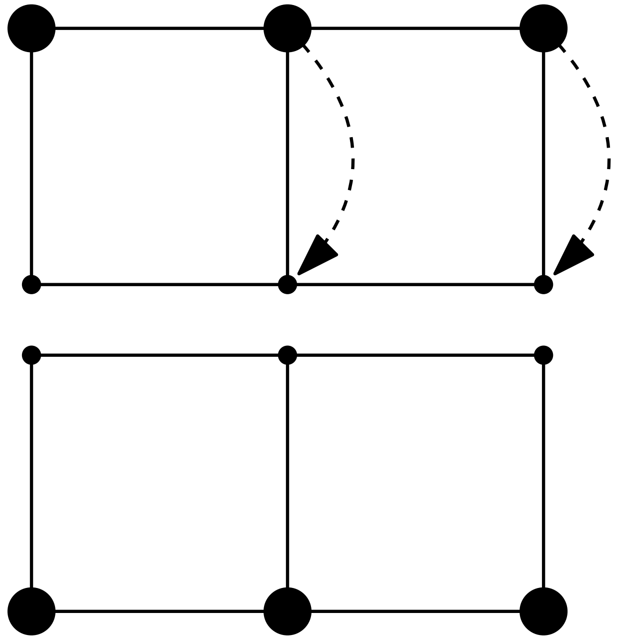


Minimal Dominating Sets



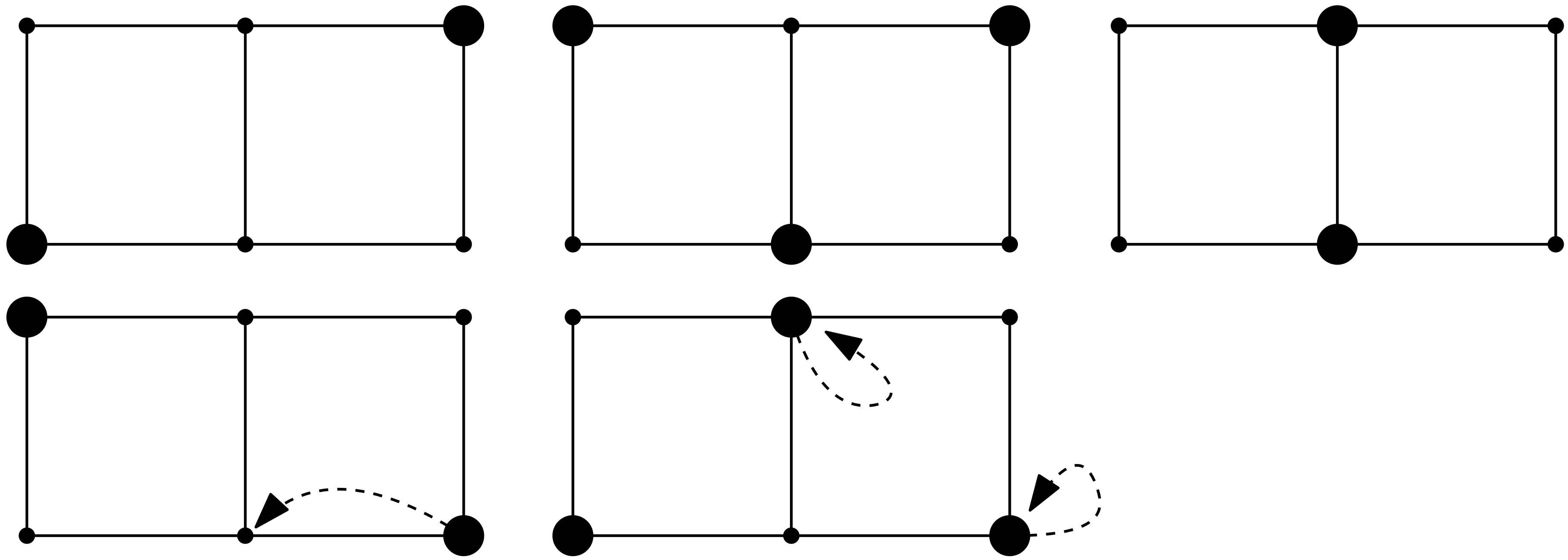
Minimal Dominating Sets

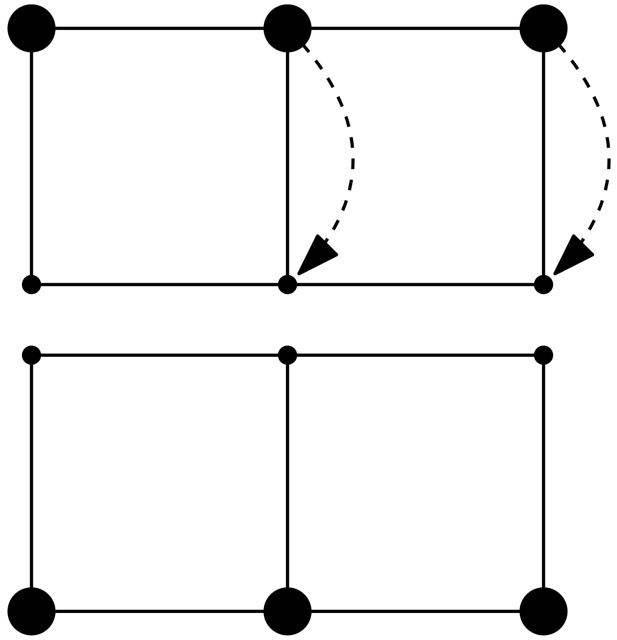




Characterization:
Every vertex $v \in D$ must have a *private neighbor*: adjacent to no other vertex in D.

Minimal Dominating Sets





Characterization:
Every vertex $v \in D$ must have
a *private neighbor*:
adjacent to no other vertex in D.

The private “neighbor” can be v itself.

Minimal Dominating Sets in Trees

How many minimal dominating sets can a tree with n vertices have, at most?

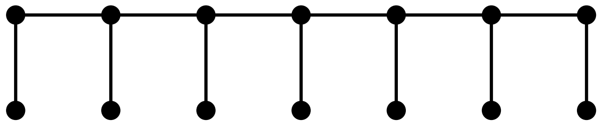
THEOREM:
The number grows like 1.4195^n.

Minimal Dominating Sets in Trees

THEOREM. Let $\lambda = \sqrt[13]{95} \approx 1.4194908$.

1. The maximum number M_n of minimal dominating sets of a tree with n vertices is
   between $0.649748 \cdot \lambda^n$ and $2\lambda^{n-2} < 0.992579 \cdot \lambda^n$.
2. For every n of the form $n = 13k + 1$, there is a tree with at least $95^k > 0.704477 \cdot \lambda^n$ minimal dominating sets.
3. The minimal dominating sets of a tree with n vertices can be enumerated with $O(n)$ setup time and with $O(n)$ delay between successive solutions.

Previous bounds: $\sqrt{2} \approx 1.4142 \le \lambda$



M. Krzywkowski (2013): $\sqrt[27]{12161} \approx 1.416756 \le \lambda \le 1.4656$

P. Golovach, P. Heggernes, M. M. Kanté,
D. Kratsch and Y. Villanger (2015): $\lambda \le \sqrt[3]{3} \approx 1.4422$

Overview

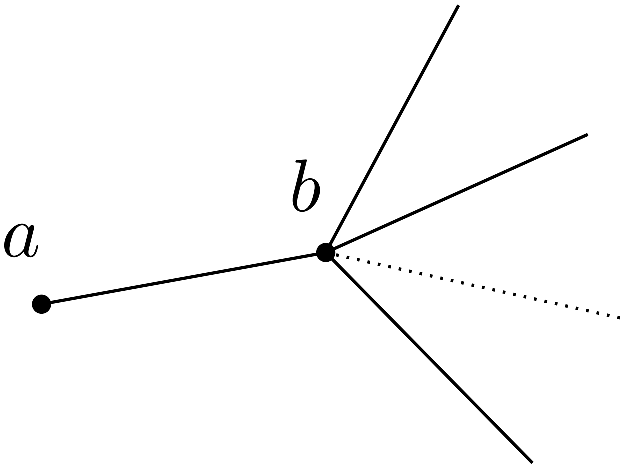
- LOWER BOUND by example
- COUNTING for a particular tree: dynamic programming
- UPPER BOUND: enclosure by a polytope
- ENUMERATION

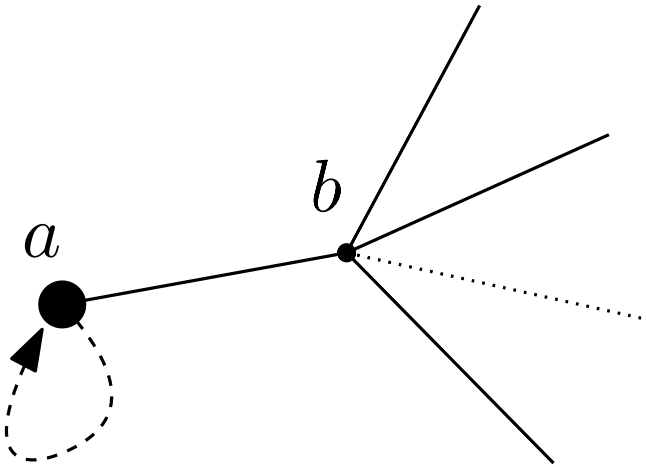
Observation

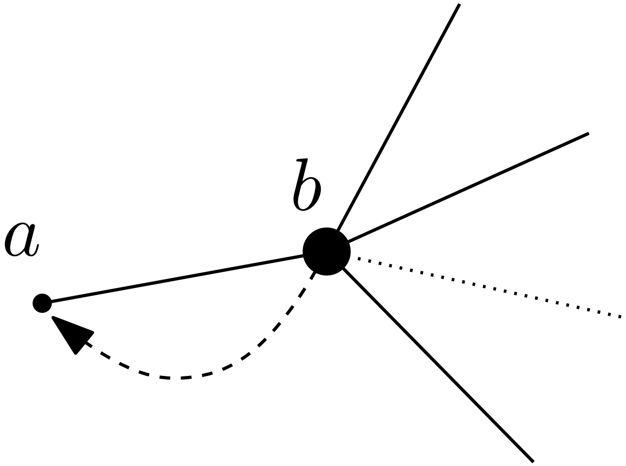
Let a have degree 1 and let b be its neighbor.

THEN: $a \in D$ or $b \in D$ but not both.

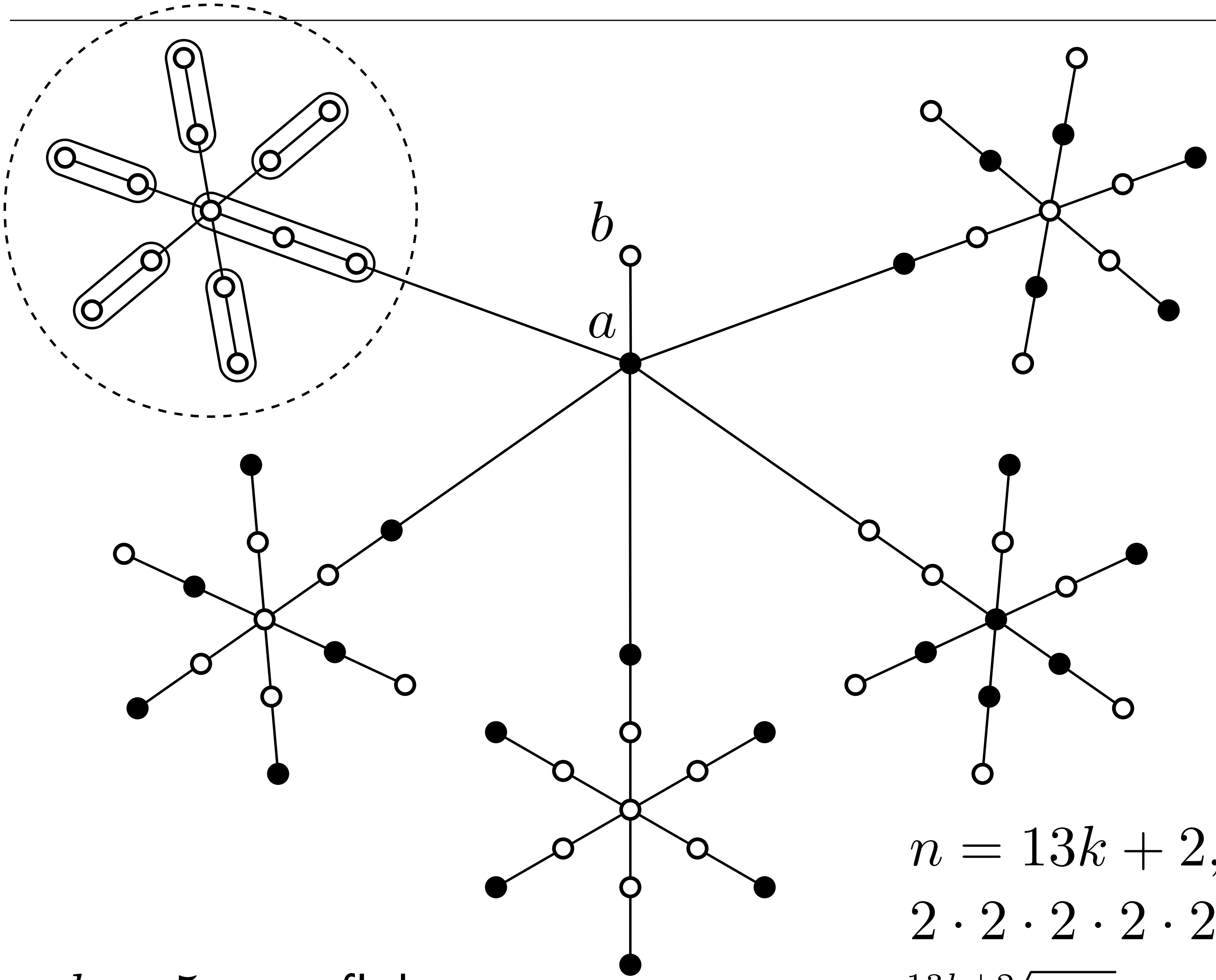
a can always be taken as the private neighbor.







Lower Bound: The Star of Snowflakes



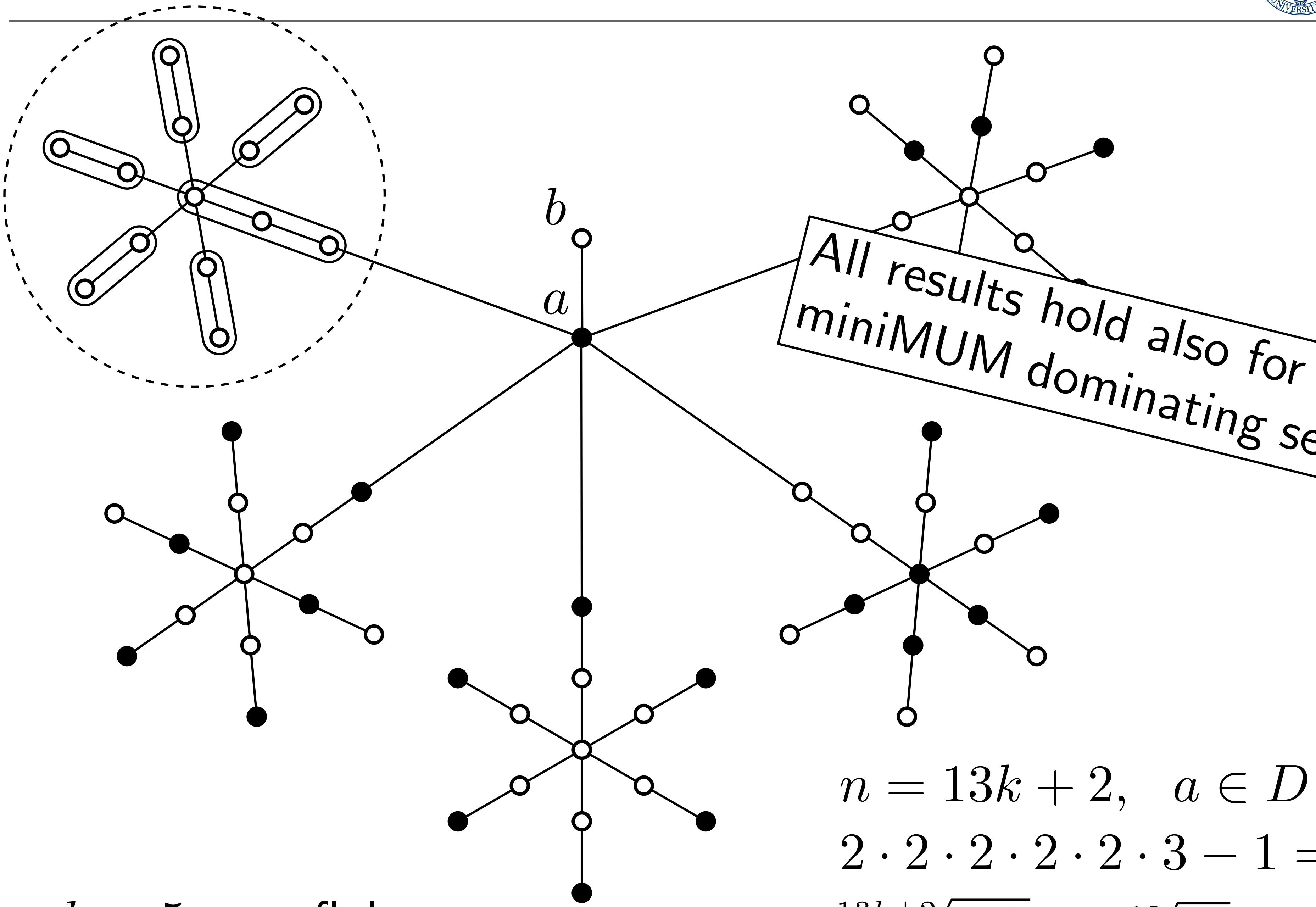
$$n = 13k + 2, \quad a \in D$$

$$2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3 - 1 = 95$$

$$\sqrt[13k+2]{95^k} \to \sqrt[13]{95}$$

$k = 5$ snowflakes

Lower Bound: The Star of Snowflakes



All results hold also for miniMUM dominating sets.

$$n = 13k + 2, \quad a \in D$$

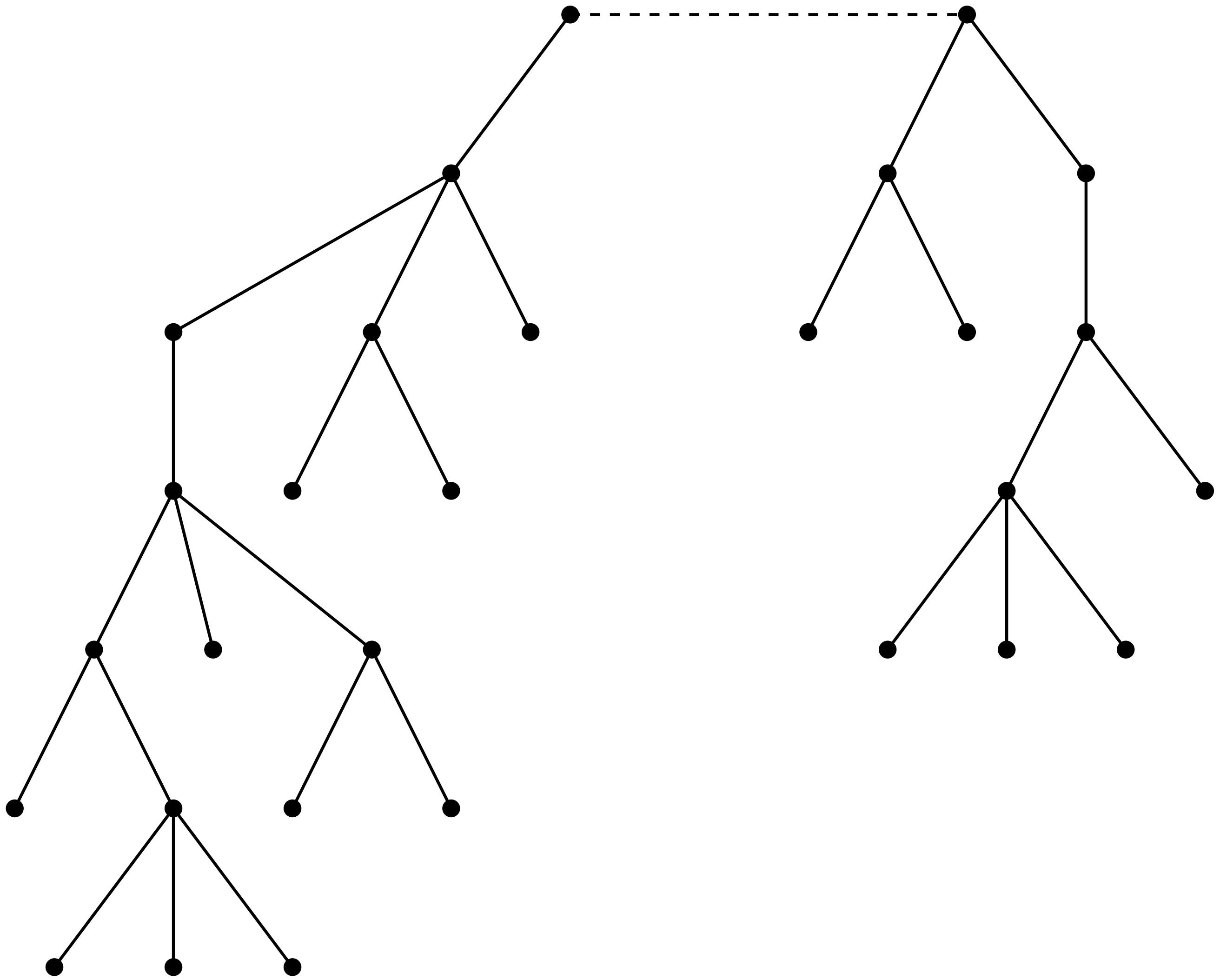
$$2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 \cdot 3 - 1 = 95$$

$$\sqrt[13k+2]{95^k} \to \sqrt[13]{95}$$

$k = 5$ snowflakes

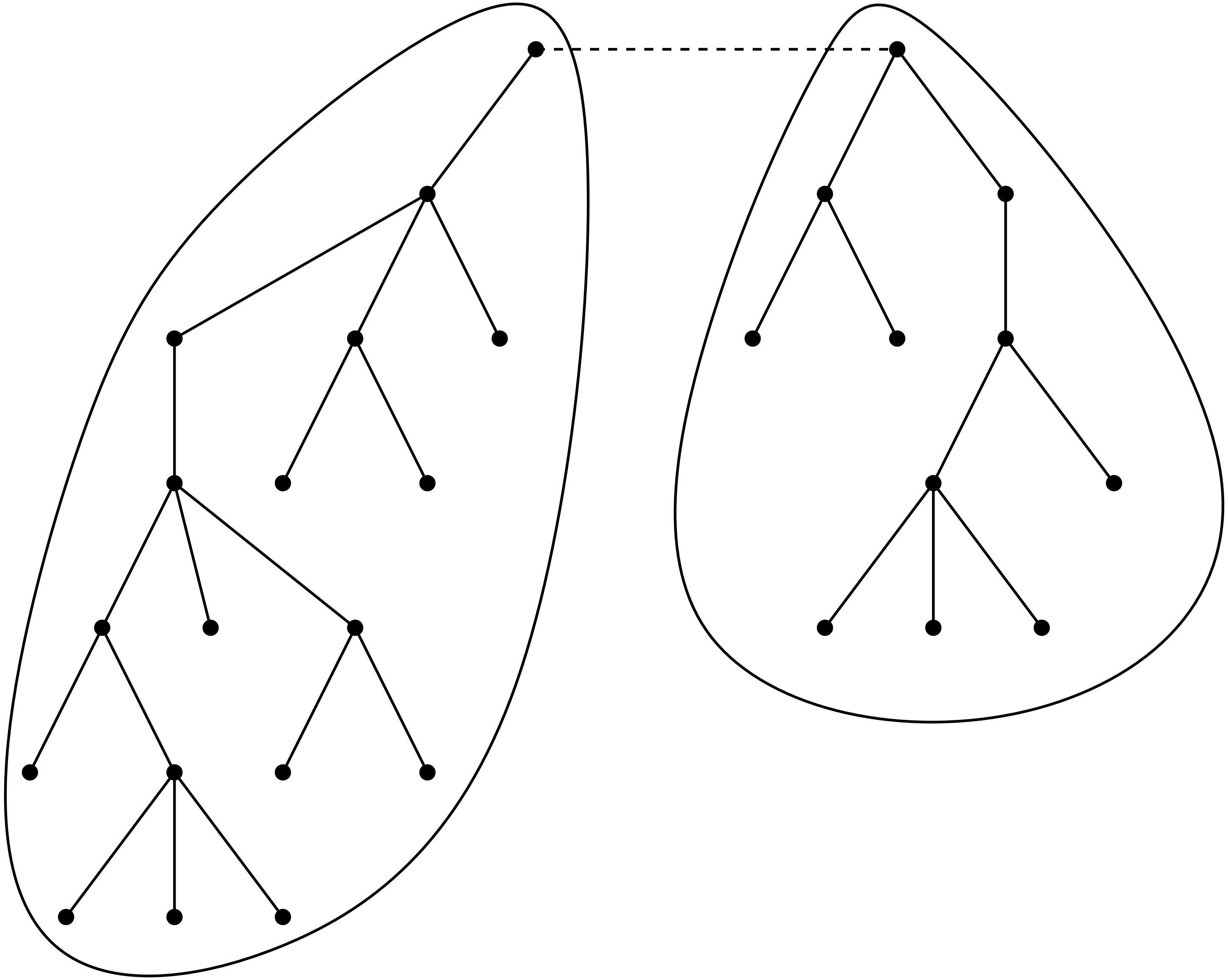
Dynamic Programming

Idea:



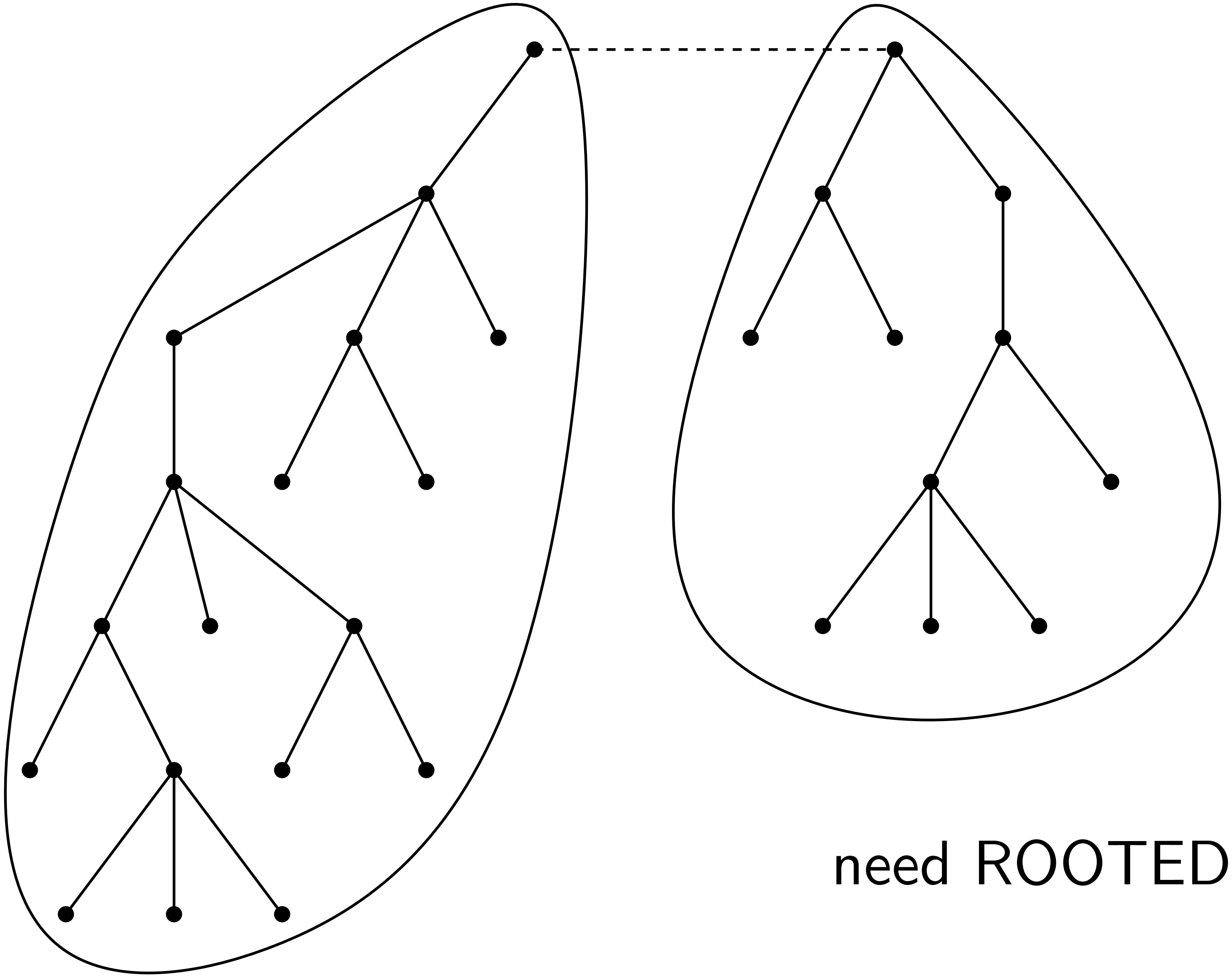
Dynamic Programming

Idea:



Dynamic Programming

Idea:

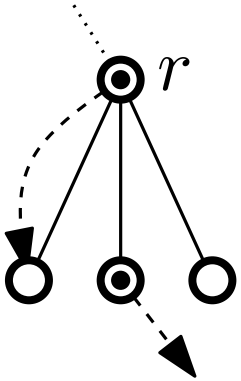


need ROOTED trees!

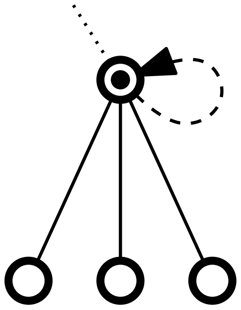
Six Categories of Partial Solutions

root $r \in D$:

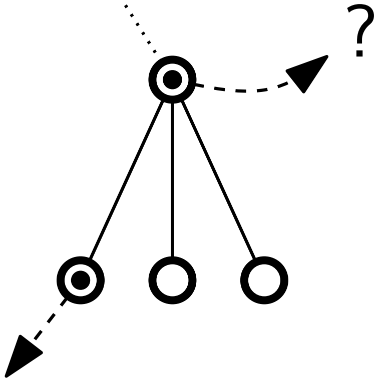
Good



Self

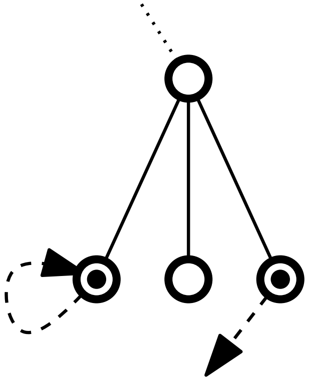


Lacking

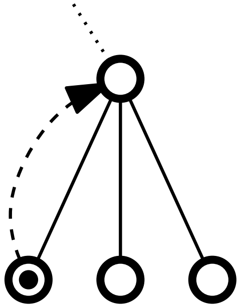


root $r \notin D$:

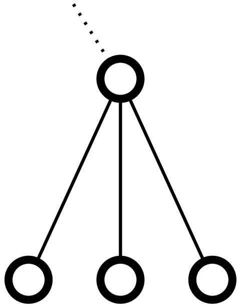
dominated



private

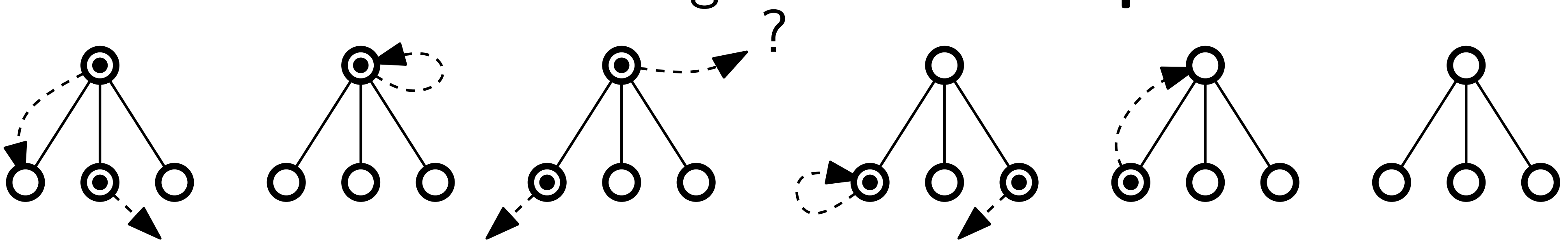


free

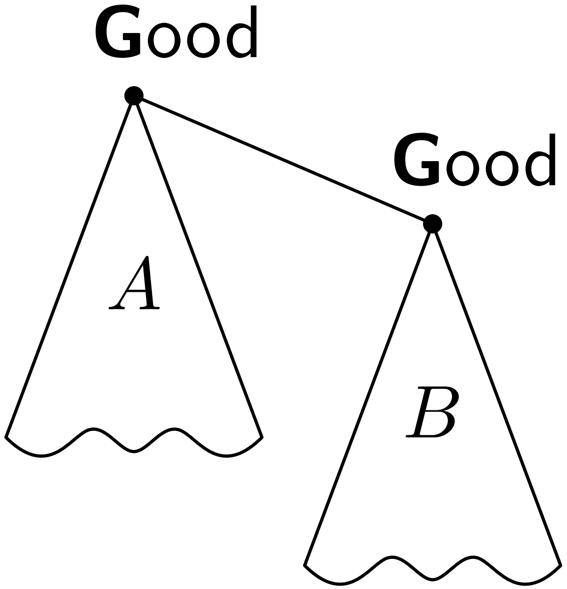


Composition of Partial Solutions

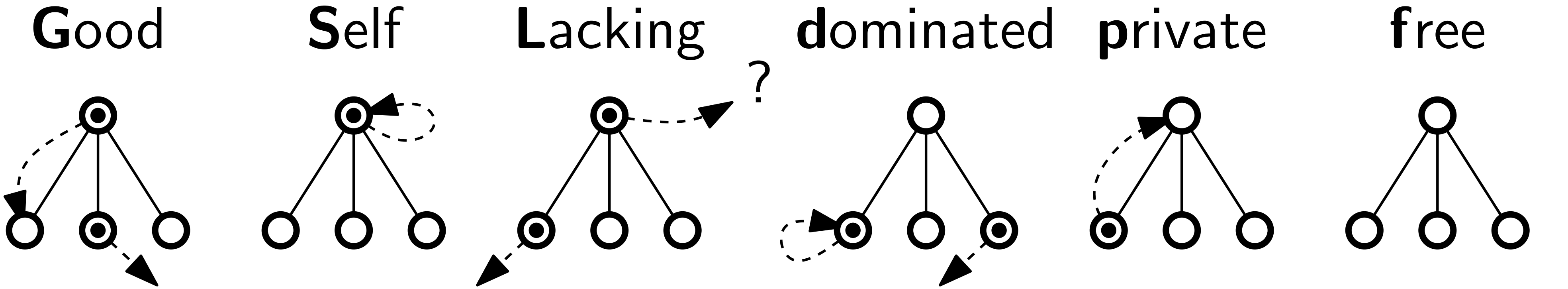
Good **S**elf **L**acking **d**ominated **p**rivate **f**ree



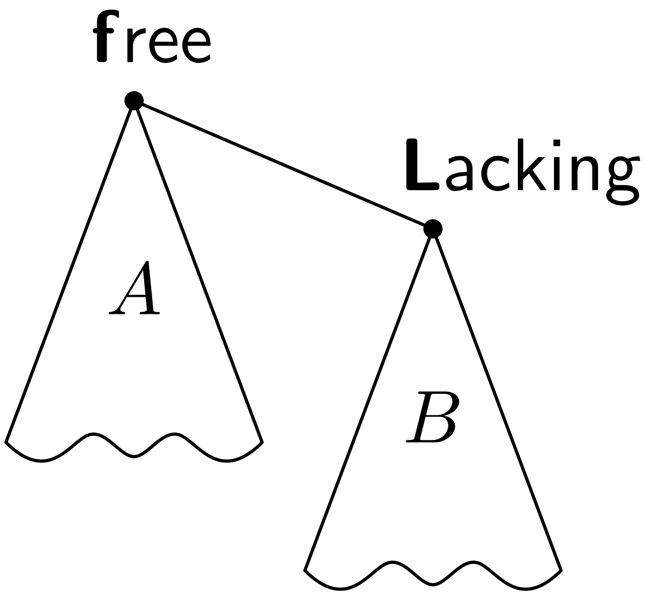
		B					
		G	**S**	**L**	**d**	**p**	**f**
A	**G**	**G**					
	S						
	L						
	d						
	p						
	f						



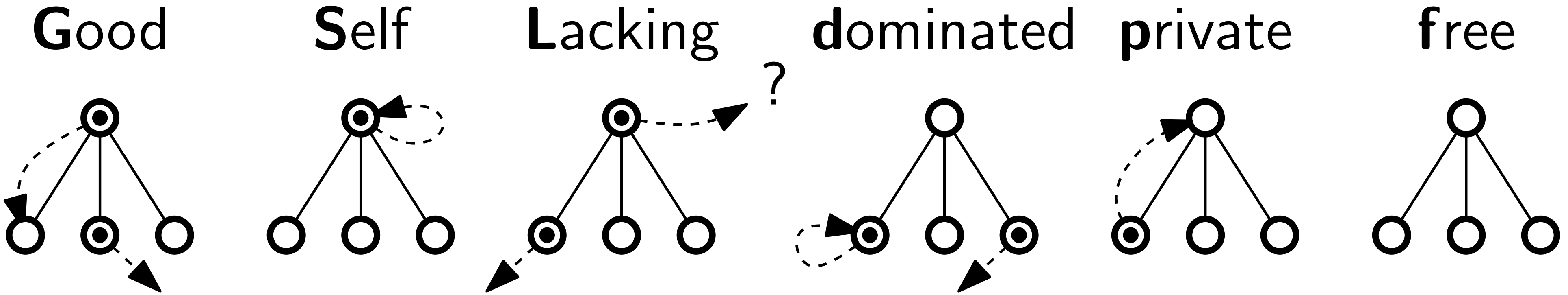
Composition of Partial Solutions



		B					
		G	**S**	**L**	**d**	**p**	**f**
A	**G**	**G**					
	S						
	L						
	d						
	p						
	f			○			

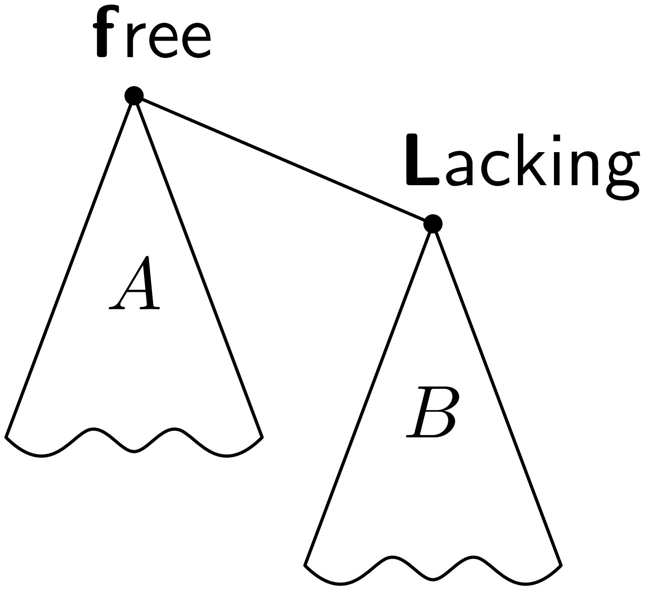


Composition of Partial Solutions

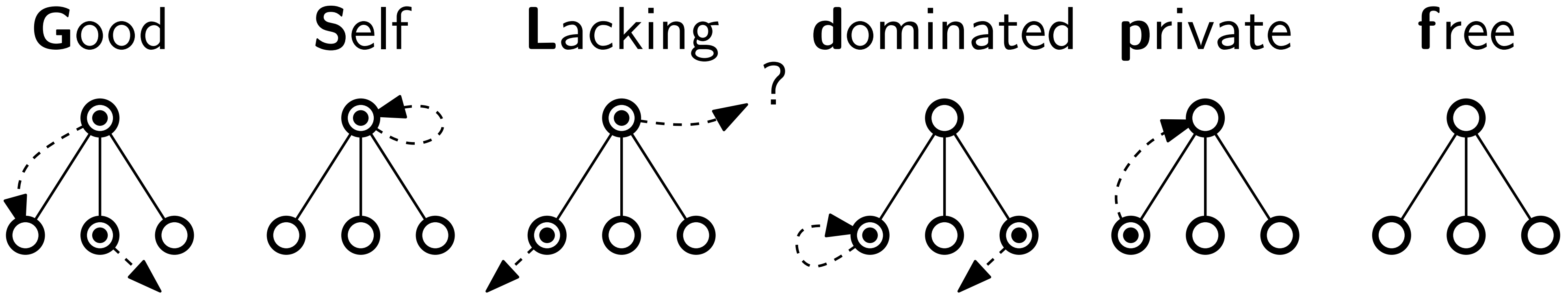


Good **S**elf **L**acking ? **d**ominated **p**rivate **f**ree

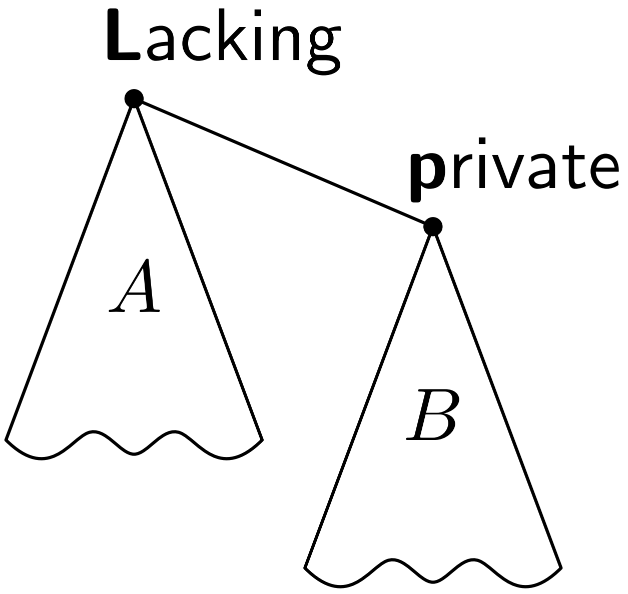
		B					
		G	**S**	**L**	**d**	**p**	**f**
A	**G**	**G**					
	S						
	L						
	d						
	p						
	f			**p**			



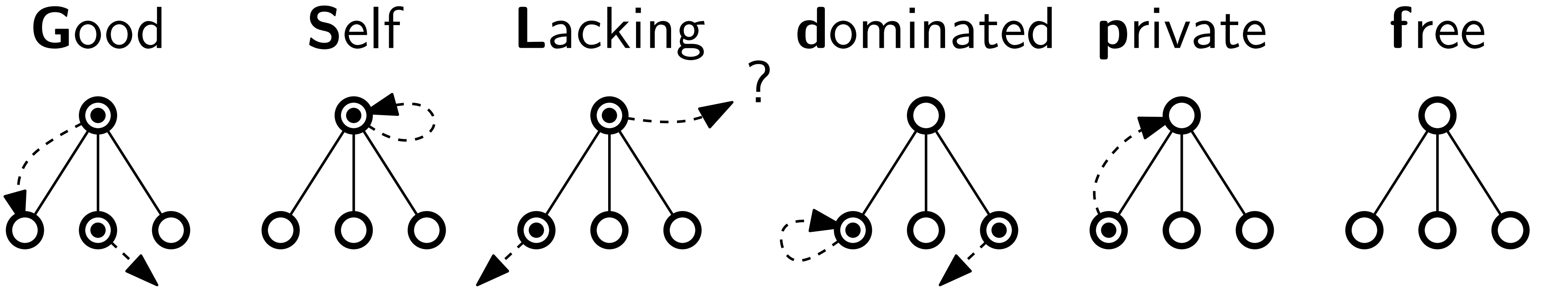
Composition of Partial Solutions



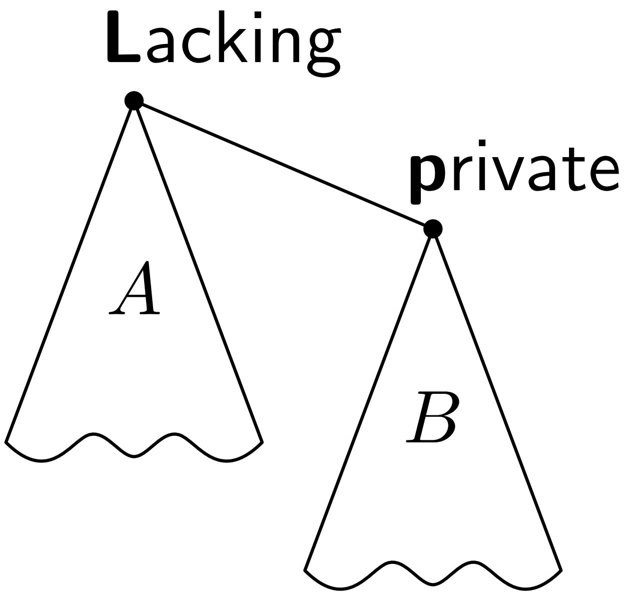
		B					
		G	**S**	**L**	**d**	**p**	**f**
A	**G**	**G**					
	S						
	L					○	
	d						
	p						
	f			**p**			



Composition of Partial Solutions

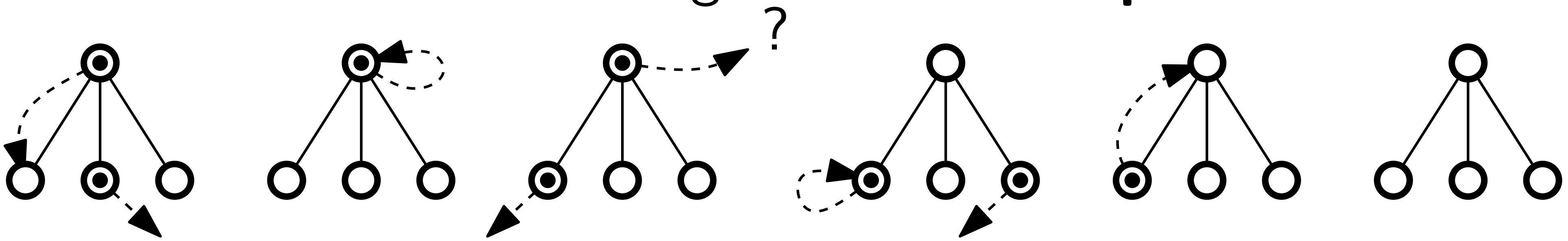


		B					
		G	**S**	**L**	**d**	**p**	**f**
A	**G**	**G**					
	S						
	L					⊖	
	d						
	p						
	f			**p**			

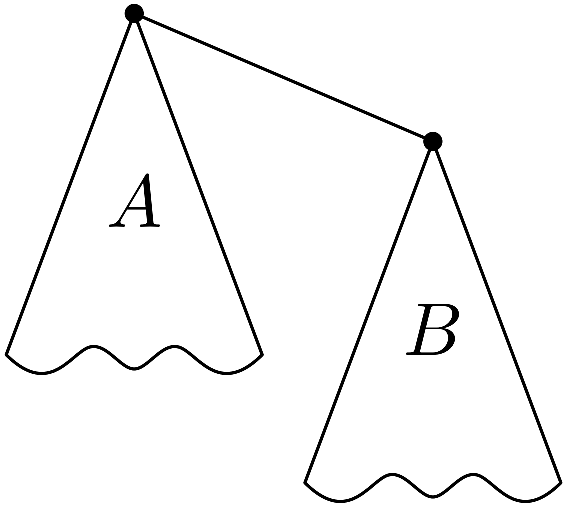


Composition of Partial Solutions

Good **S**elf **L**acking **d**ominated **p**rivate **f**ree



		B					
		G	**S**	**L**	**d**	**p**	**f**
A	**G**	**G**	—	—	**G**	—	**G**
	S	**L**	—	—	**S**	—	**G**
	L	**L**	—	—	**L**	—	**G**
	d	**d**	**d**	—	**d**	**d**	—
	p	—	—	—	**p**	**p**	—
	f	**d**	**d**	**p**	**f**	**f**	—



Dynamic Programming Recursion

		B					
		G	**S**	**L**	**d**	**p**	**f**
A	**G**	**G**	—	—	**G**	—	**G**
	S	**L**	—	—	**S**	—	**G**
	L	**L**	—	—	**L**	—	**G**
	d	**d**	**d**	—	**d**	**d**	—
	p	—	—	—	**p**	**p**	—
	f	**d**	**d**	**p**	**f**	**f**	—

Associate a vector

$$(G, S, L, d, p, f)$$

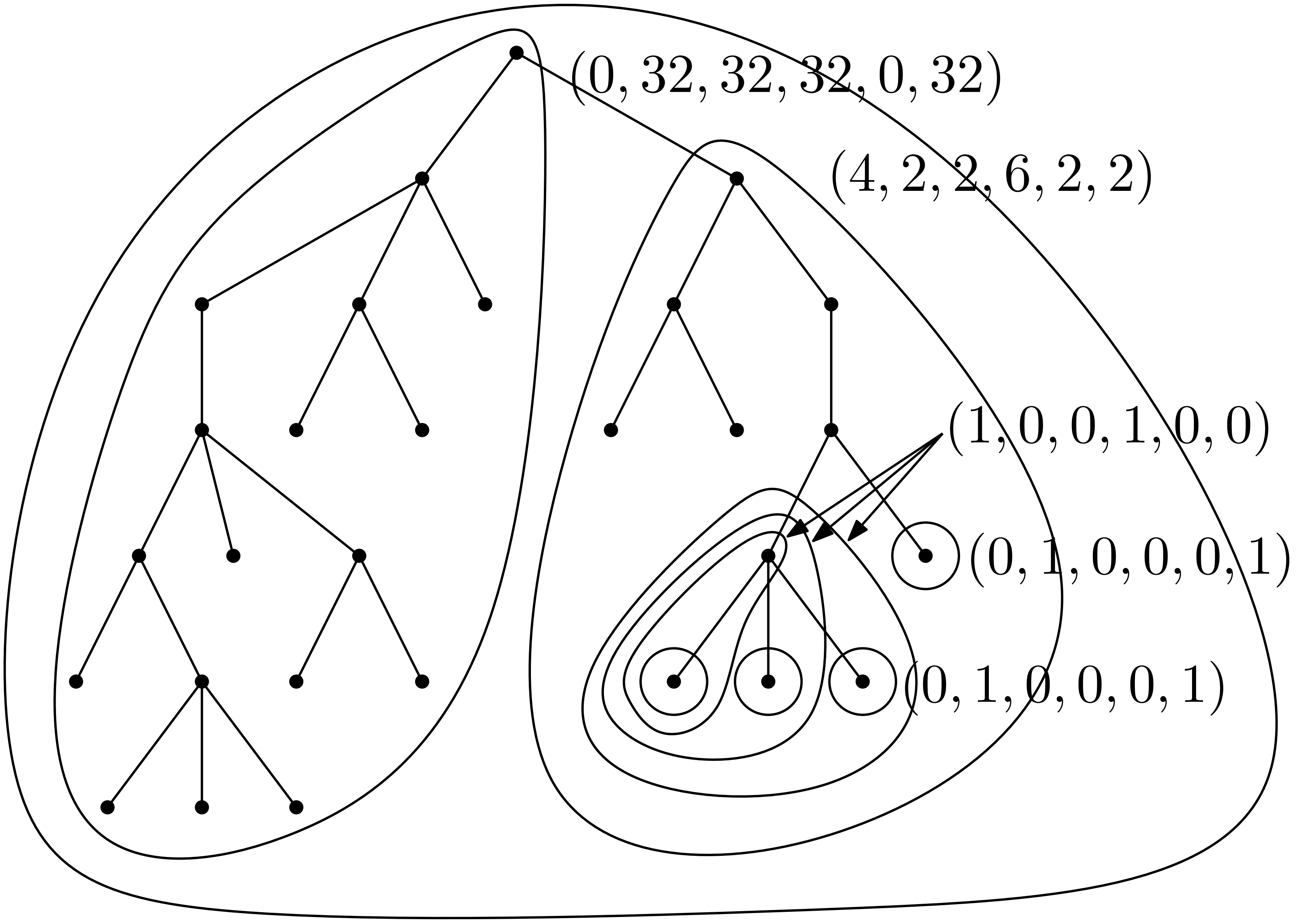
to every rooted tree.

$G =$ the number of (partial) solutions of category **G**, etc.

$$\begin{pmatrix} G_A \\ S_A \\ L_A \\ d_A \\ p_A \\ f_A \end{pmatrix} * \begin{pmatrix} G_B \\ S_B \\ L_B \\ d_B \\ p_B \\ f_B \end{pmatrix} = \begin{pmatrix} G_A G_B + G_A d_B + G_A f_B + S_A f_B + L_A f_B \\ S_A d_B \\ S_A G_B + L_A G_B + L_A d_B \\ d_A G_B + d_A S_B + d_A d_B + d_A p_B + f_A G_B + f_A S_B \\ p_A d_B + p_A p_B + f_A L_B \\ f_A d_B + f_A p_B \end{pmatrix}$$

Example (G, S, L, d, p, f)

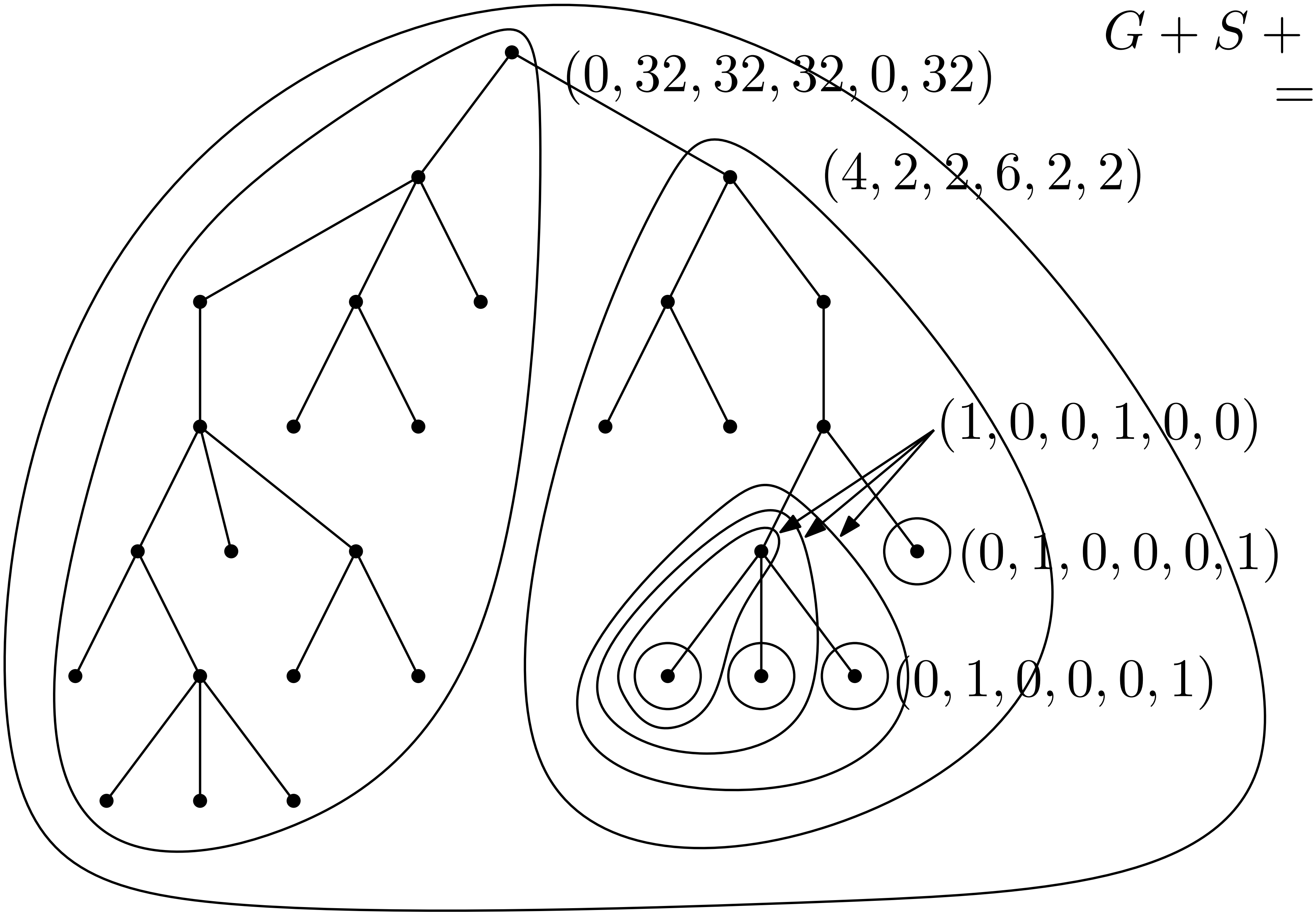
$$(G, S, L, d, p, f) = (128, 192, 448, 640, 64, 256)$$



Example (G, S, L, d, p, f)

$$(G, S, L, d, p, f) = (128, 192, 448, 640, 64, 256)$$

MDS $=$
$G + S + d + p$
$= 1024$



The Possible Numbers of MDSs

$\mathcal{V}_n = \{$ all possible vectors of rooted trees with n vertices $\}$

$$\mathcal{V}_1 := \{(0,1,0,0,0,1)\}$$

$$\mathcal{V}_n := \bigcup_{1\le i<n} \mathcal{V}_i * \mathcal{V}_{n-i}, \text{ for } n \ge 2$$

$M_n =$ the maximum number of MDSs in a tree with n vertices

$$M_n = \max\{\, G+S+d+p \mid (G,S,L,d,p,f) \in \mathcal{V}_n\}$$

M_n is supermultiplicative:

$$M_{i+j} \ge M_i M_j$$

The Possible Numbers of MDSs

n	$\sqrt[n]{M_n}$	M_n	$\text{hull}^+ \mathcal{V}_n$	$\text{hull}\, \mathcal{V}_n$	$\lvert\mathcal{V}_n\rvert$
1	1	1	1	1	1
2	1.41421356237310	2	1	1	1
3	1.25992104989487	2	2	2	2
4	1.41421356237310	4	2	2	4
5	1.31950791077289	4	4	4	7
6	1.41421356237309	8	3	5	13
7	1.36873810664220	9	6	9	24
8	1.41421356237310	16	7	13	45
9	1.38702322584422	19	11	19	85
10	1.41421356237310	32	14	32	159
11	1.40157620020641	41	17	39	308
12	1.41421356237309	64	24	73	588
13	1.40739771128108	85	26	85	1180
14	1.41421356237309	128	30	144	2326
15	1.41209815120249	177	30	176	4753
16	1.41421356237310	256	36	279	9591
17	1.41397457411881	361	39	337	19793

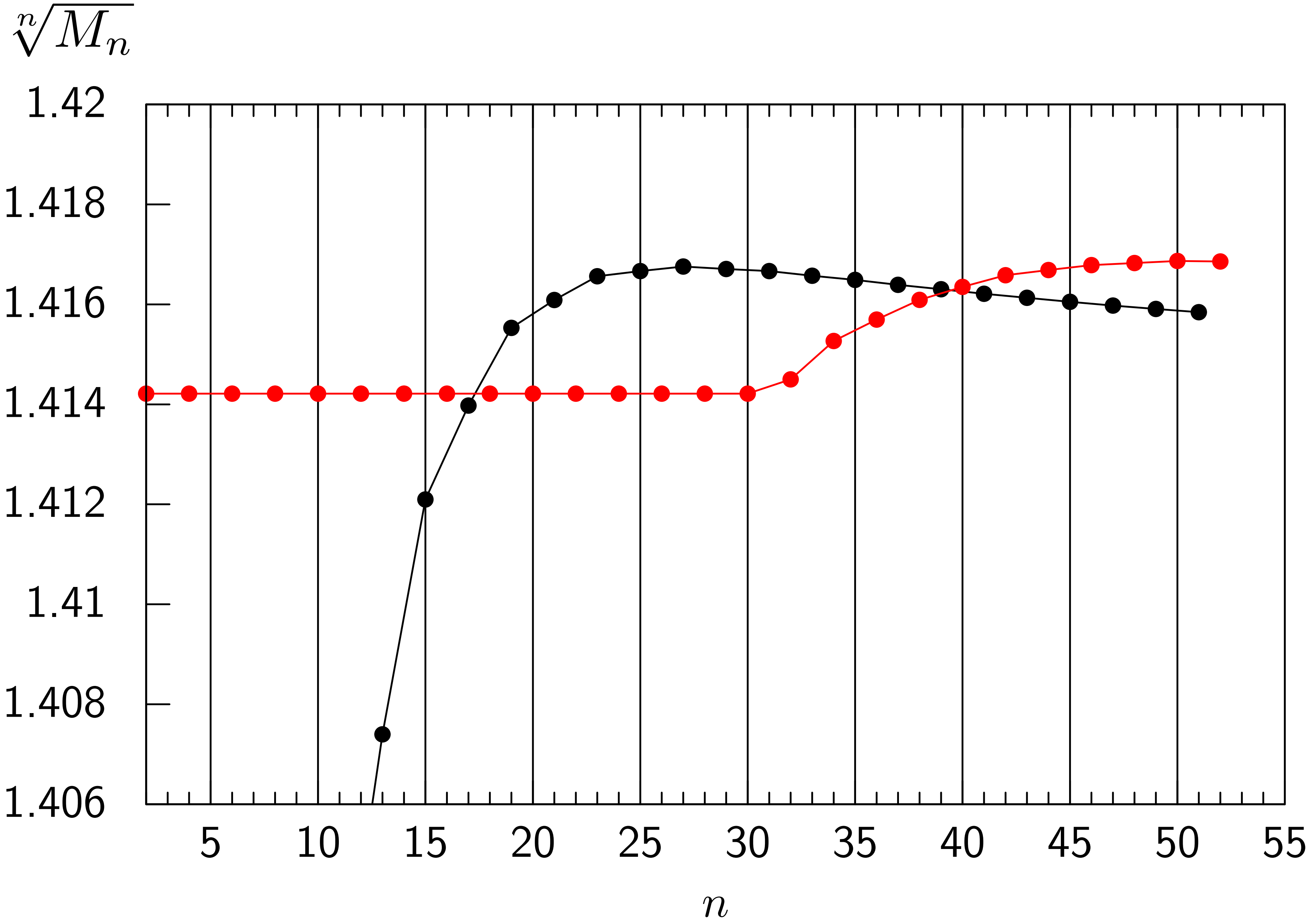
The Possible Numbers of MDSs

11	1.40157620020641	41	17	39	308
12	1.41421356237309	64	24	73	588
13	1.40739771128108	85	26	85	1180
14	1.41421356237309	128	30	144	2326
15	1.41209815120249	177	30	176	4753
16	1.41421356237310	256	36	279	9591
17	1.41397457411881	361	39	337	19793
18	1.41421356237309	512	51	492	40638
19	1.41553085871039	737	47	612	84641
20	1.41421356237310	1024	66	841	176255
21	1.41608793848702	1489	58	1055	369635
22	1.41421356237310	2048	74	1320	775935
23	1.41656252137841	3009	62	1641	1634901
24	1.41421356237309	4096	93	1969	3451490
25	1.41666558384650	6049	75	2435	7303232
26	1.41421356237310	8192	111	2805	15481738
27	1.41675632056381	12161	87	3456	32868146
28	1.41421356237309	16384	119	3871	
29	1.41670718070637	24385	102	4656	
30	1.41421356237310	32768	125	5329	

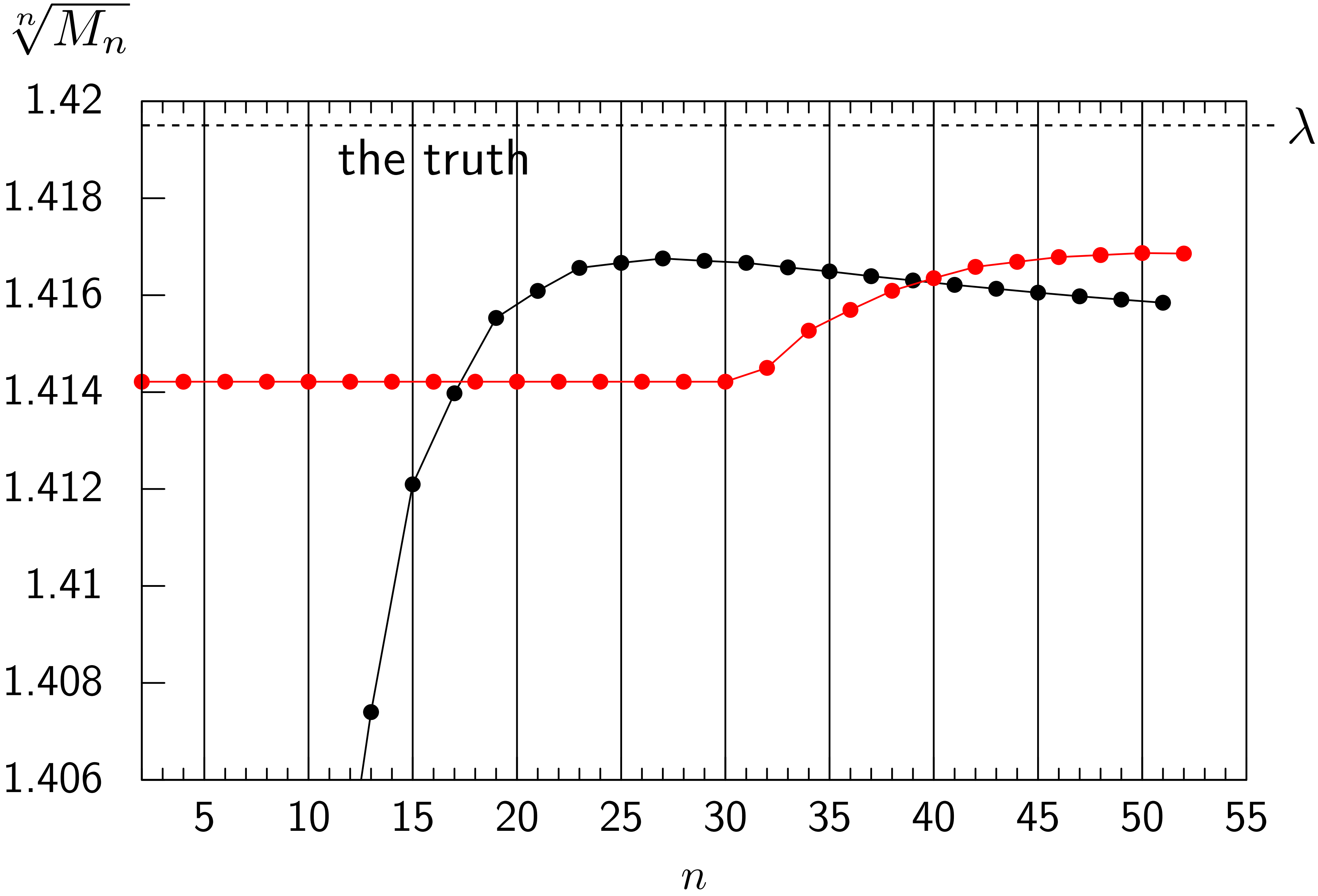
The Possible Numbers of MDSs

27	1.41675632056381	12161	87	3456
28	1.41421356237309	16384	119	3871
29	1.41670718070637	24385	102	4656
30	1.41421356237310	32768	125	5329
31	1.41666501243844	48897	116	6227
32	1.41449859435768	65960	123	7248
33	1.41657202787702	97921	129	8436
34	1.41526678247498	134432	130	9719
35	1.41648981352598	196097	146	11277
36	1.41569656428574	272224	151	12878
37	1.41639156076937	392449	177	14890
38	1.41609068088382	551392	166	16931
39	1.41630342192653	785409	193	19088
40	1.41634892845829	1113808	184	22214
41	1.41621264079532	1571329	209	24075
42	1.41658315523612	2249920	217	28344
43	1.41613031644569	3143681	212	30029
44	1.41668758343879	4529600	238	35068
45	1.41605019185075	6288385	220	36809
46	1.41678485046458	9119680	240	42438
47	1.41597689193916	12578817	233	44773
48	1.41682808199910	18332576	273	50902
49	1.41590722737106	25159681	260	
50	1.41686791092506	36852608	287	
51	1.41584303009330	50323457	264	
52	1.41685798299446	73955200	293	

The Possible Numbers of MDSs



The Possible Numbers of MDSs



Majorization and Convex Hull

$\mathcal{V}_n = \{$ all possible vectors of rooted trees with n vertices $\}$

$$\mathcal{V}_1 := \{(0,1,0,0,0,1)\}$$

$$\mathcal{V}_n := \bigcup_{1 \le i < n} \mathcal{V}_i * \mathcal{V}_{n-i}, \text{ for } n \ge 2$$

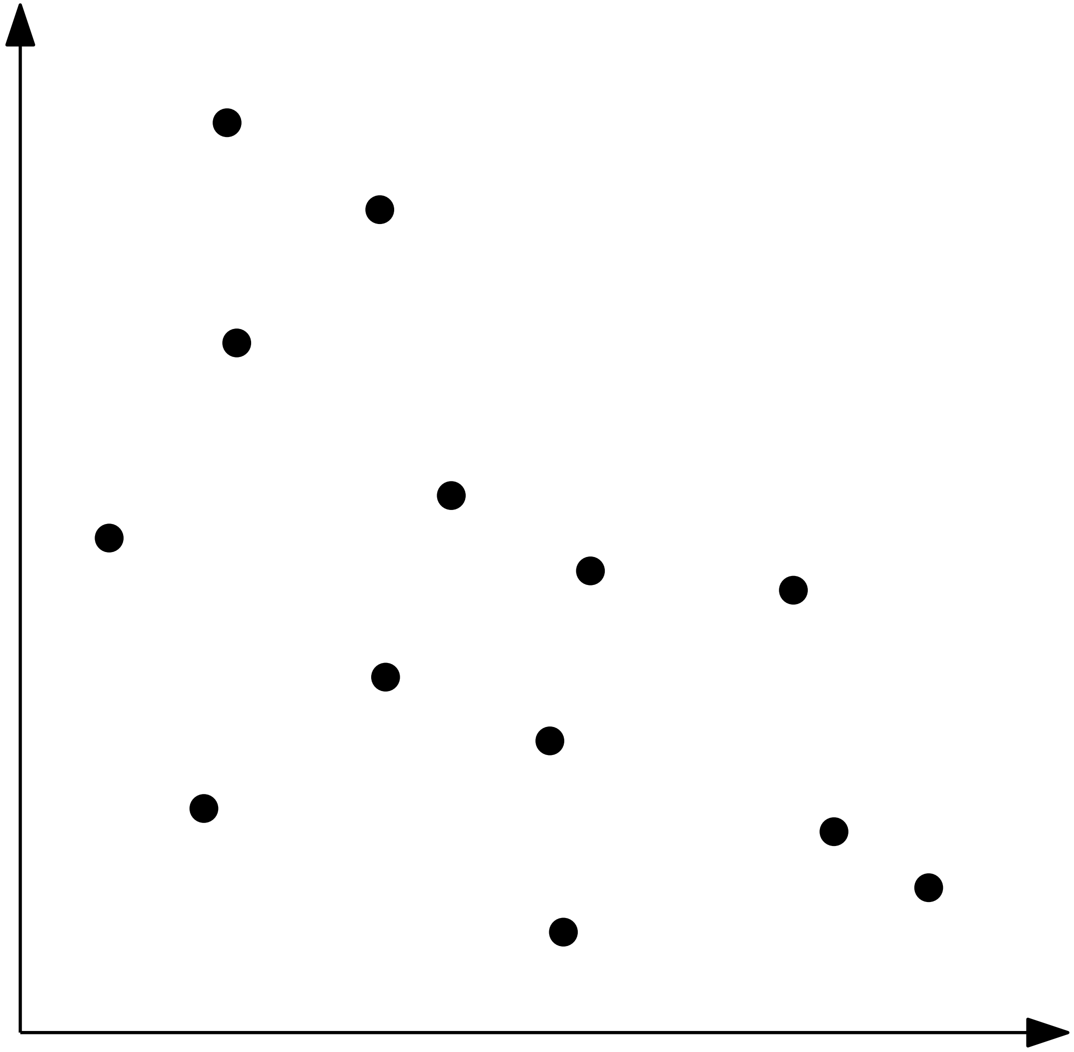
$(G_1, S_1, L_1, d_1, p_1, f_1) \le (G_2, S_2, L_2, d_2, p_2, f_2)$
$\Longrightarrow$ omit $(G_1, S_1, L_1, d_1, p_1, f_1)$ from $\mathcal{V}_n$

G $>$ **S** $>$ **L** and **d** $>$ **p**

"$*$" is a bilinear operation
$\Longrightarrow$ It suffices to keep the convex hull vertices of $\mathcal{V}_n$

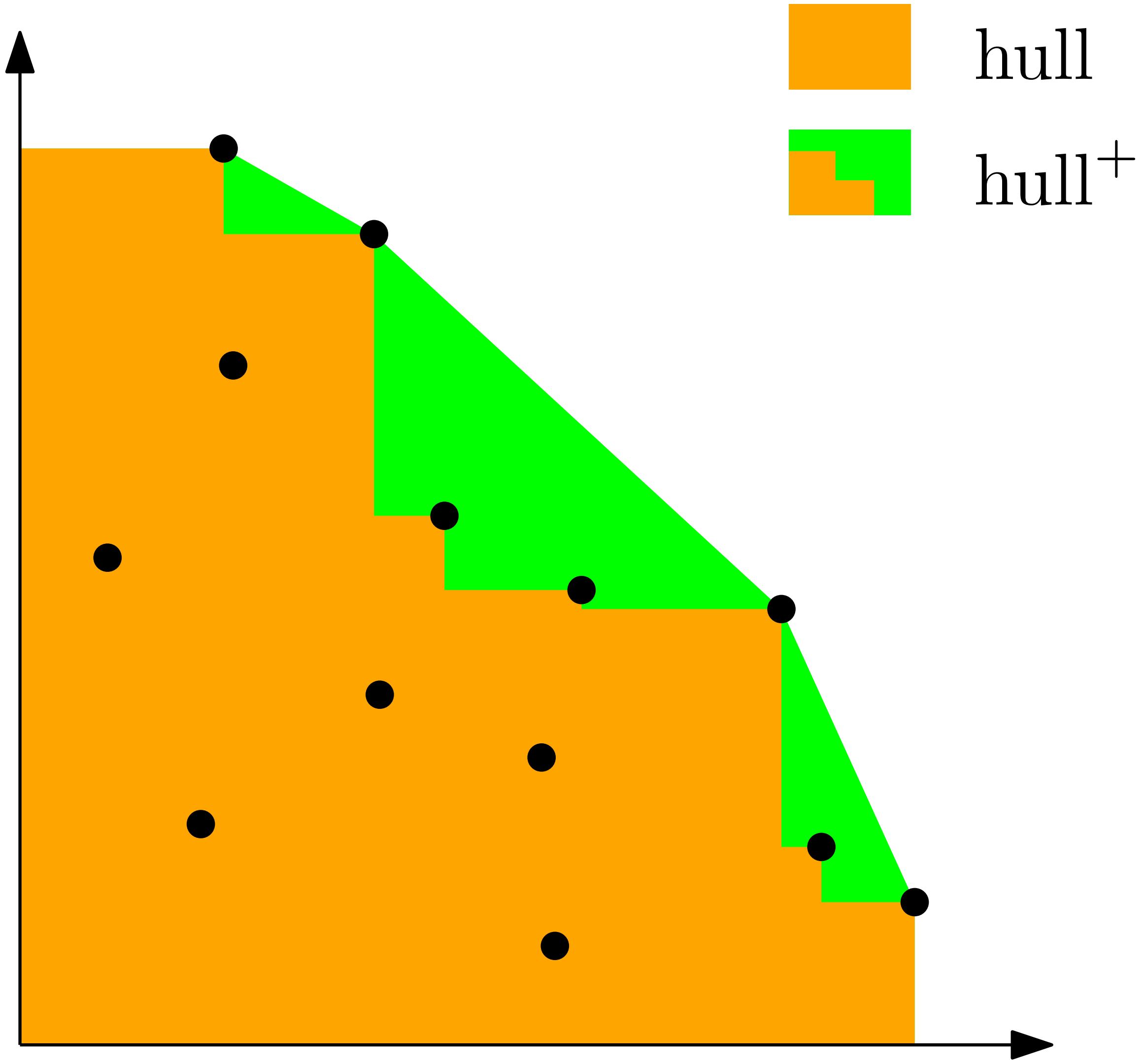
Majorization and Convex Hull

hulls in two dimensions

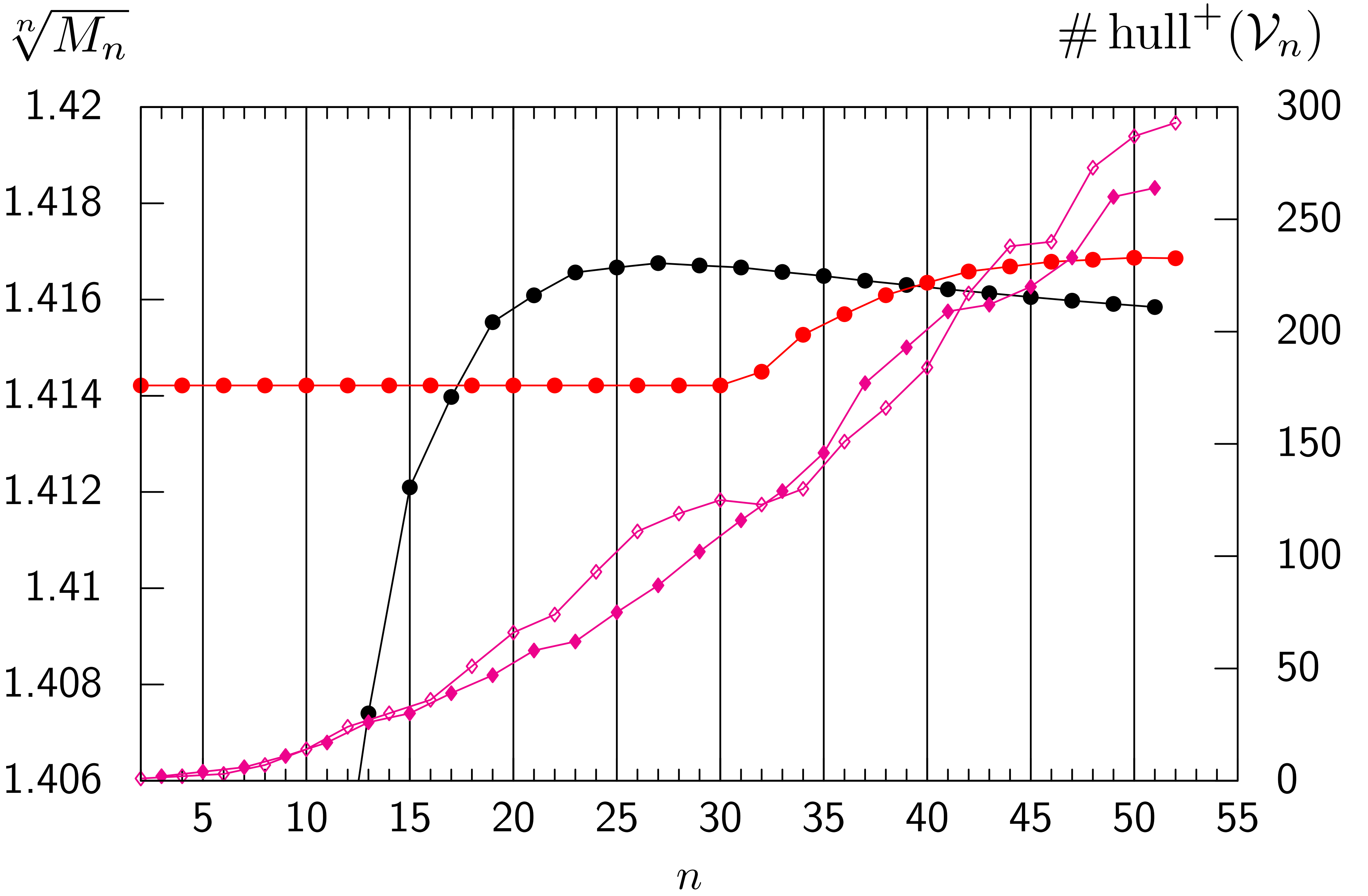


Majorization and Convex Hull

hulls in two dimensions



Majorization and Convex Hull



Upper Bound: Enclosure by a Polytope

$$\mathcal{V}_1 := \{(0,1,0,0,0,1)\}$$

$$\mathcal{V}_n := \bigcup_{1\le i<n} \mathcal{V}_i * \mathcal{V}_{n-i}, \text{ for } n \ge 2$$

PROPOSITION:

Find a bounded (convex) set P such that

$$(0,1,0,0,0,1)/\lambda \in P \quad \text{and} \quad P * P \subseteq P$$

Then $M_n = O(\lambda^n)$.

Proof: $\mathcal{V}_n \subseteq \lambda^n P$ by induction on n

In fact, $\lambda^* := \lim \sqrt[n]{M_n}$ is the smallest λ for which P exists.

Automatic method: try some λ. Set $Q := \{(0,1,0,0,0,1)/\lambda\}$. Repeatedly set $Q := \mathrm{hull}^+(Q \cup (Q * Q))$ until convergence or divergence sets in.

The Polytope P

Let $\lambda = \sqrt[13]{95} \approx 1.4194908.$ $P := \text{hull}^{+}(v_1, \ldots, v_{55})$

$$\begin{aligned}
v_1 &= v_1 * v_{32} && = (0.9, 0, 0, 0, 0, 0) \\
v_2 & && = (0, 1, 0, 0, 0, 1)\lambda^{-1} \\
v_3 &= v_2 * v_2 && = (1, 0, 0, 1, 0, 0)\lambda^{-2} \\
v_4 &= v_2 * v_3 && = (0, 1, 1, 1, 0, 1)\lambda^{-3} \\
v_5 &= v_2 * v_4 && = (1, 1, 0, 1, 1, 1)\lambda^{-4} \\
v_6 &= v_4 * v_3 && = (0, 1, 3, 3, 0, 1)\lambda^{-5} \\
v_7 &= v_2 * v_5 && = (1, 1, 1, 2, 0, 2)\lambda^{-5} \\
v_8 &= v_2 * v_6 && = (1, 3, 0, 1, 3, 3)\lambda^{-6} \\
v_9 &= v_6 * v_3 && = (0, 1, 7, 7, 0, 1)\lambda^{-7} \\
v_{10} &= v_7 * v_3 = v_4 * v_5 && = (2, 1, 3, 6, 0, 2)\lambda^{-7} \\
\ldots \\
v_{53} &= v_{24} * v_{19} && = (63, 961, 0, 63, 1922, 961)\lambda^{-23} \\
v_{54} &= v_{52} * v_3 = v_{19} * v_{24} && = (992, 1, 63, 2016, 0, 32)\lambda^{-23} \\
v_{55} &= v_{33} * v_{26} && = (127, 3969, 0, 127, 7938, 3969)\lambda^{-27}
\end{aligned}$$

The Polytope P

Let $\lambda = \sqrt[13]{95} \approx 1.4194908.$ $P := \mathrm{hull}^{+}(v_1, \dots, v_{55})$

Need to prove that $v_i * v_j \in P$:
Some products are *exactly* equal to another vertex:

$$v_2 * v_2 = v_3, \quad v_{13} * v_5 = v_{27}, \quad v_1 * v_{32} = v_1$$

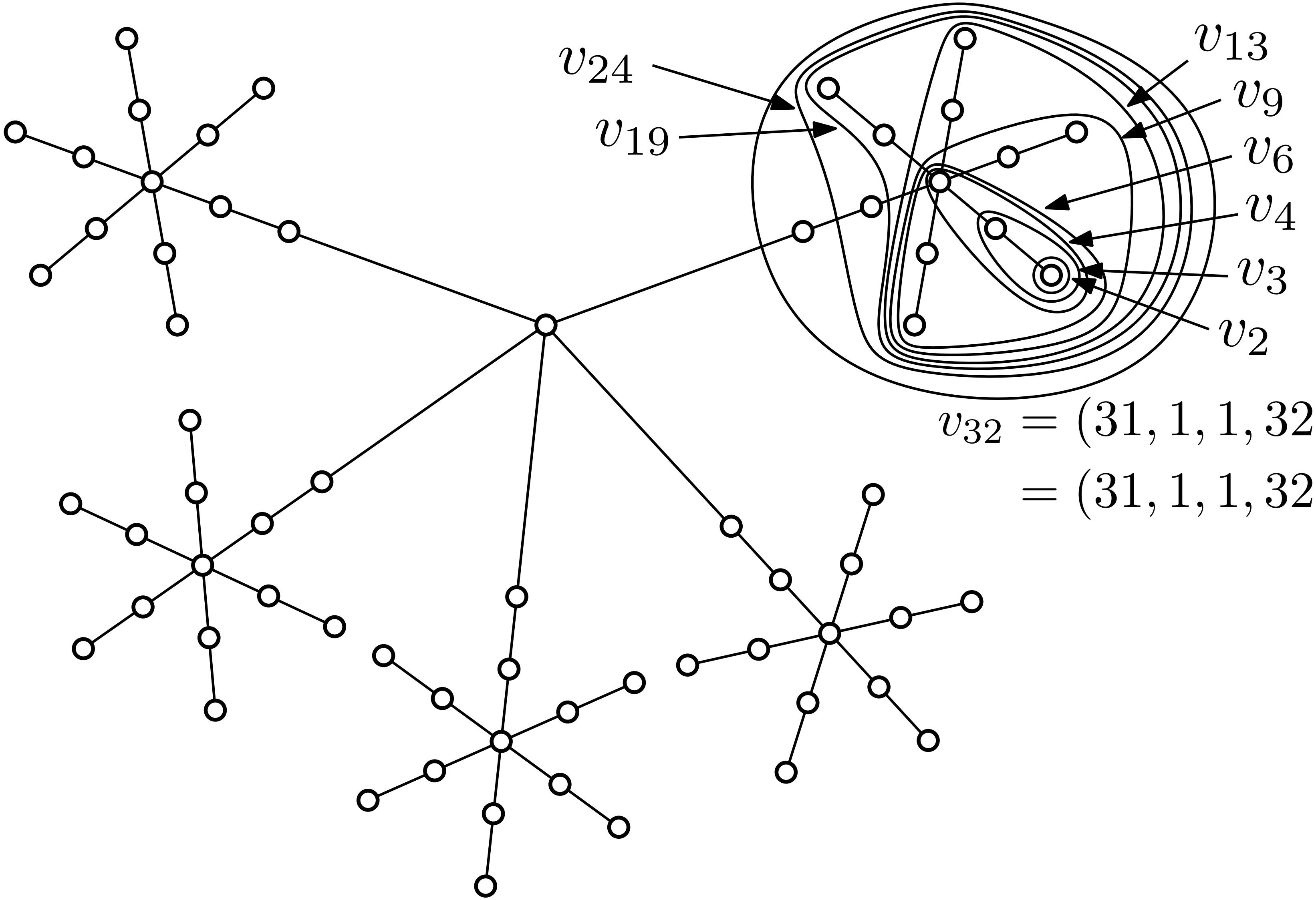
Others are proved by checking inequalities that were found by linear programming:

$$v_9 * v_{55} \le 0.3078\, v_{20} + 0.3709\, v_{28} + 0.3010\, v_{21} + 0.0203\, v_{24}$$

$$0.3078 + 0.3709 + 0.3010 + 0.0203 = 1$$

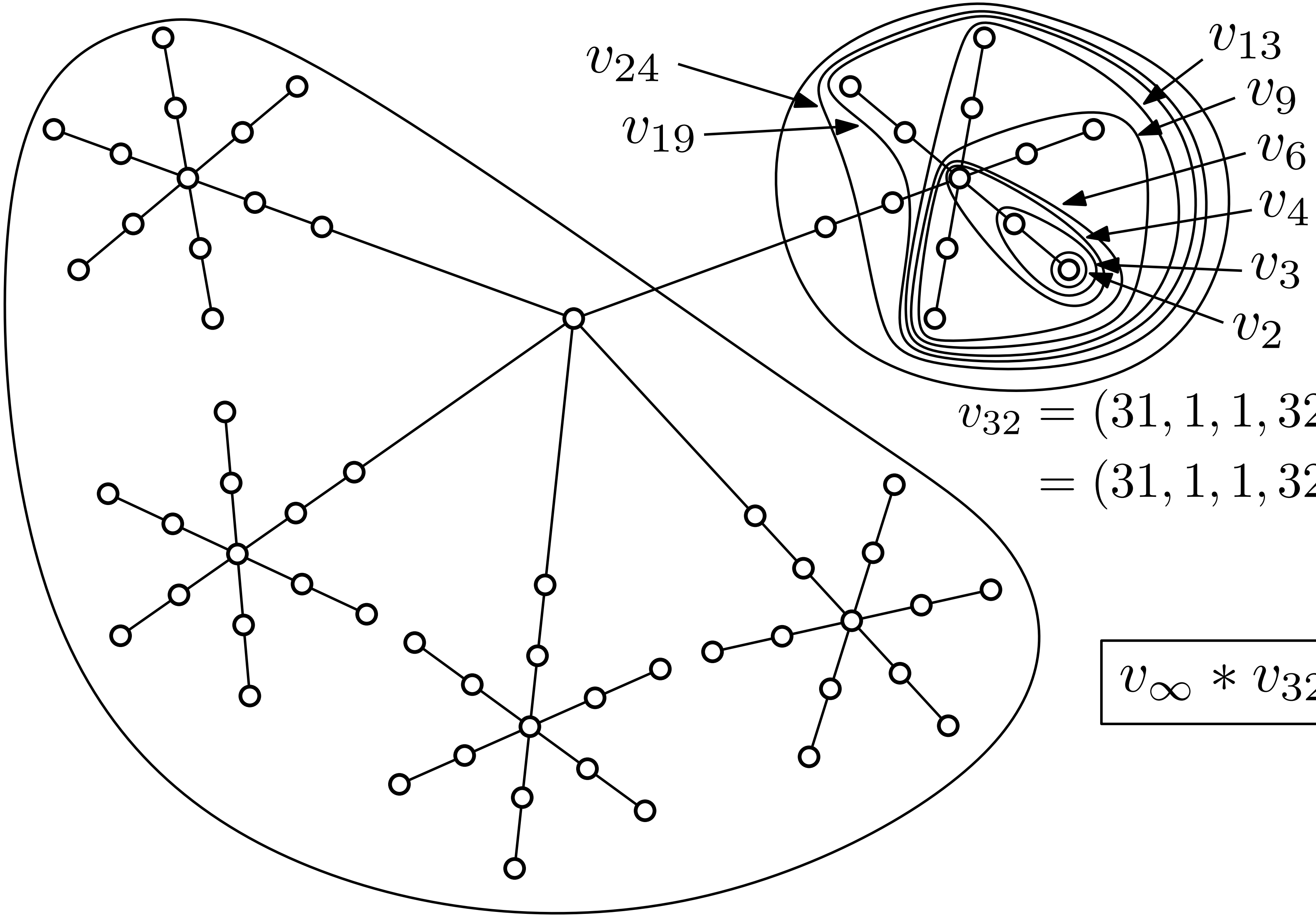
Coefficients with 4 decimal digits.
Smallest margin $\approx 0.000004.$ □

Exact Computation is Necessary



$$v_{32} = (31, 1, 1, 32, 0, 32)\lambda^{-13}$$
$$= (31, 1, 1, 32, 0, 32)/95$$

Exact Computation is Necessary

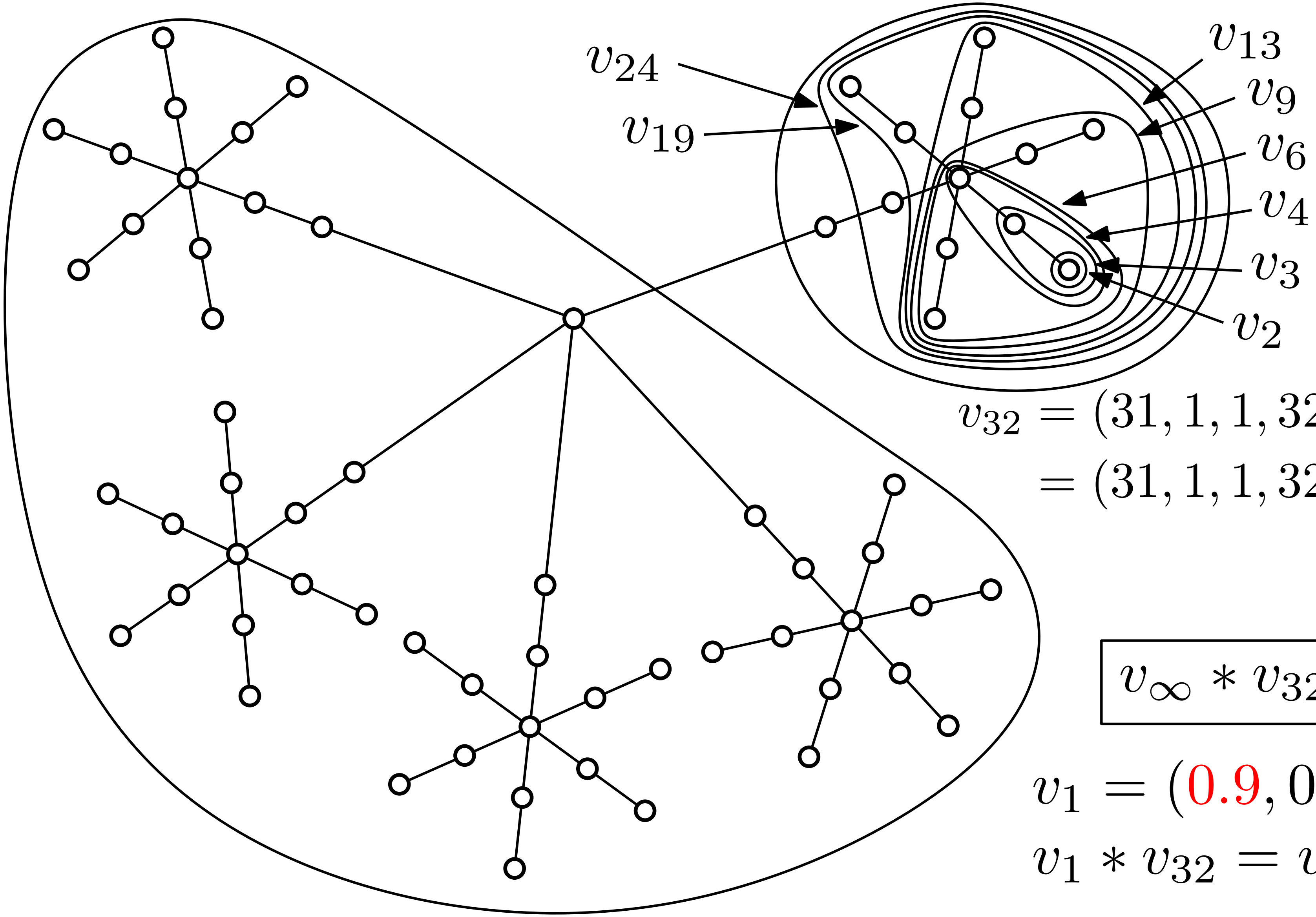


$$v_{32} = (31, 1, 1, 32, 0, 32)\lambda^{-13}$$
$$= (31, 1, 1, 32, 0, 32)/95$$

$$\boxed{v_\infty * v_{32} = v_\infty}$$

$$(95^k(1 + o(1)), \cdot, \cdot, \cdot, \cdot, \cdot)\lambda^{-(13k+1)} \to (1/\lambda, 0, 0, 0, 0, 0) = v_\infty$$

Exact Computation is Necessary



$$v_{32} = (31,1,1,32,0,32)\lambda^{-13}$$
$$= (31,1,1,32,0,32)/95$$

$$\boxed{v_\infty * v_{32} = v_\infty}$$

$$v_1 = (0.9,0,0,0,0,0)$$
$$v_1 * v_{32} = v_1$$

$$(95^k(1+o(1)),\cdot,\cdot,\cdot,\cdot,\cdot)\lambda^{-(13k+1)} \to (1/\lambda,0,0,0,0,0) = v_\infty$$

Everything is in the Transition Matrix

	G	S	L	d	p	f
G	G	—	—	G	—	G
S	L	—	—	S	—	G
L	L	—	—	L	—	G
d	d	d	—	d	d	—
p	—	—	—	p	p	—
f	d	d	p	f	f	—

. . . plus the “start vector” $\mathbf{u}_1 = (0, 1, 0, 0, 0, 1) \in \mathcal{V}_1$ and the “end weights” $(1, 1, 0, 1, 1, 0)$: $M(\mathbf{a}) = \langle (1, 1, 0, 1, 1, 0), \mathbf{a} \rangle$

$(\mathbf{a} * \mathbf{b}) * \mathbf{c} = (\mathbf{a} * \mathbf{c}) * \mathbf{b}$ “right commutative law”

$(\mathbf{a} * \mathbf{u}_1) * \mathbf{u_1} = \mathbf{a} * \mathbf{u_1}$ Twin leaves don’t matter.

Everything is in the Transition Matrix

	G	S	L	d	p	f
G	G	–	–	G	–	G
S	L	–	–	S	–	G
L	L	–	–	L	–	G
d	d	d	–	d	d	–
p	–	–	–	p	p	–
f	d	d	p	f	f	–

. . . plus the "start vector" $\mathbf{u}_1 = (0, 1, 0, 0, 0, 1) \in \mathcal{V}_1$ and the "end weights" $(1, 1, 0, 1, 1, 0)$: $M(\mathbf{a}) = \langle (1, 1, 0, 1, 1, 0), \mathbf{a} \rangle$

$(\mathbf{a} * \mathbf{b}) * \mathbf{c} = (\mathbf{a} * \mathbf{c}) * \mathbf{b}$ "right commutative law"

$(\mathbf{a} * \mathbf{u}_1) * \mathbf{u_1} = \mathbf{a} * \mathbf{u_1}$ Twin leaves don't matter.

$$(\mathbf{a} * \mathbf{u}_1) * (\mathbf{b} * \mathbf{u}_1) = (\mathbf{a} * \mathbf{u}_1) \cdot M(\mathbf{b} * \mathbf{u}_1)$$

$$\Longrightarrow M((\mathbf{a} * \mathbf{u}_1) * (\mathbf{b} * \mathbf{u}_1)) = M(\mathbf{a} * \mathbf{u}_1) \cdot M(\mathbf{b} * \mathbf{u}_1)$$

→ supermultiplicativity

Open Questions

- A theory of “eigenvalues” of bilinear operations?

  Given a transition matrix, a start vector, and end weights, how fast is the growth?

- “Gray code” enumeration of minimal dominating sets?

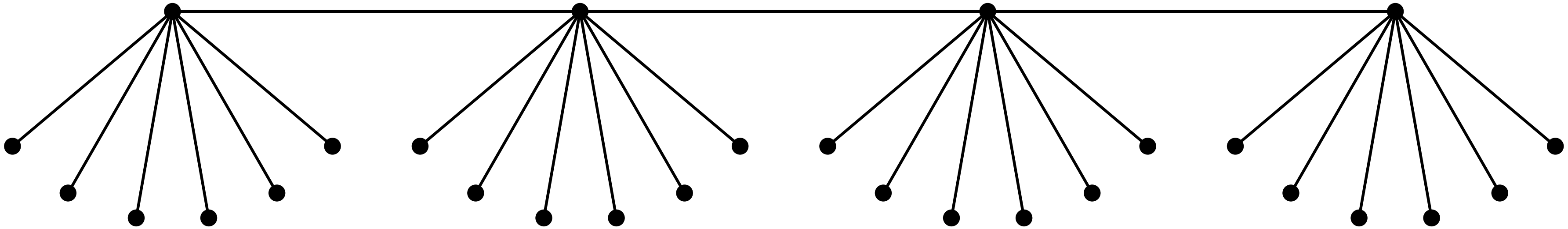
  Assume: Every vertex is adjacent to at most one leaf.
  Want: $O(1)$ changes between successive sets.
  Preferably computable in $O(1)$ time, after $O(n)$ setup (*constant delay*).

Gray Codes?

Output-sensitive enumeration:

The minimal dominating sets of a tree with n vertices can be enumerated with $O(n)$ setup time and with $O(n)$ delay between successive solutions.

Can it be done with $O(1)$ delay?



0000
0001
0011
0010
0110
0111
0101
0100
1100
1101
1111
1110
1010
1011
1001
1000

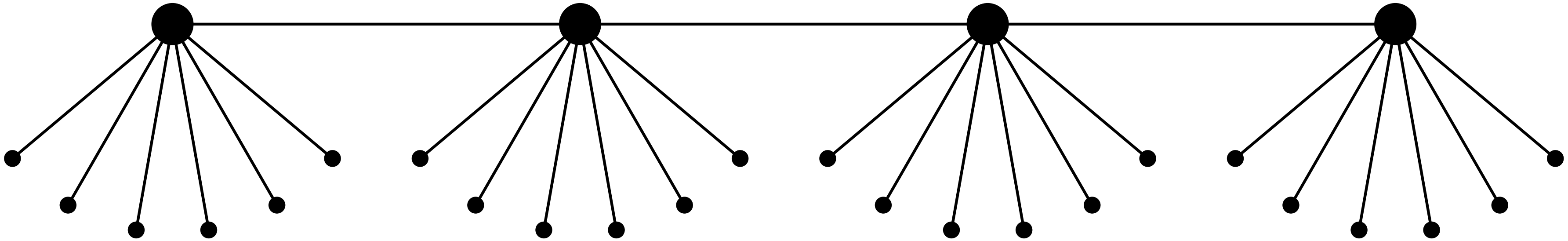
0000
0001
0011
0010
0110
0111
0101
0100
1100

Gray Codes?

Output-sensitive enumeration:

The minimal dominating sets of a tree with n vertices can be enumerated with $O(n)$ setup time and with $O(n)$ delay between successive solutions.

Can it be done with $O(1)$ delay?



0000
0001
0011
0010
0110
0111
0101
0100
1100
1101
1111
1110
1010
1011
1001
1000

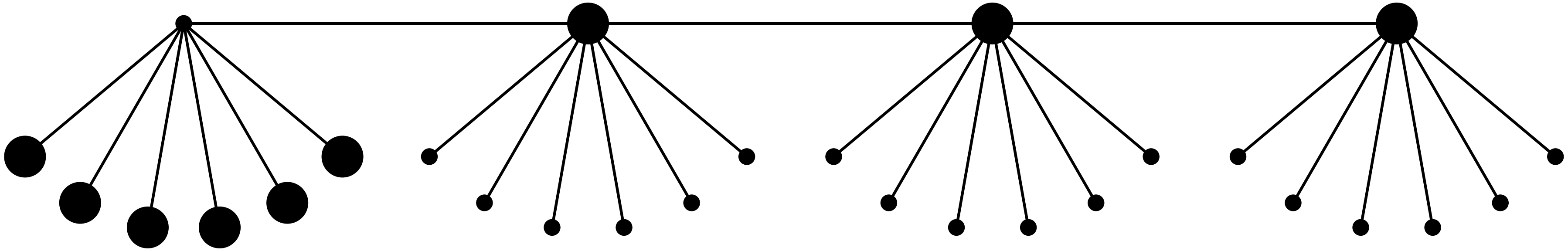
0000
0001
0011
0010
0110
0111
0101
0100
1100

Gray Codes?

Output-sensitive enumeration:

The minimal dominating sets of a tree with n vertices can be enumerated with $O(n)$ setup time and with $O(n)$ delay between successive solutions.

Can it be done with $O(1)$ delay?



0000
0001
0011
0010
0110
0111
0101
0100
1100
1101
1111
1110
1010
1011
1001
1000

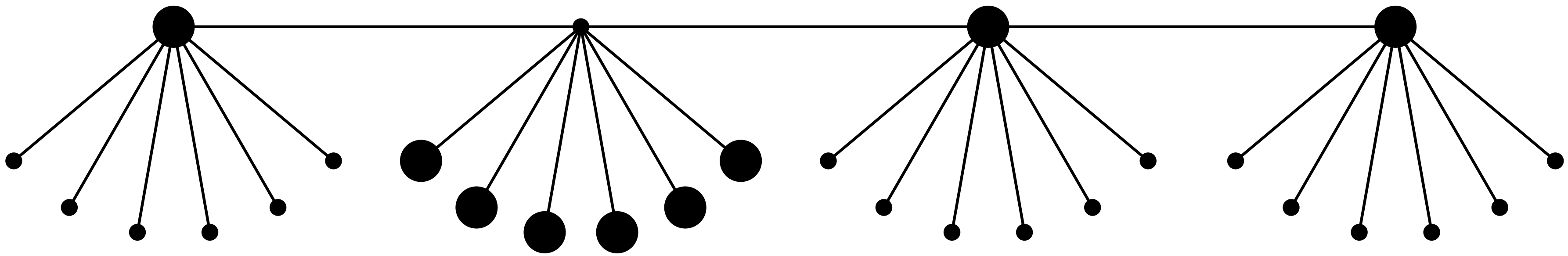
0000
0001
0011
0010
0110
0111
0101
0100
1100

Gray Codes?

Output-sensitive enumeration:

The minimal dominating sets of a tree with n vertices can be enumerated with $O(n)$ setup time and with $O(n)$ delay between successive solutions.

Can it be done with $O(1)$ delay?



0000

0001

0011

0010

0110

0111

0101

0100

1100

1101

1111

1110

1010

1011

1001

1000

0000

0001

0011

0010

0110

0111

0101

0100

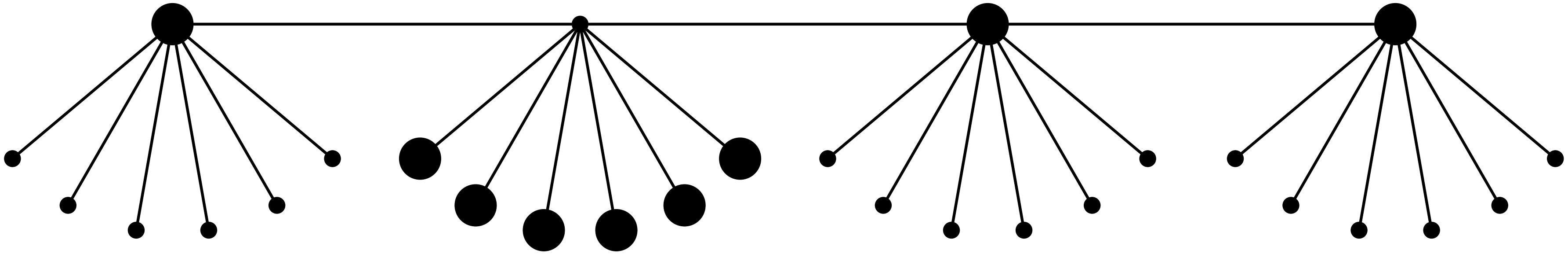
1100

Gray Codes?

Output-sensitive enumeration:

The minimal dominating sets of a tree with n vertices can be enumerated with $O(n)$ setup time and with $O(n)$ delay between successive solutions.

Can it be done with $O(1)$ delay?



Have to *cluster* leaves with a common neighbor.

0000
0001
0011
0010
0110
0111
0101
0100
1100
1101
1111
1110
1010
1011
1001
1000

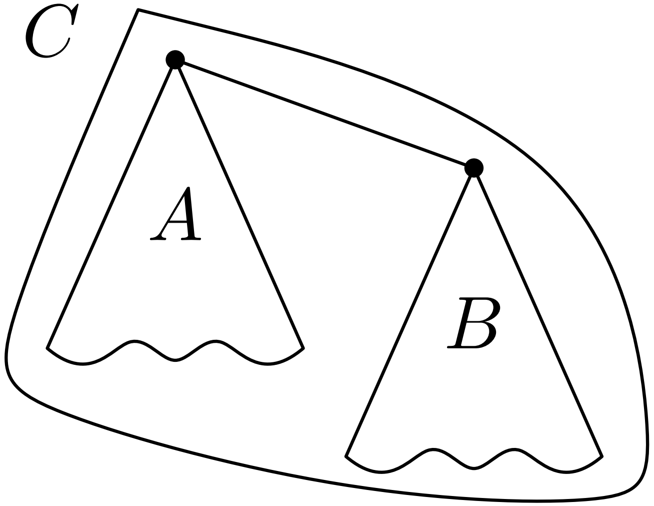
0000
0001
0011
0010
0110
0111
0101
0100
1100

Enumeration

$$\begin{pmatrix} G_A \\ S_A \\ L_A \\ d_A \\ p_A \\ f_A \end{pmatrix} * \begin{pmatrix} G_B \\ S_B \\ L_B \\ d_B \\ p_B \\ f_B \end{pmatrix} = \begin{pmatrix} G_AG_B + G_Ad_B + G_Af_B + S_Af_B + L_Af_B \\ S_Ad_B \\ S_AG_B + L_AG_B + L_Ad_B \\ d_AG_B + d_AS_B + d_Ad_B + d_Ap_B + f_AG_B + f_AS_B \\ p_Ad_B + p_Ap_B + f_AL_B \\ f_Ad_B + f_Ap_B \end{pmatrix}$$

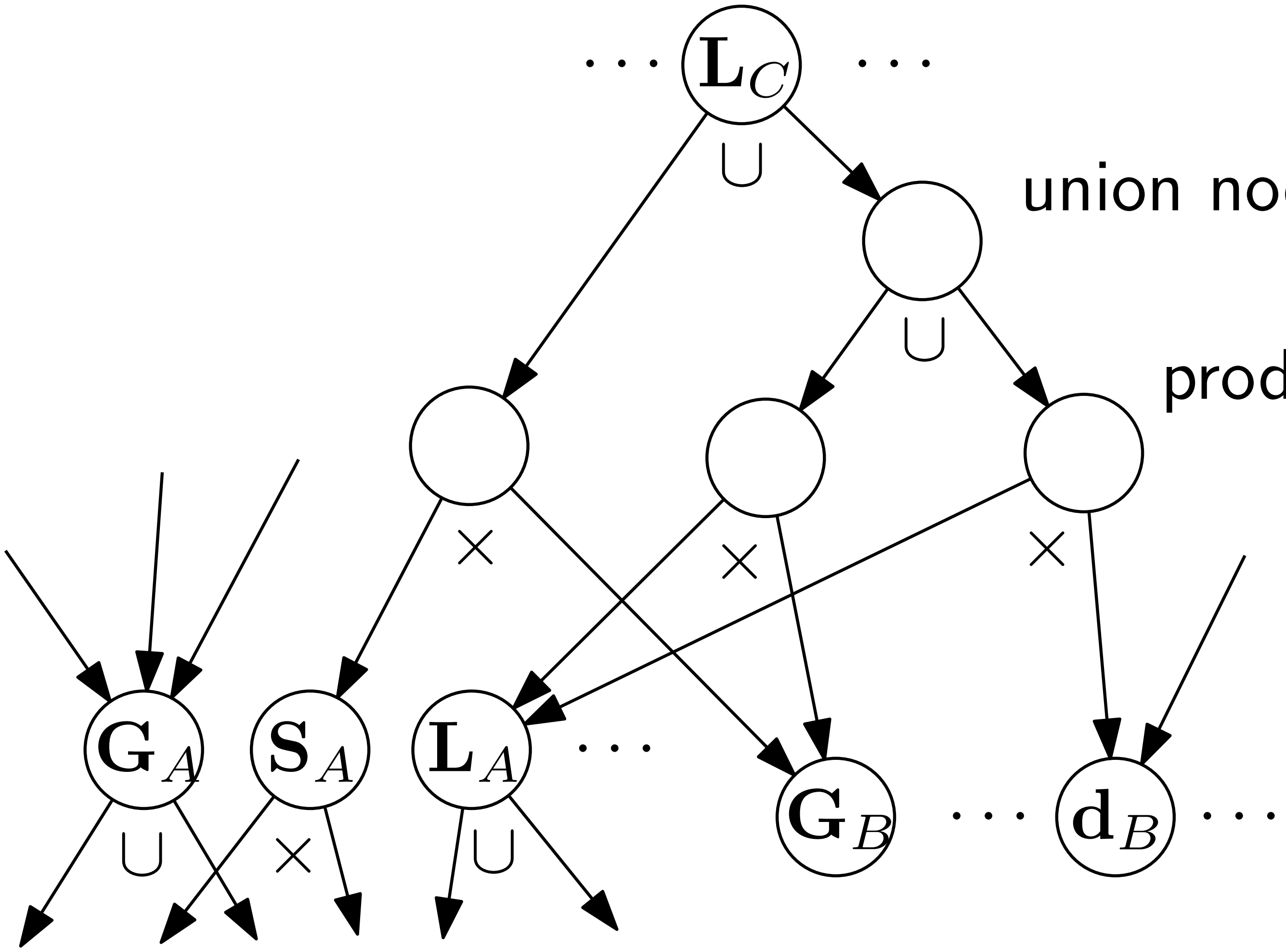
$$L_C = S_AG_B + L_AG_B + L_Ad_B \quad \text{(numbers)}$$

$$\mathbf{L}_C = (\mathbf{S}_A \times \mathbf{G}_B) \cup (\mathbf{L}_A \times \mathbf{G}_B) \cup (\mathbf{L}_A \times \mathbf{d}_B) \quad \text{(sets)}$$



The Solution DAG

$$\mathbf{L}_C = (\mathbf{S}_A \times \mathbf{G}_B) \cup (\mathbf{L}_A \times \mathbf{G}_B) \cup (\mathbf{L}_A \times \mathbf{d}_B) \quad \text{(sets)}$$

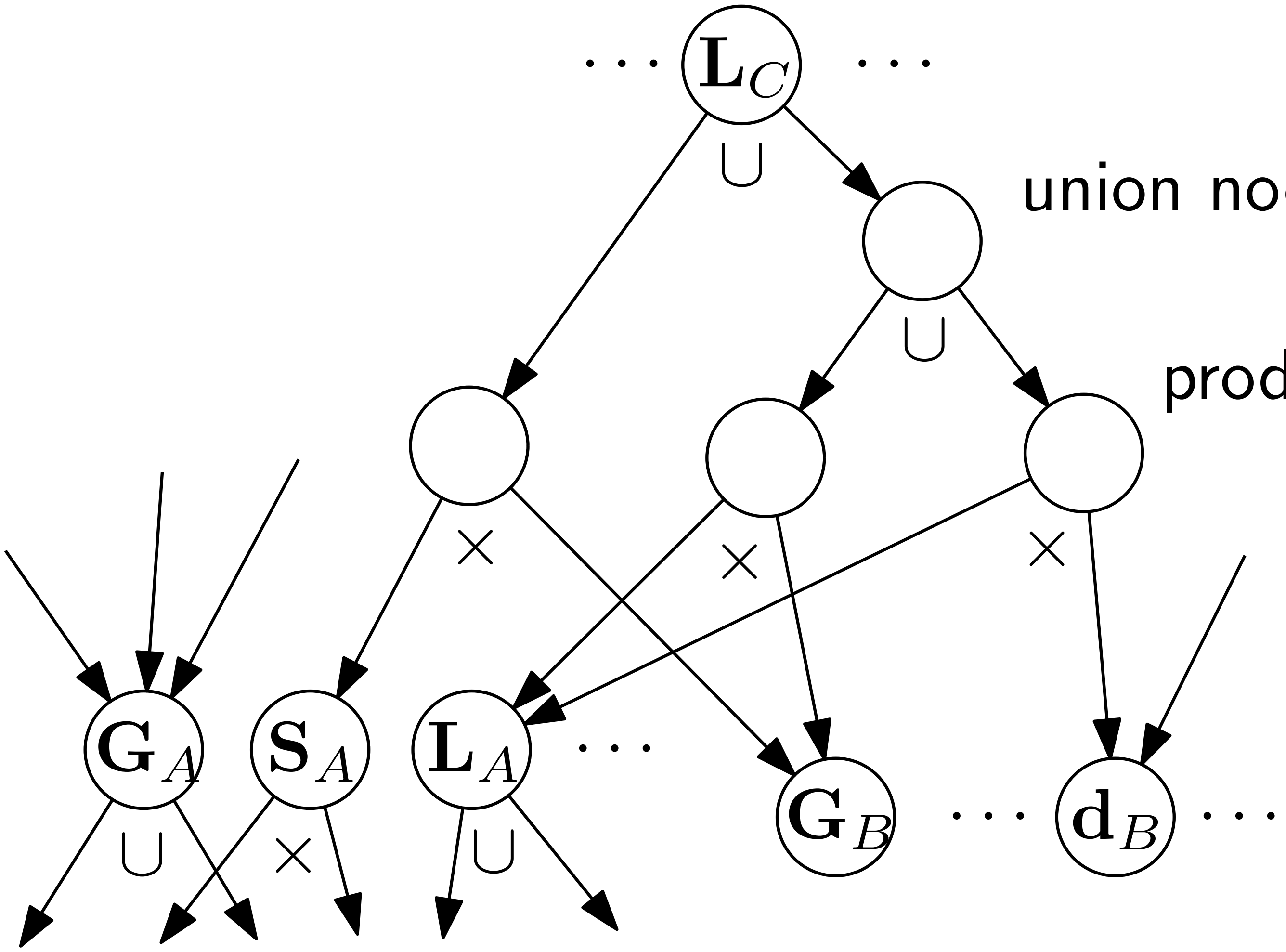


union node

product node

The Solution DAG

$$\mathbf{L}_C = (\mathbf{S}_A \times \mathbf{G}_B) \cup (\mathbf{L}_A \times \mathbf{G}_B) \cup (\mathbf{L}_A \times \mathbf{d}_B) \quad \text{(sets)}$$



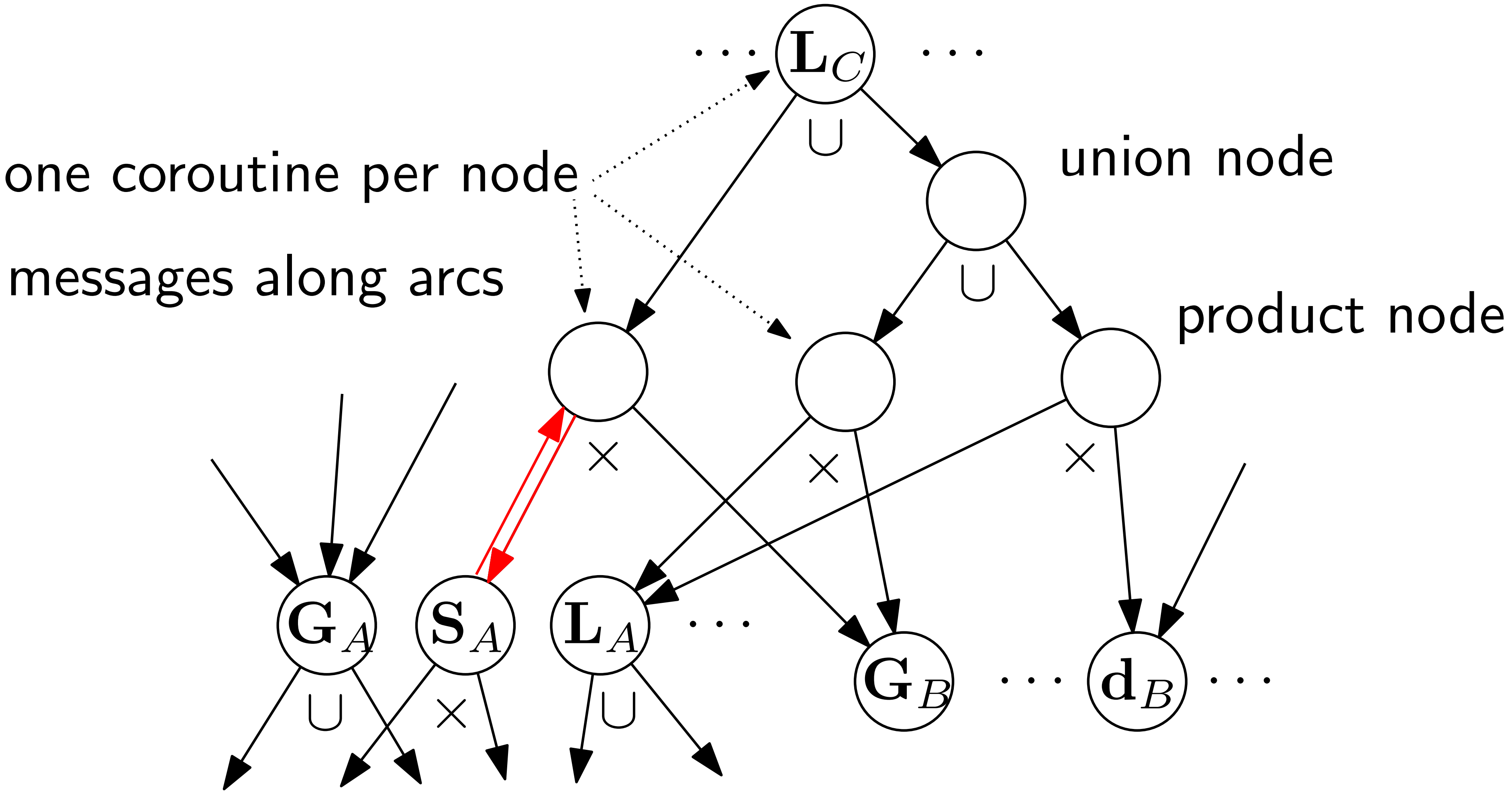
union node

product node

eliminate unneeded nodes

The Solution DAG

$$\mathbf{L}_C = (\mathbf{S}_A \times \mathbf{G}_B) \cup (\mathbf{L}_A \times \mathbf{G}_B) \cup (\mathbf{L}_A \times \mathbf{d}_B) \quad \text{(sets)}$$



one coroutine per node

messages along arcs

union node

product node

eliminate unneeded nodes

Implementation by Generator Functions

Generator functions in PYTHON:

```
def enumerate_basis_node_S(K):
    yield [a]   # category S
def enumerate_basis_node_f(K):
    yield []    # category f, empty list
def enumerate_union_node(K):
    for D in enumerate_solutions(K.child1):
        yield D
    for D in enumerate_solutions(K.child2):
        yield D
def enumerate_product_node(K):
    for D1 in enumerate_solutions(K.child1):
        for D2 in enumerate_solutions(K.child2):
            yield D1+D2 # concatenation of lists
# main call:
for D in enumerate_solutions(target_node):
    print D
```

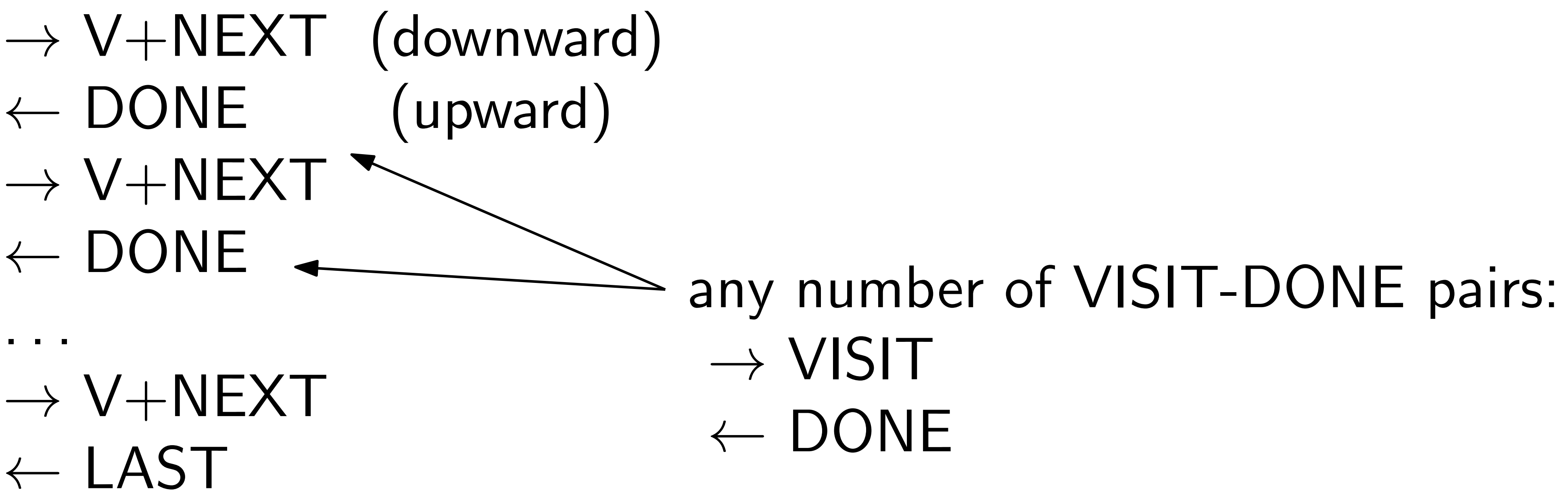
Implementation by Message Passing

Message flow along an arc:

→ V+NEXT (downward)
← DONE (upward)
→ V+NEXT
← DONE
. . .
→ V+NEXT
← LAST

Implementation by Message Passing

Message flow along an arc:



→ V+NEXT (downward)
← DONE (upward)
→ V+NEXT
← DONE
. . .
→ V+NEXT
← LAST

any number of VISIT-DONE pairs:
→ VISIT
← DONE

Implementation by Message Passing

program for a product node

VISIT →	
	VISIT → child 1
	← DONE from child 1
	VISIT → child 2
	← DONE from child 2
← DONE	

program for a product node

V+NEXT →	
	V+NEXT → child 1
	← LAST from child 1
	V+NEXT → child 2
	← LAST from child 2
← LAST	

basis node for vertex a, representing $\{\{a\}\}$

VISIT →	
report	“$a \in D$”
← DONE	
V+NEXT →	
report	“$a \in D$”
← LAST	

basis node for vertex a, representing $\{\emptyset\}$

VISIT →	
report	“$a \notin D$”
← DONE	
V+NEXT →	
report	“$a \notin D$”
← LAST	