Computing treewidth via exact and heuristic lists of minimal separators

Hisao Tamaki
Meiji University

Contents

- Overview of the approach
- Experimental results
- Technical details
  - Dynamic programming
  - Listing minimal separators

Overview of the approach: three components

msDP(G, k, Δ)

G : graph

k : positive integer,

Δ : a set of minimal separators of G

Decides if G has a tree-decomposition of width $\leq k$ that uses minimal separators only from Δ

listExact(G,k)

Generates $\Delta_k(G)$, the set of all minimal separators of cardinality $\leq k$

listHeuristic(G,T,k)

T: a tree decomposition of G

Iteratively generates expanding subsets

$$\Delta^0 \subset \Delta^1 \subset \Delta^2 \subset \Delta^3 \ldots \subseteq \Delta_k(G)$$

Three algorithms for computing the treewidth of G

Ascend

for k ascending from a trivial lower bound:

decide if $tw(G) \leq k$ by calling msDP(G, k, listExact(G, k))

if YES then stop

Descend

T := heuristic tree-decomposition of G by a greedy heuristic

heuristically improve T as log as possible

to improve T of width w, use listHeuristic(G, T, w - 1) to generate

$$\Delta^0 \subset \Delta^1 \subset \Delta^2 \subset \Delta^3 \ldots \subseteq \Delta_{w-1}(G)$$

and try msDP(G, k, Δ^i) for $i = 0, 1, 2, \cdots$

try to show T of width w is optimal by msDP(G, w – 1, listExact(G, w – 1))

Alternate

Alternate between Descend and Ascend, with some resource balancing

Random instances

$\|V\|$	$\|E\|$	tw	$\|\Delta_{tw-1}\|$	FI1	$\|\Delta_{tw}\|$	FI	$\|V\|$	$\|E\|$	tw	$\|\Delta_{tw-1}\|$	FI1	$\|\Delta_{tw}\|$	FI
40	120	14	1021	912	2356	2080	70	210	22	299681	227030	786777	602892
40	160	18	1640	1344	3952	3289	70	280	28	498944	412612	1137482	930417
40	200	20	875	735	1790	1502	70	350	33	590136	464161	1291834	1006981
40	240	22	812	667	1861	1615	70	420	37	472728	386375	991158	797858
40	280	24	631	518	1275	1103	70	490	38	106296	85958	203148	161982
40	320	25	342	296	626	521	70	560	42	150427	122423	293595	235726
40	360	27	292	246	579	474	70	630	45	150442	117591	304528	233298
40	400	28	232	203	469	405	70	700	47	101673	79286	205276	158689
50	150	16	3895	3565	9152	8099	80	240	25	1621664	1424712	4081263	3503941
50	200	20	3772	3377	7878	6956	80	320	31	2284149	1936667	5189162	4362765
50	250	24	5127	4397	10555	8949	80	400	35	988166	827068	2065839	1710869
50	300	26	2788	2299	5345	4417	80	480	39	751344	622050	1481223	1208020
50	350	29	3437	2682	6685	5293	80	560	42	458205	373407	872193	700485
50	400	31	2302	1766	4512	3586	80	640	46	608006	489690	1181883	934389
50	450	32	1163	945	2089	1656	80	720	49	471433	379049	896693	706639
50	500	34	1048	889	1987	1638	80	800	52	438636	355371	846794	671334
60	180	18	11698	9238	26313	22416	90	270	27	7947239	5585295	19521897	13560016
60	240	22	12743	10540	27052	21984	90	360	35	30498292	25231339	71039889	–
60	300	27	27359	20595	56991	41584	90	450	40	24205797	18839873	51925771	–
60	360	30	17956	13829	34793	26356	90	540	45	19877659	15311306	41166209	31119888
60	420	33	17281	13843	33755	26586	90	630	49	11958408	9812327	23932551	–
60	480	34	5862	4789	10320	8248	90	720	52	7106240	5573022	13888202	10716600
60	540	38	11693	9746	22610	18241	90	810	55	4770228	3805194	9237122	7228454
60	600	40	9612	7931	19319	15958	90	900	57	2115790	1721063	3980250	3176283

Table 1. Random graph instances used in our experiments: columns FI1 and FI show the number of feasible inbound sets for $k = \mathrm{tw}(G) - 1$ and $k = \mathrm{tw}(G)$, respectively; an empty field means unsuccessful computation within reasonable amount of resource

Some DIMACS graph coloring instances

name	$\lvert V\rvert$	$\lvert E\rvert$	tw	$\lvert\Delta_{tw-1}\rvert$	FIs for $k = tw - 1$	$\lvert\Delta_{tw}\rvert$	FIs for $k = tw$
myciel6	95	755	35	2639	2583	3938	3848
myciel7	191	2360	66	223317	219381	316296	309735
queen10_10	100	1470	72	2442357	1523527	4199412	2633702
queen11_11	121	1980	87	22351589	13793133	36424473	22429873
DSJC125.1	125	736	[48, 65]	–	–	$\geq^{\dagger}$ 23302449	–
DSJC125.5	125	3891	108	190816	158478	347012	280655
DSJC250.1	250	3218	[83, 177]	–	–	$\geq^{\dagger}$ 1248182	–
DSJC250.5	250	15668	[221,230]	–	–	$\geq^{\dagger}$ 1882525	–

Table 2. A few hard instances from DIMACS graph coloring benchmark set: FI stands for feasible inbounds; † these lower bounds are $|\Delta_{48}(G)|$, $|\Delta_{83}(G)|$, and $|\Delta_{221}(G)|$, for each G

Computing environment

CPU: Intel Core i7-6700 (4 cores), 3.40GHz, 8192KB cache

RAM:32GB

Operating system: Ubuntu 18.04.1 LTS

Programming language: Java 1.8

JVM: jre1.8.0_111

The maximum heap size: 28GB

Implementations are single threaded, except that multiple threads may be invoked for garbage collection by JVM.

The time measured is the elapsed time.

To minimize the influence of system processes, the computer is detached from the network and the graphic user interface is disabled.

Performances of min sep listing algorithms

$\|V\|$	$\|E\|$	tw − 1	$\|\Delta_{tw-1}\|$	time (secs) for generating Δ_{tw-1}: Basic	Pruning	Pruning+Nibbling	Heuristic	|heuristic list| (no. of missings)
70	210	21	299681	–	876	34	17.4	299681(0)
70	280	27	498944	–	1262	102	33.5	498944(0)
70	350	32	590136	–	1348	117	42.6	590136(0)
70	420	36	472728	–	970	134	27.2	472728(0)
70	490	37	106296	932	212	33	3.84	106296(0)
70	560	41	150427	766	234	47	5.41	150427(0)
70	630	44	150442	551	221	42	5.39	150442(0)
70	700	46	101673	280	128	30	3.45	101673(0)
80	240	24	1621664	–	–	259	68.5	1621664(0)
80	320	30	2284149	–	–	409	99.8	2284149(0)
80	400	34	988166	–	–	231	42.0	988166(0)
80	480	38	751344	–	2498	239	31.2	751344(0)
80	560	41	458205	–	1372	163	18.7	458205(0)
80	640	45	608006	–	1453	201	25.5	608006(0)
80	720	48	471433	–	1106	186	19.5	471433(0)
80	800	51	438636	–	843	166	18.0	438636(0)

Table 3. Performances of our minimal separator listing algorithms with 1-hour timeout

Performances of the treewidth algorithms on random instances (1)

			PID			ASCEND			DESCEND			ALTERNATE		
$\|V\|$	$\|E\|$	tw	UB	LB	time(secs)	UB	LB	time(secs)	UB	LB	time(secs)	UB	LB	time(secs)
40	120	14	14	14	0.530	14	14	0.505	14	14	0.325	14	14	0.335
40	160	18	18	18	0.558	18	18	0.992	18	18	0.483	18	18	0.621
40	200	20	20	20	0.287	20	20	0.664	20	20	0.308	20	20	0.617
40	240	22	22	22	0.203	22	22	0.536	22	22	0.232	22	22	0.379
40	280	24	24	24	0.122	24	24	0.429	24	24	0.221	24	24	0.300
40	320	25	25	25	0.068	25	25	0.330	25	25	0.155	25	25	0.279
40	360	27	27	27	0.058	27	27	0.297	27	27	0.166	27	27	0.267
40	400	28	28	28	0.049	28	28	0.276	28	28	0.125	28	28	0.244
50	150	16	16	16	9.24	16	16	1.97	16	16	1.67	16	16	2.34
50	200	20	20	20	6.09	20	20	2.52	20	20	1.48	20	20	3.46
50	250	24	24	24	6.28	24	24	3.54	24	24	1.71	24	24	2.15
50	300	26	26	26	1.26	26	26	2.51	26	26	1.04	26	26	2.12
50	350	29	29	29	0.974	29	29	2.95	29	29	1.38	29	29	2.04
50	400	31	31	31	0.590	31	31	2.27	31	31	0.949	31	31	1.34
50	450	32	32	32	0.224	32	32	1.51	32	32	0.528	32	32	1.09
50	500	34	34	34	0.198	34	34	1.23	34	34	0.615	34	34	1.06
60	180	18	18	18	190	18	18	5.90	18	18	4.08	18	18	4.10
60	240	22	22	22	168	22	22	7.61	22	22	5.19	22	22	5.75
60	300	27	27	27	263	27	27	18.2	27	27	10.5	27	27	16.8
60	360	30	30	30	124	30	30	15.0	30	30	8.26	30	30	14.6
60	420	33	33	33	75.7	33	33	15.2	33	33	7.70	33	33	10.8
60	480	34	34	34	2.96	34	34	7.47	34	34	3.32	34	34	6.87
60	540	38	38	38	7.19	38	38	10.9	38	38	5.44	38	38	10.5
60	600	40	40	40	3.85	40	40	8.84	40	40	4.19	40	40	5.89

Performances of the treewidth algorithms on random instances (2)

|V|	|E|	tw	PID			ASCEND			DESCEND			ALTERNATE		
			UB	LB	time(secs)	UB	LB	time(secs)	UB	LB	time(secs)	UB	LB	time(secs)
70	210	22	–	22	17808	22	22	255	22	22	129	22	22	166
70	280	28	–	28	16959	28	28	749	28	28	802	28	28	1225
70	350	33	–	33	16839	33	33	806	33	33	342	33	33	432
70	420	37	–	37	8173	37	37	733	37	37	656	37	37	870
70	490	38	38	38	2278	38	38	151	38	38	66.0	38	38	204
70	560	42	42	42	3211	42	42	214	42	42	151	42	42	225
70	630	45	45	45	3052	45	45	185	45	45	137	45	45	122
70	700	47	47	47	1178	47	47	131	47	47	56.5	47	47	116
80	240	25	–	21	4812	25	25	3283	25	25	1396	25	25	1548
80	320	31	–	27	7095	31	31	6516	31	31	3167	31	31	3632
80	400	35	–	33	15647	35	35	1573	35	35	682	35	35	1781
80	480	39	–	38	15458	39	39	1377	39	39	691	39	39	1578
80	560	42	–	42	16767	42	42	813	42	42	339	42	42	650
80	640	46	–	46	19764	46	46	1150	46	46	734	46	46	1339
80	720	49	49	49	20958	49	49	934	49	49	362	49	49	1099
80	800	52	52	52	17740	52	52	842	52	52	296	52	52	559
90	270	27	–	21	6210	–	26	8755	27	–	1357	30	26	16522
90	360	35	–	27	12283	–	32	6785	35	–	109	35	32	10972
90	450	40	–	33	19743	–	37	7398	40	–	252	40	37	9870
90	540	46	–	38	14458	–	42	7300	46	–	14896	47	42	10458
90	630	49	–	43	17118	–	47	10983	50	–	17.9	50	47	12893
90	720	52	–	47	16713	–	51	10294	52	–	128	53	51	12485
90	810	55	–	51	18279	–	55	15370	55	55	15284	55	55	19642
90	900	57	–	54	11509	57	57	8684	57	57	11450	57	57	3864

6-hour time-out
time (seconds) is that of the last improvement

Performances of the treewidth algorithms on DIMACS instances

name	$\|V\|$	$\|E\|$	tw	PID			ASCEND			DESCEND			ALTERNATE		
				UB	LB	time	UB	LB	time	UB	LB	time	UB	LB	time
myciel6	95	755	35	35	35	617	35	35	3.90	35	35	1.89	35	35	3.09
myciel7	191	2360	66	–	39	10583	66	66	779	66	66	652	66	66	584
queen10_10	100	1470	72	–	70	10792	72	72	12363	72	72	3267	72	72	8224
queen11_11	121	1980	87	–	79	14199	–	81	19525	87	–	9114	88	81	20216
DSJC125.1	125	736	-	–	38	16140	–	45	15259	65	–	0.025	65	37	12758
DSJC125.5	125	3891	108	108	108	1275	108	108	581	108	108	176	108	108	590
DSJC250.1	250	3218	-	–	71	15859	–	72	19840	177	–	0.235	177	51	10598
DSJC250.5	250	15668	-	–	215	20218	–	212	20732	230	–	48.8	231	211	19167

6-hour time-out
time (seconds) is that of the last improvement

Minimal separators

$S \subseteq V(G)$ is a separator of G

if $G \setminus S := G[V(G) \setminus S]$ is disconnected

Each connected component of $G \setminus S$ is called

a component associated with separator S

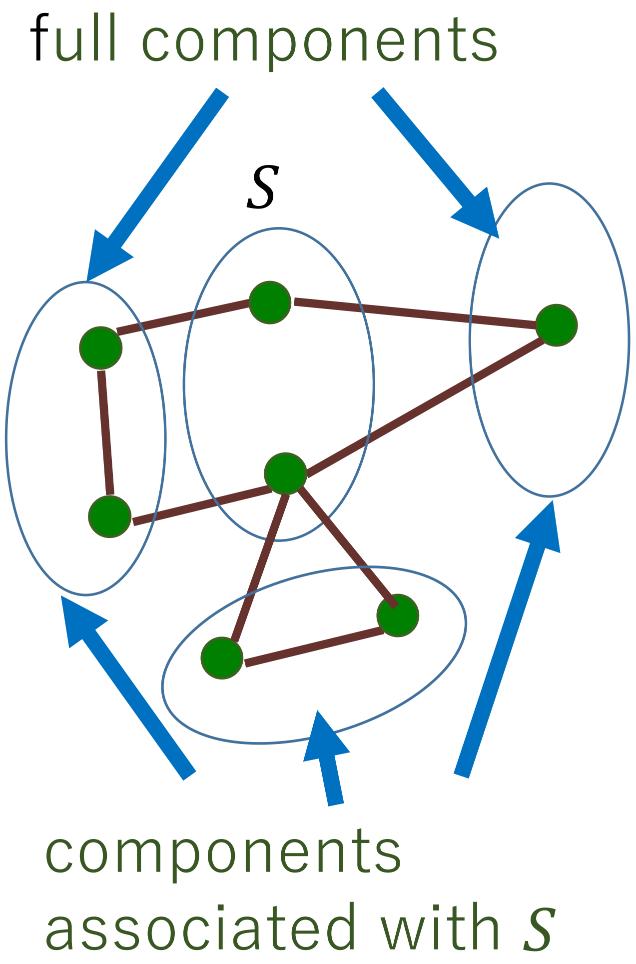
A component C associated with S is a full component

if $N(C) = S$

S is a minimal separator

if it has at least two full components associated with it

or, equivalently, if S separates a pair of vertices but no proper subset of S does not separator this pair.



Feasibility of a connected set: subproblem for DP

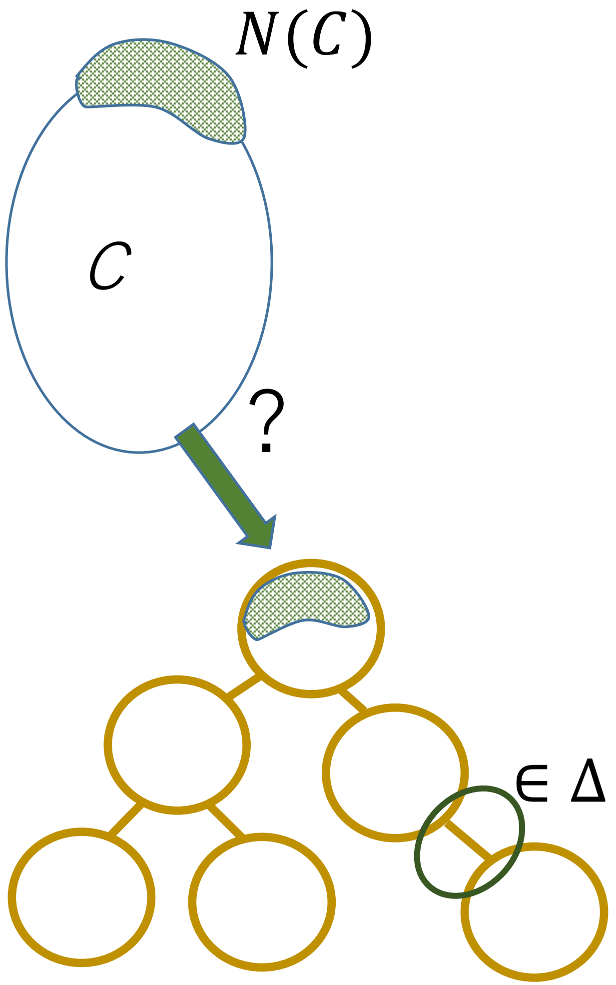
Fix G and k.

A connected $C \subseteq V(G)$ is feasible

with respect to $\Delta \subseteq \Delta_k(G)$ if

there is a tree-decomposition of $G[N[C]]$ of width $\leq k$ that

- has a bag containing $N(C)$ and
- uses separators only from Δ



Dynamic programming for treewidth

Dynamic programming of Bouchitte and Todinca 2001:

1. List minimal separators and potential maximal cliques
2. Decide the feasibility of components associated with minimal separators, through a recurrence involving potential maximal cliques

Positive instance driven (PID) variant (Tamaki 2017)

Does not list minimal separators or potential maximal cliques in advance

Generates “on the fly”

- feasible components associated with minimal separators
- potential maximal cliques needed to show their feasibility

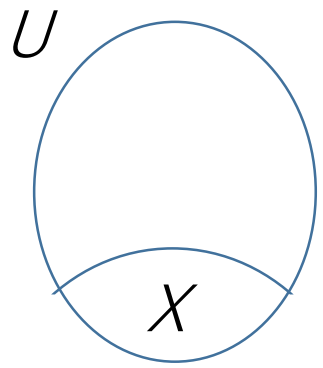
New approach

1. List minimal separators, but not potential maximal cliques
2. Decide the feasibility of components associated with minimal separators through a direct recurrence, in which potential maximal cliques are implicit

Well-formed tree-decompositions (1)

$U \subseteq V(G)$ is baggy if

- there is no connected set C such that $N(C) = U$ and
- for every non-empty $X \subseteq U$,
  - there is a connected set C containing X such that $N(C) = U \setminus X$.



Note:

A potential maximal clique is baggy, but not vice versa

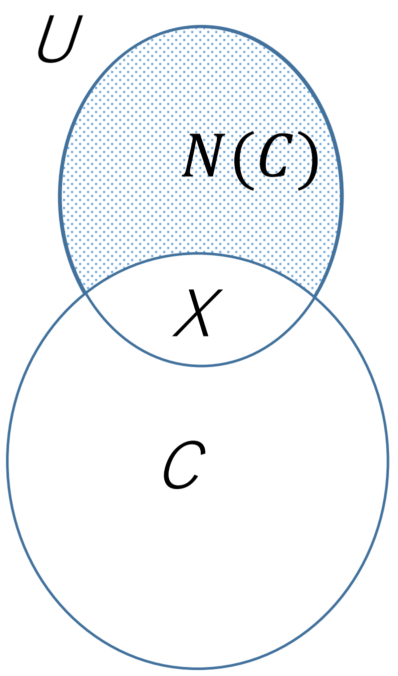
Well-formed tree-decompositions (1)

$U \subseteq V(G)$ is baggy if

- there is no connected set C such that $N(C) = U$ and
- for every non-empty $X \subseteq U$,
  - there is a connected set C containing X such that $N(C) = U \setminus X$.

Note:

A potential maximal clique is baggy, but not vice versa

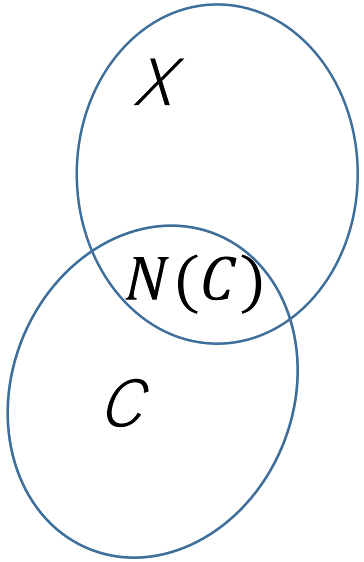


Well-formed tree-decompositions (2)

Tree-decomposition T of G is well-formed if

- every bag of T is baggy and,
- for every connected vertex set C of G such that C is a component of $G \setminus X$ for some bag X of T,
  - there is a subtree T' of T and a bag Y in T' such that
    - T' is a tree-decomposition of $G[N[C]]$,
    - Y is adjacent to a bag of T, say Z , not in T' with

    $Y \cap Z = N(C)$.



Proposition

Every graph G has a well-formed tree-decomposition of width tw(G).

Reason:

The tree-decomposition corresponding to a minimal triangulation of G is well-formed

Well-feasibility of connected sets

A connected $C \subseteq V(G)$ is well-feasible with respect to $\Delta \subseteq \Delta_k(G)$ if there is a well-formed tree-decomposition of $G[N[C]]$ that

- has a bag containing $N(C)$ and
- uses separators only from Δ

Note:

For $\Delta = \Delta_k(G)$, C is well-feasible if and only if C is feasible.

Orienting minimal separators

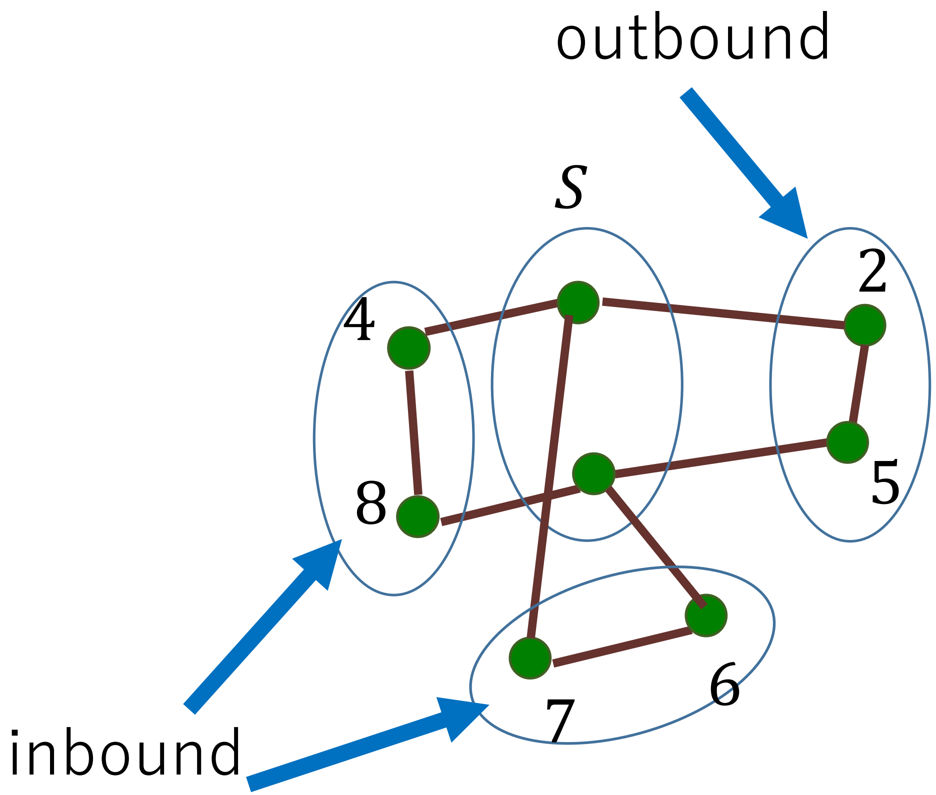
Assume a total order on the vertices:
$V(G) = \{1, 2, \dots, n\}$

Induced partial order on vertex sets:
$U < V$ if $\min U < \min V$

A connected set C is inbound if

there is a full component D associated with $N(C)$ such that $D < C$

otherwise C is outbound

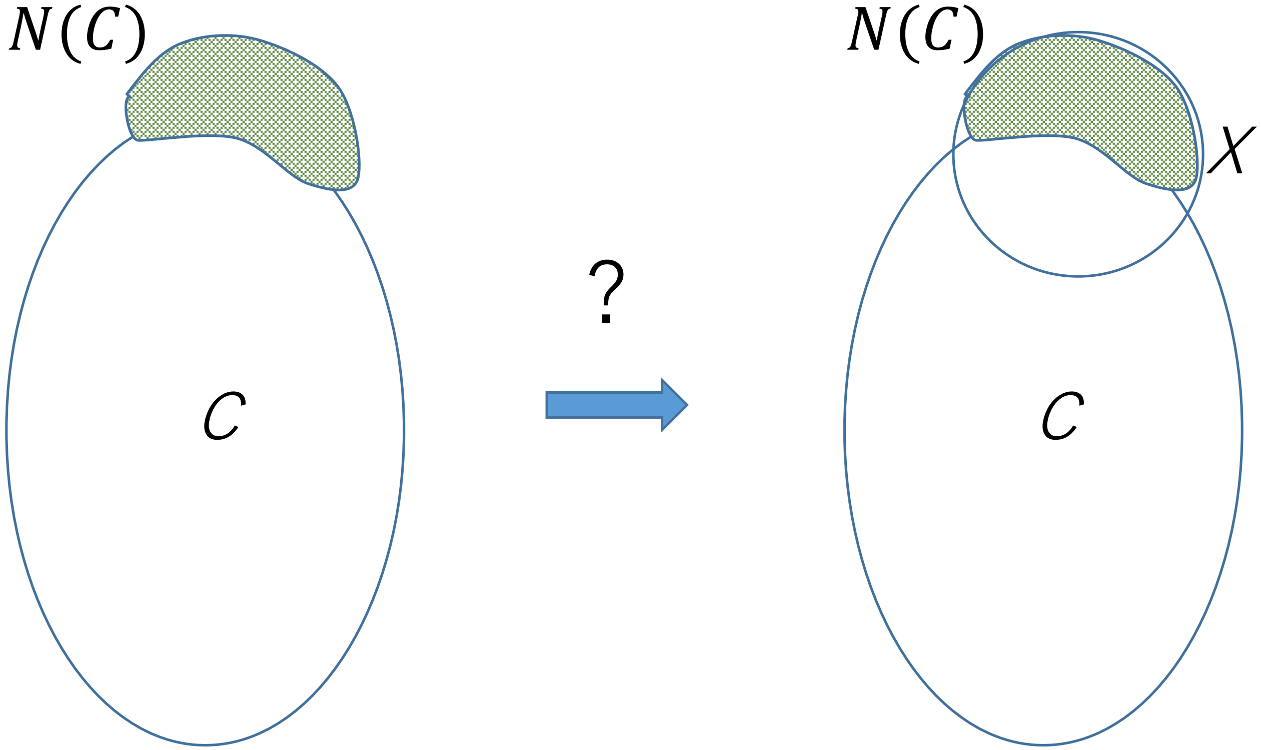


Dynamic programming algorithm (1)

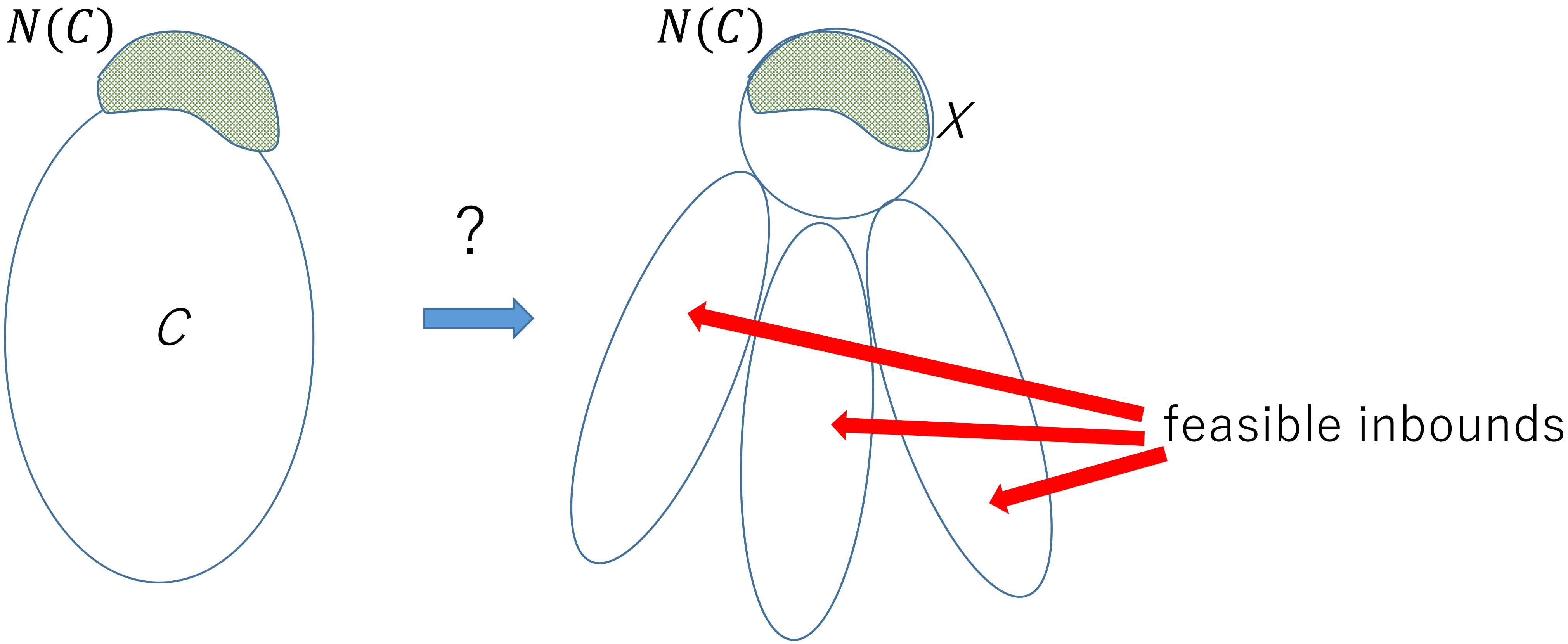
Main iteration of msDP: decides the feasibility of each inbound connected set with respect to $\Delta \subseteq \Delta_k(G)$

1: Let L be the list of all inbound connected sets C with $N(C) \in \Delta$
2: Sort L in the increasing order of cardinality
3: **for all** C in L **do**
4: **if** $\textsc{isFeasible}(C)$ **then** mark C as feasible
5: **end for**

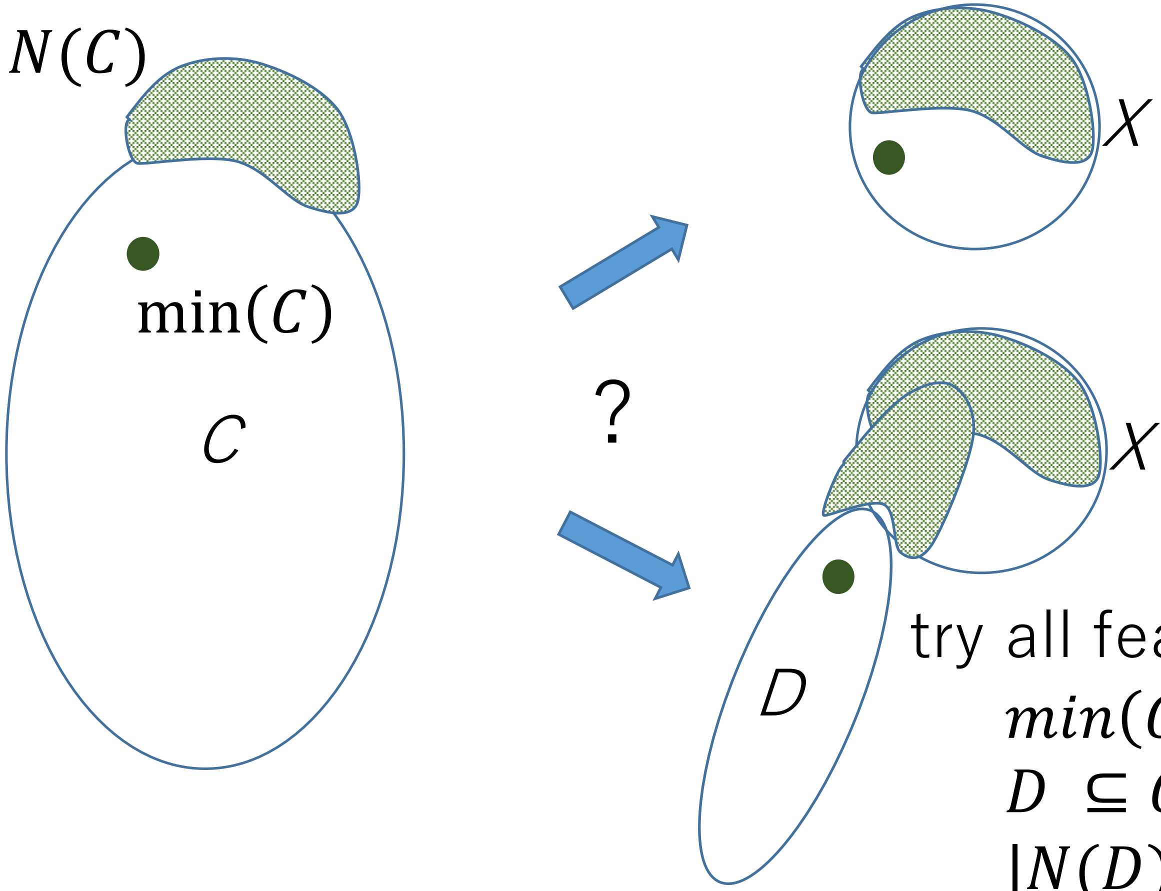
isFeasible(C)



isFeasible(C)



isFeasible(C)



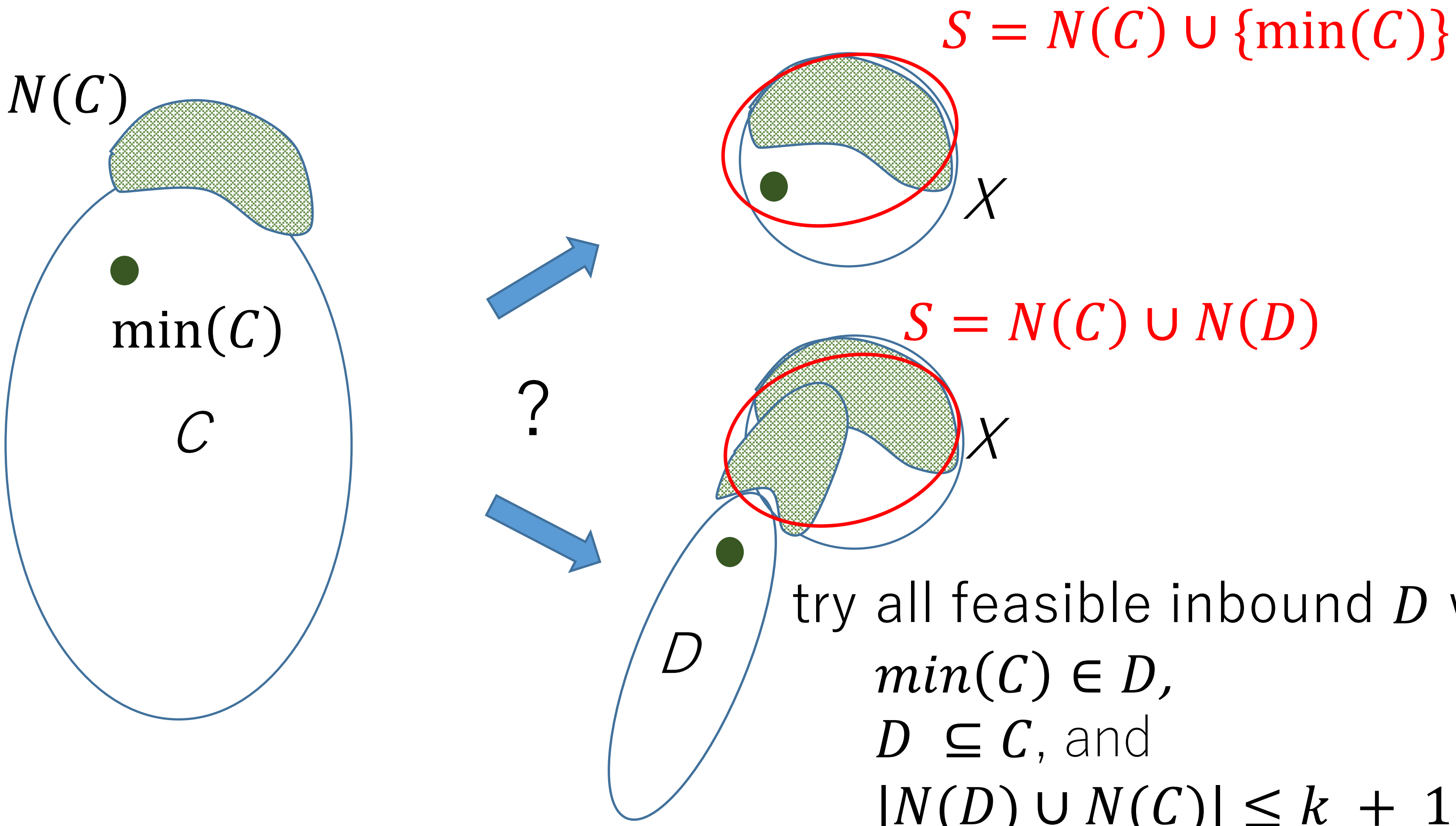
try all feasible inbound D with

$min(C) \in D$,

$D \subseteq C$, and

$|N(D) \cup N(C)| \leq k + 1$

isFeasible(C)



try all feasible inbound D with

$min(C) \in D$,

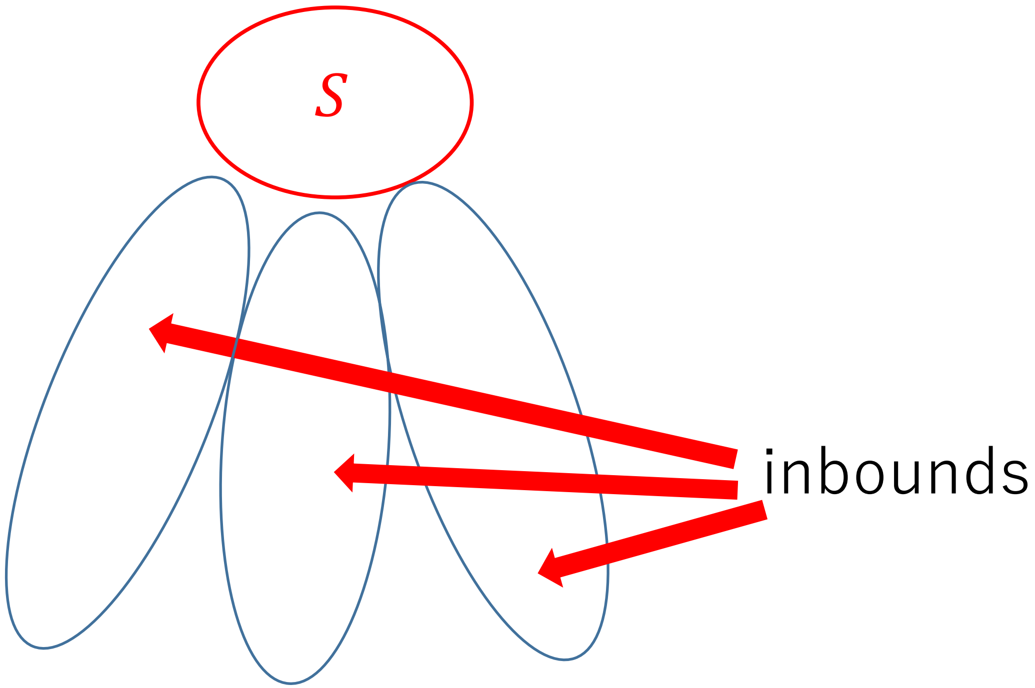
$D \subseteq C$, and

$|N(D) \cup N(C)| \leq k + 1$

isFeasible(C)

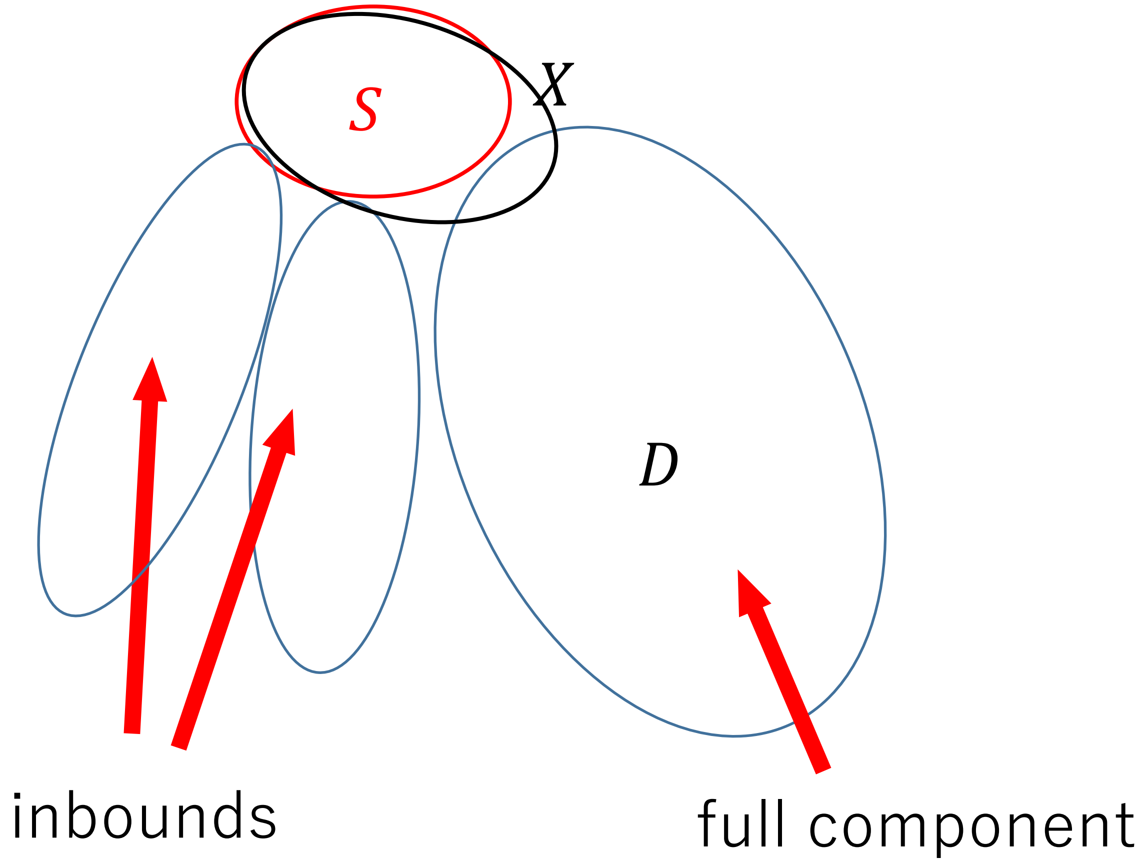
Case 1: S does not have a full component

A right bag $X = S$ is found, if all these inbounds are feasible



isFeasible(C)

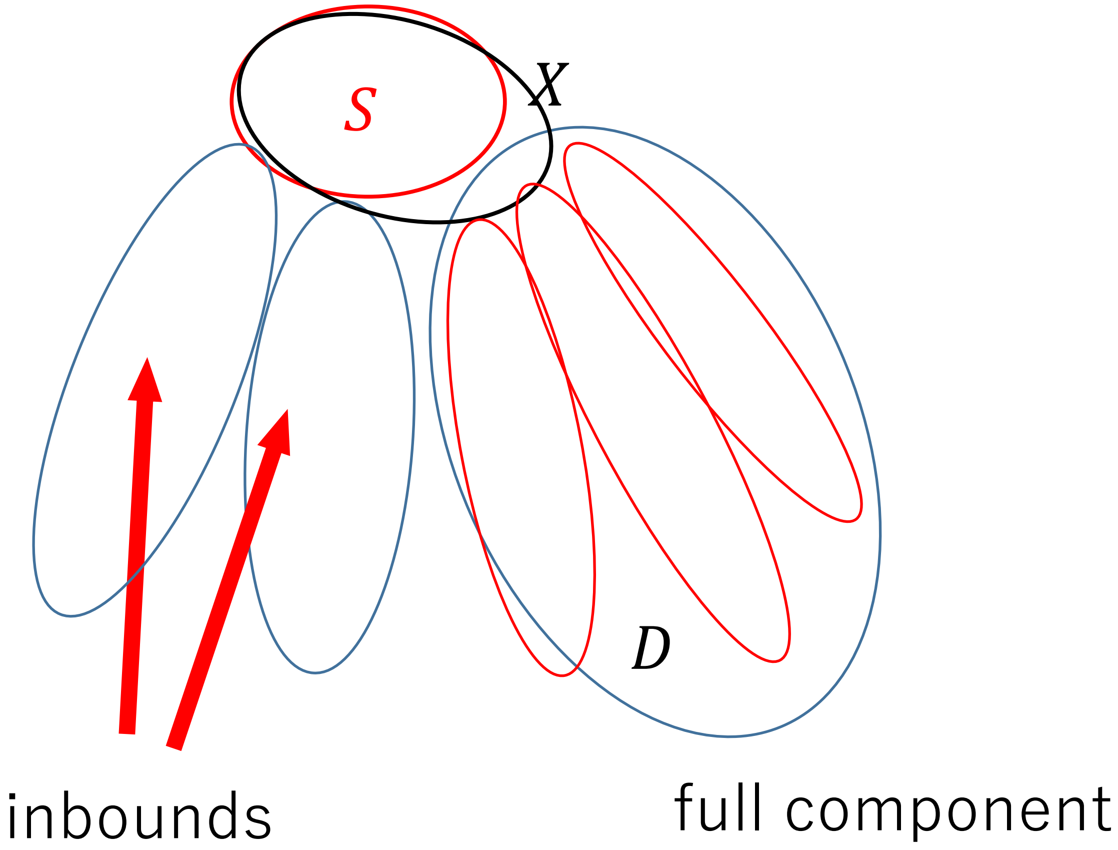
Case 2: S has a full component D



isFeasible(C)

Case 2: S has a full component D

If any of the inbounds are not marked feasible, then fail
Otherwise, call isFeasible(D)



Dynamic programming algorithm (2)

```
procedure isFeasible(C)
    if N(C) ∈ Δ and |N[C]| ≤ k + 1 then return true
    for all inbound D with min(C) ∈ D marked feasible do
        if allFeasible(N(C) ∪ N(D), C) then return true
    end for
    return allFeasible(N(C) ∪ {min(C)}, C)
end procedure
procedure allFeasible(S, C)
    if |S| > k + 1 then return false
    for all component D associated with S such that D ⊆ C do
        if N(D) = S then
            if not isFeasible(D) then return false
        else
            assert D is inbound
            if D is not marked as feasible then return false
        end if
    end for
    return true
end procedure
```

Theorem: correctness of the algorithm

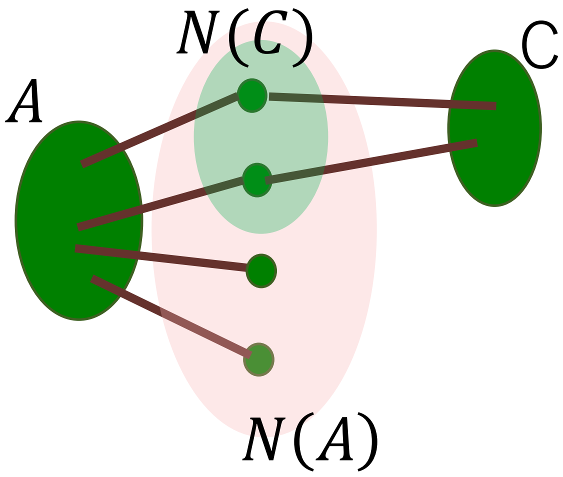
Let $C \subseteq V(G)$ be connected with $|N(C)| \leq k$.

- If call isFeasible (C) is made during the execution of our dynamic programming algorithm, and returns **true** then C is feasible with respect to Δ.
- if C is inbound with $N(C) \in \Delta$ and moreover is well-feasible with respect to Δ, then the algorithm marks C as feasible.

Structure of $\Delta(G)$: the set of all minimal separators of G

Let $A \subseteq V(G)$ be connected.

For each component C of $G \setminus A$, $N(C)$ is a minimal separator, called a minimal separator close to A

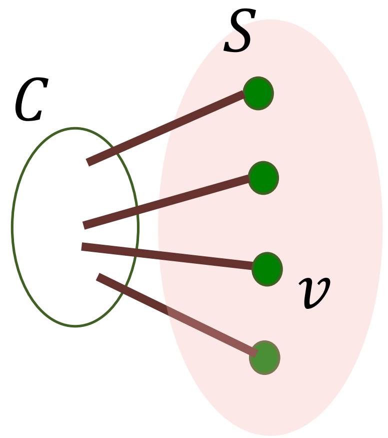


Digraph $\Lambda(G)$ on $\Delta(G)$:

has an edge from S to R if and only if R is a separator close to $C \cup \{v\}$ for some full component C of S and $v \in S$.

Theorem [Berry et al. 2000]

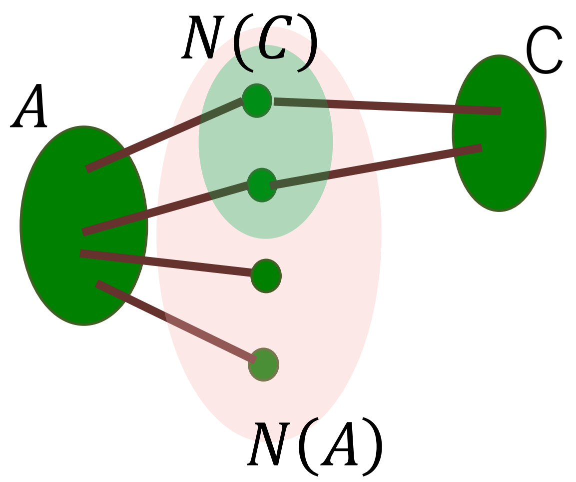
Every minimal separator of G is reachable in $\Lambda(G)$ from a separator close to a singleton



=> Algorithm for listing all minimal separators in $O(n^3)$ time per each.

Structure of $\Delta(G)$: the set of all minimal separators of G

Let $A \subseteq V(G)$ be connected.
For each component C of $G \setminus A$, $N(C)$ is a minimal separator, called a minimal separator close to A

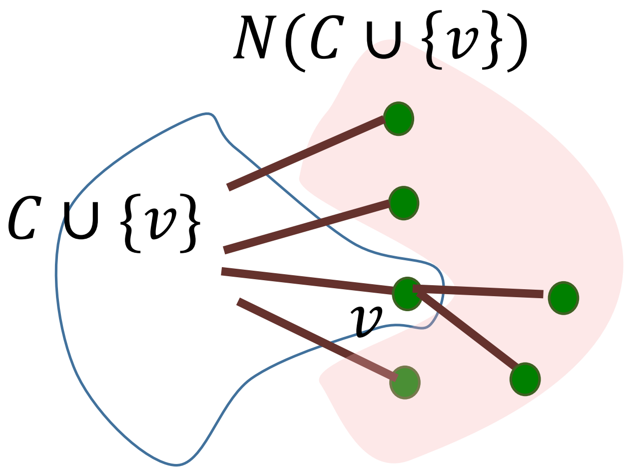


Digraph $\Lambda(G)$ on $\Delta(G)$:

has an edge from S to R if and only if R is a separator close to $C \cup \{v\}$ for some full component C of S and $v \in S$.

Theorem [Berry et al. 2000]

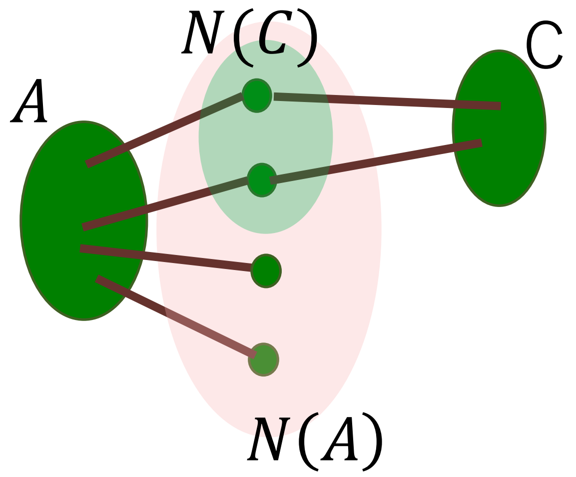
Every minimal separator of G is reachable in $\Lambda(G)$ from a separator close to a singleton



=> Algorithm for listing all minimal separators in $O(n^3)$ time per each.

Structure of $\Delta(G)$: the set of all minimal separators of G

Let $A \subseteq V(G)$ be connected.
For each component C of $G \setminus A$, $N(C)$ is a minimal separator, called a minimal separator close to A

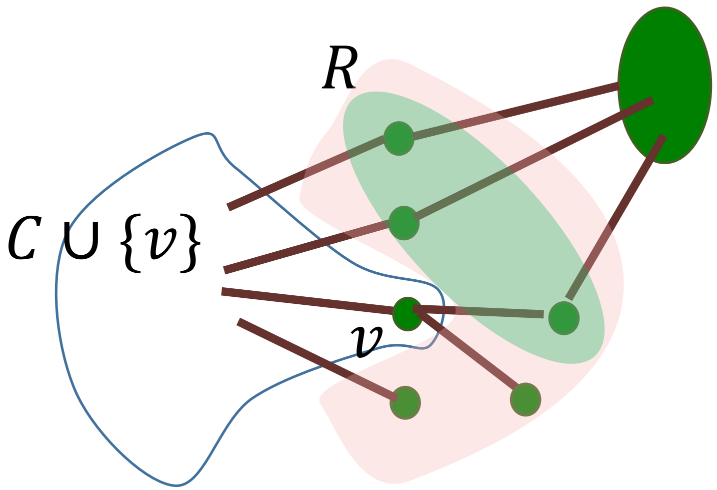


Digraph $\Lambda(G)$ on $\Delta(G)$:

has an edge from S to R if and only if R is a separator close to $C \cup \{v\}$ for some full component C of S and $v \in S$.

Theorem [Berry et al. 2000]

Every minimal separator of G is reachable in $\Lambda(G)$ from a separator close to a singleton



=> Algorithm for listing all minimal separators in $O(n^3)$ time per each.

Generating $\Delta_k(G)$

Based on Takata's algorithm for generating $\Delta(G)$

- backtrack search through $\Lambda(G)$
- pick S only if $|S| \le k$
- exploit the cardinality constraint for pruning

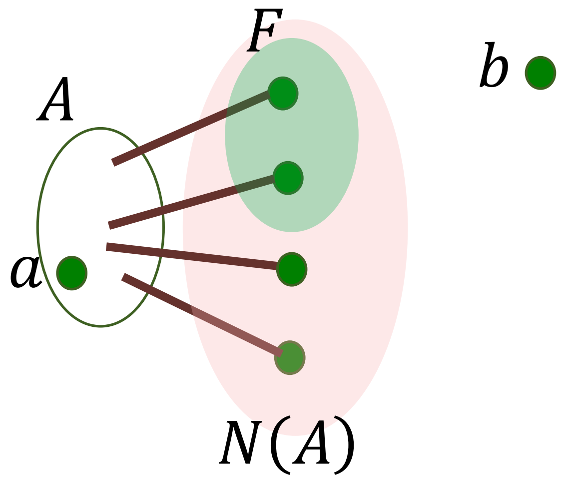
For $A \subseteq V(G)$ connected such that

$N[A]$ is an a-b minimal separator and

$F \subseteq N(A)$

define

$\Delta_{ab}(A, F)$: the set of all a-b minimal separators S satisfying $\cdots$



Generating $\Delta_k(G)$

For $A \subseteq V(G)$ connected such that

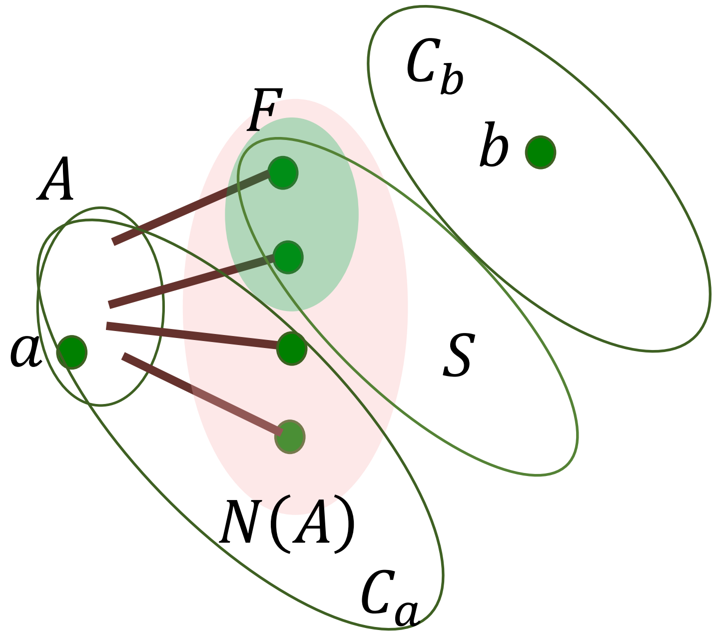
$N[A]$ is an a-b minimal separator and

$F \subseteq N(A)$

define

$\Delta_{ab}(A, F)$: the set of all a-b minimal separators S satisfying

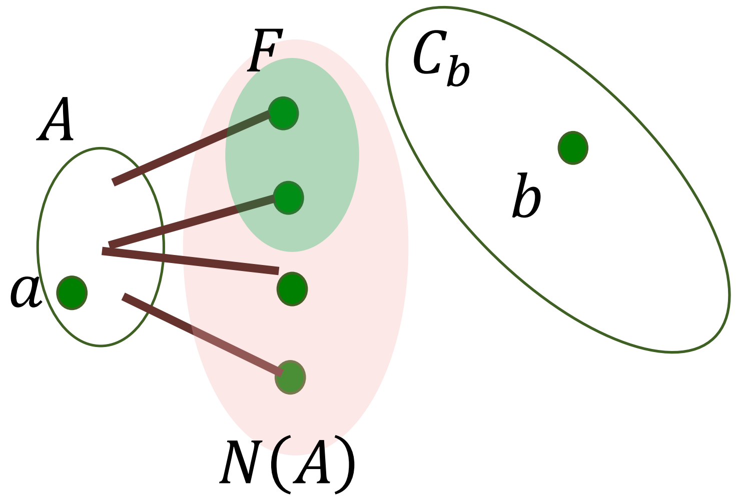
- $|S| \leq k$
- $F \subseteq S$
- The component C_a of $G \setminus S$ containing a contains A
- The component $C_b$of $G \setminus S$ containing b is disjoint from $N[A]$
- $a = \min(C_a)$and $b = \min(C_b)$
- $|C_a| \leq |C_b|$



Base cases for $\Delta_{ab}(A, F)$

$\Delta_{ab}(A, F)$ is empty if either

- $|S| > F$
- $F \setminus N(C_b) \neq \emptyset$, where C_b is the component of $G \setminus N(A)$ containing b
- $\min(A) \neq a$ or $\min(C_b) \neq b$
- $|F| = k$ and $N(A) \neq F$
- $|A| > |C_b|$
- $|N(A)| > k$ and $|A| + (|N(A)| - k) > \min\{|C_b|, \frac{(|V(G)| - k)}{2}\}$



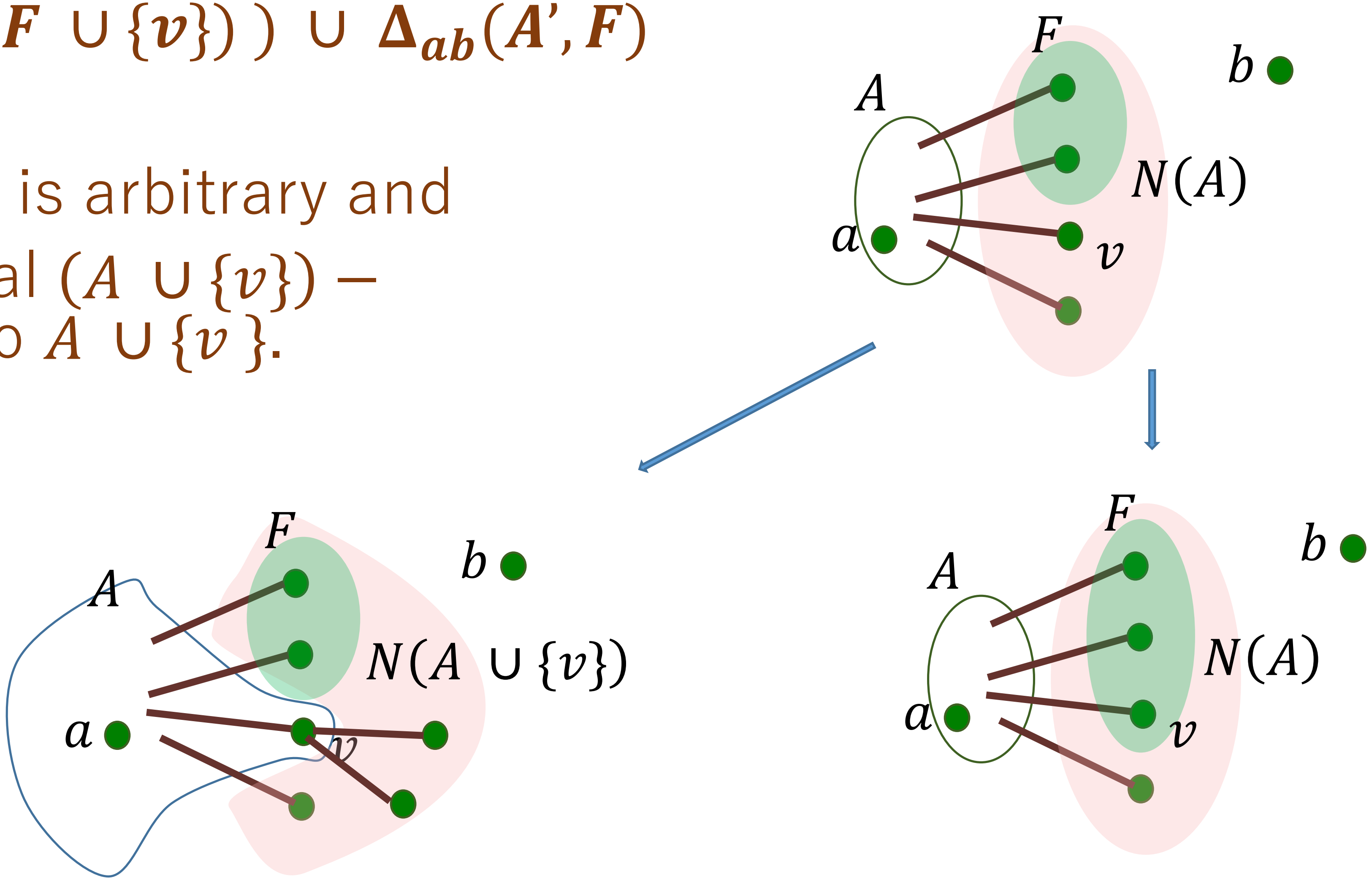
$\Delta_{ab}(A, F) = \{F\}$ if

$N(A) = N(C_b) = F, |F| \leq k$, and $|A| \leq |C_b|$

Recurrence for $\Delta_{ab}(A, F)$

$$\boldsymbol{\Delta_{ab}(A,F) = \Delta_{ab}(A, F \cup \{v\})\,) \cup \Delta_{ab}(A', F)}$$

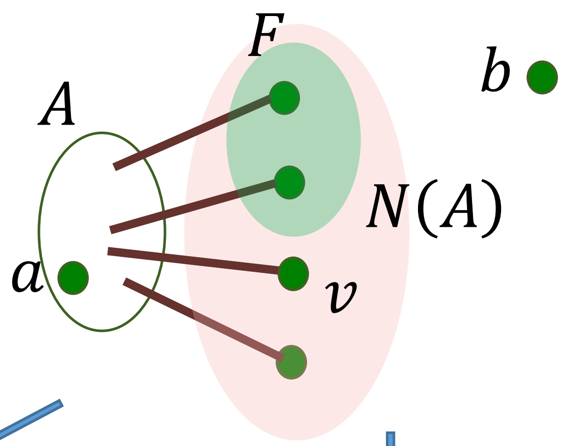
where $\boldsymbol{v} \in N(A) \setminus F$ is arbitrary and $N(A')$ is the minimal $(A \cup \{v\}) - b$ separator close to $A \cup \{v\}$.

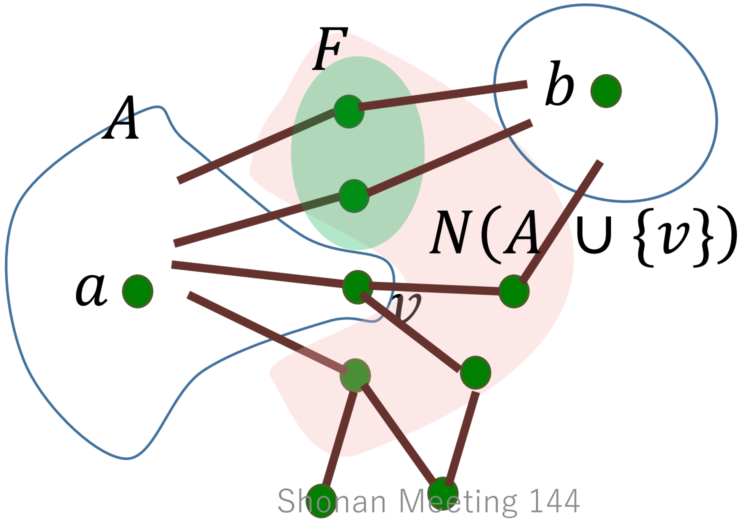


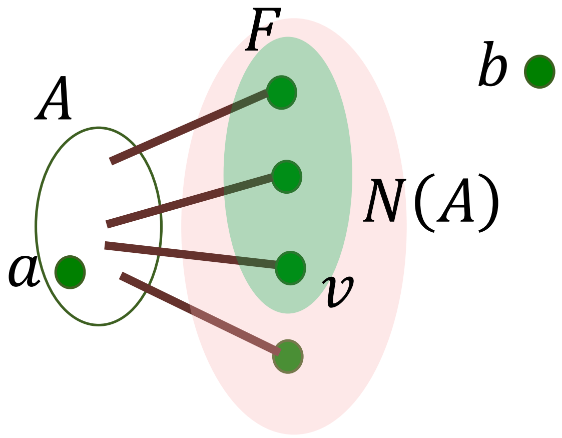
Recurrence for $\Delta_{ab}(A, F)$

$$\boldsymbol{\Delta_{ab}(A, F) = \Delta_{ab}(A, F \cup \{v\}) \cup \Delta_{ab}(A', F)}$$

where $\boldsymbol{v} \in N(A) \setminus F$ is arbitrary and $N(A')$ is the minimal $(A \cup \{v\}) - b$ separator close to $A \cup \{v\}$.



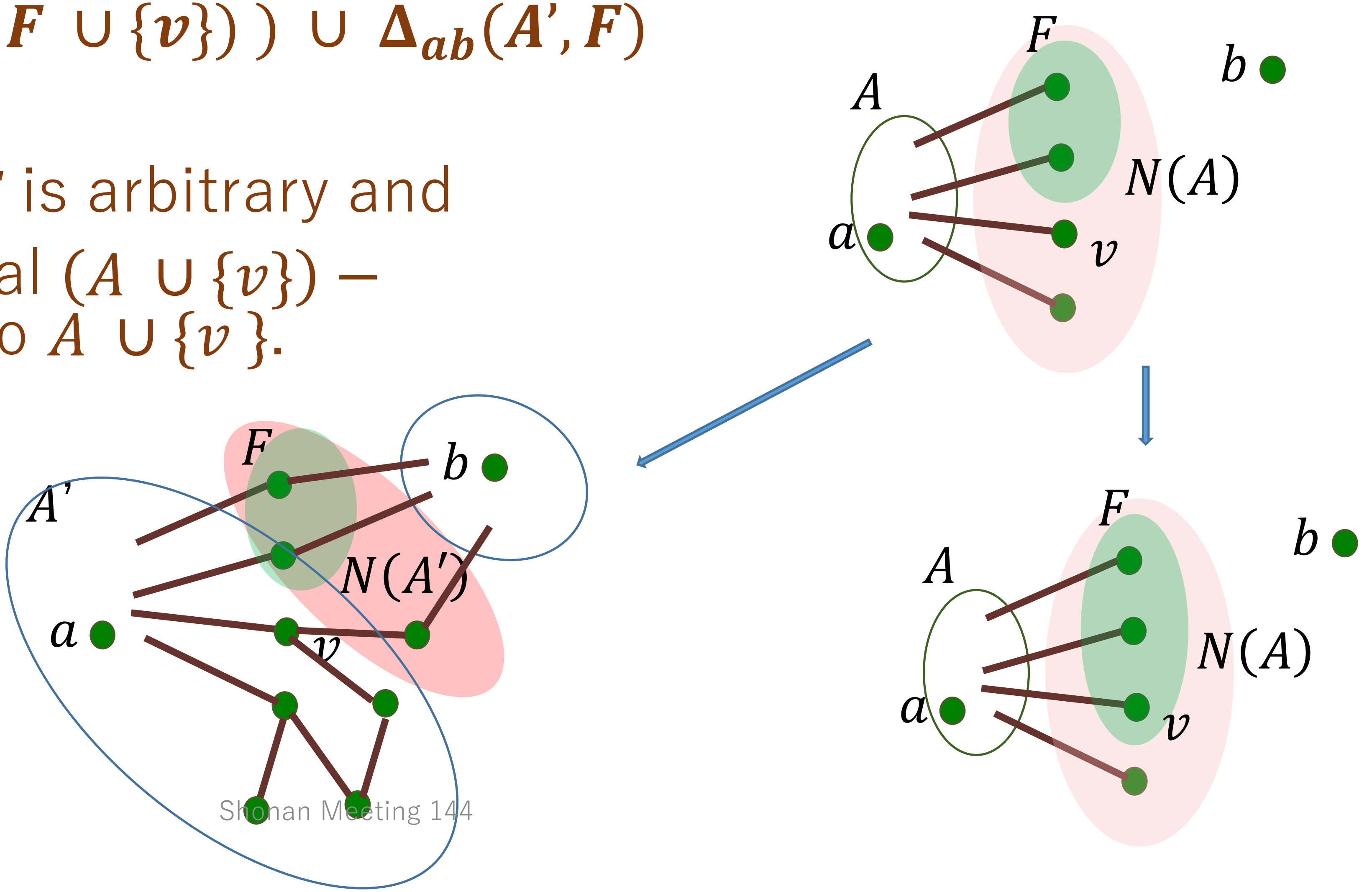




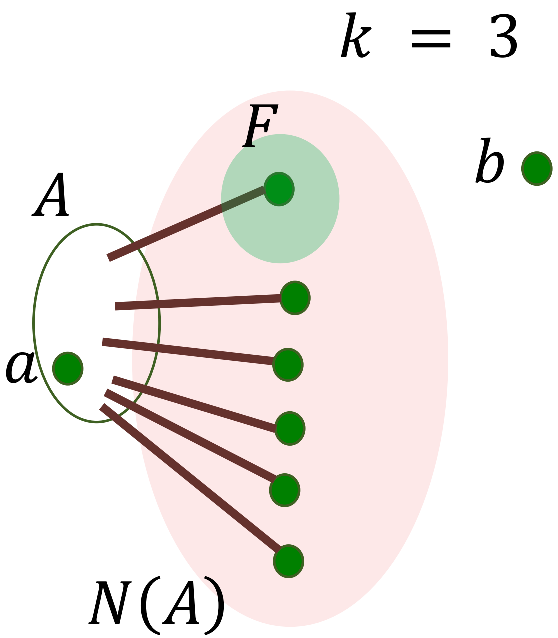
Recurrence for $\Delta_{ab}(A, F)$

$$\boldsymbol{\Delta_{ab}(A, F) = \Delta_{ab}(A, F \cup \{v\})\,) \cup \Delta_{ab}(A', F)}$$

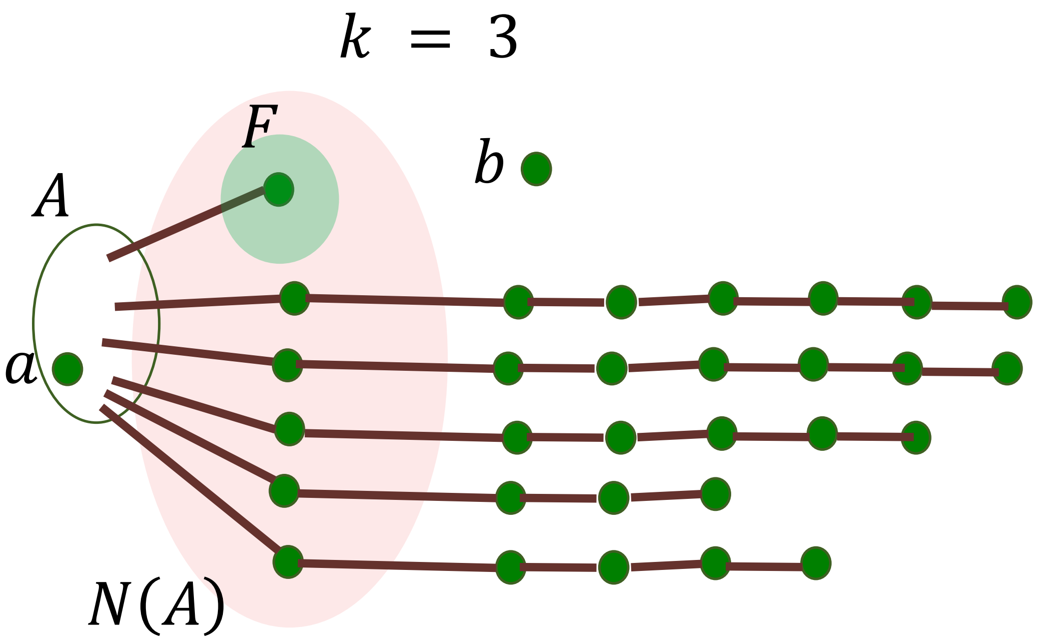
where $\boldsymbol{v} \in N(A) \setminus F$ is arbitrary and $N(A')$ is the minimal $(A \cup \{v\}) - b$ separator close to $A \cup \{v\}$.



Pruning

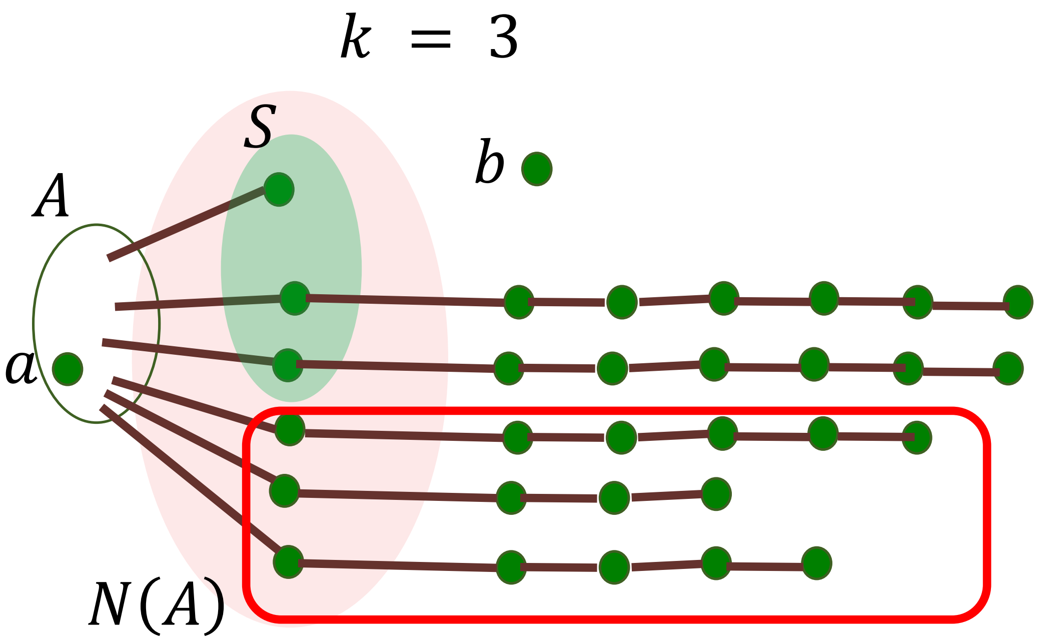


Pruning



Vertex disjoint paths from $N(A) \setminus F$ through $V(G) \setminus N[A]$

Pruning



Vertex disjoint paths from $N(A) \setminus F$ through $V(G) \setminus N[A]$

This many (15) vertices must be added to the a-side of any minimal separator in $\Delta_{ab}(A, F)$

$\Delta_{ab}(A, F)$ is empty if $|A| + 15$ is too large for a smaller of the two full components of a minimal separator

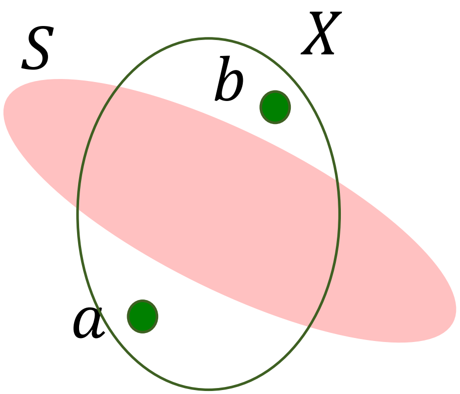
How to iterate through (a, b) pairs?

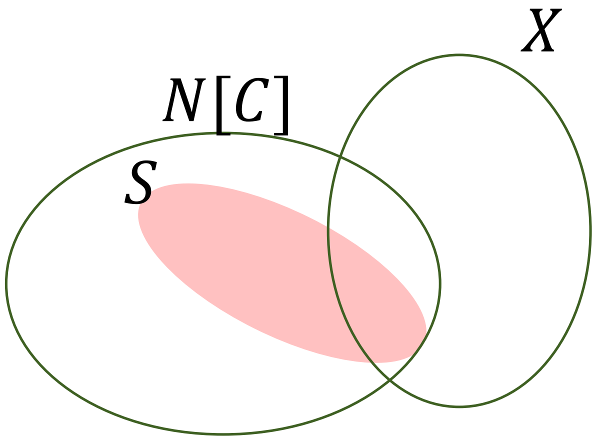
Observation:

X: an arbitrary separator of G

A minimal separator S of G either

crosses X or is *local* to some component C of $G \setminus X$





Divide and conquer?

$\Delta_k(G)$ is the union of

- Δ_{ab} for each pair $a, b \in X$
- $\Delta_k(H)$ for local graph H that arises from each component of $G \setminus X$

Should X be a balanced separator?

Divide and conquer?

$\Delta_k(G)$ is the union of

- Δ_{ab} for each pair $a, b \in X$
- $\Delta_k(H)$ for local graph H that arises from each component of $G \setminus X$

Should X be a balanced separator?

NO (experimental observation)

Having small $|X|$ or small number of non-adjacent (a, b) pairs in X is much more important!

Choose X to be $N(v)$ with the smallest number of non-adjacent pairs: v is a minfill vertex

Nibling

Heuristic listing

- Used to improve an upper bound of treewidth
- Can assume we have a tree-decomposition T of width $\geq k + 1$
- The initial set Δ^0 consists of minimal separators each of which is local to some bag of T
- The expansion $\Delta^{\{i+1\}}$of Δ^i is obtained by adding the successors of S in $\Lambda(G)$ that are in $\Delta_k(G)$, for each $S \in \Delta^i$

Thank you !