HOKKAIDO UNIVERSITY

NII 大学共同利用機関法人 情報・システム研究機構 国立情報学研究所 National Institute of Informatics

There is an correction in the proceedings paper.
Please download this slide from this URL.
http://www-ikn.ist.hokudai.ac.jp/~k-kurita/slide/IWOCA2018_errata.pdf

Efficient Enumeration of Subgraphs and Induced Subgraphs with Bounded Girth

○ Kazuhiro Kurita[1], Kunihiro Wasa[2], Alessio Conte[2],
Takeaki Uno[2], Hiroki Arimura[1]

1: Hokkaido University, Japan
2: National Institute of Informatics, Japan

Errata

There was an error in our paper in the proceedings on the time complexity in the proposed algorithm EBG-IS for enumeration of induced subgraph with bounded girth in section 4 at Lemma 8 at page 9 at Theorem 2 at page 9.

Lemma 8 at page 9	The statement of Lemma 8
Corrected	We can compute $D^{(2)}(S(Y))$ from $D^{(2)}(S(X))$ in $O(\|E(G[N[S(Y)]])\| \cdot \|C(S(Y))\|)$ time.
Wrong	We can compute $D^{(2)}(S(Y))$ from $D^{(2)}(S(X))$ in $O(\#gch(Y) \cdot \|S(Y)\| + \|C(S(Y))\|^2)$ time

Theorem 2 at page 9	The statement of Theorem 2, Note: $\mathcal{S}$ is the set of all solutions.
Corrected	EBG-IS enumerates all solutions in $O(\Sigma_{S \in \mathcal{S}} \|E(G[N[S]])\|) = O(\Sigma_{S \in \mathcal{S}} \Delta \|S\|)$ time using $O(\max_{S \in \mathcal{S}}\{\|N[S]\|^3\})$ space
Wrong	EBG-IS enumerates all solutions in $O(\Sigma_{S \in \mathcal{S}} \|N[S]\|)$ time using $O(\max_{S \in \mathcal{S}} \{\|N[S]\|^3\})$ space

Overall, the algorithm EBG-IS and it's correctness do not change at all.

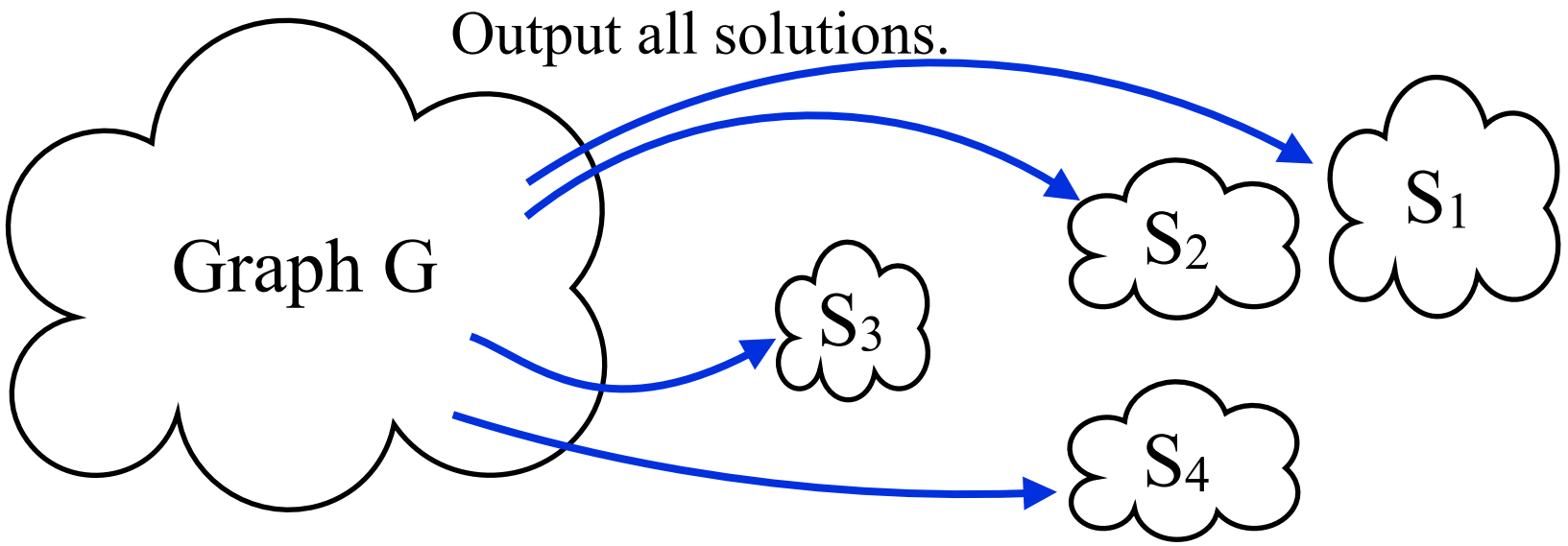
Only analysis and time complexity changes form $O(|N[S]|)$ to $O(E(G(N[S])) = O(\Sigma_{S \in \mathcal{S}} \Delta |S|)$ by a factor of $O(\Delta)$, where Δ is the maximum degree of the graph.

Introduction

A subgraph enumeration problem is defined as follows.

Input: A graph G and constraint R

Task: output all subgraphs satisfying R.

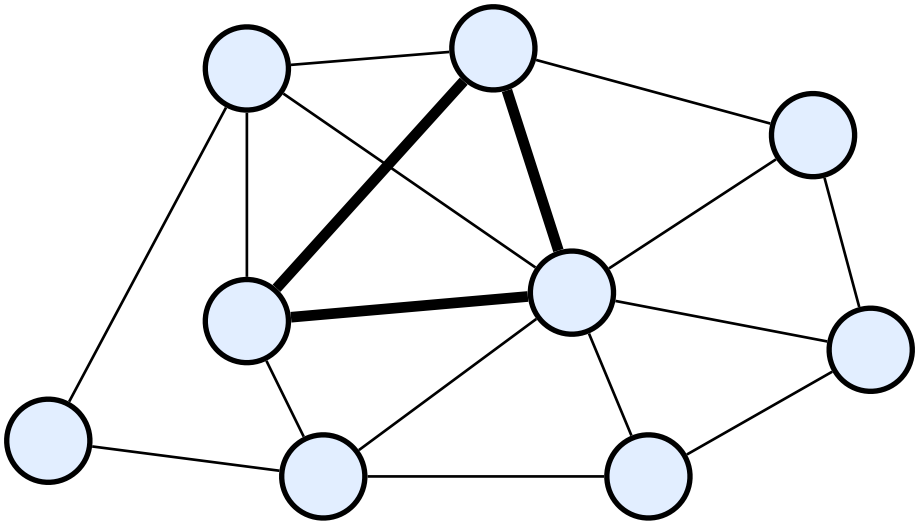


Our problems: connected (induced) subgraph enumeration

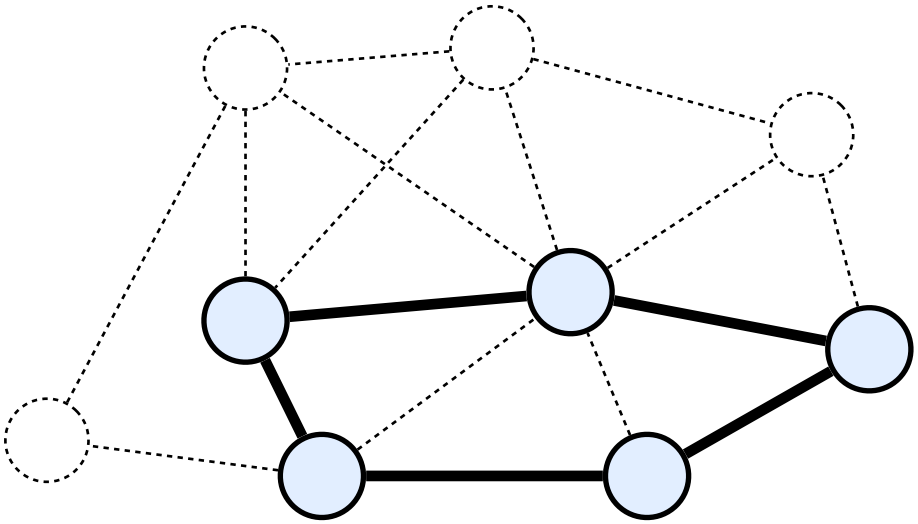
Our problems: connected (Induced) subgraph enumeration with bounded girth.

Input: A graph G and an integer k.

Task: Enumerate all connected (induced) subgraphs with girth at least k.



Girth k = 3

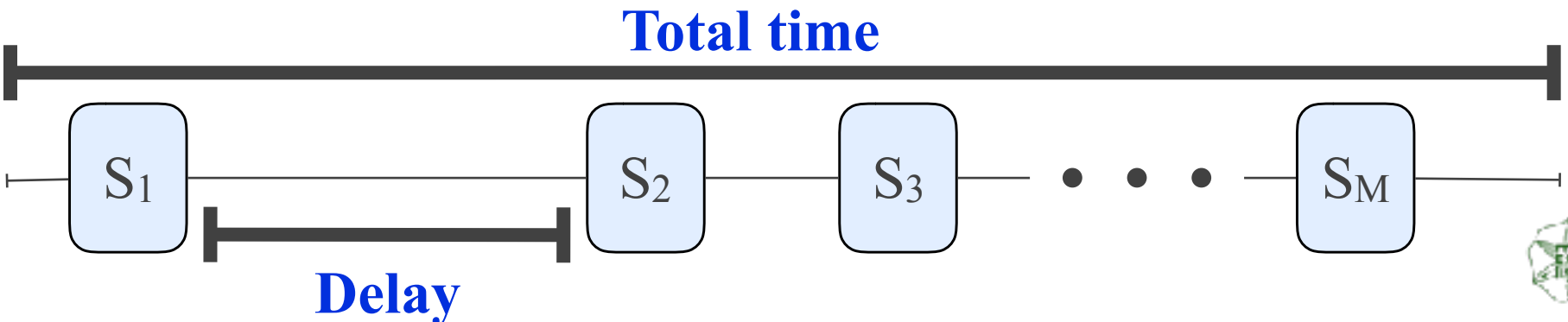


Girth k = 5

How to evaluate efficiency of an enumeration algorithm

We evaluate the efficiency of an enumeration algorithm with respect to the input size N and the output size M.

	total time
polynomial delay algorithm	Time between i-th solution and i + 1-th solution (delay) is **O(poly(N))** time.
amortized polynomial time algorithm	Total time is **O(M poly(N)).**



Main results

Main results	Time complexity (S is the set of solution.)	Space usage (S is the set of solution.)
Enumeration of induced subgraphs with bounded girth (EBG-IS)	corrected: $T = O(\Sigma_{S \in \mathcal{S}} \lvert E(G[N[S]]) \rvert)$ total time wrong: $T = O(\Sigma_{S \in \mathcal{S}} \lvert N[S] \rvert)$ total time	$O(\max_{S \in \mathcal{S}} \{\lvert N[S]^3 \} \rvert) = O(n^3)$
Enumeration of subgraphs with bounded girth (EBG-S)	$T = O(\Sigma_{S \in \mathcal{S}} \lvert V(G[S]) \rvert)$ total time	$O(\max_{S \in \mathcal{S}} \{\lvert V(G[S])^3 \} \rvert) = O(n^3)$

$|V(G[S])|$ is smaller than or equal to n.

Time complexity of EBG-IS: $T = O(\Delta |S|) = O(m)$ amortized time.

Time complexity of EBG-S: $T = O(|S|) = O(n)$ amortized time.

Related works

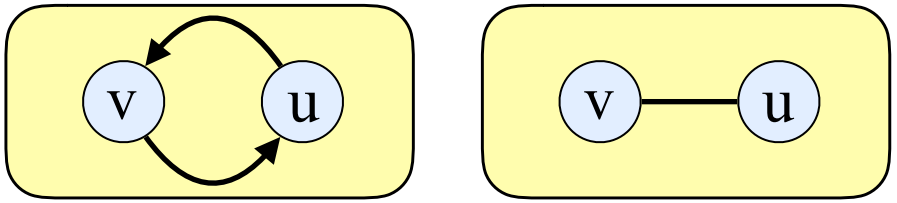
Finding a shortest cycle problem in graphs

- O(nm) time algorithm [1].
- O(n) time algorithm for planar graphs [2].

Enumerate subgraphs and induced subgraphs with bounded girth in a directed graph.

- O(n) delay [3].

We remark that the techniques in [3] **do not extend to undirected graphs**. In directed graphs, a path from u to v and a path from v to u are distinct. However, such paths may be same in undirected graphs



1. A. Itai and M. Rodeh. Finding a minimum circuit in a graph. SIAM Journal on Computing, 7(4):413–423, 1978.
2. H.-C. Chang and H.-I. Lu. Computing the girth of a planar graph in linear time. SIAM Journal on Computing, 42(3):1077–1094, 2013.
3. C. Alessio, K. Kurita, K. Wasa, T. Uno.Listing acyclic subgraphs and subgraphs of bounded girth in directed graphs. In Proc., COCOA 2017, volume 10628 of LNCS, pages 169–181. Springer International Publishing, 2017.

The basic algorithm

Key point: Binary partition

Binary partition

The basic algorithm recursively partitions the solution space into some subspaces. If the size of solution space becomes exactly one, the algorithm outputs the solution. Otherwise, the algorithm petitions the subspace.

The algorithm divides the solution space as follows.

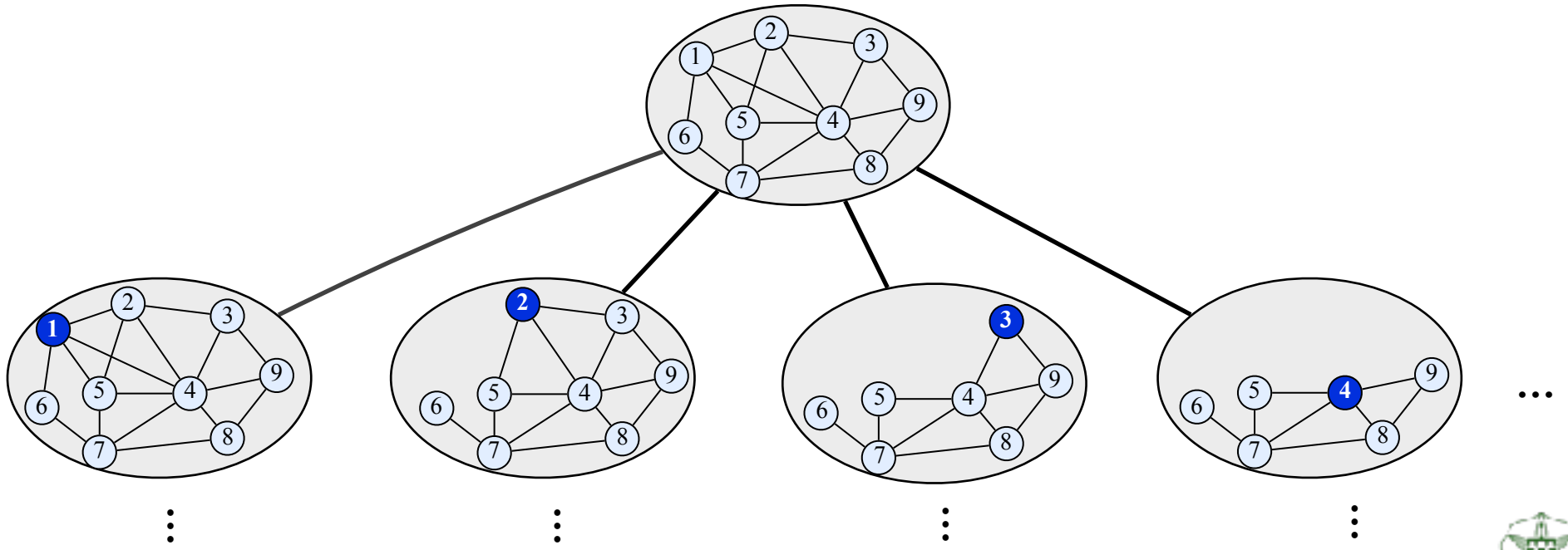


Fig 1. How to enumerate all solutions.

A candidate set

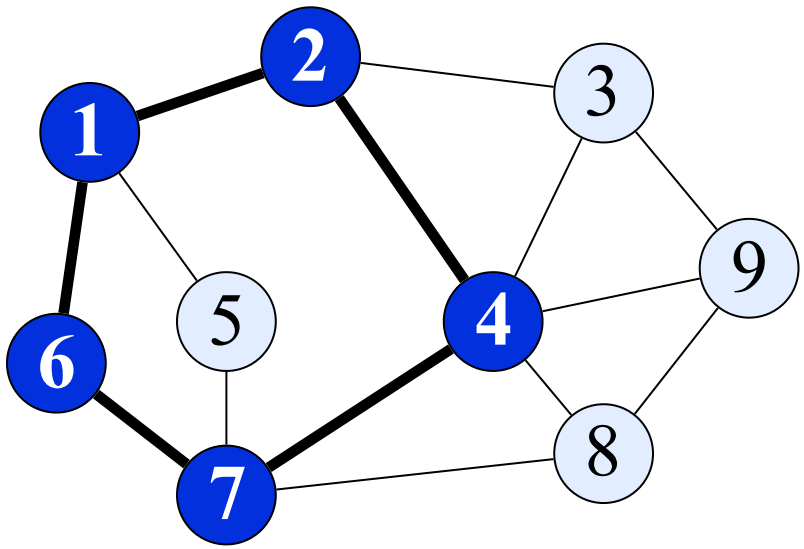
We define **a candidate set C** for a solution S.

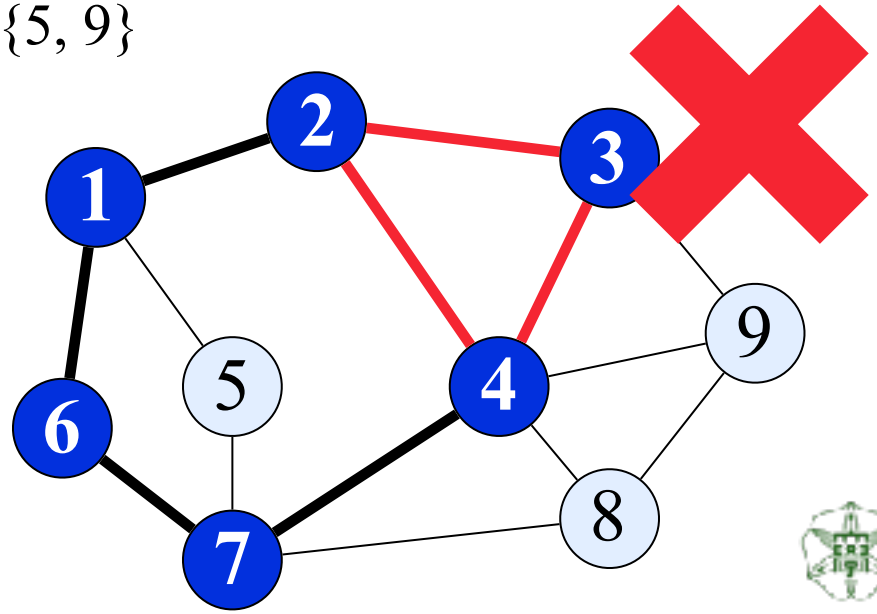
- $C = \{v \in V | g(G[S \cup \{v\}]) \geq k$ and $G[S \cup \{v\}]$ is connected. $\}$

g(G) means the girth of G.

It implies that $S \cup \{v\}$ is a solution for any vertex v in C.

If $S = \{1, 2, 4, 6, 7\}$ and $k = 4$, then $C = \{5, 9\}$





Analysis of the basic algorithm

It is known that the girth of a graph can be computed in $O(nm)$ time. Hence, we can update C in $O(n^2m)$ time.

Since the size of a recursion tree is equal to the number of solution, this algorithm outputs all solutions in $O(n^2m)$ time per solution.

In each transition, the basic algorithm needs $O(n^2m)$ time.

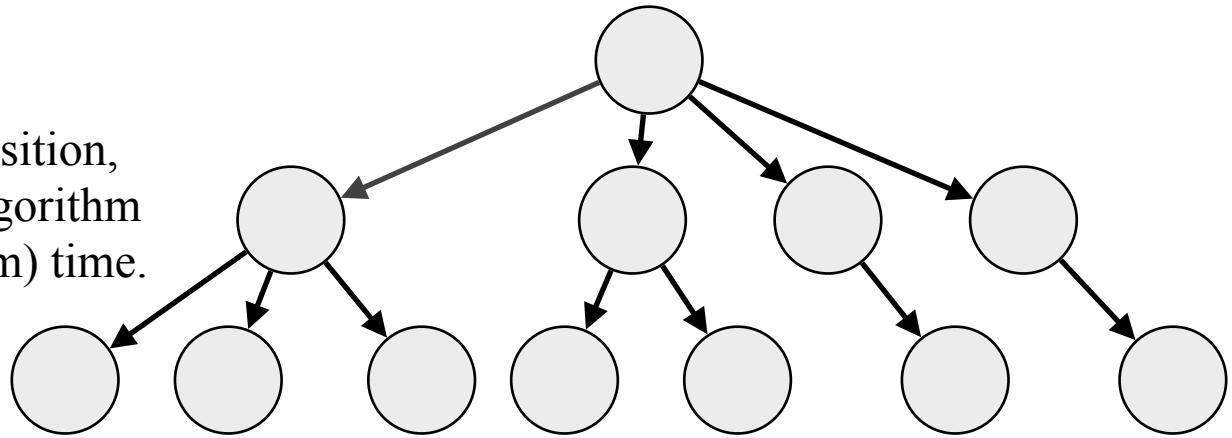


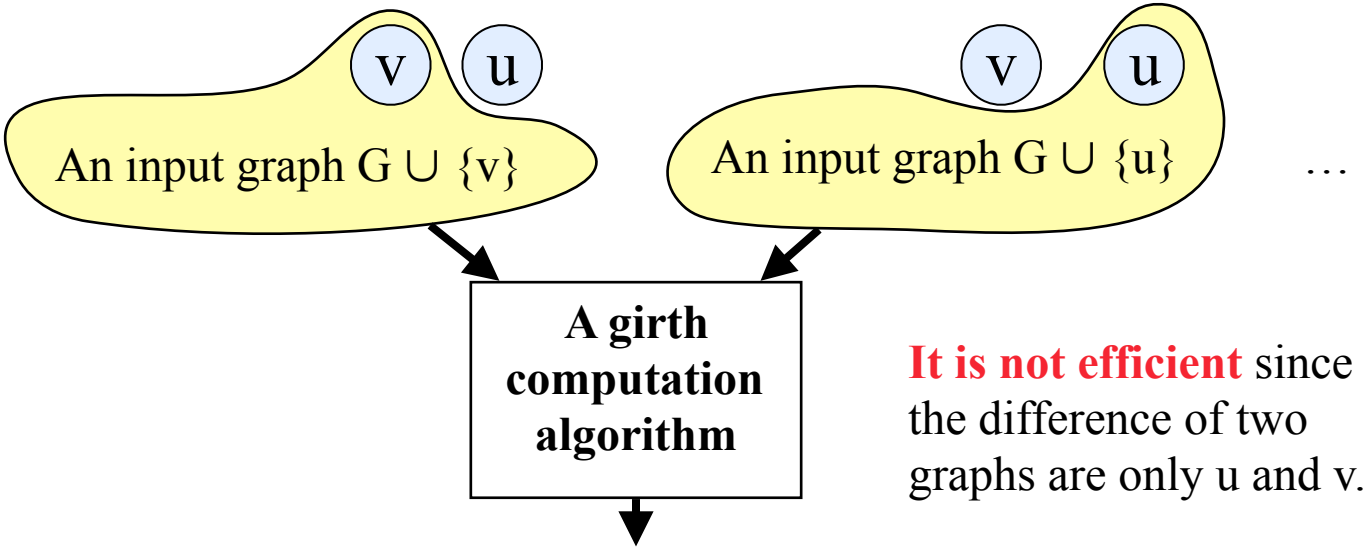
Fig 2. A recursion tree made by the basic algorithm.

Out line of efficient enumeration algorithm

The bottle neck of the basic algorithm is **the update of a candidate set C**.

In the basic algorithm, we use a girth computation algorithm many times. It is not efficient since the difference of an input graph is very small in our problem.

To update girth efficiently, **we use the two distance matrices**.



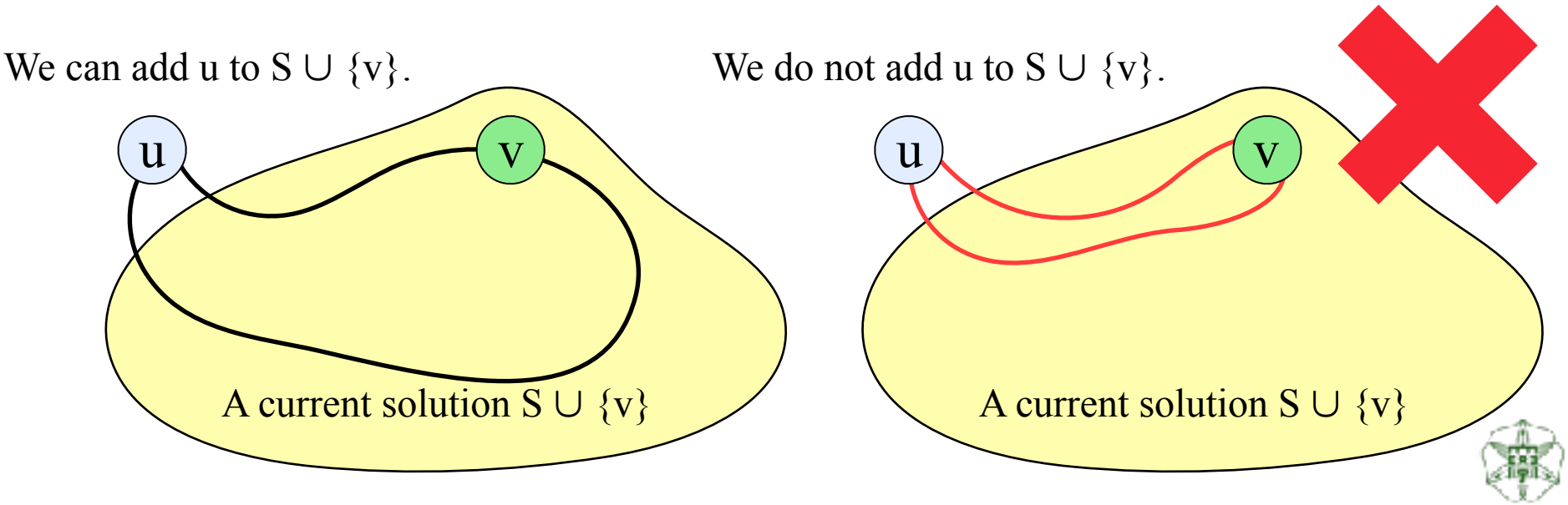
Efficient Enumeration of Induced Subgraphs with Bounded Girth

Key point: Using distance matrices

How to update a candidate set C

Let v be a vertex added to S. We have to update a candidate set C. If we do not add a vertex u in G[S ∪ {v, u}], then there is a cycle with the length less than k and including u and v.

Hence, we want to find a shortest cycle including u and v.

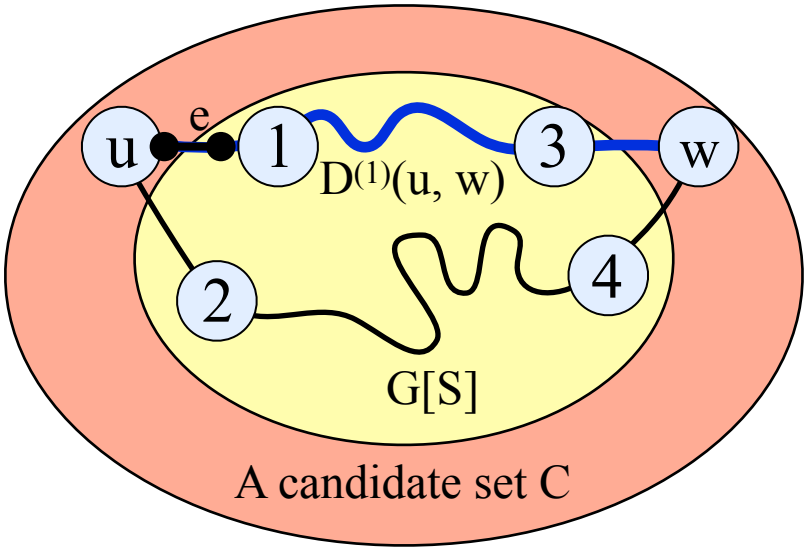


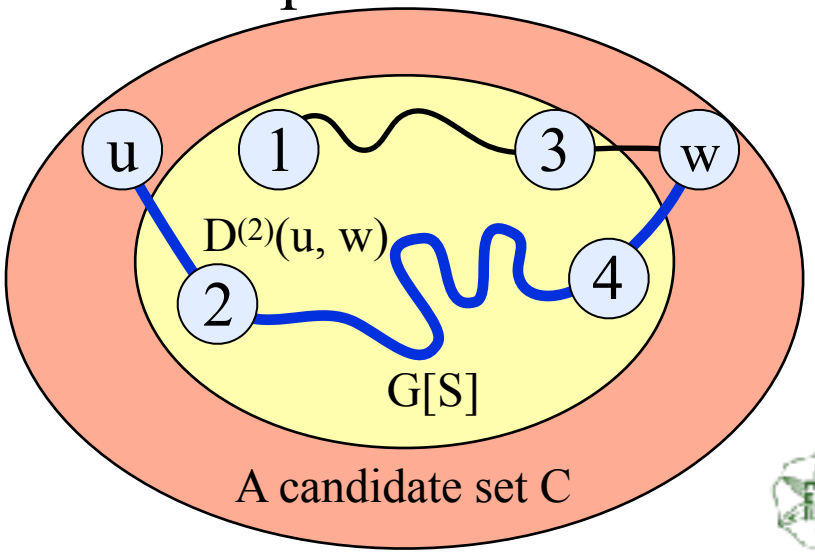
How to update a candidate set C

For any $u \in C$, to find a shortest cycle include v and u, we use **two distance matrices $D^{(1)}$ and $D^{(2)}$**.

- For any $u \in C \cup S$ and $w \in C$, $D^{(1)}(u, w)$ is the distance between u and w in $G[S \cup \{u, w\}]$.

- For any $u \in C$ and $w \in C$, $D^{(2)}(u, w)$ is the distance between u and w in $G[S \cup \{u, w\}] \setminus \{e\}$, where e is an edge incident to u and is included by a shortest path.



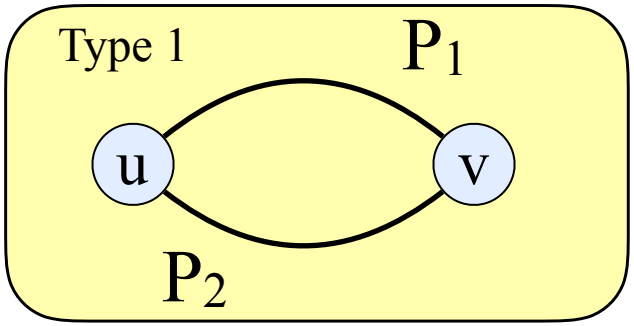


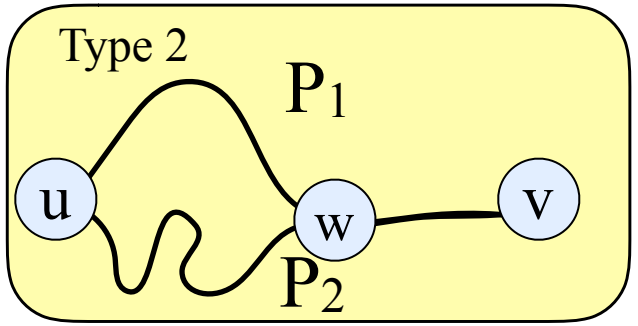
Update of a candidate set by using $D^{(1)}$ and $D^{(2)}$

We consider two paths from u to v. One path is a shortest path P_1, the other path P_2 has the length $D^{(2)}(u, v)$.

We consider two types.

- **Type 1**: P_1 and P_2 are disjoint without u and v.
- **Type 2**: P_1 and P_2 have common vertex w.





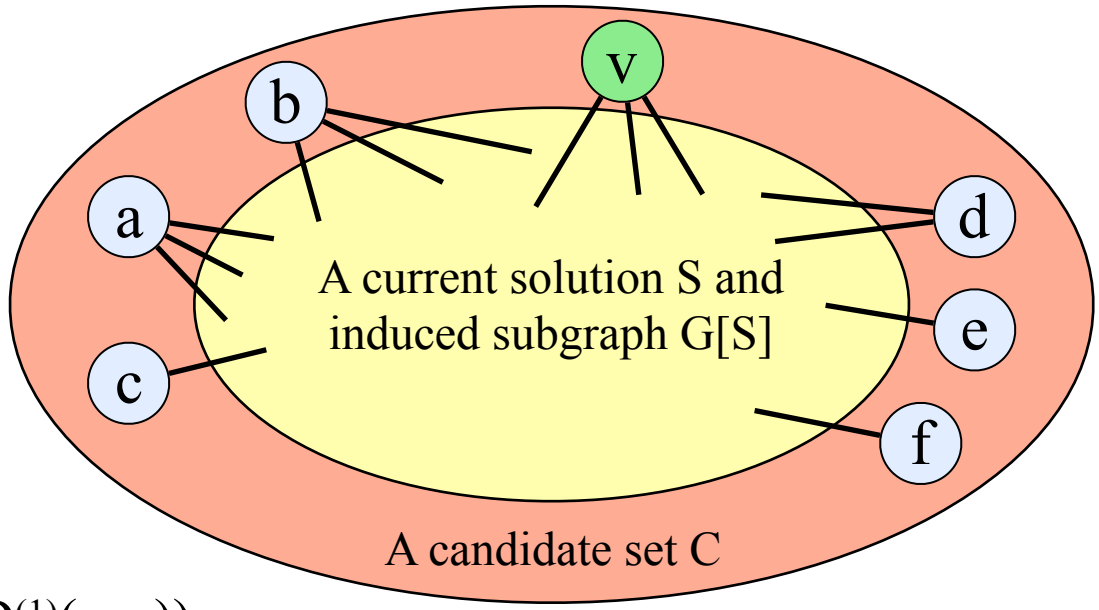
It is sufficient to consider type 1.

Since we have $D^{(1)}$ and $D^{(2)}$, we can compute $S \cup \{u, v\}$ is solution or not in constant time.

Update of $D^{(1)}$

Let $D^{(1)}$' and $D^{(2)}$' be the distance matrix for $G[S \cup \{v\}]$.

Since we have the old distance matrices $D^{(1)}$, we can compute $D^{(1)}$' in $O(|D^{(1)}'|) = O(|C'| \times |C' \cup S|)$ time by using Floyd-Warshall algorithm.



$D^{(1)}$'(b, c) is equal to
$\min(D^{(1)}(b, c), D^{(1)}(b, v) + D^{(1)}(v, c))$.

Update of $D^{(2)}$

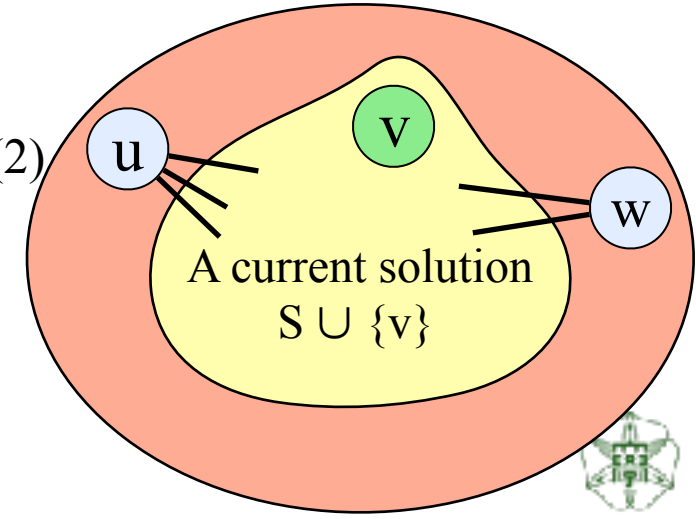
To update of $D^{(2)}$', when we update $D^{(2)}$'(u, w), we check all neighbor of u and select second minimum length in those length.

We can compute the distance between $x \in N(u)$ and v in constant time by using $D^{(1)}$'.

Hence, we can update $D^{(2)}$'(u, w) in |N(u)| time, and the total time of the update of $D^{(2)}$ is | C'| times the number of edges crossing C' and $S \cup \{v\}$.

Corrected: Total time of the update of $D^{(2)}$

$$O(|C'| \times E(G[N[S']]))$$



Error in proceedings:

$$\#gch(Y) \times |S(Y)| + |C(S(Y))|^2$$

Analysis of the time complexity

The total time of this algorithm is as follows:

- $O(|C| + |C'| \times |C' \cup S| + |E(G[N[S']])| \times |C'|)$ time

Since each internal node in a recursion tree T has $O(|C'|)$ children, the time complexity is reduces as follows:

- $O(|C| + |C' \cup S| + |E(G[N[S']])|)$ time

$O(|C| + |C'| \times |C' \cup S| + |E(G[N[S']])| \times |C'|)$ time

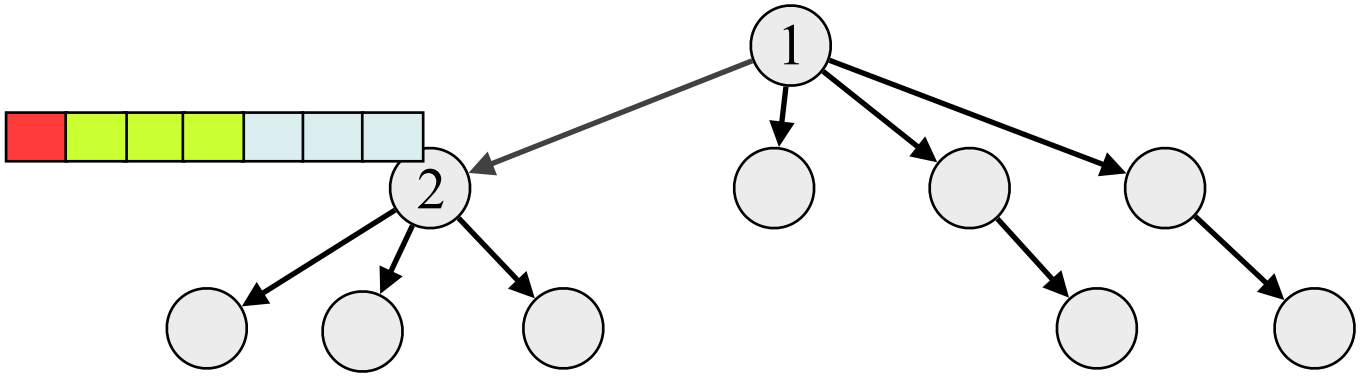


Fig 3. A recursion tree made by a basic algorithm.

Analysis of the time complexity

Each internal node has only one red box, green box, and blue box. Hence, the time complexity is as follows:

- $O(|C| + |C' \cup S| + |E(G[N[S']])|)$ time

$|C|$ and $|C' \cup S|$ are at most $O(N[G[S]])$. Hence, the time complexity is $O(|E(G[N[S]])|) = O(\Delta |S|)$ amortized time.

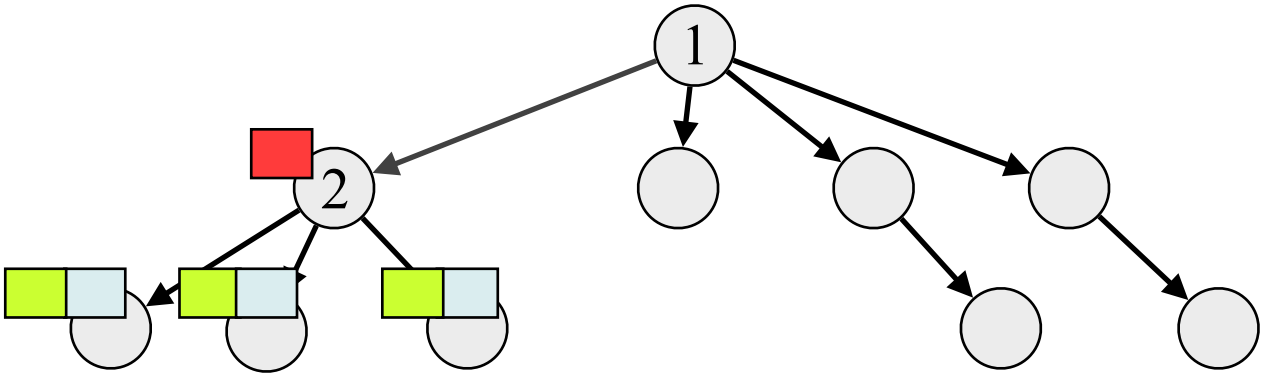


Fig 3. A recursion tree made by a basic algorithm.

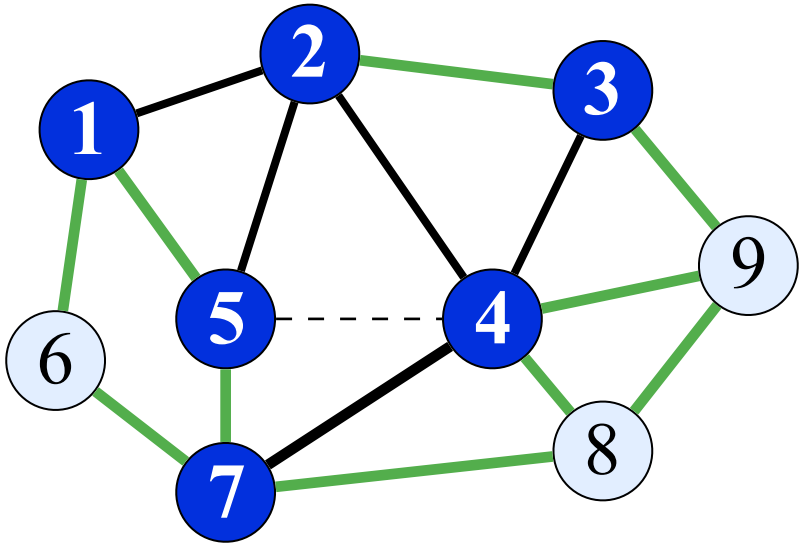
Efficient Enumeration for Subgraphs with Bounded Girth

Key point: the ordering of selecting edges

The Difficulty of amortized O(n) time Enumeration

In enumeration of subgraphs with bounded girth, it is easy to find a shortest cycle including an edge e = {u, w}.

However, it is difficult to update a candidate set in O(n) time since the size of a candidate set is at most m.



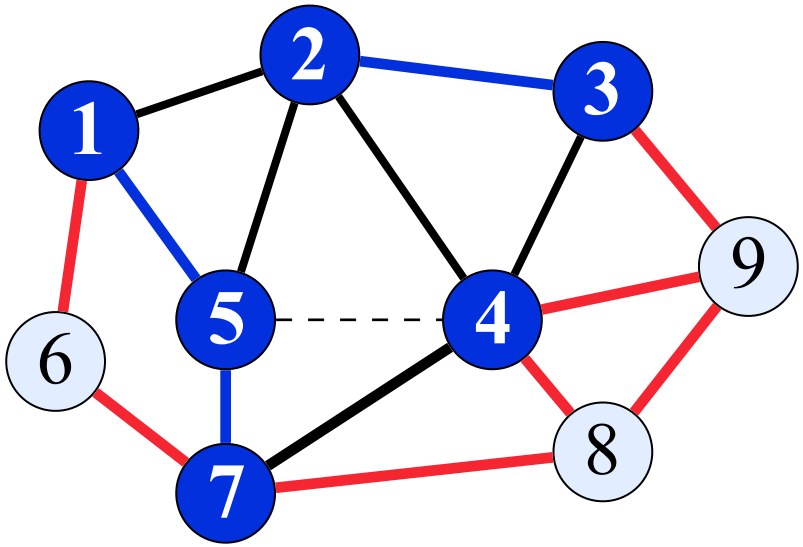
Black edge: solution, Green edge: C

Our main strategy

To bound the maximum size of C, we subdivide C into C_{in} and C_{out} as follows.

- $C_{in} = \{\{u, v\} \in C \mid u, v \in V(G[S])\}$
- $C_{out} = C / C_{in}$

If the algorithm adds an edge to a solution S, then C_{out} does not change by the definition of C_{out} and the property of girth.



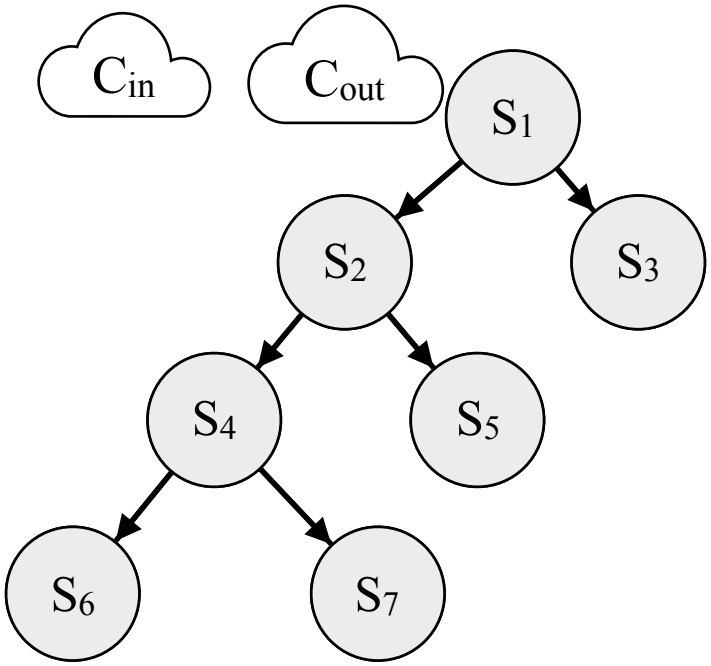
Black edge: solution, Blue edge: C_{in}, Red edge: C_{out}

The size of a candidate set

We consider an enumeration tree T.

In each node in T, the size of C_{in} is monotonically decreasing if C_{in} is not empty set.

If C_{in} is empty set, then the size of C_{in} becomes at most Δ.



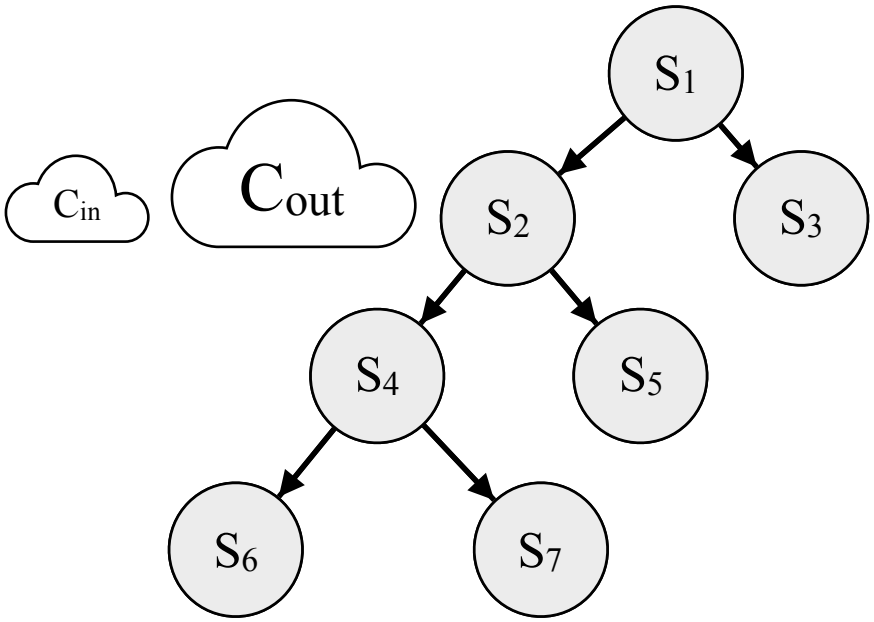
An enumeration tree T.

The size of a candidate set

We consider an enumeration tree T.

In each node in T, the size of C_{in} is monotonically decreasing if C_{in} is not empty set.

If C_{in} is empty set, then the size of C_{in} becomes at most Δ.



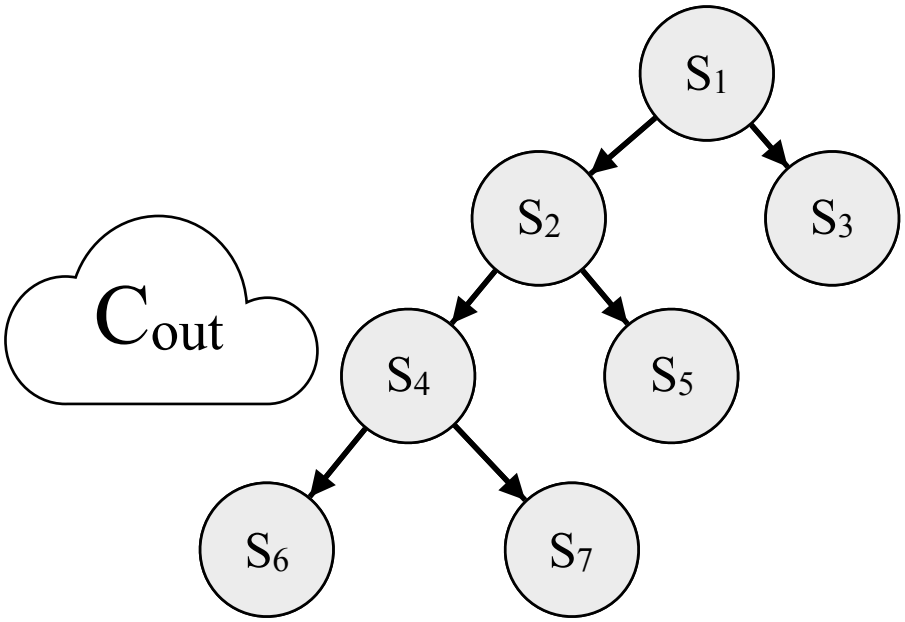
An enumeration tree T.

The size of a candidate set

We consider an enumeration tree T.

In each node in T, the size of C_{in} is monotonically decreasing if C_{in} is not empty set.

If C_{in} is empty set, then the size of C_{in} becomes at most Δ.



An enumeration tree T.

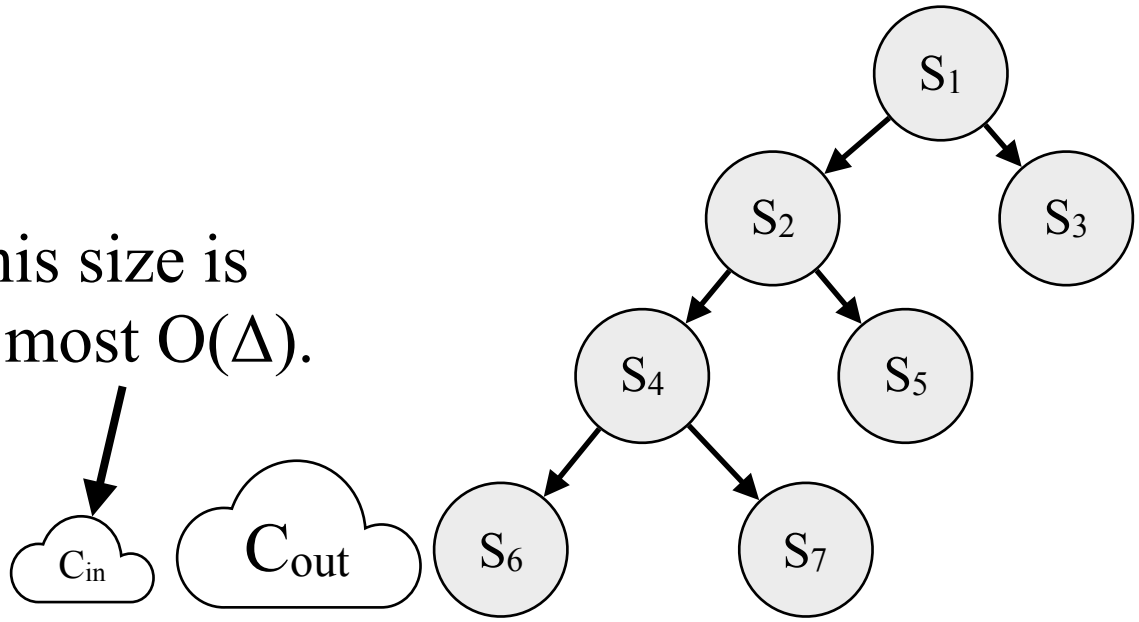
The size of a candidate set

We consider an enumeration tree T.

In each node in T, the size of C_{in} is monotonically decreasing if C_{in} is not empty set.

If C_{in} is empty set, then the size of C_{in} becomes at most Δ.

This size is at most $O(\Delta)$.



An enumeration tree T.

Conclusion

We present algorithms for enumerating induced subgraphs and subgraphs with bounded girth.

Our algorithms can apply non-connected version and weighted version in the same time complexity and space usage.

Main results	Time complexity (S is the set of solution.)	Space usage (S is the set of solution.)
Enumeration of induced subgraphs with bounded girth	$T = O(\Sigma_{S \in \mathcal{S}} \lvert E(G(N[S])) \rvert)$ $= O(\Sigma_{S \in \mathcal{S}} \Delta \lvert S \rvert)$ $= O(m \lvert \mathcal{S} \rvert)$ total time	$O(\max_{S \in \mathcal{S}} \{\lvert N[S]^3 \} \rvert)$ $= O(n^3)$ space
Enumeration of subgraphs with bounded girth	$T = O(\Sigma_{S \in \mathcal{S}} \lvert V(G[S]) \rvert)$ $= O(\Sigma_{S \in \mathcal{S}} \lvert S \rvert)$ $= O(n \lvert \mathcal{S} \rvert)$ total time	$O(\max_{S \in \mathcal{S}} \{\lvert V(G[S])^3 \} \rvert)$ $= O(n^3)$ space