UNIVERSITÀ DEGLI STUDI DI SALERNO

ARE WE READY TO USE GRAPHS IN PATTERN RECOGNITION?

Prof. Mario Vento
University of Salerno

Machine Intelligence lab for Video, Image and Audio processing

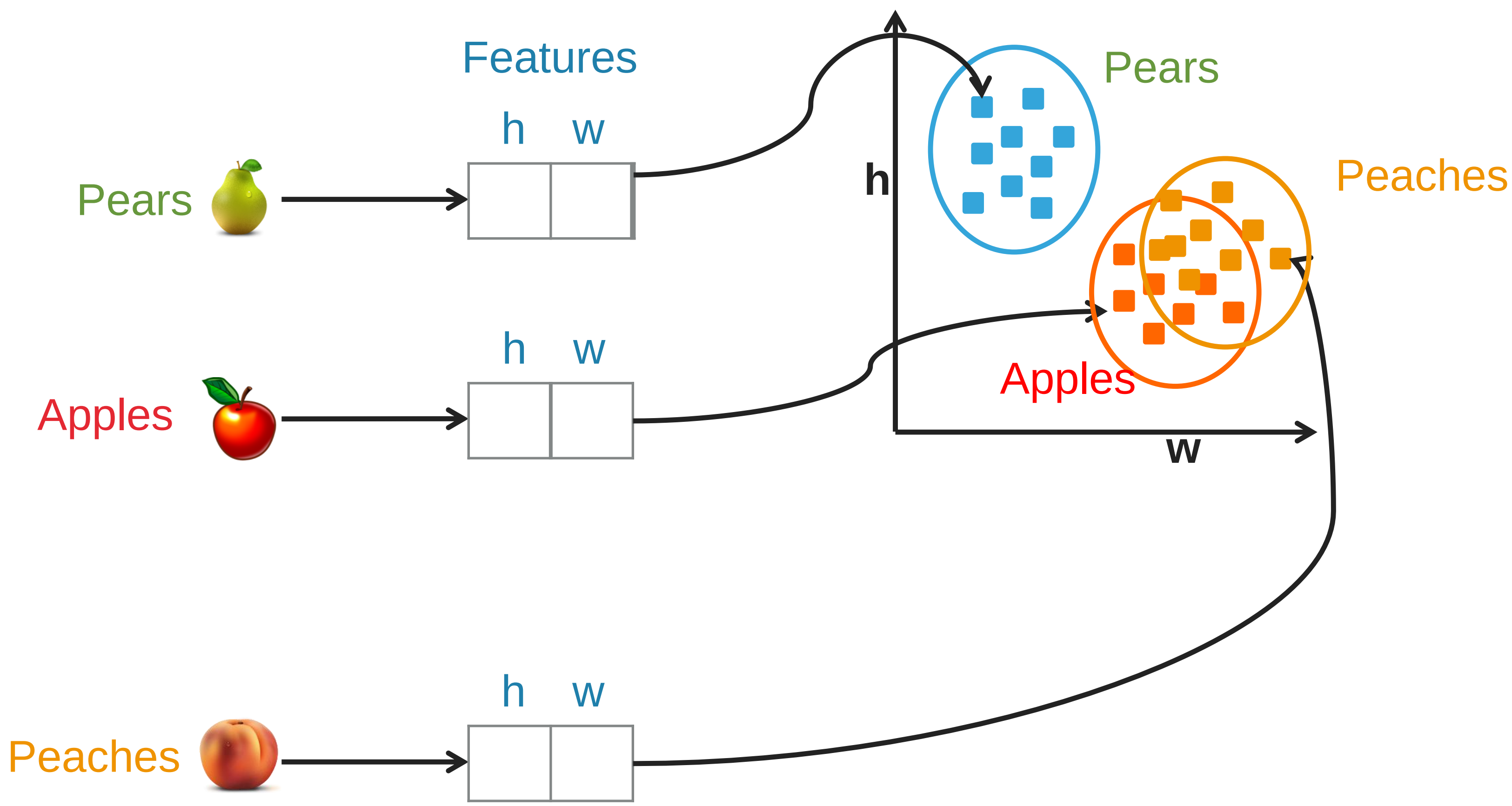
OUTLINE

- Overview of Graph based methods in PR
- Vector vs Graphs
- Pure methods… working in the graph space
  - Exact and inexact matching methods
  - Graph Edit Distance
- Impure Methods: From graph to Vectors
- Conclusions

Vectors vs Graphs

GRAPHS VS VECTORS

- SPR is based on a well founded mathematical framework: Vector Spaces
- Describing patterns by a vector of a set of measures has an immediate meaning
  - The pattern is a point in a Vector Space
  - (If the features are good) the distance of the points stands as a similarity measure between the corresponding patterns
  - Plenty of learning and classification methods



Similarity ≈ Euclidean Distance

ARE VECTORS REALLY EFFECTIVE?

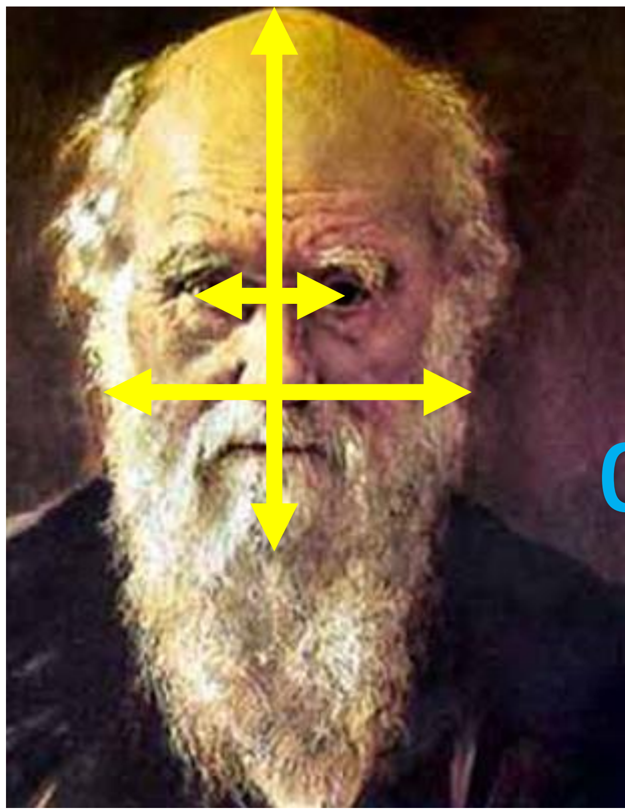
- The world is made of complex patterns
- Discriminant information are spread in different parts of the objects and concentrate somewhere
- Patterns generally contains subparts which are related each other
- Descriptions based on a set of features are not always effective: it is difficult to separate the parts containing the discriminant power
- … drawbacks pop up when the patterns have structural relationships
- … and statistical distribution of a feature set is not so effective to catch the differences

ARE VECTORS REALLY EFFECTIVE?

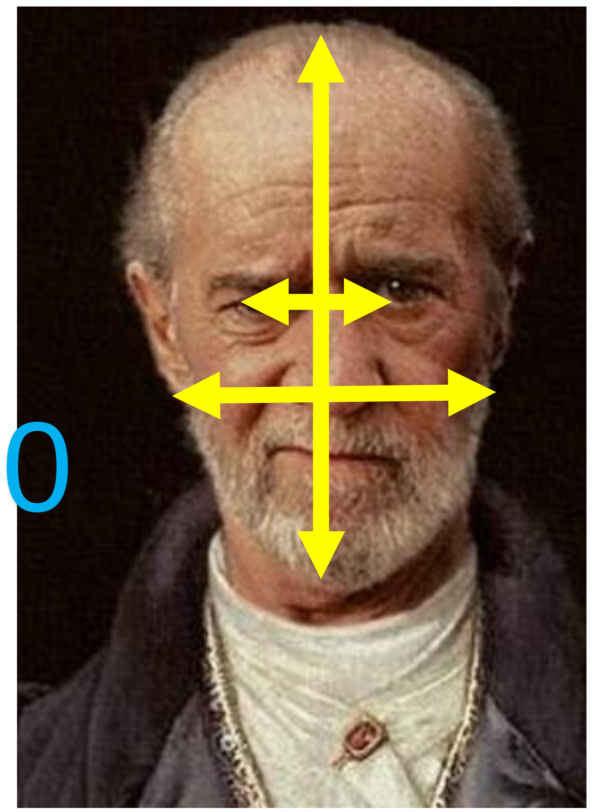
- The world is made of complex patterns
- Discriminant information are spread in different parts of the objects and concentrate somewhere
- Patterns generally contains subparts which are related each other
- Descriptions based on a set of features are not always effective: it is difficult to separate the parts containing the discriminant power
- … drawbacks pop up when the patterns have structural relationships
- … and statistical distribution of a feature set is not so effective to catch the differences

Inadequacy to deal with the decomposition into parts

ARE THEY THE SAME PERSON? WHERE DIFFERENCES LIE?



d=2,10



F_W	F_H	M1
17.15	25.21	7.00

F_W	F_H	M1
15.05	25.25	7.03

FROM VECTORS TO GRAPHS

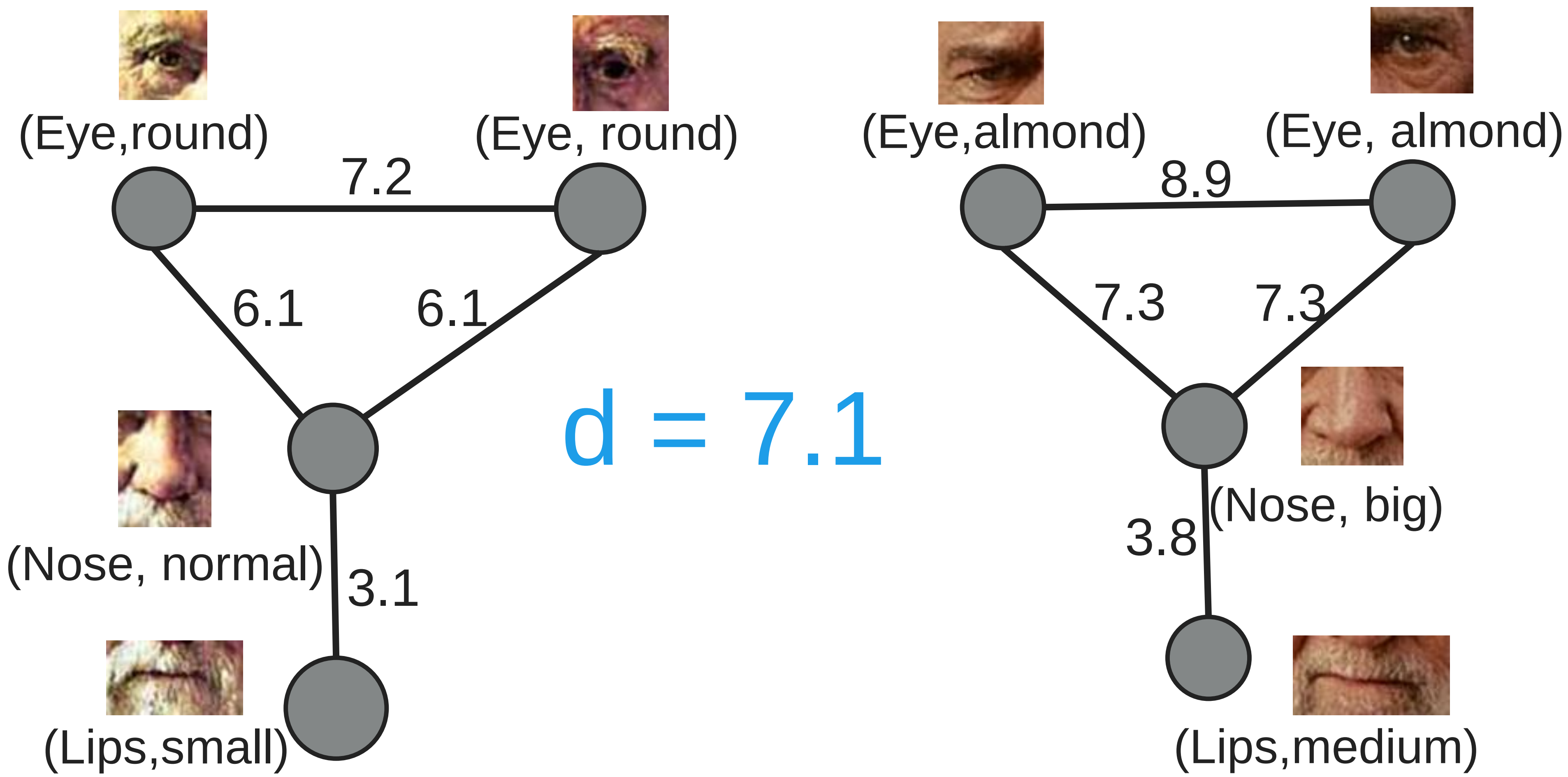
- Graph-based representations decompose the object into parts
  - Single parts are individually described
  - Relations between the parts are represented
  - Feature vectors are added to edges and nodes
  - Different information for different nodes

- Object comparison exploits contextual knowledge (inside a single object)

FROM VECTORS TO GRAPHS

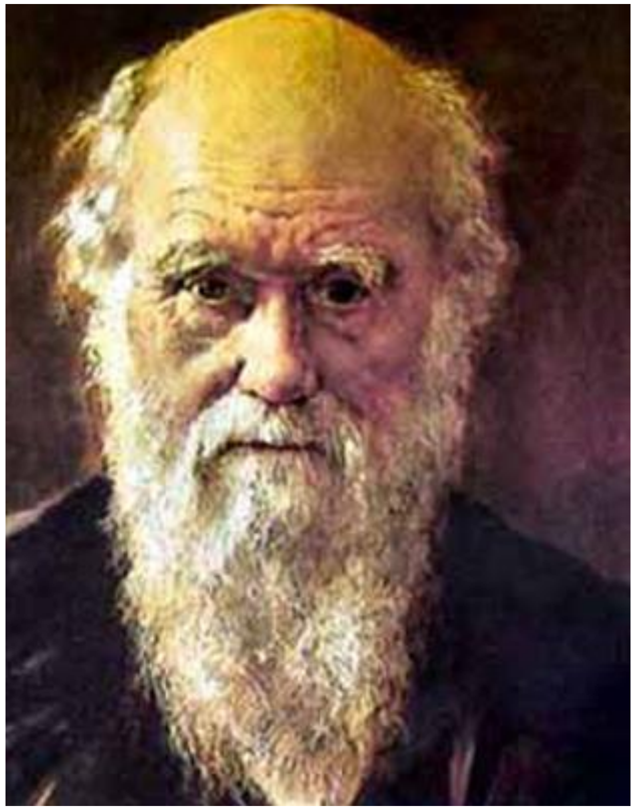
- Graph-based representations decompose the object into parts
  - Single parts are individually described
  - Relations between the parts are represented
  - Feature vectors are added to edges and nodes
  - Different information for different nodes

- Object comparison exploits contextual knowledge (inside a single object)

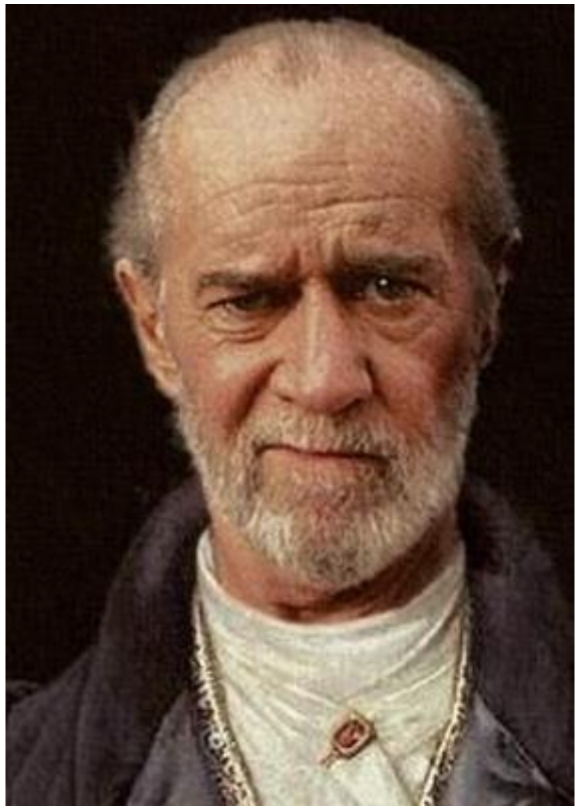


ARE THEY THE SAME PERSON?

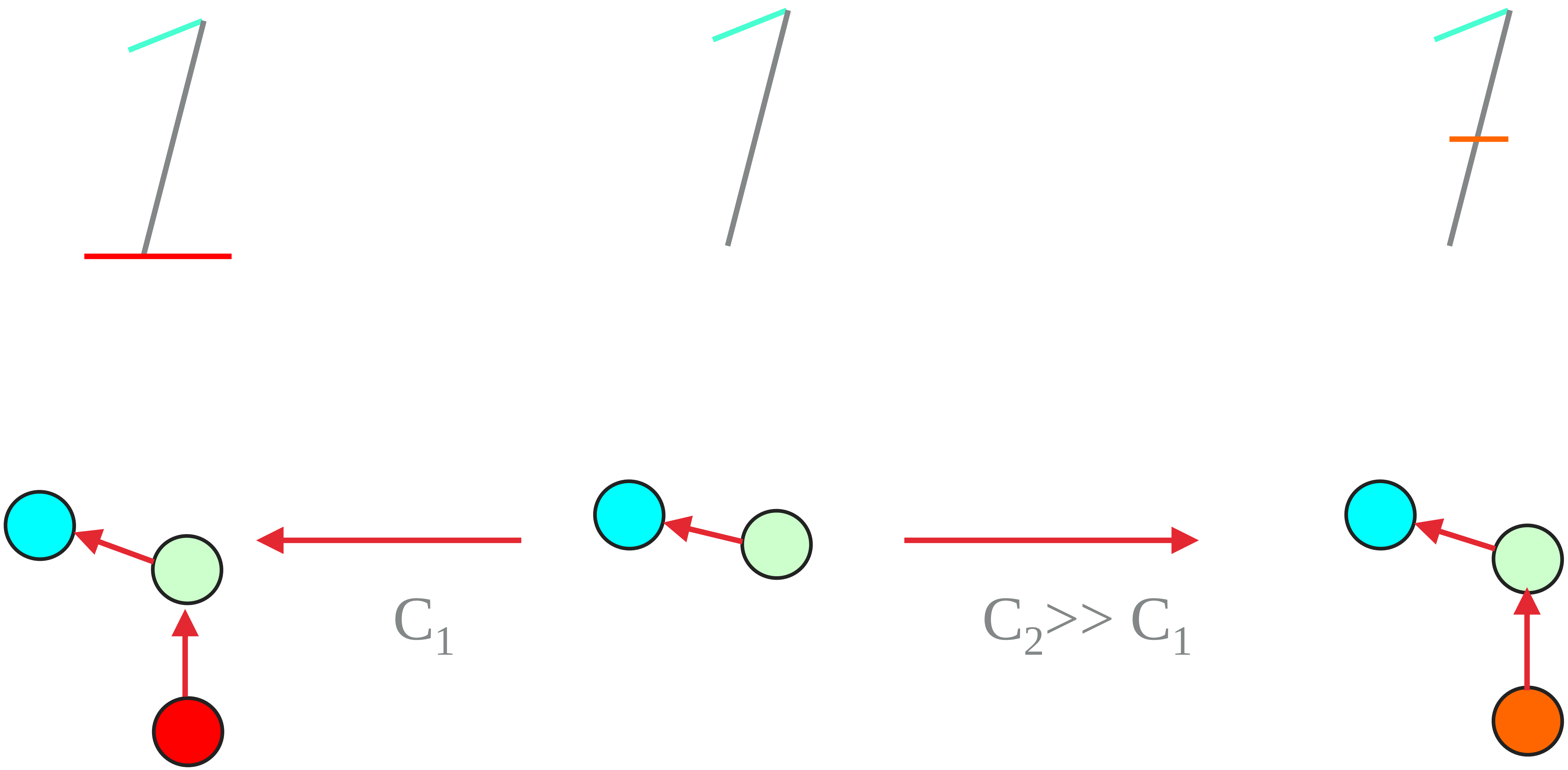
Charles Darwin



George Carlin



PRESERVING THE SEMANTIC VALUE OF THE CONTEXT



PATTERN RECOGNITION WITH GRAPHS

Statistical Pattern Recognition

- Objects as vectors of numerical features in a finite space
- Classification means evaluation of euclidean distance
- Learning means dividing the space in zones associated to the classes

Structural Pattern Recognition

- Objects are Graphs with attributes
- Classification is Graph comparison
- Learning is graph generalization

Basic Problems

Graph Comparison

- Equals or not?
- Similarity?

Graph Learning

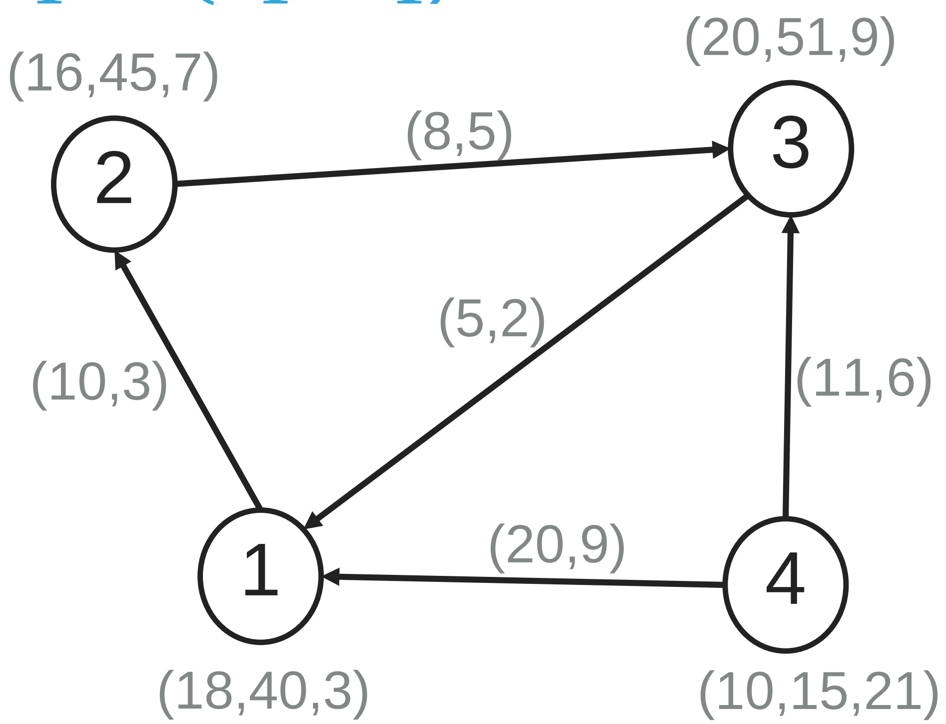
- Symbolic graph prototypes?
- Prototype graphs?
- Learned automatically ?

METHODS IN THE GRAPH SPACE

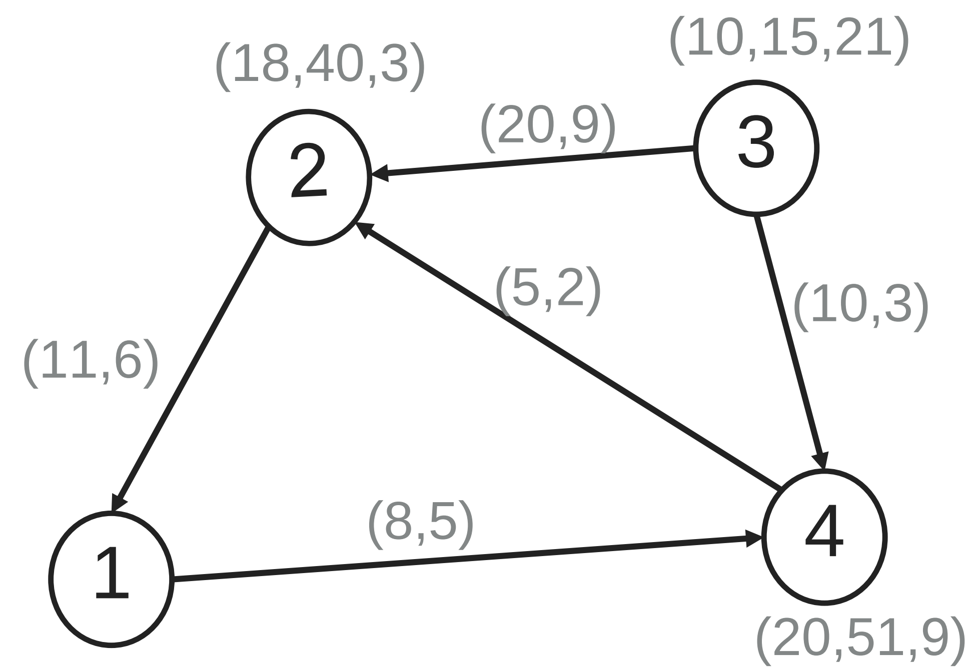
WORKING INTO GRAPH SPACE - PURE METHODS

- Graph Comparison: compatibility between the structure of the input patterns
  - Morphism between graph structures
  - Node and Edge compatibilities

$G_1 = (V_1, E_1)$



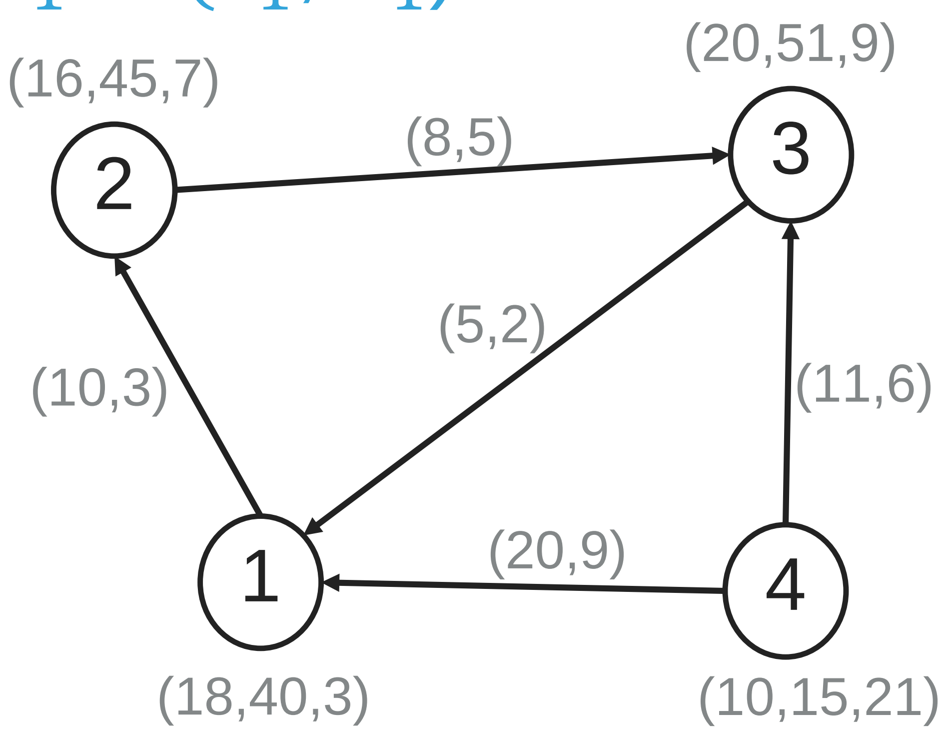
$G_2 = (V_2, E_2)$



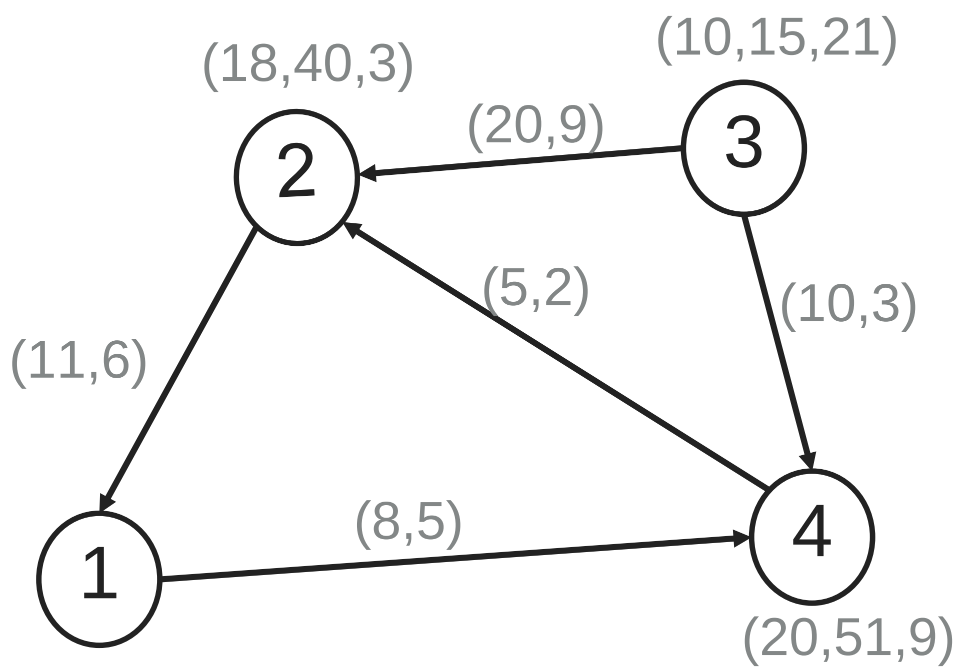
GRAPH MATCHING

- Graph matching consists in finding an association among the nodes of two or more graphs that respects a given set of constraints.
  - Common constrains are given by edges and attributes.
  - A mapping function M associates the nodes of one graph (eg. G_1) to the nodes of another graph (eg. G_2)

$G_1 = (V_1, E_1)$



$G_2 = (V_2, E_2)$



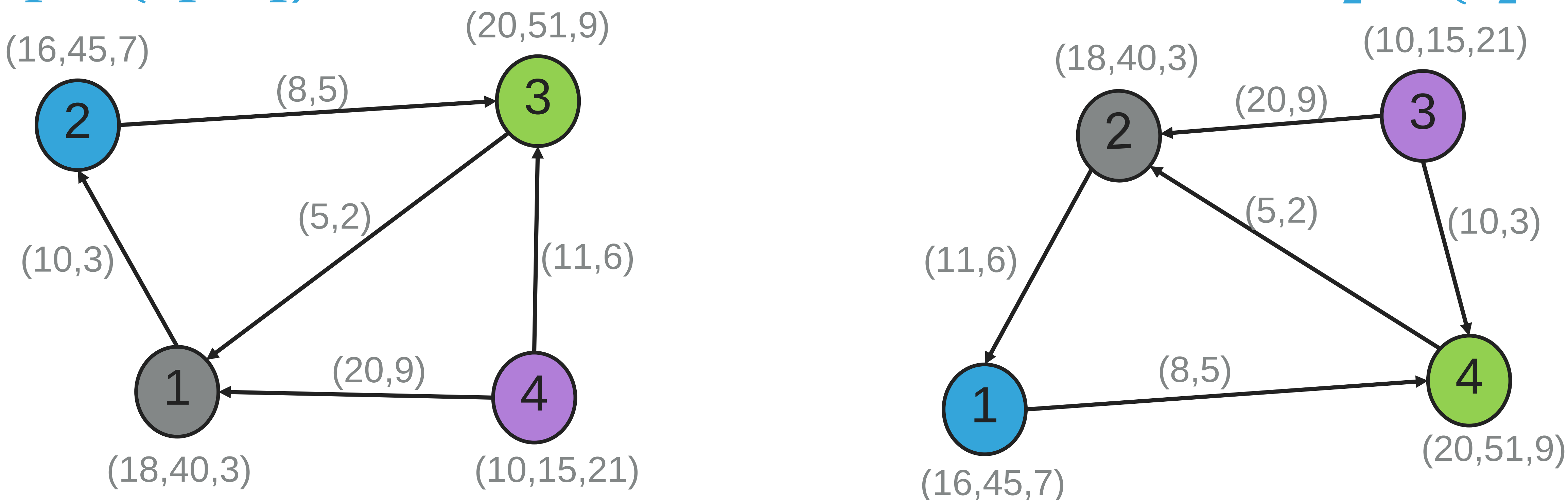
GRAPH MATCHING

- Graph matching consists in finding an association among the nodes of two or more graphs that respects a given set of constraints.
  - Common constrains are given by edges and attributes.
  - A mapping function M associates the nodes of one graph (eg. G_1) to the nodes of another graph (eg. G_2)

M = {(1,2), (2,1), (3,4), (4,3)}

$G_1 = (V_1, E_1)$

$G_2 = (V_2, E_2)$



EXACT GRAPH MATCHING

- Exact Matching Constraints:
  - Edge preserving: if two nodes in the first graph are linked by an edge, the corresponding nodes of the second graph must have an edge too
  - Non-edge preserving: preserves the absence of an edge
  - Satisfying some general constraints:
    - Isomorphism
    - Sub graph isomorphism
    - Monomorphism
    - Homomorphism

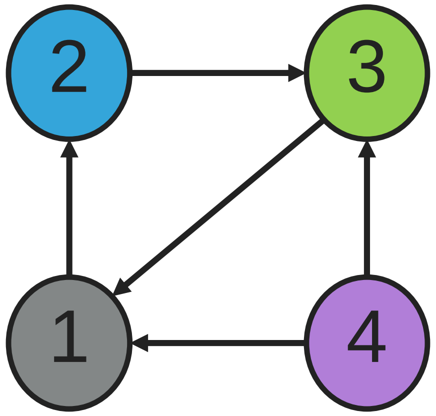
EXACT GRAPH MATCHING

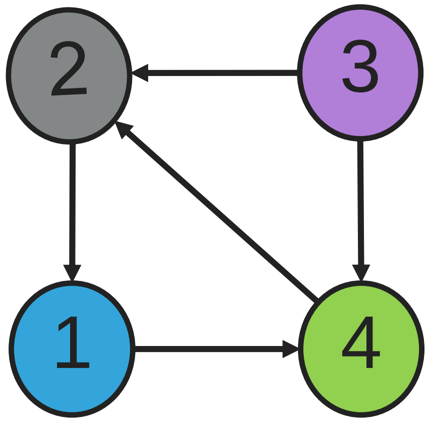
- Exact Matching Constraints:
  - Edge preserving: if two nodes in the first graph are linked by an edge, the corresponding nodes of the second graph must have an edge too
  - Non-edge preserving: preserves the absence of an edge
  - Satisfying some general constraints:
    - Isomorphism
    - Sub graph isomorphism
    - Monomorphism
    - Homomorphism

GRAPH MORPHISMS

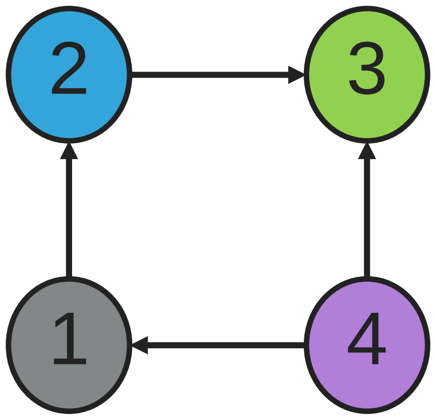
Structure Preserving Morphism:
Edge Preserving + Non-Edge Preserving

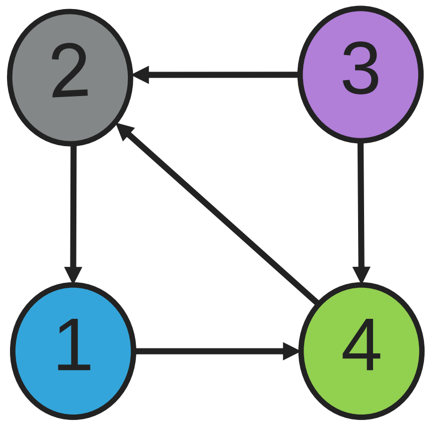
- Graph Isomorphism
  - Structure Preserving
  - Preserves edges and non-edges





- Graph Monomorphism
  - Preserves only edges

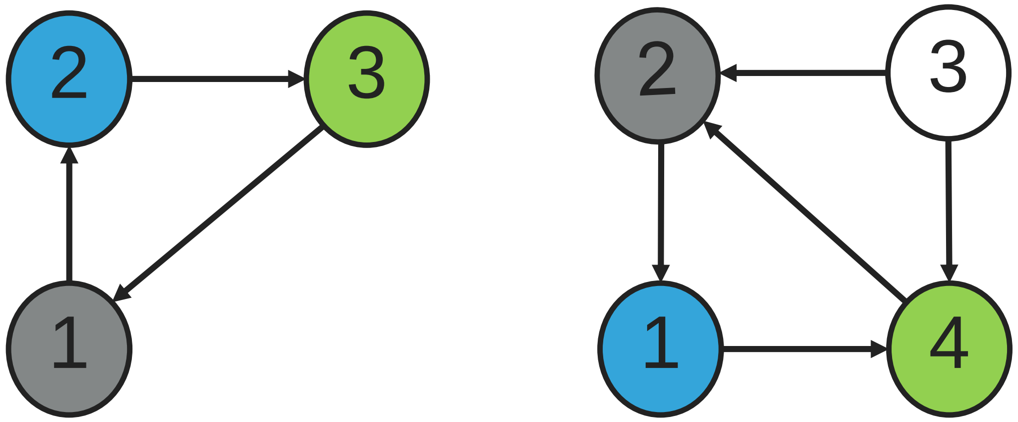




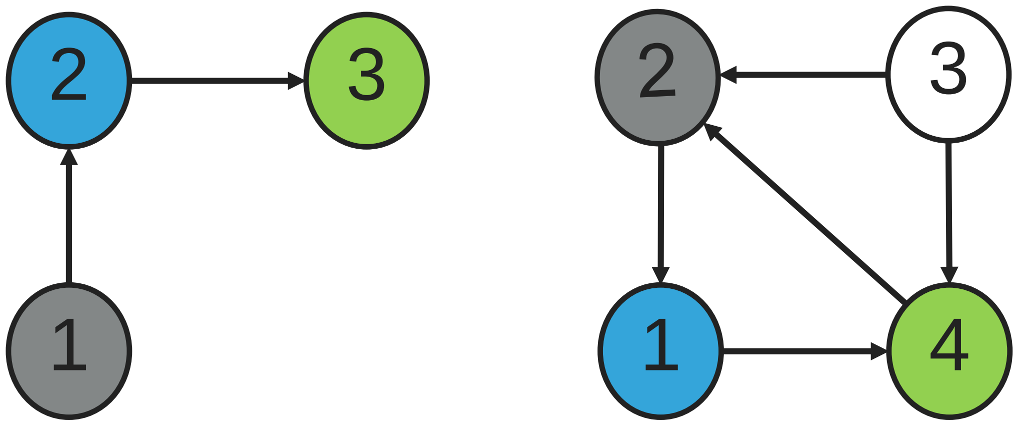
GRAPH MORPHISMS

Structure Preserving Morphism:
Edge Preserving + Non-Edge Preserving

- Subgraph Isomorphism
  - Induced Subgraph Isomorphism
    - Preserves edges and non-edges on a subgraph of the reference graph



  - Subgraph Isomorphism (Monomorph)
    - Preserves only edges on a subgraph of the reference graph



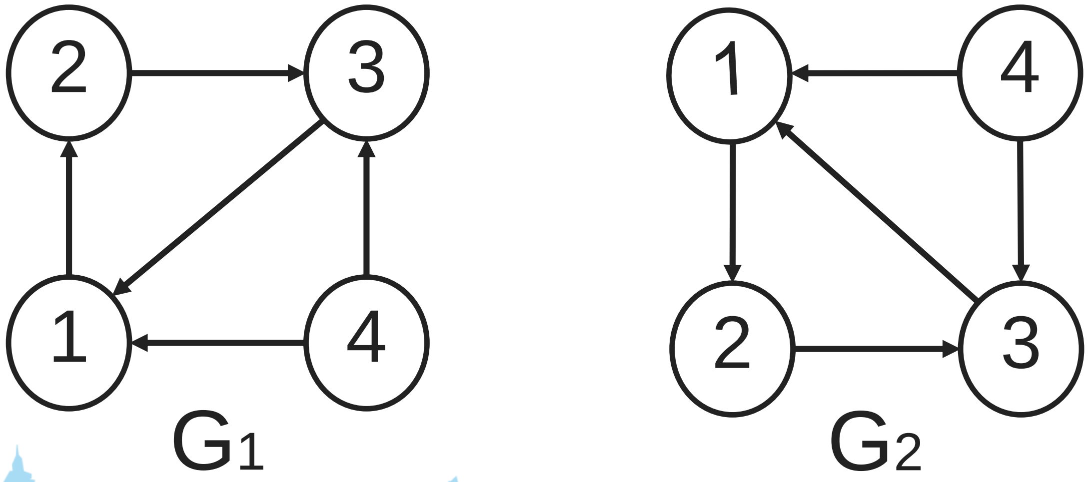
EXACT GRAPH MATCHING (EGM)

EGM: TREE SEARCH

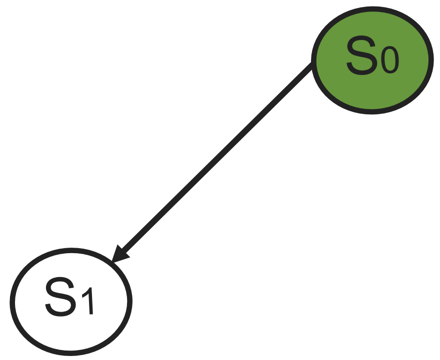
- Use of State Space Representation (SSR): Each state represents a Partial Match (consistent with the matching type)
- A partial match is iteratively expanded by adding to it new pairs of matched nodes.
- The candidate pair to add, is chosen using some necessary conditions ensuring its consistency with the matching type.
- Use some heuristic condition to prune as early as possible unfruitful search paths.

EGM: TREE EXHAUSTIVE SEARCH

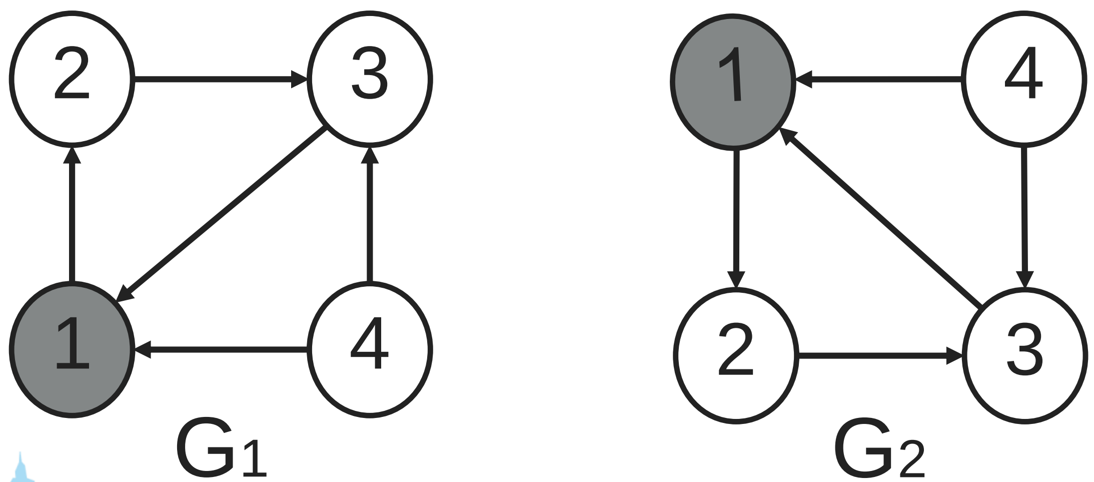
State	Mapping
S_0	$M(S_0)=\{\}$



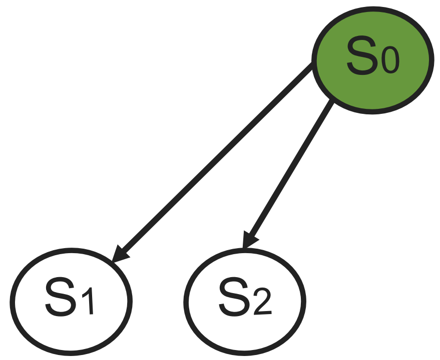
EGM: TREE EXHAUSTIVE SEARCH



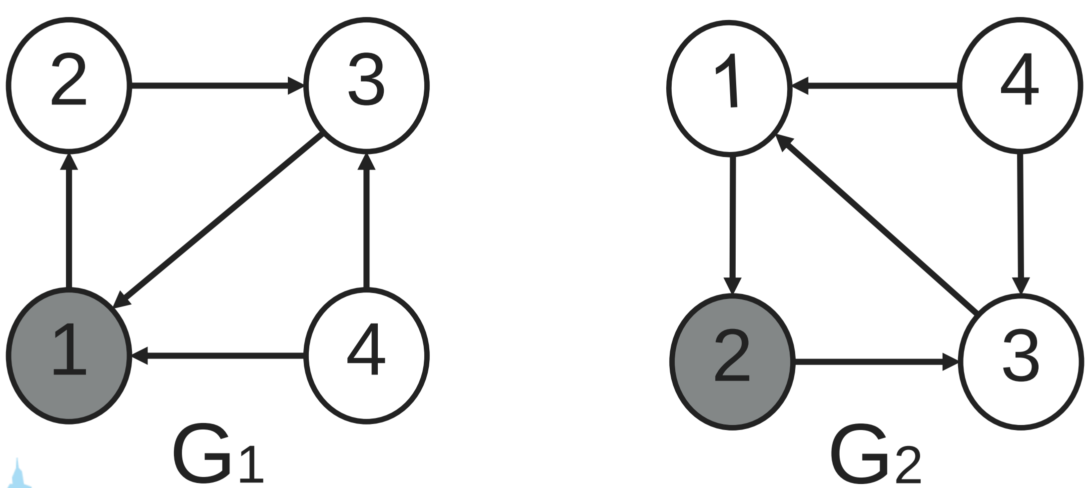
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$



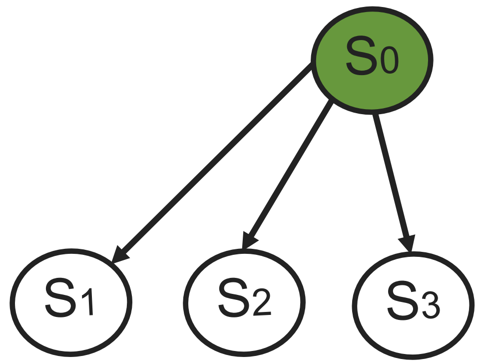
EGM: TREE EXHAUSTIVE SEARCH



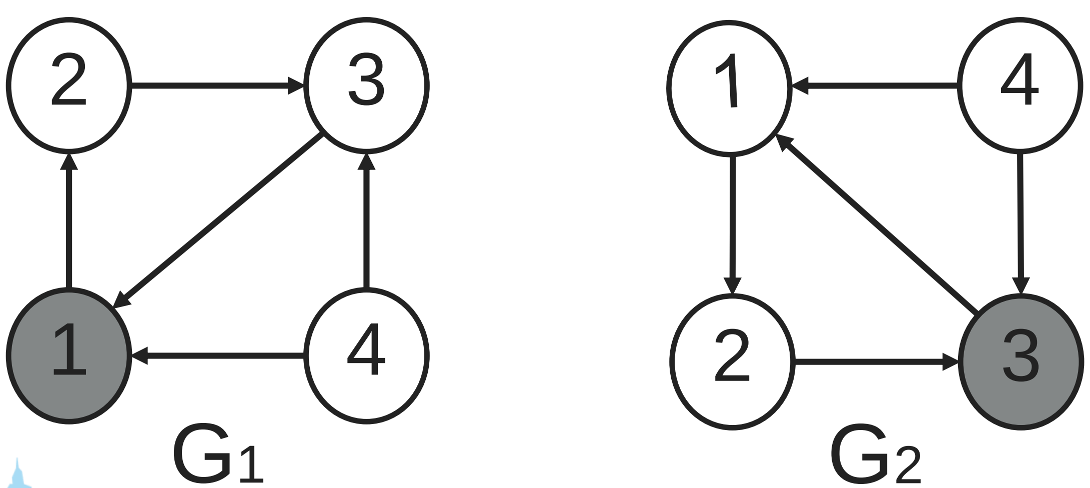
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$



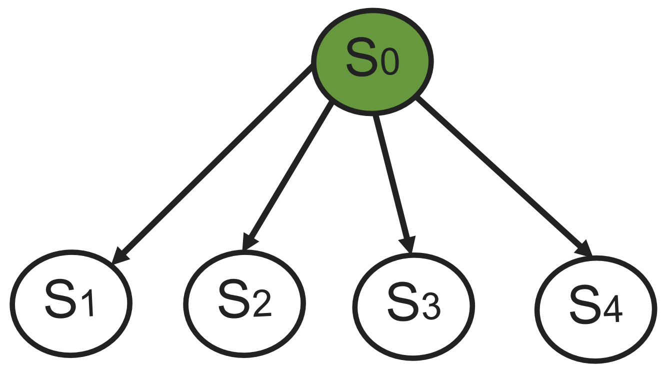
EGM: TREE EXHAUSTIVE SEARCH



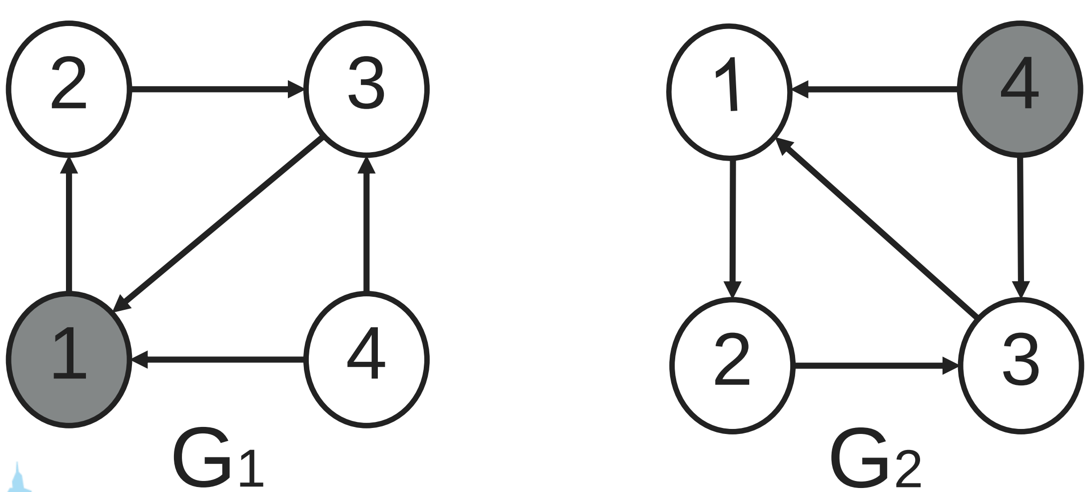
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$



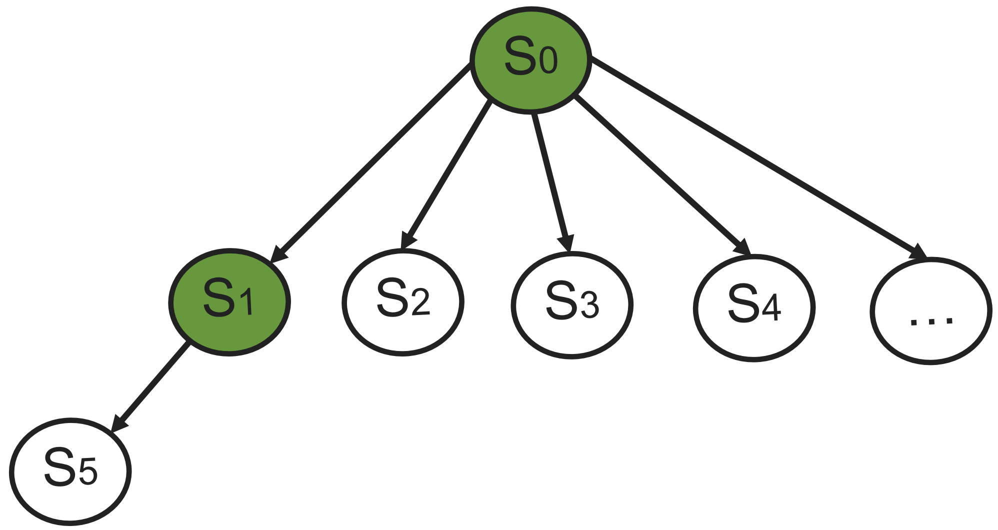
EGM: TREE EXHAUSTIVE SEARCH



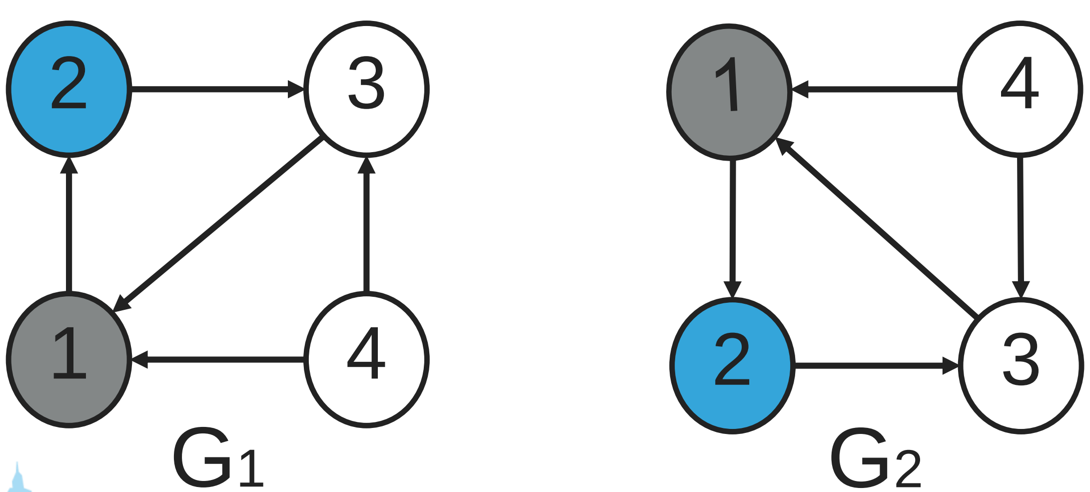
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$
S_4	$M(S_4) = \{(1, 4)\}$



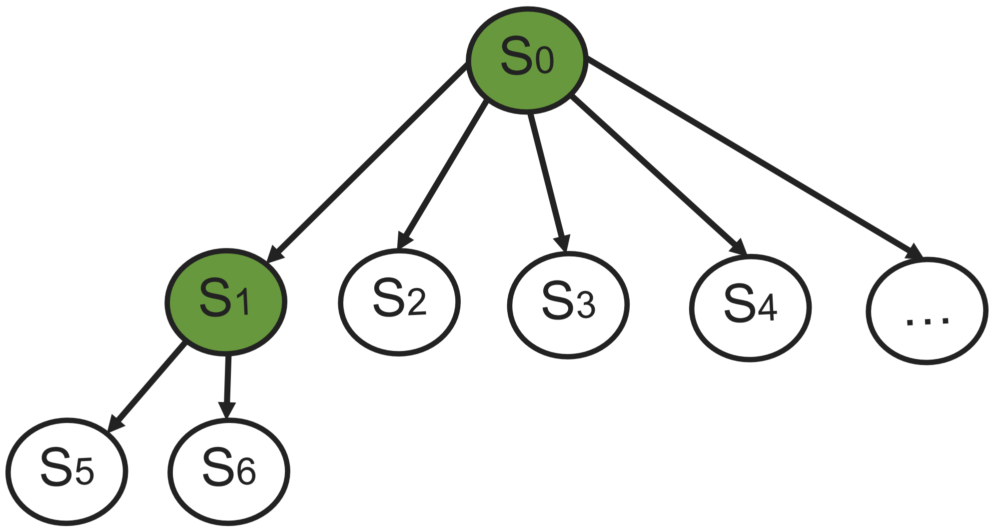
EGM: TREE EXHAUSTIVE SEARCH



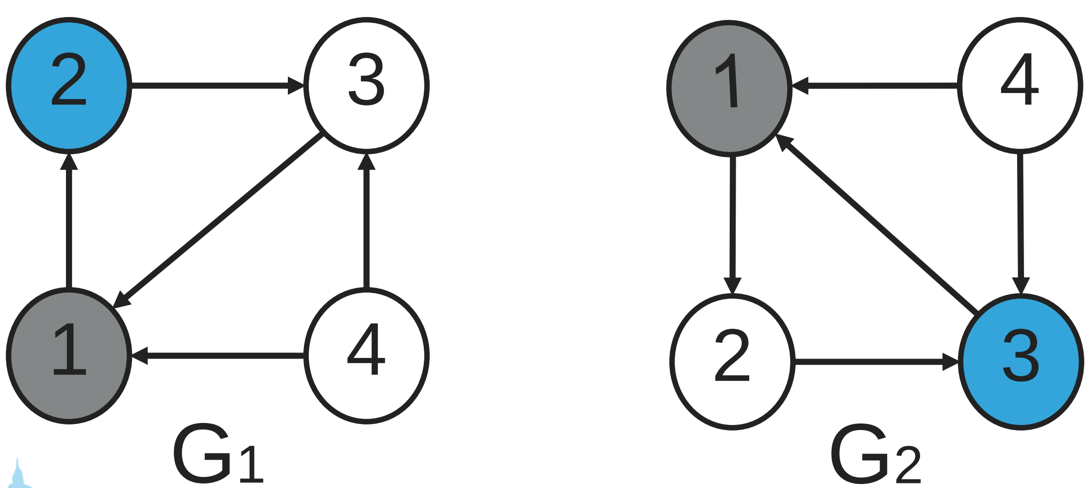
State	**Mapping**
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$
S_4	$M(S_4) = \{(1, 4)\}$
	...
S_5	$M(S_5) = \{(1, 1),(2,2)\}$



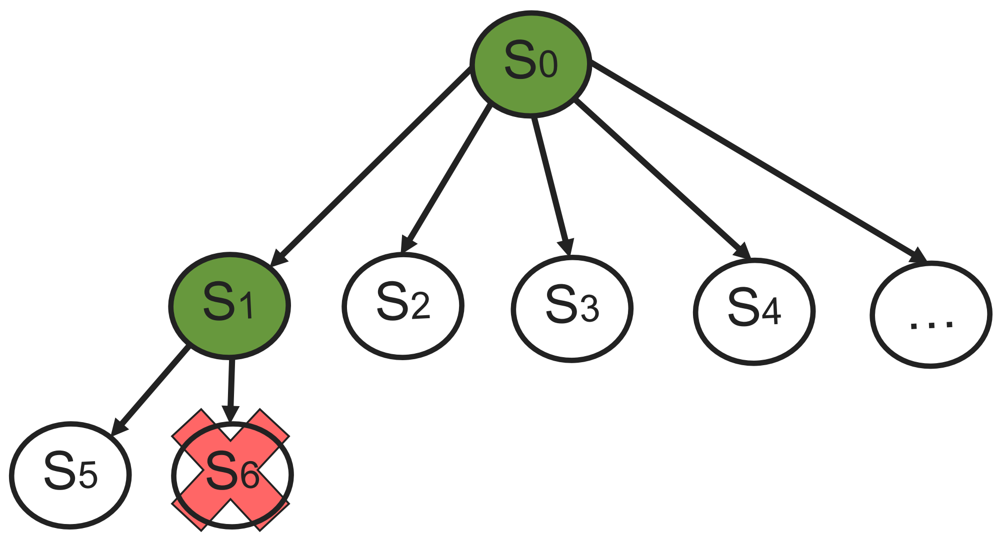
EGM: TREE EXHAUSTIVE SEARCH



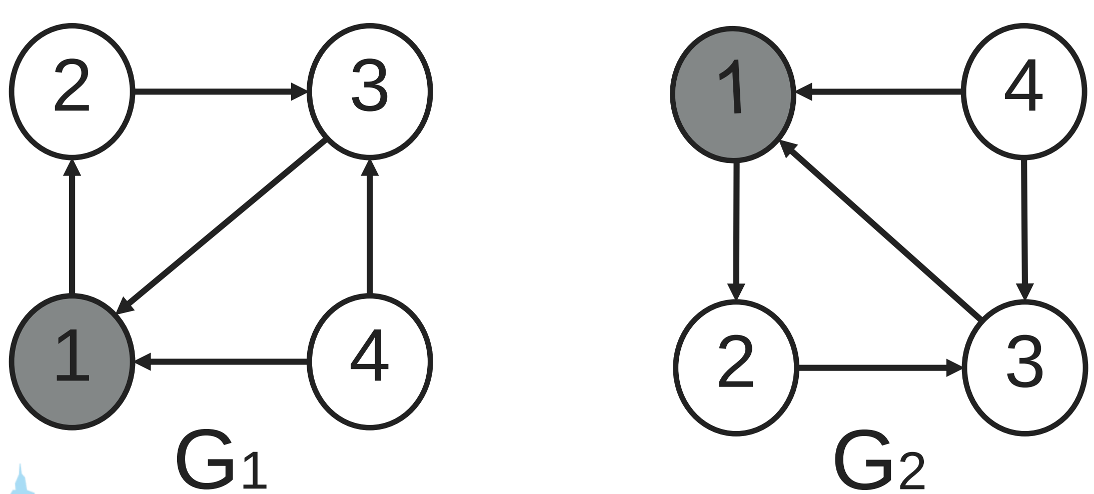
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$
S_4	$M(S_4) = \{(1, 4)\}$
	...
S_5	$M(S_5) = \{(1, 1),(2,2)\}$
S_6	$M(S_6) = \{(1, 1),(2,3)\}$



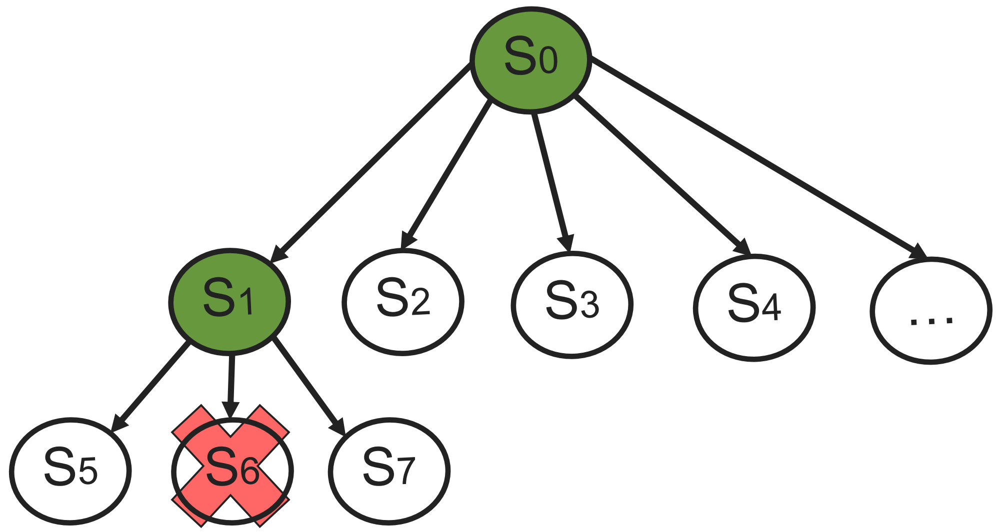
EGM: TREE EXHAUSTIVE SEARCH



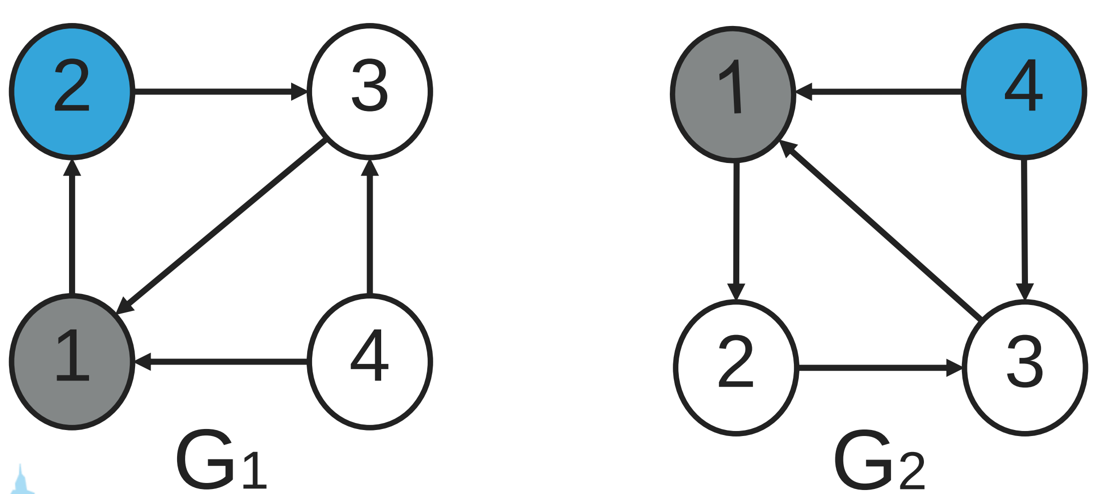
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$
S_4	$M(S_4) = \{(1, 4)\}$
	...
S_5	$M(S_5) = \{(1, 1),(2,2)\}$
~~S_6~~	~~$M(S_6) = \{(1, 1),(2,3)\}$~~



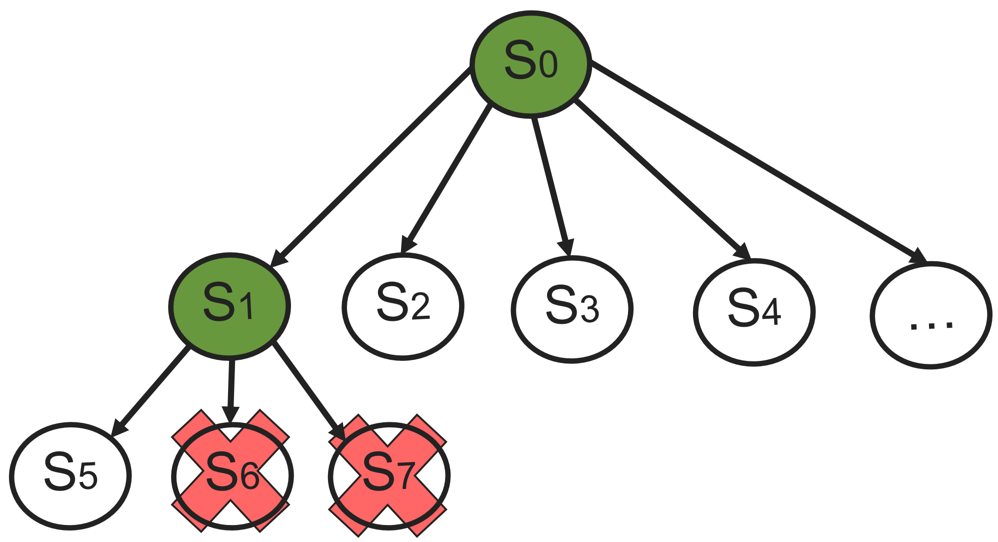
EGM: TREE EXHAUSTIVE SEARCH



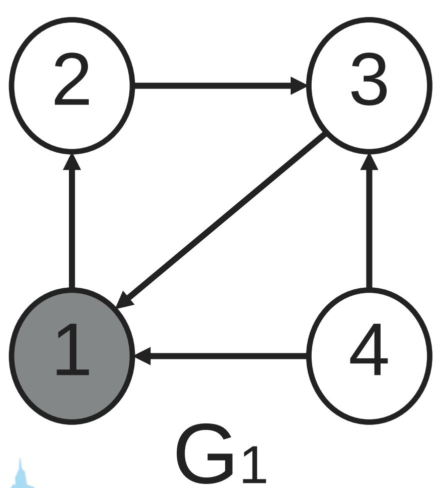
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$
S_4	$M(S_4) = \{(1, 4)\}$
	...
S_5	$M(S_5) = \{(1, 1),(2,2)\}$
~~S_6~~	~~$M(S_6) = \{(1, 1),(2,3)\}$~~
S_7	$M(S_7) = \{(1, 1),(2,4)\}$

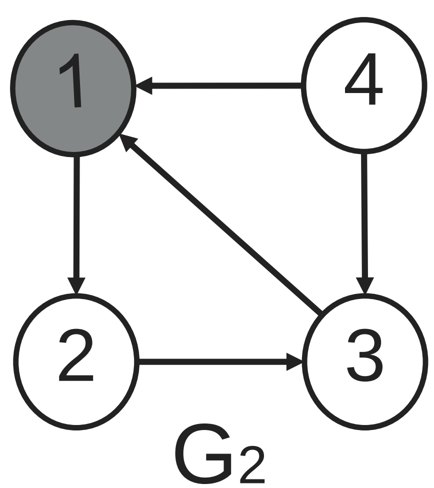


EGM: TREE EXHAUSTIVE SEARCH

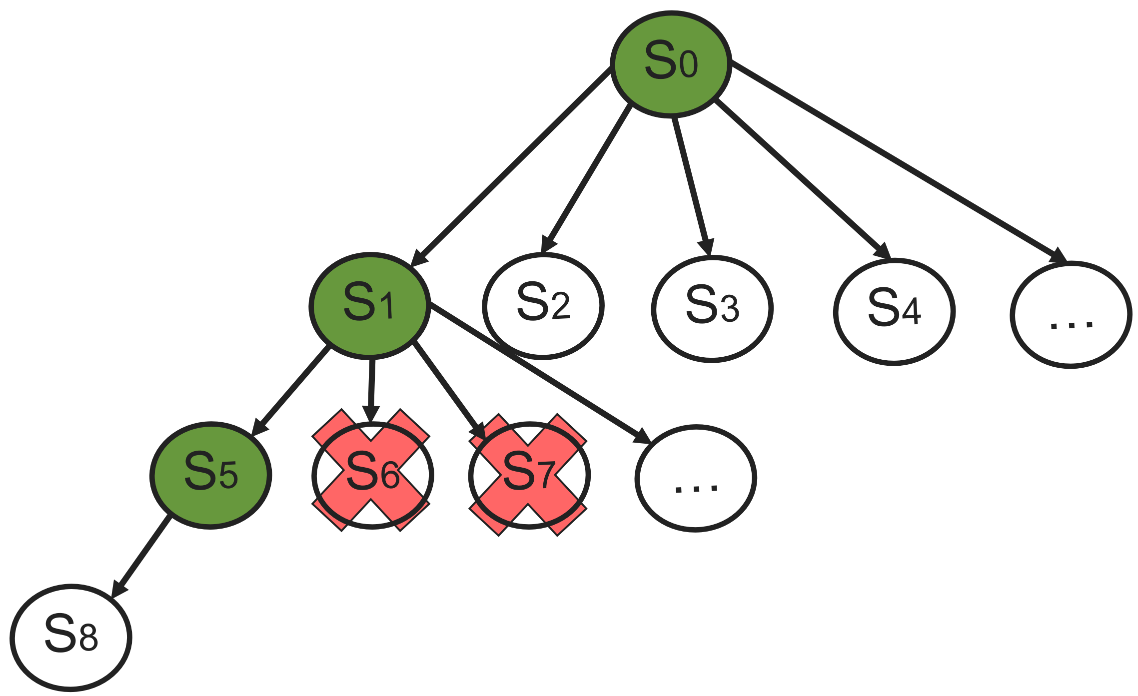


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$
S_4	$M(S_4) = \{(1, 4)\}$
	...
S_5	$M(S_5) = \{(1, 1),(2,2)\}$
~~S_6~~	~~$M(S_6) = \{(1, 1),(2,3)\}$~~
~~S_7~~	~~$M(S_7) = \{(1, 1),(2,4)\}$~~

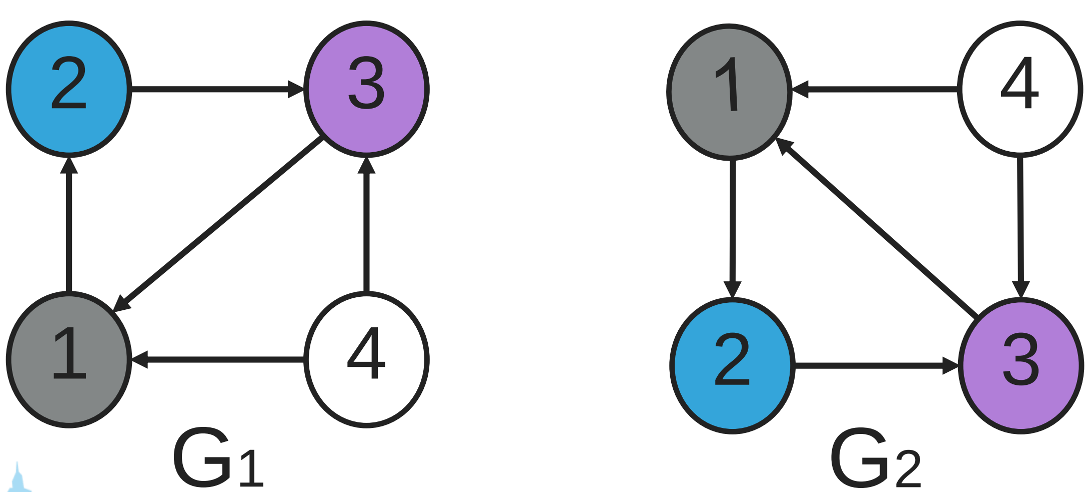




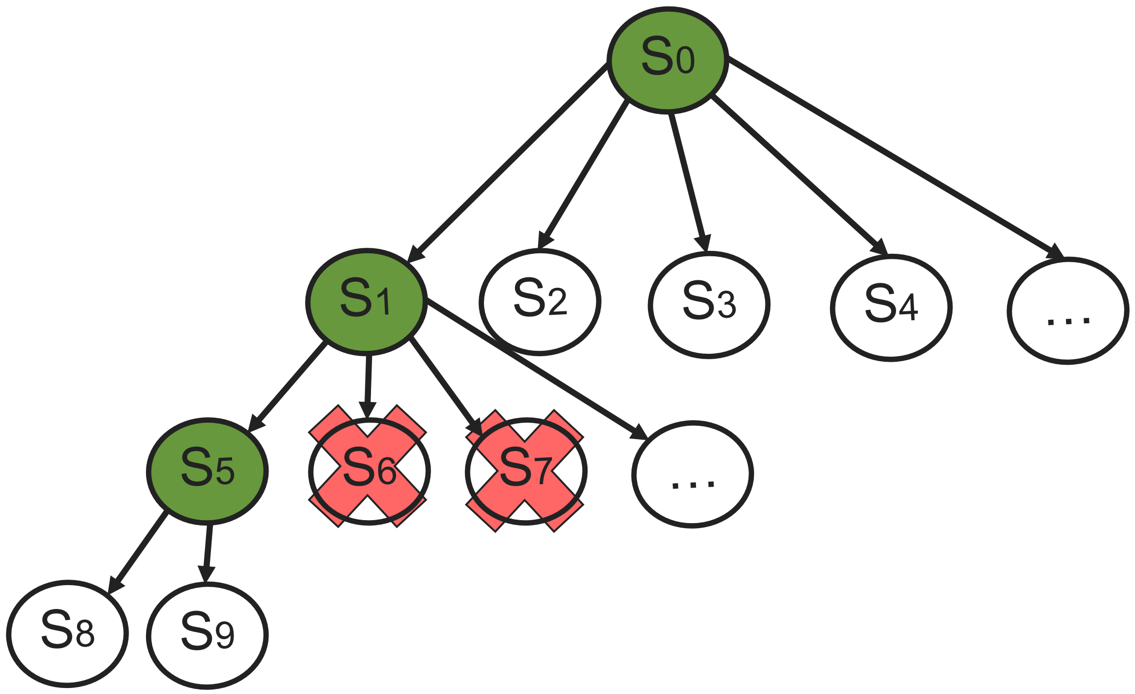
EGM: TREE EXHAUSTIVE SEARCH



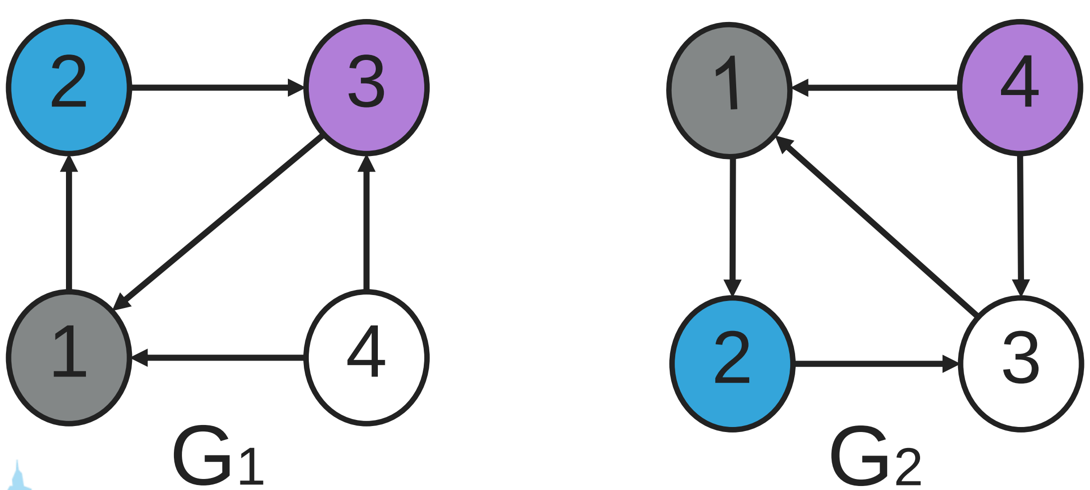
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$
S_4	$M(S_4) = \{(1, 4)\}$
	...
S_5	$M(S_5) = \{(1, 1),(2,2)\}$
~~S_6~~	~~$M(S_6) = \{(1, 1),(2,3)\}$~~
~~S_7~~	~~$M(S_7) = \{(1, 1),(2,4)\}$~~
	...
S_8	$M(S_8) = \{(1, 1),(2,2),(3,3)\}$



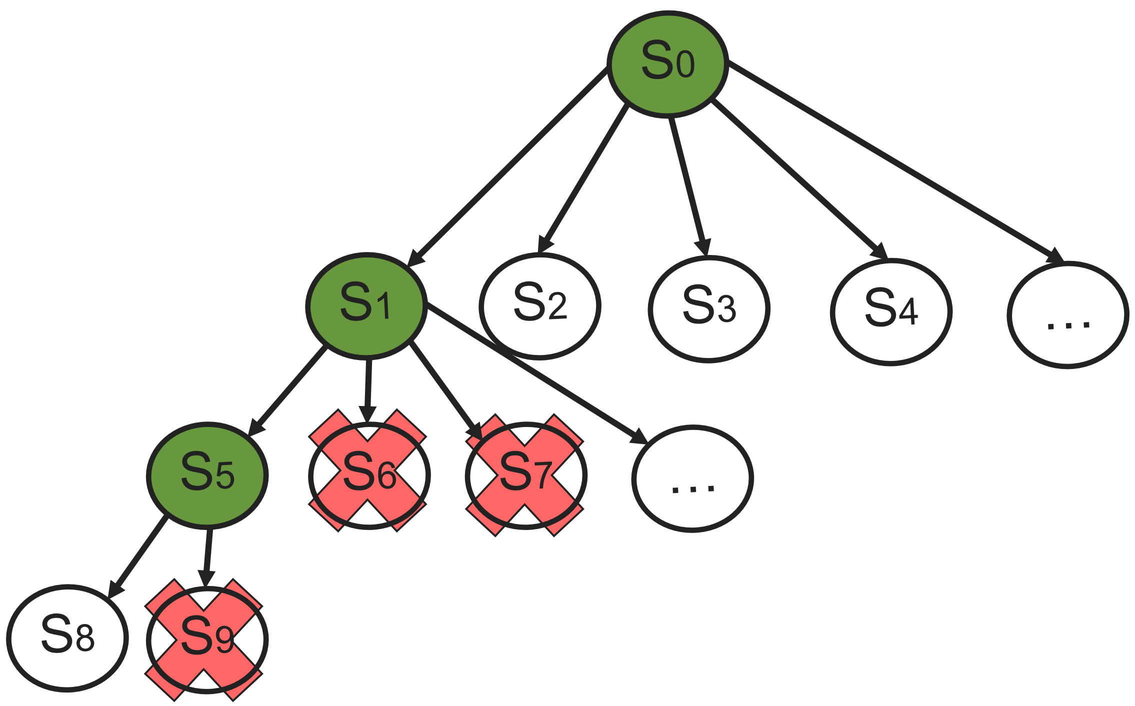
EGM: TREE EXHAUSTIVE SEARCH



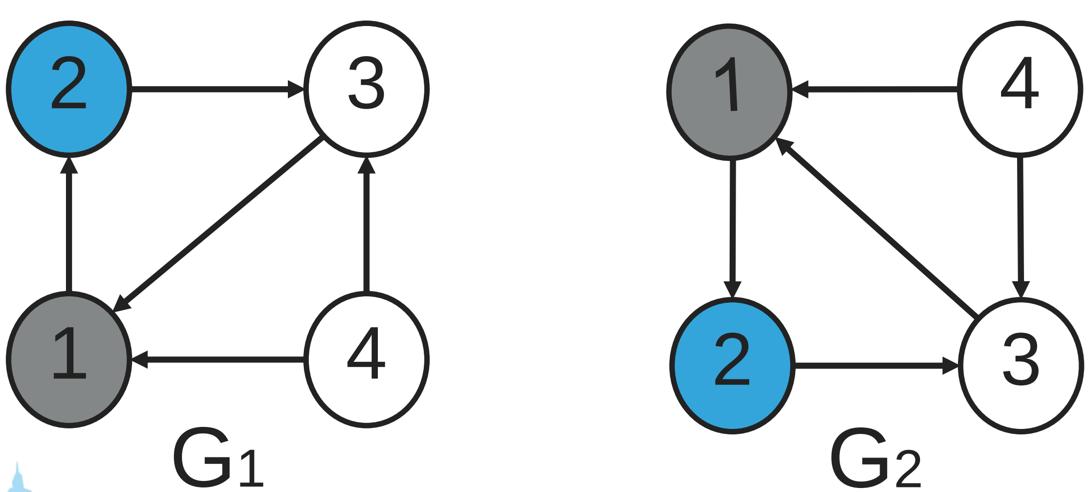
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$
S_4	$M(S_4) = \{(1, 4)\}$
	...
S_5	$M(S_5) = \{(1, 1),(2,2)\}$
~~S_6~~	~~$M(S_6) = \{(1, 1),(2,3)\}$~~
~~S_7~~	~~$M(S_7) = \{(1, 1),(2,4)\}$~~
	...
S_8	$M(S_8) = \{(1, 1),(2,2),(3,3)\}$
S_9	$M(S_9) = \{(1, 1),(2,2),(3,4)\}$



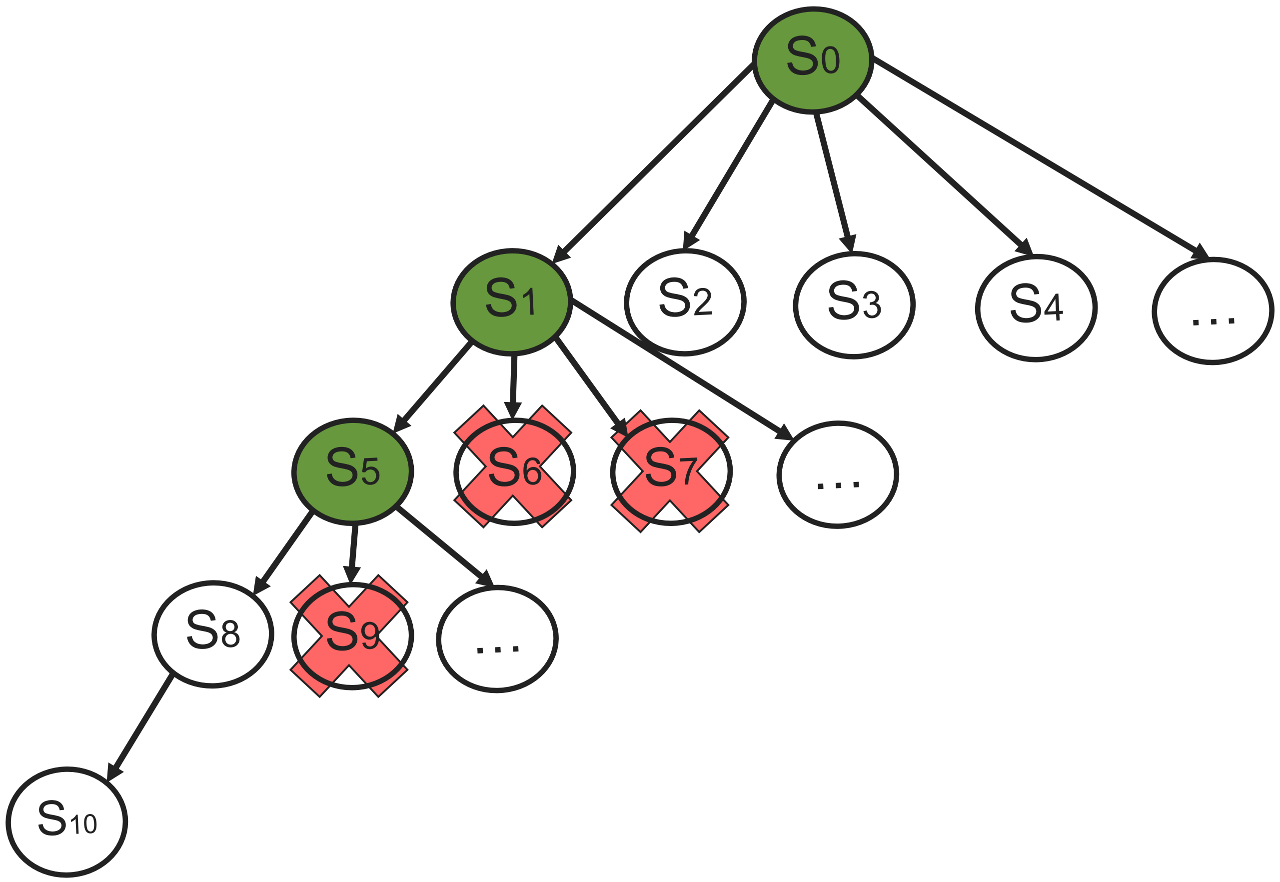
EGM: TREE EXHAUSTIVE SEARCH



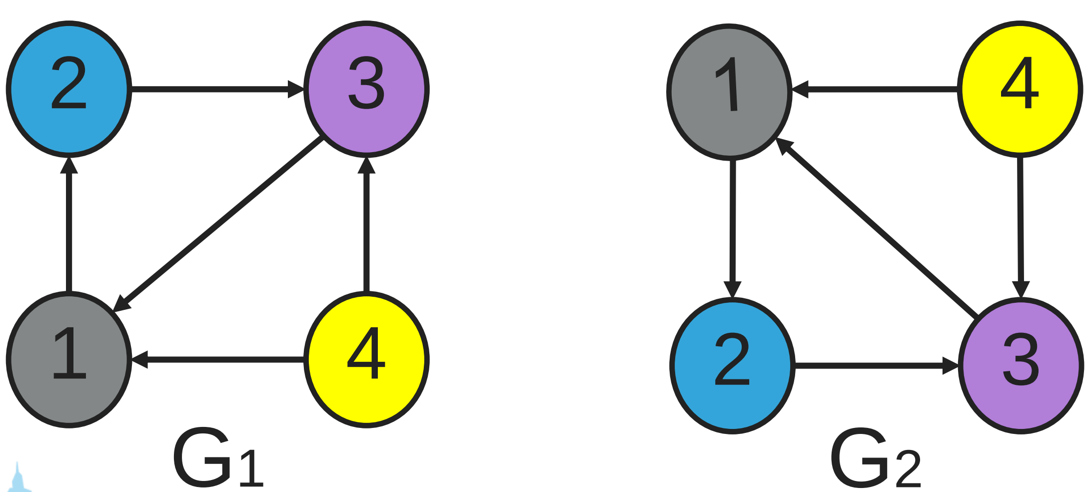
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$
S_4	$M(S_4) = \{(1, 4)\}$
	...
S_5	$M(S_5) = \{(1, 1),(2,2)\}$
~~S_6~~	~~$M(S_6) = \{(1, 1),(2,3)\}$~~
~~S_7~~	~~$M(S_7) = \{(1, 1),(2,4)\}$~~
	...
S_8	$M(S_8) = \{(1, 1),(2,2),(3,3)\}$
~~S_9~~	~~$M(S_9) = \{(1, 1),(2,2),(3,4)\}$~~



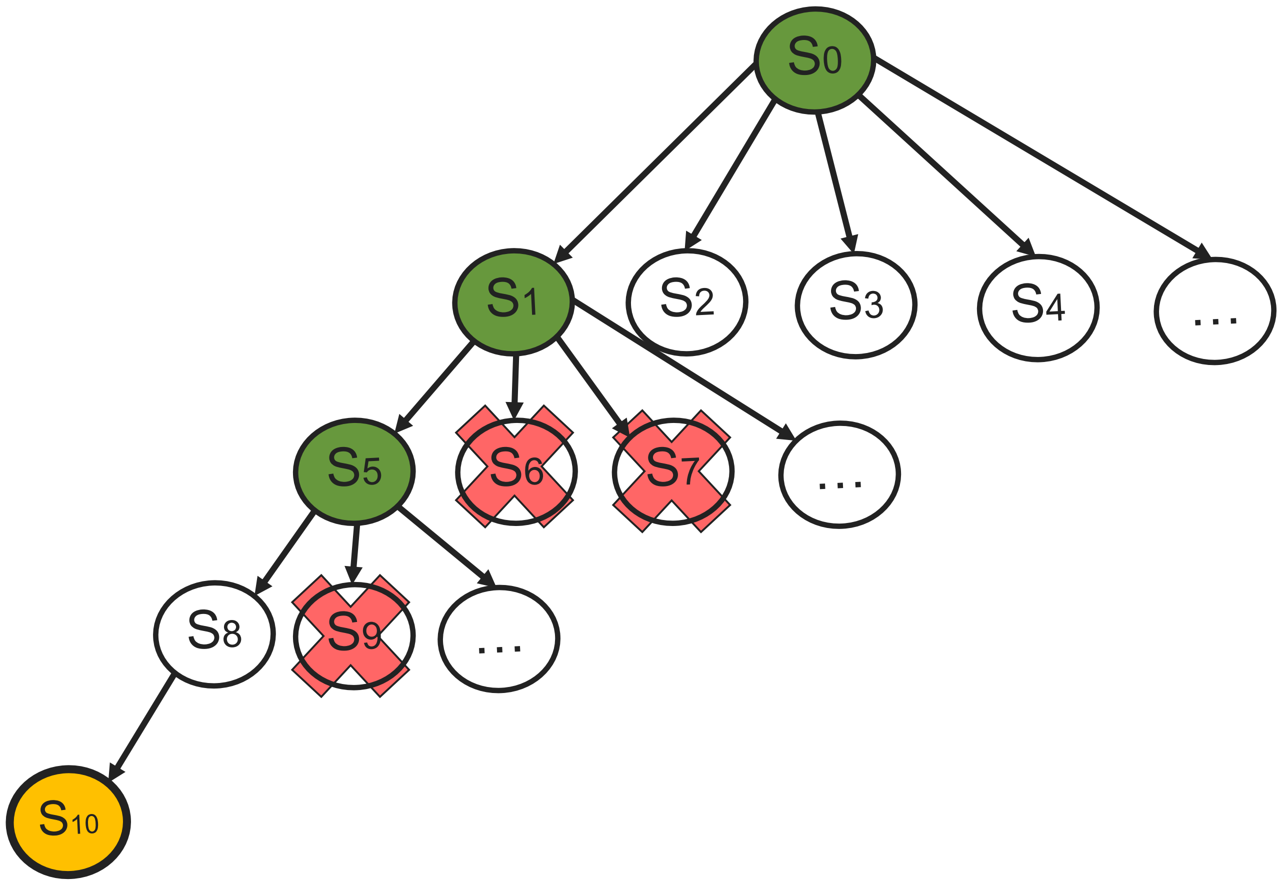
EGM: TREE EXHAUSTIVE SEARCH



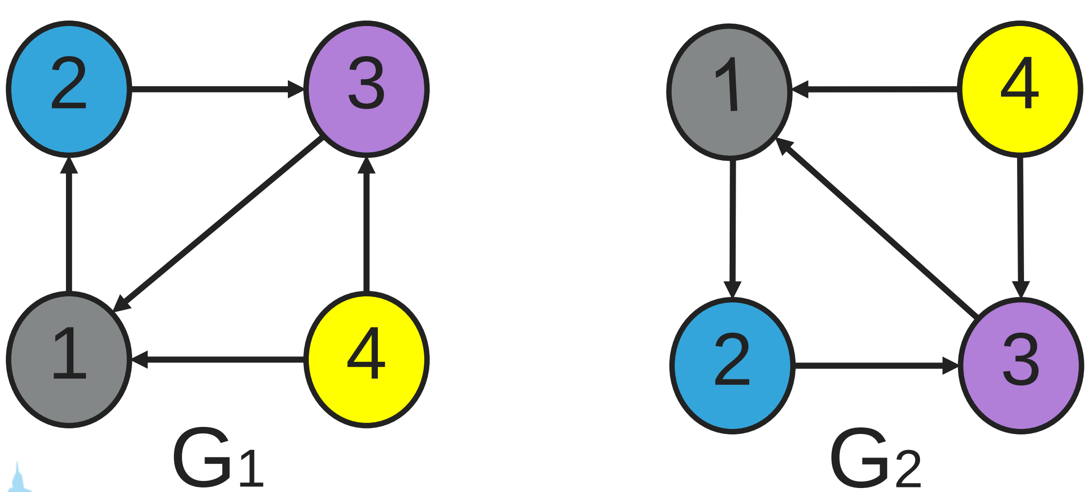
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$
S_4	$M(S_4) = \{(1, 4)\}$
	…
S_5	$M(S_5) = \{(1, 1),(2,2)\}$
~~S_6~~	~~$M(S_6) = \{(1, 1),(2,3)\}$~~
~~S_7~~	~~$M(S_7) = \{(1, 1),(2,4)\}$~~
	…
S_8	$M(S_8) = \{(1, 1),(2,2),(3,3)\}$
~~S_9~~	~~$M(S_9) = \{(1, 1),(2,2),(3,4)\}$~~
	…
S_{10}	$M(S_{10}) = \{(1, 1),(2,2),(3,3),(4,4)\}$



EGM: TREE EXHAUSTIVE SEARCH



Goal State!



State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(1, 2)\}$
S_3	$M(S_3) = \{(1, 3)\}$
S_4	$M(S_4) = \{(1, 4)\}$
	…
S_5	$M(S_5) = \{(1, 1),(2,2)\}$
~~S_6~~	~~$M(S_6) = \{(1, 1),(2,3)\}$~~
~~S_7~~	~~$M(S_7) = \{(1, 1),(2,4)\}$~~
	…
S_8	$M(S_8) = \{(1, 1),(2,2),(3,3)\}$
~~S_9~~	~~$M(S_9) = \{(1, 1),(2,2),(3,4)\}$~~
	…
S_{10}	$M(S_{10}) = \{(1, 1),(2,2),(3,3),(4,4)\}$

EGM: FROM EXHAUSTIVE TO INFORMED SEARCH

- Exhaustive (Brute force) search is too expensive
- The idea is to add heuristics for pruning the tree without pruning solutions (admissibility)
- For each couple of nodes we evaluate in advance (look-ahead) if a given state have no successors, so as to avoid its further expansion
- This is done with feasibility rules, inducing necessary conditions for having successors of a given state
- The higher the look-ahead the lower the number of explored states
- A higher look-ahead imposes computations on the current state
- The more asymmetric is the graph the more is the reduction of the matching time

ULLMANN'S ALGORITHM (1976)

- Based on tree search with backtracking
- Finds isomorphism and subgraph isomorphism
- Pruning by a bit matrix M for each state
  - Mij=1 iff the matching of ni and nj is possible
- Refinement: delete 1's from M of the current state
- Then, the state is extended using the node pairs corresponding to the remaining 1's

- PROS: The solution is found for each pair of graphs (the refinement procedures converges in a finite number of steps);
- CONS: Exponential time; very high memory requirements;

ULLMANN'S ALGORITHM (1976)

- Based on tree search with backtracking
- Finds isomorphism and subgraph isomorphism
- Pruning by a bit matrix M for each state
    - Mij=1 iff the matching of ni and nj is possible
- Refinement: delete 1's from M of the current state
- Then, the state is extended using the node pairs corresponding to the remaining 1's

- PROS: The solution is found for each pair of graphs (the refinement procedures converges in a finite number of steps);

- CONS: Exponential time; very high memory requirements;

VF2 (2004)

- The problem is formulated in terms of State Space Representation, where each state represents a partial mapping solution;
- A depth-first search is used;
- Five Feasibility rules are defined in order to prune the state space. These rules take into account:
  - the consistency of the current solution;
  - the consistency of the "future" solutions (1- and 2-look-ahead)

VF2 (2004)

- The problem is formulated in terms of State Space Representation, where each state represents a partial mapping solution;
- A depth-first search is used;
- Five Feasibility rules are defined in order to prune the state space. These rules take into account:
  - the consistency of the current solution;
  - the consistency of the “future” solutions (1- and 2-look-ahead)

Outline of VF2

M(s) is the partial mapping of state s, (current set of pairs of matched nodes)

P(s) is the set of candidate pairs for extending the state s

F(s,n,m) is the feasibility predicate, which indicates whether the addition of pair (n,m) to state s can produce an inconsistent mapping

VF2 (2004)

- In more details, the feasibility rules are defined as follows:

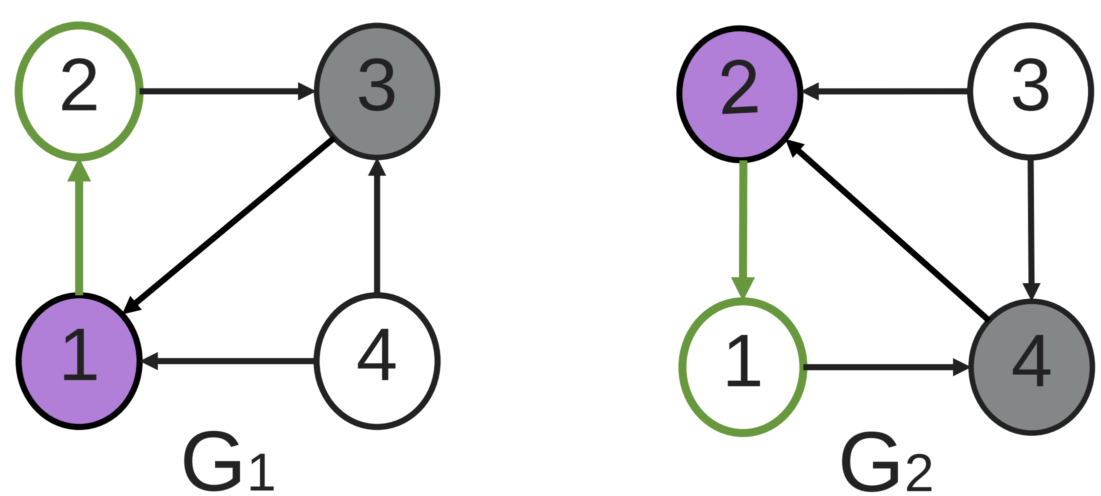
$$F(s,n,m) = F_{\text{syn}}(s,n,m) \wedge F_{\text{sem}}(s,n,m)$$

$$F_{\text{syn}}(s,n,m) = R_{\text{pred}} \wedge R_{\text{succ}} \wedge R_{\text{in}} \wedge R_{\text{out}} \wedge R_{\text{new}}$$

$$\begin{aligned} R_{pred}(s,n,m) \iff & \\ & \forall n' \in \text{Pred}(G_1,n) \cap M_1(s) \\ & \exists m' \in \text{Pred}(G_2,m) \cap M_2(s) : (n',m') \in M(s) \\ \wedge \; & \forall m' \in \text{Pred}(G_2,m) \cap M_2(s) \\ & \exists n' \in \text{Pred}(G_1,n) \cap M_1(s) : (n',m') \in M(s) \end{aligned}$$

- R_{pred} checks that, if the candidate couple (n,m) has predecessors already included in M(s), these predecessors must be coupled each other in M(s).

- M(s) = {(1,2),(3,4)}
- $M_1(s) = \{1,3\}$
- $M_2(s) = \{2,4\}$
- n = 2, m = 1



VF2 (2004)

- In more details, the feasibility rules are defined as follows:

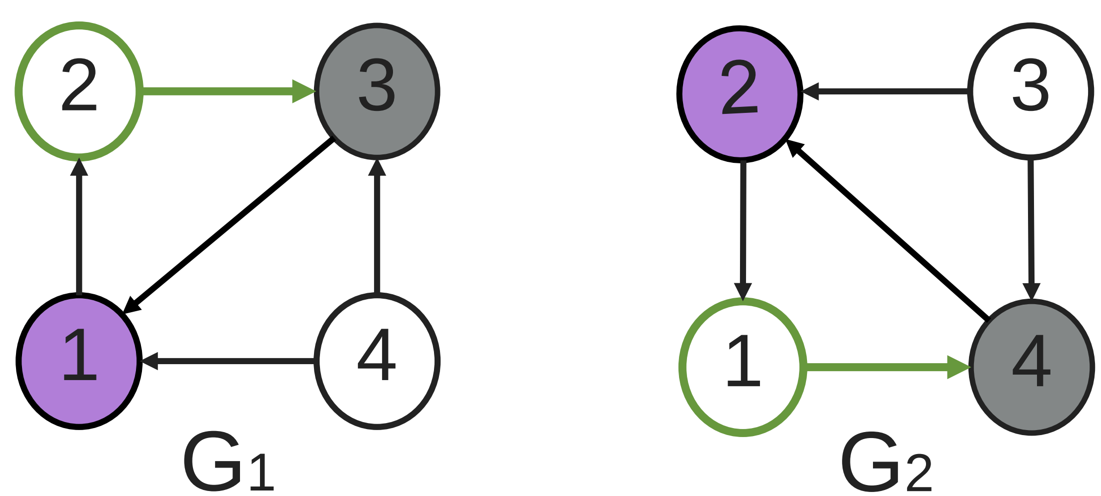
$$F(s,n,m) = F_{\text{syn}}(s,n,m) \wedge F_{\text{sem}}(s,n,m)$$

$$F_{\text{syn}}(s,n,m) = R_{\text{pred}} \wedge R_{\text{succ}} \wedge R_{\text{in}} \wedge R_{\text{out}} \wedge R_{\text{new}}$$

$$\begin{aligned} R_{succ}(s,n,m) \iff & \\ & \forall n' \in \text{Succ}(G_1,n) \cap M_1(s) \\ & \exists m' \in \text{Succ}(G_2,m) \cap M_2(s) : (n',m') \in M(s) \\ \wedge \ & \forall m' \in \text{Succ}(G_2,m) \cap M_2(s) \\ & \exists n' \in \text{Succ}(G_1,n) \cap M_1(s) : (n',m') \in M(s) \end{aligned}$$

- R_{succ} checks that, if the candidate couple (n,m) has successors already included in M(s), these successors must be coupled each other in M(s).

- M(s) = {(1,2),(3,4)}
- $M_1(s)$ = {1,3}
- $M_2(s)$ = {2,4}
- n = 2, m = 1



VF2 (2004)

- In more details, the feasibility rules are defined as follows:

$$F(s,n,m) = F_{\text{syn}}(s,n,m) \wedge F_{\text{sem}}(s,n,m)$$

$$F_{\text{syn}}(s,n,m) = R_{\text{pred}} \wedge R_{\text{succ}} \wedge R_{\text{in}} \wedge R_{\text{out}} \wedge R_{\text{new}}$$

  - The remaining three rules are introduced for pruning the search tree:
    - R_{in} and R_{out} perform a 1-look-ahead
    - R_{new} a 2-look- ahead

VF2 (2004)

- In more details, the feasibility rules are defined as follows:

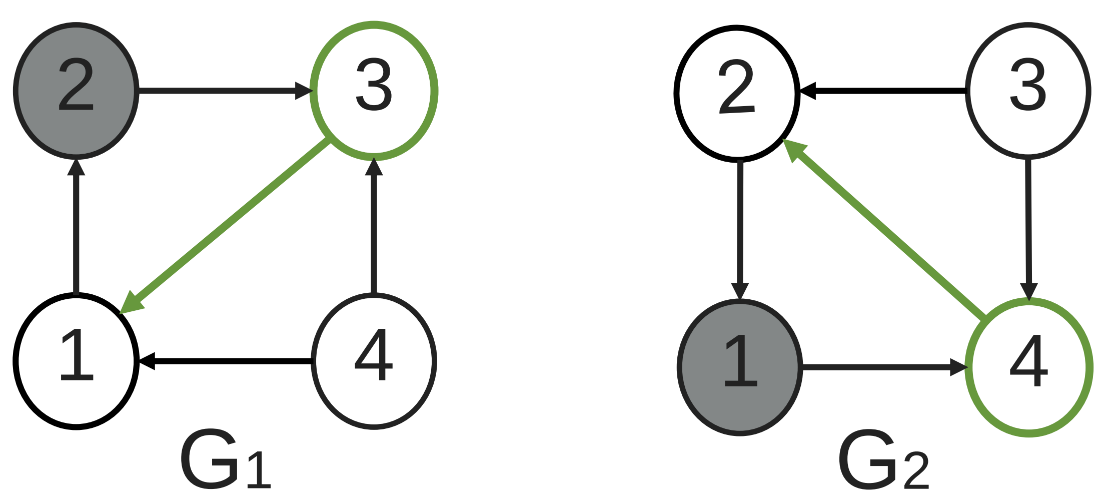
$$F(s,n,m) = F_{\text{syn}}(s,n,m) \wedge F_{\text{sem}}(s,n,m)$$

$$F_{\text{syn}}(s,n,m) = R_{\text{pred}} \wedge R_{\text{succ}} \wedge R_{\text{in}} \wedge R_{\text{out}} \wedge R_{\text{new}}$$

$$R_{\text{in}}(s,n,m) \iff$$
$$(\text{Card}(\text{Succ}(G_1,n) \cap T_1^{\text{in}}(s)) \geq \text{Card}(\text{Succ}(G_2,m) \cap T_2^{\text{in}}(s))) \wedge$$
$$(\text{Card}(\text{Pred}(G_1,n) \cap T_1^{\text{in}}(s)) \geq \text{Card}(\text{Pred}(G_2,m) \cap T_2^{\text{in}}(s))),$$

- $M(s) = \{(2,1)\}$
- $M_1(s) = \{1\}$
- $M_2(s) = \{2\}$
- $T_1^{in}(s) = \{1\}$
- $T_2^{in}(s) = \{2\}$
- $T_1^{out}(s) = \{3\}$
- $T_2^{out}(s) = \{4\}$
- $n = 3$, $m = 4$

- R_{in} implement a 1-look-ahead
  - R_{in} checks that, if the candidate couple (n,m) has predecessors or successors included in $T^{in}(s)$, their number must be equal.



VF2 (2004)

- In more details, the feasibility rules are defined as follows:

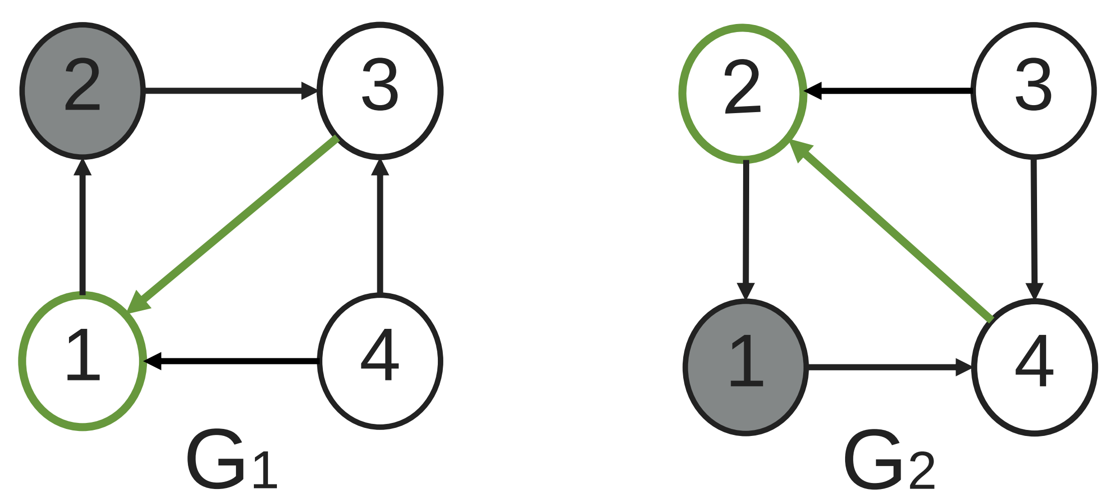
$$F(s,n,m) = F_{\text{syn}}(s,n,m) \wedge F_{\text{sem}}(s,n,m)$$

$$F_{\text{syn}}(s,n,m) = R_{\text{pred}} \wedge R_{\text{succ}} \wedge R_{\text{in}} \wedge R_{\text{out}} \wedge R_{\text{new}}$$

$$R_{\text{out}}(s,n,m) \Longleftrightarrow$$
$$(\text{Card}(\text{Succ}(G_1,n) \cap T_1^{\text{out}}(s)) \geq \text{Card}(\text{Succ}(G_2,m) \cap T_2^{\text{out}}(s))) \wedge$$
$$(\text{Card}(\text{Pred}(G_1,n) \cap T_1^{\text{out}}(s)) \geq \text{Card}(\text{Pred}(G_2,m) \cap T_2^{\text{out}}(s))),$$

- R_{out} implement a 1-look-ahead
  - R_{out} checks that, if the candidate couple (n,m) has predecessors or successors in Tout(s), their number must be equal.

- M(s) = {(2,1)}
- $M_1(s) = \{1\}$
- $M_2(s) = \{2\}$
- $T_1^{in}(s) = \{1\}$
- $T_2^{in}(s) = \{2\}$
- $T_1^{out}(s) = \{3\}$
- $T_2^{out}(s) = \{4\}$
- Couple (1, 2)



VF2 (2004)

- In more details, the feasibility rules are defined as follows:

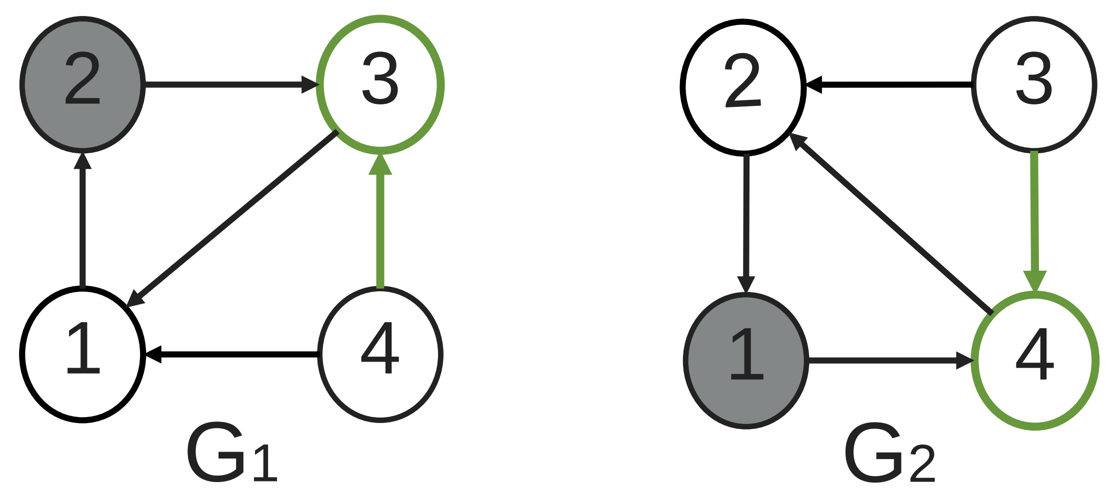
$$F(s,n,m) = F_{\text{syn}}(s,n,m) \wedge F_{\text{sem}}(s,n,m)$$

$$F_{\text{syn}}(s,n,m) = R_{\text{pred}} \wedge R_{\text{succ}} \wedge R_{\text{in}} \wedge R_{\text{out}} \wedge R_{\text{new}}$$

$$R_{\text{new}}(s,n,m) \Longleftrightarrow$$
$$\text{Card}(\tilde{N}_1(s) \cap \text{Pred}(G_1,n)) \geq \text{Card}(\tilde{N}_2(s) \cap \text{Pred}(G_2,n)) \wedge$$
$$\text{Card}(\tilde{N}_1(s) \cap \text{Succ}(G_1,n)) \geq \text{Card}(\tilde{N}_2(s) \cap \text{Succ}(G_2,n)).$$

- $M(s) = \{(2,1)\}$
- $M_1(s) = \{1\}$
- $M_2(s) = \{2\}$
- $T_1^{in}(s) = \{1\}$
- $T_2^{in}(s) = \{2\}$
- $T_1^{out}(s) = \{3\}$
- $T_2^{out}(s) = \{4\}$
- $T_1^{out}(s) = \{3\}$
- $T_2^{out}(s) = \{4\}$
- $N_1(s) = \{4\}$
- $N_2(s) = \{3\}$
- Couple (3, 4)

- R_{new} a 2-look- ahead
  - R_{new} checks that, if the candidate couple (n,m) has predecessors or successors included in N(s), their number must be equal.



VF2: EXAMPLE

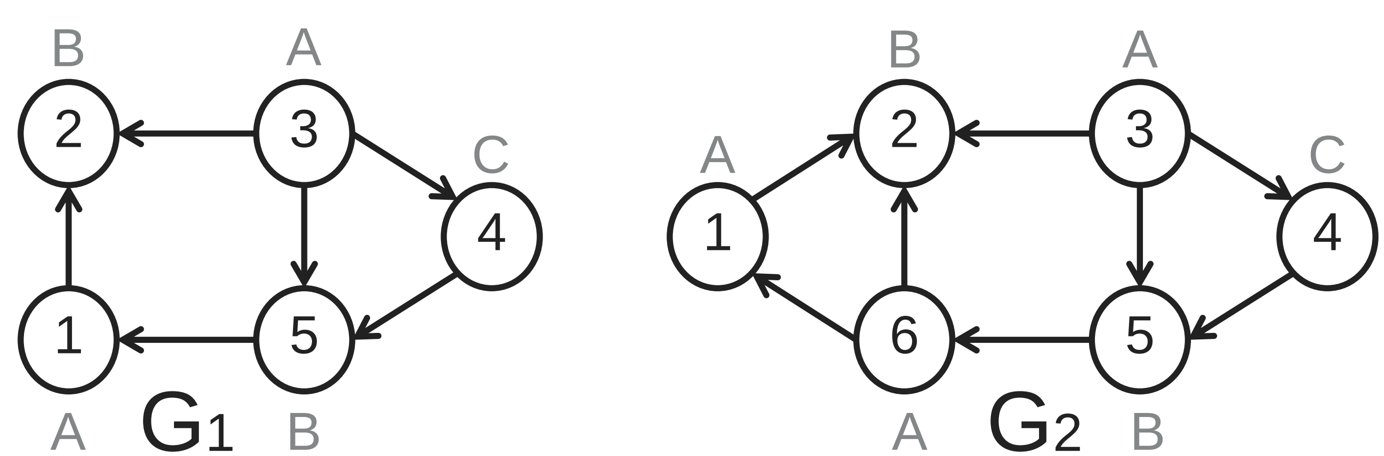
State	Mapping
S_0	$M(S_0)=\{\}$

T_1^{in}

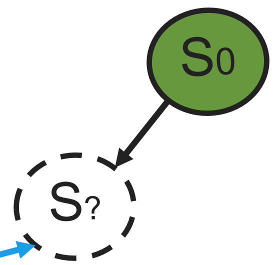
T_2^{in}

T_1^{out}

T_2^{out}



VF2: EXAMPLE



Feasibility Check for S_0 U (1,1)

$$F(s_0,1,1) = F_{syn}(s_0,1,1) \wedge F_{sem}(s_0,1,1)$$

$$F_{syn}(s_0,1,1) = R_{pred} \wedge R_{succ} \wedge R_{in} \wedge R_{out} \wedge R_{new}$$

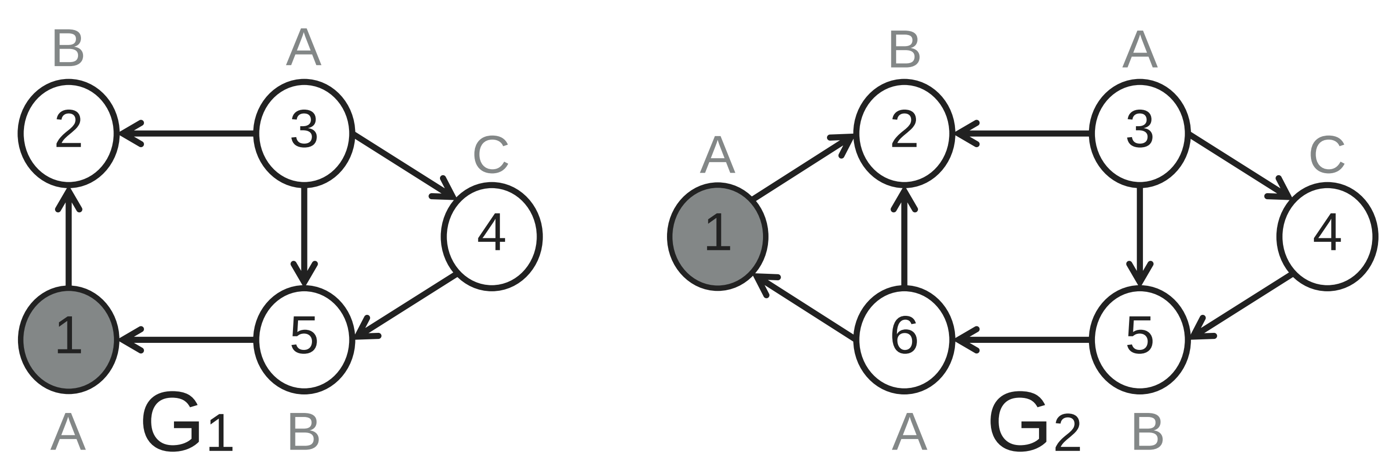
State	Mapping
S_0	$M(S_0)=\{\}$
$S_?$	$M(S_?)=\{(1,1)\}$

T_1^{in}

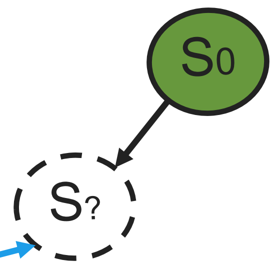
T_2^{in}

T_1^{out}

T_2^{out}



VF2: EXAMPLE



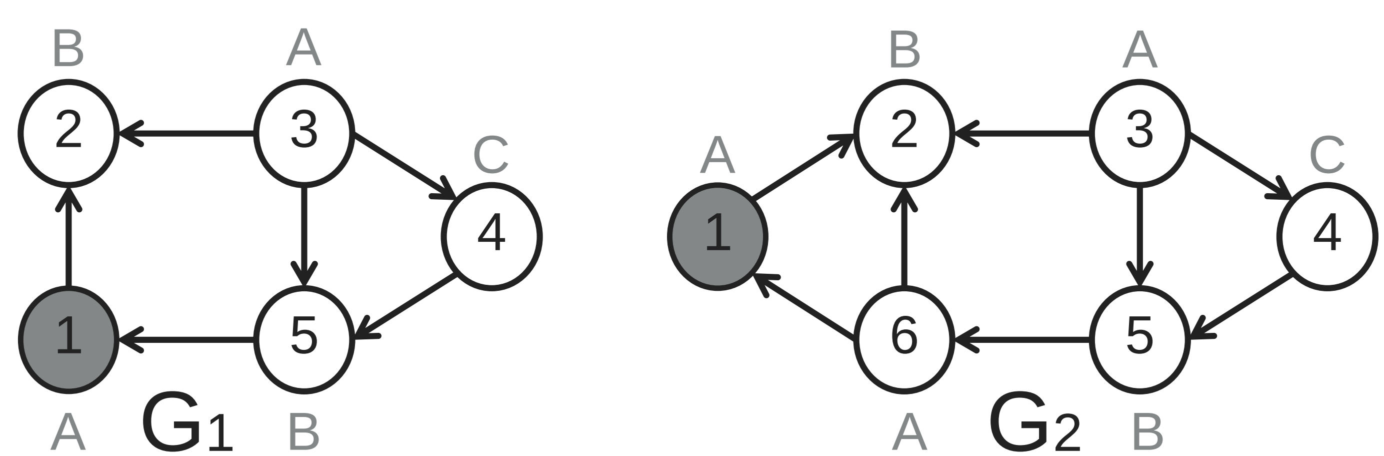
Feasibility Check for $S_0 \cup (1,1)$

$$F(s_0,1,1) = F_{syn}(s_0,1,1) \wedge F_{sem}(s_0,1,1)$$

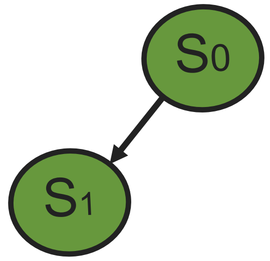
$$F_{syn}(s_0,1,1) = R_{pred} \wedge R_{succ} \wedge R_{in} \wedge R_{out} \wedge R_{new}$$

State	Mapping
S_0	$M(S_0)=\{\}$
$S_?$	$M(S_?)=\{(1,1)\}$

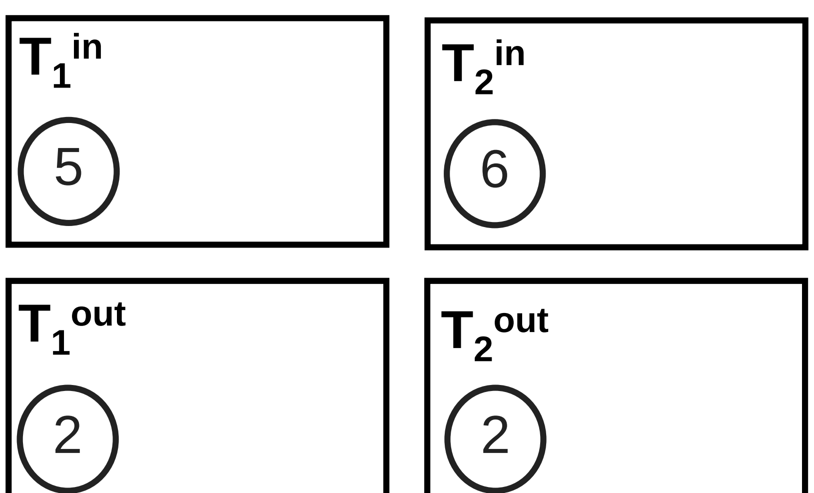
T_1^{in}	T_2^{in}
T_1^{out}	T_2^{out}

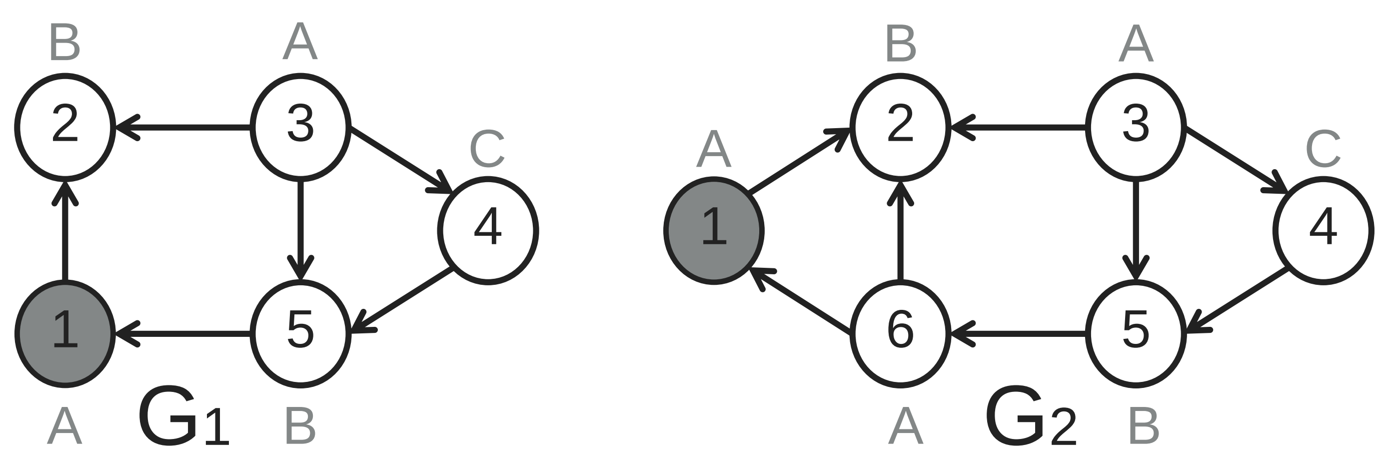


VF2: EXAMPLE

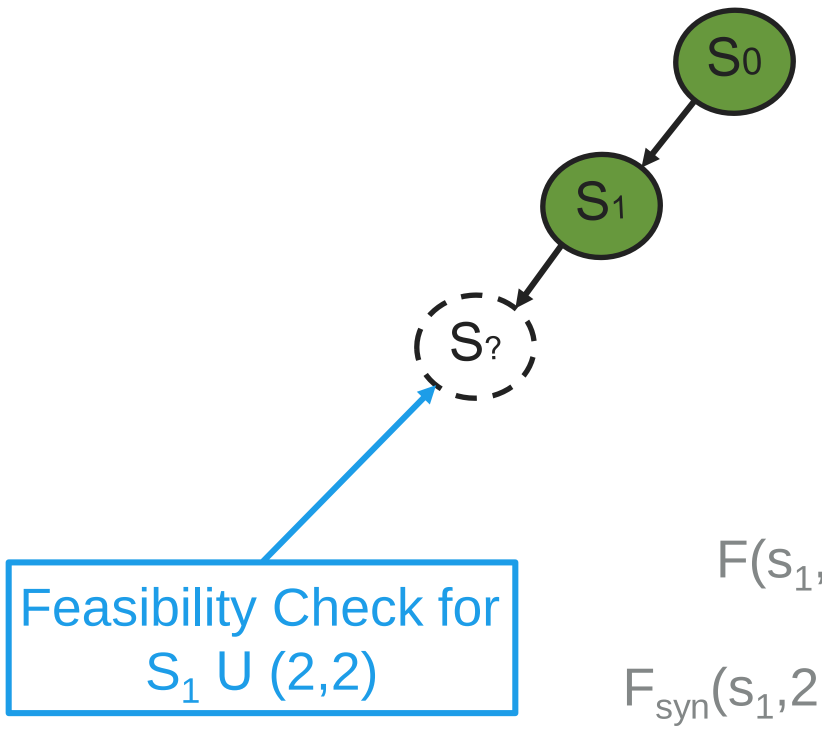


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(1,1)\}$





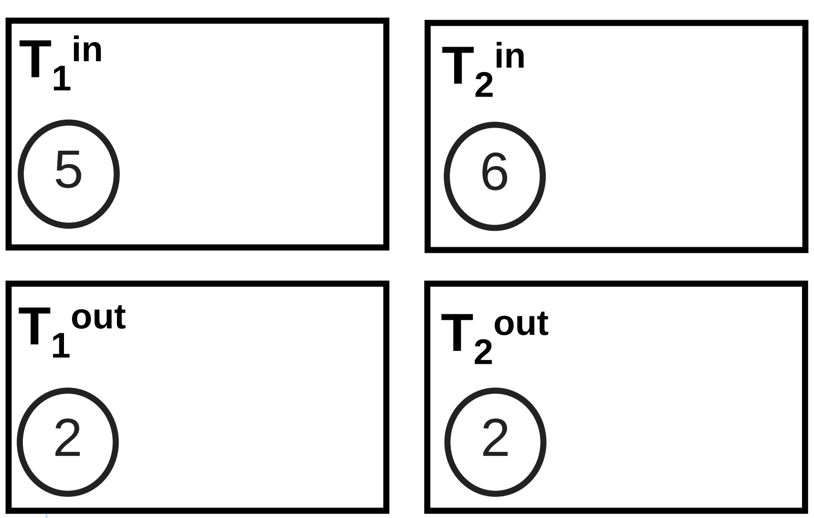
VF2: EXAMPLE

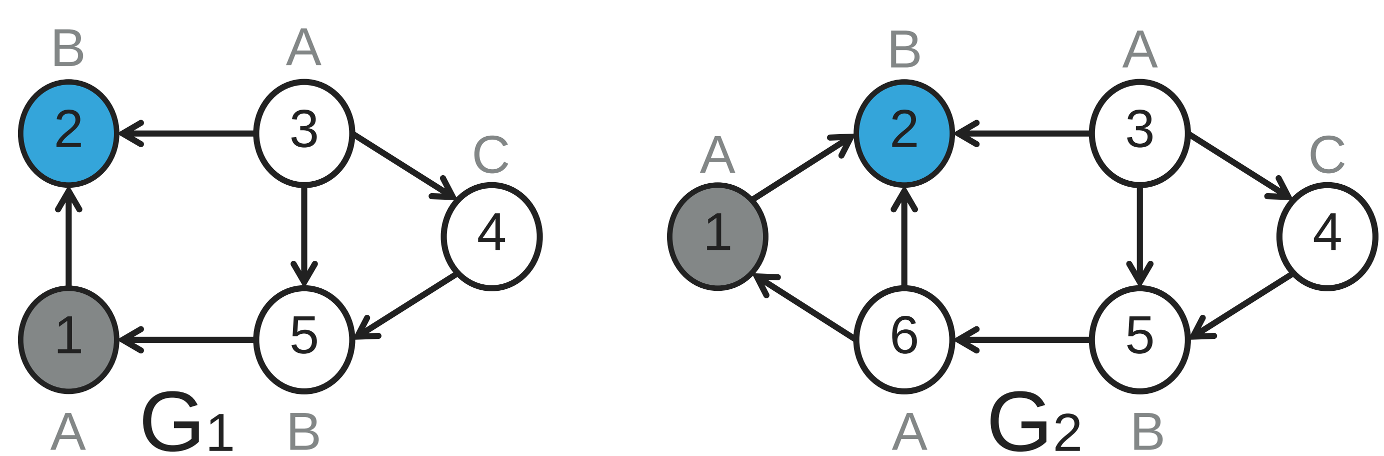


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(1,1)\}$
$S_?$	$M(S_?)=\{(1,1),(2,2)\}$

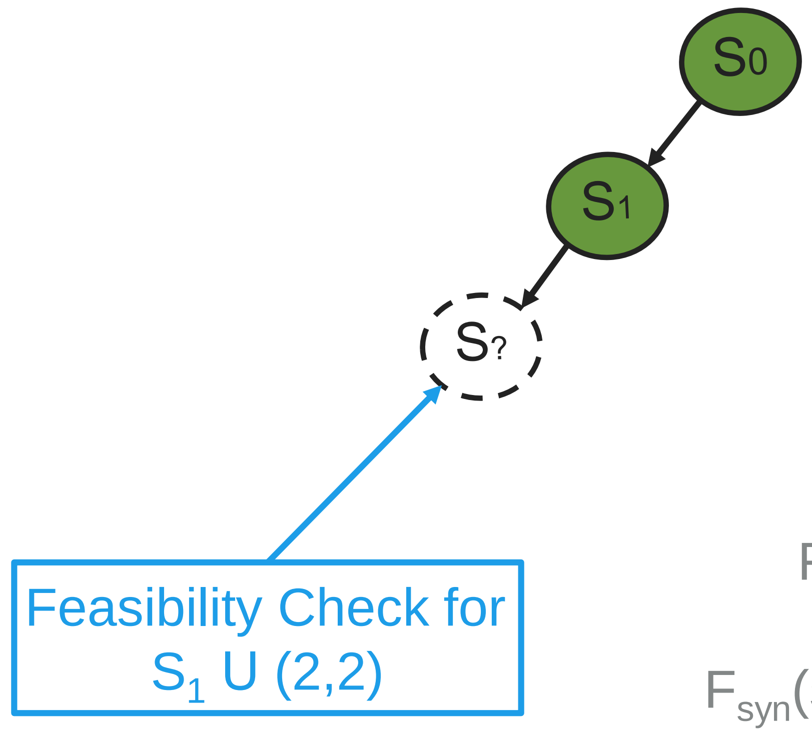
$$F(s_1,2,2) = F_{syn}(s_1,2,2) \wedge F_{sem}(s_1,2,2)$$

$$F_{syn}(s_1,2,2) = R_{pred} \wedge R_{succ} \wedge R_{in} \wedge R_{out} \wedge R_{new}$$





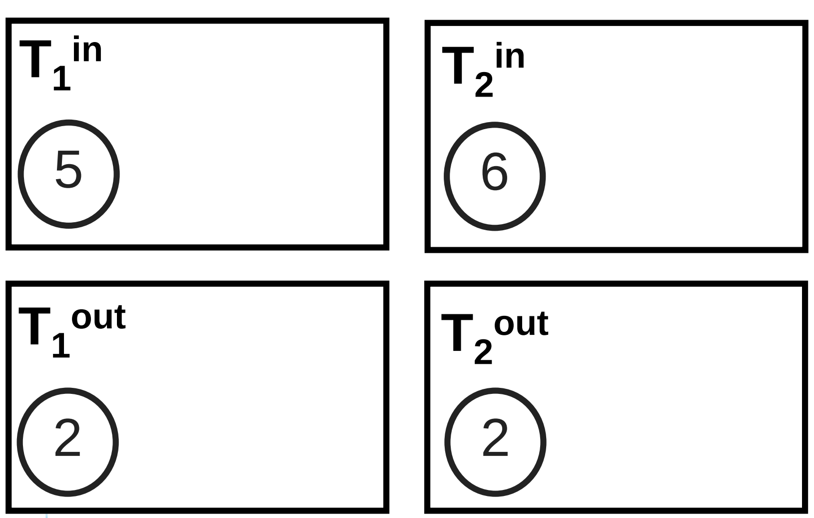
VF2: EXAMPLE

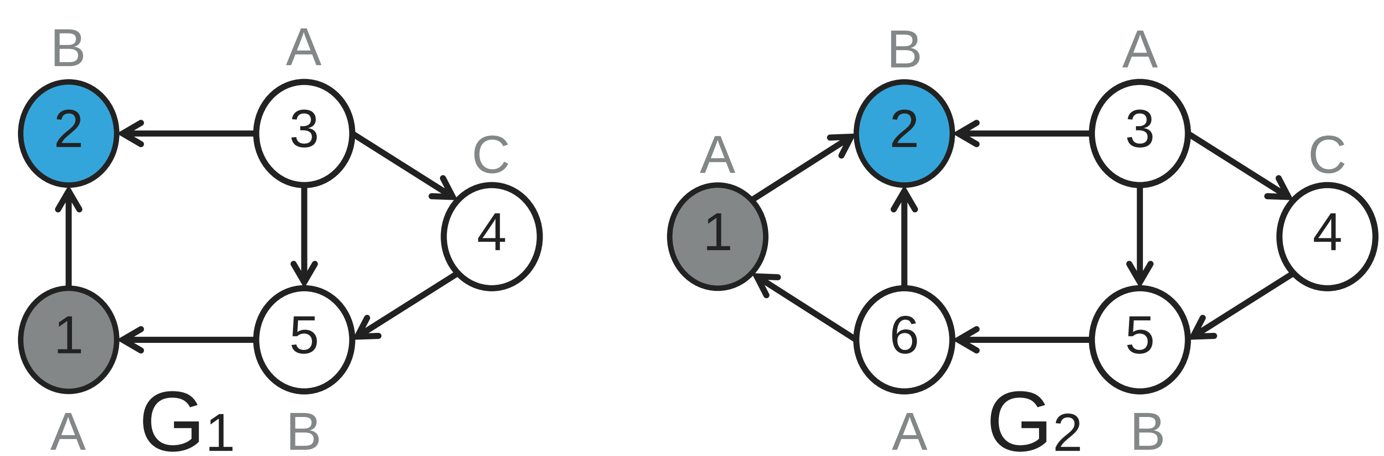


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(1,1)\}$
$S_?$	$M(S_?)=\{(1,1),(2,2)\}$

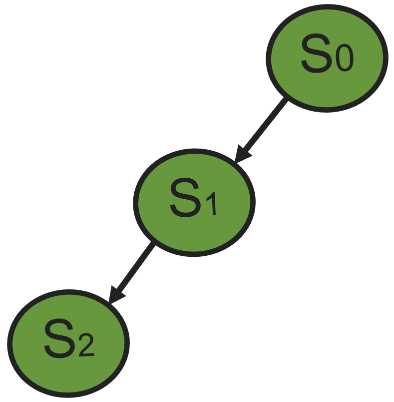
$$F(s_1,2,2) = F_{syn}(s_1,2,2) \wedge F_{sem}(s_1,2,2)$$

$$F_{syn}(s_1,2,2) = R_{pred} \wedge R_{succ} \wedge R_{in} \wedge R_{out} \wedge R_{new}$$

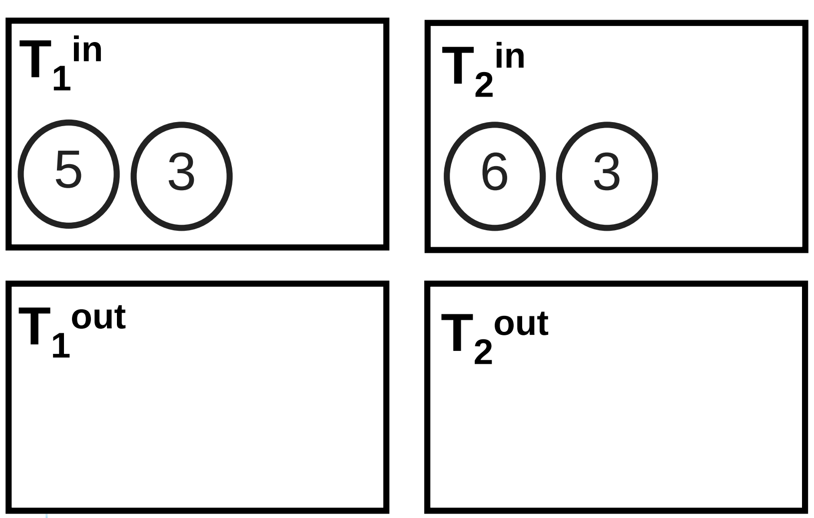


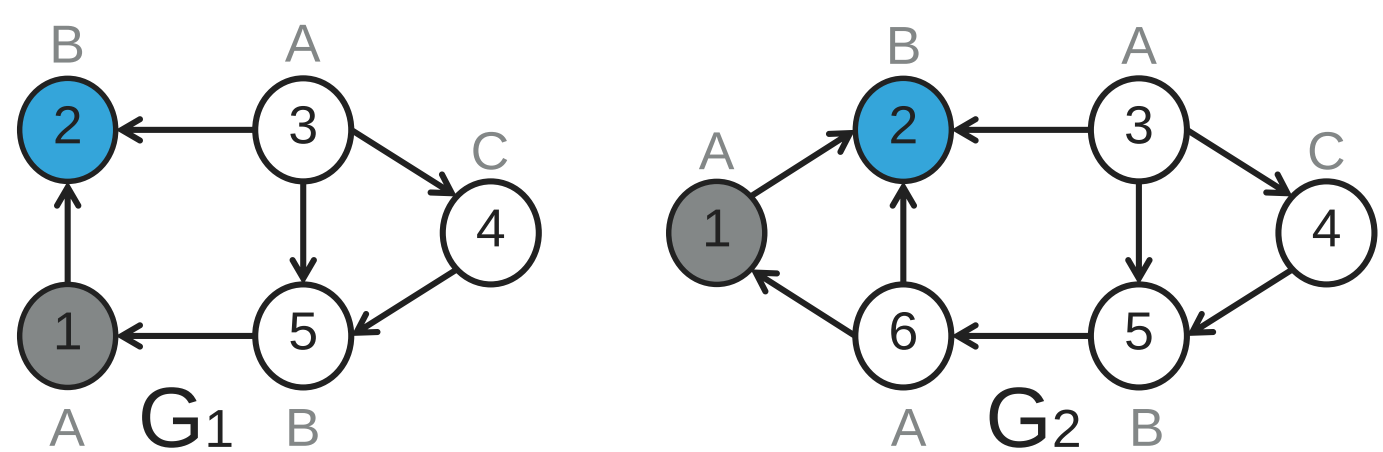


VF2: EXAMPLE

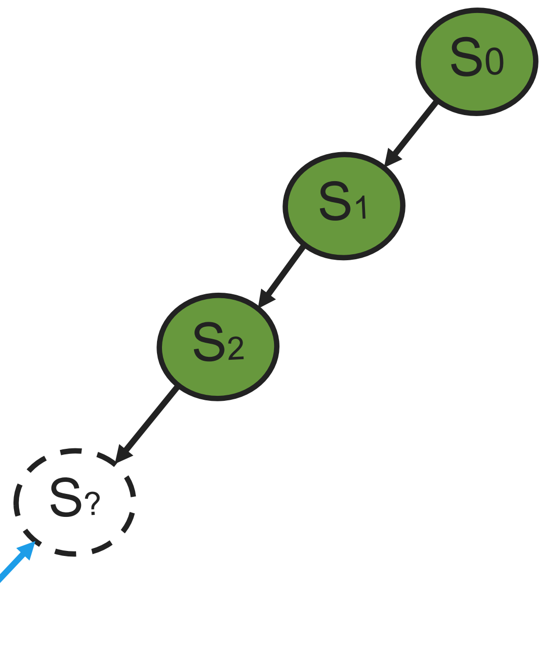


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(1,1)\}$
S_2	$M(S_2)=\{(1,1),(2,2)\}$





VF2: EXAMPLE

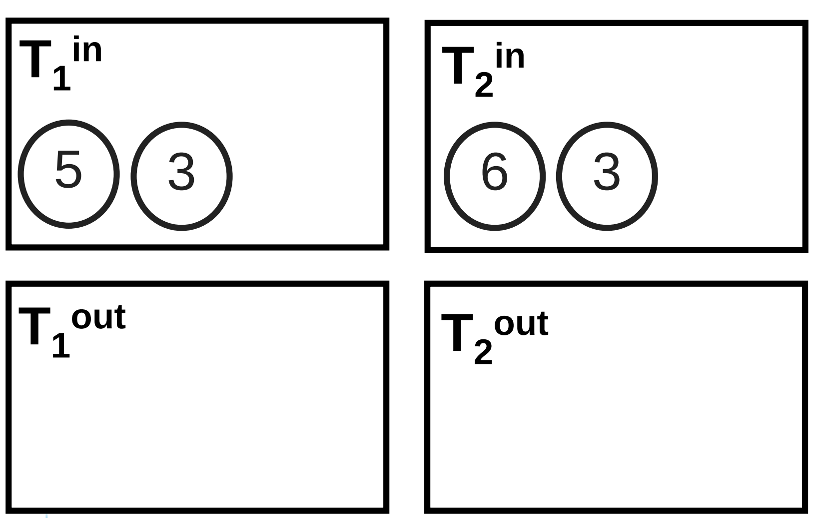


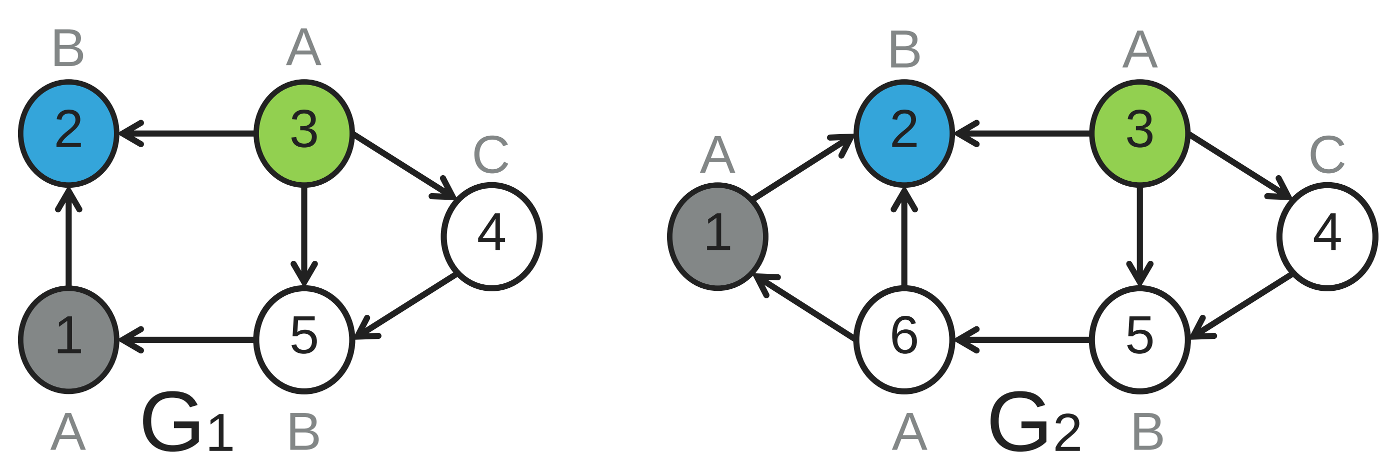
Feasibility Check for S_2 U (3,3)

State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(1,1)\}$
S_2	$M(S_2)=\{(1,1),(2,2)\}$
$S_?$	$M(S_?)=\{(1,1),(2,2),(3,3)\}$

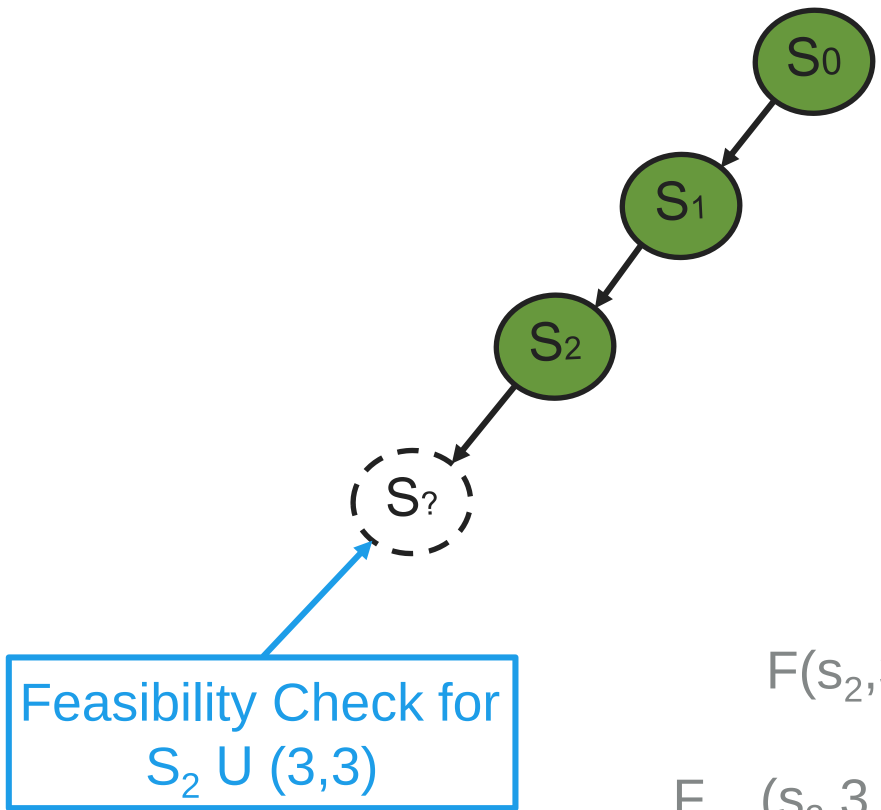
$$F(s_2,3,3) = F_{syn}(s_2,3,3) \wedge F_{sem}(s_2,3,3)$$

$$F_{syn}(s_2,3,3) = R_{pred} \wedge R_{succ} \wedge R_{in} \wedge R_{out} \wedge R_{new}$$





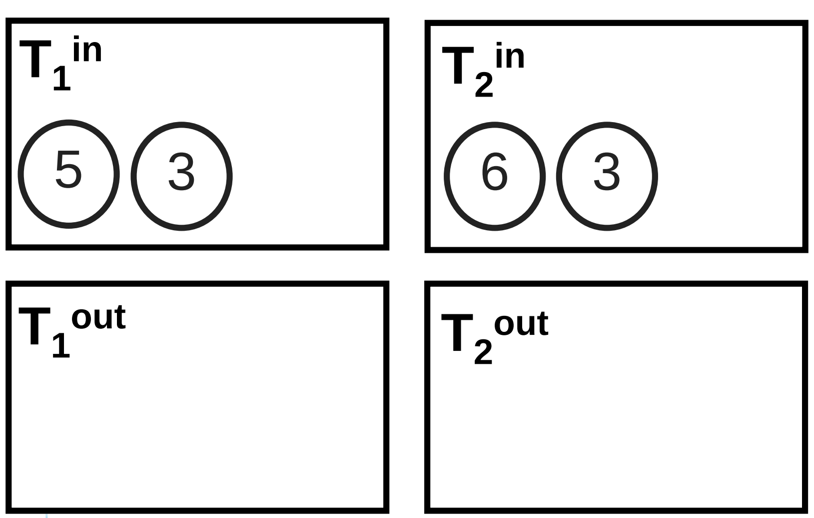
VF2: EXAMPLE

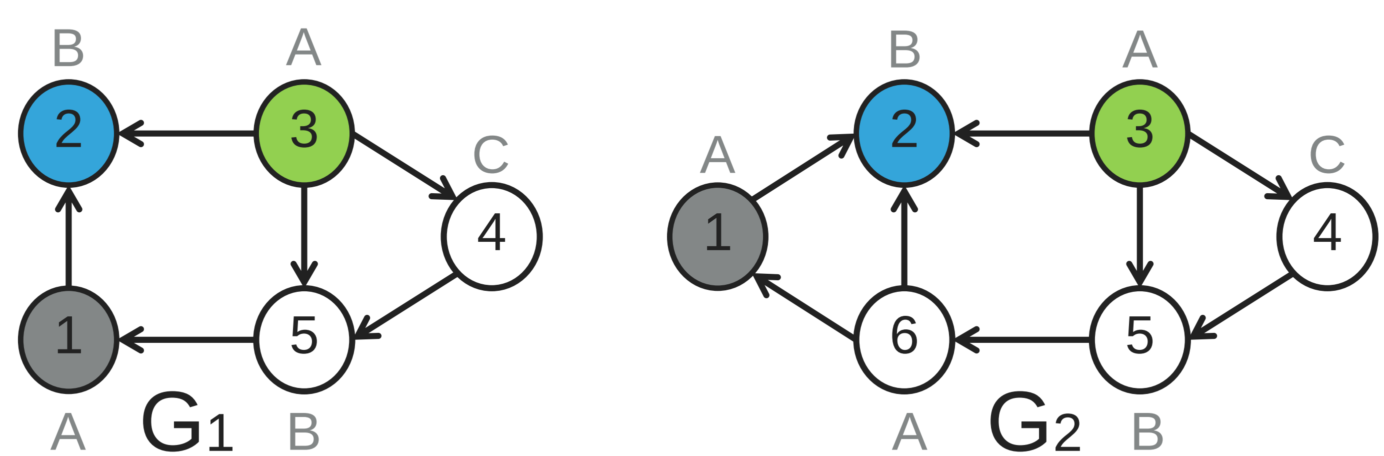


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(1,1)\}$
S_2	$M(S_2)=\{(1,1),(2,2)\}$
$S_?$	$M(S_?)=\{(1,1),(2,2),(3,3)\}$

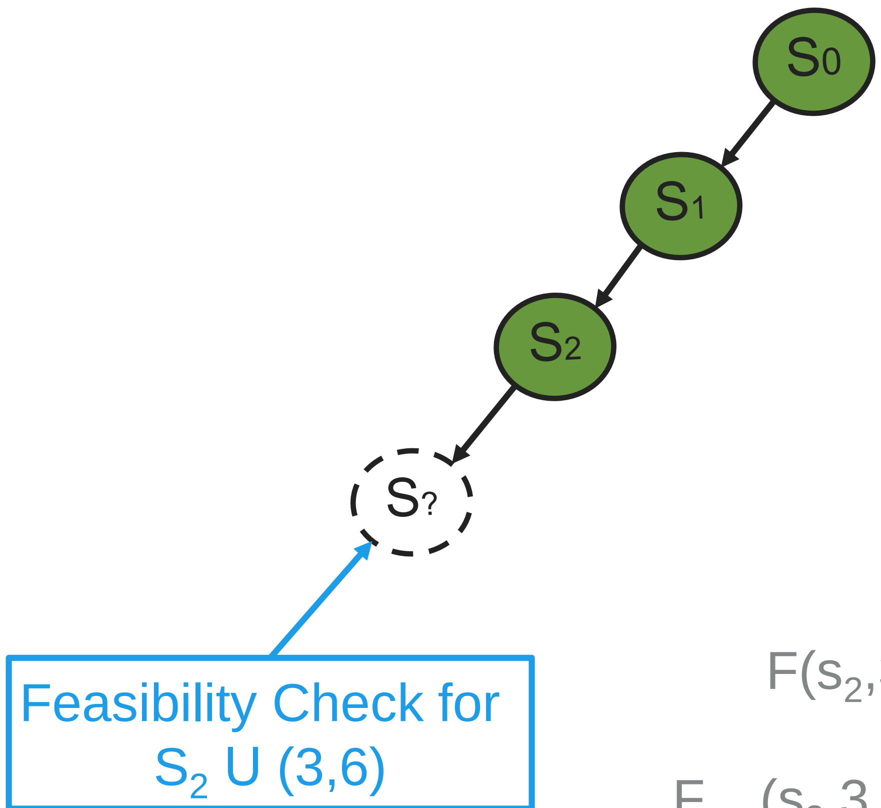
$F(s_2,3,3) = F_{syn}(s_2,3,3) \wedge F_{sem}(s_2,3,3)$

$F_{syn}(s_2,3,3) = R_{pred} \wedge R_{succ} \wedge R_{in} \wedge R_{out} \wedge R_{new}$





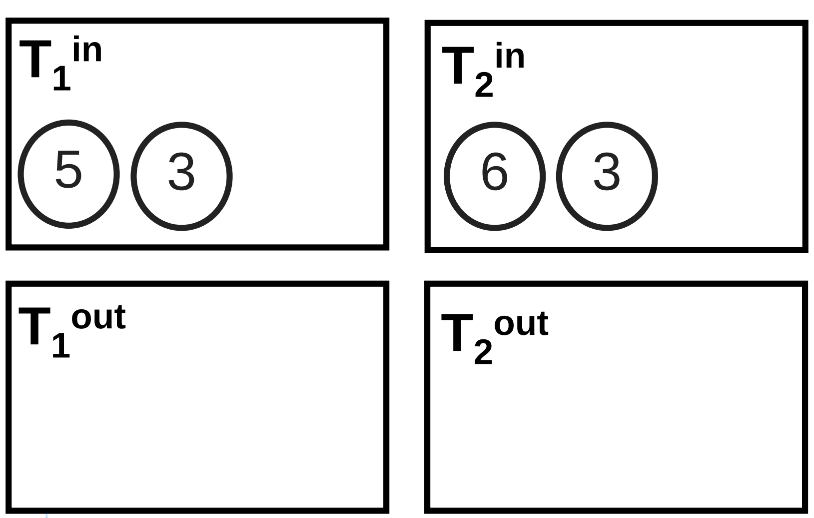
VF2: EXAMPLE

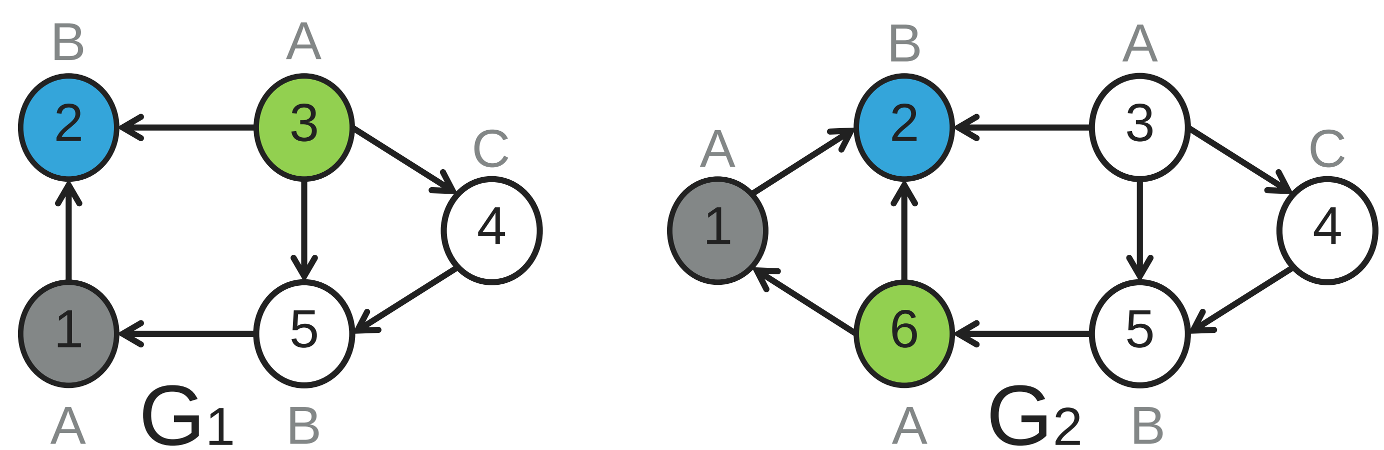


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(1,1)\}$
S_2	$M(S_2)=\{(1,1),(2,2)\}$
$S_?$	$M(S_?)=\{(1,1),(2,2),(3,6)\}$

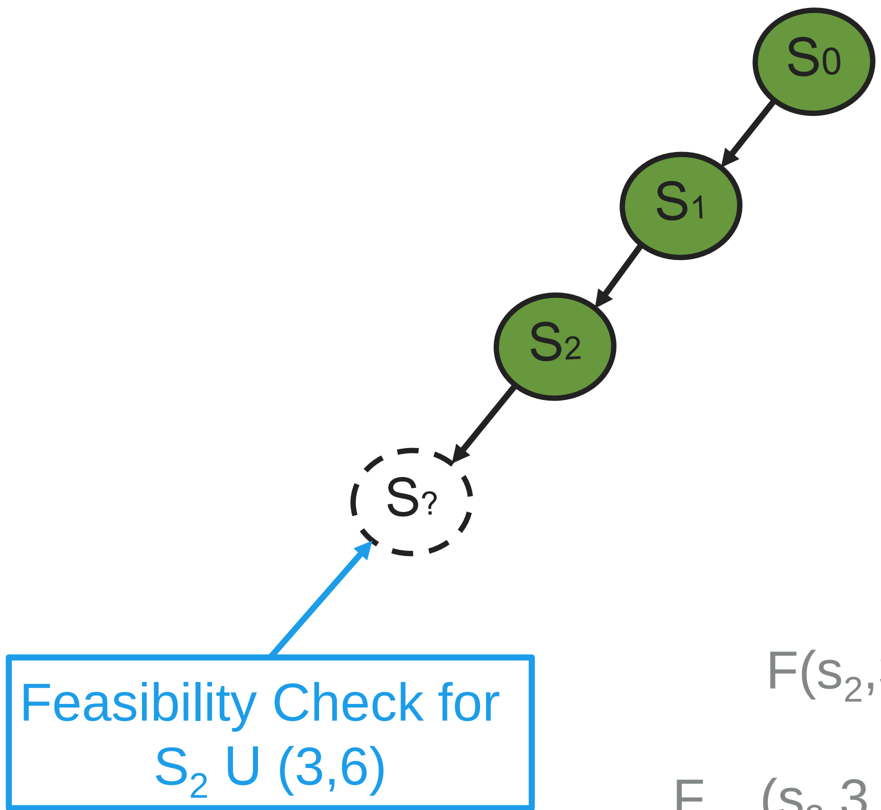
$$F(s_2,3,6) = F_{syn}(s_2,3,6) \wedge F_{sem}(s_2,3,6)$$

$$F_{syn}(s_2,3,6) = R_{pred} \wedge R_{succ} \wedge R_{in} \wedge R_{out} \wedge R_{new}$$





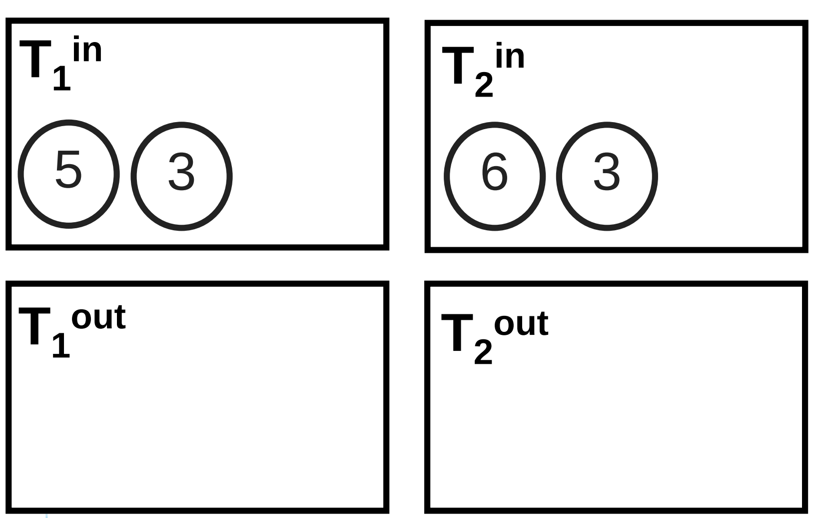
VF2: EXAMPLE

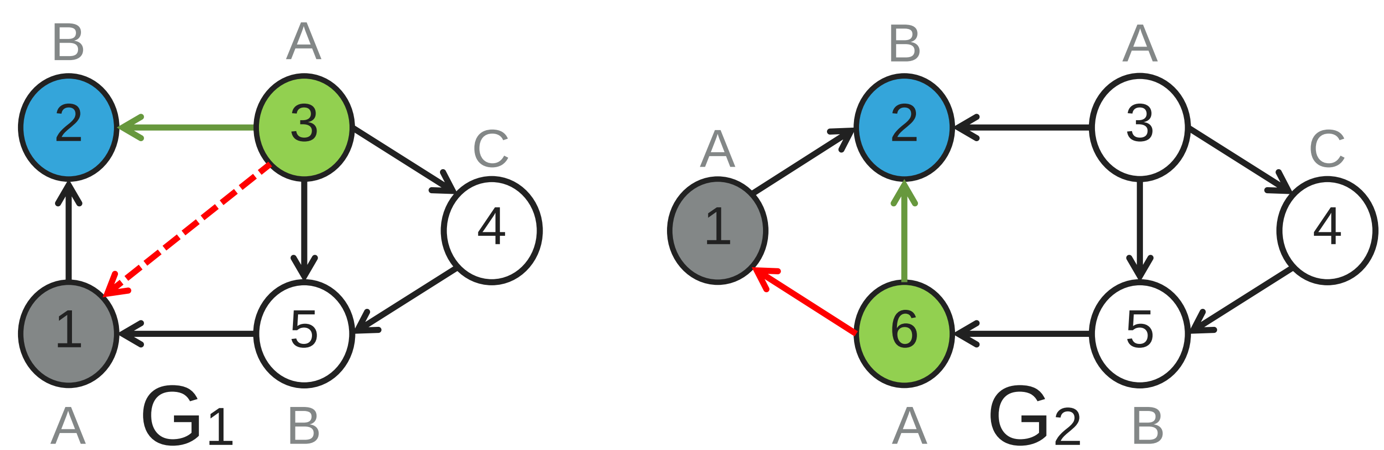


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(1,1)\}$
S_2	$M(S_2)=\{(1,1),(2,2)\}$
$S_?$	$M(S_?)=\{(1,1),(2,2),(3,6)\}$

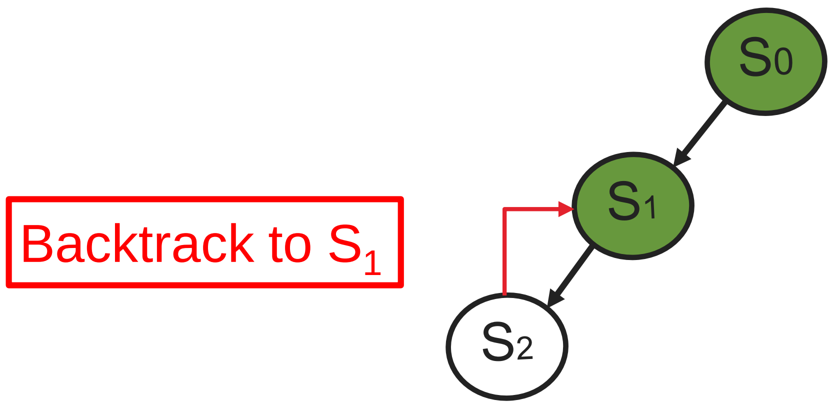
$F(s_2,3,6) = F_{syn}(s_2,3,6) \wedge F_{sem}(s_2,3,6)$

$F_{syn}(s_2,3,6) = R_{pred} \wedge R_{succ} \wedge R_{in} \wedge R_{out} \wedge R_{new}$

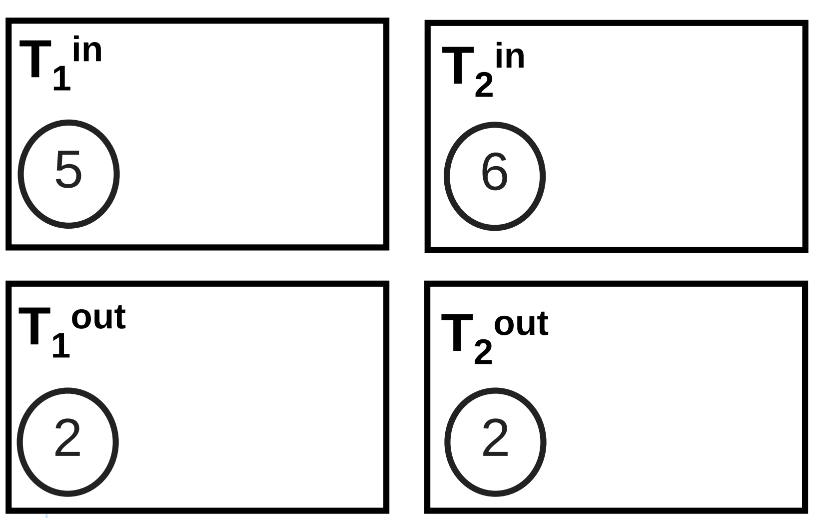


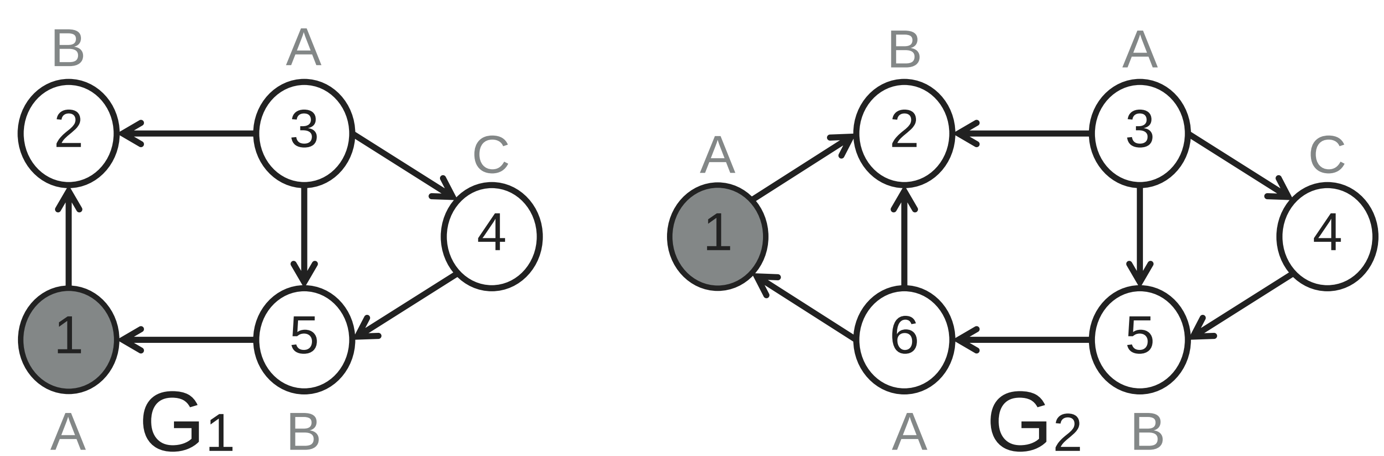


VF2: EXAMPLE

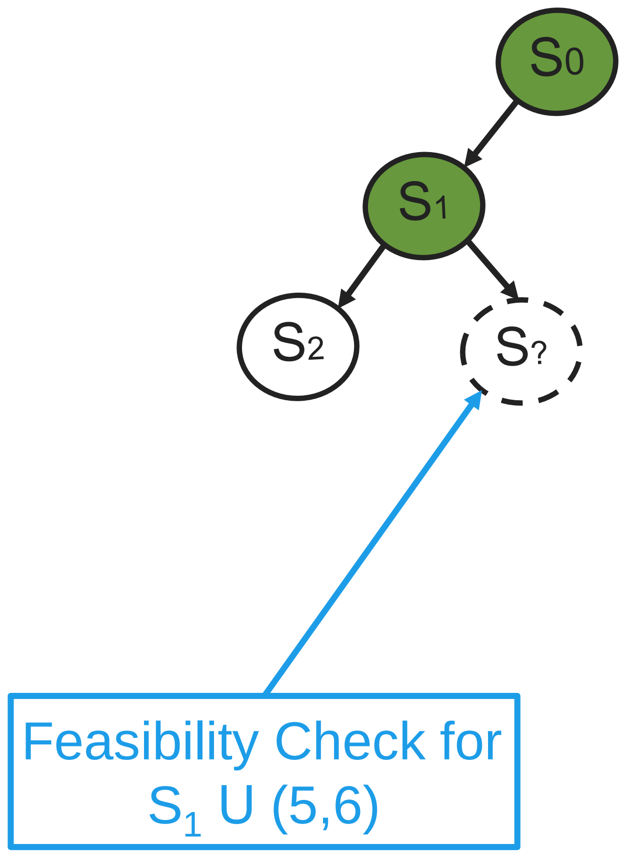


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(1,1)\}$
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~





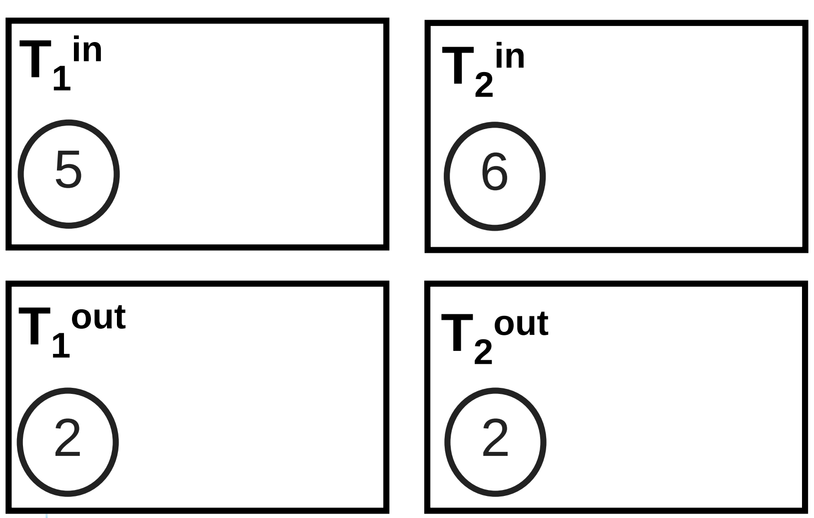
VF2: EXAMPLE

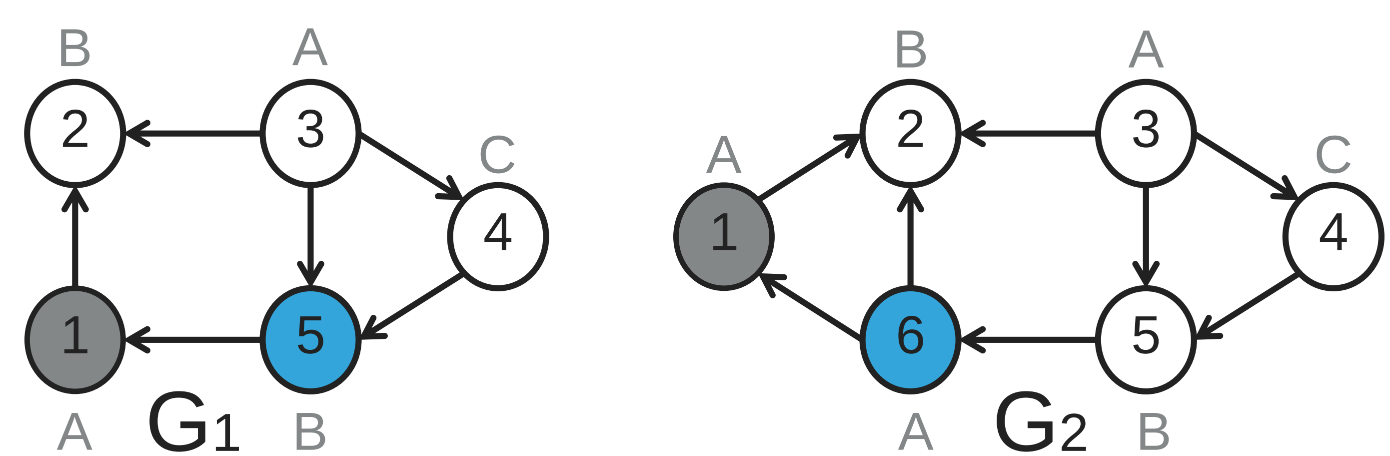


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(1,1)\}$
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
$S_?$	$M(S_?)=\{(1,1),(5,6)\}$

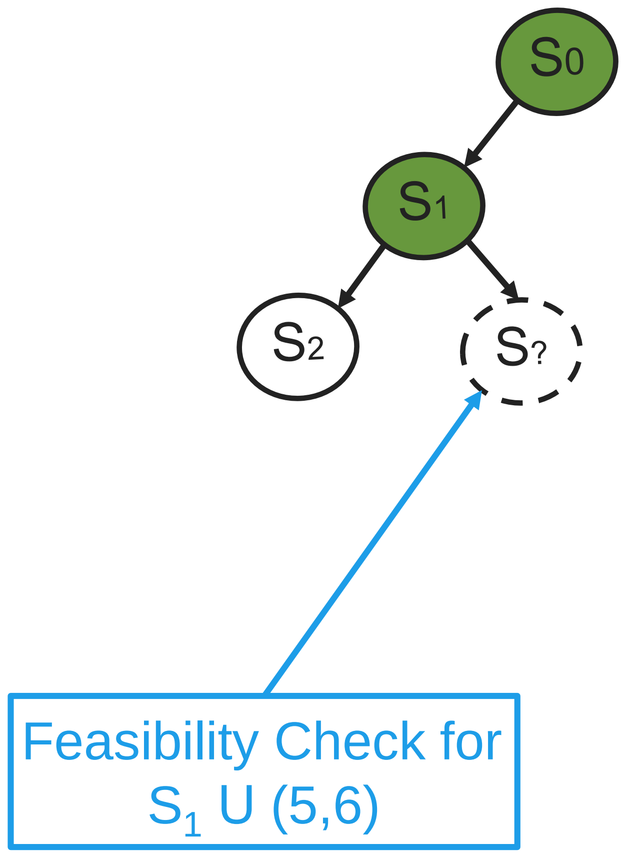
$$F(s_1,5,6) = F_{syn}(s_1,5,6) \wedge F_{sem}(s_1,5,6)$$

$$F_{syn}(s_1,5,6) = R_{pred} \wedge R_{succ} \wedge R_{in} \wedge R_{out} \wedge R_{new}$$





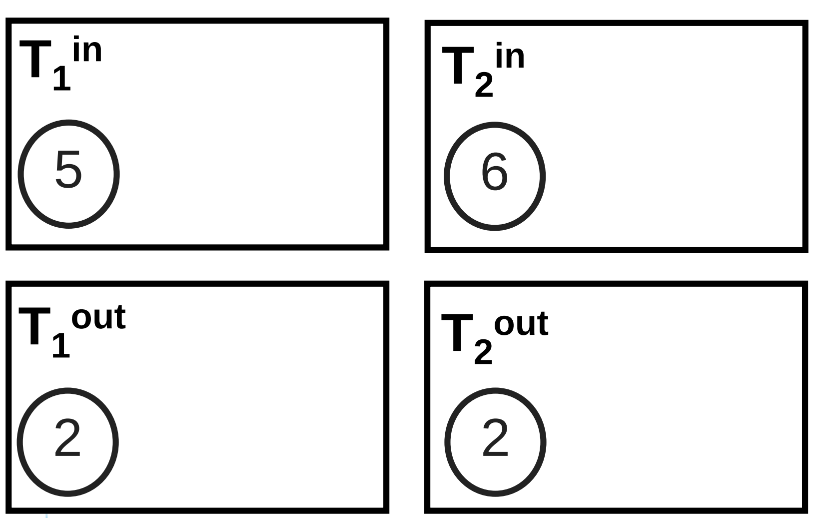
VF2: EXAMPLE

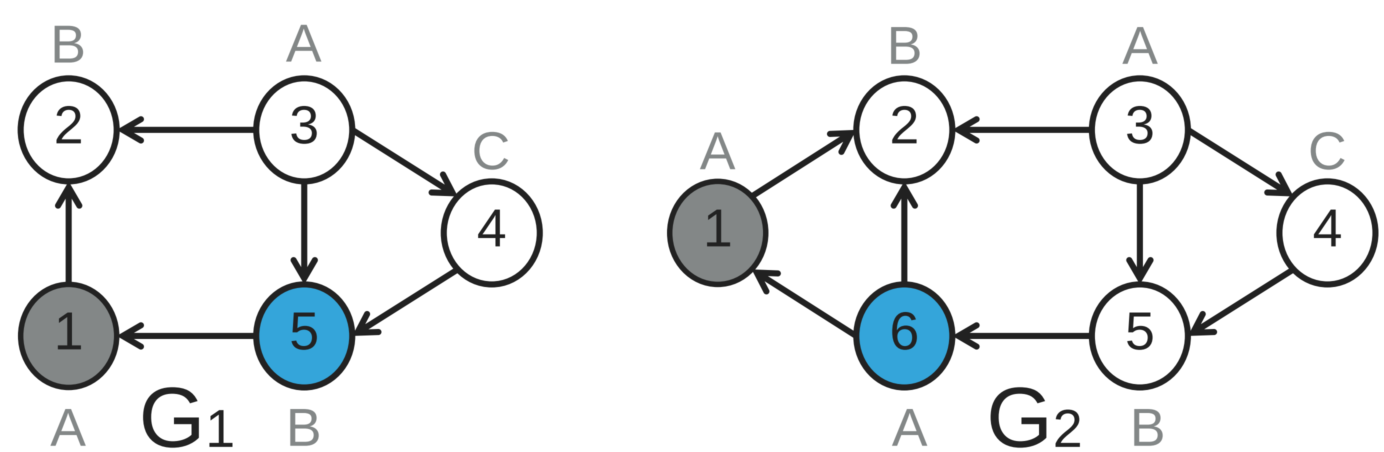


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(1,1)\}$
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
$S_?$	$M(S_?)=\{(1,1),(5,6)\}$

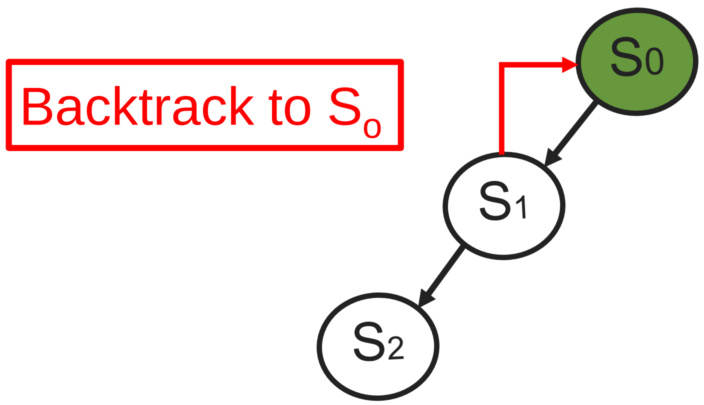
$$F(s_1,5,6) = F_{syn}(s_1,5,6) \wedge F_{sem}(s_1,5,6)$$

$$F_{syn}(s_1,5,6) = R_{pred} \wedge R_{succ} \wedge R_{in} \wedge R_{out} \wedge R_{new}$$

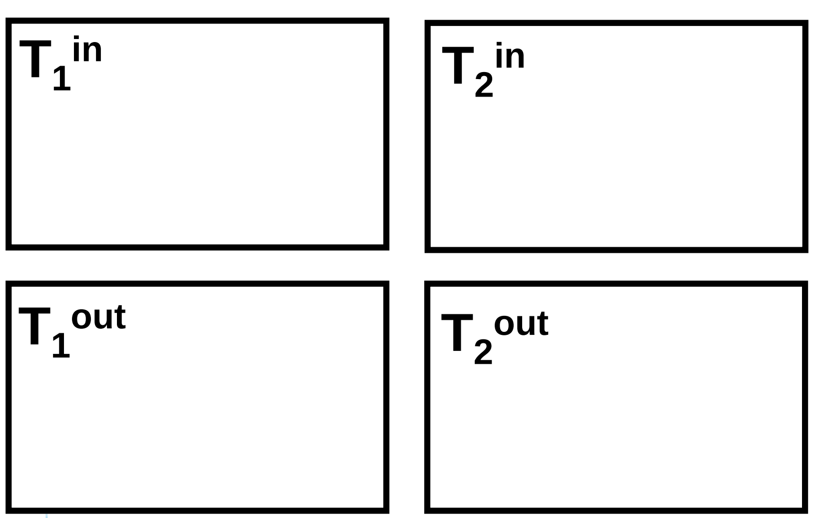


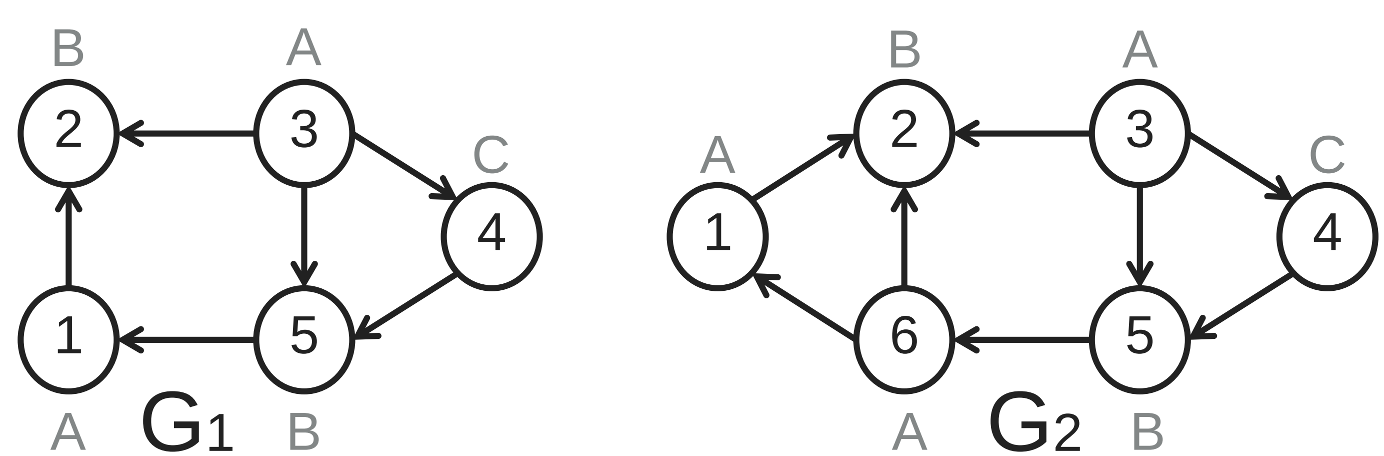


VF2: EXAMPLE

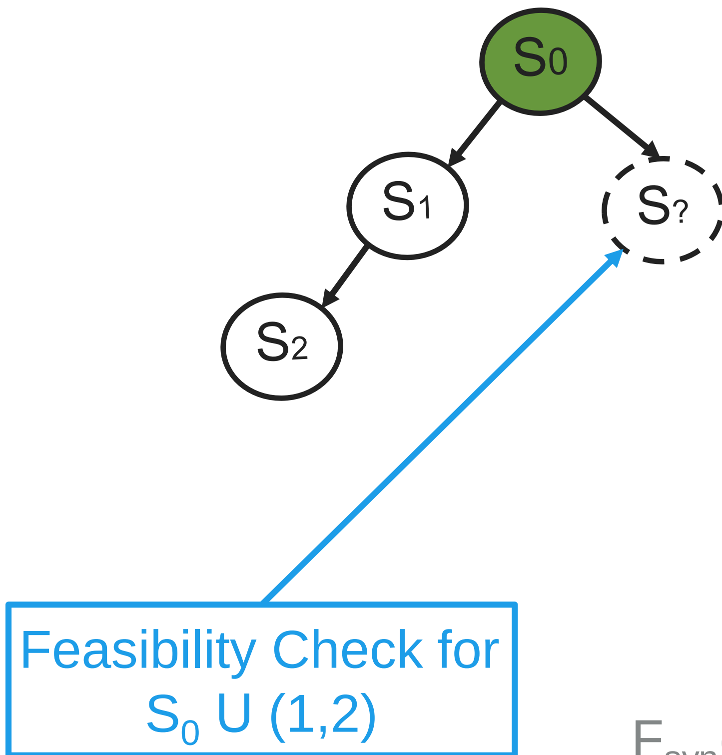


State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~





VF2: EXAMPLE



State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
$S_?$	$M(S_?)=\{(1,2)\}$

$F(s_0,1,2) = F_{syn}(s_0,1,2) \wedge F_{sem}(s_0,1,2)$

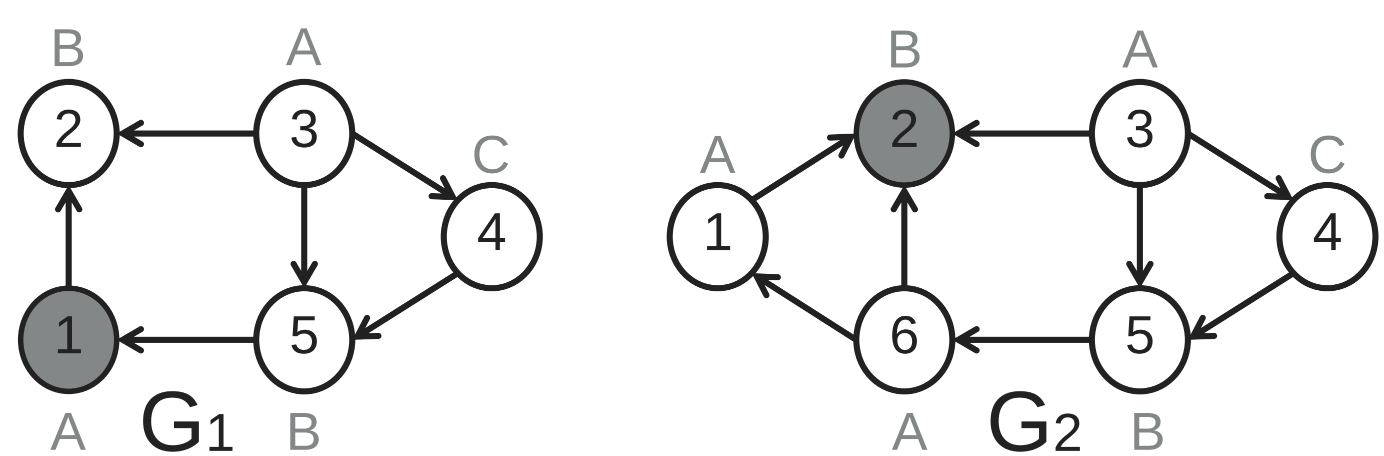
$F_{syn}(s_0,1,2) = R_{pred} \wedge R_{succ} \wedge R_{in} \wedge R_{out} \wedge R_{new}$

T_1^{in}

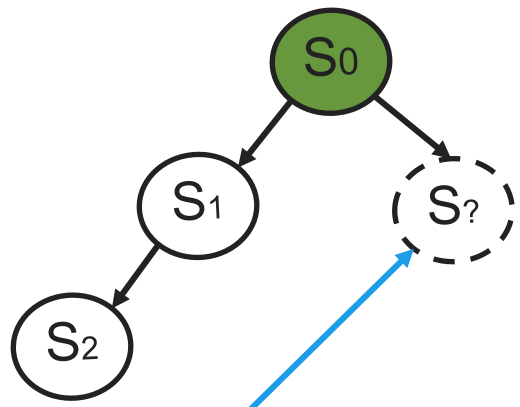
T_2^{in}

T_1^{out}

T_2^{out}



VF2: EXAMPLE



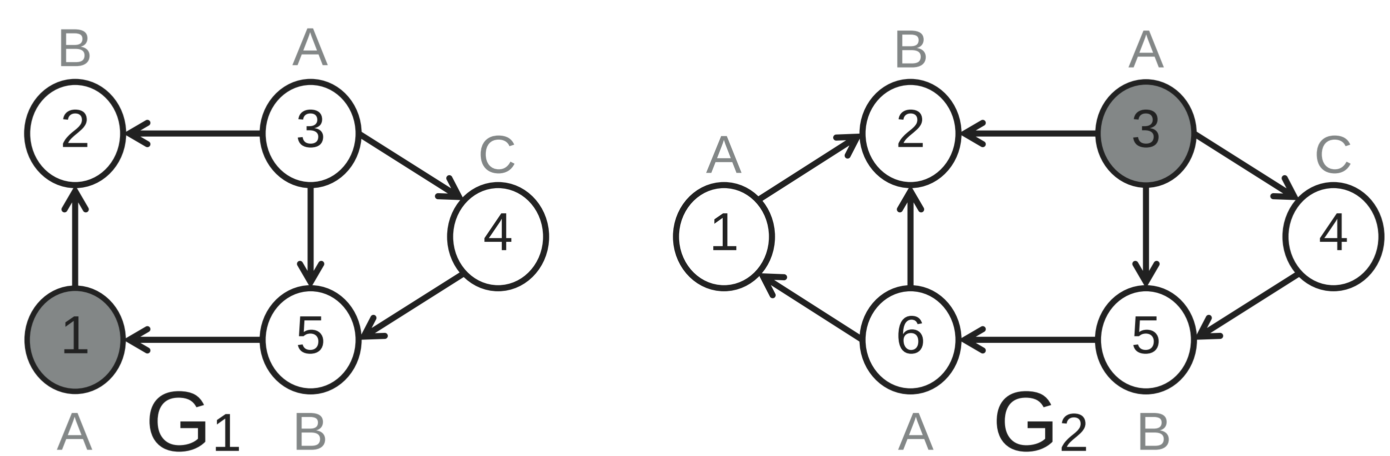
Feasibility Check for $S_0 \cup (1,3)$

State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
$S_?$	$M(S_?)=\{(1,3)\}$

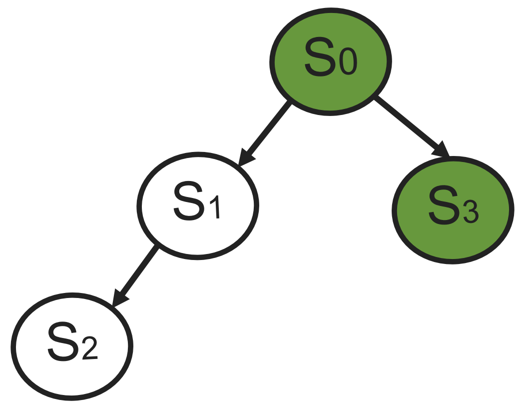
$F(s_0,1,3) = F_{syn}(s_0,1,3) \wedge F_{sem}(s_0,1,3)$ ✔ ✔

$F_{syn}(s_0,1,3) = F_{pred} \wedge F_{succ} \wedge F_{in} \wedge F_{out} \wedge F_{new}$ ✔ ✔ ✔ ✔ ✔

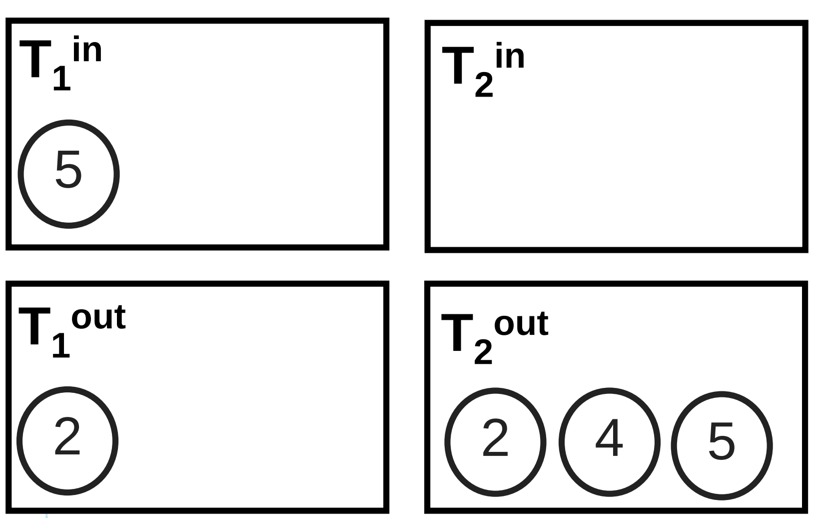
T_1^{in}	T_2^{in}
T_1^{out}	T_2^{out}

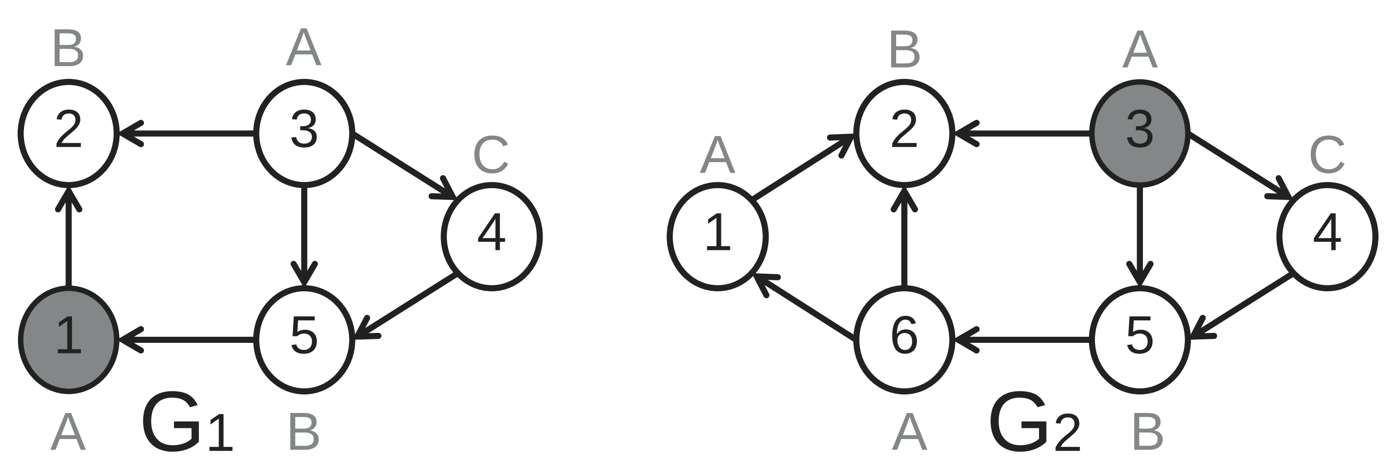


VF2: EXAMPLE

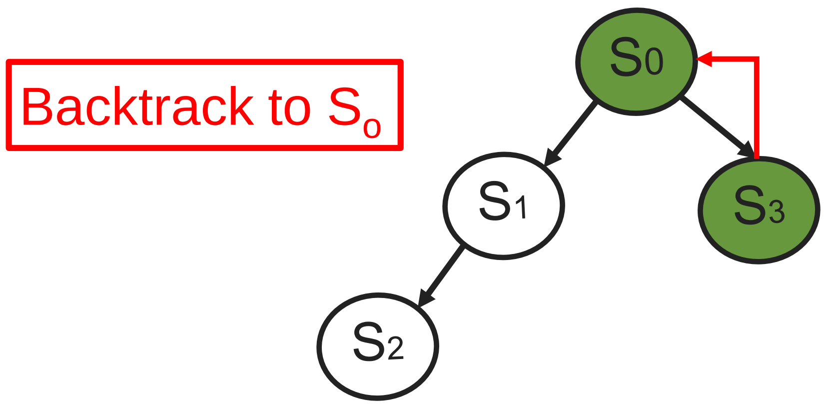


State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
S_3	$M(S_3)=\{(1,3)\}$

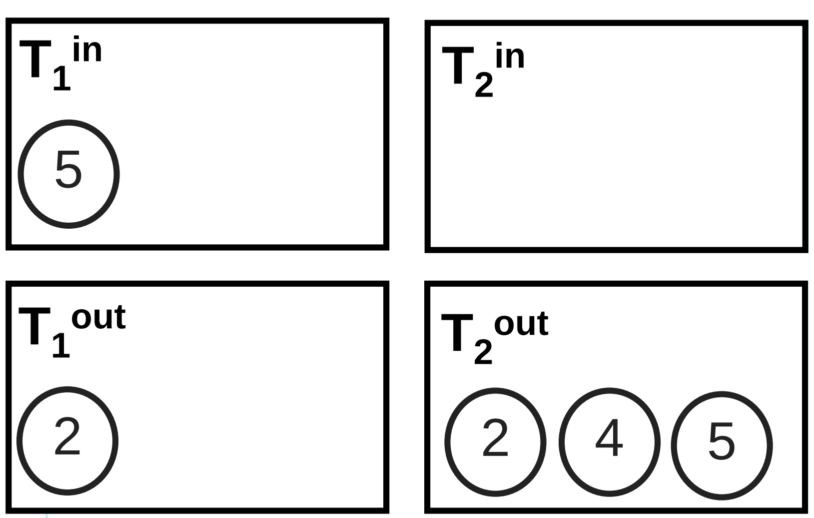


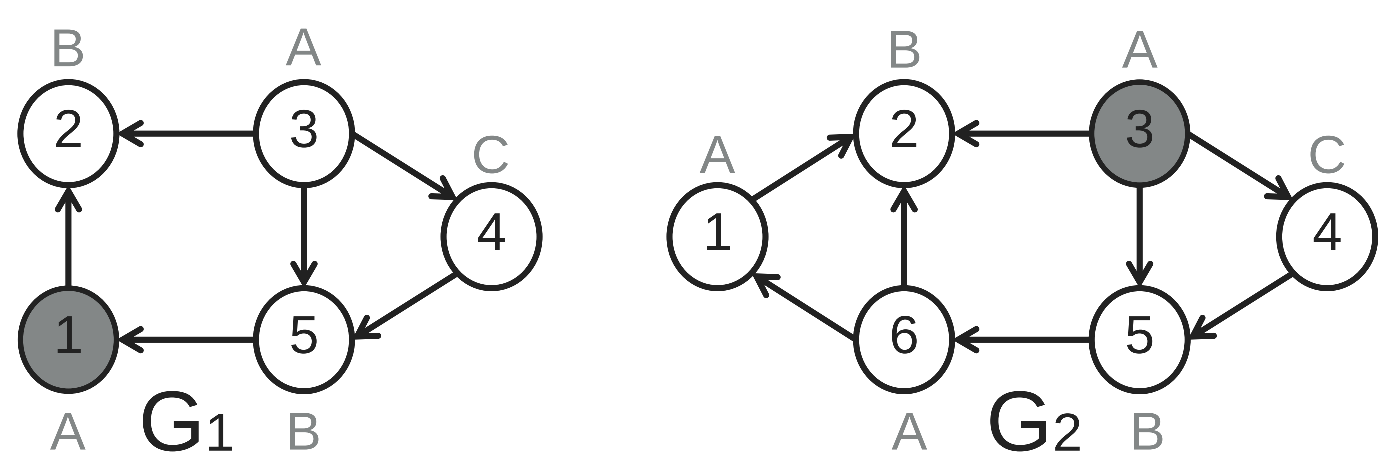


VF2: EXAMPLE

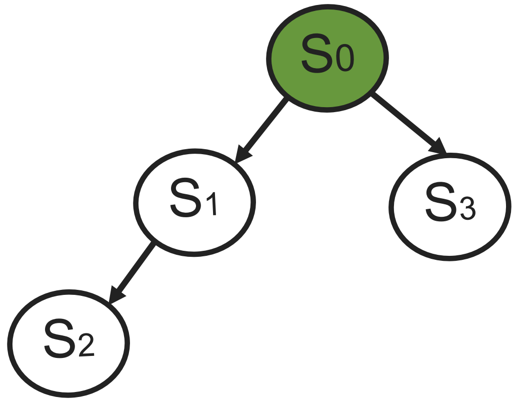


State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
~~S_3~~	~~$M(S_3)=\{(1,3)\}$~~



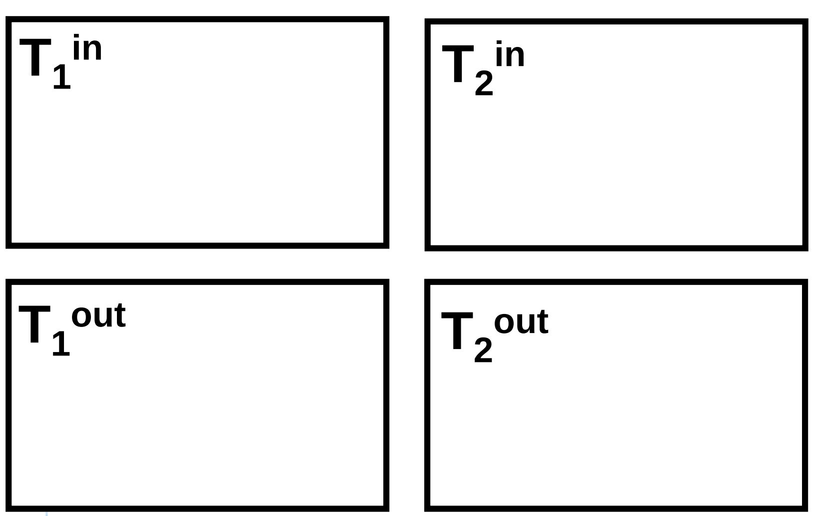


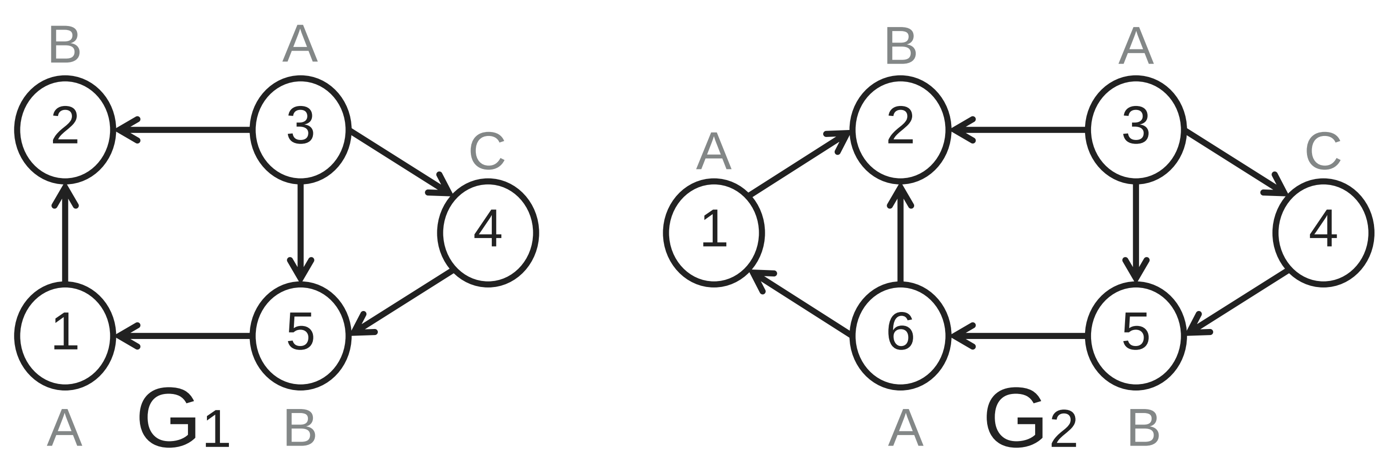
VF2: EXAMPLE



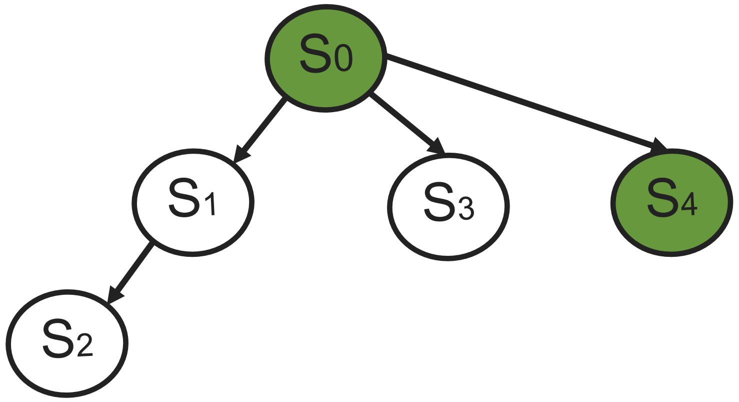
State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
~~S_3~~	~~$M(S_3)=\{(1,3)\}$~~

- The feasibility check fails for the couples (1,4) and (1,5) due to F_{sem}

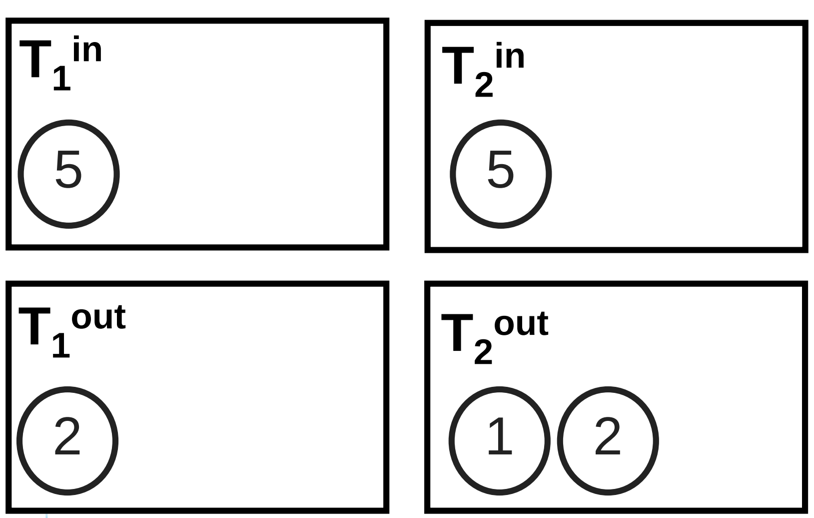


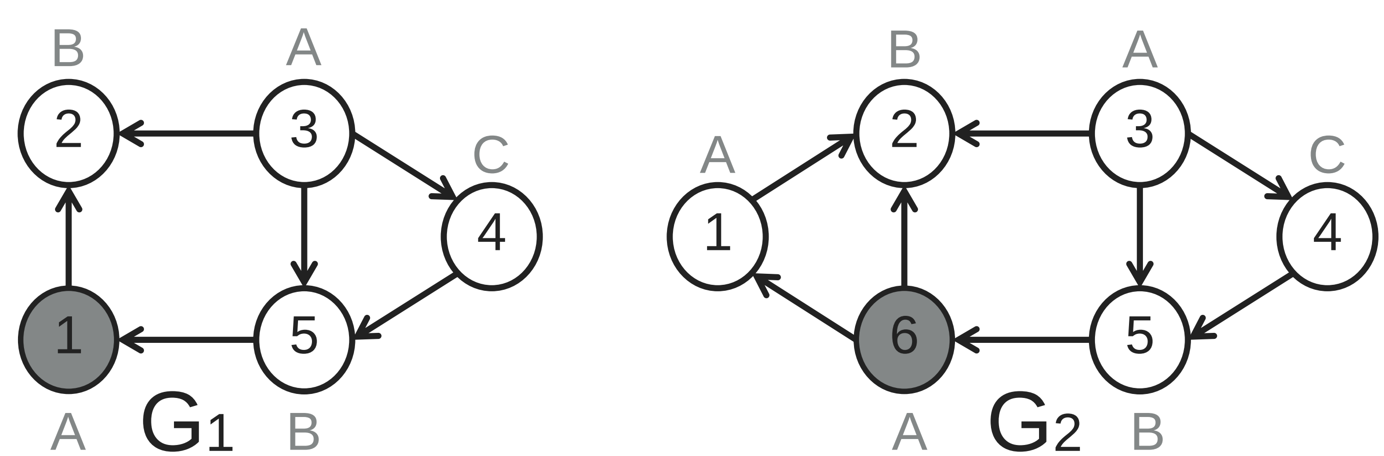


VF2: EXAMPLE

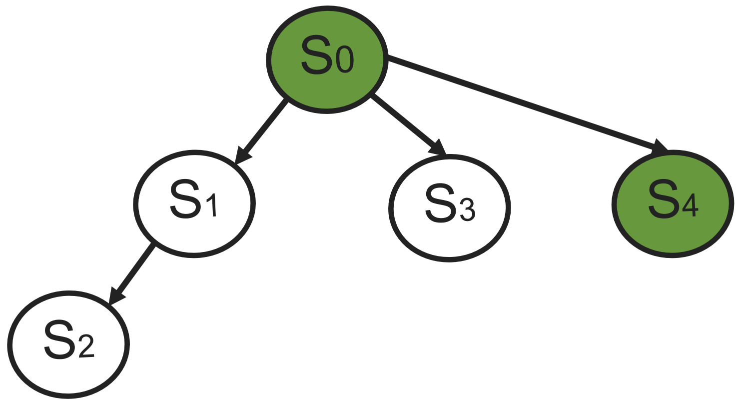


State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
~~S_3~~	~~$M(S_3)=\{(1,3)\}$~~
S_4	$M(S_4)=\{(1,4)\}$



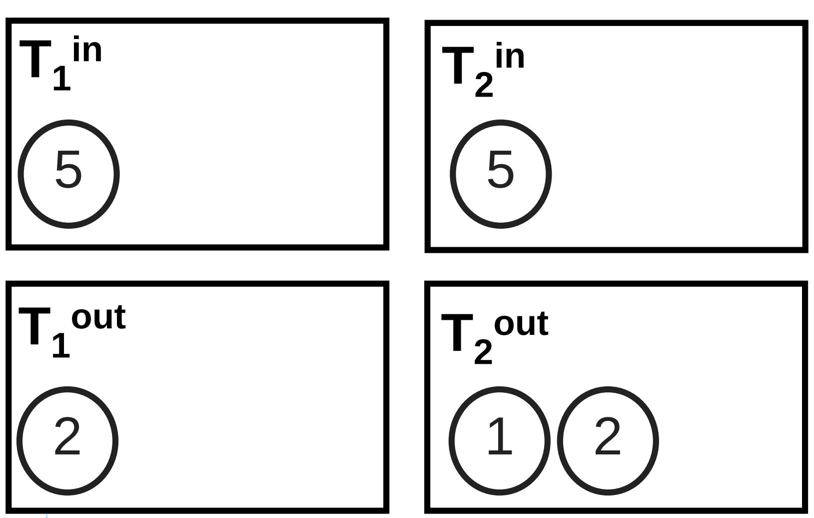


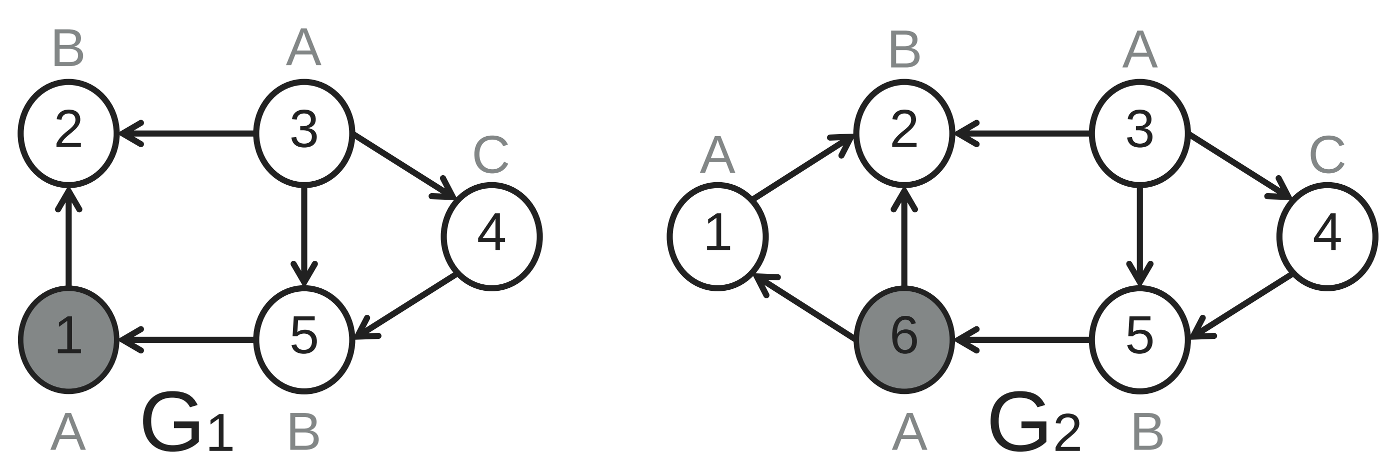
VF2: EXAMPLE



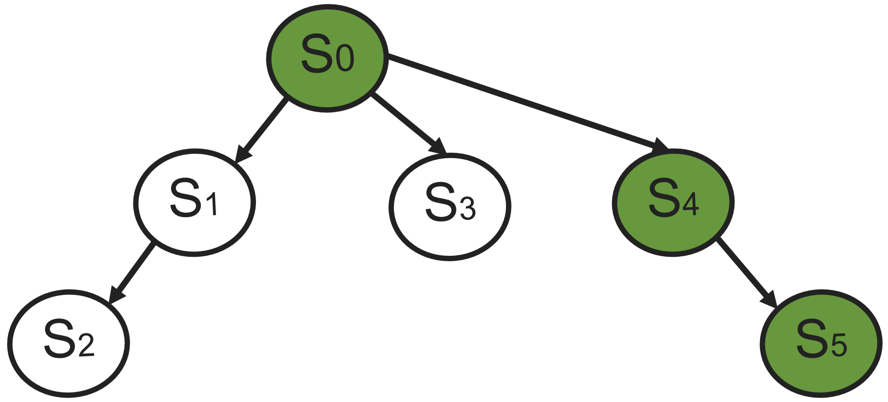
State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
~~S_3~~	~~$M(S_3)=\{(1,3)\}$~~
S_4	$M(S_4)=\{(1,4)\}$

- The feasibility check fails for the couple (2,1) due to F_{sem}

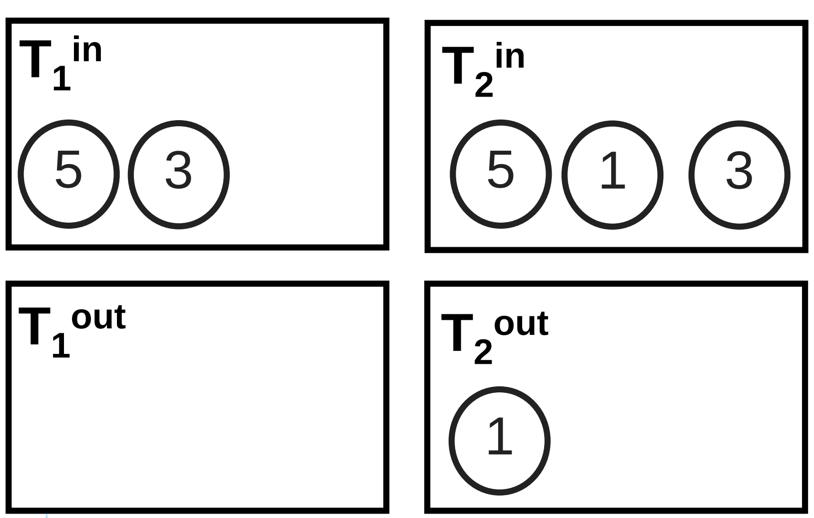


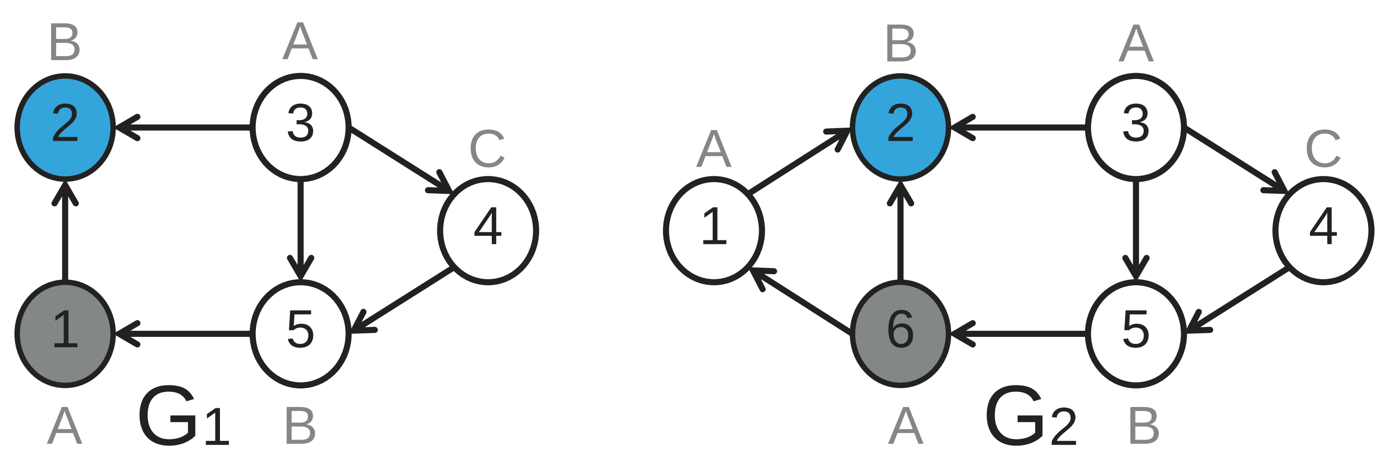


VF2: EXAMPLE

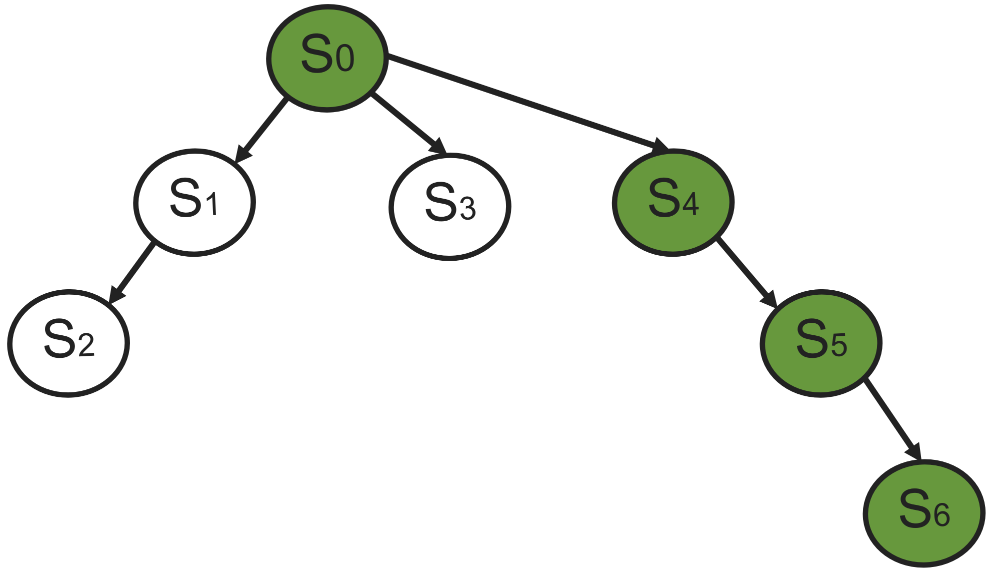


State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
~~S_3~~	~~$M(S_3)=\{(1,3)\}$~~
S_4	$M(S_4)=\{(1,4)\}$
S_5	$M(S_5)=\{(1,4), (2,2)\}$

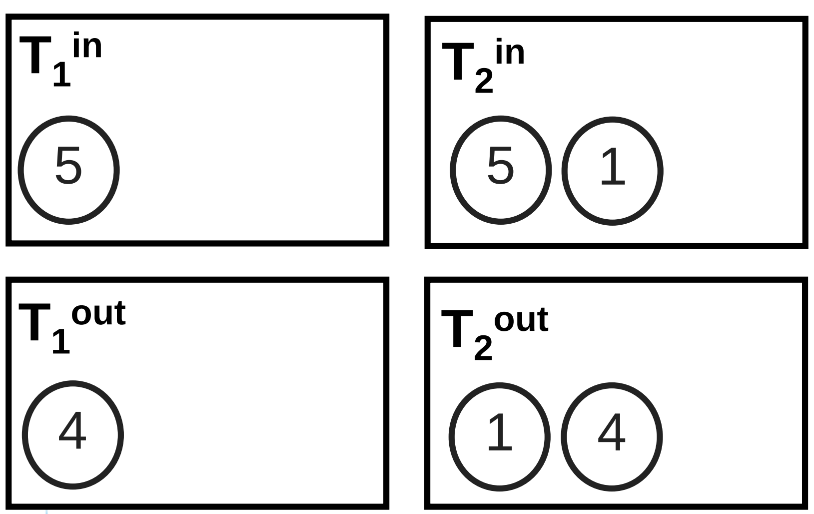


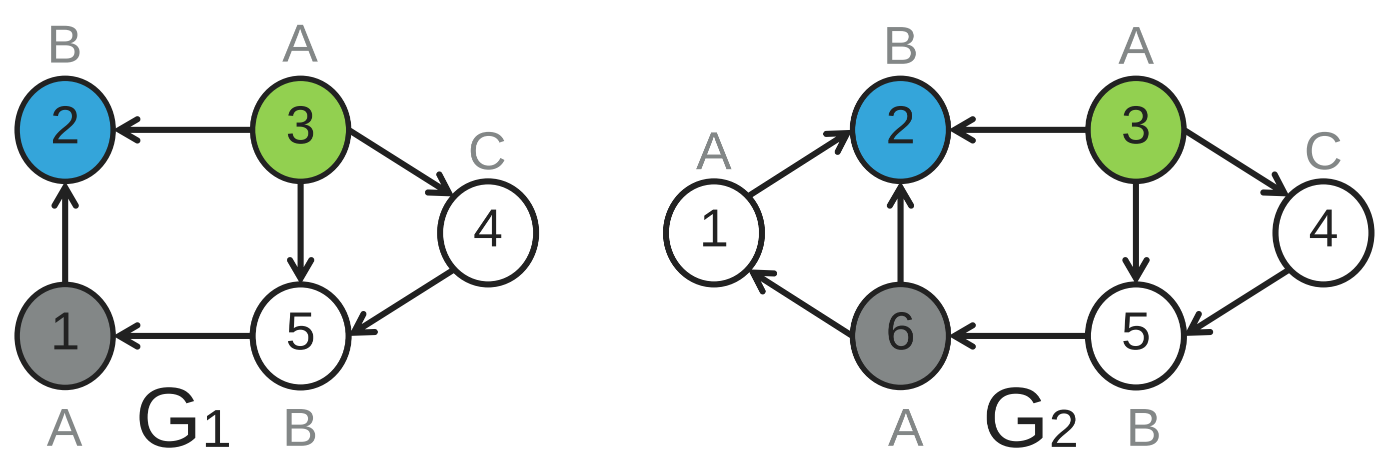


VF2: EXAMPLE

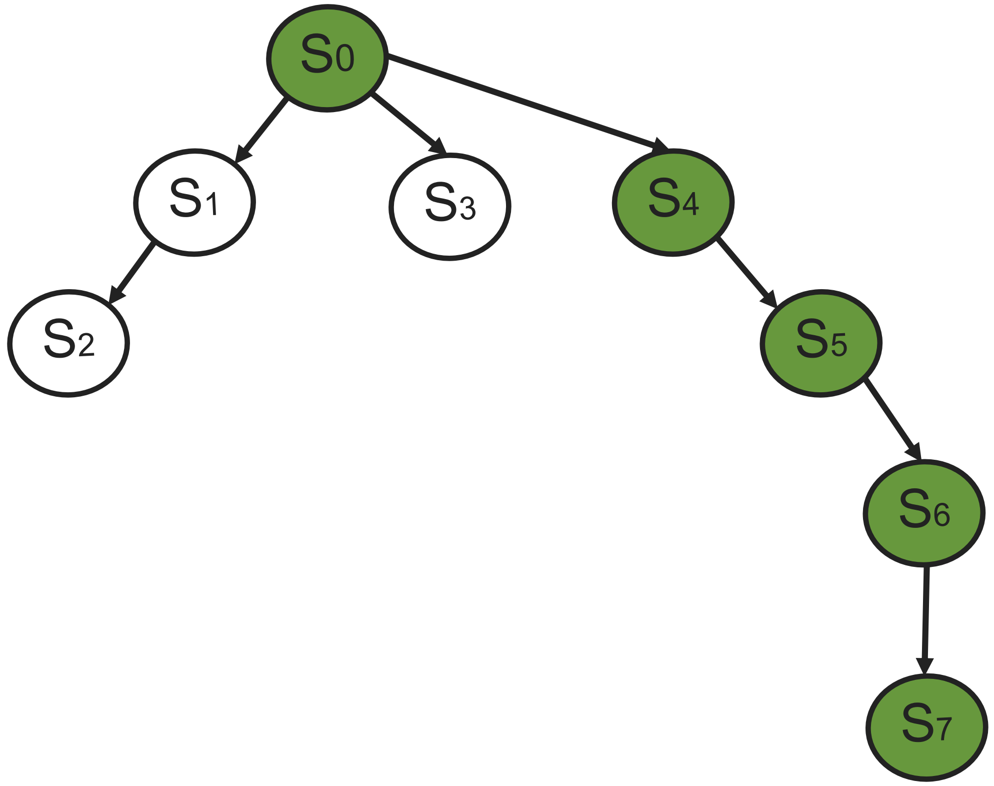


State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
~~S_3~~	~~$M(S_3)=\{(1,3)\}$~~
S_4	$M(S_4)=\{(1,4)\}$
S_5	$M(S_5)=\{(1,4), (2,2)\}$
S_6	$M(S_6)=\{(1,4), (2,2), (3,3)\}$

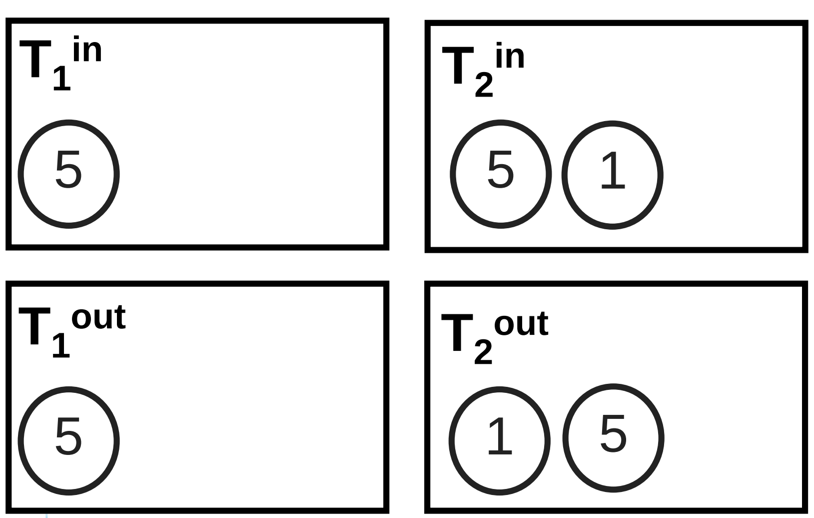


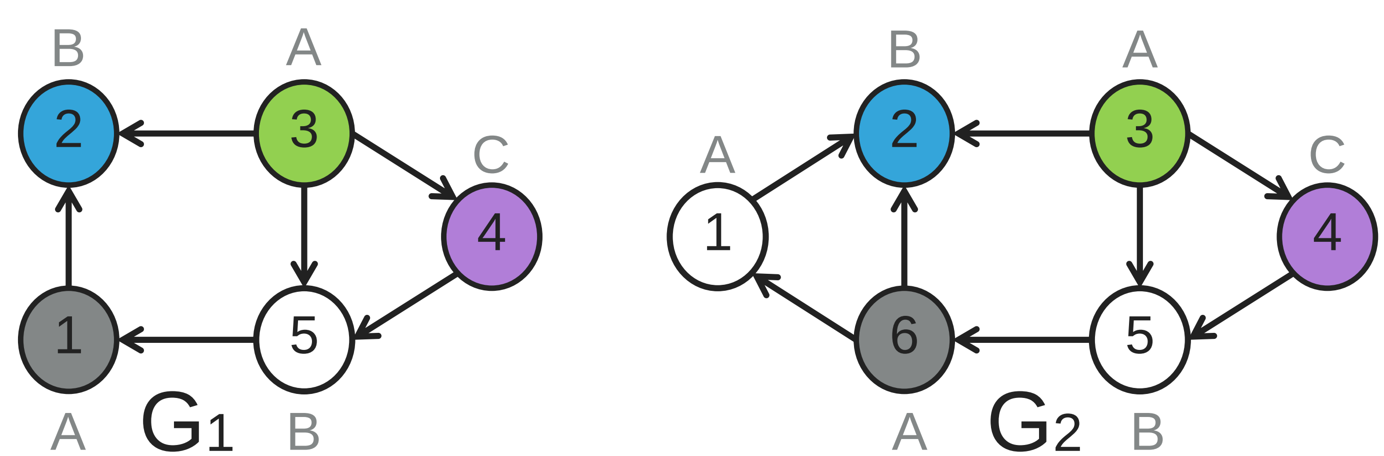


VF2: EXAMPLE

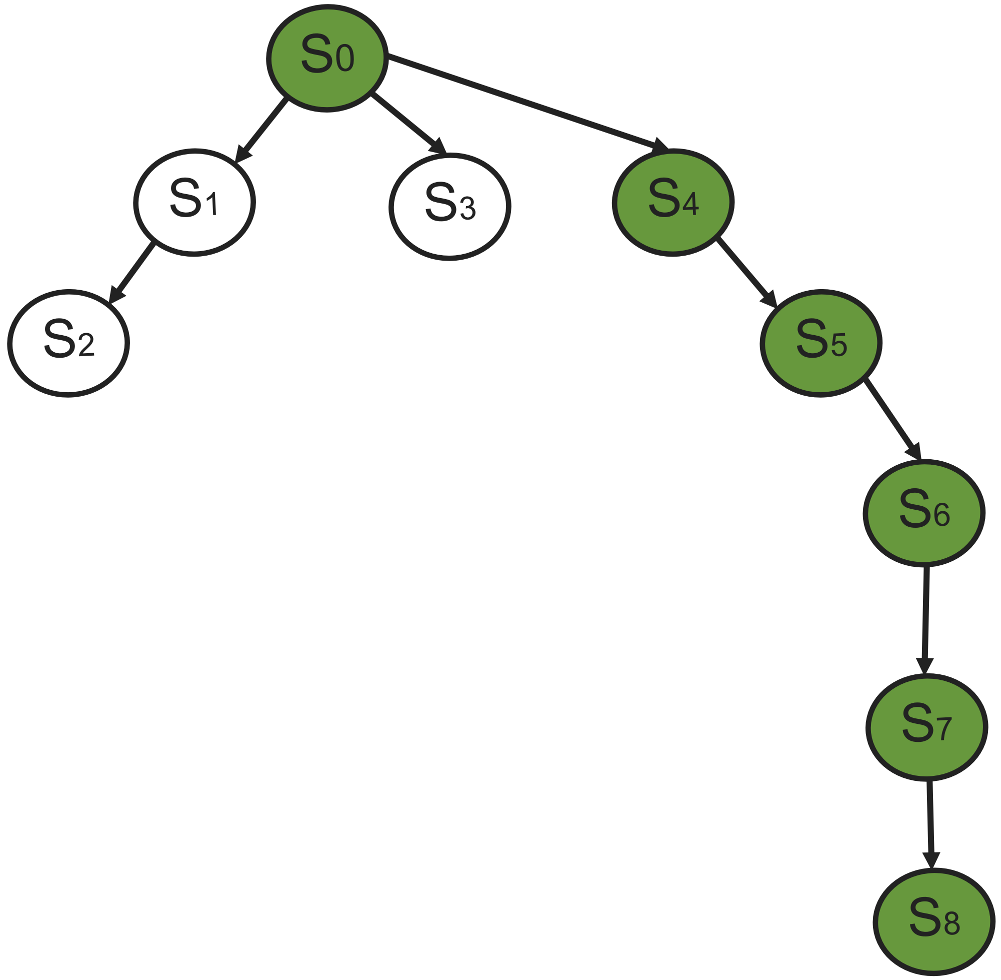


State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
~~S_3~~	~~$M(S_3)=\{(1,3)\}$~~
S_4	$M(S_4)=\{(1,4)\}$
S_5	$M(S_5)=\{(1,4), (2,2)\}$
S_6	$M(S_6)=\{(1,4), (2,2), (3,3)\}$
S_7	$M(S_7)=\{(1,4), (2,2), (3,3), (4,4)\}$

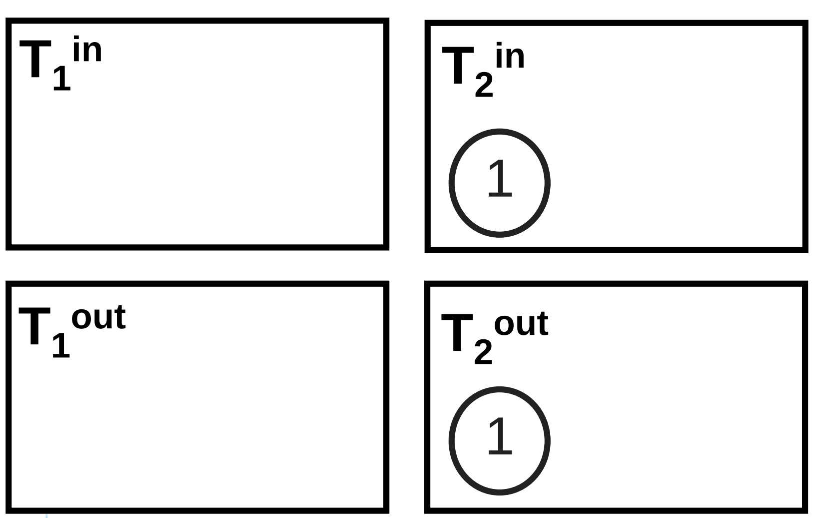


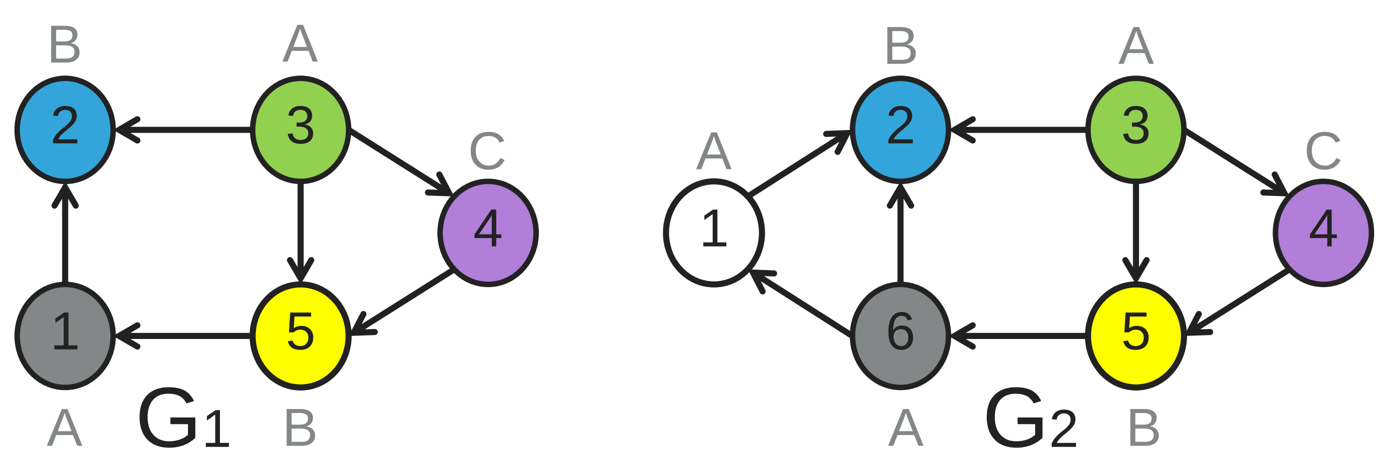


VF2: EXAMPLE

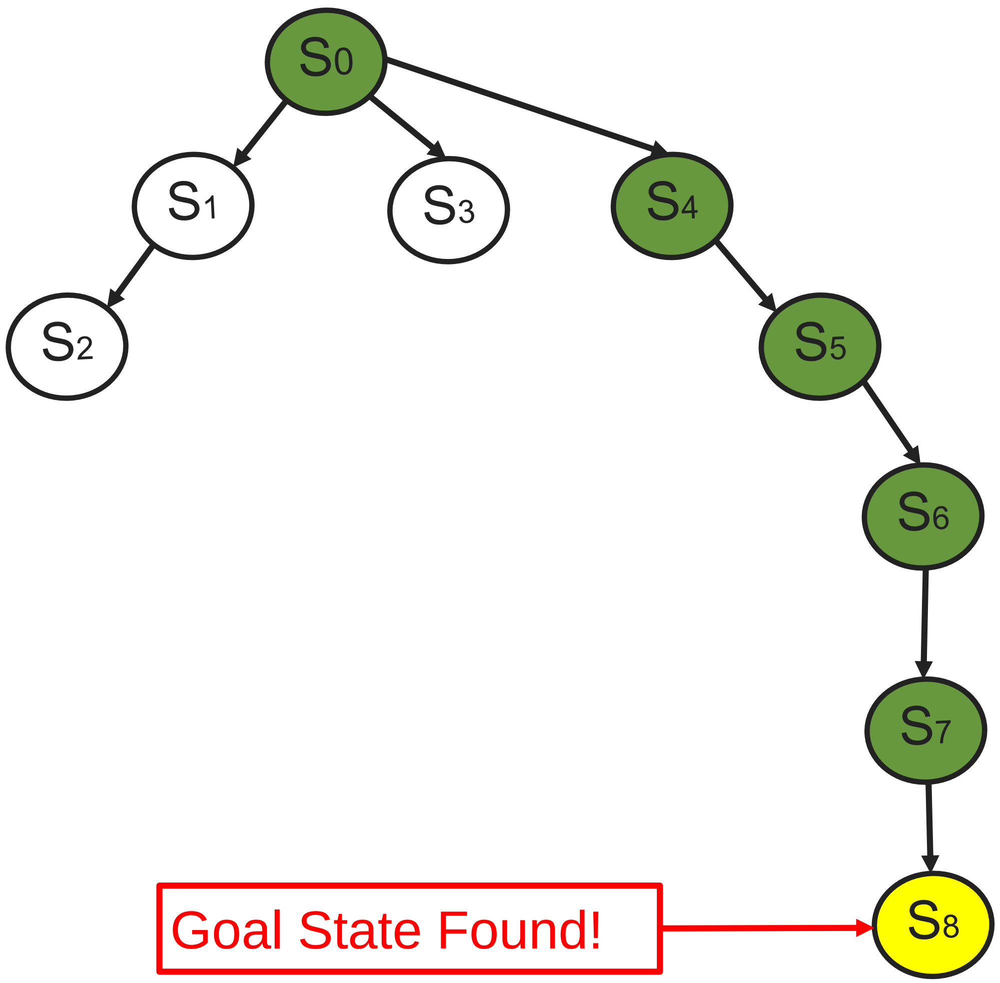


State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
~~S_3~~	~~$M(S_3)=\{(1,3)\}$~~
S_4	$M(S_4)=\{(1,4)\}$
S_5	$M(S_5)=\{(1,4), (2,2)\}$
S_6	$M(S_6)=\{(1,4), (2,2), (3,3)\}$
S_7	$M(S_7)=\{(1,4), (2,2), (3,3), (4,4)\}$
S_8	$M(S_8)=\{(1,4), (2,2), (3,3), (4,4), (5,5)\}$

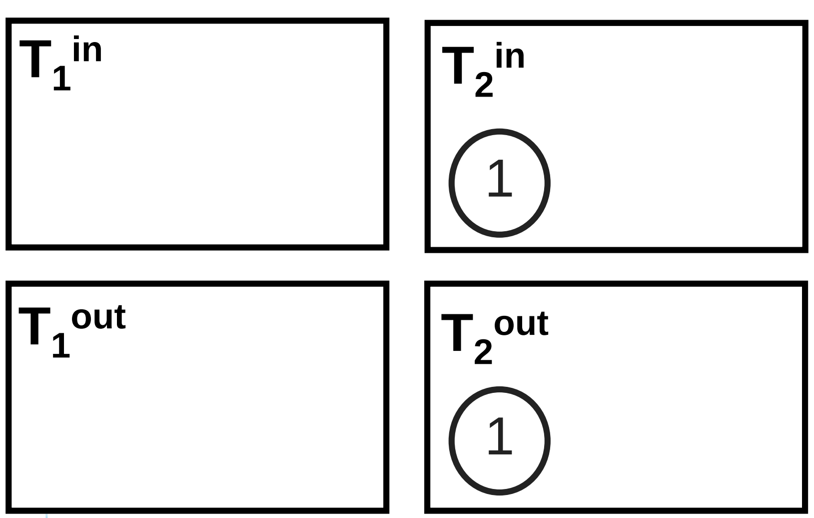


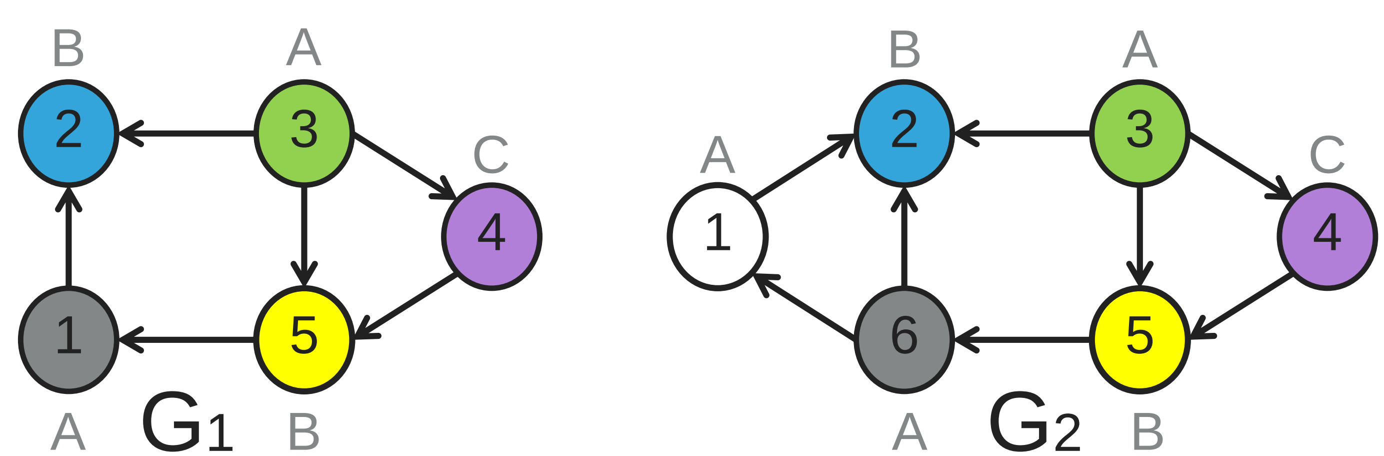


VF2: EXAMPLE



State	Mapping
S_0	$M(S_0)=\{\}$
~~S_1~~	~~$M(S_1)=\{(1,1)\}$~~
~~S_2~~	~~$M(S_2)=\{(1,1),(2,2)\}$~~
~~S_3~~	~~$M(S_3)=\{(1,3)\}$~~
S_4	$M(S_4)=\{(1,4)\}$
S_5	$M(S_5)=\{(1,4), (2,2)\}$
S_6	$M(S_6)=\{(1,4), (2,2), (3,3)\}$
S_7	$M(S_7)=\{(1,4), (2,2), (3,3), (4,4)\}$
S_8	**$M(S_8)=\{(1,4), (2,2), (3,3), (4,4), (5,5)\}$**





FROM VF2 TO VF3 (2017)

- Why it is necessary to improve VF?
  - At the time of VF2 the main applications were:
    - Fingerprint recognition
    - Analysis of simple biological structures (molecules, small proteins)
    - Simple computer vision problems
  - These applications generate sparse graphs of at least 1000 nodes
  - …More than ten years later… new challenging problems arose
    - Complex biological networks (eg. Proteins, interaction netwoks)
    - Social network analysis
    - Knowledge base as graphs
  - Dense and large graphs from thousands to billions of nodes

VF3: INHERITANCE FROM VF2

- **State Space Representation**
  - Each state is partial solution.
  - A goal is consistent complete solution.

- **Depth-First** search with backtracking
  - State space as a tree by a total order relationship.

- **Feasibility rules** to explore the space
  - Exploration of consistent states only.
  - 2-levels Look-ahead.

VF3: WHAT IS NEW?

1. Node ordering:
   - Space exploration insensitive wrt permutation of the nodes
   - Most constrained and rare node first
     - Use a sort of node probability combined with structural constraints
   - Defining an *exploration sequence* on the pattern graph so as to pre-calculate the sets used to explore it

2. Definition of a node classification function
   - Nodes are divided in disjoined subsets (feasibility sets)
   - Nodes in different feasibility sets, they will never be matched

3. Pattern sets pre-calculation
   - All the sets on the pattern graph are computed before the matching
     - Reduced the time to process each state

VF3: SORTING EXAMPLE

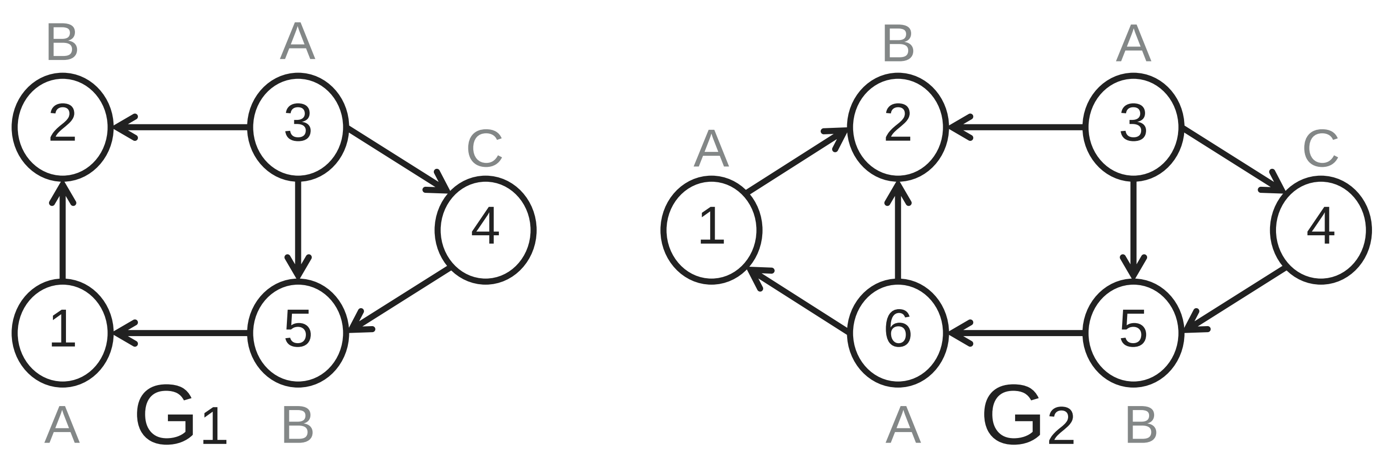
$$P_f(u) = Pr(\lambda_{v2}(v) = \lambda_{v1}(u), d^{in}(v) \geq d^{in}(u), d^{out}(v) \geq d^{out}(u))$$

$$P_f(u) = P_l(\lambda_{v1}(u)) \cdot \sum_{d' \geq d^{in}(u)} P_d^{in}(d') \cdot \sum_{d' \geq d^{out}(u)} P_d^{out}(d')$$

Node	$\sum P_d^{out}$	$\sum P_d^{in}$	P_l	P_f
1	0.83	0.83	0.50	0.34
2	1.00	0.33	0.33	0.11
3	0.16	1.00	0.50	0.08
4	0.83	0.83	0.16	0.11
5	0.83	0.33	0.33	0.09

Ng = {}

Node	Connections with Ng	degree
1	0	2
2	0	2
3	0	3
4	0	2
5	0	3



VF3: SORTING EXAMPLE

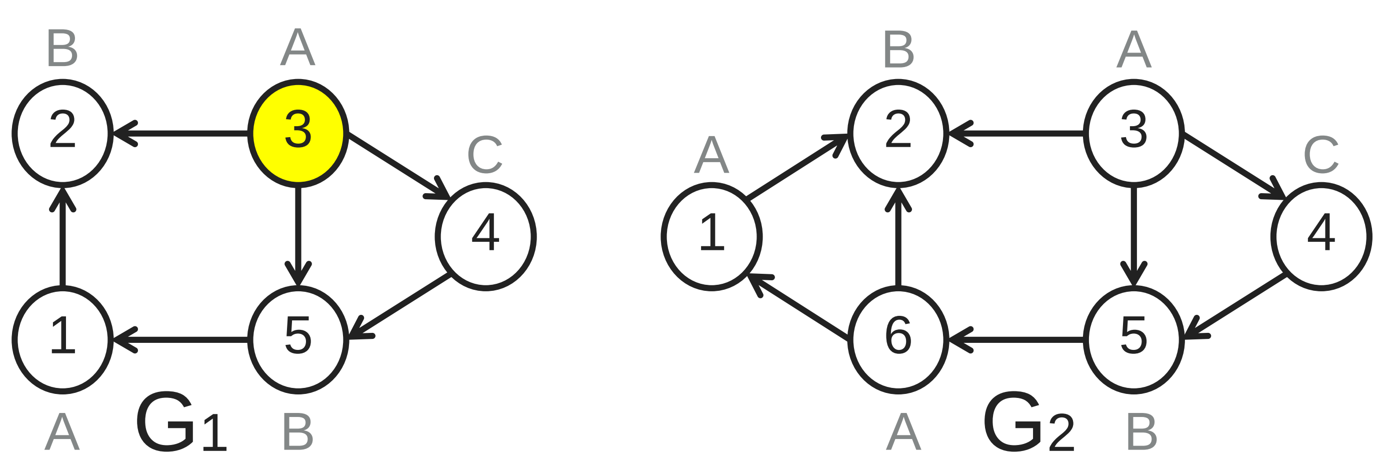
$$P_f(u) = \quad Pr(\lambda_{v2}(v) = \lambda_{v1}(u), d^{in}(v) \geq d^{in}(u), d^{out}(v) \geq d^{out}(u))$$

$$P_f(u) = \quad P_l(\lambda_{v1}(u)) \cdot \sum_{d' \geq d^{in}(u)} P_d^{in}(d') \cdot \sum_{d' \geq d^{out}(u)} P_d^{out}(d')$$

Node	$\sum P_d^{out}$	$\sum P_d^{in}$	P_l	P_f
1	0.83	0.83	0.50	0.34
2	1.00	0.33	0.33	0.11
3	0.16	1.00	0.50	0.08
4	0.83	0.83	0.16	0.11
5	0.83	0.33	0.33	0.09

Ng = {}

Node	Connections with Ng	degree
1	0	2
2	0	2
3	0	3
4	0	2
5	0	3



VF3: SORTING EXAMPLE

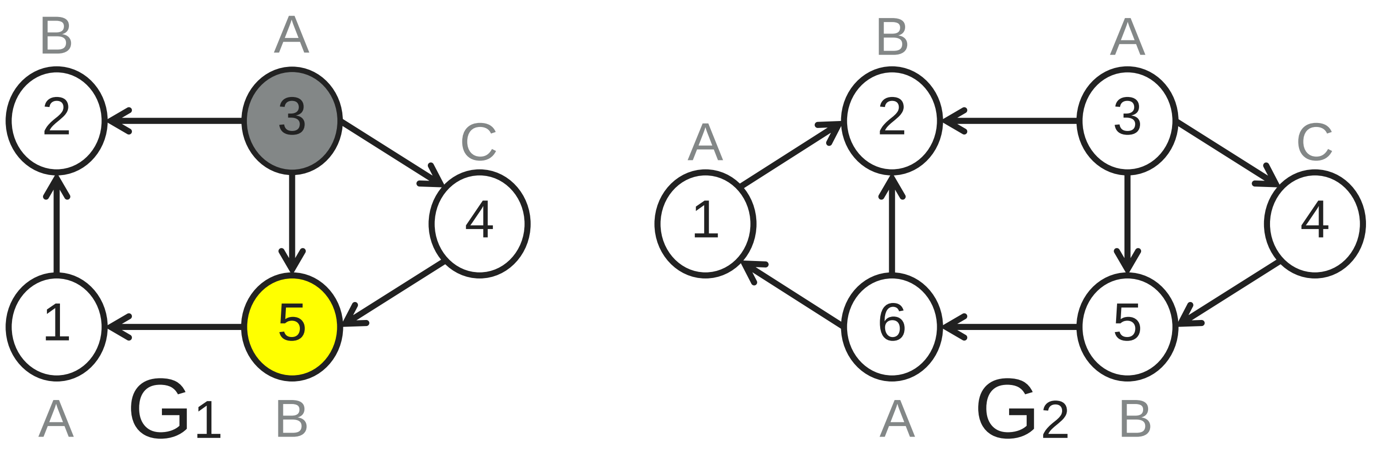
$$P_f(u) = Pr(\lambda_{v2}(v) = \lambda_{v1}(u), d^{in}(v) \geq d^{in}(u), d^{out}(v) \geq d^{out}(u))$$

$$P_f(u) = P_l(\lambda_{v1}(u)) \cdot \sum_{d' \geq d^{in}(u)} P_d^{in}(d') \cdot \sum_{d' \geq d^{out}(u)} P_d^{out}(d')$$

Node	$\sum P_d^{out}$	$\sum P_d^{in}$	P_l	P_f
1	0.83	0.83	0.50	0.34
2	1.00	0.33	0.33	0.11
3	0.16	1.00	0.50	0.08
4	0.83	0.83	0.16	0.11
5	0.83	0.33	0.33	0.09

Ng = {3}

Node	Connections with Ng	degree
1	0	2
2	1	2
3	-	3
4	1	2
5	1	3



VF3: SORTING EXAMPLE

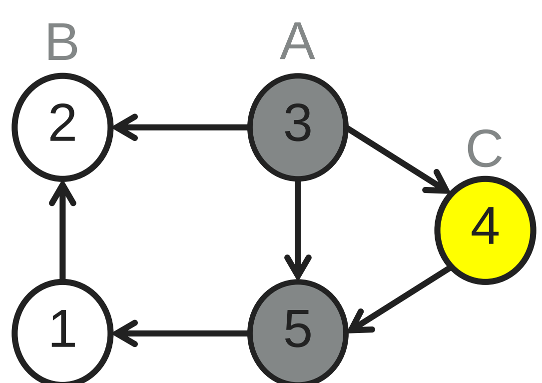
$$P_f(u) = \quad Pr(\lambda_{v2}(v) = \lambda_{v1}(u), d^{in}(v) \geq d^{in}(u), d^{out}(v) \geq d^{out}(u))$$

$$P_f(u) = \quad P_l(\lambda_{v1}(u)) \cdot \sum_{d' \geq d^{in}(u)} P_d^{in}(d') \cdot \sum_{d' \geq d^{out}(u)} P_d^{out}(d')$$

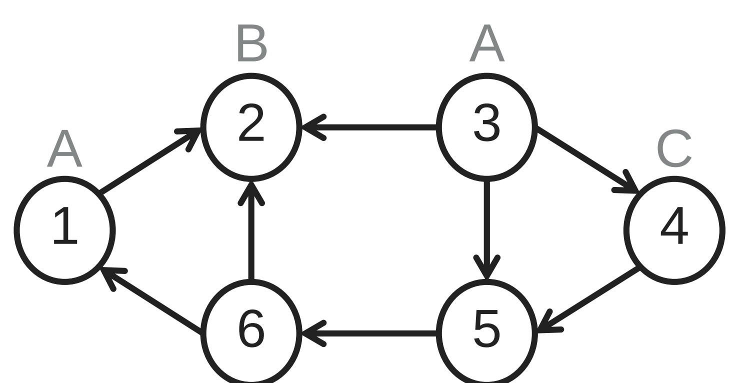
Node	$\sum P_d^{out}$	$\sum P_d^{in}$	P_l	P_f
1	0.83	0.83	0.50	0.34
2	1.00	0.33	0.33	0.11
3	0.16	1.00	0.50	0.08
4	0.83	0.83	0.16	0.11
5	0.83	0.33	0.33	0.09

Ng = {3,5}

Node	Connections with Ng	degree
1	1	2
2	1	2
3	-	3
4	2	2
5	-	3



G1



G2

VF3: SORTING EXAMPLE

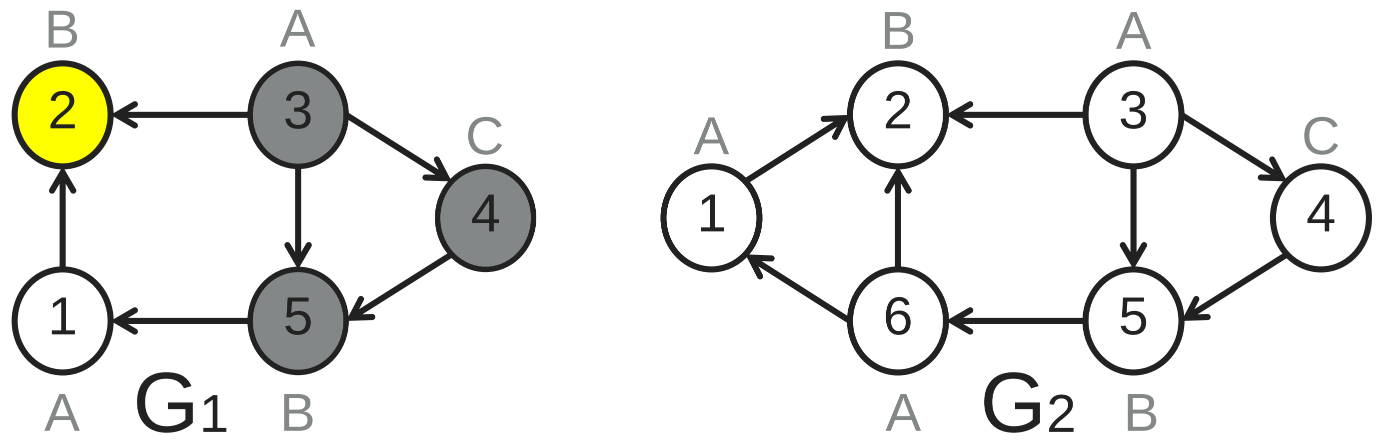
$$P_f(u) = Pr(\lambda_{v2}(v) = \lambda_{v1}(u), d^{in}(v) \geq d^{in}(u), d^{out}(v) \geq d^{out}(u))$$

$$P_f(u) = P_l(\lambda_{v1}(u)) \cdot \sum_{d' \geq d^{in}(u)} P_d^{in}(d') \cdot \sum_{d' \geq d^{out}(u)} P_d^{out}(d')$$

Node	$\sum P_d^{out}$	$\sum P_d^{in}$	P_l	P_f
1	0.83	0.83	0.50	0.34
2	1.00	0.33	0.33	0.11
3	0.16	1.00	0.50	0.08
4	0.83	0.83	0.16	0.11
5	0.83	0.33	0.33	0.09

Ng = {3,5,4}

Node	Connections with Ng	degree
1	1	2
2	1	2
3	-	3
4	-	2
5	-	3



VF3: SORTING EXAMPLE

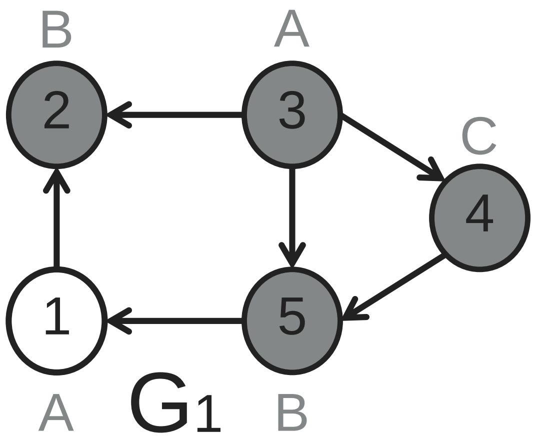
$$P_f(u) = \quad Pr(\lambda_{v2}(v) = \lambda_{v1}(u), d^{in}(v) \geq d^{in}(u), d^{out}(v) \geq d^{out}(u))$$

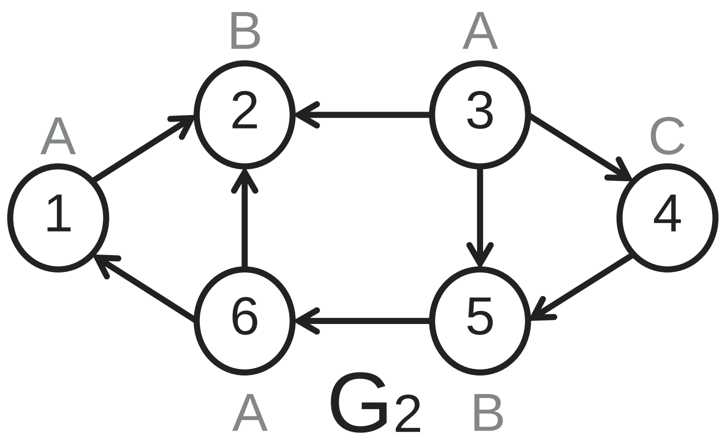
$$P_f(u) = \quad P_l(\lambda_{v1}(u)) \cdot \sum_{d' \geq d^{in}(u)} P_d^{in}(d') \cdot \sum_{d' \geq d^{out}(u)} P_d^{out}(d')$$

Node	$\sum P_d^{out}$	$\sum P_d^{in}$	P_l	P_f
1	0.83	0.83	0.50	0.34
2	1.00	0.33	0.33	0.11
3	0.16	1.00	0.50	0.08
4	0.83	0.83	0.16	0.11
5	0.83	0.33	0.33	0.09

Ng = {3,5,4,2}

Node	Connections with Ng	degree
1	2	2
2	-	2
3	-	3
4	-	2
5	-	3





VF3: SORTING EXAMPLE

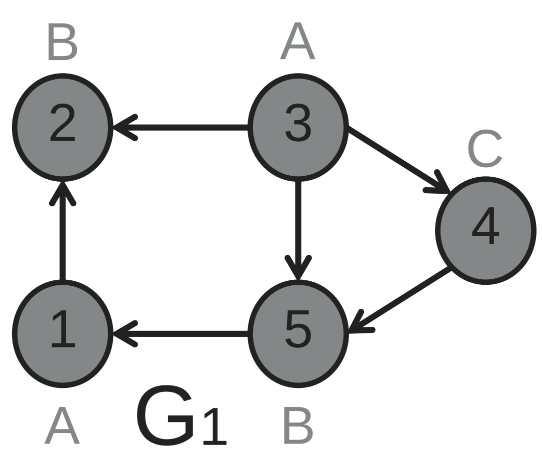
$$P_f(u) = Pr(\lambda_{v2}(v) = \lambda_{v1}(u), d^{in}(v) \geq d^{in}(u), d^{out}(v) \geq d^{out}(u))$$

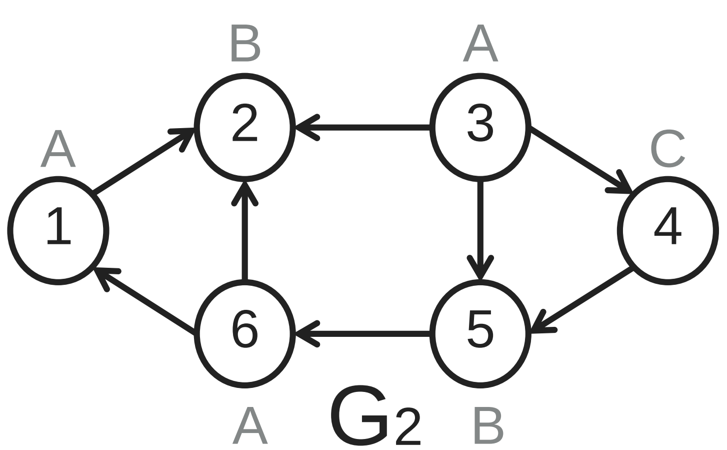
$$P_f(u) = P_l(\lambda_{v1}(u)) \cdot \sum_{d' \geq d^{in}(u)} P_d^{in}(d') \cdot \sum_{d' \geq d^{out}(u)} P_d^{out}(d')$$

Node	$\sum P_d^{out}$	$\sum P_d^{in}$	P_l	P_f
1	0.83	0.83	0.50	0.34
2	1.00	0.33	0.33	0.11
3	0.16	1.00	0.50	0.08
4	0.83	0.83	0.16	0.11
5	0.83	0.33	0.33	0.09

Ng = {3,5,4,2,1}

Node	Connections with Ng	degree
1	-	2
2	-	2
3	-	3
4	-	2
5	-	3



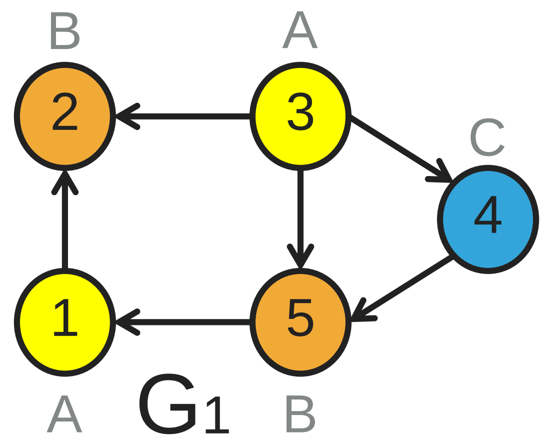


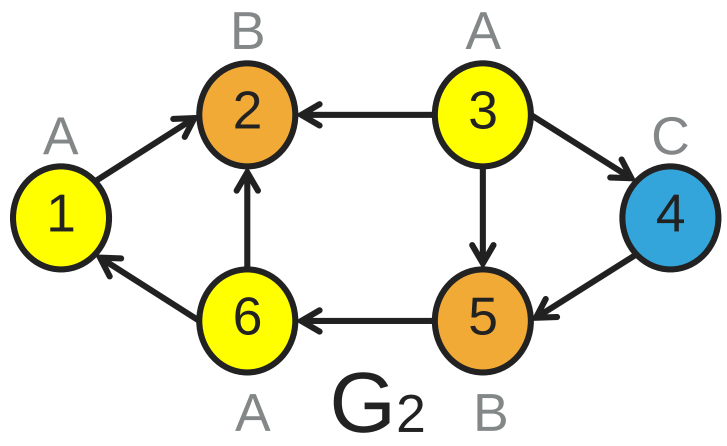
VF3: CLASSIFICATION EXAMPLE

- Let us define a classification function ψ
  - Ψ assigns a class c_i to each node of G1 and G2
  - A simple classification function assigns a different class for each different label in G1 and G2

Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2

Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2
6	A	c_1

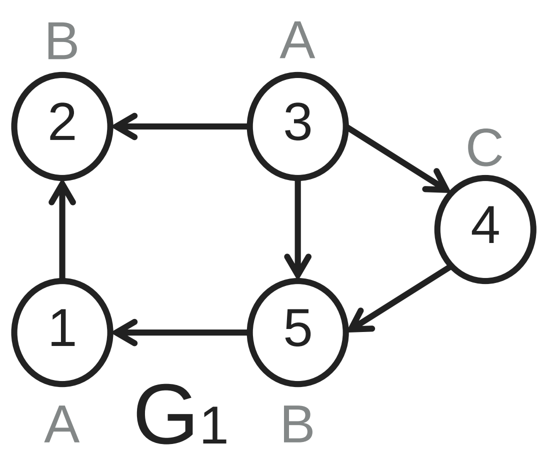




VF3: PREPROCESSING EXAMPLE

Level	Mapped	$\tilde{P}_1^{c1}$	$\tilde{P}_1^{c2}$	$\tilde{P}_1^{c3}$	$\tilde{S}_1^{c1}$	$\tilde{S}_1^{c2}$	$\tilde{S}_1^{c3}$
0	{}	{}	{}	{}	{}	{}	{}

Ng = {3,5,4,2,1}

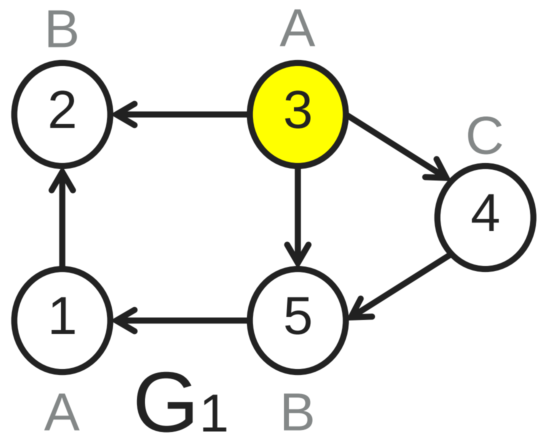


Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2

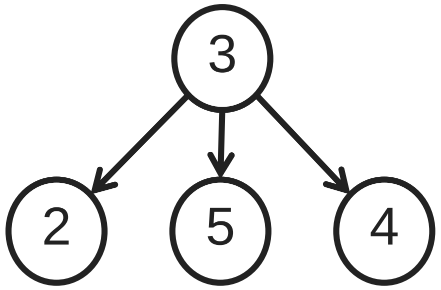
VF3: PREPROCESSING EXAMPLE

Level	Mapped	$\tilde{P}_1^{c1}$	$\tilde{P}_1^{c2}$	$\tilde{P}_1^{c3}$	$\tilde{S}_1^{c1}$	$\tilde{S}_1^{c2}$	$\tilde{S}_1^{c3}$
0	{}	{}	{}	{}	{}	{}	{}
1	{3}	{}	{}	{}	{}	{2,5}	{4}

Ng = {3,5,4,2,1}



Parent tree

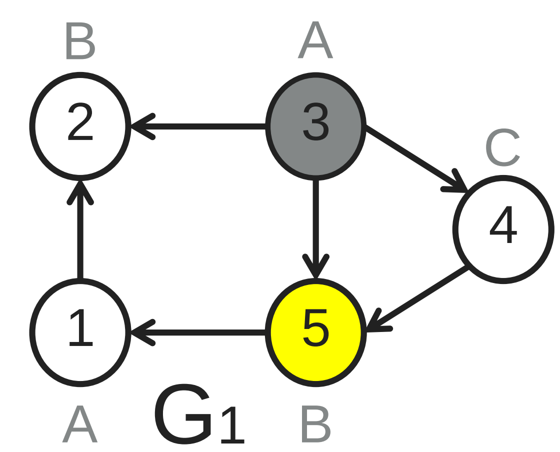


Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2

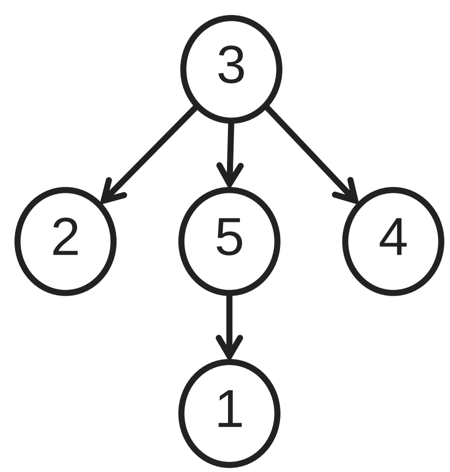
VF3: PREPROCESSING EXAMPLE

Level	Mapped	$\tilde{P}_1^{c1}$	$\tilde{P}_1^{c2}$	$\tilde{P}_1^{c3}$	$\tilde{S}_1^{c1}$	$\tilde{S}_1^{c2}$	$\tilde{S}_1^{c3}$
0	{}	{}	{}	{}	{}	{}	{}
1	{3}	{}	{}	{}	{}	{2,5}	{4}
2	{3,5}	{}	{}	{4}	{1}	{2}	{4}

Ng = {3,5,4,2,1}



Parent tree

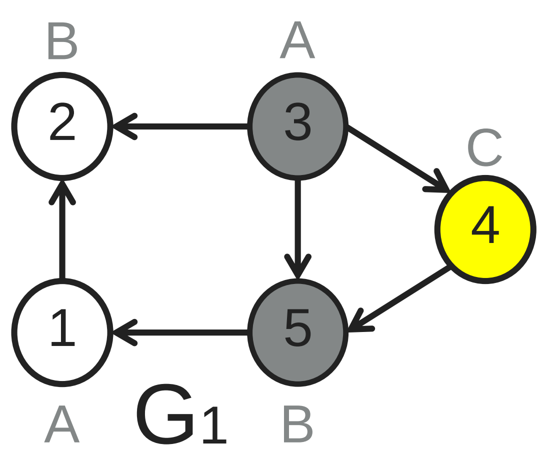


Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2

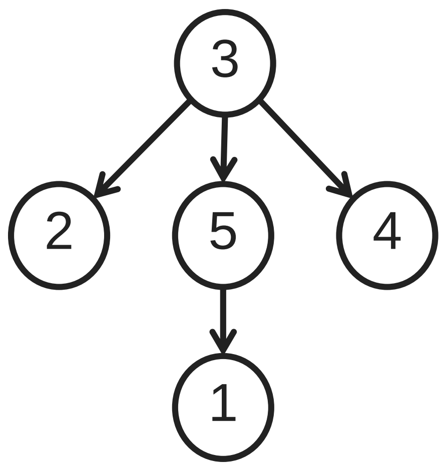
VF3: PREPROCESSING EXAMPLE

Level	Mapped	$\tilde{P}_1^{c1}$	$\tilde{P}_1^{c2}$	$\tilde{P}_1^{c3}$	$\tilde{S}_1^{c1}$	$\tilde{S}_1^{c2}$	$\tilde{S}_1^{c3}$
0	{}	{}	{}	{}	{}	{}	{}
1	{3}	{}	{}	{}	{}	{2,5}	{4}
2	{3,5}	{}	{}	{4}	{1}	{2}	{4}
3	{3,5,4}	{}	{}	{}	{1}	{2}	{}

Ng = {3,5,4,2,1}



Parent tree

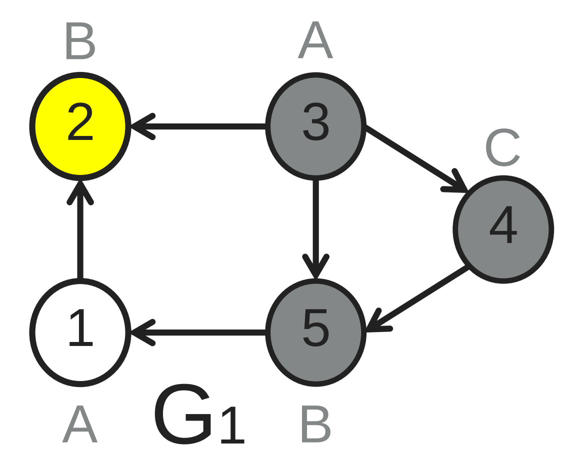


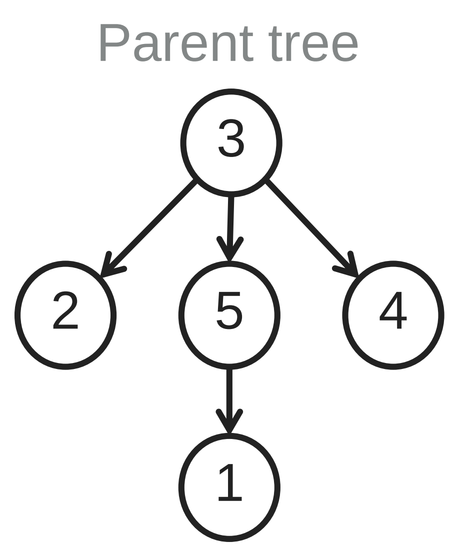
Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2

VF3: PREPROCESSING EXAMPLE

Level	Mapped	$\tilde{P}_1^{c1}$	$\tilde{P}_1^{c2}$	$\tilde{P}_1^{c3}$	$\tilde{S}_1^{c1}$	$\tilde{S}_1^{c2}$	$\tilde{S}_1^{c3}$
0	{}	{}	{}	{}	{}	{}	{}
1	{3}	{}	{}	{}	{}	{2,5}	{4}
2	{3,5}	{}	{}	{4}	{1}	{2}	{4}
3	{3,4,5}	{}	{}	{}	{1}	{2}	{}
4	{2,3,4,5}	{1}	{}	{}	{1}	{}	{}

Ng = {3,5,4,2,1}



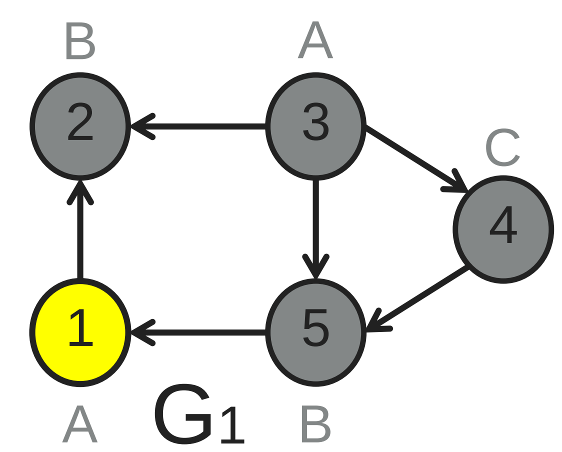


Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2

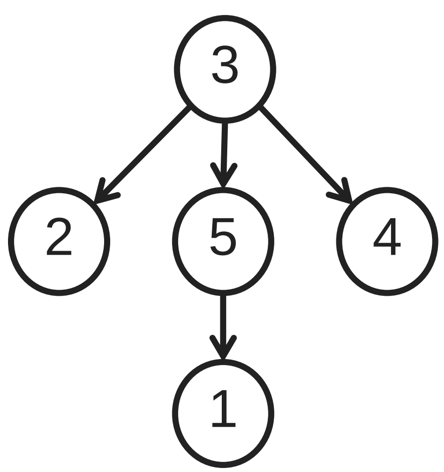
VF3: PREPROCESSING EXAMPLE

Level	Mapped	$\tilde{P}_1^{c1}$	$\tilde{P}_1^{c2}$	$\tilde{P}_1^{c3}$	$\tilde{S}_1^{c1}$	$\tilde{S}_1^{c2}$	$\tilde{S}_1^{c3}$
0	{}	{}	{}	{}	{}	{}	{}
1	{3}	{}	{}	{}	{}	{2,5}	{4}
2	{3,5}	{}	{}	{4}	{1}	{2}	{4}
3	{3,4,5}	{}	{}	{}	{1}	{2}	{}
4	{2,3,4,5}	{1}	{}	{}	{1}	{}	{}
5	{1,2,3,4,5}	{}	{}	{}	{}	{}	{}

Ng = {3,5,4,2,1}



Parent tree

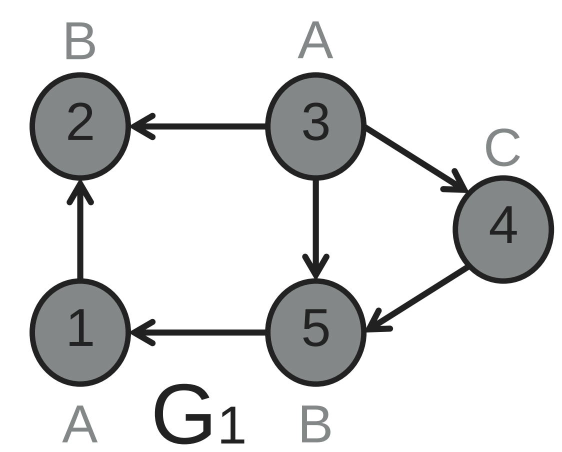


Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2

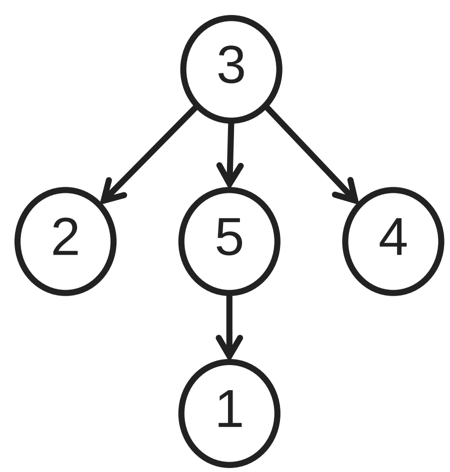
VF3: PREPROCESSING EXAMPLE

Level	Mapped	$\tilde{P}_1^{c1}$	$\tilde{P}_1^{c2}$	$\tilde{P}_1^{c3}$	$\tilde{S}_1^{c1}$	$\tilde{S}_1^{c2}$	$\tilde{S}_1^{c3}$
0	{}	{}	{}	{}	{}	{}	{}
1	{3}	{}	{}	{}	{}	{2,5}	{4}
2	{3,5}	{}	{}	{4}	{1}	{2}	{4}
3	{3,4,5}	{}	{}	{}	{1}	{2}	{}
4	{2,3,4,5}	{1}	{}	{}	{1}	{}	{}
5	{1,2,3,4,5}	{}	{}	{}	{}	{}	{}

Ng = {3,5,4,2,1}



Parent tree



Node	Label	Class
1	A	c_1
2	B	c_2
3	A	c_1
4	C	c_3
5	B	c_2

VF3 FEASIBILITY SETS AND RULES

Straightened look-ahead:

- The feasibility rules are evaluated on the feasibility sets
  - A more contrainted feasibility check
    - Stronger pruning of unfruitful states

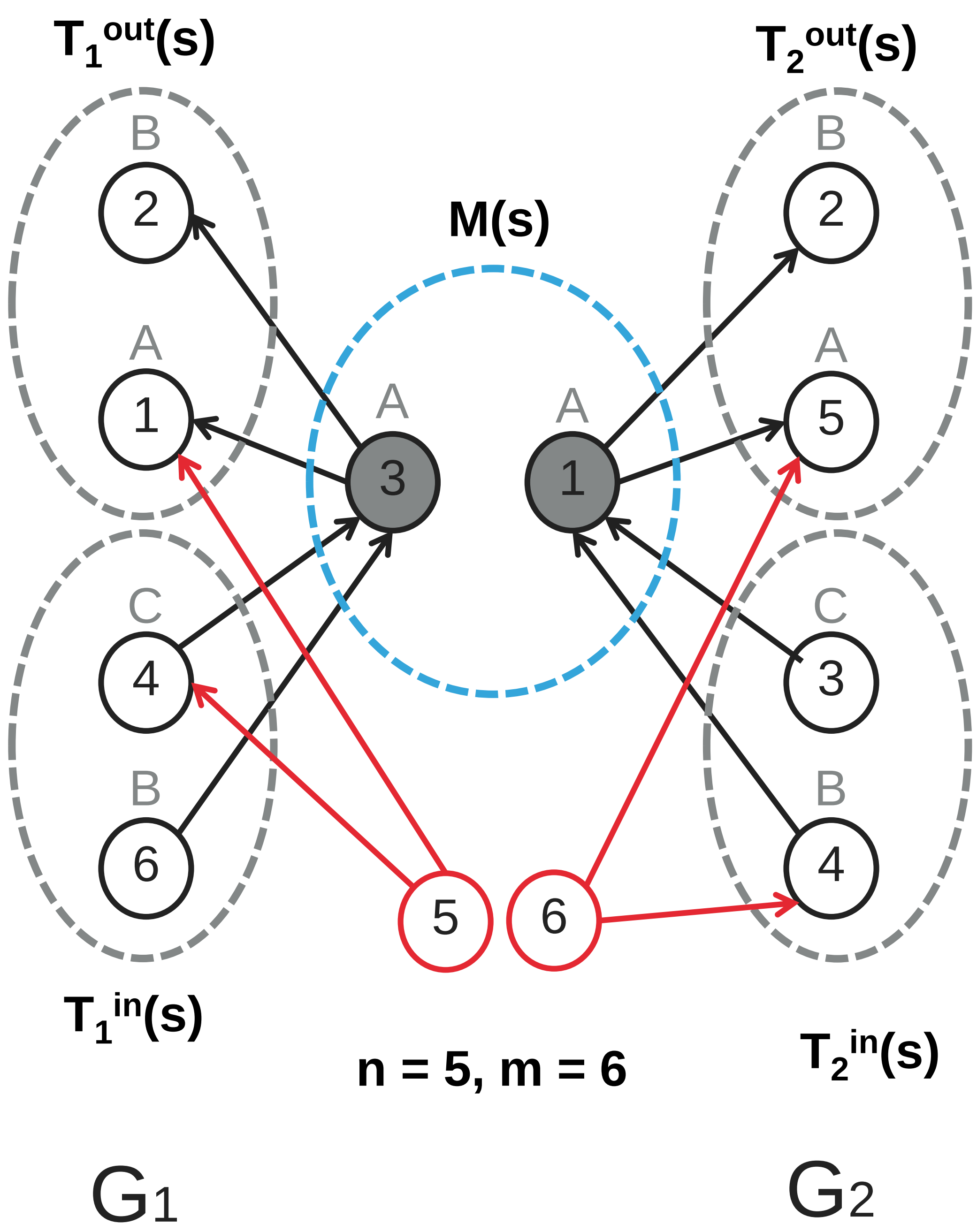
$$\textsc{IsFeasible}(s_c, u_n, v_n) = F_s(s_c, u_n, v_n) \wedge F_t(s_c, u_n, v_n)$$

$$F_t(s_c, u_n, v_n) = \\ F_c(s_c, u_n, v_n) \wedge F_{la1}(s_c, u_n, v_n) \wedge F_{la2}(s_c, u_n, v_n)$$

$$\begin{aligned} F_c(s_c, u_n, v_n) \Leftrightarrow \\ &\forall u' \in \mathcal{S}_1(u_n) \cap \widetilde{M}_1(s_c)\, \exists v' = \widetilde{\mu}(s_c, u') \in \mathcal{S}_2(v_n) \\ &\wedge\ \forall u' \in \mathcal{P}_1(u_n) \cap \widetilde{M}_1(s_c)\, \exists v' = \widetilde{\mu}(s, u') \in \mathcal{P}_2(v_n) \\ &\wedge\ \forall v' \in \mathcal{S}_2(v_n) \cap \widetilde{M}_2(s_c)\, \exists u' = \widetilde{\mu}^{-1}(s_c, v') \in \mathcal{S}_1(u_n) \\ &\wedge\ \forall v' \in \mathcal{P}_2(v_n) \cap \widetilde{M}_2(s_c)\, \exists u' = \widetilde{\mu}^{-1}(s_c, v') \in \mathcal{P}_1(u_n) \end{aligned}$$

VF3 FEASIBILITY RULES AND SETS

- Feasibility Sets and Classification

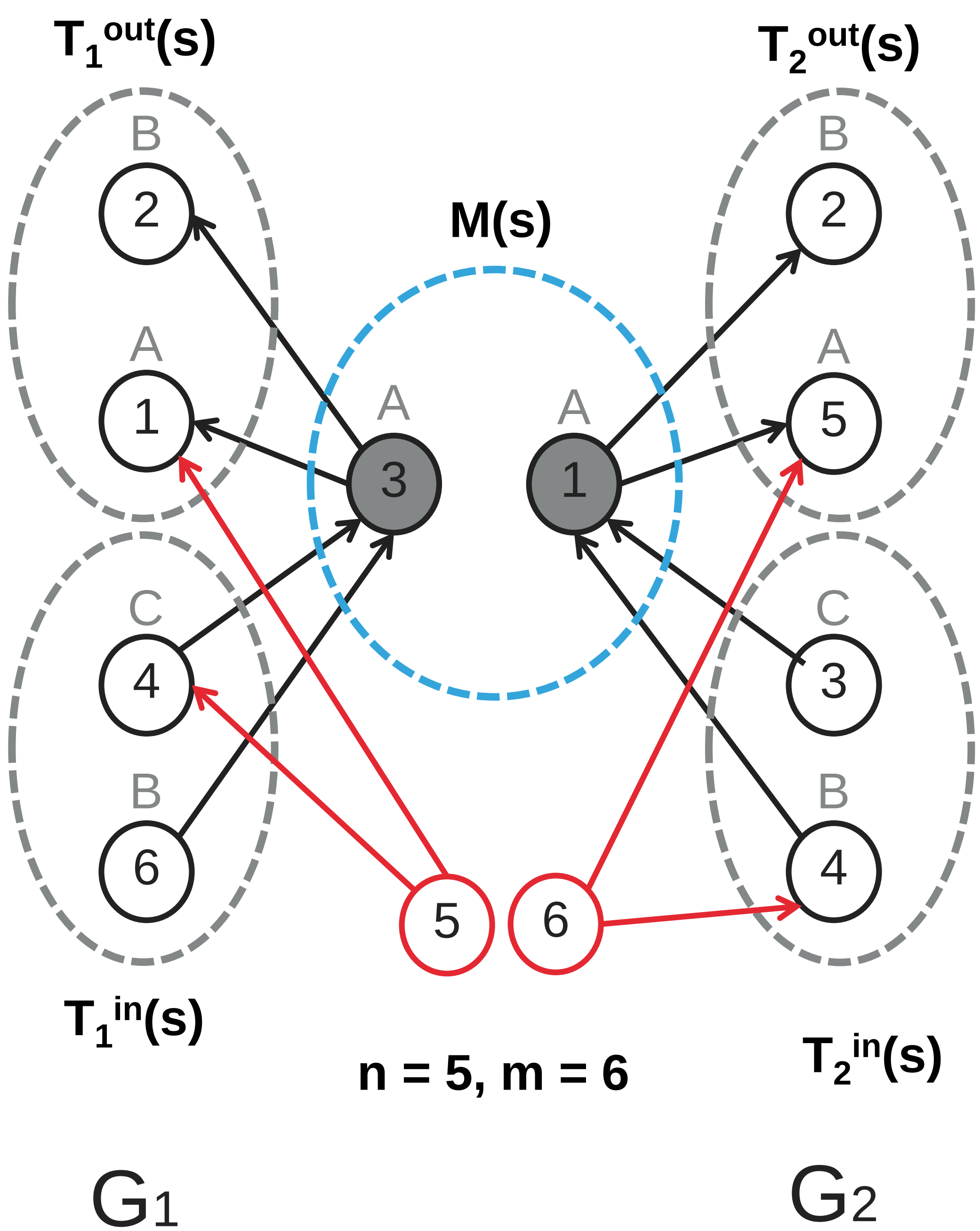


$$R_{\text{in}}(s,n,m) \Longleftrightarrow$$
$$(\text{Card}(\text{Succ}(G_1,n) \cap T_1^{\text{in}}(s)) \geq \text{Card}(\text{Succ}(G_2,m) \cap T_2^{\text{in}}(s))) \wedge$$
$$(\text{Card}(\text{Pred}(G_1,n) \cap T_1^{\text{in}}(s)) \geq \text{Card}(\text{Pred}(G_2,m) \cap T_2^{\text{in}}(s))),$$

$$R_{\text{out}}(s,n,m) \Longleftrightarrow$$
$$(\text{Card}(\text{Succ}(G_1,n) \cap T_1^{\text{out}}(s)) \geq \text{Card}(\text{Succ}(G_2,m) \cap T_2^{\text{out}}(s))) \wedge$$
$$(\text{Card}(\text{Pred}(G_1,n) \cap T_1^{\text{out}}(s)) \geq \text{Card}(\text{Pred}(G_2,m) \cap T_2^{\text{out}}(s))),$$

VF3 FEASIBILITY RULES AND SETS

- Feasibility Sets and Classification

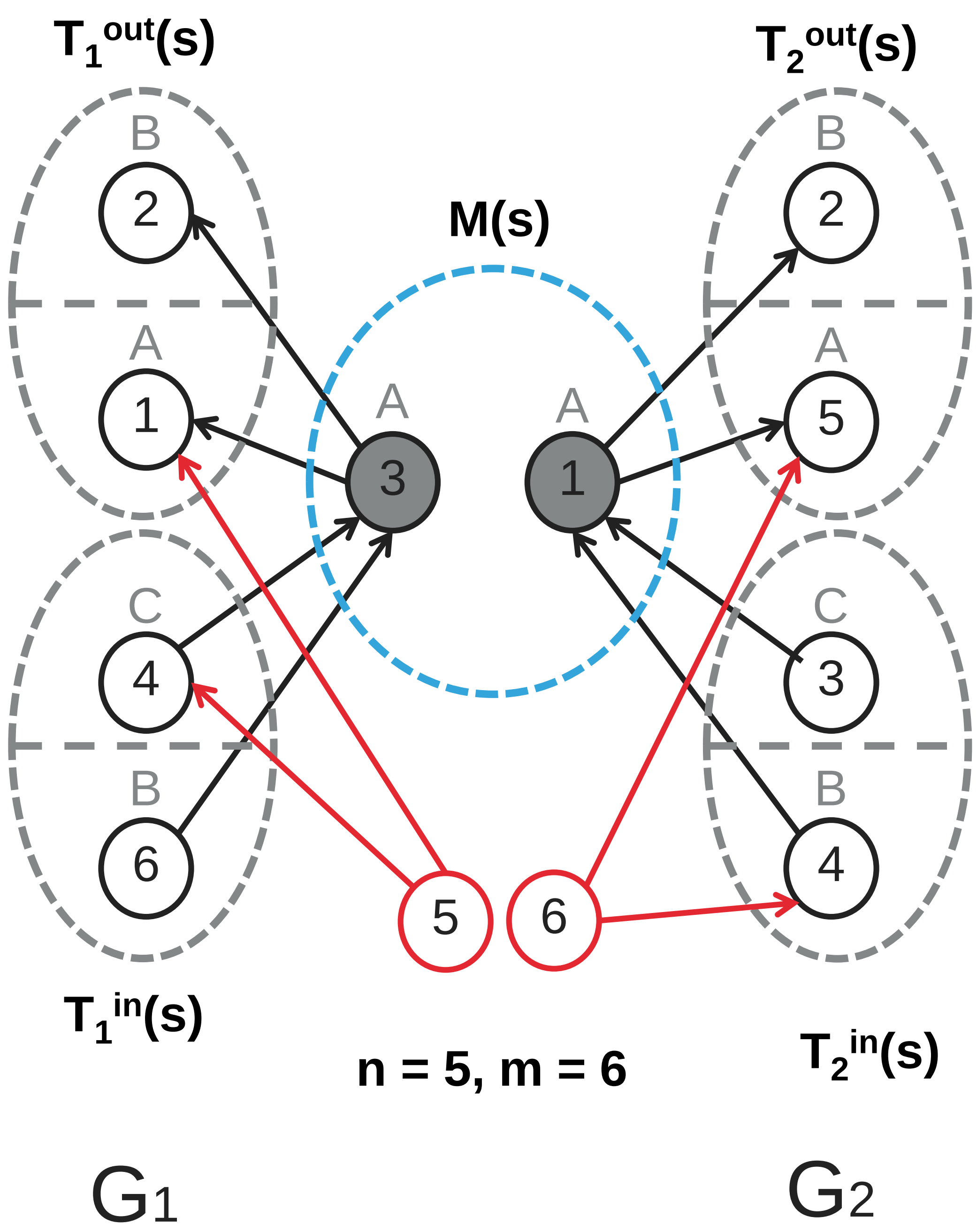


$$R_{in}(s,n,m) \Longleftrightarrow$$
$$(\text{Card}(\text{Succ}(G_1,n) \cap T_1^{in}(s)) \geq \text{Card}(\text{Succ}(G_2,m) \cap T_2^{in}(s))) \wedge$$
$$(\text{Card}(\text{Pred}(G_1,n) \cap T_1^{in}(s)) \geq \text{Card}(\text{Pred}(G_2,m) \cap T_2^{in}(s))),$$

$$R_{out}(s,n,m) \Longleftrightarrow$$
$$(\text{Card}(\text{Succ}(G_1,n) \cap T_1^{out}(s)) \geq \text{Card}(\text{Succ}(G_2,m) \cap T_2^{out}(s))) \wedge$$
$$(\text{Card}(\text{Pred}(G_1,n) \cap T_1^{out}(s)) \geq \text{Card}(\text{Pred}(G_2,m) \cap T_2^{out}(s))),$$

VF3 FEASIBILITY RULES AND SETS

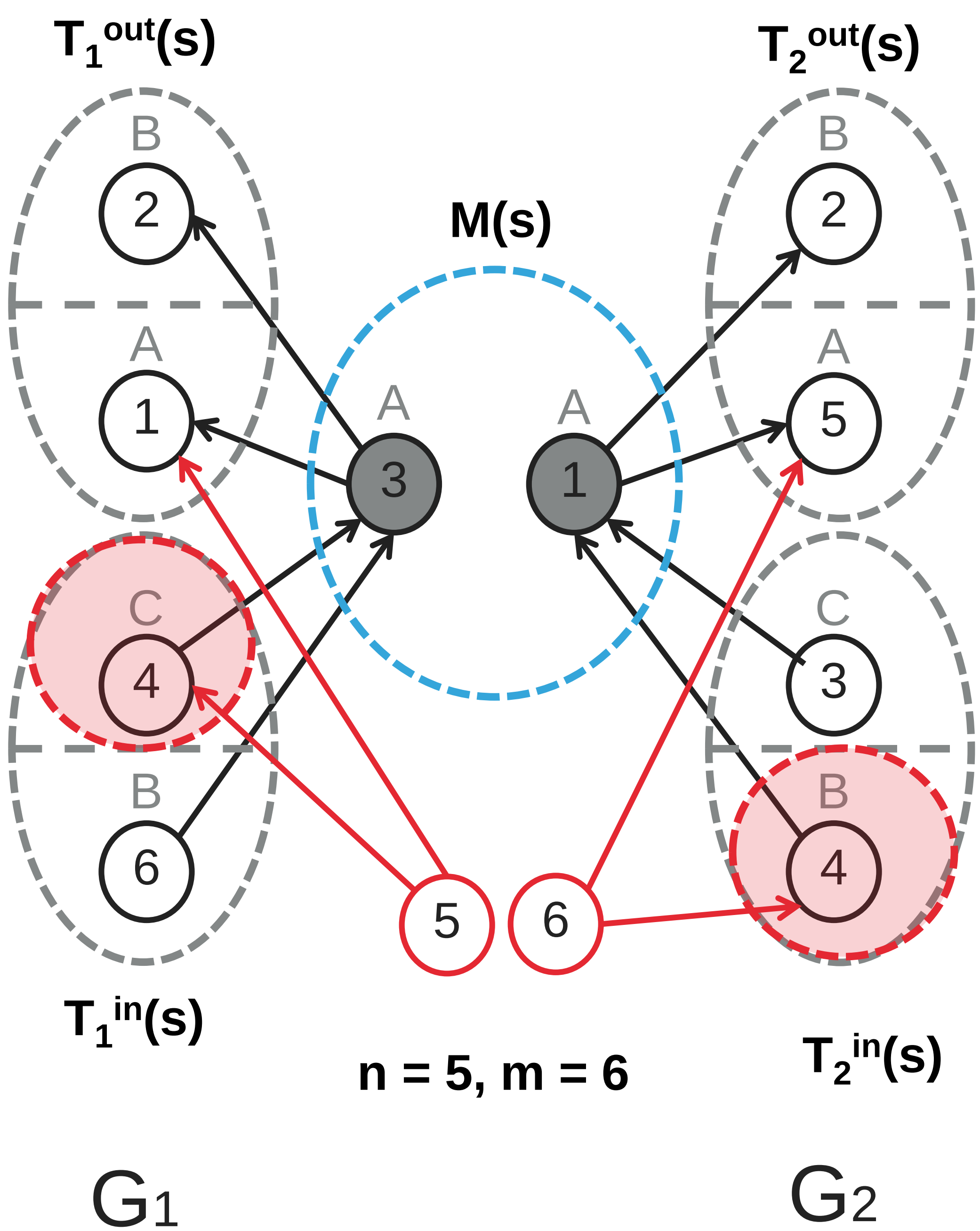
- Feasibility Sets and Classification



$$
\begin{aligned}
F^{i}_{la1}(s_c, u_n, v_n) \iff & \\
& |\mathcal{P}_1(u_n) \cap \widetilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \widetilde{\mathcal{P}}_2^{c_i}(s_c)| \\
\wedge \quad & |\mathcal{P}_1(u_n) \cap \widetilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \widetilde{\mathcal{S}}_2^{c_i}(s_c)| \\
\wedge \quad & |\mathcal{S}_1(u_n) \cap \widetilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \widetilde{\mathcal{P}}_2^{c_i}(s_c)| \\
\wedge \quad & |\mathcal{S}_1(u_n) \cap \widetilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \widetilde{\mathcal{S}}_2^{c_i}(s_c)|
\end{aligned}
$$

VF3 FEASIBILITY RULES AND SETS

- Feasibility Sets and Classification



$$F^i_{la1}(s_c, u_n, v_n) \iff$$

$$|\mathcal{P}_1(u_n) \cap \widetilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \widetilde{\mathcal{P}}_2^{c_i}(s_c)|$$
$$\wedge \quad |\mathcal{P}_1(u_n) \cap \widetilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \widetilde{\mathcal{S}}_2^{c_i}(s_c)|$$
$$\wedge \quad |\mathcal{S}_1(u_n) \cap \widetilde{\mathcal{P}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \widetilde{\mathcal{P}}_2^{c_i}(s_c)|$$
$$\wedge \quad |\mathcal{S}_1(u_n) \cap \widetilde{\mathcal{S}}_1^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \widetilde{\mathcal{S}}_2^{c_i}(s_c)|$$

VF3 FEASIBILITY RULES AND SETS

Straightened look-ahead:

- The feasibility rules are evaluated on the feasibility sets
  - A more contrainted feasibility check
    - Stronger pruning of unfruitful states

$$\textsc{IsFeasible}(s_c, u_n, v_n) = F_s(s_c, u_n, v_n) \wedge F_t(s_c, u_n, v_n)$$

$$F_t(s_c, u_n, v_n) = \\ \quad F_c(s_c, u_n, v_n) \wedge F_{la1}(s_c, u_n, v_n) \wedge F_{la2}(s_c, u_n, v_n)$$

$$F_{la1}(s_c, u_n, v_n) \iff F^1_{la1}(s_c, u_n, v_n) \wedge \ldots \wedge F^q_{la1}(s_c, u_n, v_n)$$

$$\begin{aligned} F^i_{la1}(s_c, u_n, v_n) \iff & \\ & |\mathcal{P}_1(u_n) \cap \widetilde{\mathcal{P}}^{c_i}_1(s_c)| \leq |\mathcal{P}_2(v_n) \cap \widetilde{\mathcal{P}}^{c_i}_2(s_c)| \\ & \wedge \quad |\mathcal{P}_1(u_n) \cap \widetilde{\mathcal{S}}^{c_i}_1(s_c)| \leq |\mathcal{P}_2(v_n) \cap \widetilde{\mathcal{S}}^{c_i}_2(s_c)| \\ & \wedge \quad |\mathcal{S}_1(u_n) \cap \widetilde{\mathcal{P}}^{c_i}_1(s_c)| \leq |\mathcal{S}_2(v_n) \cap \widetilde{\mathcal{P}}^{c_i}_2(s_c)| \\ & \wedge \quad |\mathcal{S}_1(u_n) \cap \widetilde{\mathcal{S}}^{c_i}_1(s_c)| \leq |\mathcal{S}_2(v_n) \cap \widetilde{\mathcal{S}}^{c_i}_2(s_c)| \end{aligned}$$

VF3 FEASIBILITY RULES AND SETS

Straightened look-ahead:

- The feasibility rules are evaluated on the feasibility sets
  - A more contrainted feasibility check
    - Stronger pruning of unfruitful states

$$\textsc{IsFeasible}(s_c, u_n, v_n) = F_s(s_c, u_n, v_n) \wedge F_t(s_c, u_n, v_n)$$

$$F_t(s_c, u_n, v_n) = \\ F_c(s_c, u_n, v_n) \wedge F_{la1}(s_c, u_n, v_n) \wedge F_{la2}(s_c, u_n, v_n)$$

$$F_{la2}(s_c, u_n, v_n) \iff \\ F_{la2}^1(s_c, u_n, v_n) \wedge \ldots \wedge F_{la2}^q(s_c, u_n, v_n)$$

$$F_{la2}^i(s_c, u_n, v_n) \iff \\ |\mathcal{P}_1(u_n) \cap \widetilde{V_1}^{c_i}(s_c)| \leq |\mathcal{P}_2(v_n) \cap \widetilde{V_2}^{c_i}(s_c)| \\ \wedge \quad |\mathcal{S}_1(u_n) \cap \widetilde{V_1}^{c_i}(s_c)| \leq |\mathcal{S}_2(v_n) \cap \widetilde{V_2}^{c_i}(s_c)|$$

VF3: TRANSITION FUNCTION

New State Transition Function

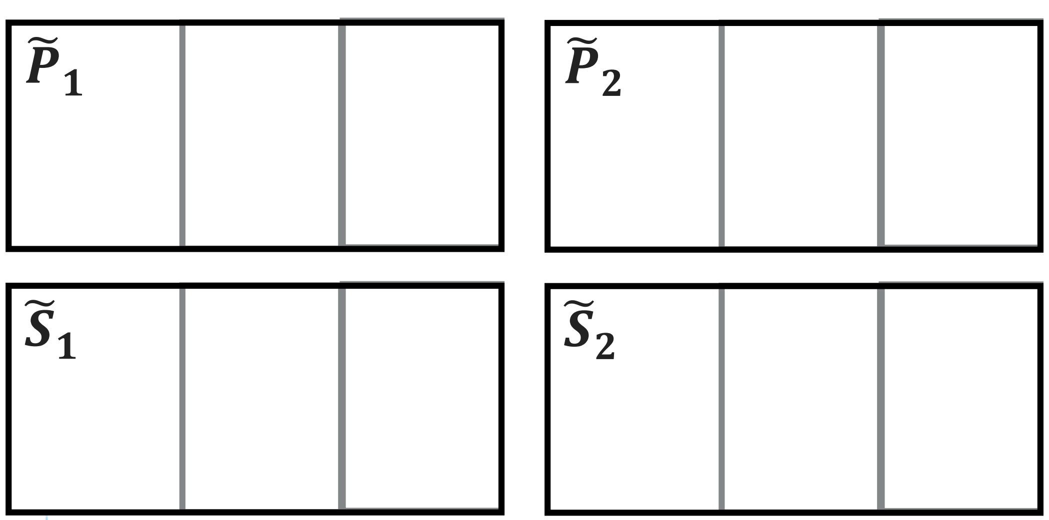
- New heuristic to choose the next candidate pair of nodes to generate a new state
  - VF2 wasted a big amount of time in selecting a new couple of nodes
    - Many tried couples were unfruitful

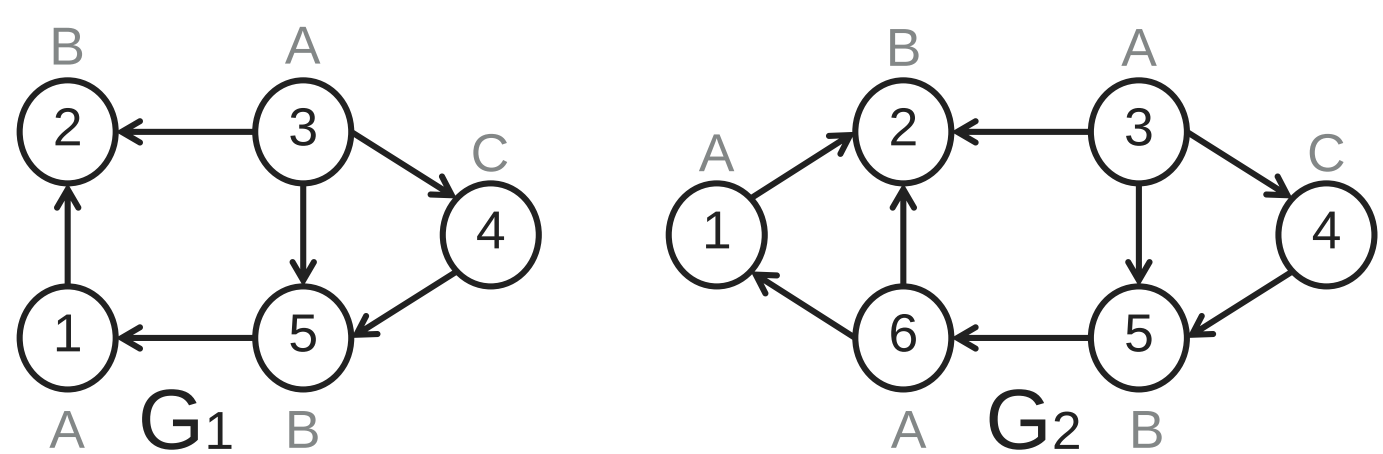
- Definition of a new set of candidate nodes
  - The first node is given by the exploration sequence
  - The second node is selected from a proper set of unmatched nodes
    - The set used by VF3 is a small subset of the terminal sets of VF2
    - Nodes belonging to different feasibility sets (and classes) will never be paired by VF3
  - The time for choosing a new candidate is considerably lower from VF2 to VF3

VF3: EXAMPLE

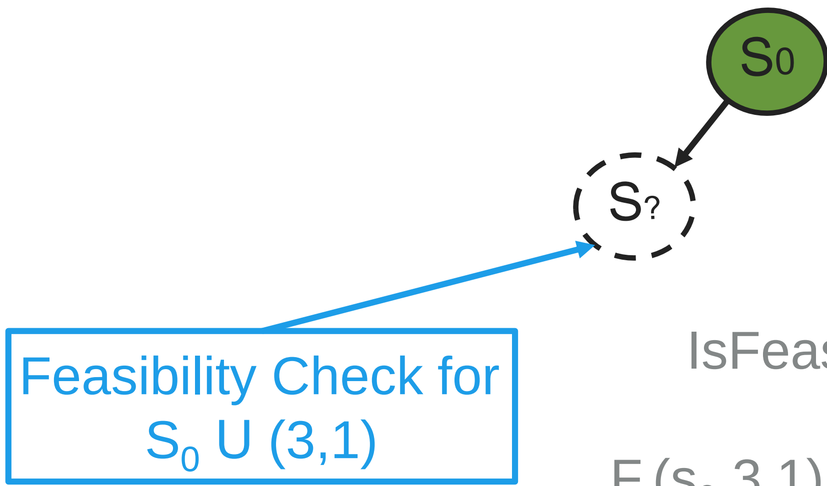
State	Mapping
S_0	$M(S_0)=\{\}$

$Ng = \{3,5,4,2,1\}$





VF3: EXAMPLE

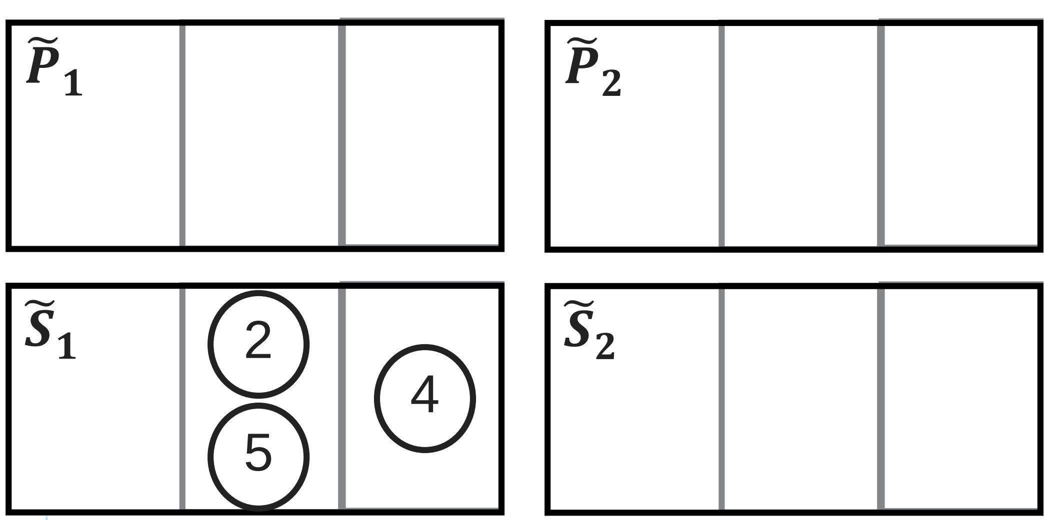


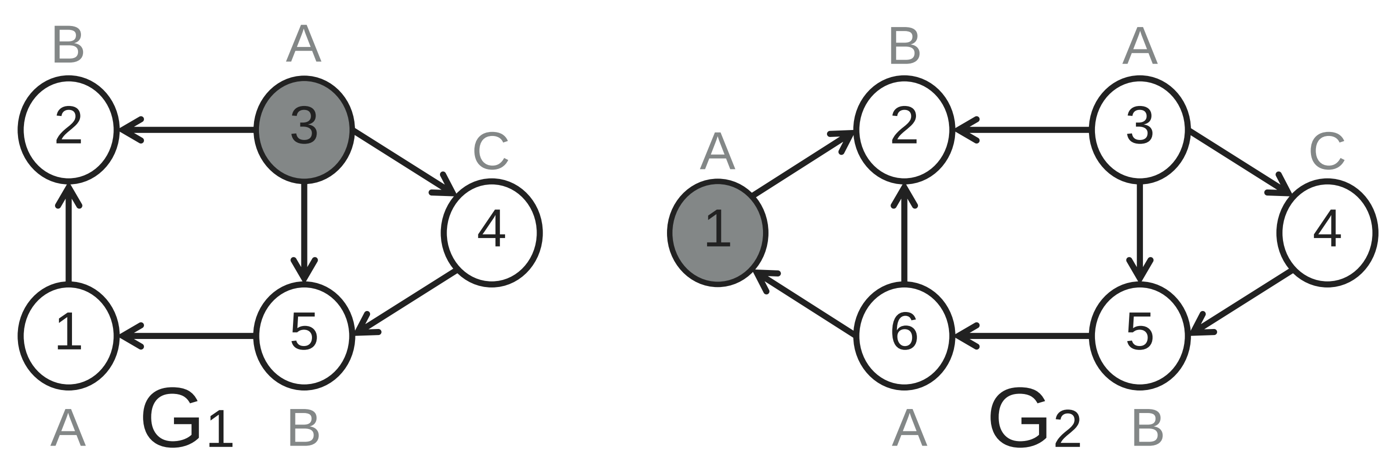
State	Mapping
S_0	$M(S_0)=\{\}$
$S_?$	$M(S_?)=\{(3,1)\}$

$$IsFeasible(s_0,3,1) = F_s(s_0,3,1) \wedge F_t(s_0,3,1)$$

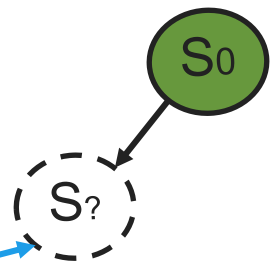
$$F_t(s_0,3,1) = F_c(s_0,3,1) \wedge F_{la1}(s_0,3,1) \wedge F_{la2}(s_0,3,1)$$

Ng = {3,5,4,2,1}





VF3: EXAMPLE



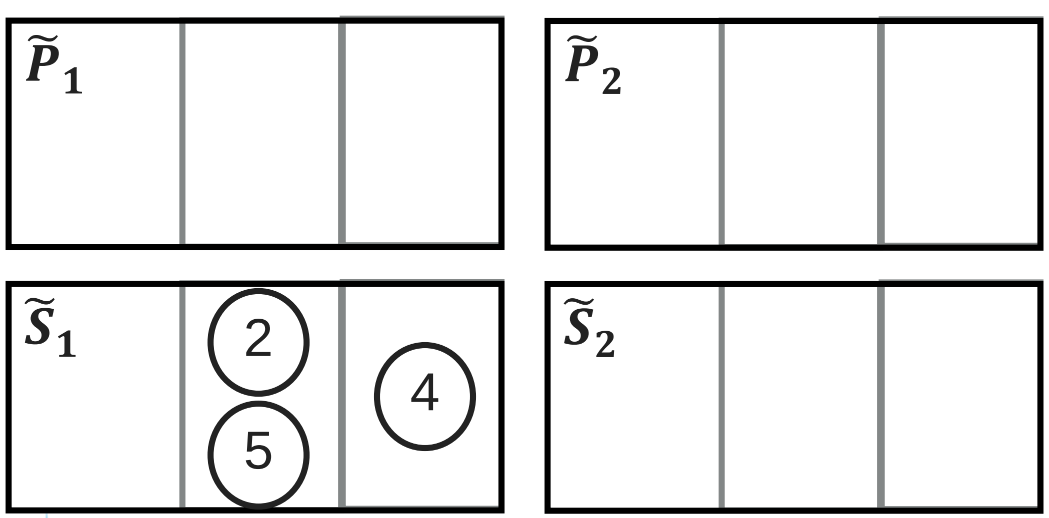
Feasibility Check for $S_0 \cup (3,1)$

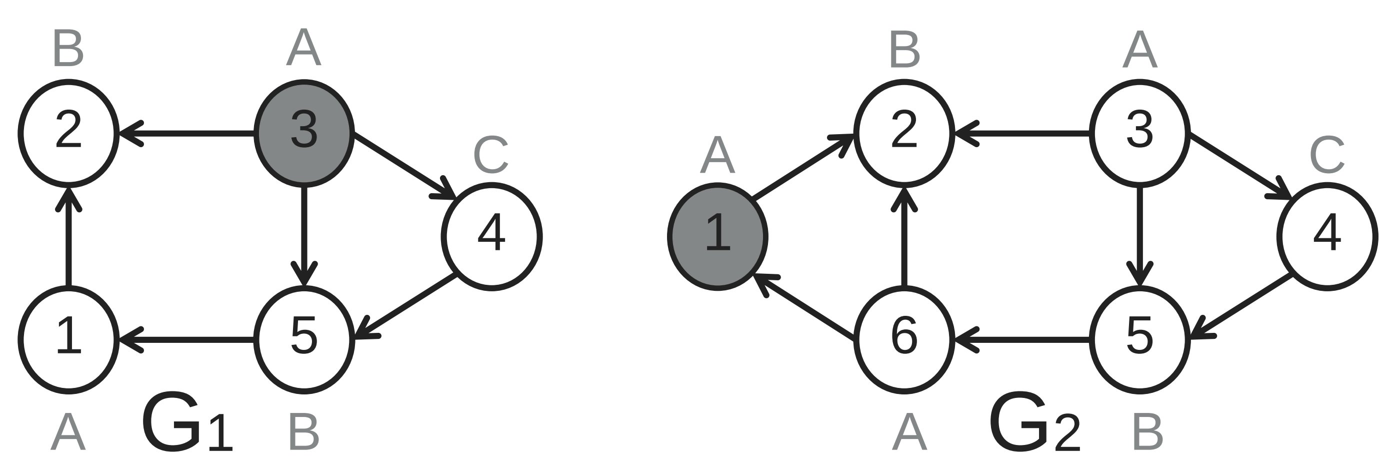
State	Mapping
S_0	$M(S_0)=\{\}$
$S_?$	$M(S_?)=\{(3,1)\}$

$IsFeasible(s_0,3,1) = F_s(s_0,3,1) \wedge F_t(s_0,3,1)$

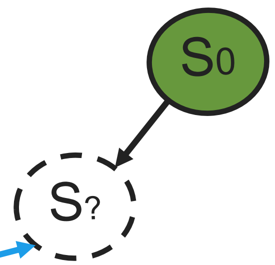
$F_t(s_0,3,1) = F_c(s_0,3,1) \wedge F_{la1}(s_0,3,1) \wedge F_{la2}(s_0,3,1)$

Ng = {3,5,4,2,1}





VF3: EXAMPLE



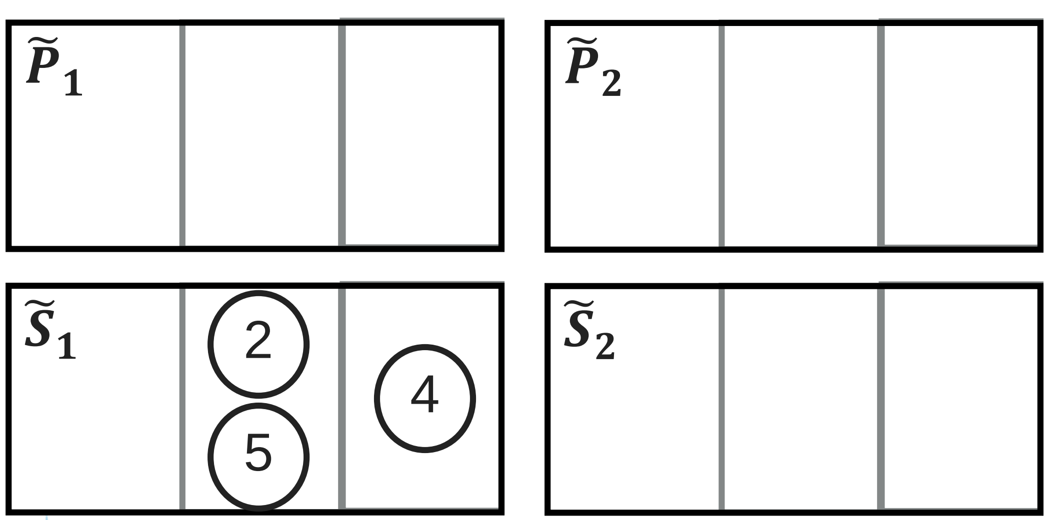
State	Mapping
S_0	$M(S_0)=\{\}$
$S_?$	$M(S_?)=\{(3,3)\}$

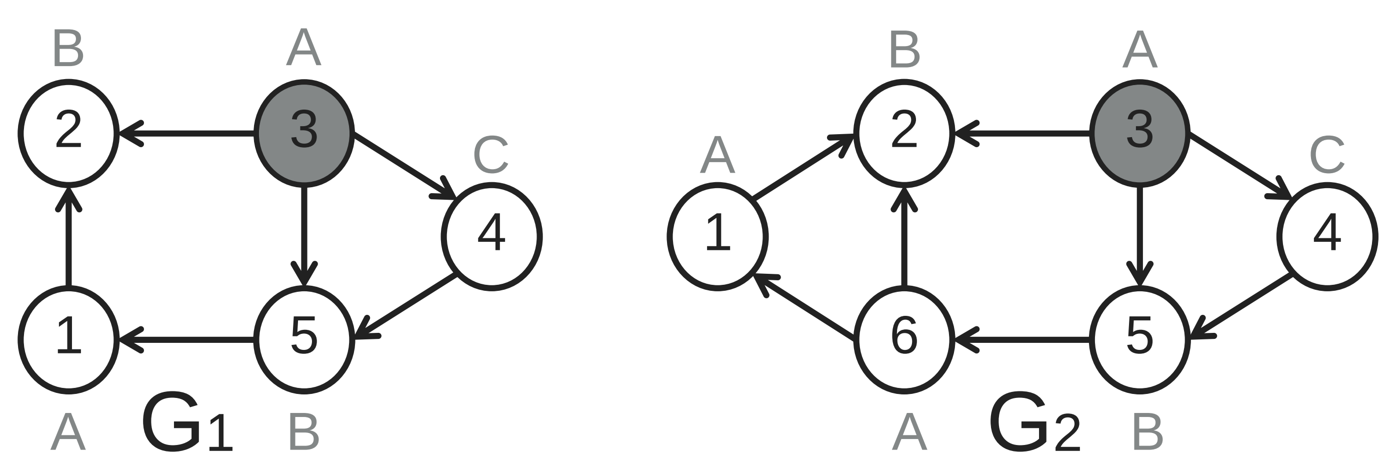
Feasibility Check for S_0 U (3,3)

$IsFeasible(s_0,3,3) = F_s(s_0,3,3) \wedge F_t(s_0,3,3)$

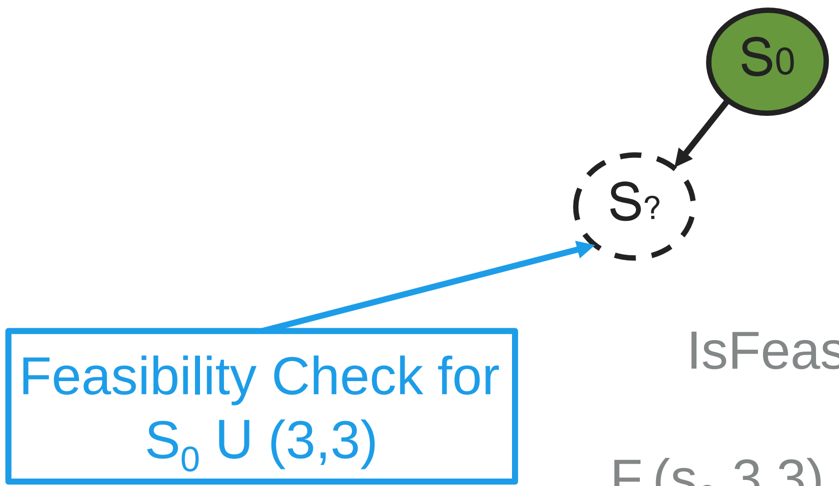
$F_t(s_0,3,3) = F_c(s_0,3,3) \wedge F_{la1}(s_0,3,3) \wedge F_{la2}(s_0,3,3)$

Ng = {3,5,4,2,1}





VF3: EXAMPLE

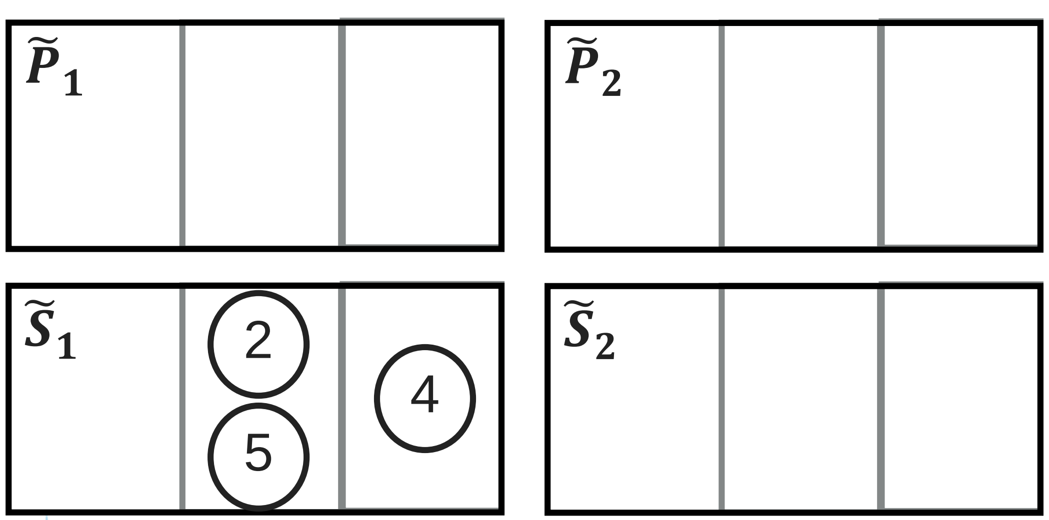


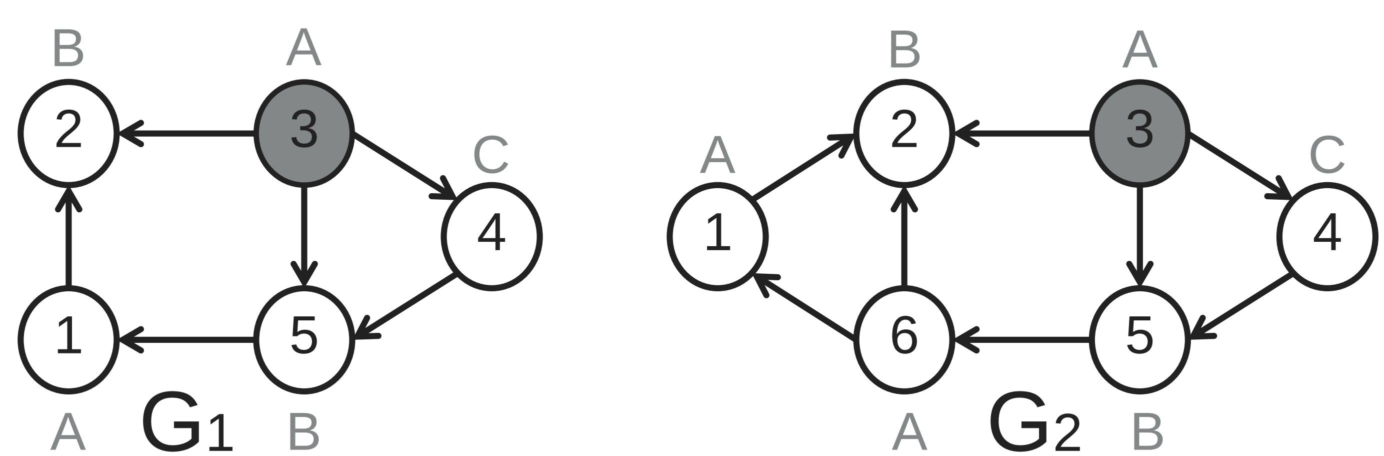
State	Mapping
S_0	$M(S_0)=\{\}$
$S_?$	$M(S_?)=\{(3,3)\}$

$IsFeasible(s_0,3,3) = F_s(s_0,3,3) \wedge F_t(s_0,3,3)$ ✔ ✔

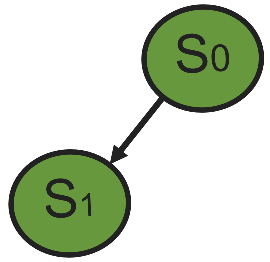
$F_t(s_0,3,3) = F_c(s_0,3,3) \wedge F_{la1}(s_0,3,3) \wedge F_{la2}(s_0,3,3)$ ✔ ✔ ✔

Ng = {3,5,4,2,1}



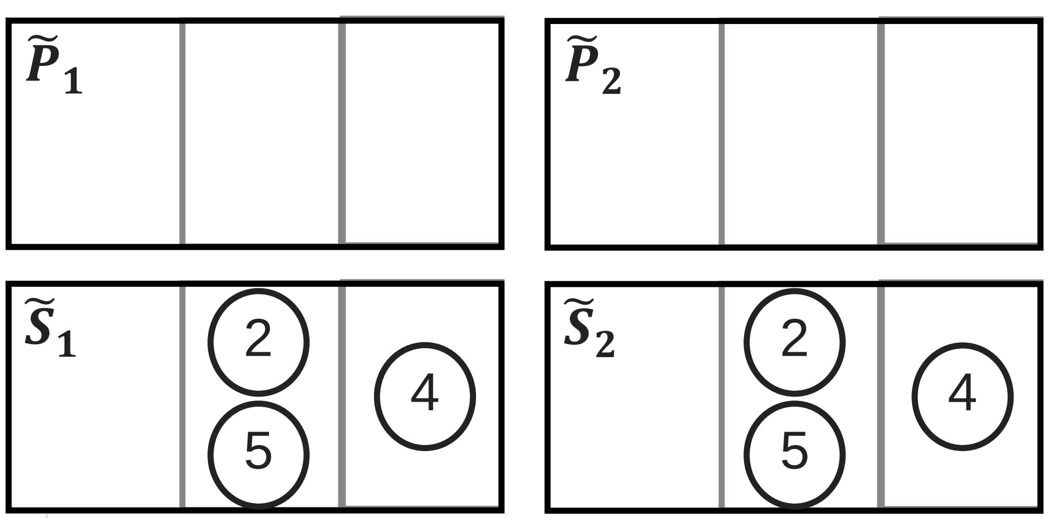


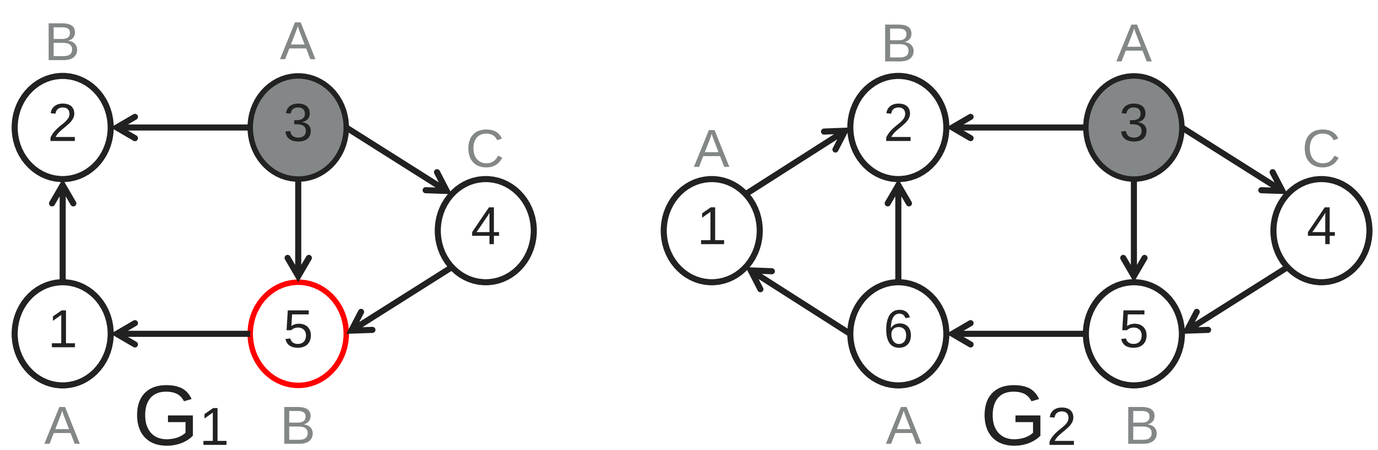
VF3: EXAMPLE



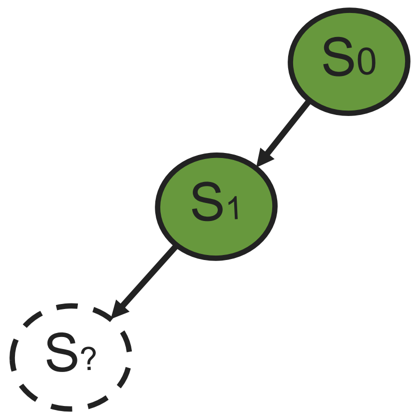
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$

Ng = {3,5,4,2,1}





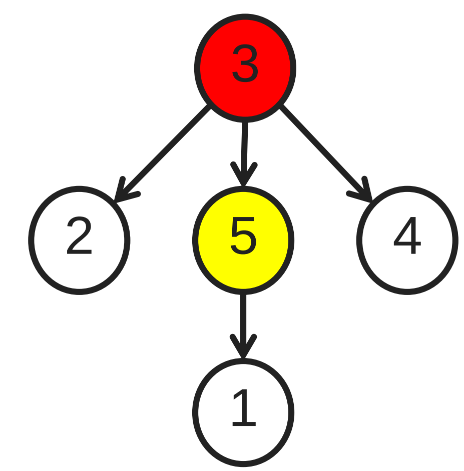
VF3: EXAMPLE



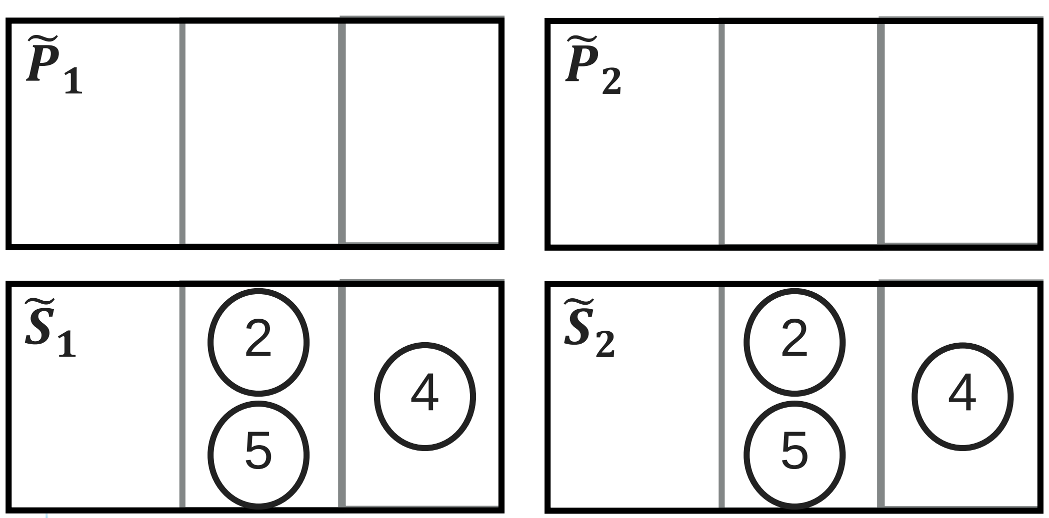
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$
$S_?$	$M(S_?)=\{(3,3), (5,?)\}$

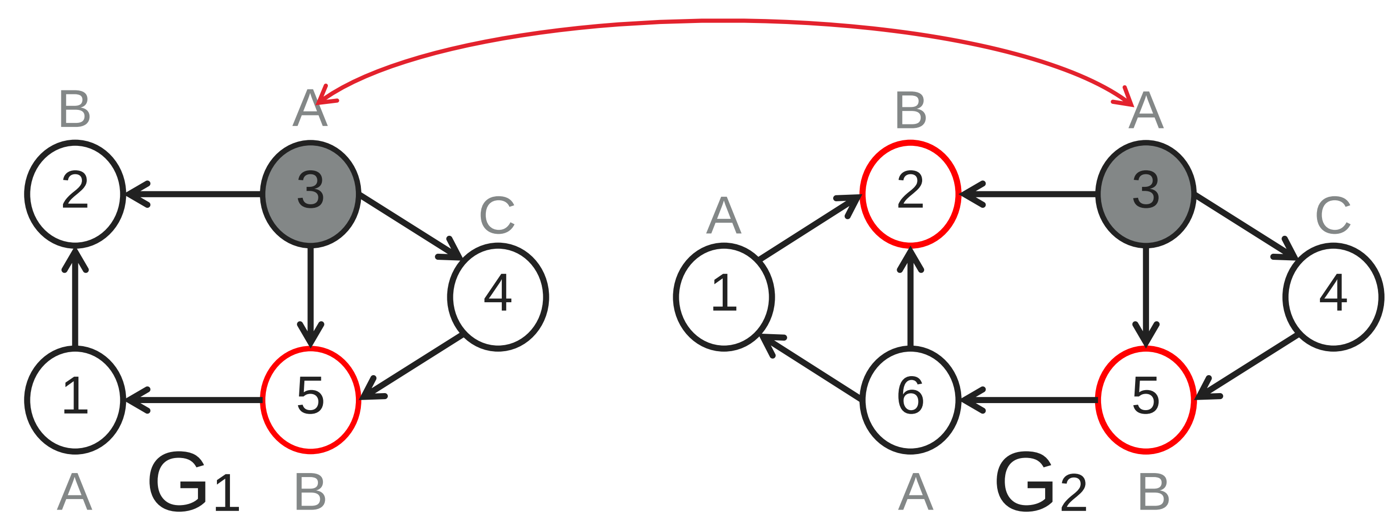
- The next candidate of G1 is given by Ng: 5
- The candidates for G2 are taken by considering the neighbours of the node mapped to the parent of 5 (3 in the example) and belonging to the same class of 5
  - Terminal sets are not considered to this purpose

Parent tree

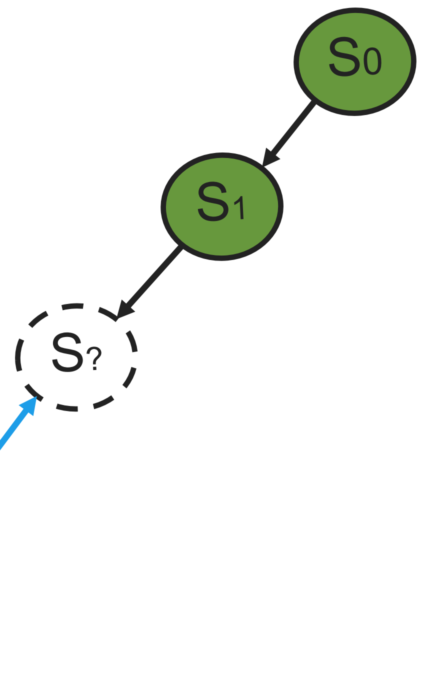


Ng = {3,5,4,2,1}





VF3: EXAMPLE



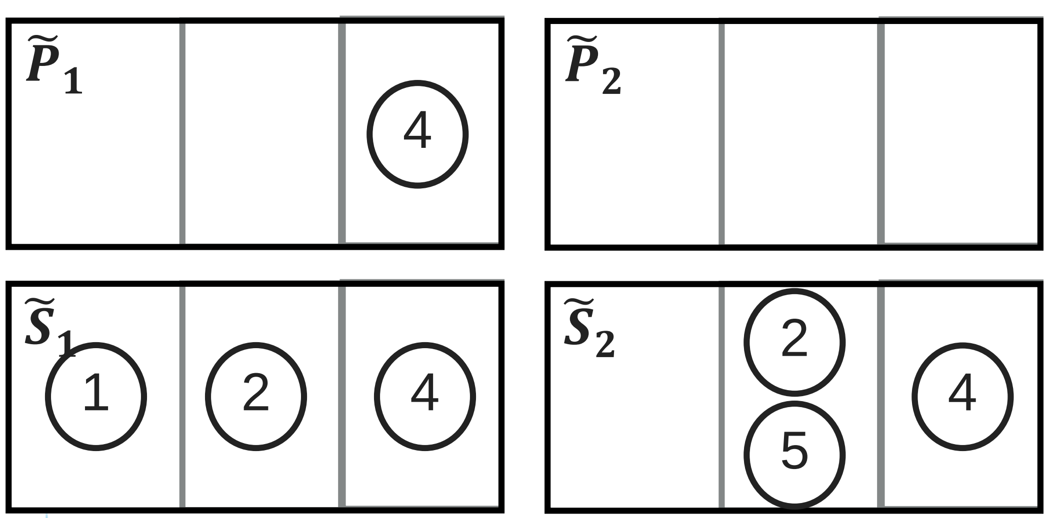
Feasibility Check for
S_1 U (5,2)

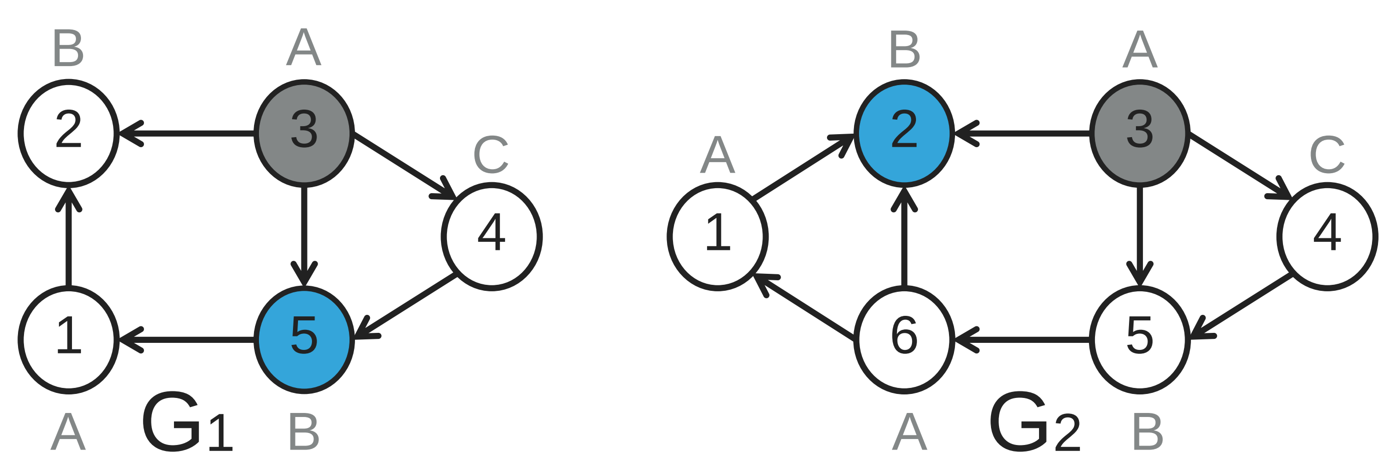
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$
$S_?$	$M(S_?)=\{(3,3), (5,2)\}$

IsFeasible(s_1,5,2) = F_t(s_1,5,2)∧ F_{la}(s_1,5,2)

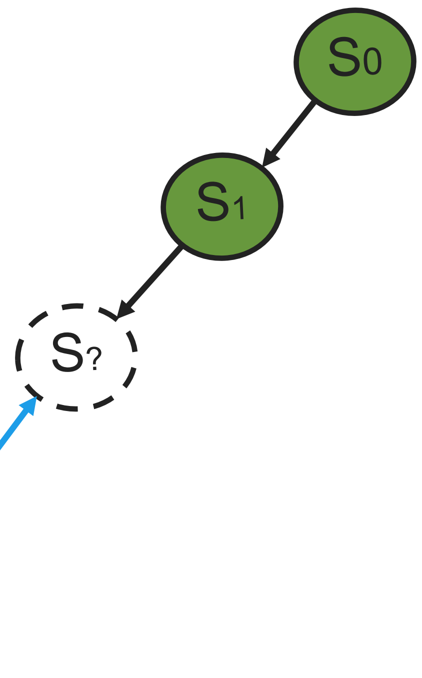
F_t(s_1,5,2) = F_c(s_1,5,2) ∧ F_{la1}(s_1,5,2) ∧ F_{la2}(s_1,5,2)

Ng = {3,5,4,2,1}





VF3: EXAMPLE



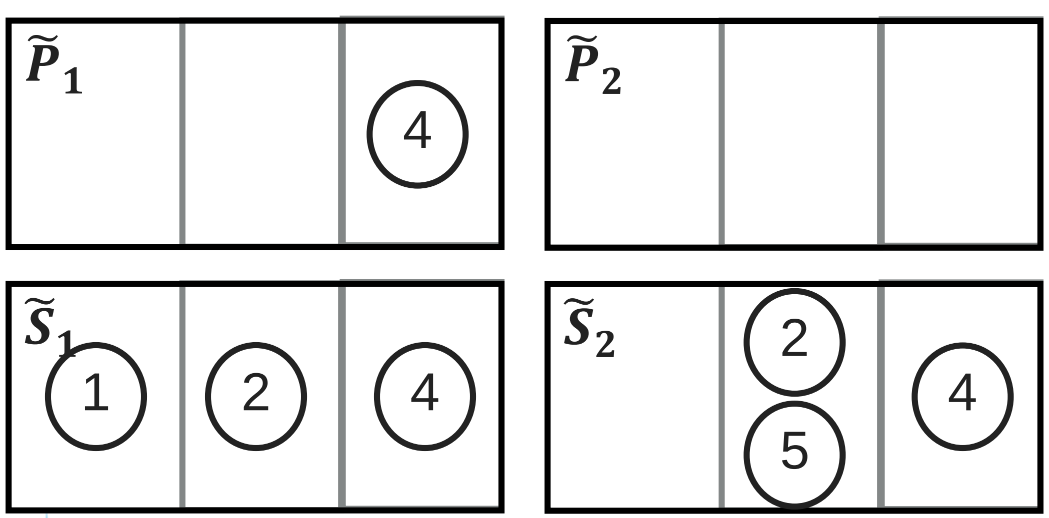
Feasibility Check for
S_1 U (5,5)

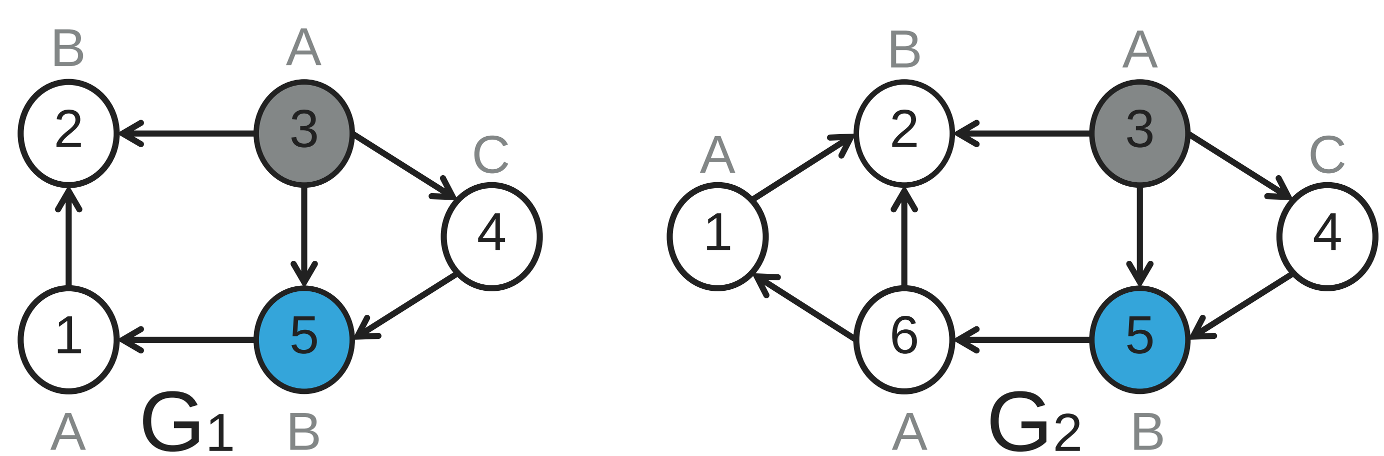
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$
$S_?$	$M(S_?)=\{(3,3), (5,5)\}$

$IsFeasible(s_1,5,5) = F_s(s_1,5,5) \wedge F_t(s_1,5,5)$

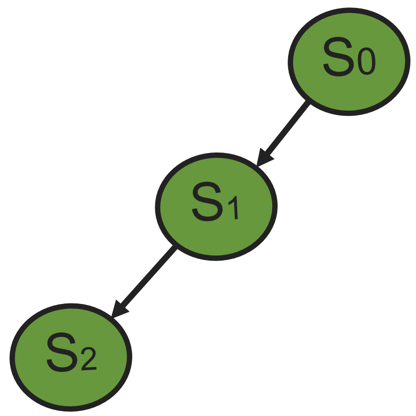
$F_t(s_1,5,5) = F_c(s_1,5,5) \wedge F_{la1}(s_1,5,5) \wedge F_{la}(s_1,5,5)$

Ng = {3,5,4,2,1}



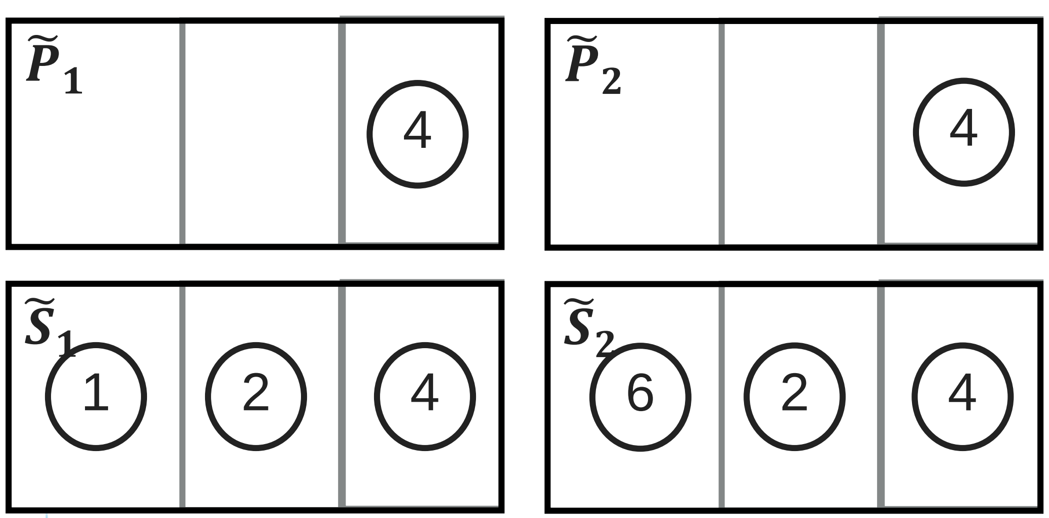


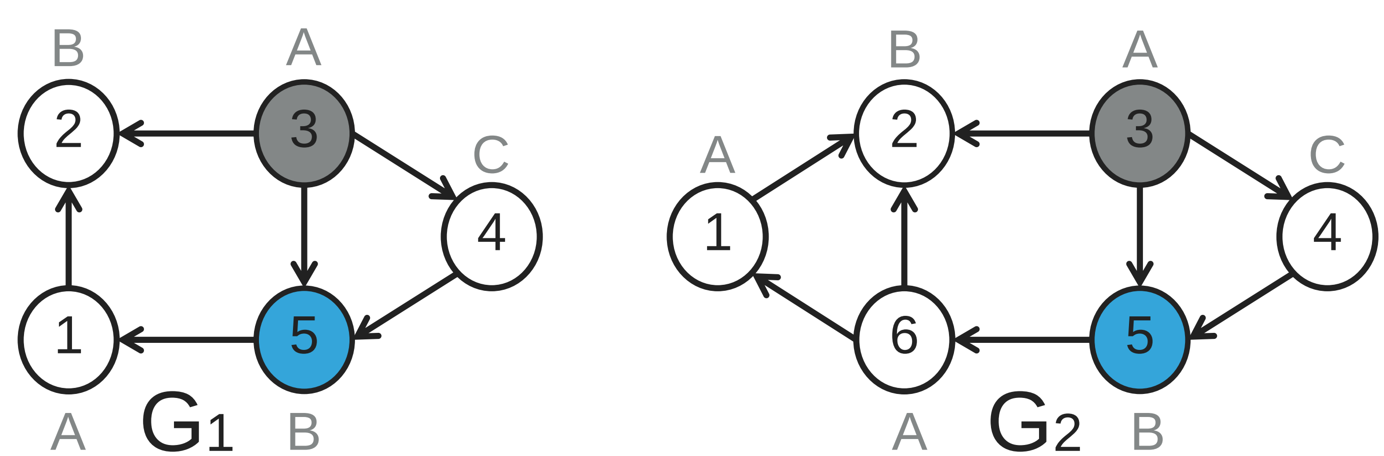
VF3: EXAMPLE



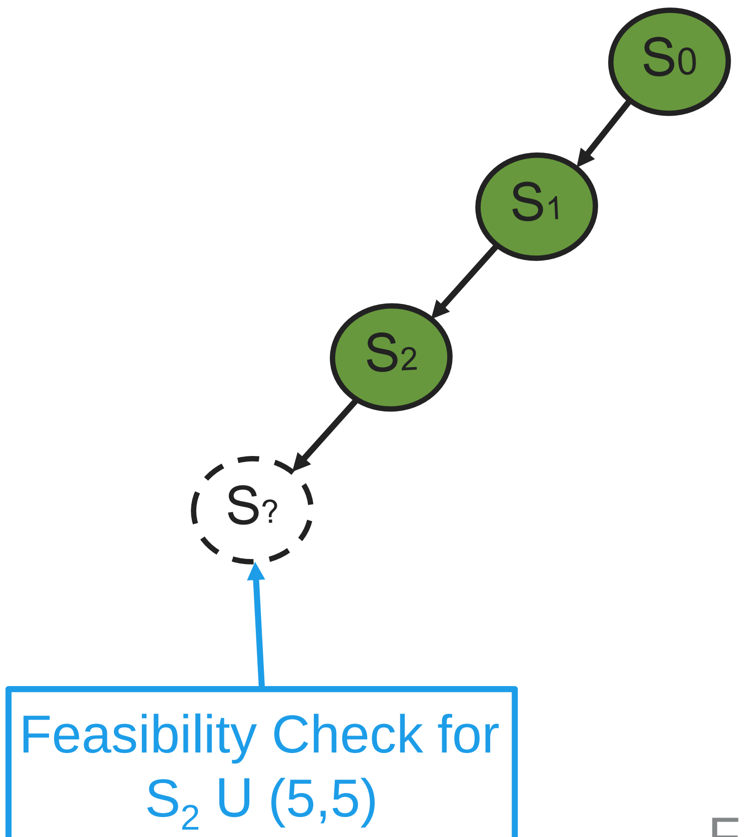
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$
S_2	$M(S_2)=\{(3,3), (5,5)\}$

Ng = {3,5,4,2,1}





VF3: EXAMPLE

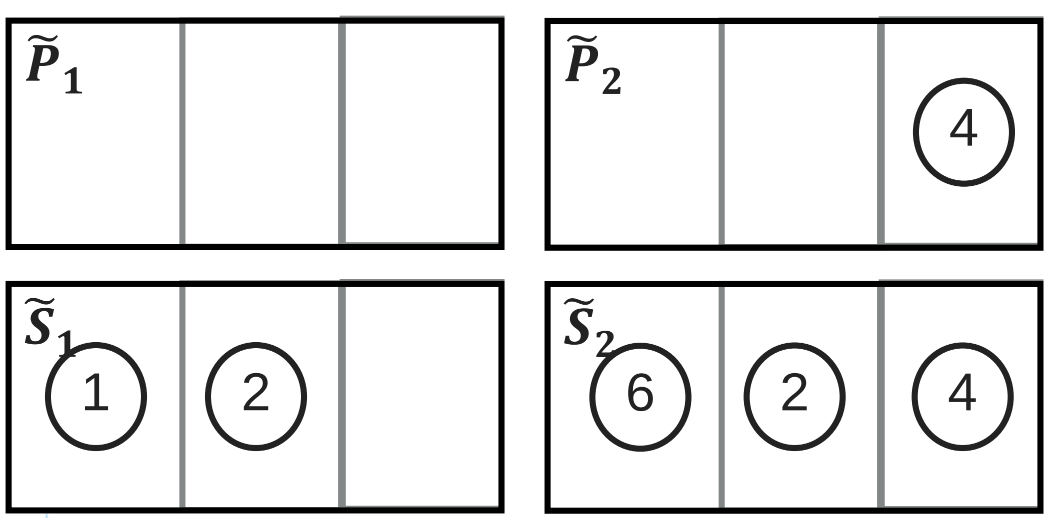


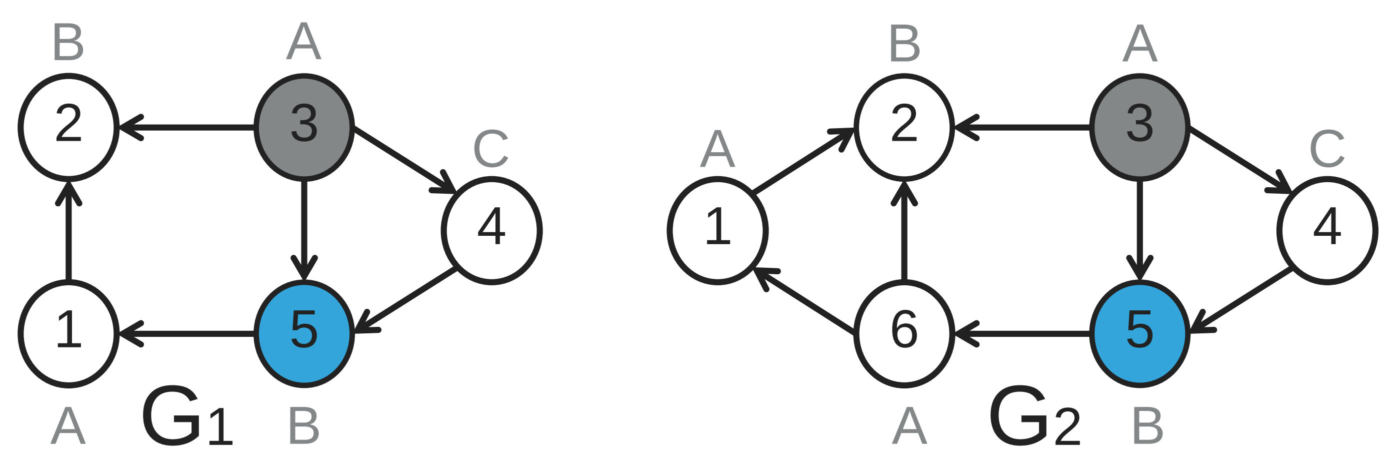
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$
S_2	$M(S_2)=\{(3,3), (5,5)\}$
$S_?$	$M(S_?)=\{(3,3), (5,5), (4,4)\}$

$IsFeasible(s_2,5,5) = F_s(s_2,5,5) \wedge F_t(s_2,5,5)$

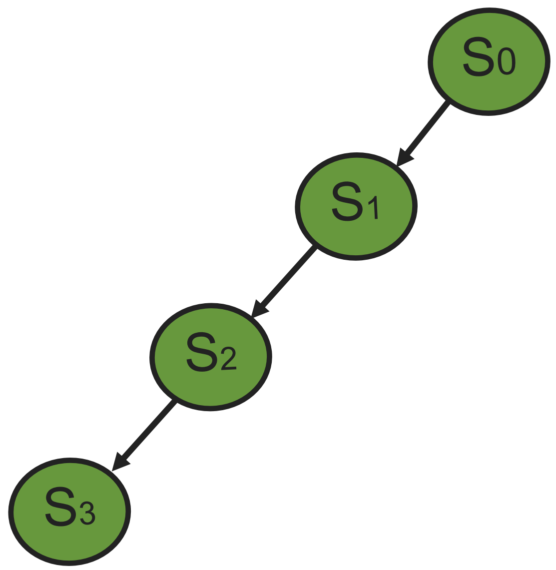
$F_t(s_2,5,5) = F_c(s_2,5,5) \wedge F_{la1}(s_2,5,5) \wedge F_{la}(s_2,5,5)$

Ng = {3,5,4,2,1}



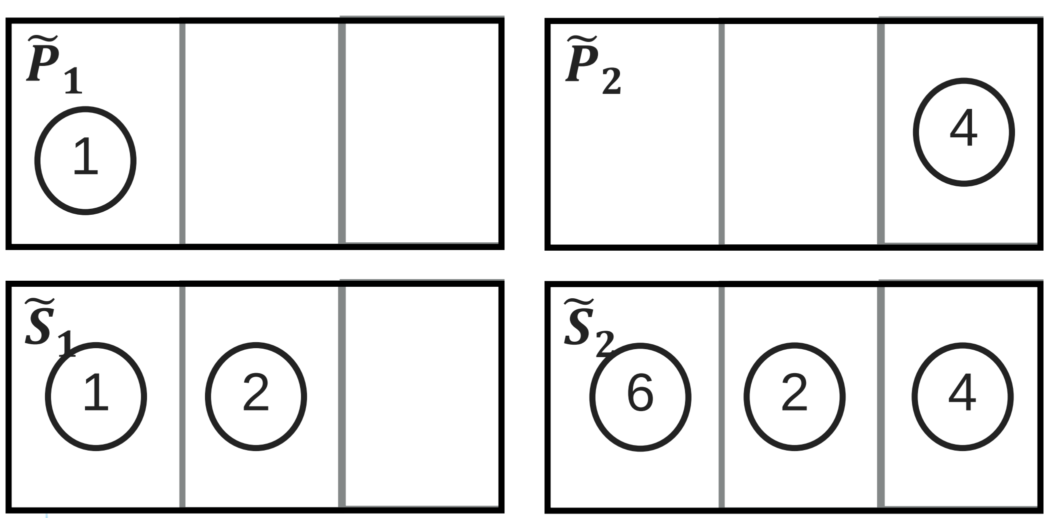


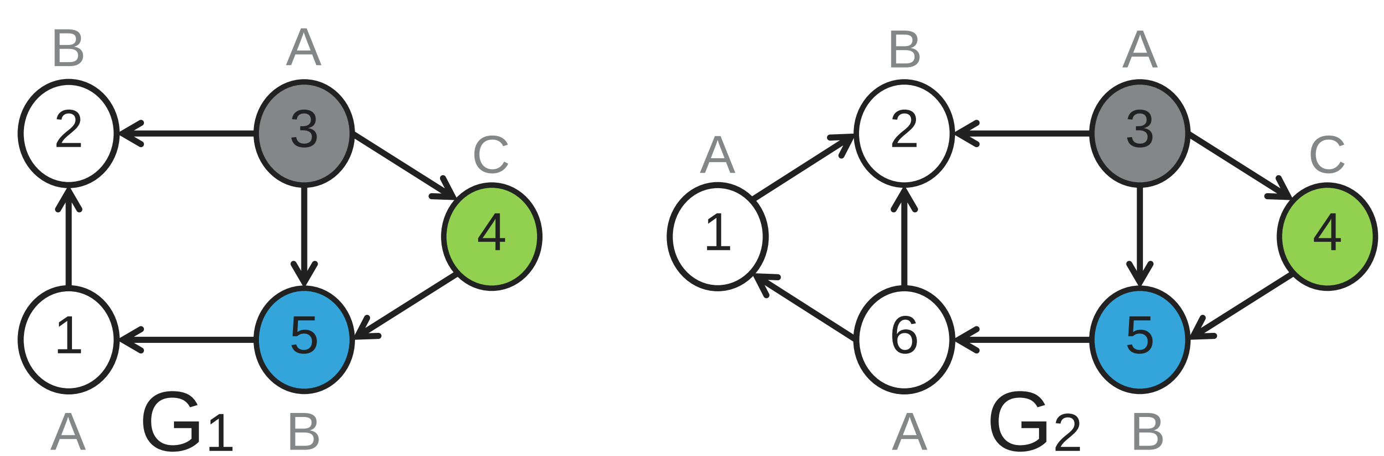
VF3: EXAMPLE



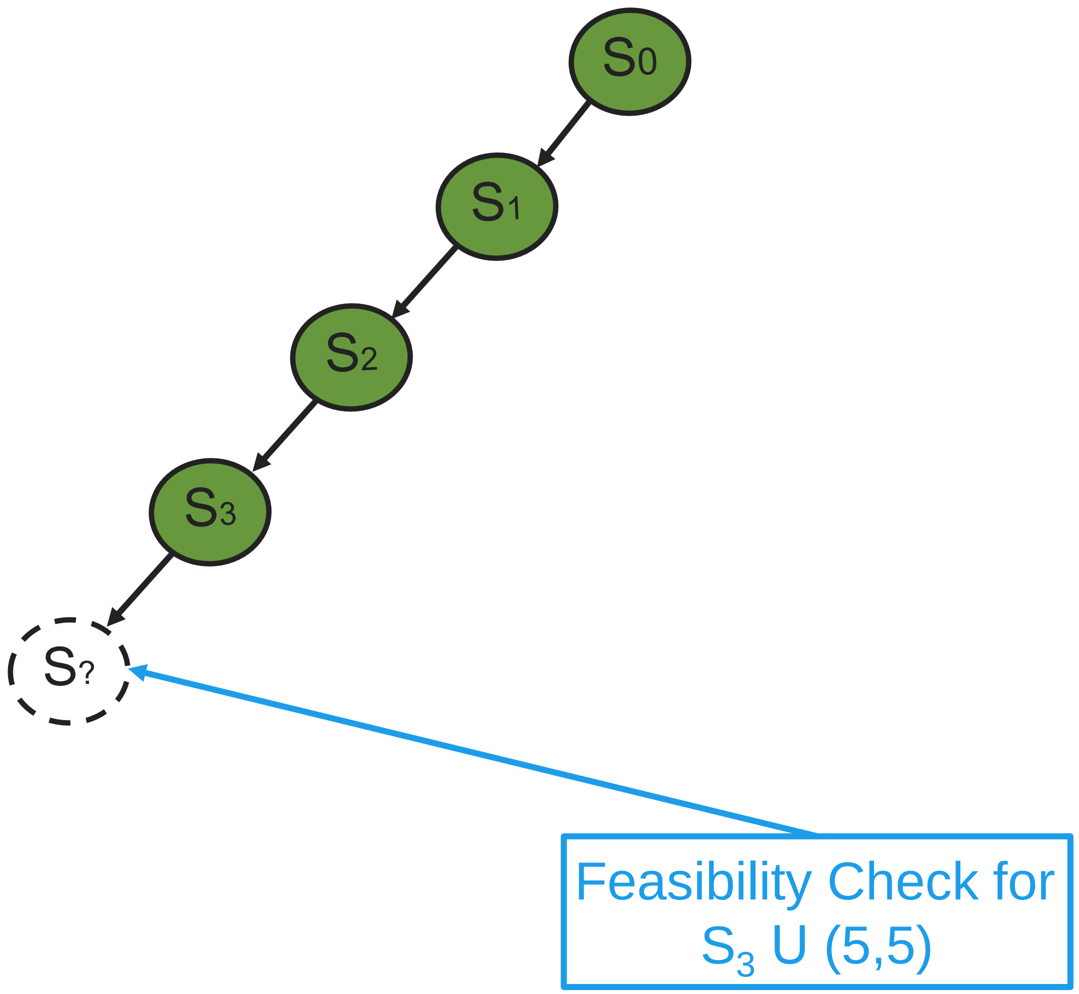
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$
S_2	$M(S_2)=\{(3,3), (5,5)\}$
S_3	$M(S_3)=\{(3,3), (5,5), (4,4)\}$

Ng = {3,5,4,2,1}





VF3: EXAMPLE

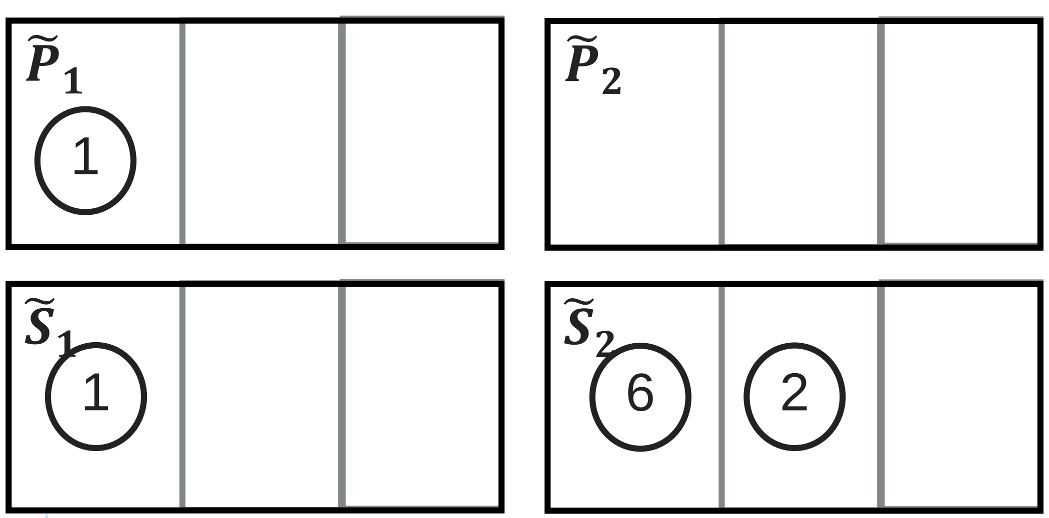


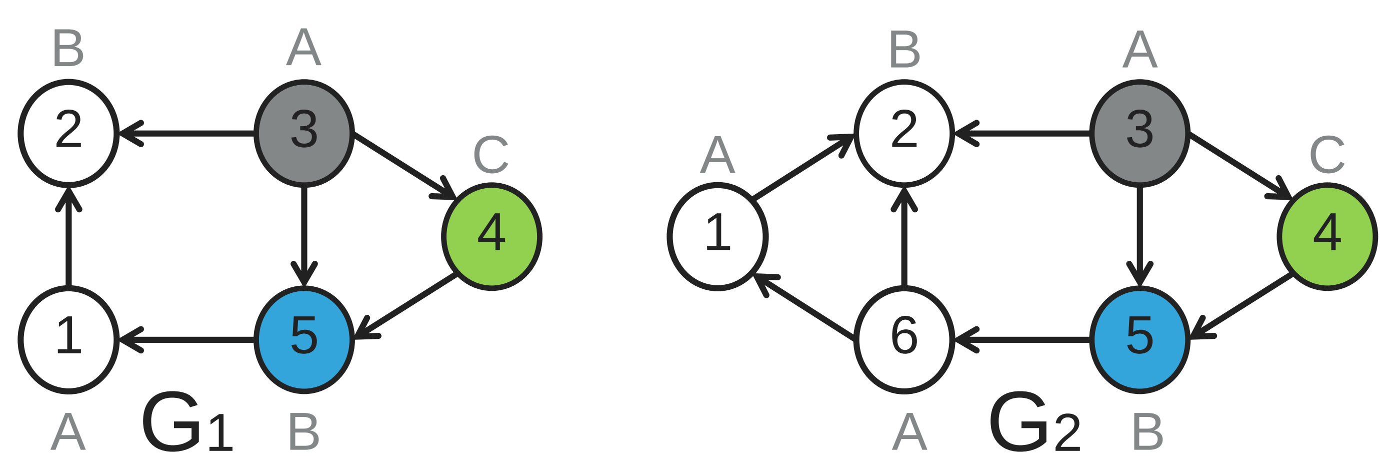
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$
S_2	$M(S_2)=\{(3,3), (5,5)\}$
S_3	$M(S_3)=\{(3,3), (5,5), (4,4)\}$
$S_?$	$M(S_?)=\{(3,3), (5,5), (4,4), (2,2)\}$

$IsFeasible(s_3,5,5) = F_s(s_3,5,5) \wedge F_t(s_3,5,5)$

$F_t(s_3,5,5) = F_c(s_3,5,5) \wedge F_{la1}(s_3,5,5) \wedge F_{la}(s_3,5,5)$

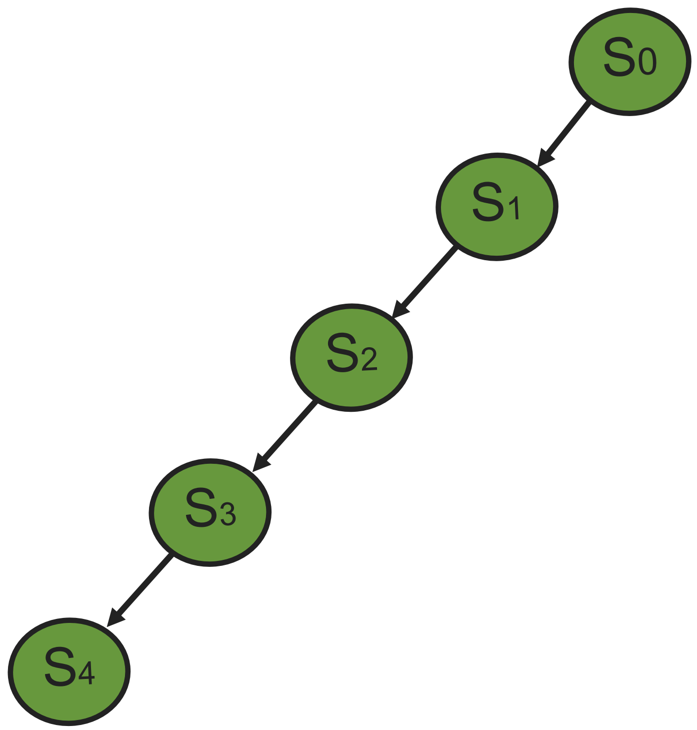
Ng = {3,5,4,2,1}





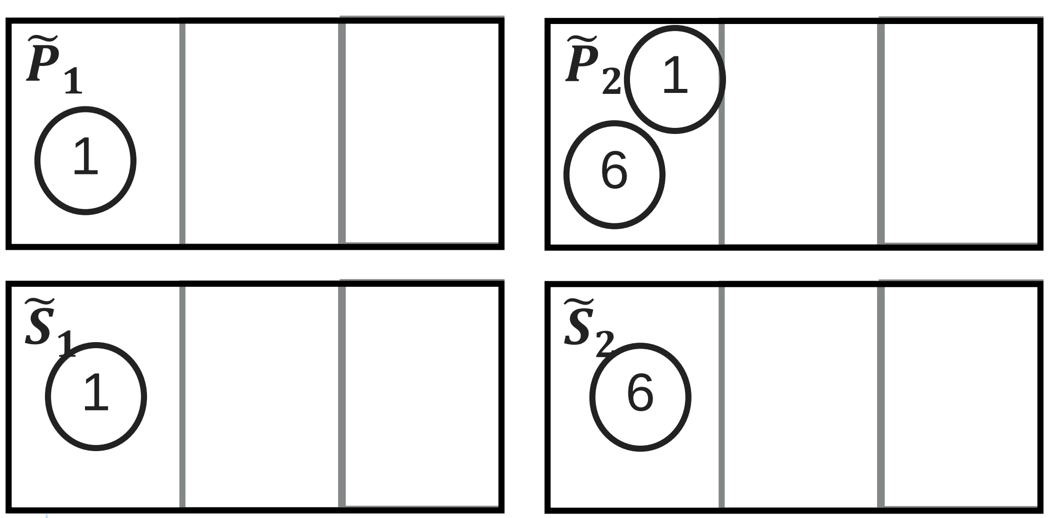
MIVIA

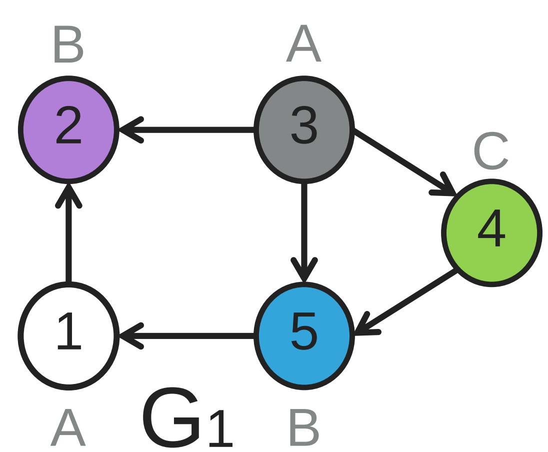
VF3: EXAMPLE

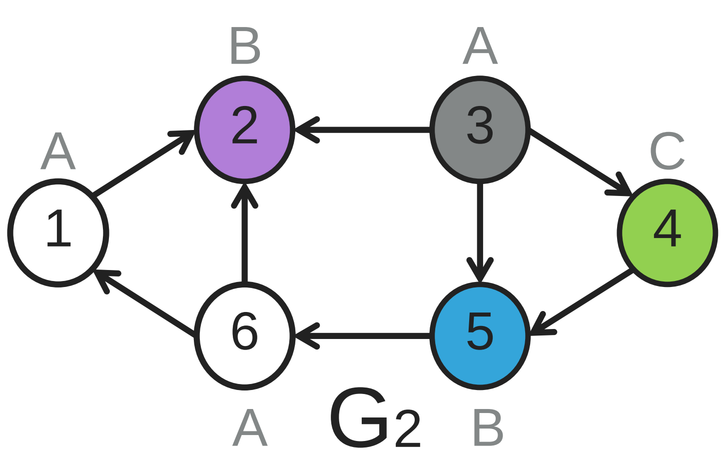


State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$
S_2	$M(S_2)=\{(3,3), (5,5)\}$
S_3	$M(S_3)=\{(3,3), (5,5), (4,4)\}$
S_4	$M(S_4)=\{(3,3), (5,5), (4,4), (2,2)\}$

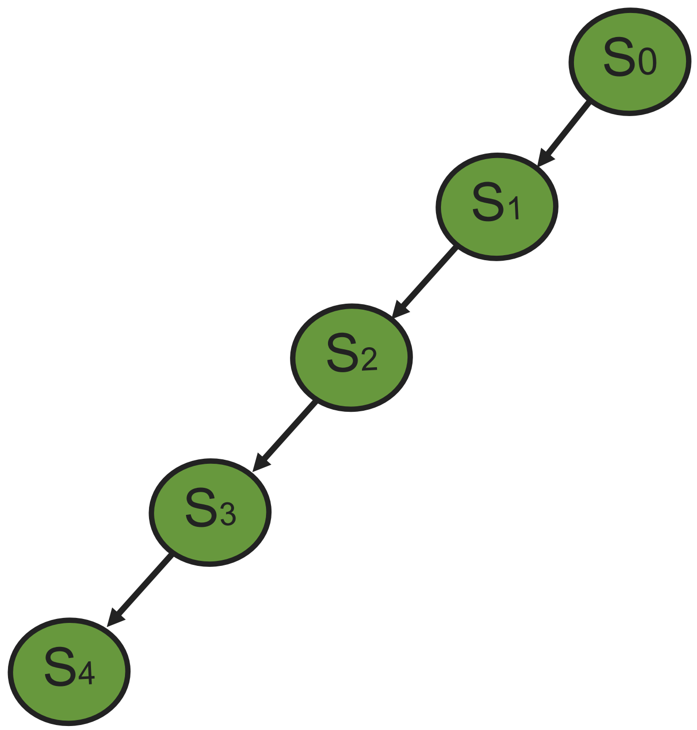
Ng = {3,5,4,2,1}





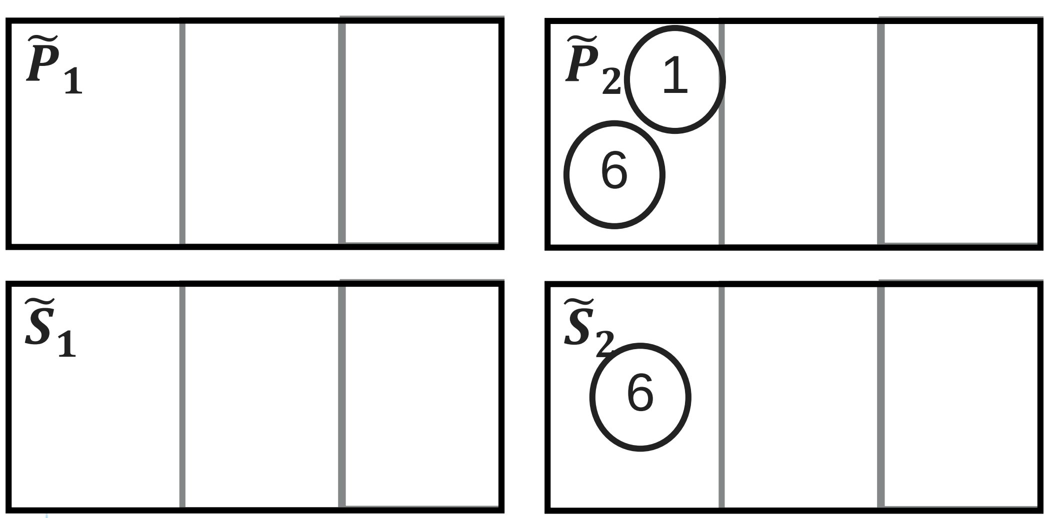


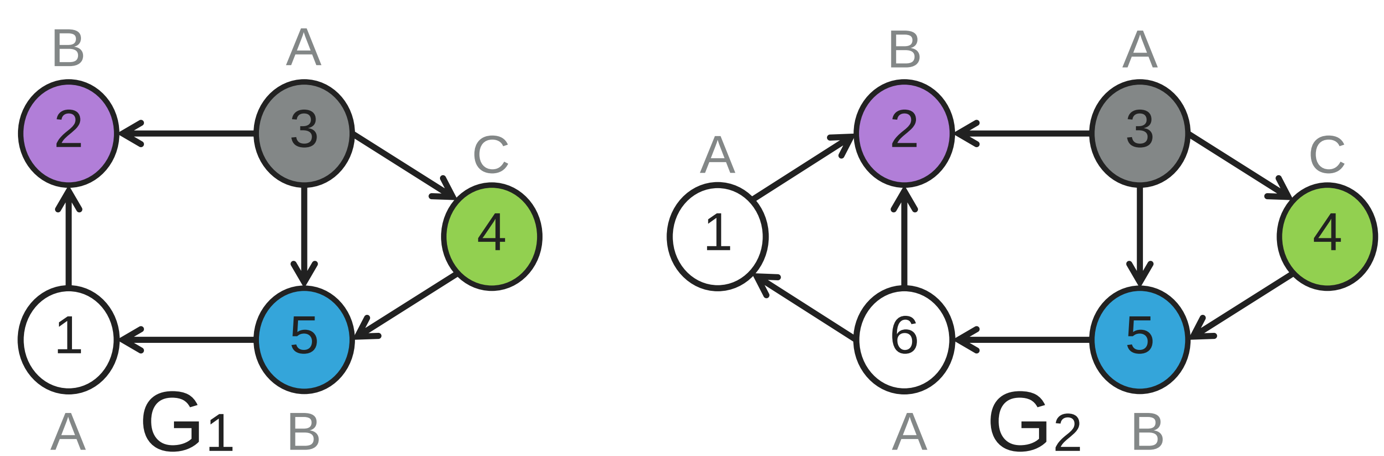
VF3: EXAMPLE



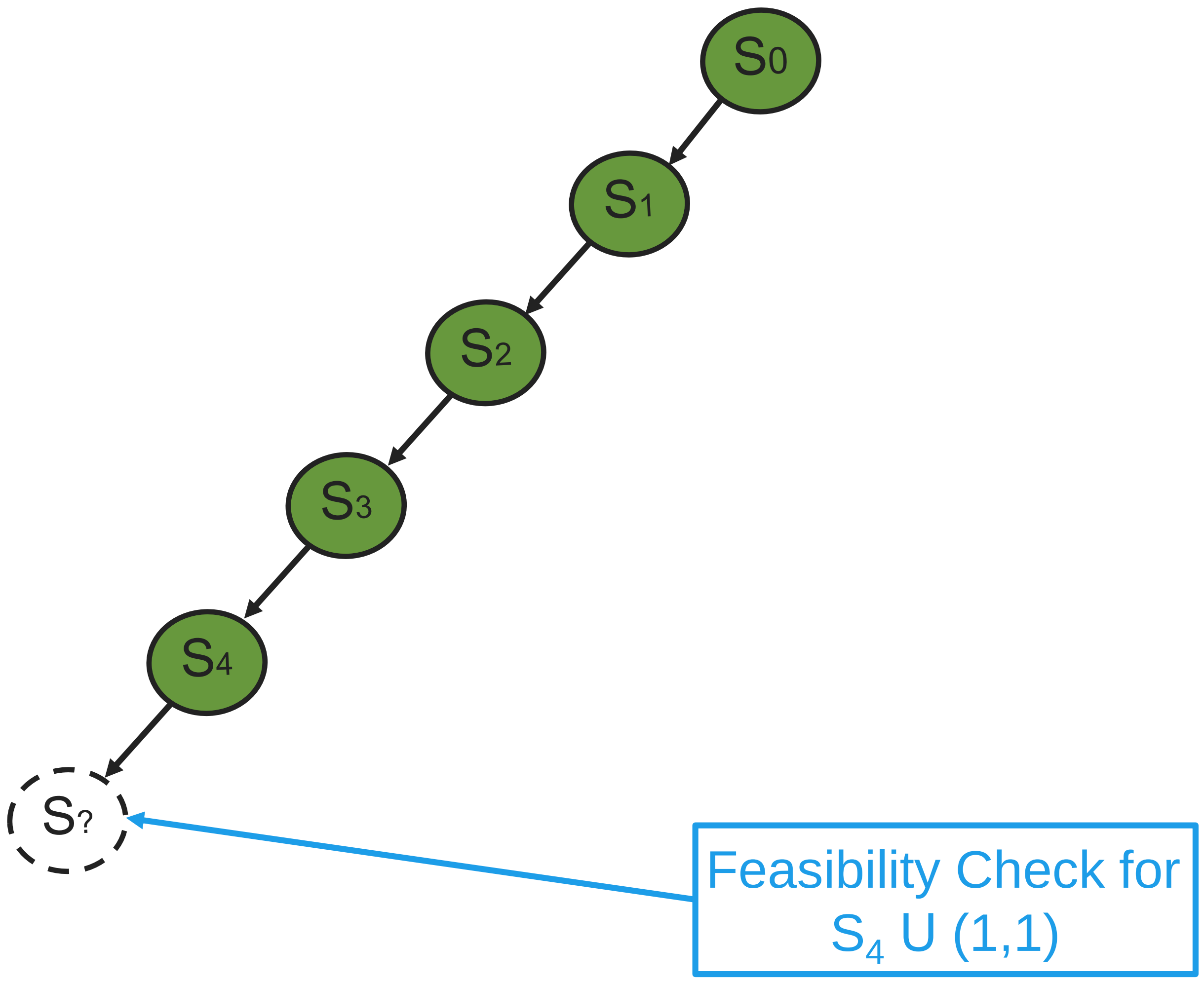
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$
S_2	$M(S_2)=\{(3,3), (5,5)\}$
S_3	$M(S_3)=\{(3,3), (5,5), (4,4)\}$
S_4	$M(S_4)=\{(3,3), (5,5), (4,4), (2,2)\}$

Ng = {3,5,4,2,1}





VF3: EXAMPLE

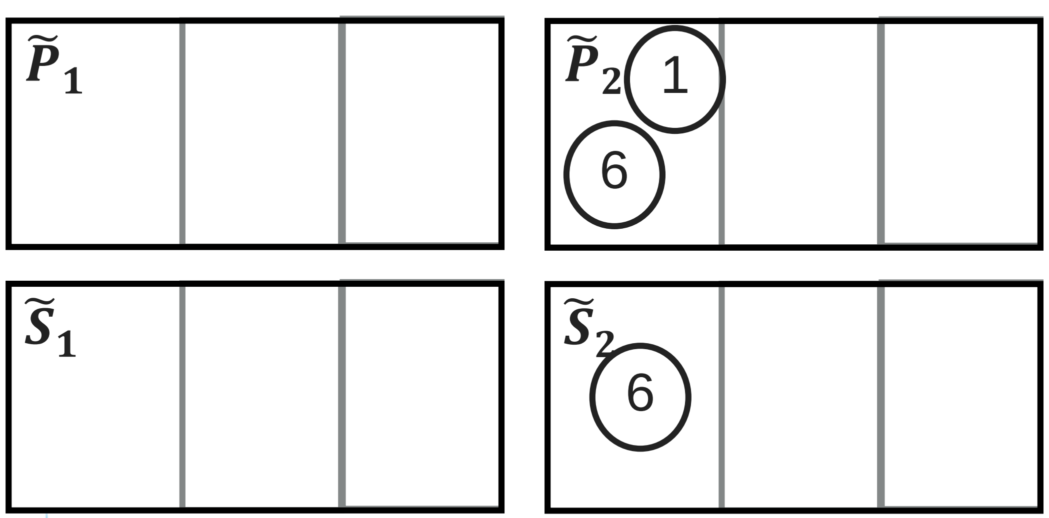


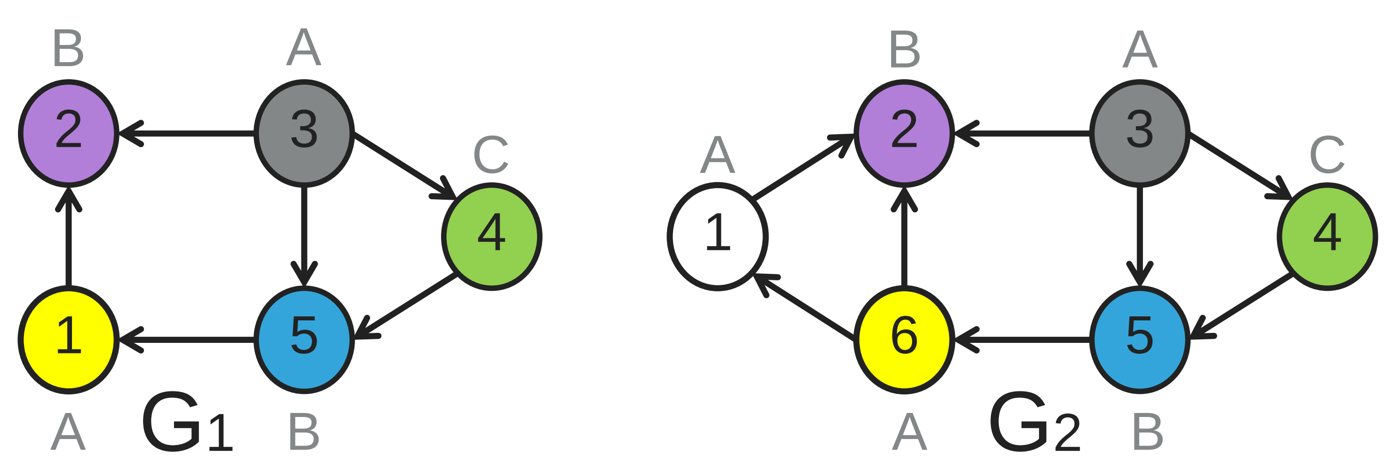
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$
S_2	$M(S_2)=\{(3,3), (5,5)\}$
S_3	$M(S_3)=\{(3,3), (5,5), (4,4)\}$
S_4	$M(S_4)=\{(3,3), (5,5), (4,4), (2,2)\}$
$S_?$	$M(S_?)=\{(3,3), (5,5), (4,4), (2,2), (1,1)\}$

$IsFeasible(s_4,1,1) = F_s(s_4,1,1) \wedge F_t(s_4,1,1)$

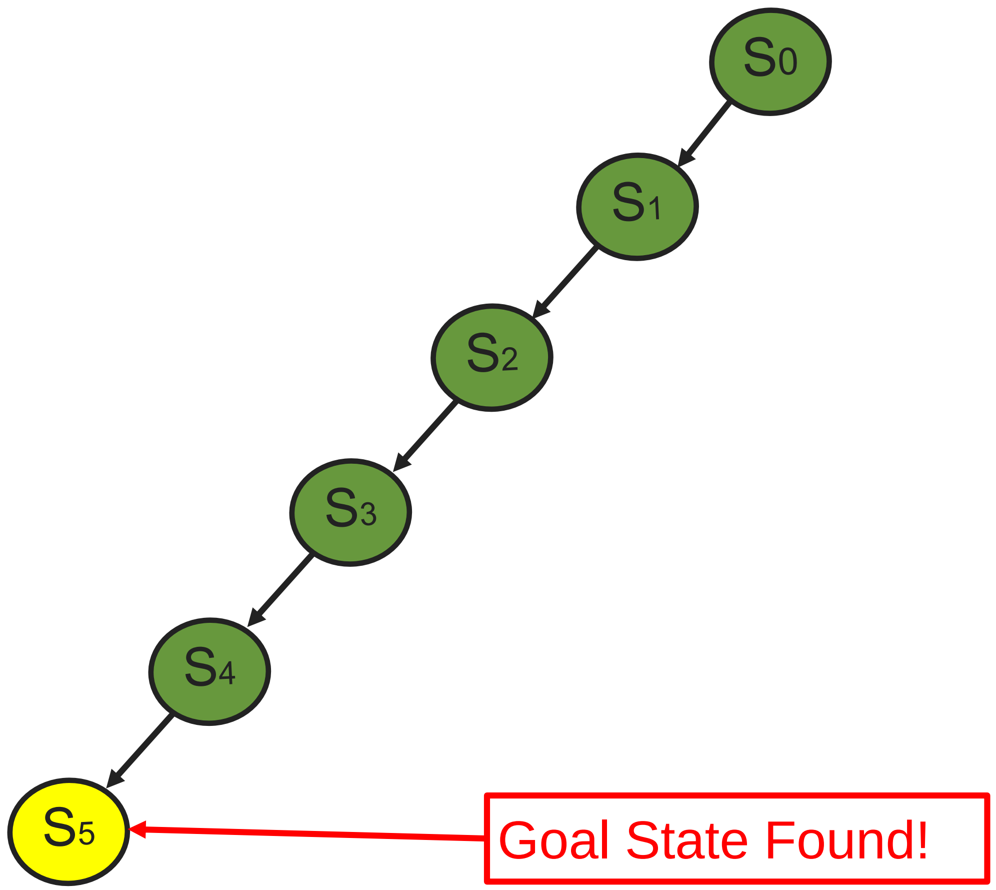
$F_t(s_4,1,1) = F_c(s_4,1,1) \wedge F_{la1}(s_4,1,1) \wedge F_{la}(s_4,1,1)$

Ng = {3,5,4,2,1}



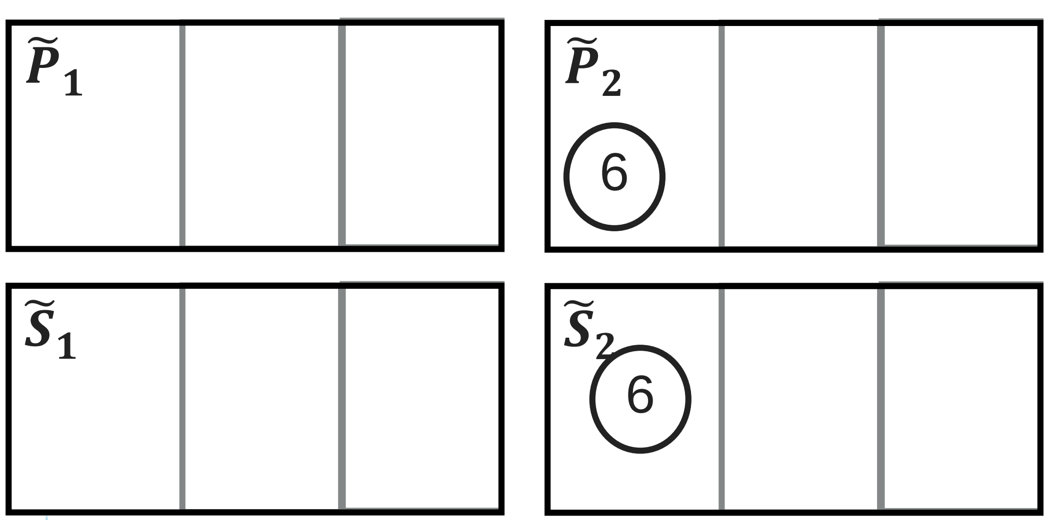


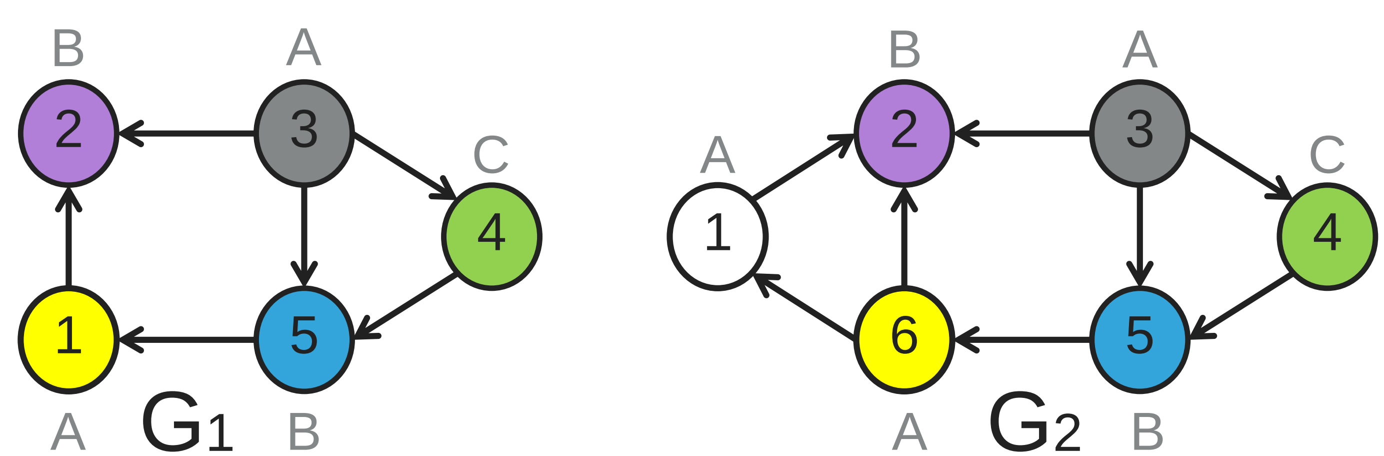
VF3: EXAMPLE



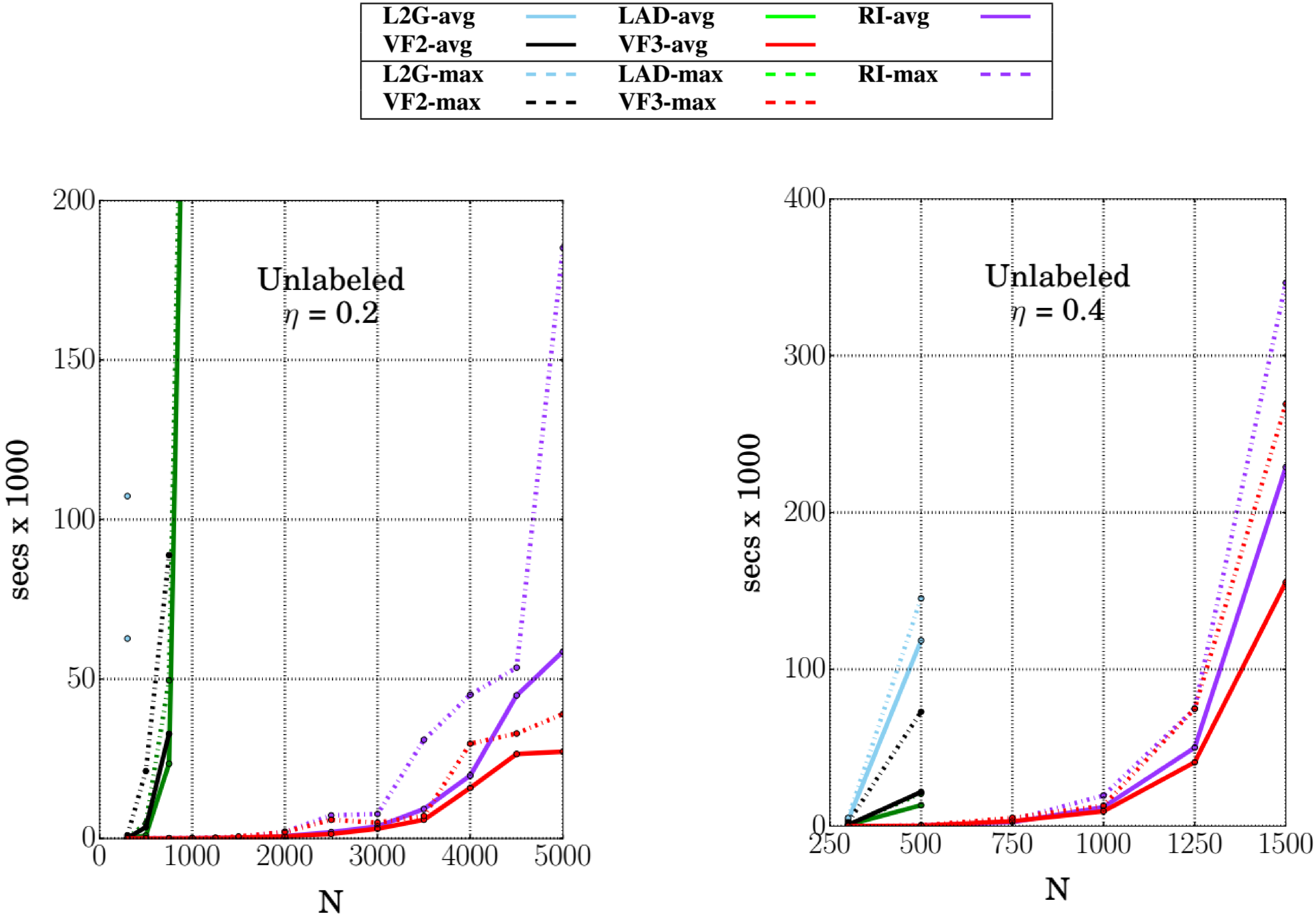
State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1)=\{(3,3)\}$
S_2	$M(S_2)=\{(3,3), (5,5)\}$
S_3	$M(S_3)=\{(3,3), (5,5), (4,4)\}$
S_4	$M(S_4)=\{(3,3), (5,5), (4,4), (2,2)\}$
S_5	$M(S_5)=\{(3,3), (5,5), (4,4), (2,2), (1,1)\}$

Ng = {3,5,4,2,1}

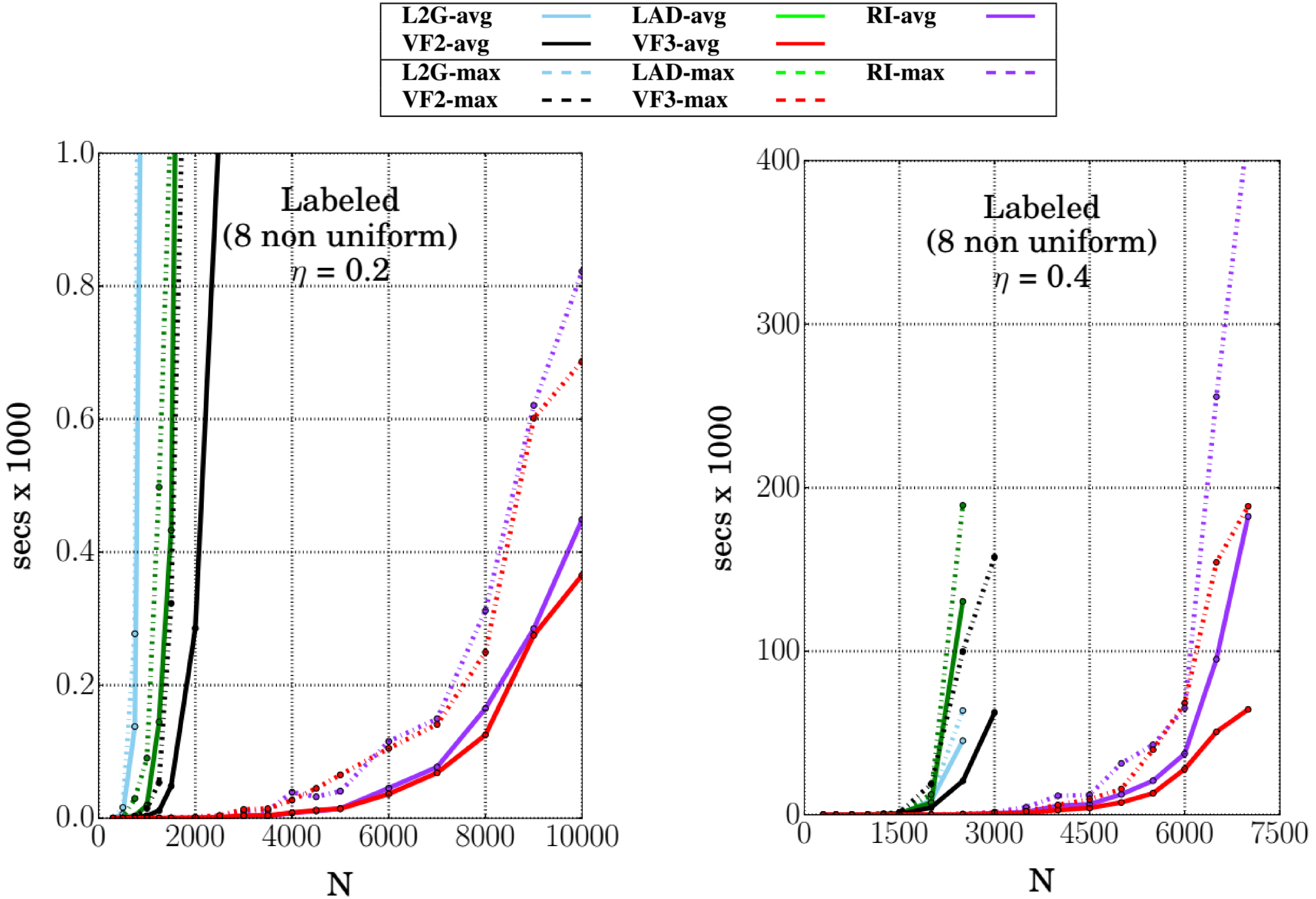




VF3 IN ACTION



VF3 IN ACTION



VF3 SUMMARY

- Linear memory complexity !
- Computation complexity is among $\Theta(n^{3.5})$ and $\Theta(n^{4.5})$ for Labeled Graphs and $\eta=0.2$ and $\eta=0.4$ respectively
- On the same graphs VF2 is $\Theta(n^{5.7})$

EGM: IS VF3 READY?

- The pruning procedure is costly, but allows to have polynomial times
- The efficiency increases as the asymmetry of the graphs is higher
- Sometimes (simple graphs) the cost of pruning overcomes advantages
- Uniformly Labeled Graphs with 10000 nodes and η=0.2 are matched in three minutes
- VF2 on UL-graphs η=0.2 matches graphs with 5000 nodes and η=0.2 in three hours
- VF3 on the same graphs takes 10 sec!
- We are ready for matching huge and dense graphs !

EGM: ARE WE READY?

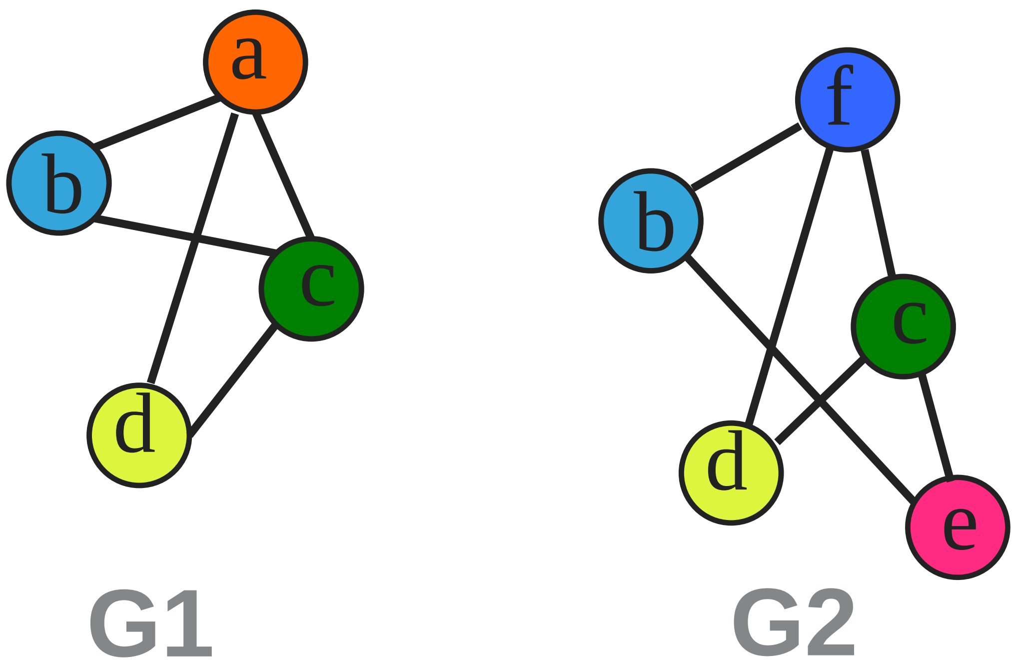
- Tree Search Methods:
  - RI/RI-DS: Giugno, Bonnici 2013 (very fast, especially simple graphs)
  - VF2 Plus: Vento,Foggia,Carletti 2015
  - VF3: Vento, Foggia, Carletti, Saggese 2017 (Fastest)
- Index Based Methods:
  - TurboIso: Han 2013
  - QuickSI: Shang 2008
- Constraint Satisfaction Methods:
  - LAD: Solnon 2010 (promising at the beginning wrt VF2)
- The top two performant methods are Tree Search Based: VF2 RI

EGM: SOME USEFUL LINKS

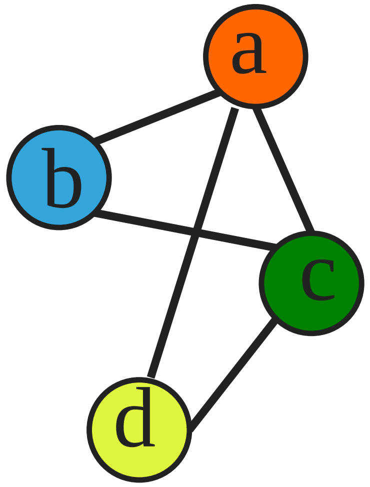
- IAPR TC15 Graph-based representations in Pattern Recognition
  http://iapr-tc15.greyc.fr
- A performance comparison of five algorithms for graph isomorphism
  http://mivia.unisa.it/wp-content/uploads/2013/05/foggia01.pdf
- A wide repository of graphs and GM algorithms
  http://mivia.unisa.it/datasets/graph-database
- Contest on graph matching algorithms for pattern search in biological databases see http://biograph2014.unisa.it

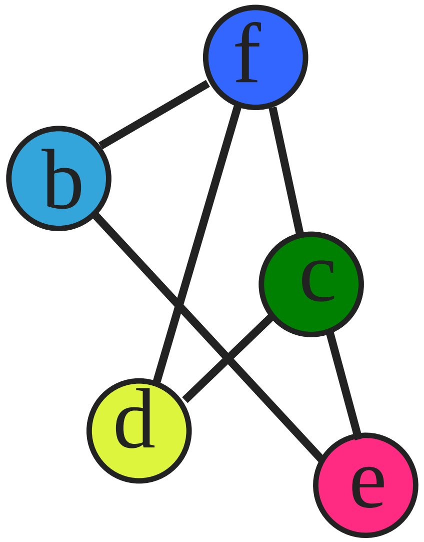
INEXACT GRAPH MATCHING

IGM: EXAMPLE



IGM: EXAMPLE

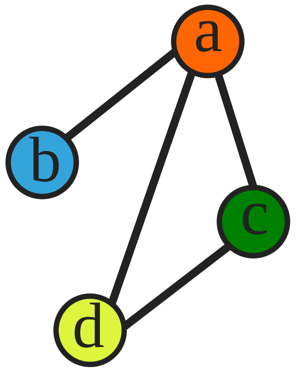




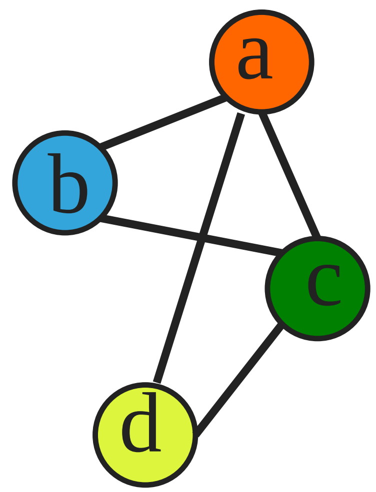
G1 **G2**

$G1_1$

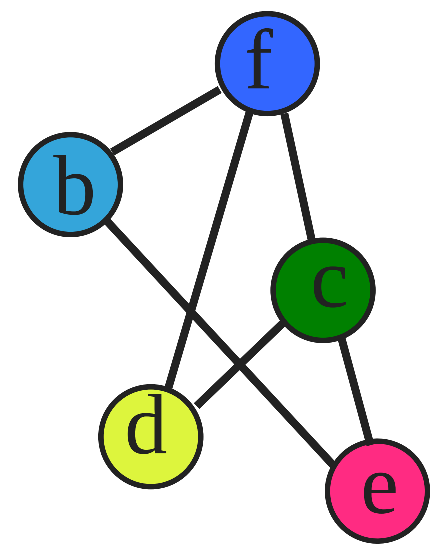
edge-remove (b,c)



IGM: EXAMPLE



G1



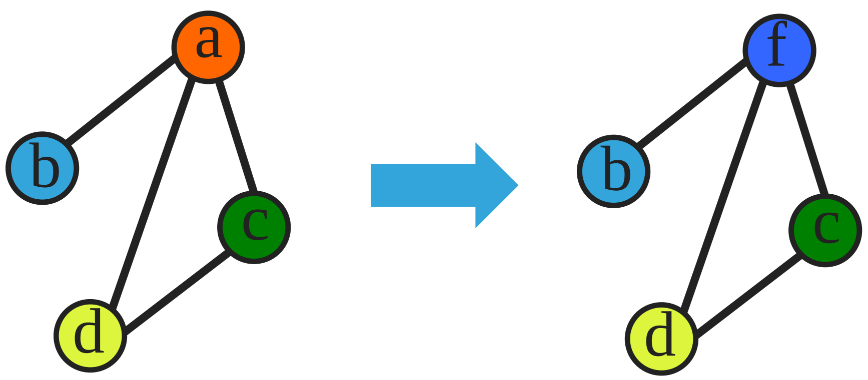
G2

$G1_1$

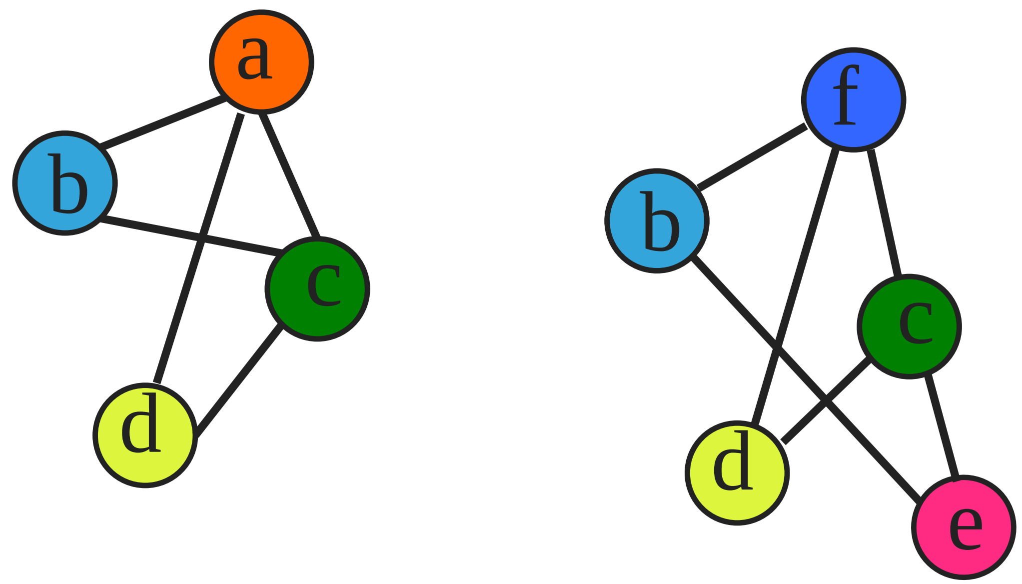
edge-remove (b,c)

$G1_2$

label-change(a,f)



IGM: EXAMPLE

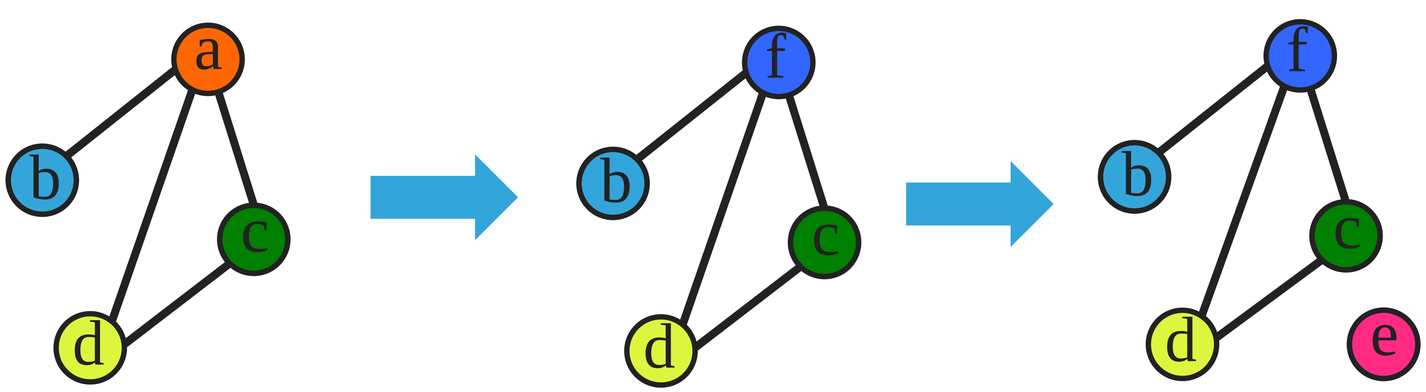


G1 **G2**

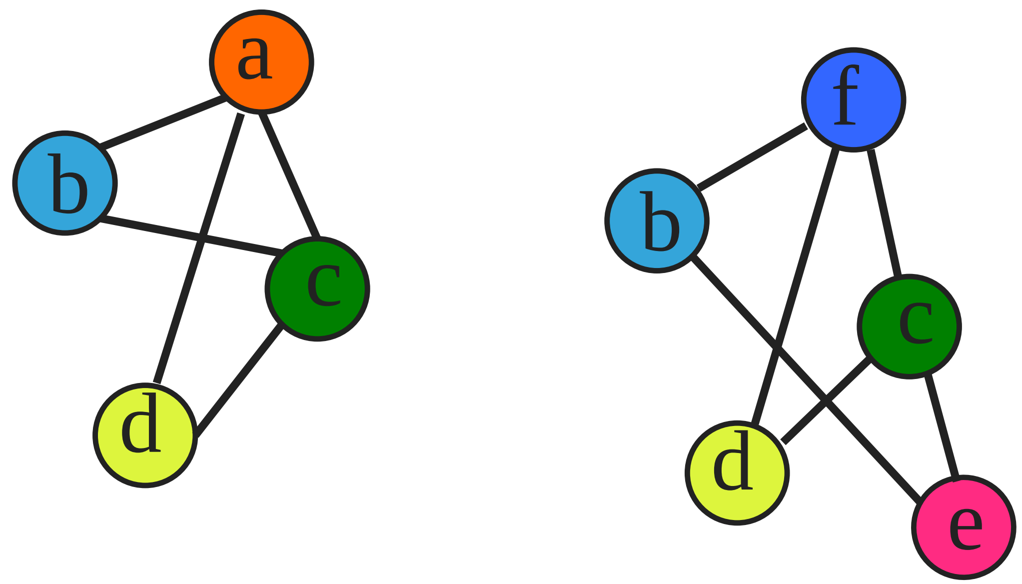
$G1_1$ edge-remove (b,c)

$G1_2$ label-change(a,f)

$G1_3$ node-add(e)



IGM: EXAMPLE



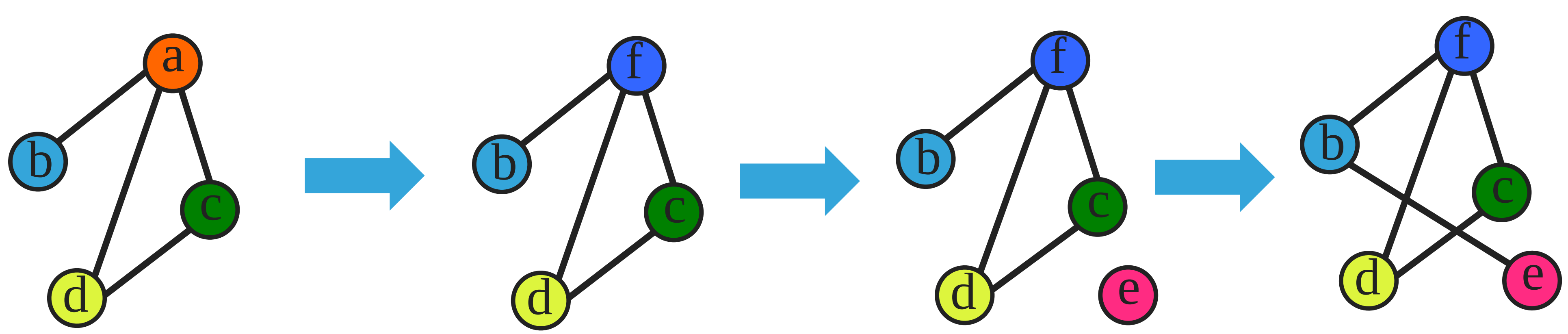
G1 G2

$G1_1$ edge-remove (b,c)

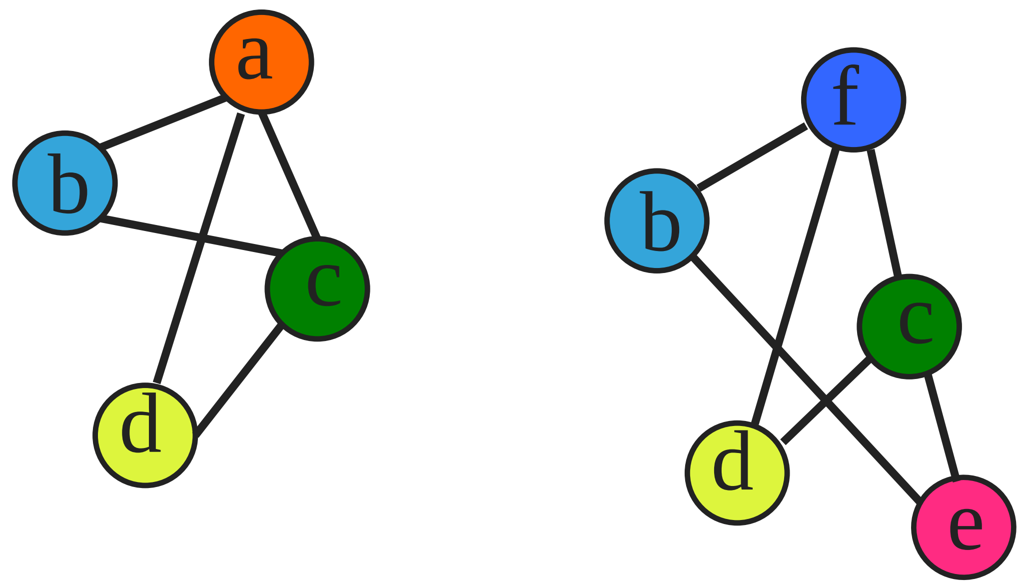
$G1_2$ label-change(a,f)

$G1_3$ node-add(e)

$G1_4$ edge-add(b,e)



IGM: EXAMPLE



G1 G2

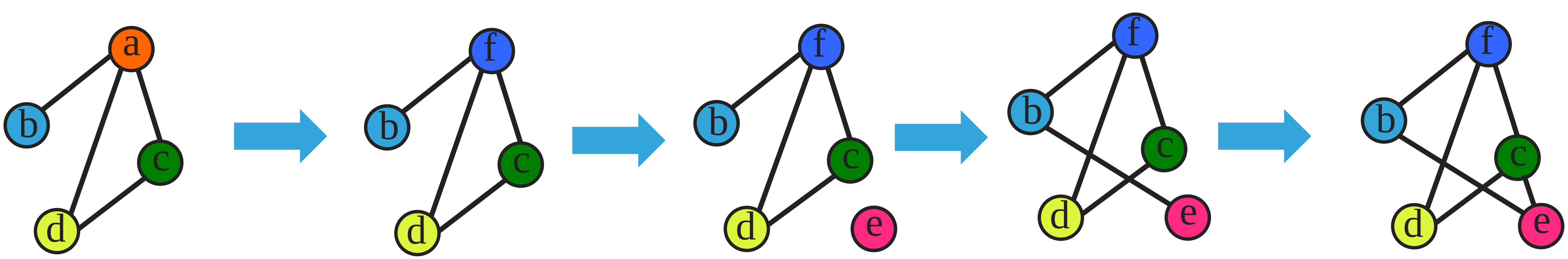
$G1_1$ edge-remove (b,c)

$G1_2$ label-change(a,f)

$G1_3$ node-add(e)

$G1_4$ edge-add(b,e)

$G1_5$= G2 edge-add(c,e)



IGM: TREE SEARCH

- Use tree search with backtracking.
- Direct the search by the cost of the Partial Matching.
- Use, as heuristic, estimate of the matching cost for the remaining nodes.
- Use A* like algorithm to prune unfruitful paths.

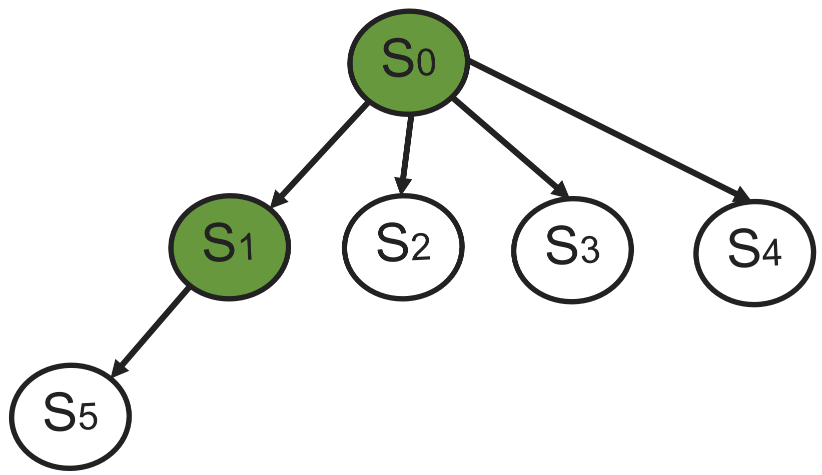
Some important methods

Tsai–Fu,1979	Gharaman,1980
Shapiro–Haralick,1981	Eshera – Fu,1984
Wong,1990	Dumay,1992
Sasha,1994	Cordella,1996
Allen,1997	Serratosa,1999
Berretti,2000	Gregory – Kittler,2002

ERROR-CORRECTING EDIT DISTANCE

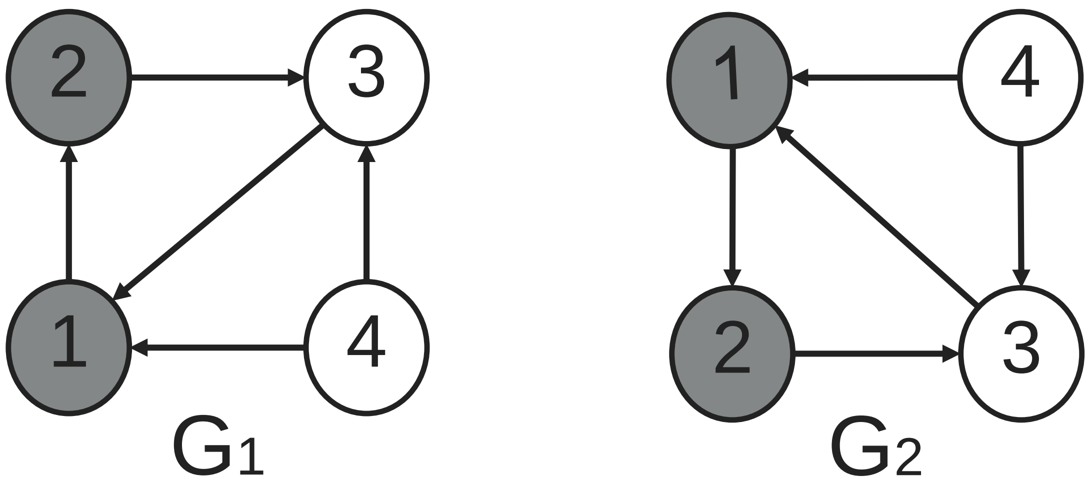
- Uses a tree based search for finding the edit paths which can transform G1 into G2 having the minimum cost
- The algorithm works into SSR: a path in the tree represents an edit path among the graphs
- A current edit-path is closed if it allows to reach a same intermediate state with an higher edit-cost

IGM: TREE BASED EDIT-PATHS



State	Mapping
S_0	$M(S_0)=\{\}$
S_1	$M(S_1) = \{(1, 1)\}$
S_2	$M(S_2) = \{(2, 2)\}$
S_3	$M(S_3) = \{(3, 3)\}$
S_4	$M(S_4) = \{(4, 4)\}$
S_5	$M(S_5) = \{(1, 1),(2,2)\}$

- Each state represents a possible partial matching by considering the needed edit operations and a cost is associated to.



IGM: BIPARITE GED

- Linear Sum Assignment Problem (LSAP): Riesen-Bunke 2009
  - Cost Matrix Improvents
    - Bags of Walks: Brun, Gaüzère 2014
    - K-Neighborhood: Carletti, Gaüzère 2015
  - Final Mapping Improvements
    - Beam Search: Riesen, Bunke 2014

- Quadratic Assignment Problem (QAP):
  - Brun, Vento, Foggia, Bougleux, Carletti, Gaüzère 2016
  - Quadratic cost function formulation
  - The GED is computed solving a QAP

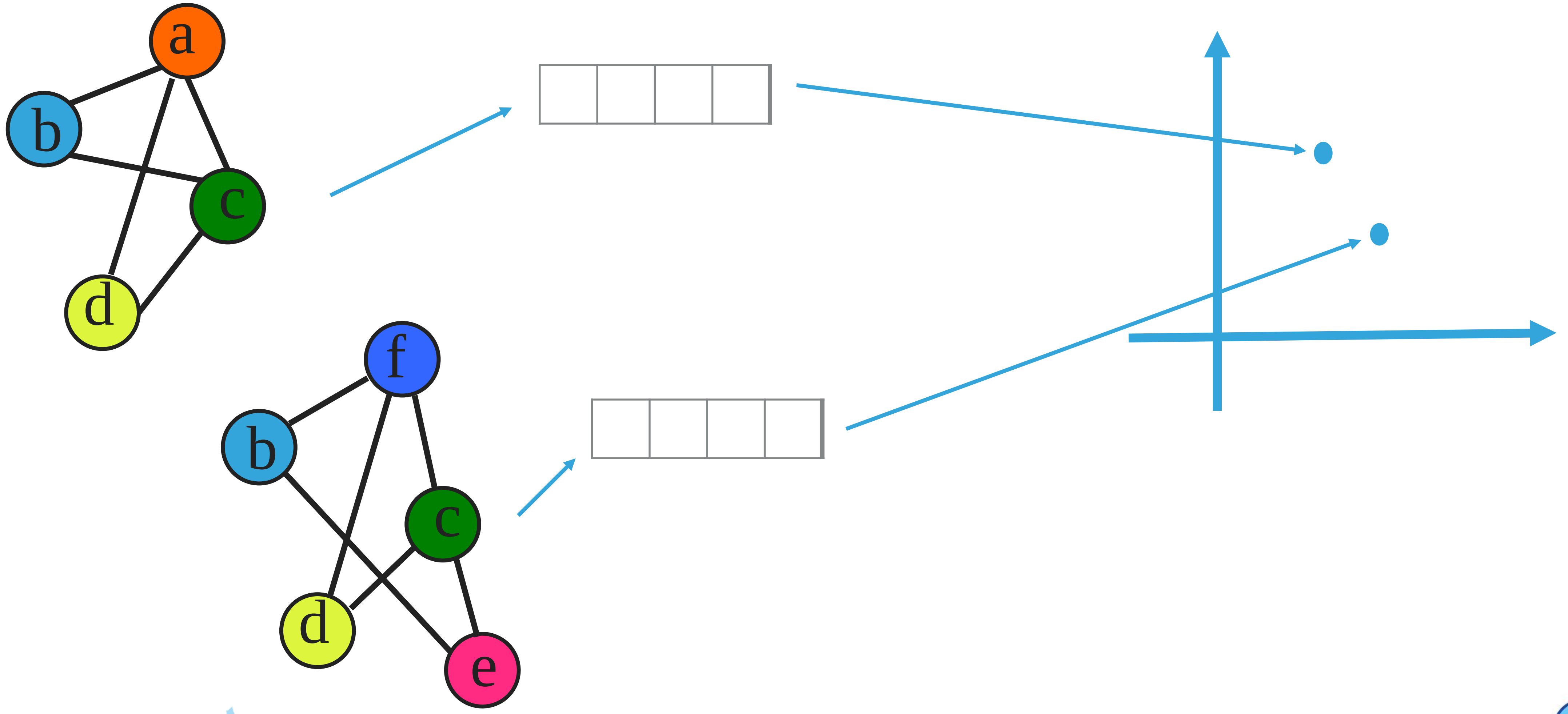
ARE WE READY? COMPUTATIONAL EFFORT

- The time needed for GED
- A* on 10 nodes about 7 seconds
- LAP on 30 nodes about 10 ms average distance 138
- LAP on 10 nodes about 10^-3 sec average error 18
- QAP on 30 nodes about 20 ms average distance 63
- QAP on 10 nodes about 10^-3 secs average error 6
- QAP 100 nodes about 1.3 secs and 90% of perfect approximation average error 2

IMPURE METHODS

USING A VECTOR SPACE

- Impure methods report in the graph space the main operations defined in a vector space
- The idea is to reuse on graphs the basic methods in SPR:
  - The first important achievement is the concept of distance between graphs



IMPURE GRAPH MATCHING AND LEARNING

- Once the graph distance has been defined we can reuse effective SPR methods for learning and classifying graphs, as:
  - NN, K-NN, (K-K')-NN
- The latter allows to solve a supervised classification problem given a set of labelled graphs, only using the distance

IMPURE GRAPH MATCHING AND LEARNING

- Once the graph distance has been defined we can reuse effective SPR methods for learning and classifying graphs, as:
  - NN, K-NN, (K-K')-NN
- The latter allows to solve a supervised classification problem given a set of labelled graphs, only using the distance

ARE WE READY?

ARE WE READY?

- We have a graph distance (edit cost dist.)

… not all the properties are always valid:

  - $D(G1,G2) \geq 0$ holds
  - $D(G1,G2) = D(G2,G1)$
  - $D(G1,Gx) \leq D(G2,Gx) + D(Gx,G2)$

ARE WE READY?

- We have a graph distance (edit cost dist.)

  … not all the properties are always valid:

  - D(G1,G2) ≥ 0 holds
  - D(G1,G2) = D(G2,G1)
  - D(G1,Gx) ≤ D(G2,Gx) + D(Gx,G2)
- The latter two hold if the costs are properly defined
- Require exhaustive A* search and proper definition of edit costs

NN CLASSIFIER FOR GRAPHS

- Given a set S of labeled graphs (reference set)
- An input graph G (to classify)
- The graph Gx of S having the minimum edit distance from G is determined
- G is assigned the class of Gx
- With simple variants we obtain K-NN and (K-K')-NN

OTHER ACHIEVEMENTS WITH GRAPH DISTANCE

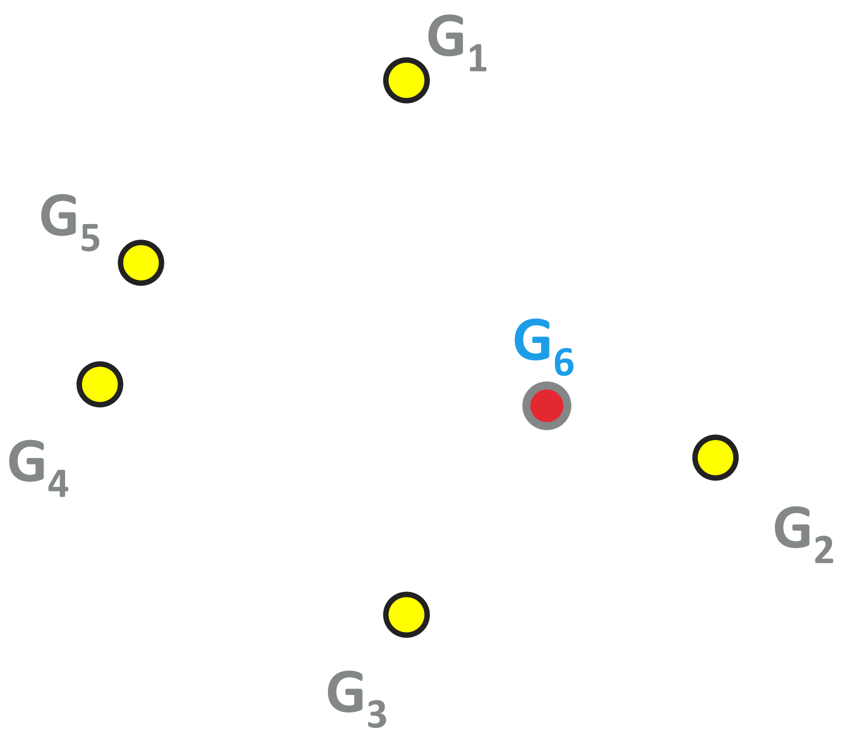
- Many learning procedures in a vector space use the centroid of points as a prototype (K-means clustering)
- Points are graphs and centroids prototypes.
- We are required to define graph centroids:
  - Median graphs
  - Generalized Median Graph

GRAPH PROTOTYPE AS THE MEDIAN GRAPH

- Given a set S of graphs of a same class, a prototype C of S is the graph which minimizes the sum of its distances from the graph in S (**Median Graph**)

- Differences with a space vector: the space is discrete and the median graph belonging to S!

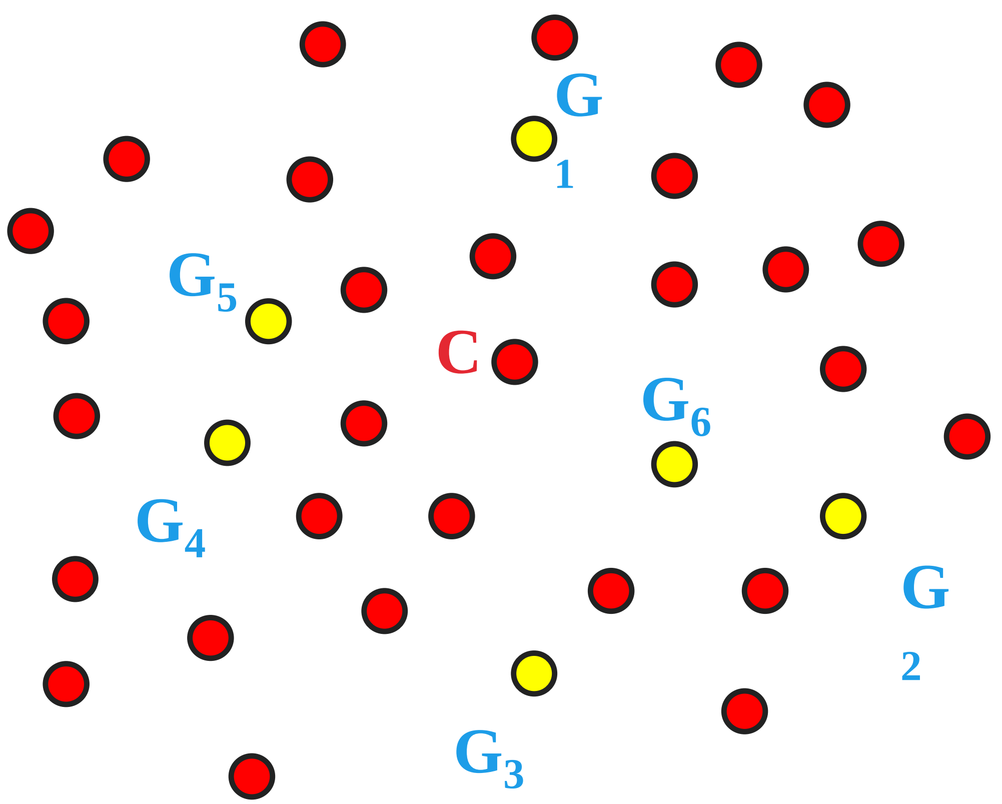
$$\hat{S} = \arg\min_{g_1 \in S} \sum_{g_2 \in S} d(g_1, g_2)$$



GENERALIZED MEDIAN GRAPH

- The graph Generalized Median of a set S of graphs graph is obtained by preliminarily buiding S’
- S’ is a very wide set of graphs obtained by the graphs in S changing in any possible mode all the (edge and node) labels to any of the graphs in S

$$\hat{S} = \underset{g_1 \in S}{\operatorname{argmin}} \sum_{g_2 \in S} d(g_1, g_2)$$



EXTENDING LVQ ON GRAPHS

- A simple LVQ algorithm determining the vector centroids of n classes C_1, C_n, is:

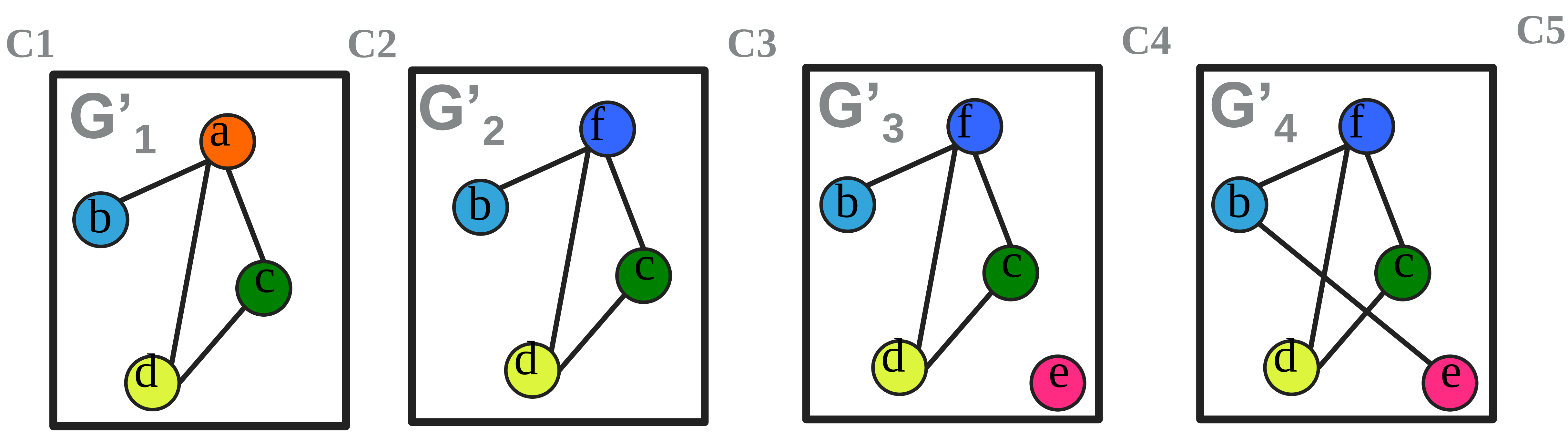
  1. Pick a graph G_{in} randomly from TS and calculate the nearest quantization graph C_w
  2. Move C_w (median graph) towards G_{in} by a fraction of the distance, if the class of C_w is correct, otherwise move in the opposite
  3. Repeat until the total error on the training becomes stable

- Given two graphs G1 and G2 we need to transform G1 into another graph G' so as to reduce the distance between G' and G2

GRAPH LVQ: WHAT WE NEED?

- The distance between graphs gives the edit operations $e_1, e_2, \ldots, e_k$ that transform G1 into G2 with minimum cost
- Dropping the k first edit operations, we have a new sequence that transforms G' into another graph G2
- It derives that G' is more similar than G1 to G2 by the amount given by the sum of the edit cost dropped

EXAMPLE



ARE WE READY ?

- The extension to graphs of vector space operations has a practical huge impact:
  - Standard learning and classification methods

- The theoretical framework isn't robust:
  - The graph distance is discrete, as related to edit operations!
  - This distance is affected by uncontrollable errors as the edit distance depends on domain-based edit cost assignment

REFERENCES FOR THIS TALK

- “Thirty years of graph matching in Pattern Recognition” , Conte, Foggia, Sansone, Vento, IJPRAI, 2004
- “Graph matching and learning in pattern recognition in the last 10 years”, Foggia, Percannella, Vento, IJPRAI, 2014
- “A long trip in the charming world of graphs for PR”, Vento, PatRec, 2015

THANKS FOR YOUR ATTENTION