List 3-Colouring of $(P_2 + P_5)$-Free and $(P_3 + P_4)$-Free Graphs

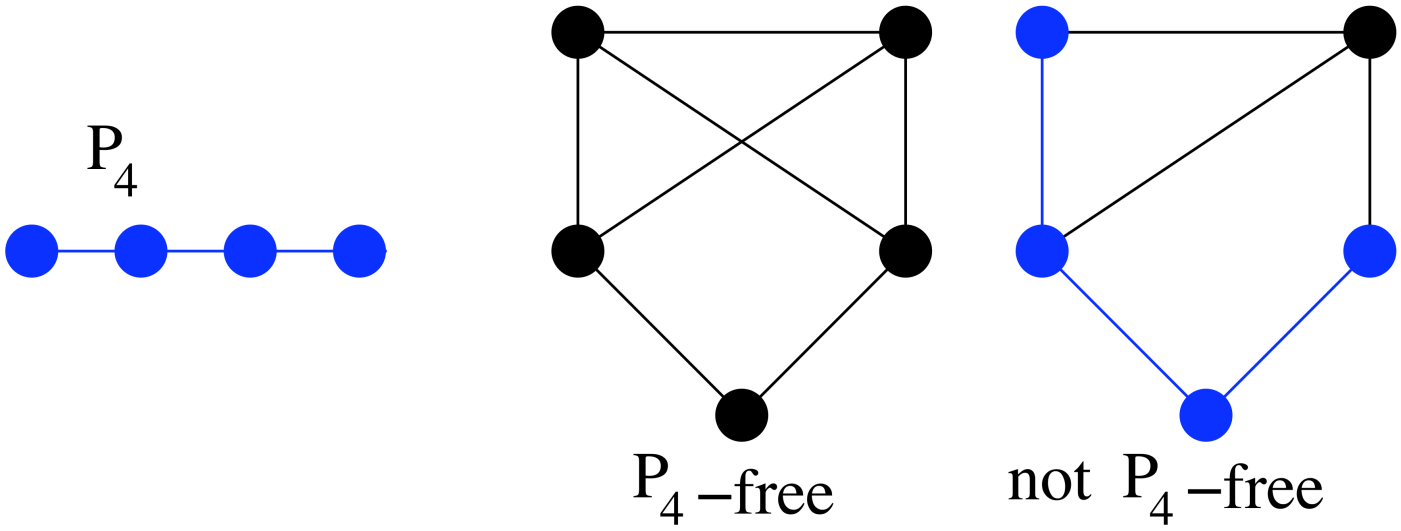
Tereza Klimošová Josef Malík Tomáš Masařík
Jana Novotná Daniël Paulusma Veronika Slívová

Algo Seminar, UIB

May 18, 2018

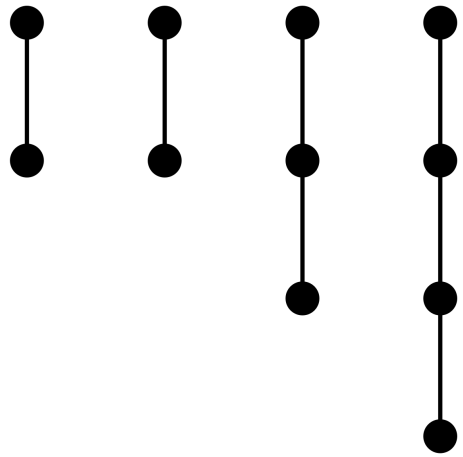
Preliminaries

- Graph G is H-free graph if G has no induced subgraph isomorphic to H.



Preliminaries

- Graph G is H-free graph if G has no induced subgraph isomorphic to H.
- Linear forest is a disjoint union of paths



$$H = 2P_2 + P_3 + P_4$$

Preliminaries

- Graph G is H-free graph if G has no induced subgraph isomorphic to H.
- Linear forest is a disjoint union of paths

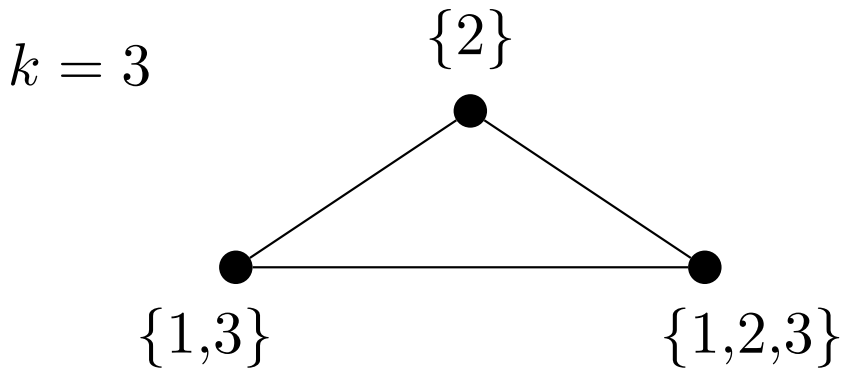
k-Colouring: Given a graph G, does there exists a colouring of G which uses at most k colours?

Preliminaries

- Graph G is H-free graph if G has no induced subgraph isomorphic to H.
- Linear forest is a disjoint union of paths

k-Colouring: Given a graph G, does there exists a colouring of G which uses at most k colours?

List k-Colouring: Given a graph G and a list assignment L, a subset of colours $\{1,\ldots,k\}$, does there exist a colouring of G that respects L?



State-of-the-art

- For every $k \geq 3$, k-Colouring on H-free graphs is NP-complete if H contains a cycle [Emden-Weinert, Hougardy, Kreuter 98] or

State-of-the-art

- For every $k \geq 3$, k-Colouring on H-free graphs is NP-complete if H contains a cycle [Emden-Weinert, Hougardy, Kreuter 98] or an induced claw [Holyer 81 & Leven, Galil 83].

State-of-the-art

- For every $k \geq 3$, k-Colouring on H-free graphs is NP-complete if H contains a cycle [Emden-Weinert, Hougardy, Kreuter 98] or an induced claw [Holyer 81 & Leven, Galil 83].
- Hence, it remains to consider the case where H is a linear forest.

State-of-the-art, Summary for P_t-free graphs

	k-Colouring				List k-Colouring			
t	$k=3$	$k=4$	$k=5$	$k\geq 6$	$k=3$	$k=4$	$k=5$	$k\geq 6$
$t\leq 5$	P	P	P	P	P	P	P	P
$t=6$	P	P	NP-c	NP-c	P	NP-c	NP-c	NP-c
$t=7$	P	NP-c	NP-c	NP-c	P	NP-c	NP-c	NP-c
$t\geq 8$	?	NP-c	NP-c	NP-c	?	NP-c	NP-c	NP-c

State-of-the-art, Summary for P_t-free graphs

	k-Colouring				List k-Colouring			
t	$k=3$	$k=4$	$k=5$	$k\geq 6$	$k=3$	$k=4$	$k=5$	$k\geq 6$
$t\leq 5$	P	P	P	P	P	P	P	P
$t=6$	P	P	NP-c	NP-c	P	NP-c	NP-c	NP-c
$t=7$	P	NP-c	NP-c	NP-c	P	NP-c	NP-c	NP-c
$t\geq 8$	?	NP-c	NP-c	NP-c	?	NP-c	NP-c	NP-c

Theorem [Bonomo, Chudnovsky, Maceli, Schaudt, Stein, Zhong 17]

List 3-Colouring is polynomial-time solvable for P_7-free graphs.

State-of-the-art, Summary for P_t-free graphs

	k-Colouring				List k-Colouring			
t	$k=3$	$k=4$	$k=5$	$k\geq 6$	$k=3$	$k=4$	$k=5$	$k\geq 6$
$t\leq 5$	P	P	P	P	P	P	P	P
$t=6$	P	P	NP-c	NP-c	P	NP-c	NP-c	NP-c
$t=7$	P	NP-c	NP-c	NP-c	P	NP-c	NP-c	NP-c
$t\geq 8$	?	NP-c	NP-c	NP-c	?	NP-c	NP-c	NP-c

Theorem [Bonomo, Chudnovsky, Maceli, Schaudt, Stein, Zhong 17]

List 3-Colouring is polynomial-time solvable for P_7-free graphs.

Theorem [Edwards 86]

2-List Colouring is polynomial-time solvable for general graphs.

3-Colouring and List 3-Colouring

Theorem [survey: Golovach, Johnson, Paulusma, Song 17]

For a graph H with $|V(H)| \leq 6$, 3-Colouring and List 3-Colouring are polynomial-time solvable on H-free graphs if H is a linear forest, and NP-complete otherwise.

There are two open cases that must be solved in order to obtain the same statement for graphs H with $|V(H)| \leq 7$. These cases are

- $H = P_2 + P_5$
- $H = P_3 + P_4$.

Our results

Theorem

List 3-Colouring *is polynomial-time solvable for* $(P_2 + P_5)$*-free graphs and for* $(P_3 + P_4)$*-free graphs.*

Our results

Theorem

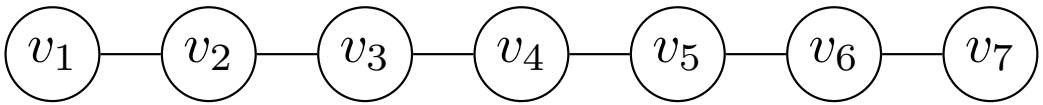
List 3-Colouring *is polynomial-time solvable for* $(P_2 + P_5)$*-free graphs and for* $(P_3 + P_4)$*-free graphs.*

Theorem

5-Colouring *is* NP*-complete for* $(P_3 + P_5)$*-free graphs.*

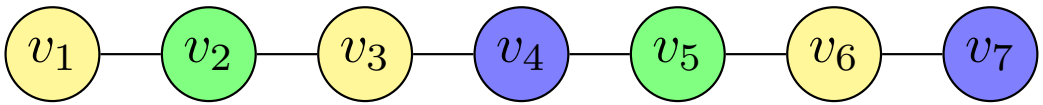
Proof Sketch

- Assume that G contains an induced P_7,



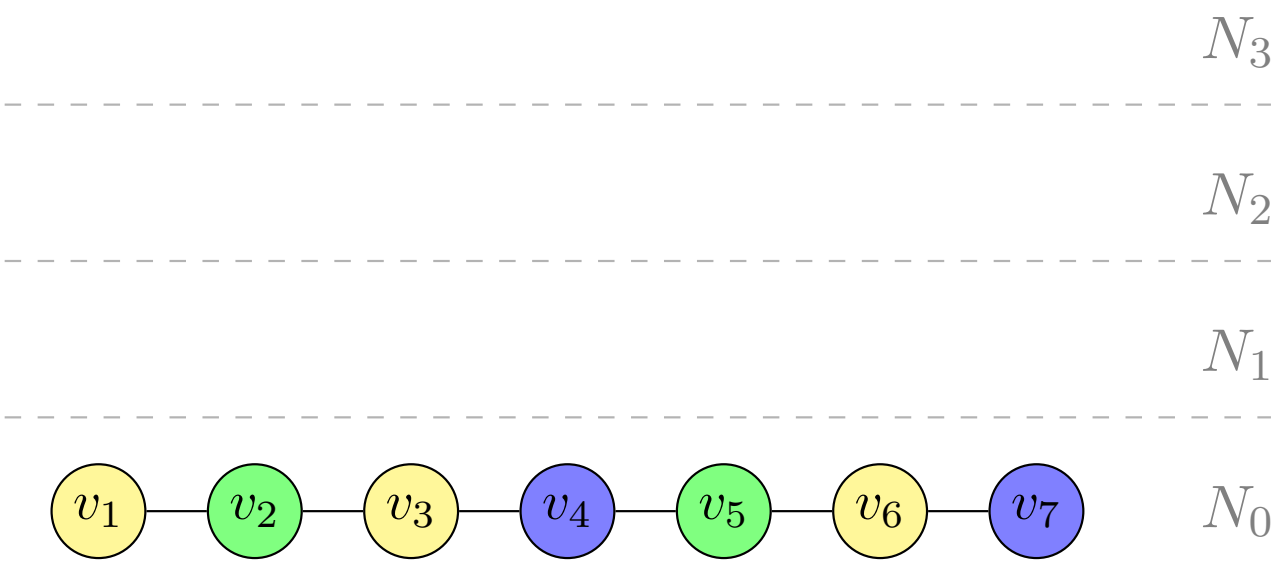
Proof Sketch

- Assume that G contains an induced P_7,
- consider every possibility of colouring the vertices of this P_7,



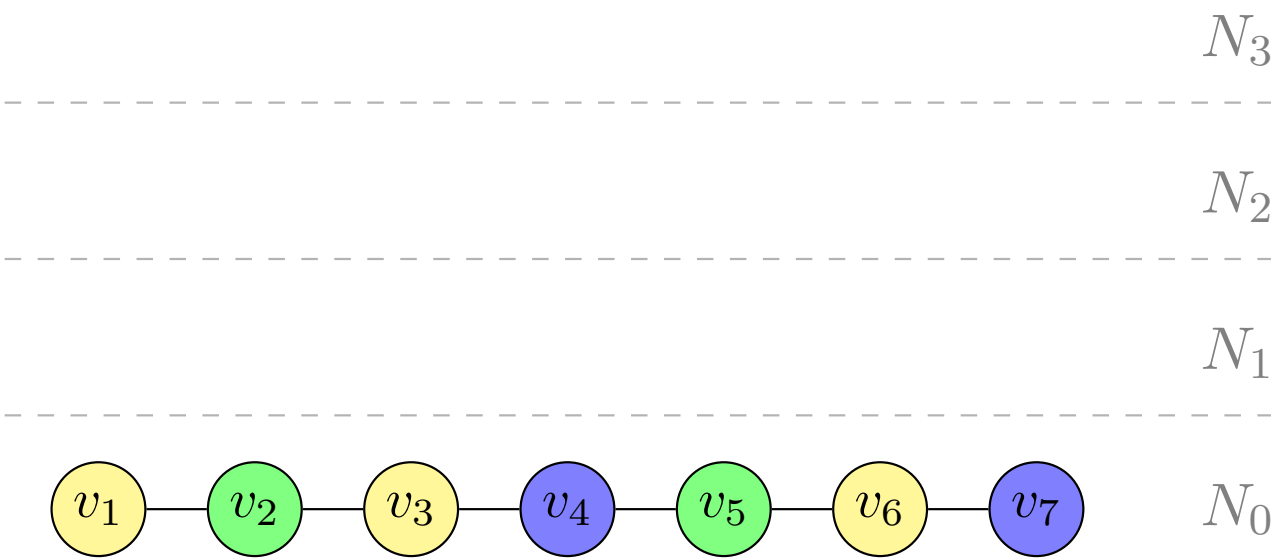
Proof Sketch

- Assume that G contains an induced P_7,
- consider every possibility of colouring the vertices of this P_7,
- partition vertices of G into four "layers" according to their distance to the P_7,



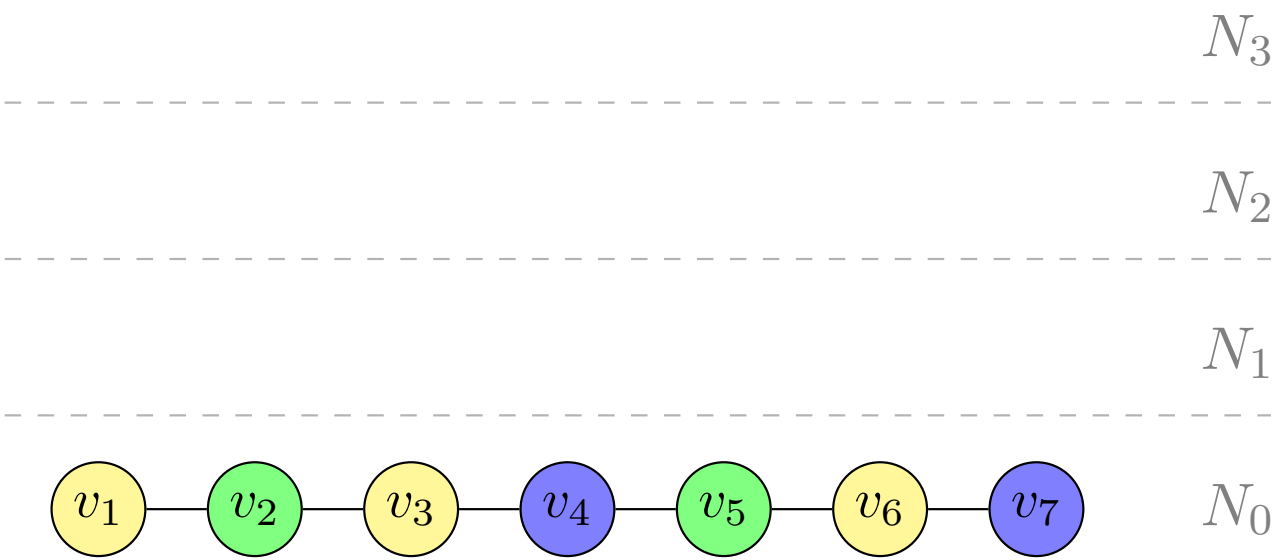
Proof Sketch

- Assume that G contains an induced P_7,
- consider every possibility of colouring the vertices of this P_7,
- partition vertices of G into four "layers" according to their distance to the P_7,
- reduce to a polynomial number of instances of 2-List Coloring.



Proof Sketch

- Assume that G contains an induced P_7,
- consider every possibility of colouring the vertices of this P_7,
- partition vertices of G into four "layers" according to their distance to the P_7,
- reduce to a polynomial number of instances of 2-List Coloring.



Most parts works for $(P_3 + P_5)$-free graphs.

Active vertices

- We want to reduce vertices with lists of size 3.

Active vertices

- We want to reduce vertices with lists of size 3.
- Vertices with list of size 3 are only in N_2.

Active vertices

- We want to reduce vertices with lists of size 3.
- Vertices with list of size 3 are only in N_2.

Definition

We call a vertex u in N_2 and its neighbours in N_1 active if $|L(u)| = 3$.

- We use A_1,A_2 for the set of all active vertices in N_1, N_2.

Active vertices

- We want to reduce vertices with lists of size 3.
- Vertices with list of size 3 are only in N_2.

Definition

We call a vertex u in N_2 and its neighbours in N_1 active if $|L(u)| = 3$.

- We use A_1,A_2 for the set of all active vertices in N_1, N_2.

Goal

Make A_2 empty.

Overview

- Preprocessing
- Phase 1: Reduce private neighbours of non-adjacent vertices in P_7
- Phase 2: Two types of lists in the first layer
- Phase 3: One type of lists in the first layer

Overview

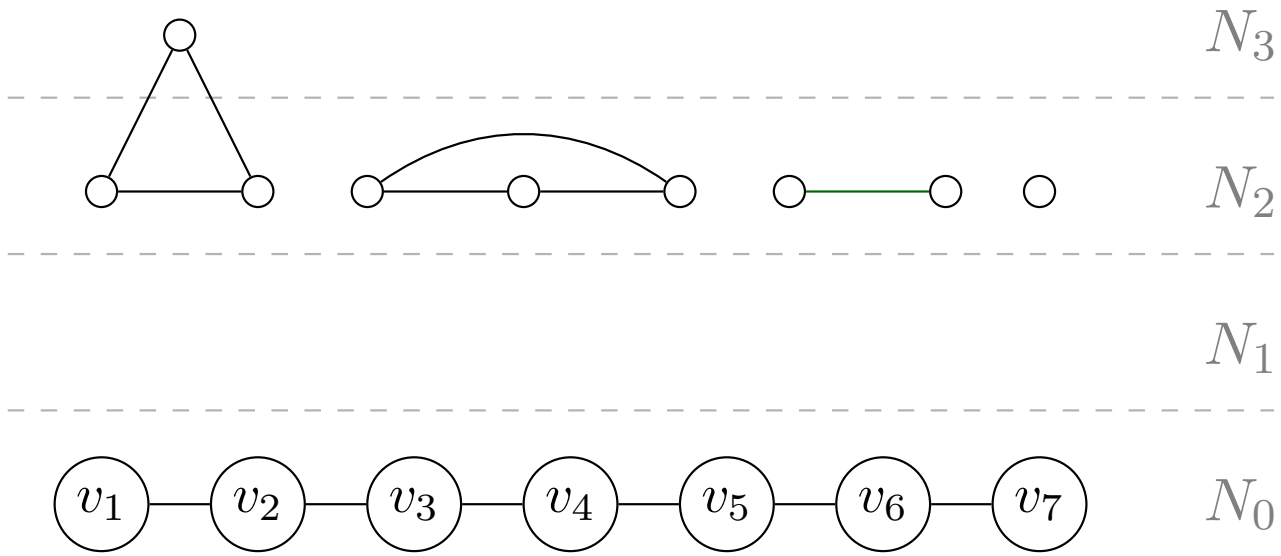
- Preprocessing
- Phase 1: Reduce private neighbours of non-adjacent vertices in P_7
- Phase 2: Two types of lists in the first layer
- Phase 3: One type of lists in the first layer

Preprocessing

- Set of rules which are applied everytime.

Preprocessing

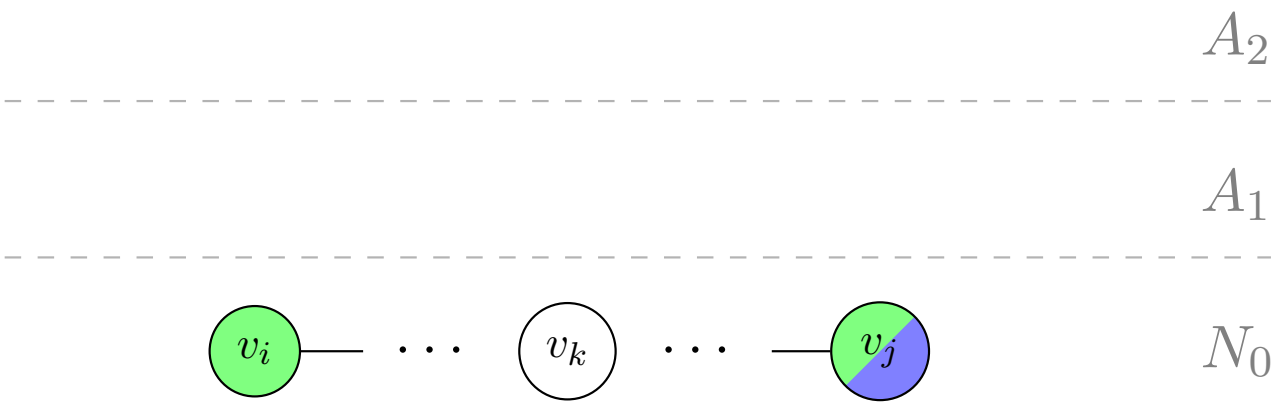
- Set of rules which are applied everytime.
- Problem can be reduced to this problem:



Overview

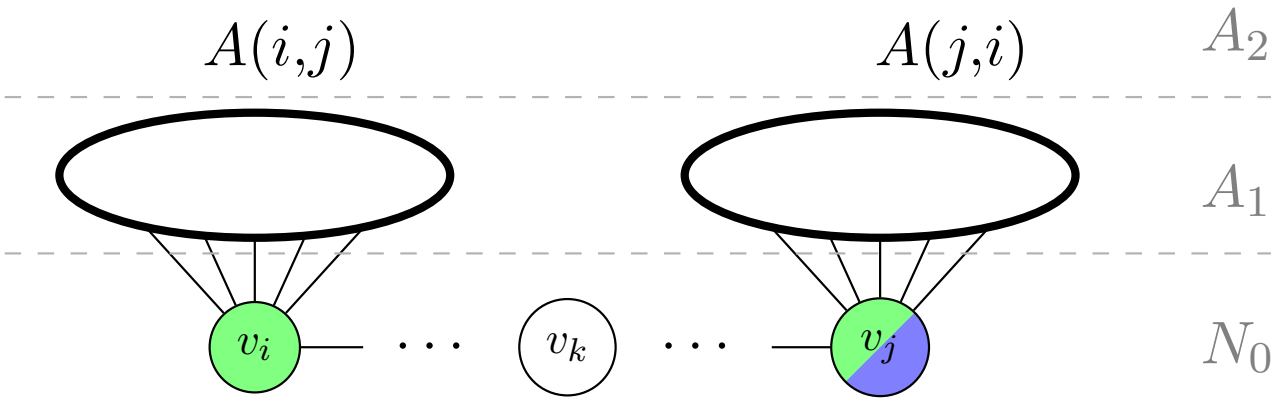
- Preprocessing
- Phase 1: Reduce private neighbours of non-adjacent vertices in P_7
- Phase 2: Two types of lists in the first layer
- Phase 3: One type of lists in the first layer

Phase I



Phase I

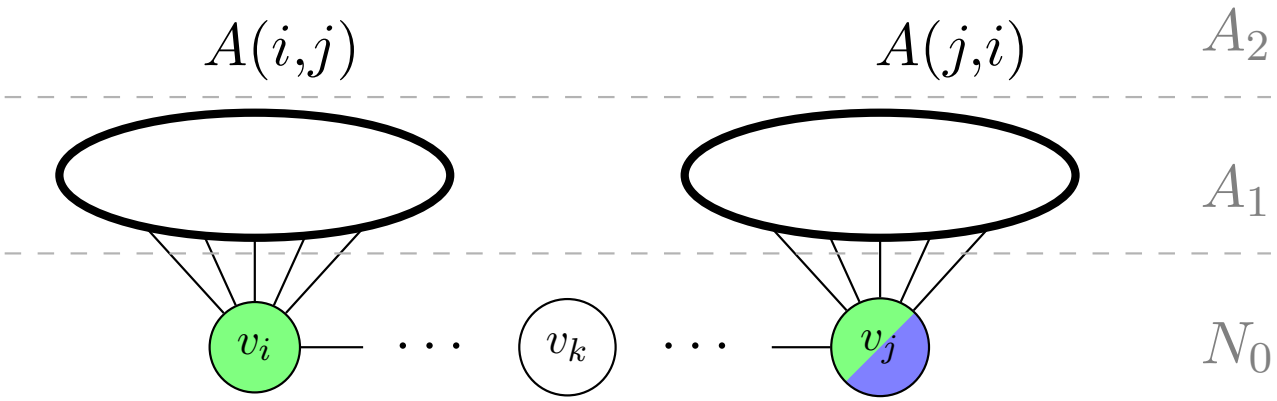
- $A(i,j)$ . . . set of vertices in A_1 adjacent to v_i and nonadjacent to v_j, similarly $A(j,i)$.



Phase I

Goal

At least one of $A(i,j), A(j,i)$ empty for nonadjacent v_i, v_j.

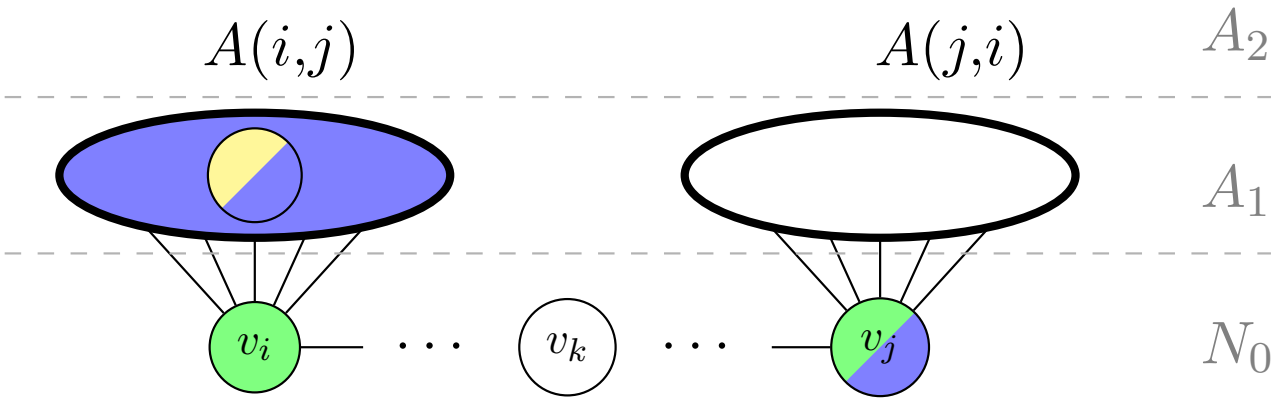


Phase I

Goal

At least one of $A(i,j), A(j,i)$ empty for nonadjacent v_i, v_j.

- Branching: at most one vertex in $A(i,j)$ is yellow $\rightarrow n+1$ branches,

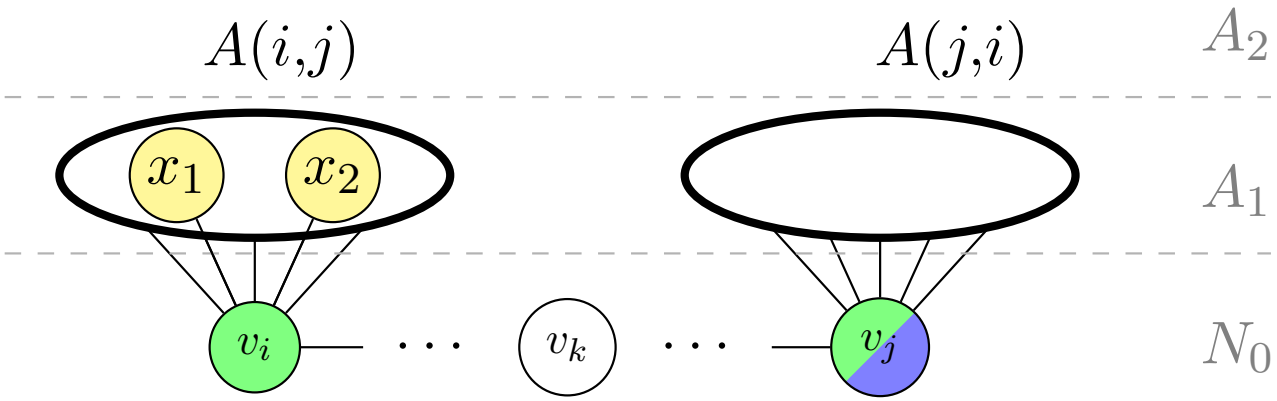


Phase I

Goal

At least one of $A(i,j), A(j,i)$ empty for nonadjacent v_i, v_j.

- Branching: at most one vertex in $A(i,j)$ is yellow $\rightarrow n+1$ branches, or two vertices are yellow $\rightarrow n^2$ branches.

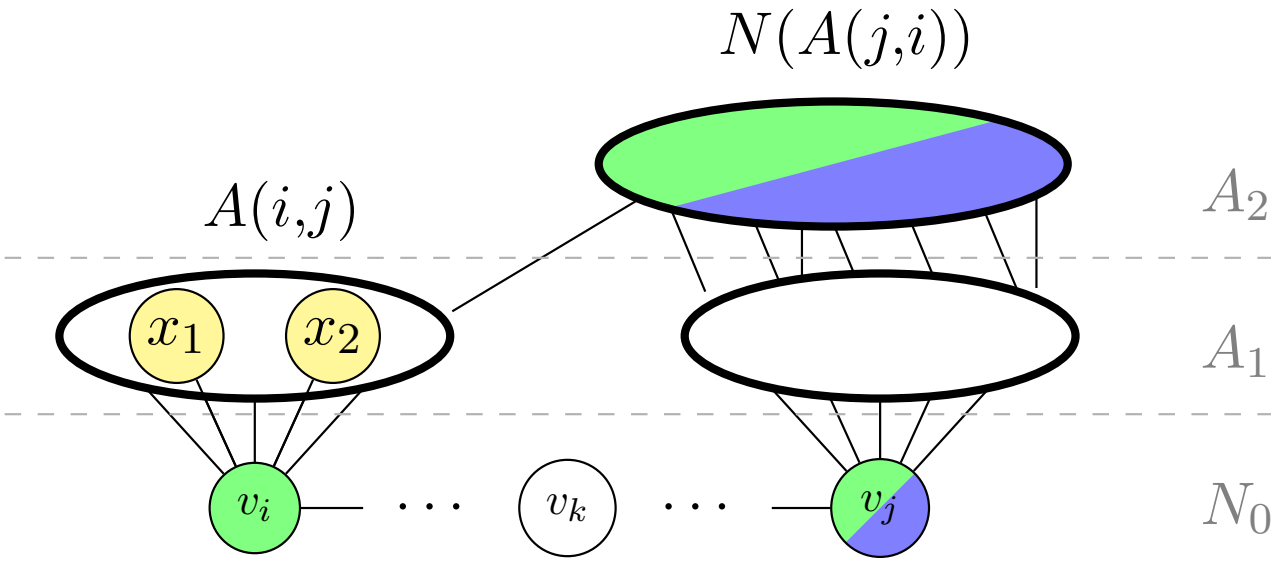


Phase I

Goal

At least one of $A(i,j), A(j,i)$ empty for nonadjacent v_i, v_j.

- Branching: Each vertex in $N(A(j,i)) \cap A_2$ given the color of $v_j \rightarrow$ one branch,

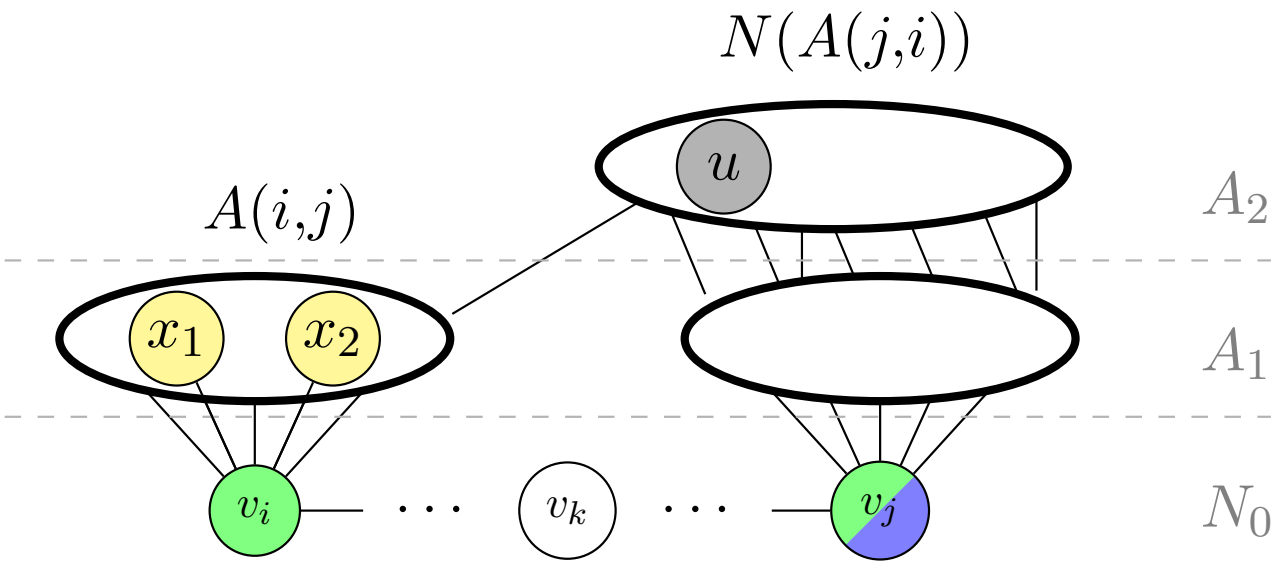


Phase I

Goal

At least one of $A(i,j), A(j,i)$ empty for nonadjacent v_i, v_j.

- Branching: Each vertex in $N(A(j,i)) \cap A_2$ given the color of $v_j \rightarrow$ one branch, or one vertex in $N(A(j,i)) \cap A_2$ has different color $\rightarrow 2n$ branches.

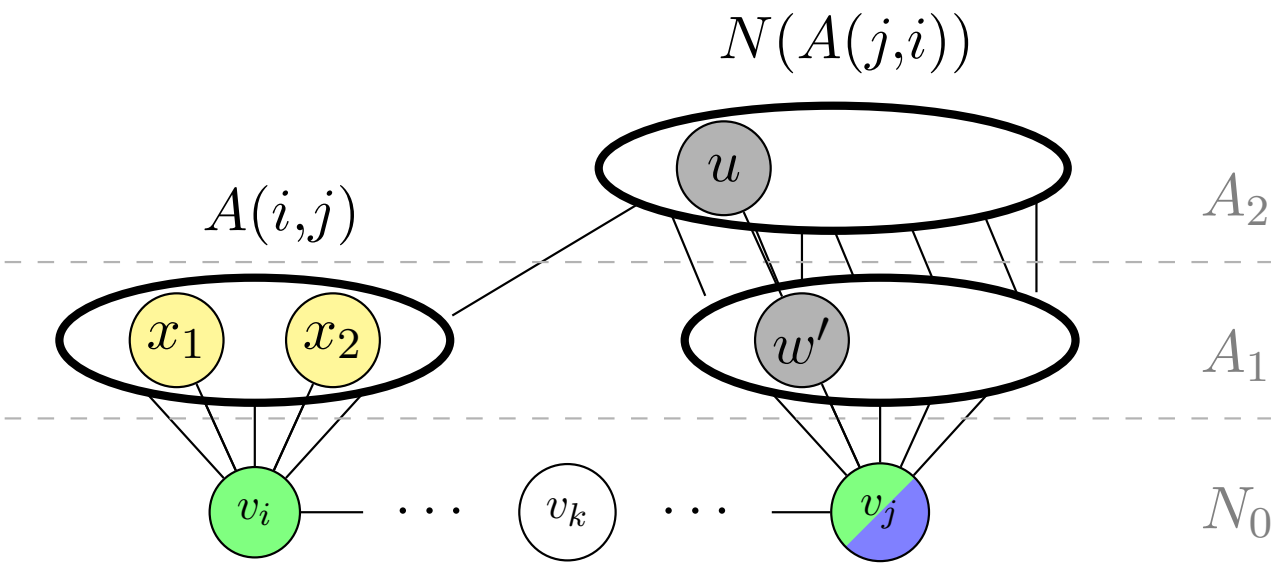


Phase I

Goal

At least one of $A(i,j), A(j,i)$ empty for nonadjacent v_i, v_j.

- Either $A(j,i) = \emptyset$, or find an induced P_5.

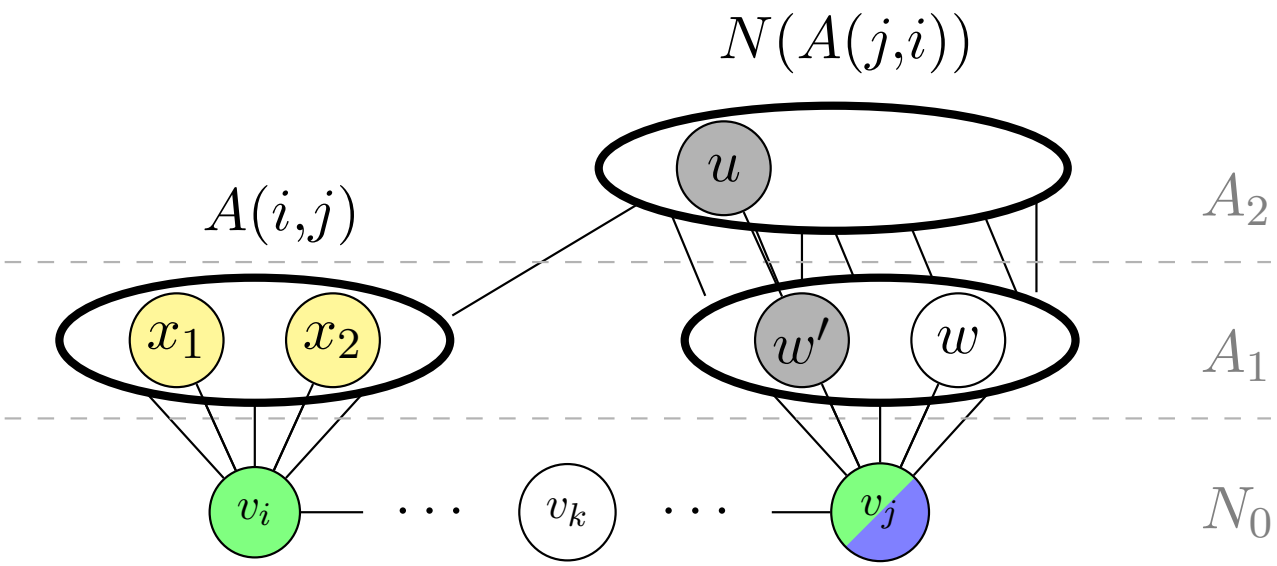


Phase I

Goal

At least one of $A(i,j), A(j,i)$ empty for nonadjacent v_i, v_j.

- Either $A(j,i) = \emptyset$, or find an induced P_5.

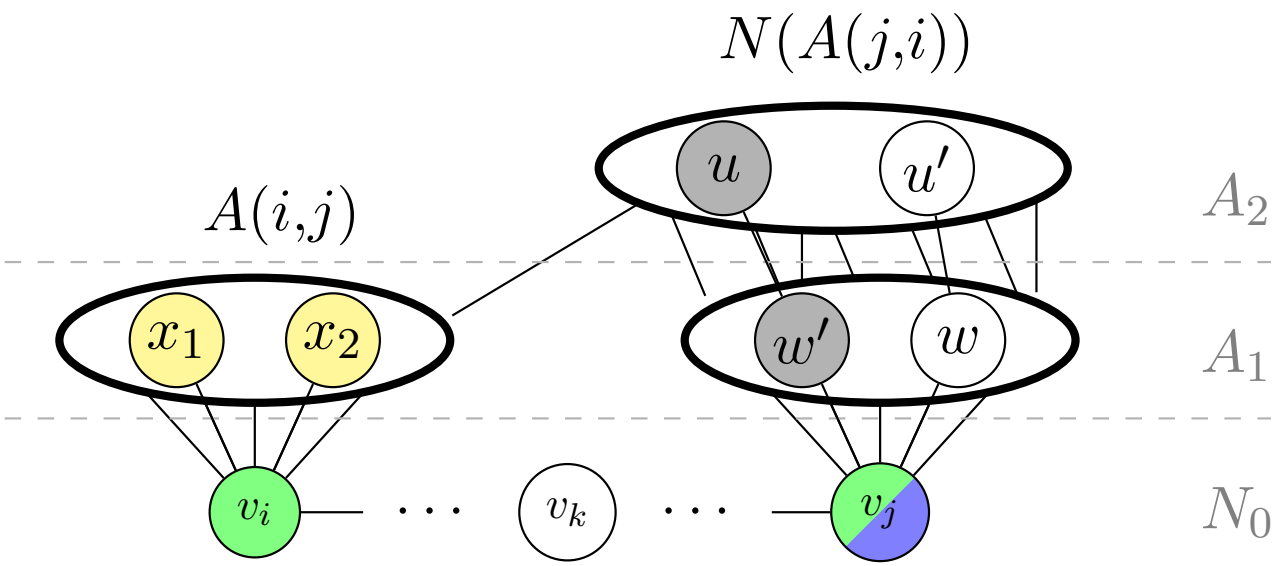


Phase I

Goal

At least one of $A(i,j), A(j,i)$ empty for nonadjacent v_i, v_j.

- Either $A(j,i) = \emptyset$, or find an induced P_5.

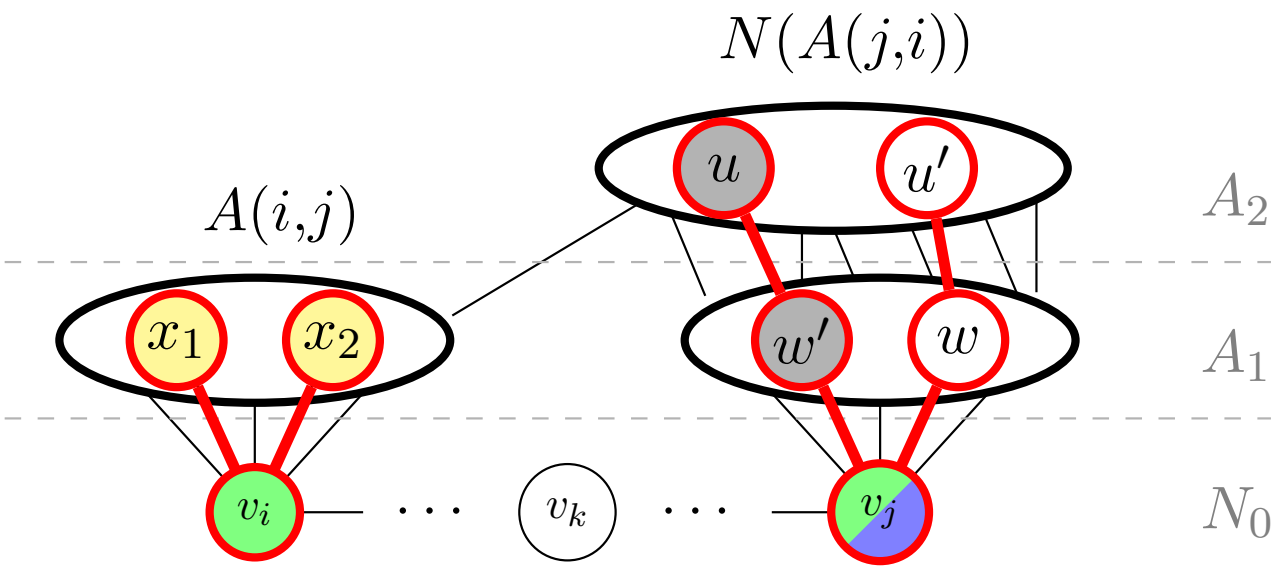


Phase I

Goal

At least one of $A(i,j), A(j,i)$ empty for nonadjacent v_i, v_j.

- Either $A(j,i) = \emptyset$, or find an induced P_5.

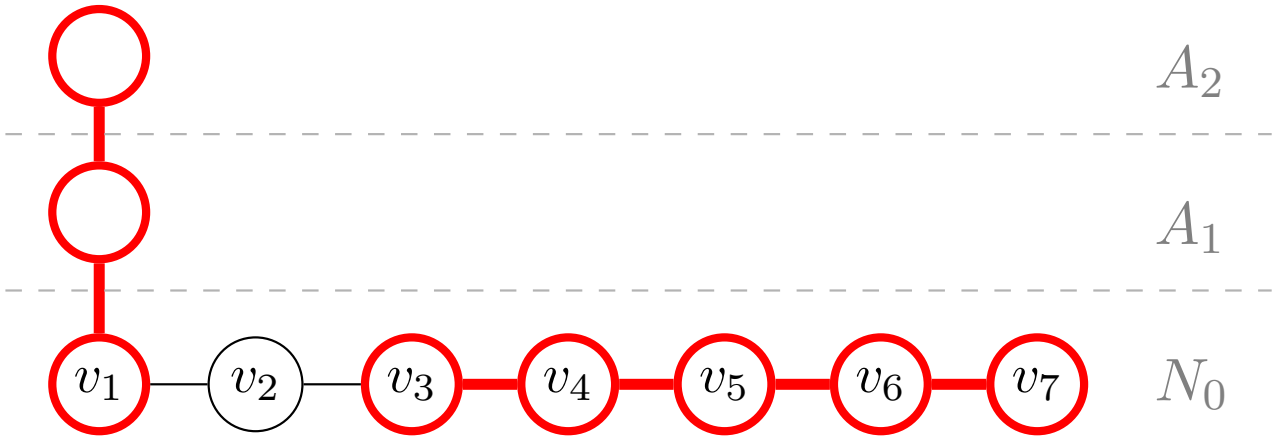


Phase I: Consequences

- $A(i,j)$ or $A(j,i)$ is empty for nonadjacent v_i, v_j
  $\rightarrow$ at most two types of lists of $\{b,y\},\{b,g\},\{y,g\}$ can occur in A_1,

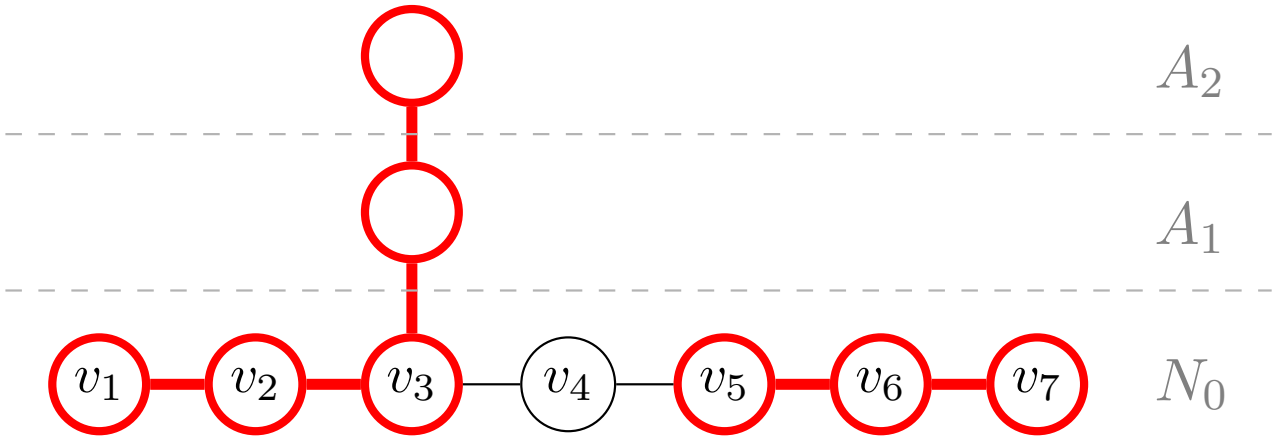
Phase I: Consequences

- $A(i,j)$ or $A(j,i)$ is empty for nonadjacent v_i, v_j
  $\rightarrow$ at most two types of lists of $\{b,y\},\{b,g\},\{y,g\}$ can occur in A_1,
- neighbours of v_i in A_1, $i \in \{1,3,5,7\}$, have another neighbour in N_0.



Phase I: Consequences

- $A(i,j)$ or $A(j,i)$ is empty for nonadjacent v_i, v_j
  $\rightarrow$ at most two types of lists of $\{b,y\},\{b,g\},\{y,g\}$ can occur in A_1,
- neighbours of v_i in A_1, $i \in \{1,3,5,7\}$, have another neighbour in N_0.

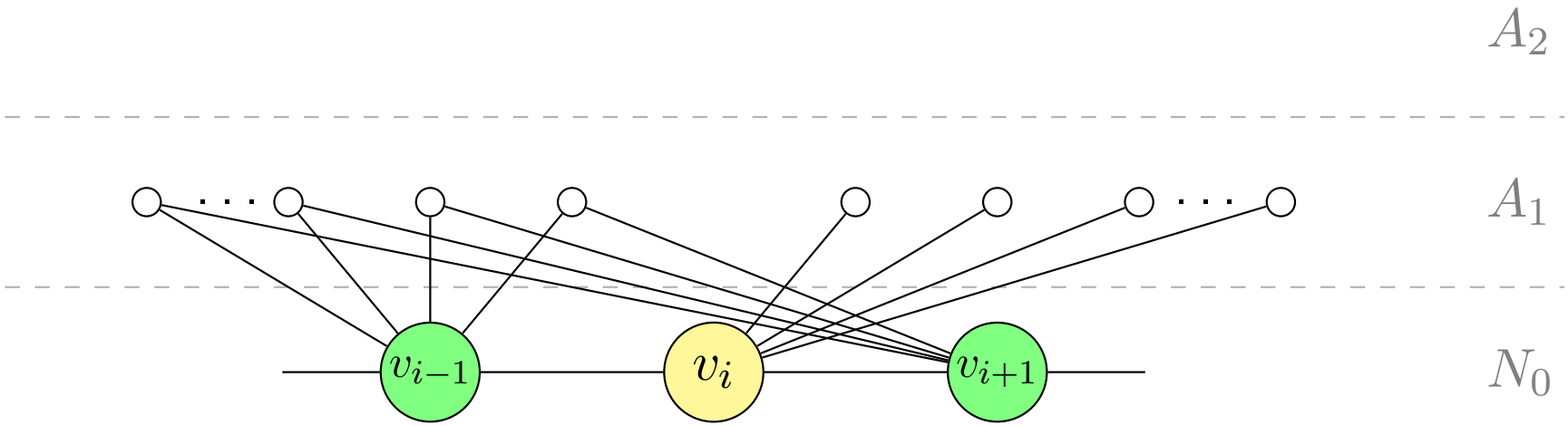


Overview

- Preprocessing
- Phase 1: Reduce private neighbours of non-adjacent vertices in P_7
- Phase 2: Two types of lists in the first layer
- Phase 3: One type of lists in the first layer

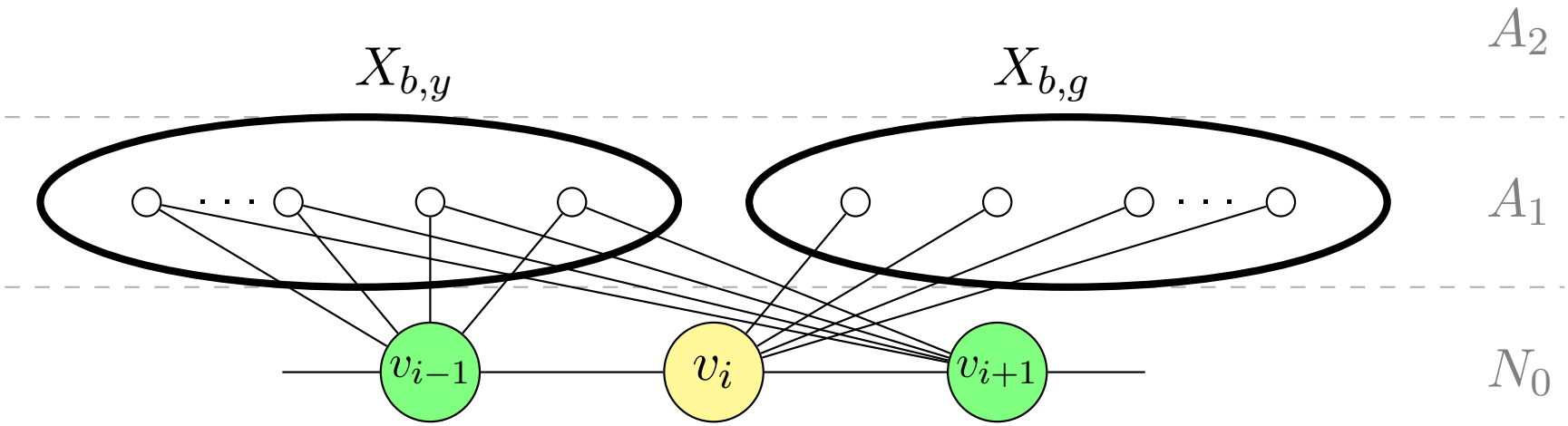
Phase 2: Different lists in A_2

- The only possible configuration:



Phase 2: Different lists in A_2

- The only possible configuration:

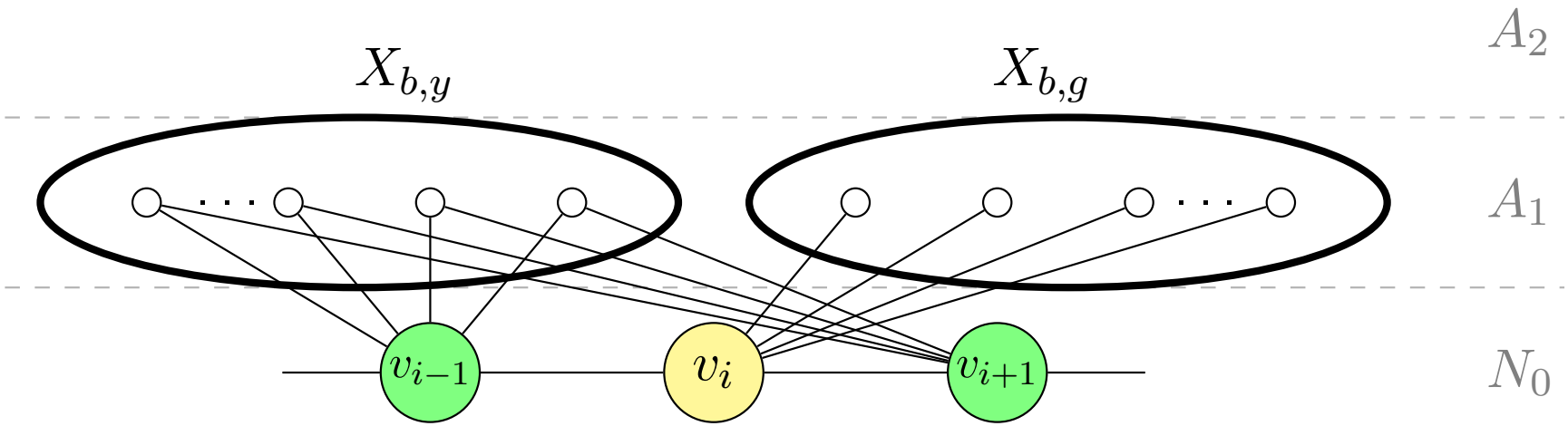


- Denote $X_{b,y}$ and $X_{b,g}$ vertices in A_1 with lists $\{b,y\}$ and $\{b,g\}$, respectivelly.

Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

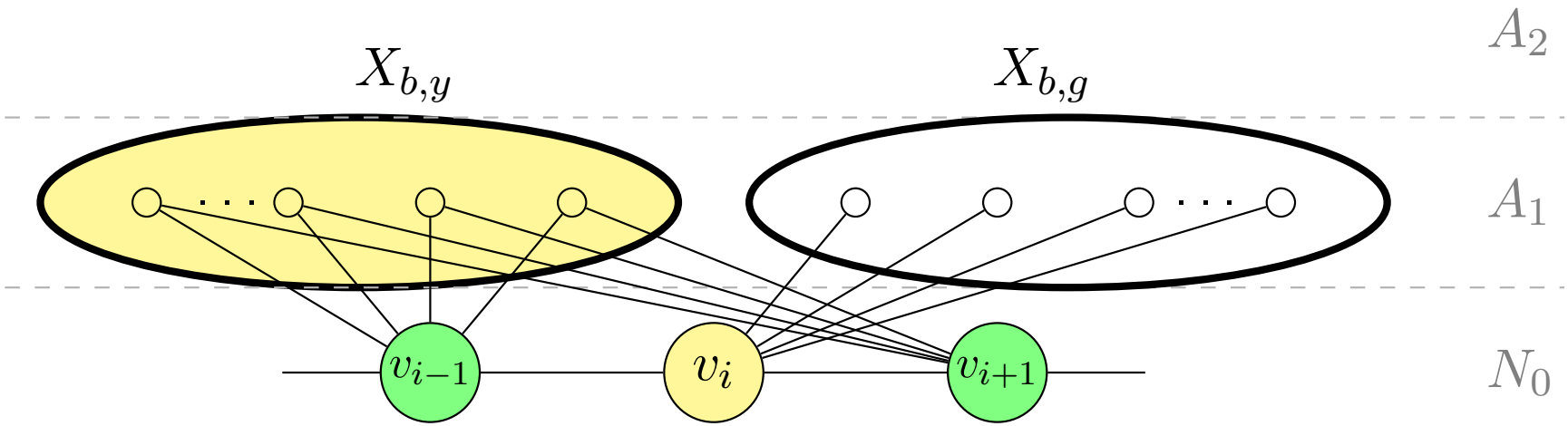


Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

- Either colour all vertices in $X_{b,y}$ by yellow $\rightarrow$ one branche,

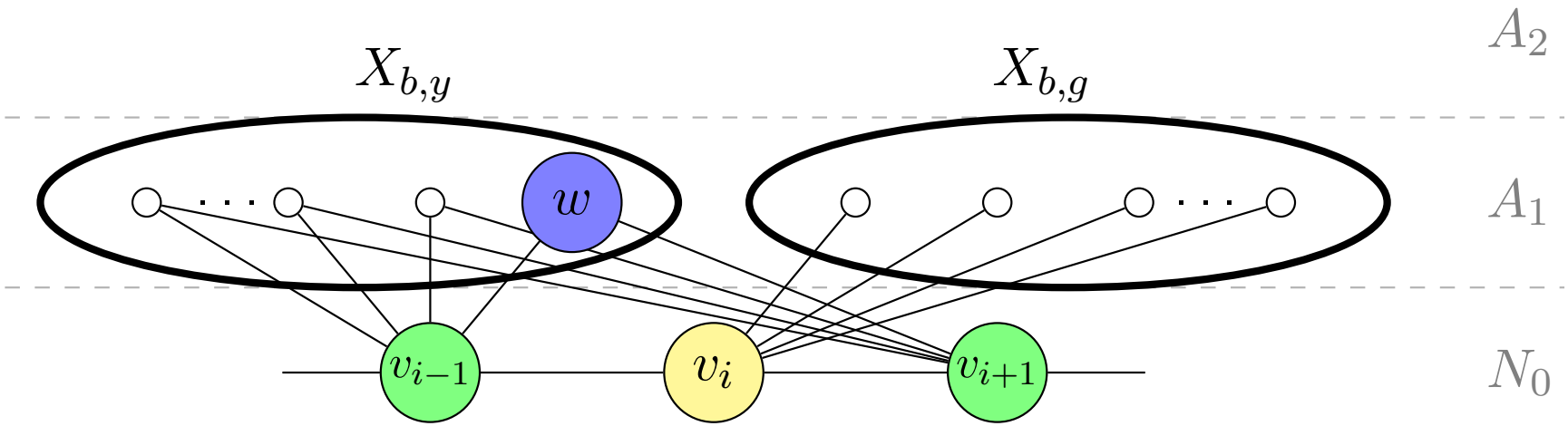


Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

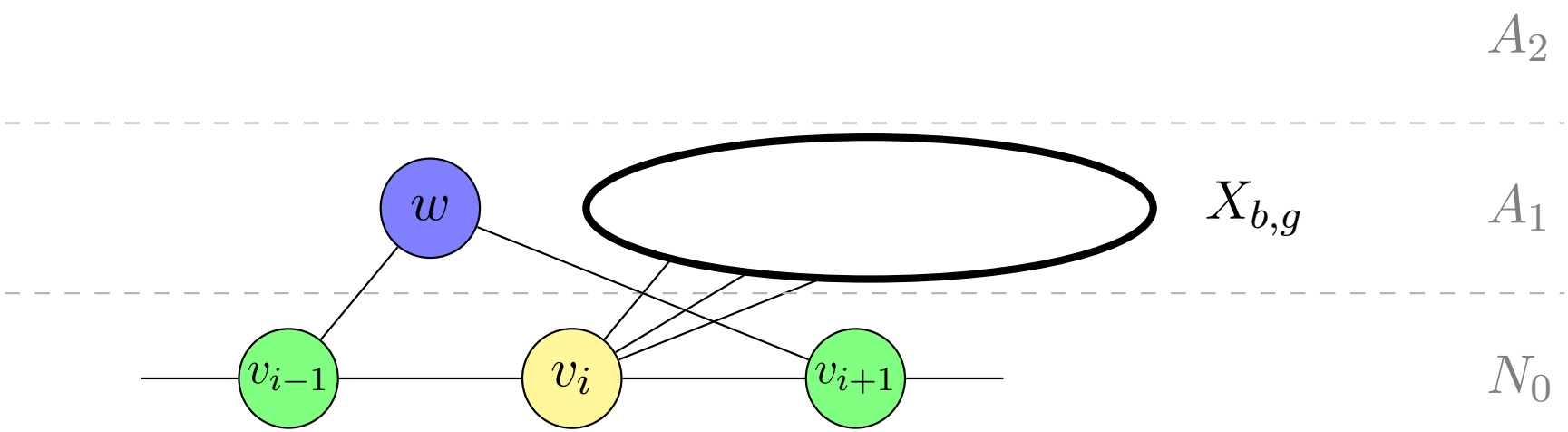
- Either colour all vertices in $X_{b,y}$ by yellow $\rightarrow$ one branche,
- or at least one vertex is blue $\rightarrow n$ branches.



Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

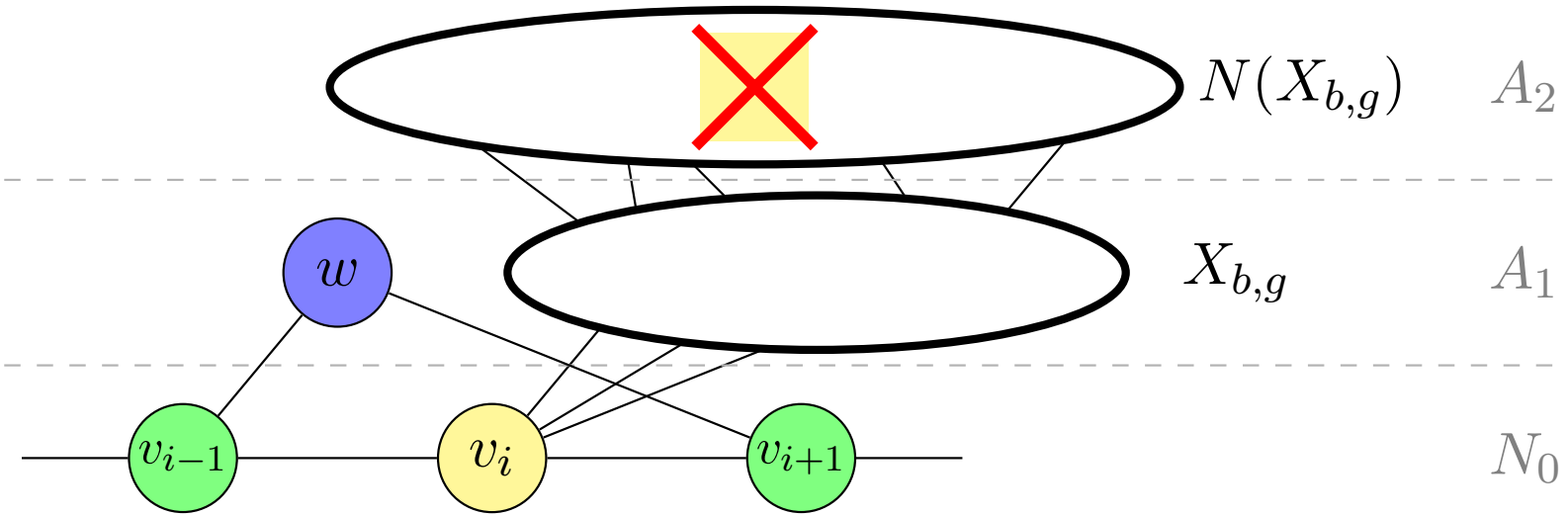


Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

- Remove yellow color from all vertices in $N(X_{b,g}) \cap A_2$
  $\rightarrow X_{b,g} = \emptyset$

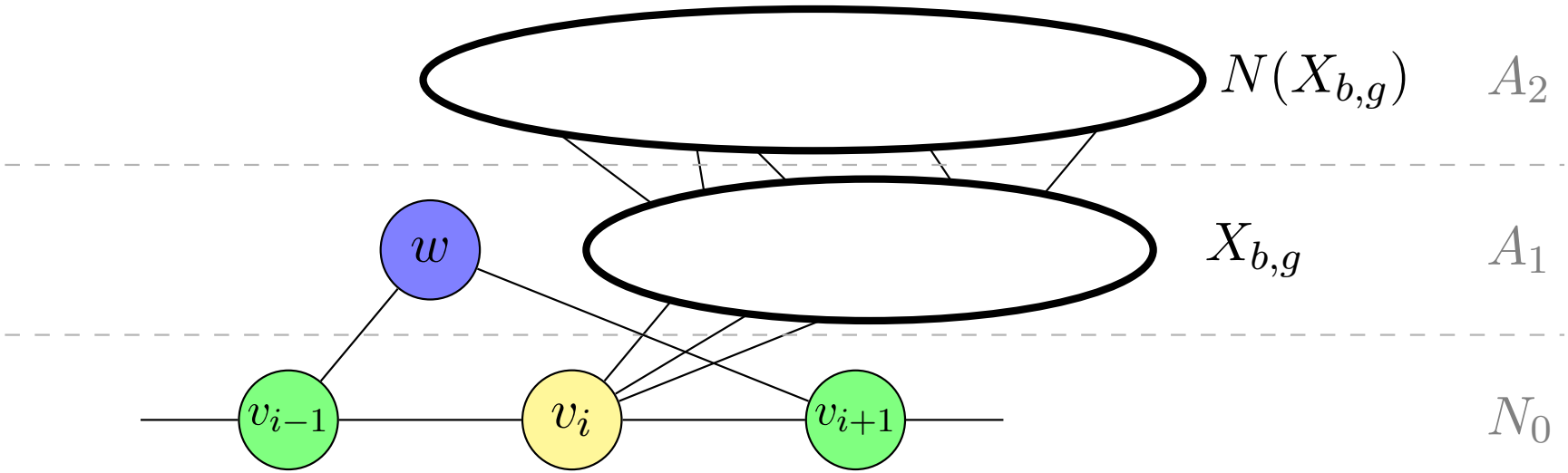


Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

- Remove yellow color from all vertices in $N(X_{b,g}) \cap A_2$
  $\rightarrow X_{b,g} = \emptyset$

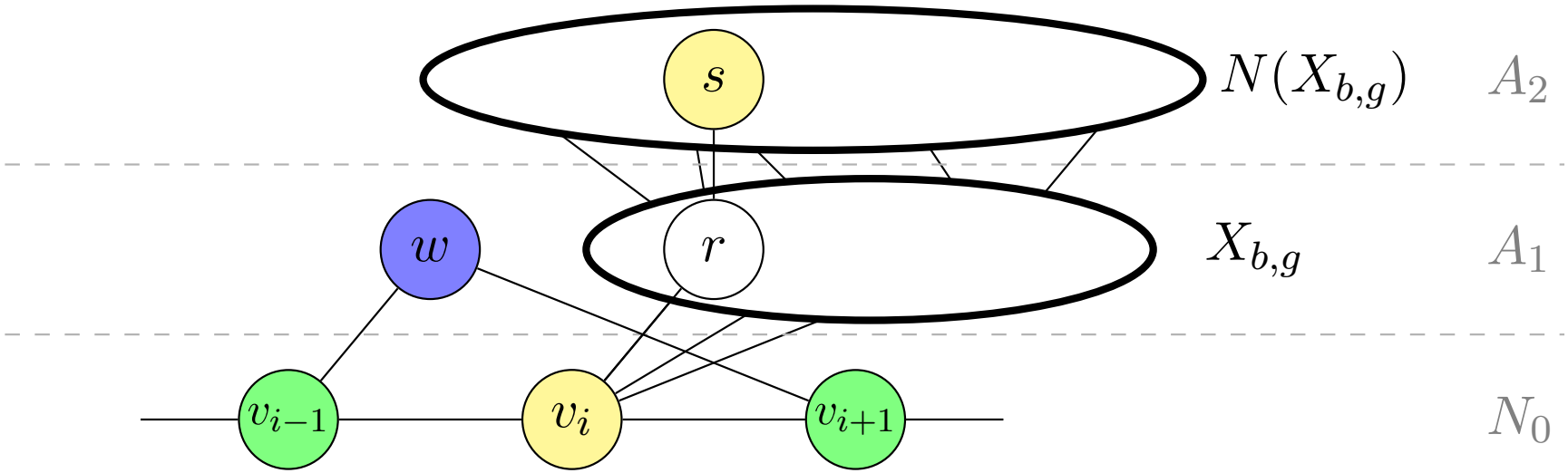


Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

- Pick vertex s in $N(X_{b,g}) \cap A_2$, give it yellow color.

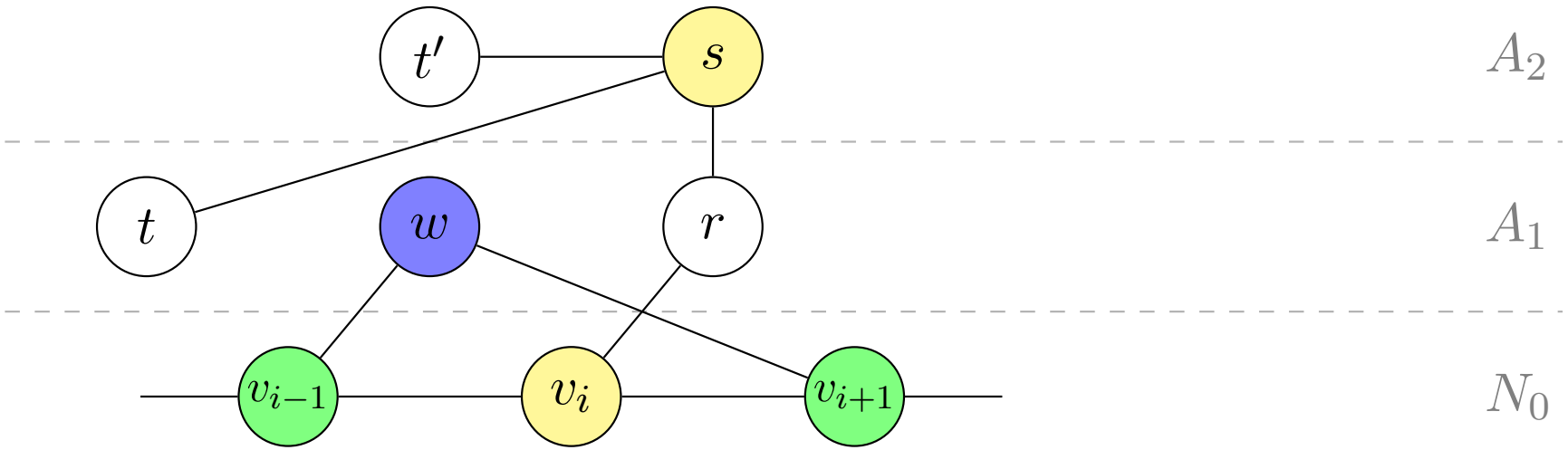


Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

- s has two other neighbors t, t'.

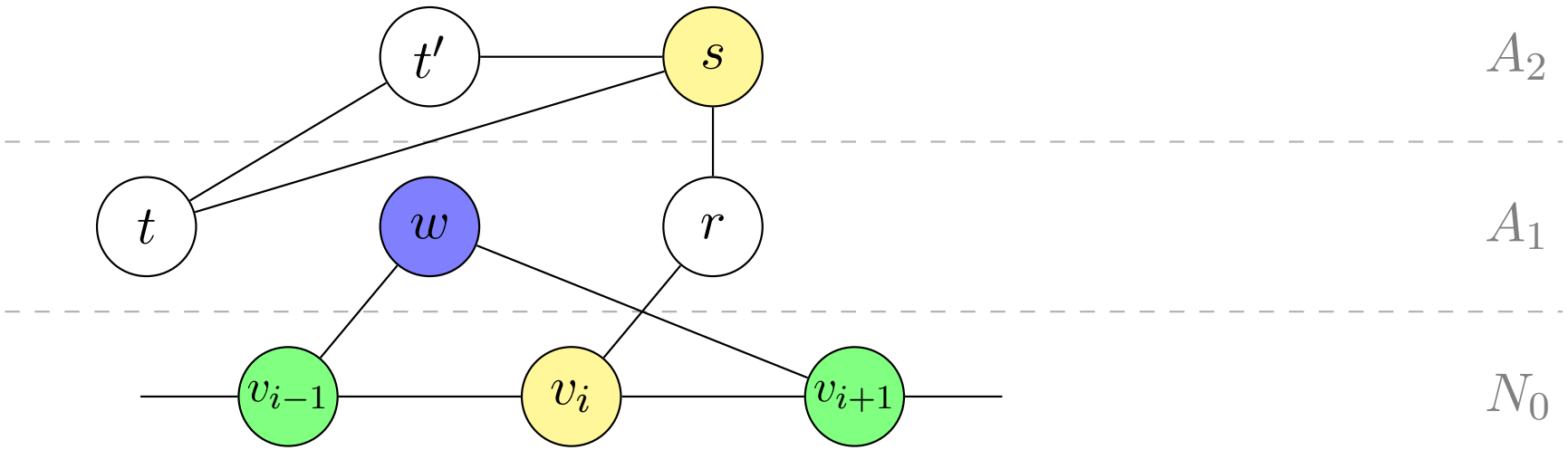


Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

- s has two other neighbors t, t'.

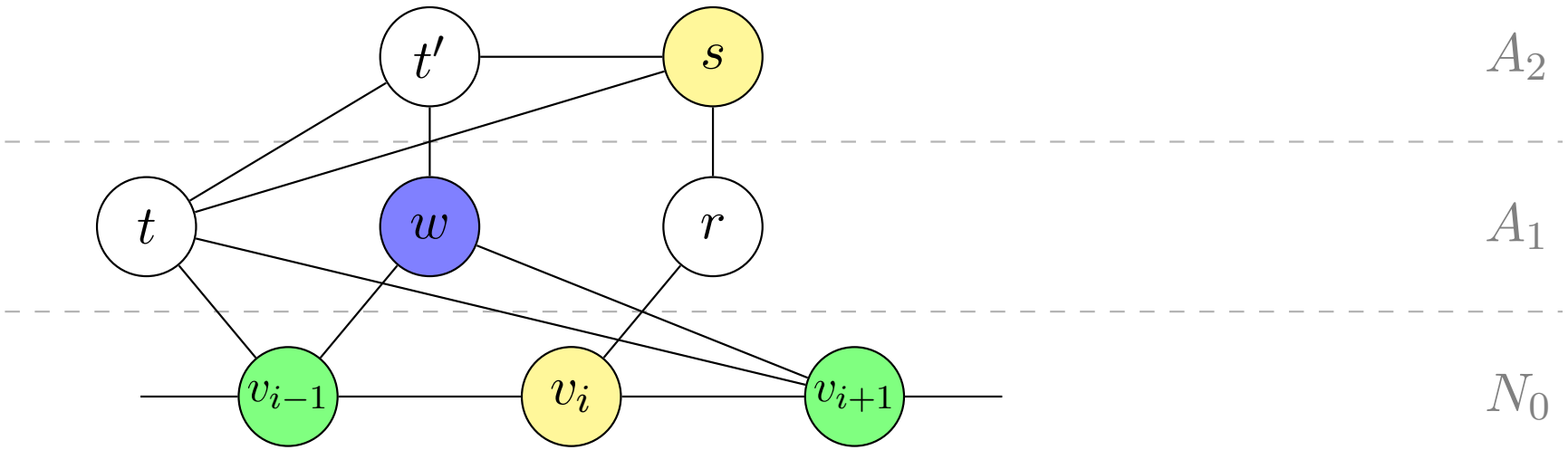


Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

- s has two other neighbors t, t'.

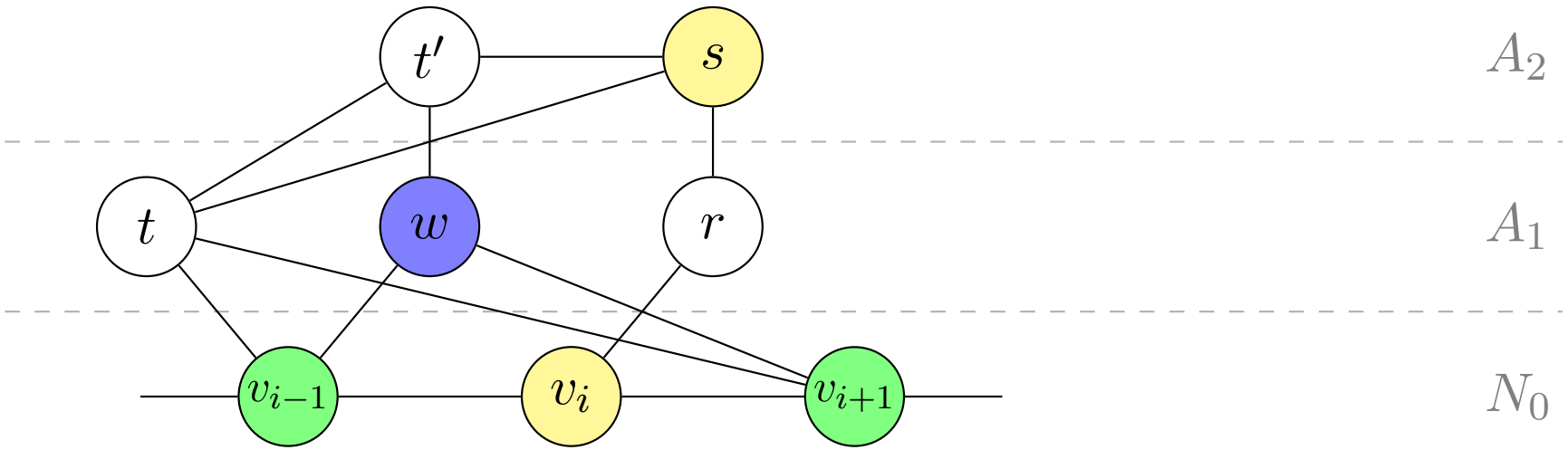


Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

- s has two other neighbors t, t'.

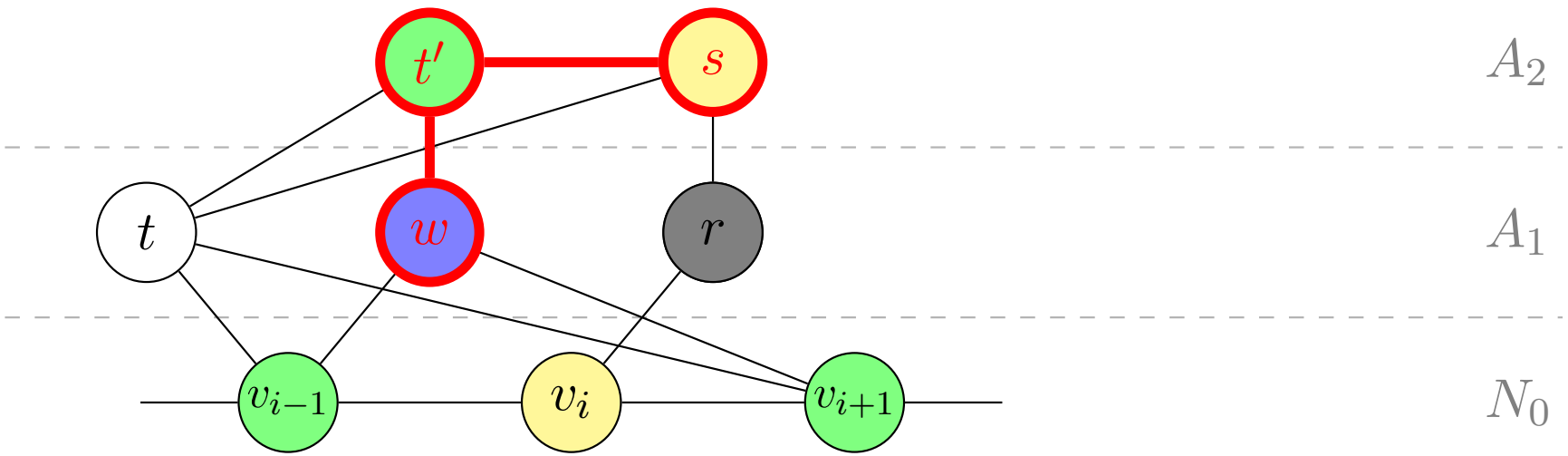


Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

- Find an induced P_3, color it and r.

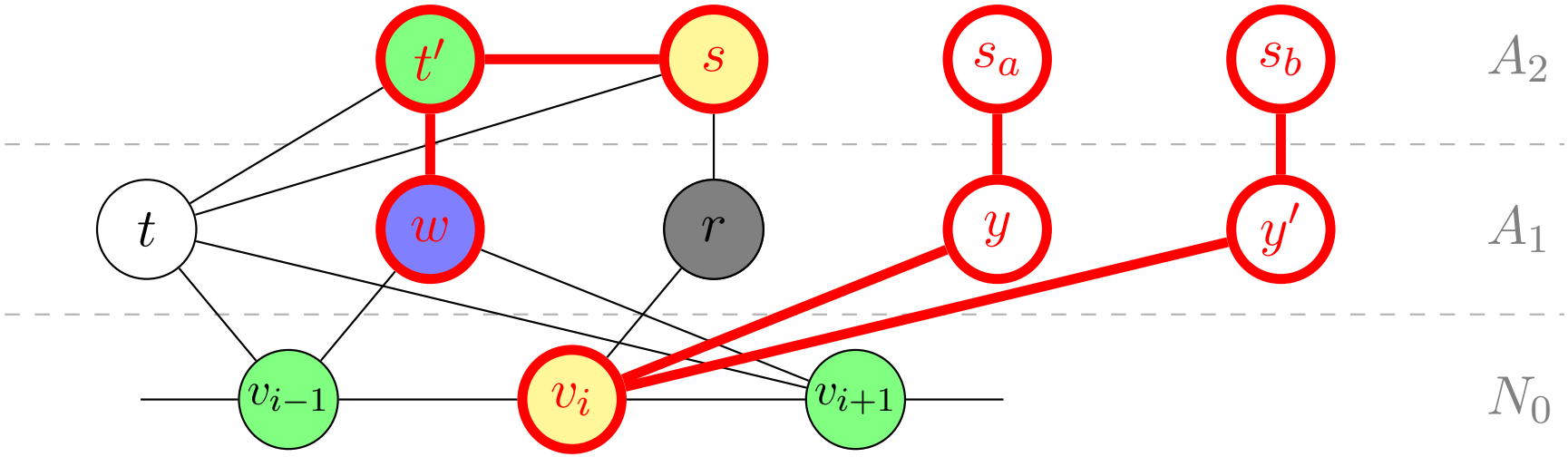


Phase 2: Branching

Goal

At least one of $X_{b,y}$ or $X_{b,g}$ is empty.

- If $|X_{b,g}| > 1$, find an induced P_5,
- otherwise event. color remaining one vertex in $X_{b,g}$
  $\rightarrow X_{b,g} = \emptyset$.



Overview

- Preprocessing
- Phase 1: Reduce private neighbours of non-adjacent vertices in P_7
- Phase 2: Two types of lists in the first layer
- Phase 3: One type of lists in the first layer

Phase 3: Same lists in A_1

$(P_2 + P_5)$-free graphs

- N_2 is an independent set.

Phase 3: Same lists in A_1

$(P_2 + P_5)$-free graphs

- N_2 is an independent set.
- Colour all vertices in A_2 by the colour which is not in A_1.

Phase 3: Same lists in A_1

$(P_2 + P_5)$-free graphs

- N_2 is an independent set.
- Colour all vertices in A_2 by the colour which is not in A_1.

$(P_3 + P_4)$-free graphs

- We need to do more branching.

Thank you for your attention!