GuP: Fast Subgraph Matching by Guard-based Pruning

Junya Arai, Yasuhiro Jujiwara, Makoto Onizuka (June, 2023)

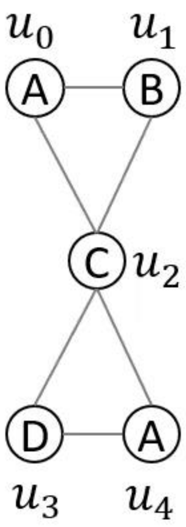
My Intuitive Understanding of Guards

- R(u,v) stores data vertices of which at least one must be "reserved" in order for a full-embedding to be possible. If all vertices of R(u,v) are assigned before (u,v), then we need not continue extending M.
- NV(u,v) stores assignments (u',v') that guarantee a deadend, so GuP can avoid fruitless searches earlier on. We want as small a nogood guard as possible, so more fruitless partial embeddings can be detected when they are still small.

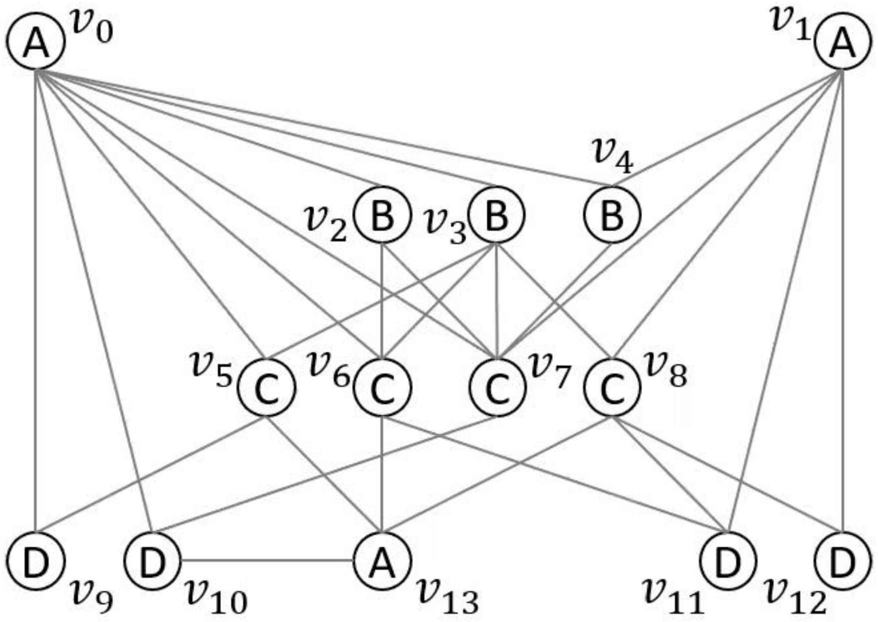
Candidate filtering

Extract data vertices that can be destinations of a specific query vertex

- Compare local structures (e.g., label, degree, adjacent labels)



Query graph Q

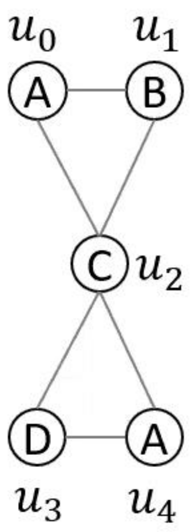


Data graph G

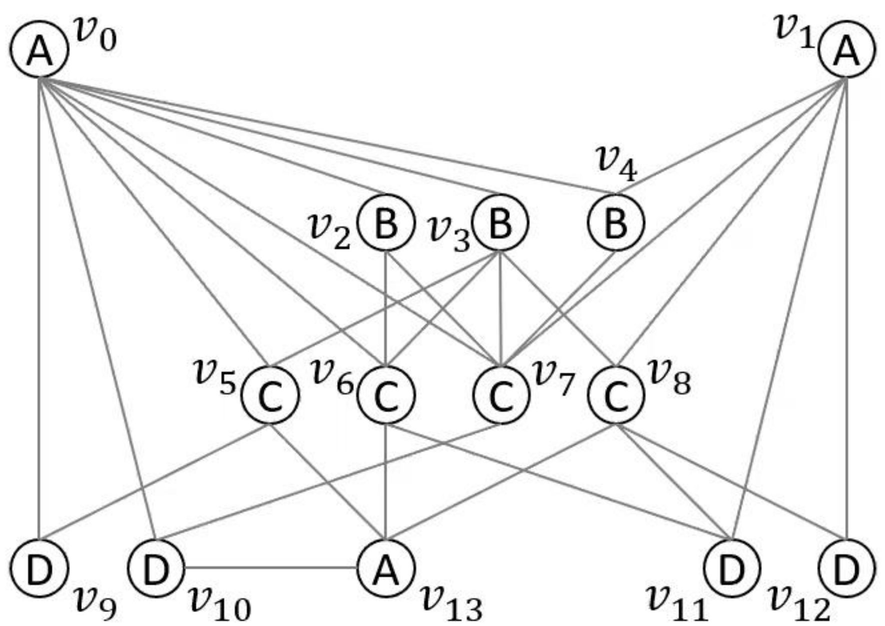
- $C(u_0) = \{v_0, v_1\}$
- $C(u_1) = \{v_2, v_3, v_4\}$
- $C(u_2) = \{v_5, v_6, v_7, v_8\}$
- $C(u_3) = \{v_9, v_{10}, v_{11}, v_{12}\}$
- $C(u_4) = \{v_0, v_1, v_{13}\}$

Candidate space [Han+ SIGMOD'19]

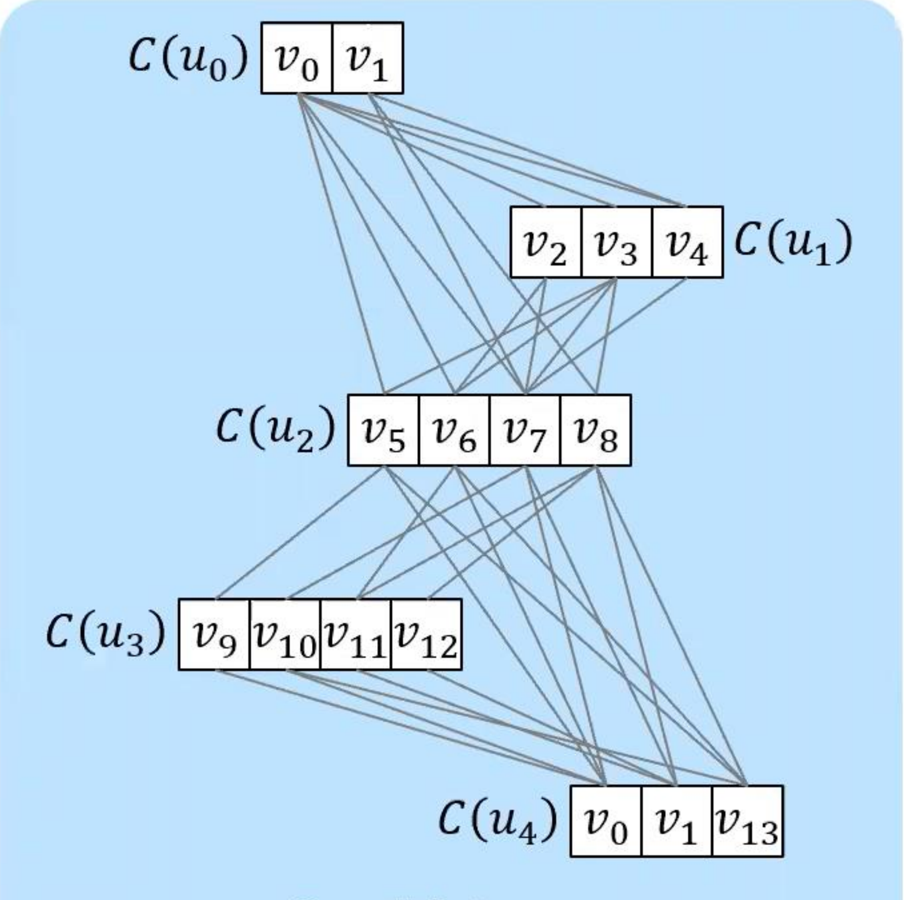
Candidate vertices + edges between them



Query graph Q

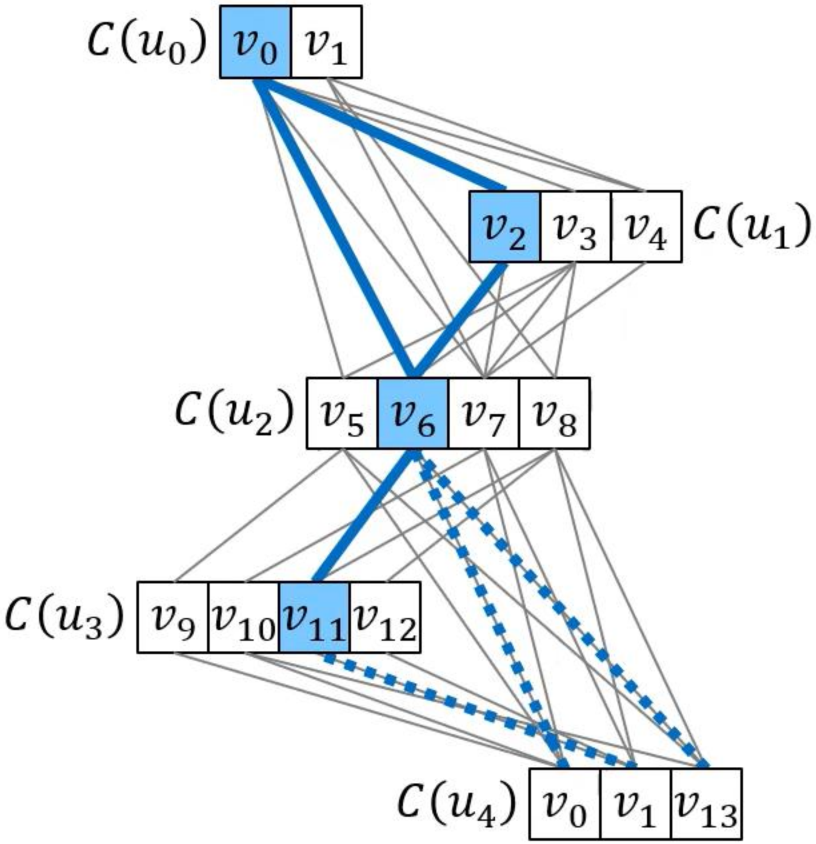


Data graph G



Candidate space

Backtracking search



Recursively extend **partial embedding M** until it becomes a **full embedding**

$M = \emptyset$

$M = \{(u_0, v_0)\}$

$M = \{(u_0, v_0), (u_1, v_2)\}$

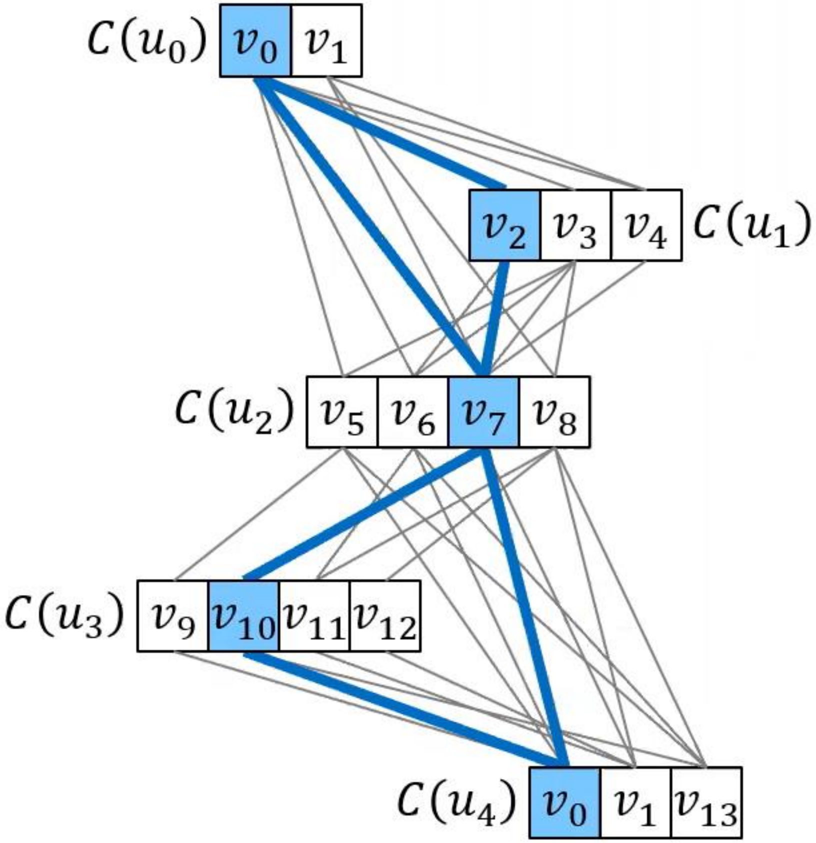
$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_6)\}$

$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_6), (u_3, v_{11})\}$

$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_6), (u_3, v_{11}), (u_4, \_)\}$ ✘

- v_6 and v_{11} lack a common neighbor for u_4
- "**Deadend**"

Backtracking search



$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_6), (u_3, v_{11}), (u_4, \_)\}$ ✗

$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_6)\}$

$M = \{(u_0, v_0), (u_1, v_2)\}$

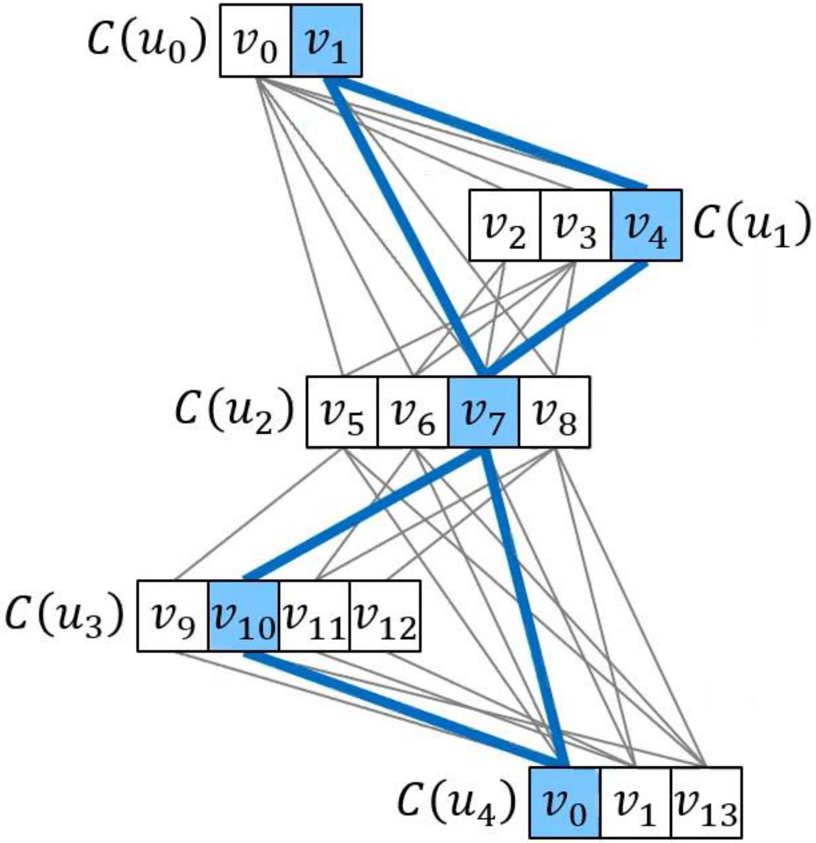
$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_7)\}$

$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_7), (u_3, v_{10})\}$

$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_7), (u_3, v_{10}), (u_4, v_0)\}$ ✗

- Non-injective mapping (using v_0 twice)

Backtracking search



$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_7), (u_3, v_{10}), (u_4, v_0)\}$ ✘

...

$M = \{(u_0, v_0)\}$

$M = \emptyset$

$M = \{(u_0, v_1)\}$

...

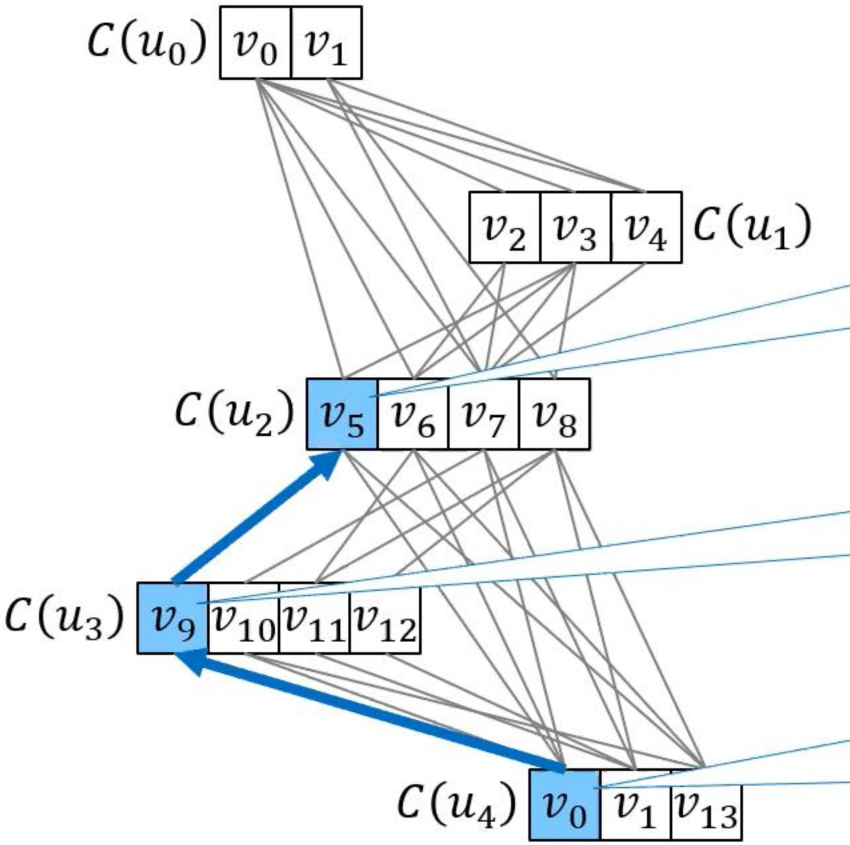
$M = \{(u_0, v_1), (u_1, v_4), (u_2, v_7), (u_3, v_{10}), (u_4, v_0)\}$ ✔

No full embedding found

$\Rightarrow M = \{(u_0, v_0)\}$ is a deadend

All the recursions from $M = \{(u_0, v_0)\}$ were fruitless

Reservation guard: basic idea



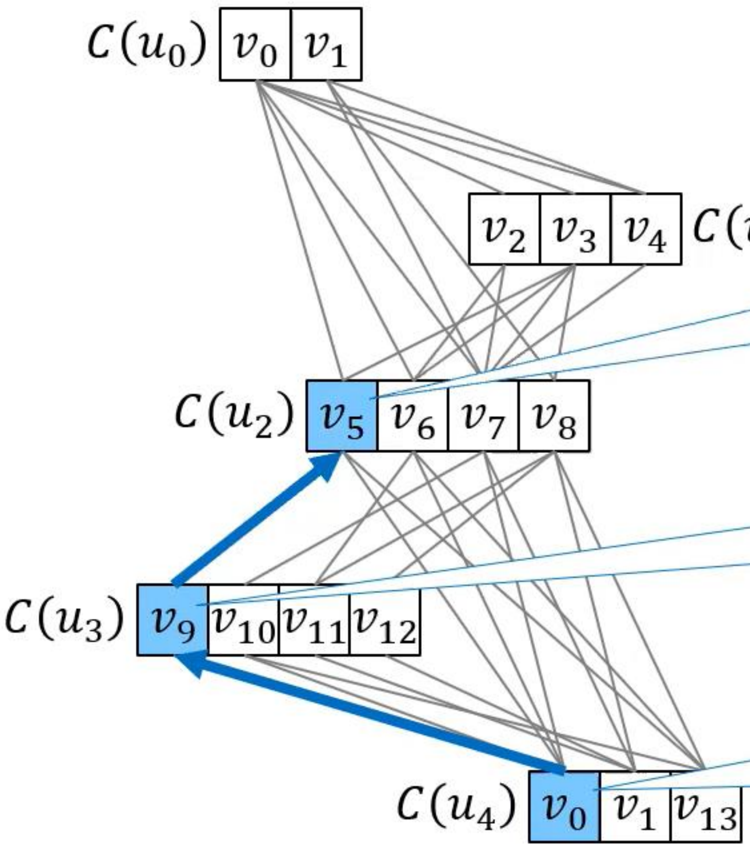
Propagate the injectivity constraint to earlier backtracking steps

Inconsistent if v_0 is used in M because v_9 is the only adjacent candidate for u_3

Inconsistent if v_0 is used in M because v_0 is the only adjacent candidate for u_4

Inconsistent if v_0 is used in M because of the injectivity constraint

Reservation guard: formalization



Reservation guard $R(u_i, v) \subseteq V_G$

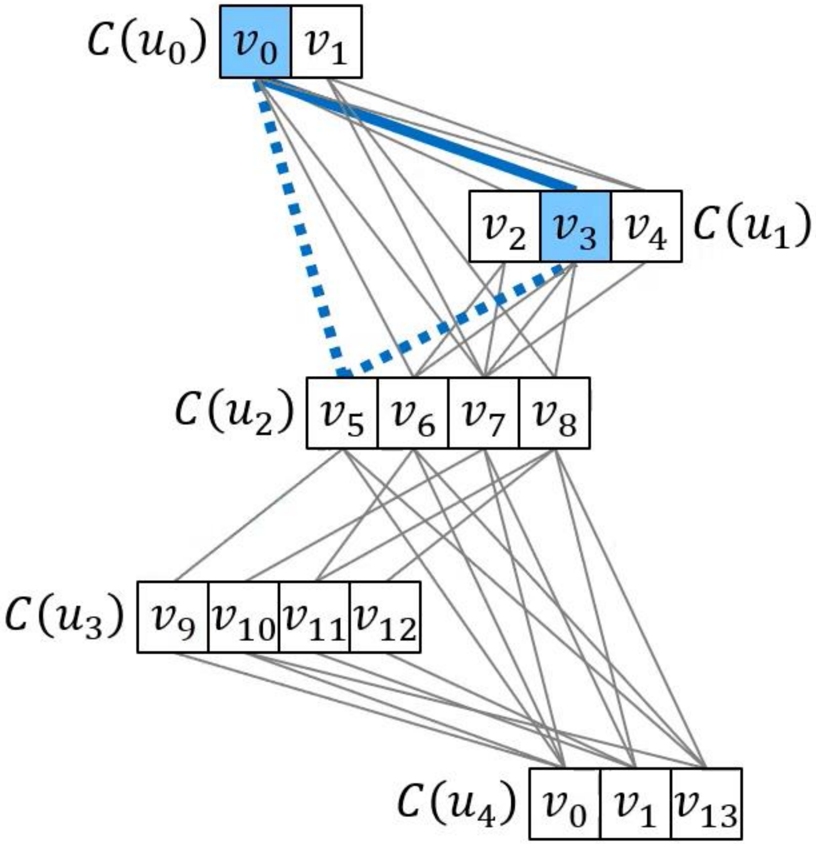
Filter out v if all the vertices in $R(u_i, v)$ is used in M

$$R(u_2, v_5) = R(u_3, v_9) = \{v_0\}$$

$$R(u_3, v_9) = R(u_4, v_0) = \{v_0\}$$

$$R(u_4, v_0) = \{v_0\}$$

Reservation guard: running example



Filtered out because $R(u_2, v_5) = \{v_0\}$ and v_0 is already used for u_0

...

$M = \{(u_0, v_0), (u_1, v_3)\}$

$M = \{(u_0, v$ $(u_1, v_3), (u_2, v_5)\}$

$M = \{(u_0, v$ $(u_1, v_3), (u_2, v_5), (u_3, v_9)\}$

$M = \{(u_0, v$ $v_5), (u_3, v_9), (u_4, v_0)\}$ ✘

Pruned

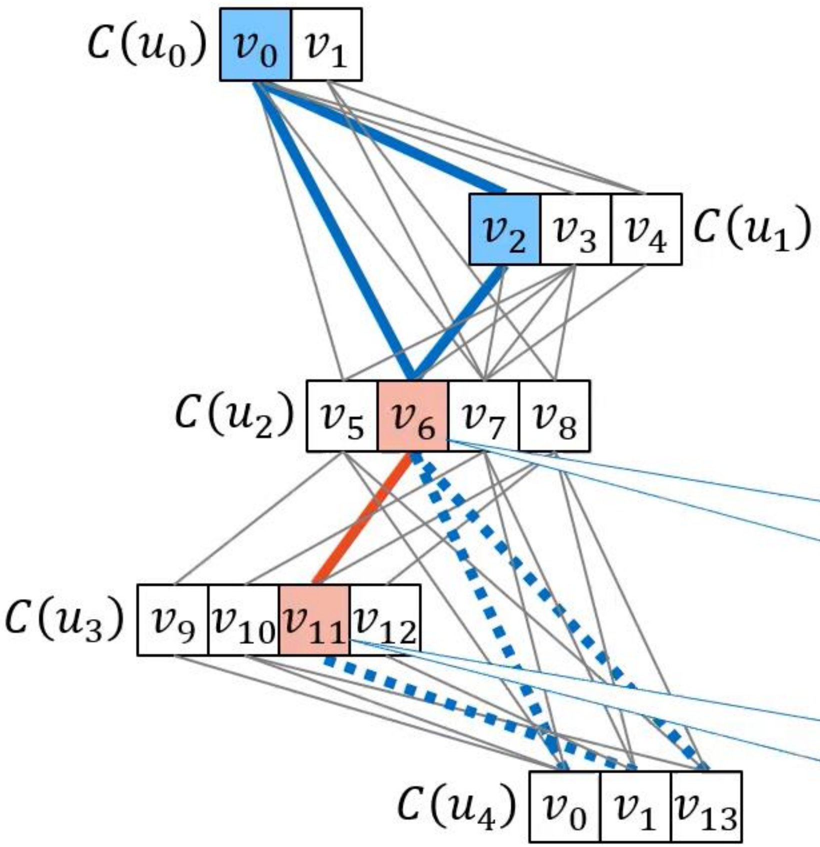
$M = \{(u_0, v$ $(u_1, v_3), (u_2, v_5), (u_3, v_9)\}$

$M = \{(u_0, v$ $(u_1, v_3), (u_2, v_5)\}$

$M = \{(u_0, v$ $(u_1, v_3)\}$

$M = \{(u_0, v_0), (u_1, v_3), (u_3, v_6)\}$

Nogood guard: basic idea



Learn inconsistent assignments from deadends encountered during backtracking

…

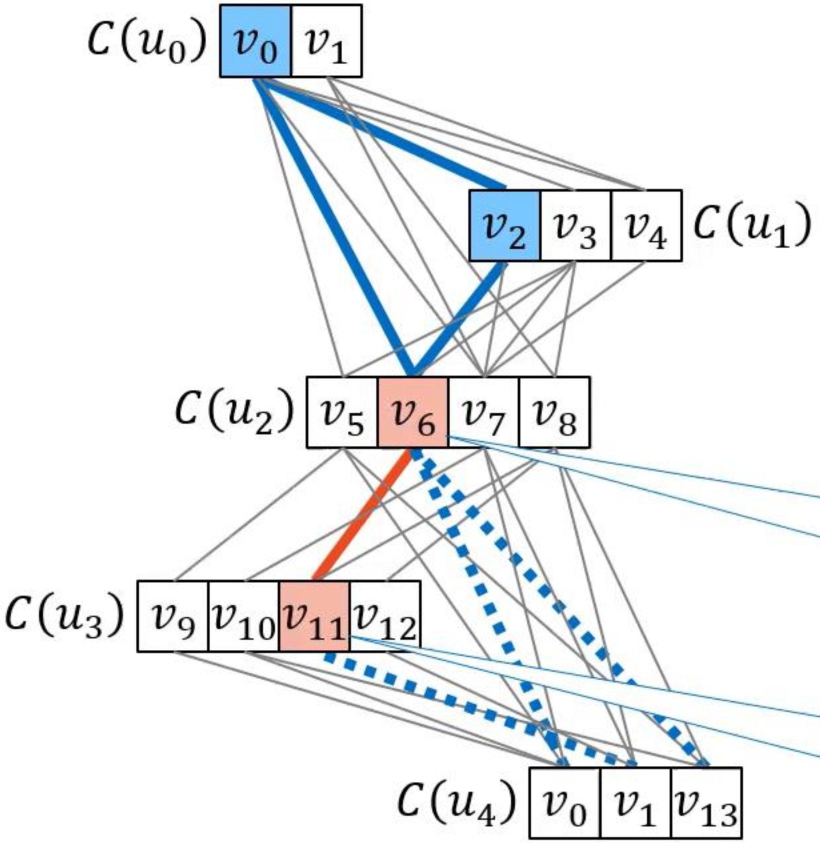
$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_6), (u_3, v_{11})\}$

$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_6), (u_3, v_{11}), (u_4, \_)\}$ ✘

Inconsistent with arbitrary M because its only neighbor v_{11} is inconsistent with (u_2, v_6)

Inconsistent with M containing (u_2, v_6) because v_6 and v_{11} lack common neighbor for u_4

Nogood guard: formalization



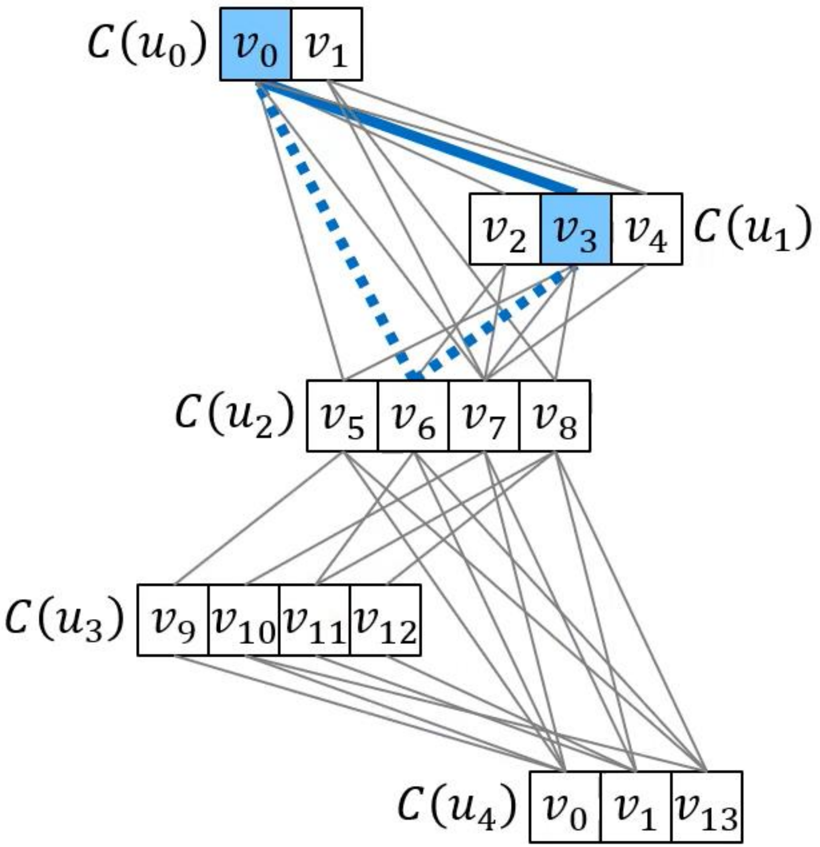
Nogood guard $NV(u_i, v) \subseteq V_Q \times V_G$

Filter out v for M s.t. $NV(u_i, v) \subseteq M$

$$NV(u_2, v_6) = \emptyset$$

$$NV(u_3, v_{11}) = \{(u_2, v_6)\}$$

Nogood guard: running example



$NV(u_3, v_{11}) \leftarrow \{(u_2, v_6)\}$

$NV(u_2, v_6) \leftarrow \emptyset$

…

$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_6), (u_3, v_{11})\}$

$M = \{(u_0, v_0), (u_1, v_2), (u_2, v_6), (u_3, v_{11}), (u_4, \_)\}$ ✘

…

$M = \{(u_0, v_0), (u_1, v_3)\}$

$M = \{(u_0, v_0), (u_1, v_3), (u_2, v_6)\}$

Filtered out because $NV(u_2, v_6) \subseteq M$

$M = \{(u_0, v_0), (u_1, v_3), (u_2, v_6), (u_3, v_{11})\}$

$M = \{(u_0, v_0), (u_1, v_3), (u_2, v_6), (u_3, v_{11}), (u_4, \_)\}$ ✘

Pruned

$M = \{(u_0, v_0), (u_1, v_3), (u_2, v_6), (u_3, v_{11})\}$

$M = \{(u_0, v_0), (u_1, v_3), (u_2, v_6)\}$

$M = \{(u_0, v_0), (u_1, v_3)\}$

$M = \{(u_0, v_0), (u_1, v_3), (u_2, v_7)\}$