Succinct Data Structures for Small Clique-Width Graphs

Seungbum Jo (Chungbuk National University, South Korea)

Joint work with

Sankardeep Chakraborty (NII, Japan)

Kunihiko Sadakane (The University of Tokyo, Japan)

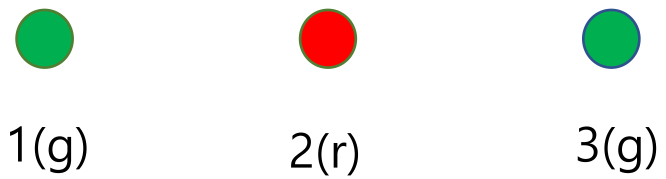
Srinivasa Rao Satti (NTNU, Norway)

DCC 2021

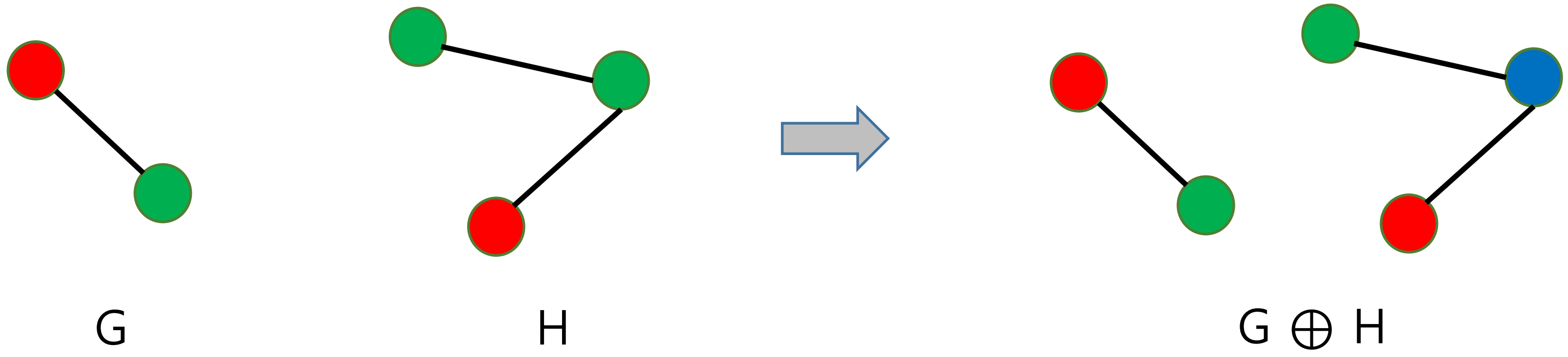
Clique-Width

Consider the following four operations on (undirected) graphs.

1. **Create** a vertex v with color i (denoted as v(i))



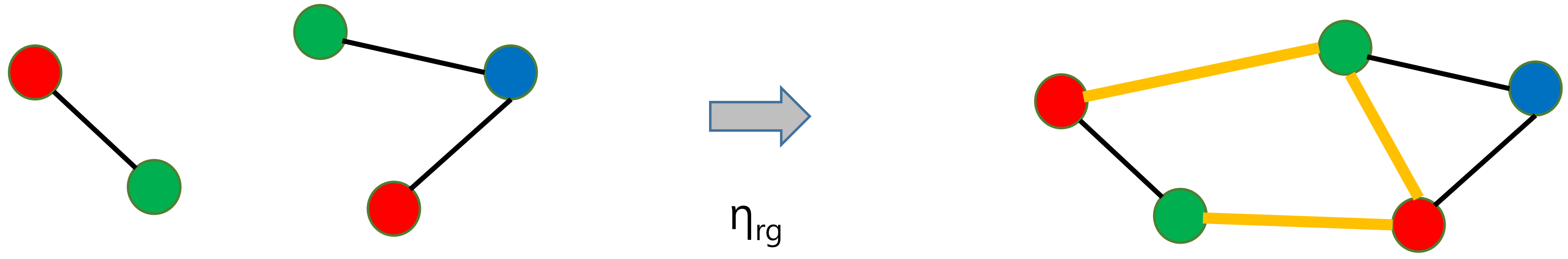
2. **Disjoint union** of labeled (colored) graph G and H (denoted as G ⨁ H)



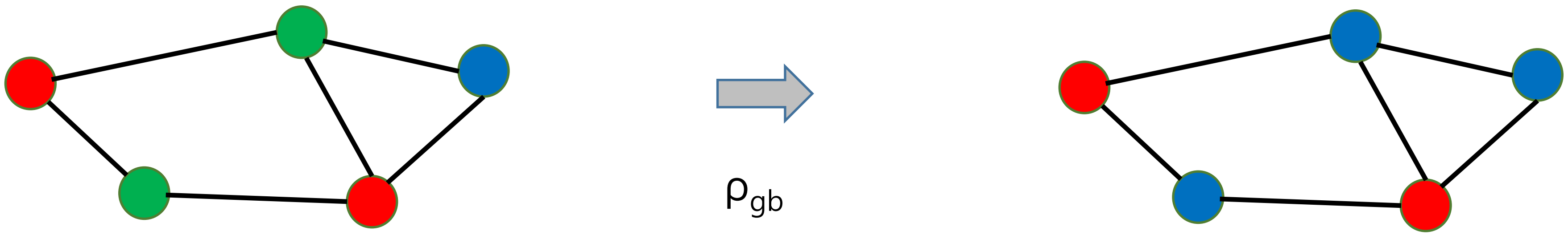
Clique-Width

Consider the following four operations on (undirected) graphs.

3. **Join** the color i and j (denoted as η_{ij})

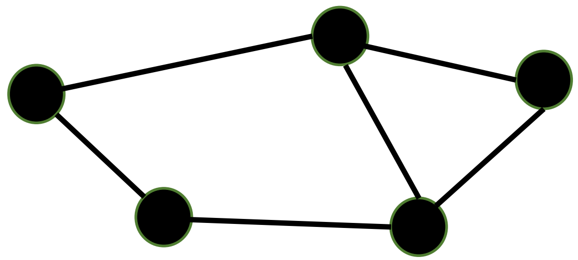


4. **Recolor** all the vertices of color i to j (denoted as ρ_{ij})



Clique-Width

- Clique-width of G (cw(G)) : **Number of minimum colors** to construct G using the previous four operations.



cw(G) = 3

Some examples

Clique-width 2 : Cliques, cographs….
Clique-width 3 : Distance-hereditary graph, 3-leaf power….

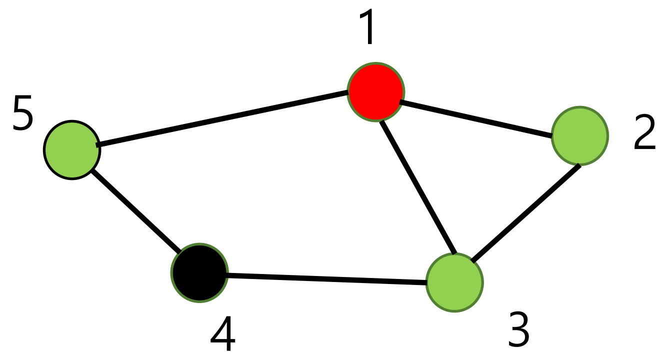
- Computing the clique-width of arbitrary graph is NP-hard.
- For small clique-width graphs, many NP-hard problems on general graphs (3-colorability, Hamiltonian cycles …) can be solved in polynomial time.

Problem statement

Problem : Given an undirected, unlabeled **graph G with n vertices whose clique-width is k**, is there any space-efficient data structure to support the following queries in efficient time ?

For any vertices u, v ∈ G

1. degree (v) : returns the degree of v.
2. neighborhood(v) : returns all the vertices adjacent to v.
3. adjacent (u, v) : returns true iff u and v are adjacent.



degree (1) = 3

neighborhood (1) = {2, 3, 5}

adjacent (1, 4) = false

Previous results & Our results

Problem : Given an undirected, unlabeled **graph G with n vertices whose clique-width is k**, is there any space-efficient data structure to support degree, neighborhood, and adjacent queries in efficient time ?

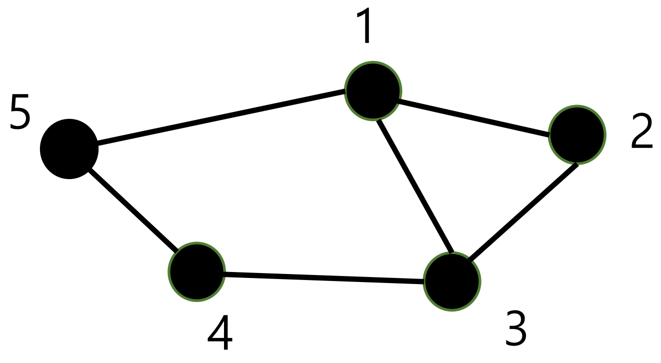
	Space (in bits)	degree	neighborhood (per neighbor)	adjacent
Kamali (2018)	O(kn) (O(knlog*n) bits for degree queries)	O(klog *n)	O(1)	O(1)
Our results ($k \le \epsilon\sqrt{\log \mathrm{n} / \log\log n}$)	f(n,k) + o(f(n,k))	O(k)	O(log n / k)	O(k)

- f(n, k): **Information-theoretical lower bound** of space to represent G.
- Kamali (2018) showed that (k-8)n ≤ f(n, k) ≤ 9kn, for k ≥ 9.
- Our data structure is **succinct** when $k \le \epsilon\sqrt{\log \mathrm{n} / \log\log n}$.
- For constant k, our data structure supports degree and adjacent query in O(1) time, and neighborhood query in O(log n) time per neighbor.

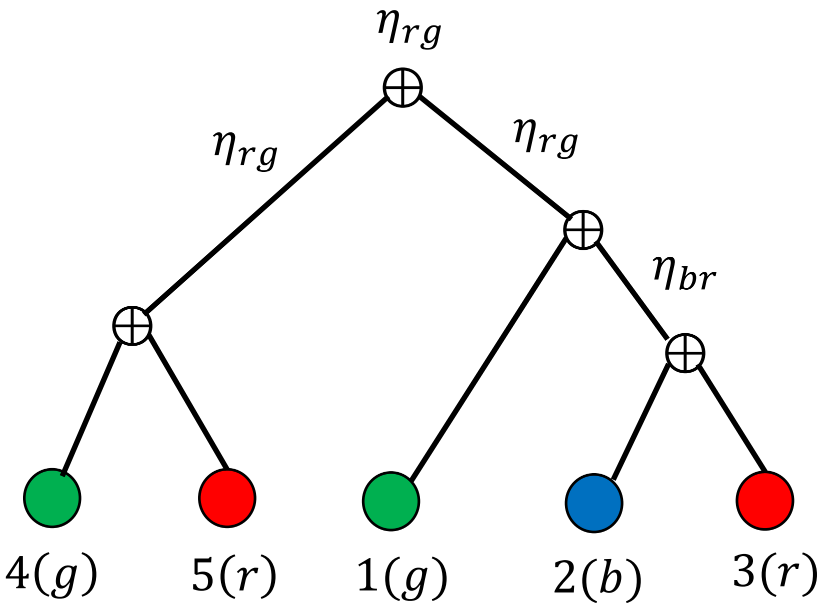
Succinct Encoding

- If one knows the clique-width of G (= k) there exists a **k-expression** that constructs G.
- Any k-expression can be represented by a labeled tree structure, named **union tree T**.
- One can reconstruct G from the union tree T of G

→ **Encoding of T gives an encoding of G !**



G (clique-width :3)



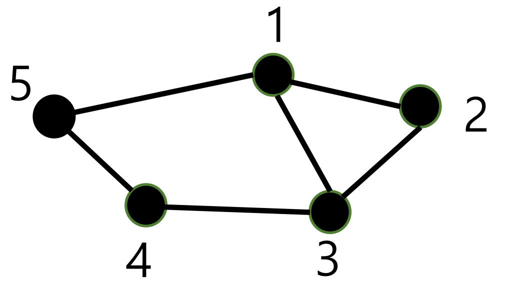
Union tree T of G

$\eta_{rg}(\eta_{rg}(4(g) \oplus 5(r)) \oplus \eta_{rg}, \eta_{br}((1(g) \oplus 2(b)) \oplus 3(r)))$ 3-expression of G

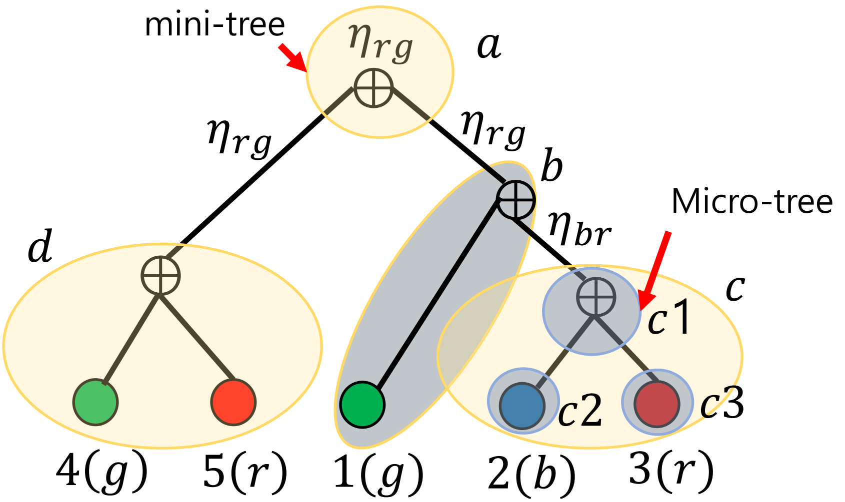
Succinct Encoding

Outline of the encoding :

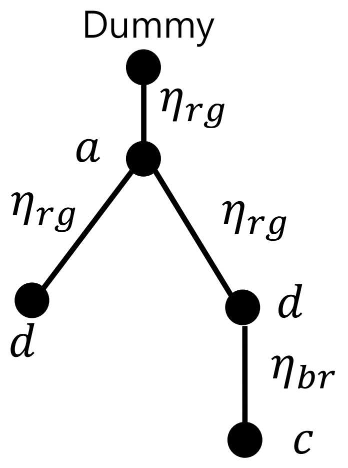
How to encode T ? : **Tree covering** [FM14]



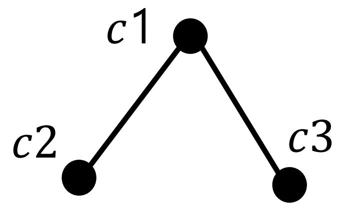
G



T



Tree over mini-tree



Mini-tree over micro-tree

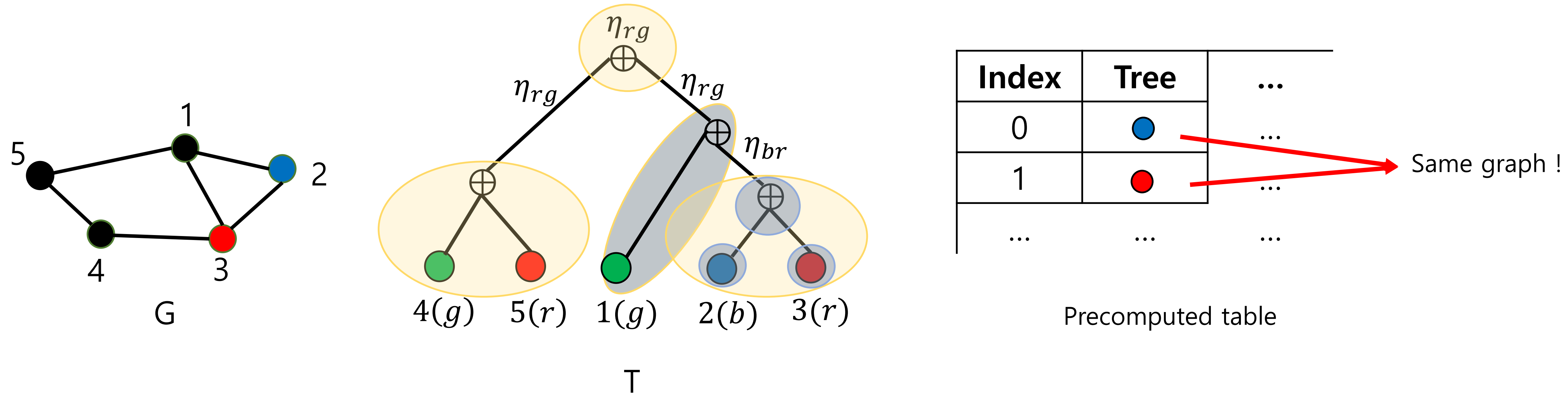
Index	**Tree**	**…**
0		…
1		…
…	…	…

Precomputed table

- Two-level decomposition algorithm (T → mini-tree → micro-tree)
- Tree over mini-trees, and mini-tree over micro-trees are stored using the pointer-based representation.
- Each micro-tree is stored as the **index of the precomputed table**, which stores all possible types of the micro-trees (with additional information for queries).
- Can support wide range of navigation queries on trees in O(1) time.

Succinct Encoding

Outline of the encoding



- Tree on micro-trees and mini-tree over micro trees can be stored in o(kn) bits of space.

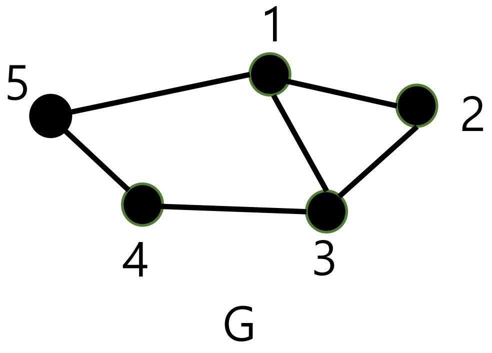
Problem : #non-isomorphic (colored, labeled) micro-trees >> #non-isomorphic clique-width k graphs with same size

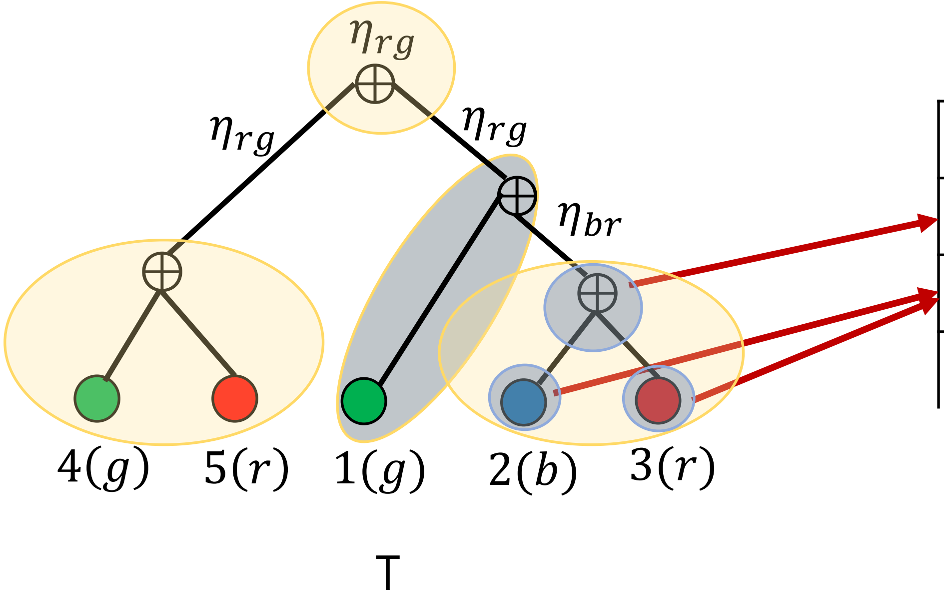
→ Size of the **pre-computed table is too large to encode every micro trees of T** using the index of the table in succinct space.

How to solve this problem ?

Succinct Encoding

Outline of the encoding





Index	Graph	...
0	Empty	...
1	●	...
...	...	...

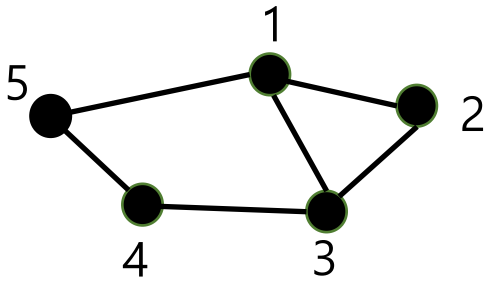
Precomputed table

Solution : Maintain a precomputed table to store every clique-width k graph (proportional to the size of the micro-tree of T), instead of the micro-tree.

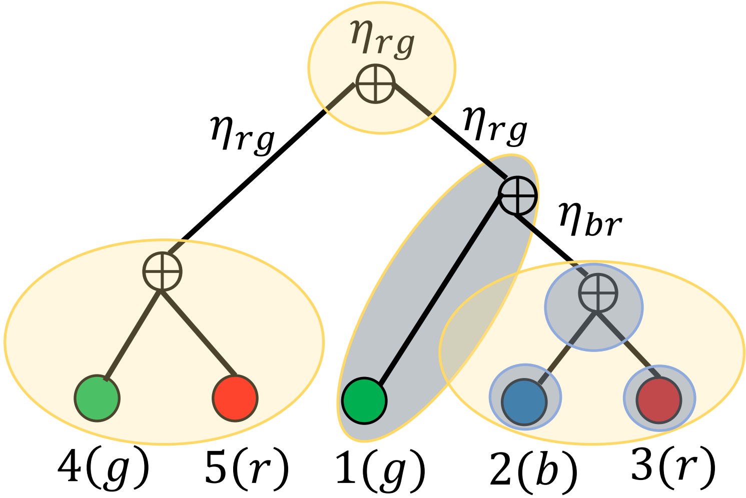
Problem : Loose the information about the colors of vertices.

Succinct Encoding

Outline of the encoding



G



T

Index	Graph	...
0	Empty	...
1	●	...
...	...	...

Precomputed table

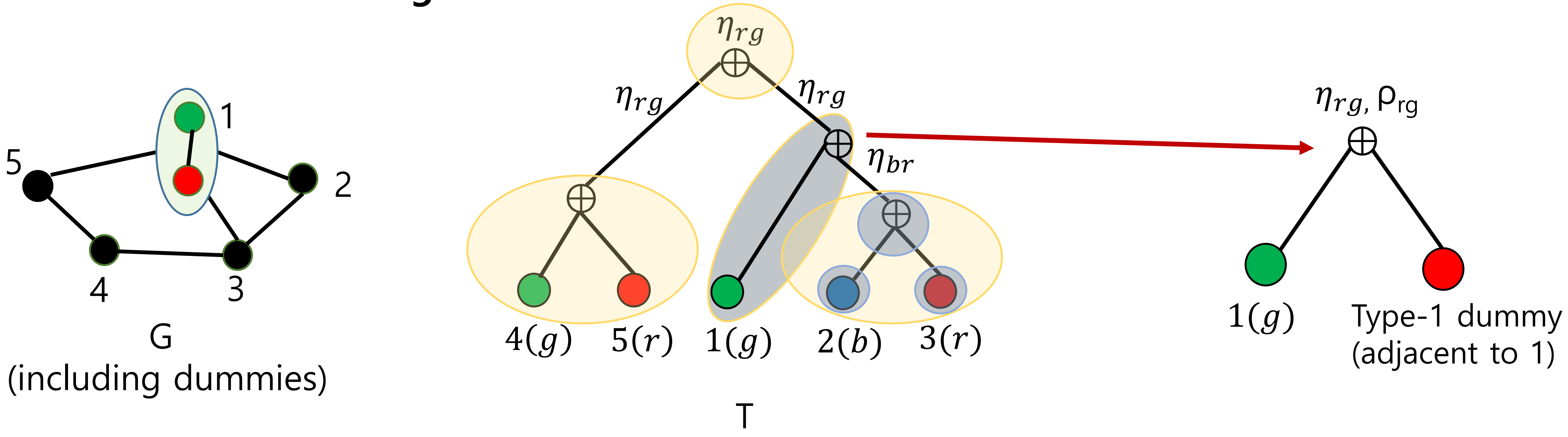
Problem : Loose the information about the colors of vertices.

Key observation : We only need color of vertices **at the root of the micro-tree.**

→ Add some dummy nodes to decode them.

Succinct Encoding

Outline of the encoding



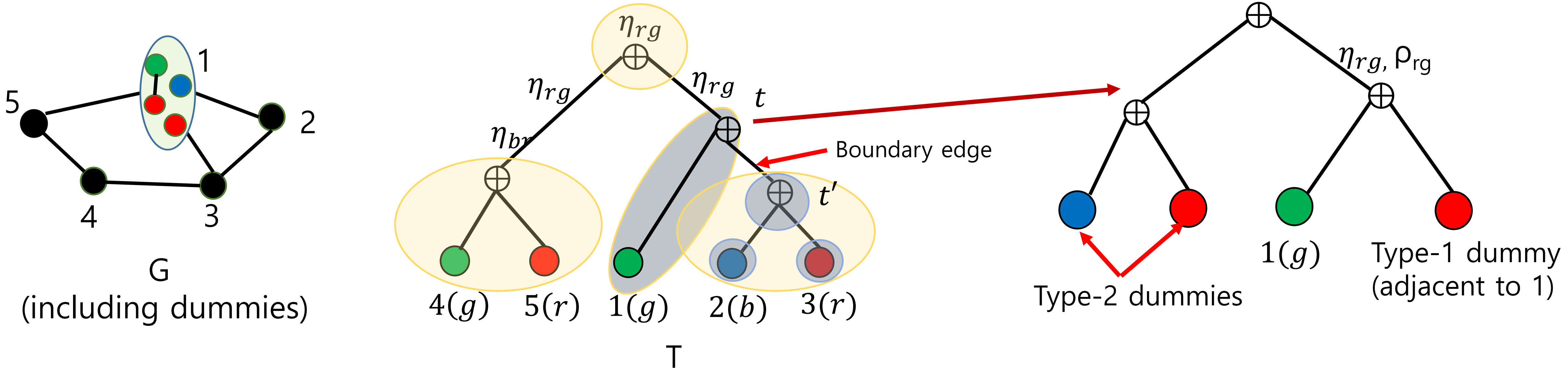
We define two types of dummy nodes on each micro-tree

Type-1 dummy nodes : To decode the color of each vertices at the root of the micro-tree.

→ Decode the color by checking the adjacency with the dummy nodes (using precomputed table).

Succinct Encoding

Outline of the encoding

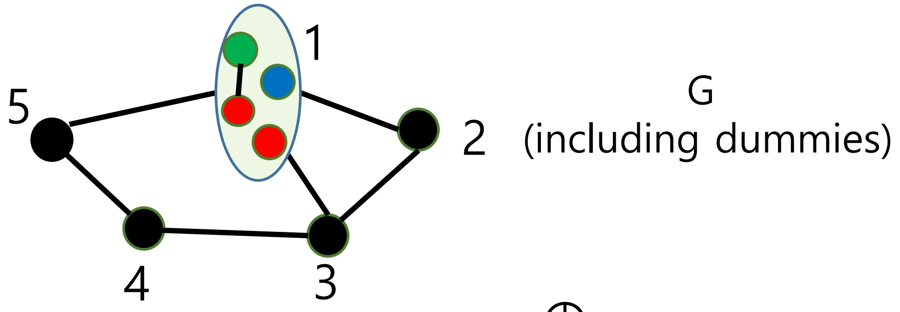


We define two types of dummy nodes on each micro-tree

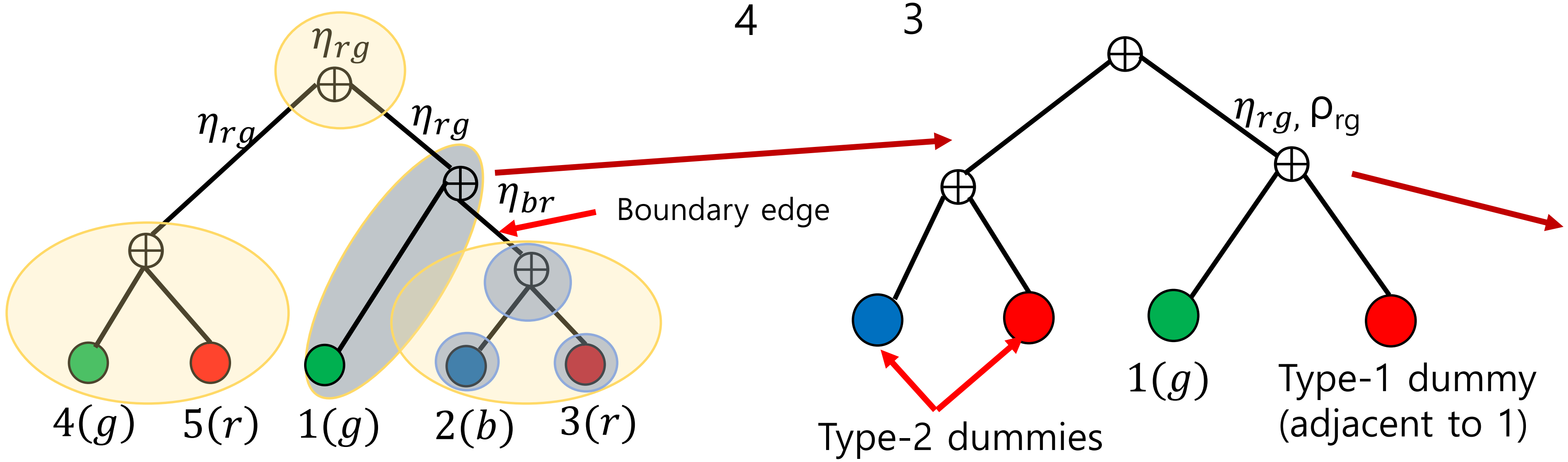
- Each micro tree t is connected by at most 1 another micro-tree t′ by a boundary edge.

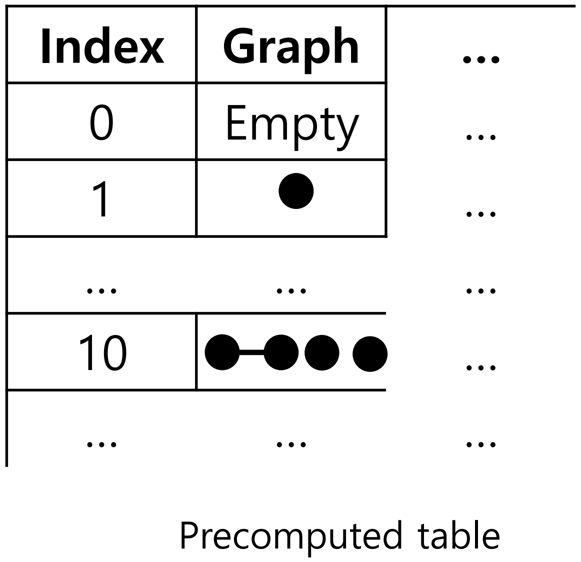
Type-2 dummy nodes : To decode the connection between the vertices in t and all the vertices in the subtree at the root of t′.

Succinct Encoding



Outline of the encoding





- We encode the corresponding graph of the micro-tree (with Type-1 and 2 dummy nodes) as an index of the precomputed table.

- The additional information of dummy nodes (position, colors….) is stored explicitly.

- Since the k is small ($k \leq \epsilon\sqrt{\log \mathrm{n}/\log\log n}$), and there exists at most 2k dummy nodes for each micro-tree of T, all the additional information can be stored in succinct space.

Query Algorithms

- Maintain the similar auxiliary structures of Kamali (2018), with some modifications for keeping the information on the nodes in tree over micro-trees.

- There exists some time blow-up for neighborhood queries, since we need to search every vertex in the micro-tree which has at least one neighborhood of the query vertex.

Conclusion

- Succinct data structure for the graphs with small bounded clique-width.

- Compare to the Kamali (2018)'s result, our data structure can support degree queries in O(k) time, still using succinct space.

- Since the cograph is equivalent to the graph with clique-width 2, our data structure gives a succinct data structure for cographs.

Further improvements (not in the paper)

- Succinct ds for cograph with O(1) time neighborhood query (per neighbor).
- Succinct ds for distance-hereditary graphs and Ptolemaic graphs (subclasses of the graph whose clique-width is 3).

Open question

: Currently succinct data structure for graphs with bounded width parameter is only considered for tree-width (FK14) and clique-width. Can we design succinct data structures w.r.t. other width parameters?