Comparaison de réseaux biologiques

graphe orienté $vs.$ graphe non orienté

Présenté par: H.MOHAMED BABOU

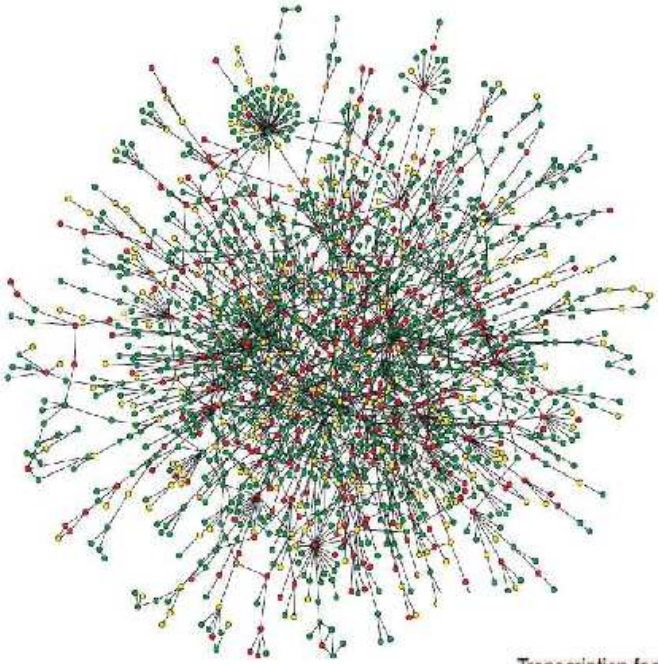
Encadré par: G. FERTIN & I. RUSU

Lundi 10 janvier 2011

Plan

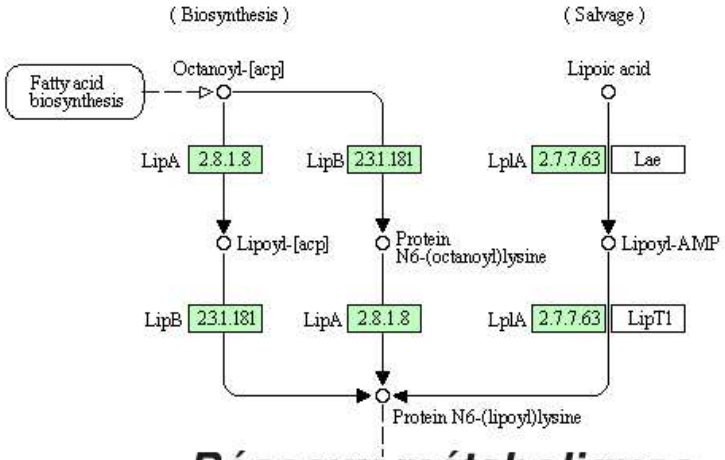
- Réseaux biologiques
- Nouvelle approche : LPDAG-BIJ
- NP-complétude et inapproximabilité
- Une heuristique & algorithme exact
- Conclusion & perspectives

Réseaux biologiques



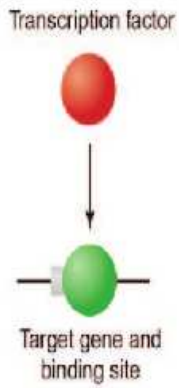
PPI

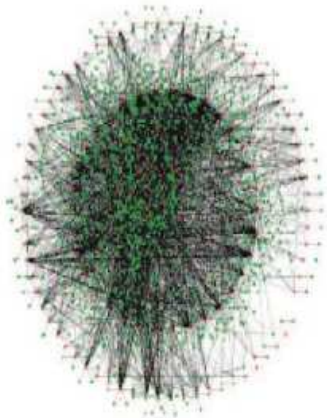
H. Jeong et al. [2001]



Réseaux métaboliques

***KEGG, pathway* eco00785**





Régulation des gènes

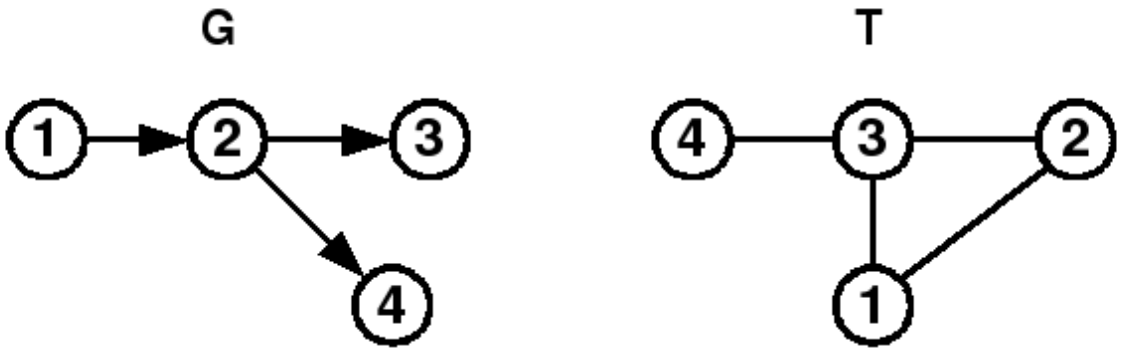
M.M. Babu et al. [2004]

- **Input :** Un graphe orienté acyclique $G = (V, A)$, un graphe non orienté $T = (V, E)$, un arc $x \to y$ de G.
- **Output :** Le plus long chemin (simple) dans G qui contient $x \to y$ et induisant un sous-graphe connexe dans T.

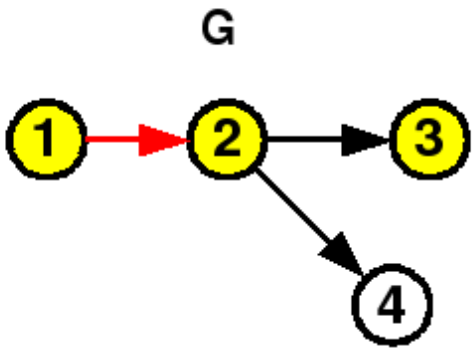
- **Input :** Un graphe orienté acyclique $G = (V, A)$, un graphe non orienté $T = (V, E)$, un arc $x \to y$ de G.
- **Output :** Le plus long chemin (simple) dans G qui contient $x \to y$ et induisant un sous-graphe connexe dans T.

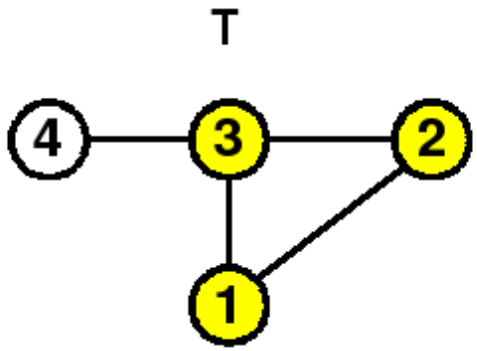
Définition. Un chemin dans G est dit **consistent** s'il induit dans T un sous-graphe connexe.

Exemple 1



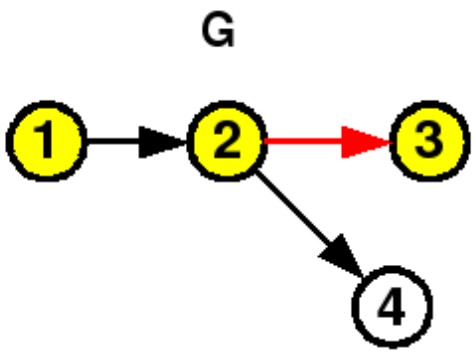
Exemple 1

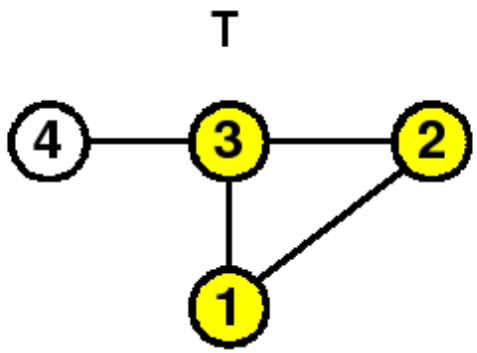




$\mathbf{1} \rightarrow \mathbf{2} : 1 \rightarrow 2 \rightarrow 3$

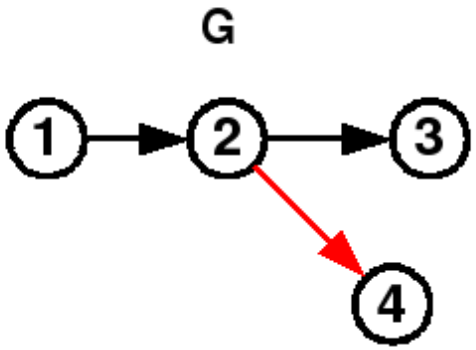
Exemple 1

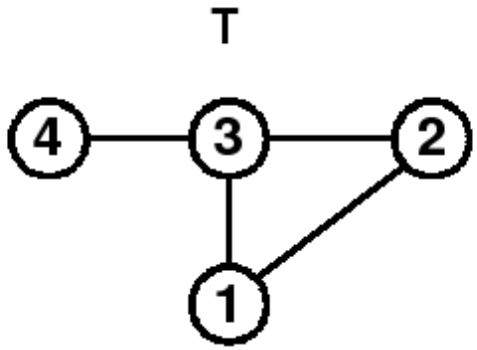




$\mathbf{2} \rightarrow \mathbf{3} : 1 \rightarrow 2 \rightarrow 3$

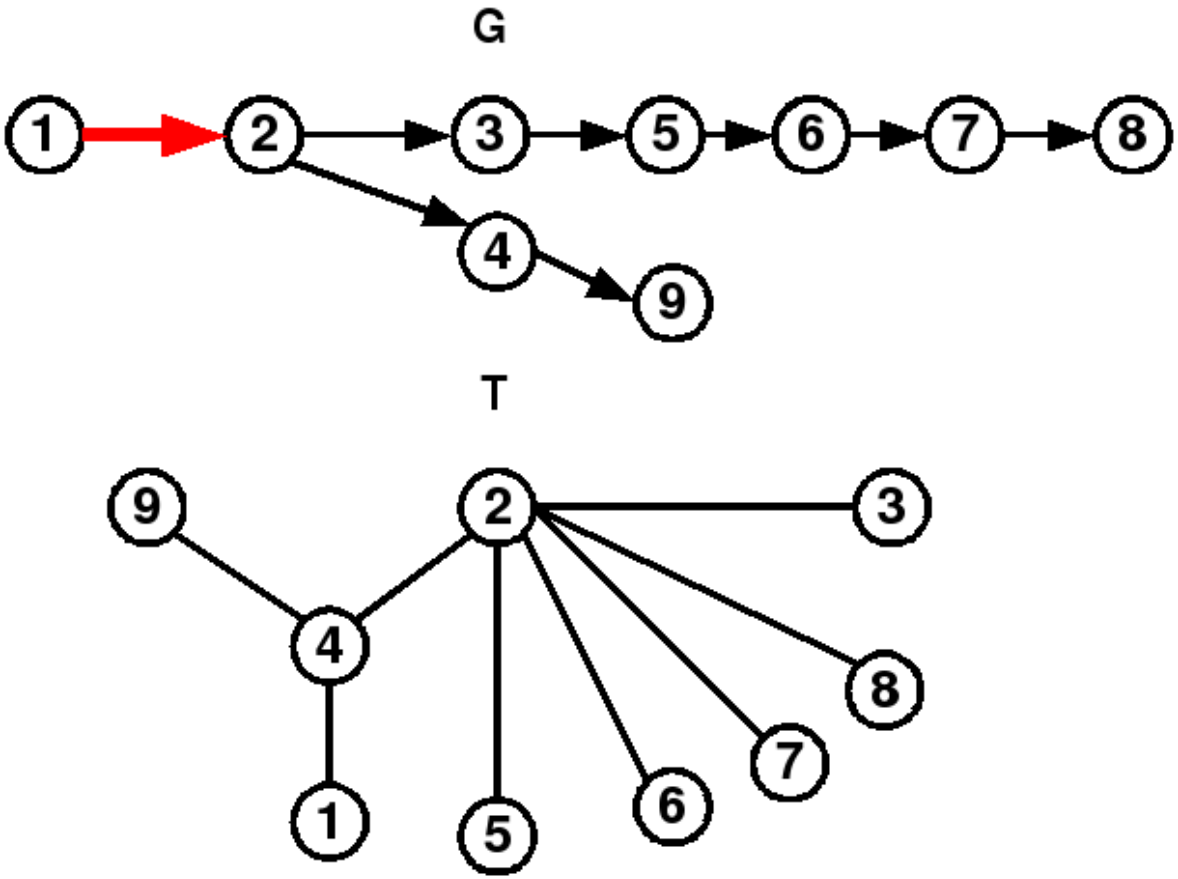
Exemple 1



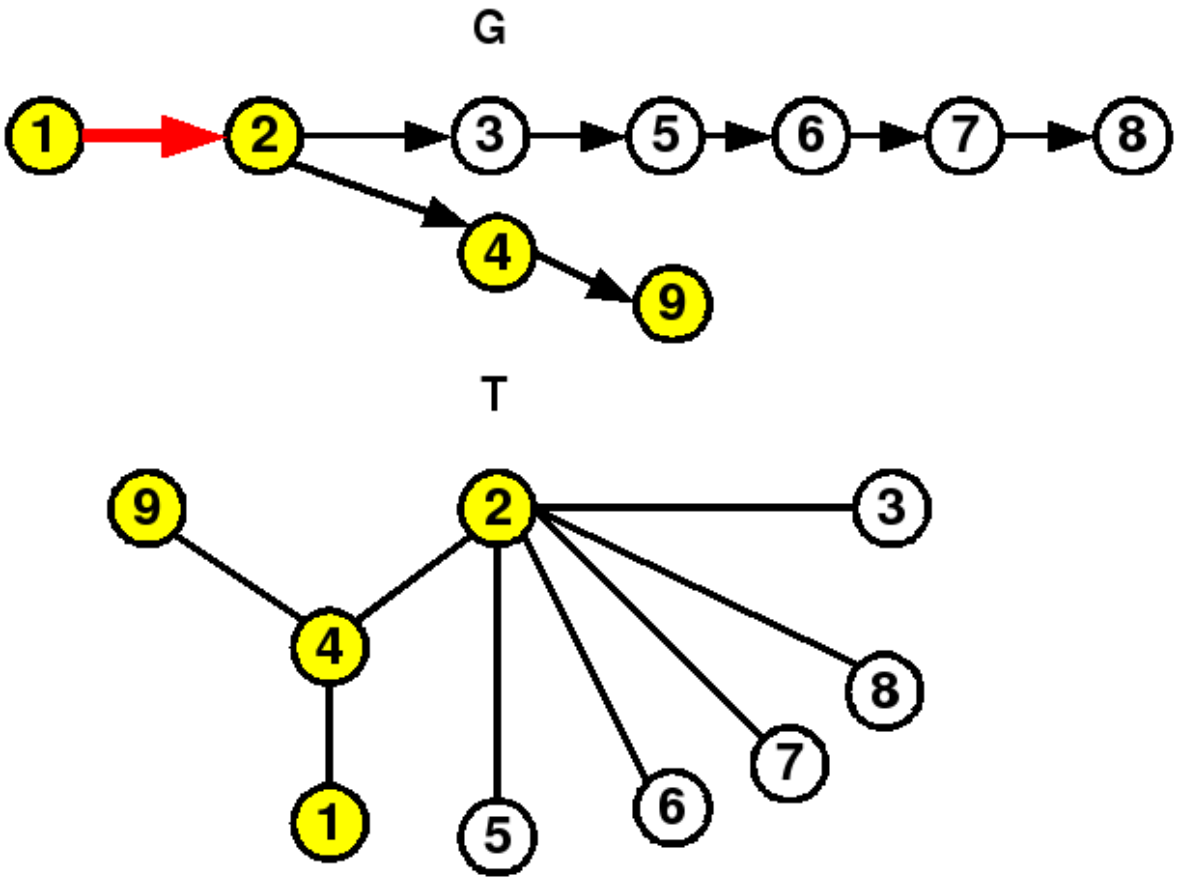


$\mathbf{2} \rightarrow \mathbf{4} : \phi$

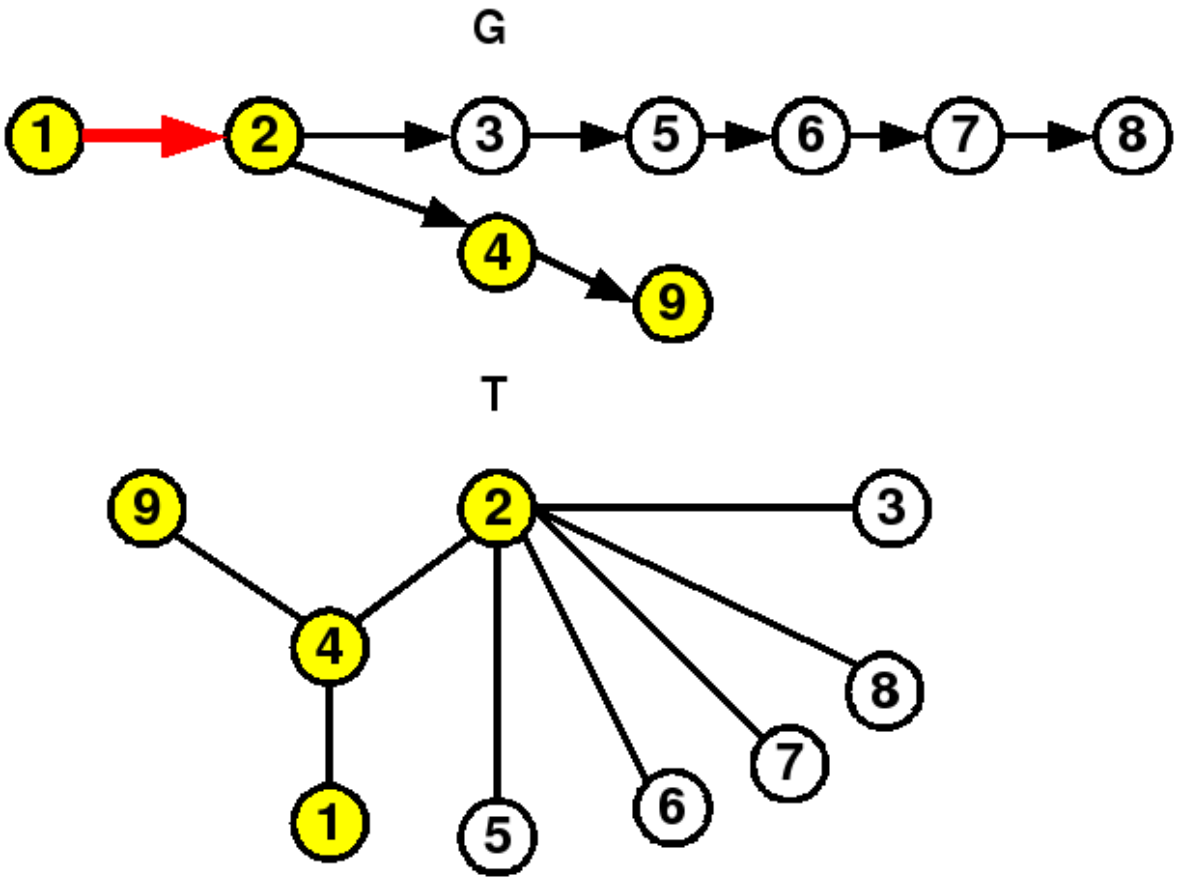
Exemple 2



Exemple 2



Exemple 2



$\mathbf{1} \rightarrow \mathbf{2} : 1 \rightarrow 2 \rightarrow 3$

Le plus long chemin consistent $\neq$ Le plus long chemin dans G.

Le plus long chemin consistent n'est pas un **sous chemin** du plus long chemin dans G.

LPdag-BIJ : version de décision

- **Input :** Un graphe orienté acyclique $G = (V, A)$, un graphe non orienté $T = (V, E)$, un arc $x \to y$ de G, un entier k.
- **Question :** Existe il un chemin consistent (simple) contenant $x \to y$ et de taille au moins k ?

LPdag-BIJ : version de décision

Théorème : LPdag-BIJ est NP-complet même si le graphe T est un arbre de diamètre 4.

LPDAG-BIJ : version de décision

Théorème : LPDAG-BIJ est NP-complet même si le graphe T est un arbre de diamètre 4.

- LPDAG-BIJ est dans NP.

3-SAT : Soit F une formule booléenne de clauses $C_i = (x_{i,1}, x_{i,2}, x_{i,3})$, $1 \leq i \leq p$, sur l'ensemble des littéraux $\{l_1, l_2, \ldots, l_n\}$. Existe il une affectation des littéraux $\{l_1, l_2, \ldots, l_n\}$ satisfaisant toutes les clauses C_i ?

On construit la formule booléenne F' de clauses :

- toutes les clauses de F

On construit la formule booléenne F' de clauses :

- toutes les clauses de F
- toutes les clauses $C_{p+j} = (x_{p+j,1}, x_{p+j,2}, x_{p+j,3}) = (l_j, l_j, \overline{l_j})$ pour tout $1 \leq j \leq n$

On construit la formule booléenne F' de clauses :

- toutes les clauses de F
- toutes les clauses $C_{p+j} = (x_{p+j,1}, x_{p+j,2}, x_{p+j,3}) = (l_j, l_j, \overline{l_j})$ pour tout $1 \leq j \leq n$

Soit l_{n+1} un nouveau littéral. On ajoute à F' les deux clauses :

LPDAG-BIJ : réduction de *3-SAT*

On construit la formule booléenne F' de clauses :

- toutes les clauses de F
- toutes les clauses $C_{p+j} = (x_{p+j,1}, x_{p+j,2}, x_{p+j,3}) = (l_j, l_j, \overline{l_j})$ pour tout $1 \leq j \leq n$

Soit l_{n+1} un nouveau littéral. On ajoute à F' les deux clauses :

- $C_{p+n+1} = (x_{p+n+1,1}, x_{p+n+1,2}, x_{p+n+1,3}) = (l_{n+1}, l_{n+1}, l_{n+1})$

On construit la formule booléenne F' de clauses :

- toutes les clauses de F
- toutes les clauses $C_{p+j} = (x_{p+j,1}, x_{p+j,2}, x_{p+j,3}) = (l_j, l_j, \overline{l_j})$ pour tout $1 \leq j \leq n$

Soit l_{n+1} un nouveau littéral. On ajoute à F' les deux clauses :

- $C_{p+n+1} = (x_{p+n+1,1}, x_{p+n+1,2}, x_{p+n+1,3}) = (l_{n+1}, l_{n+1}, l_{n+1})$
- $C_{p+n+2} = (x_{p+n+2,1}, x_{p+n+2,2}, x_{p+n+2,3}) = (l_{n+1}, l_{n+1}, l_{n+1})$

$p = 2$ et $n = 3$
$L = \{l_1, l_2, l_3\}$, $F = (l_1 \vee \bar{l_2} \vee l_3) \wedge (l_1 \vee \bar{l_2} \vee \bar{l_3})$

$F' = (l_1 \vee \bar{l_2} \vee l_3) \wedge (l_1 \vee \bar{l_2} \vee \bar{l_3}) \wedge$
$(l_1 \vee l_1 \vee \bar{l_1}) \wedge (l_2 \vee l_2 \vee \bar{l_2}) \wedge (l_3 \vee l_3 \vee \bar{l_3}) \wedge$
$(l_4 \vee l_4 \vee l_4) \wedge (l_4 \vee l_4 \vee l_4)$

$P = p + n + 2 = 7$: Le nombre des clauses de F'.
$N = n + 1 = 4$: Le nombre des littéraux de F'

réduction de *3-SAT*

A partir de la formule F', on construit une instance du problème $\text{LP}_{\text{DAG}}\text{-BIJ}$.

- un DAG $G = (V, A)$
- un graphe non orienté $T = (V, E)$
- un arc $x \longrightarrow y$ de G
- un entier k

Réduction de *3-SAT* : construction du graphe G

G est construit en $P + N + 1$ niveaux, de 0 à $P + N$.

- niveau 0 : un nœud s.

Réduction de *3-SAT* : construction du graphe G

G est construit en $P + N + 1$ niveaux, de 0 à $P + N$.

- niveau 0 : un nœud s.
- niveau i, $1 \leq i \leq P$: 3 nœuds $x_{i,1}, x_{i,2}, x_{i,3}$ correspondent aux variables de clause C_i.

Réduction de *3-SAT* : construction du graphe G

G est construit en $P + N + 1$ niveaux, de 0 à $P + N$.

- niveau 0 : un nœud s.
- niveau i, $1 \leq i \leq P$: 3 nœuds $x_{i,1}, x_{i,2}, x_{i,3}$ correspondent aux variables de clause C_i.
- niveau $P + j$, $1 \leq j \leq N - 1$: 2 nœuds a_j, b_j.

Réduction de *3-SAT* : construction du graphe G

G est construit en $P + N + 1$ niveaux, de 0 à $P + N$.

- niveau 0 : un nœud s.
- niveau i, $1 \leq i \leq P$: 3 nœuds $x_{i,1}, x_{i,2}, x_{i,3}$ correspondent aux variables de clause C_i.
- niveau $P + j$, $1 \leq j \leq N - 1$: 2 nœuds a_j, b_j.
- level $P + N$: un nœud a_N.

Réduction de *3-SAT* : construction du graphe G

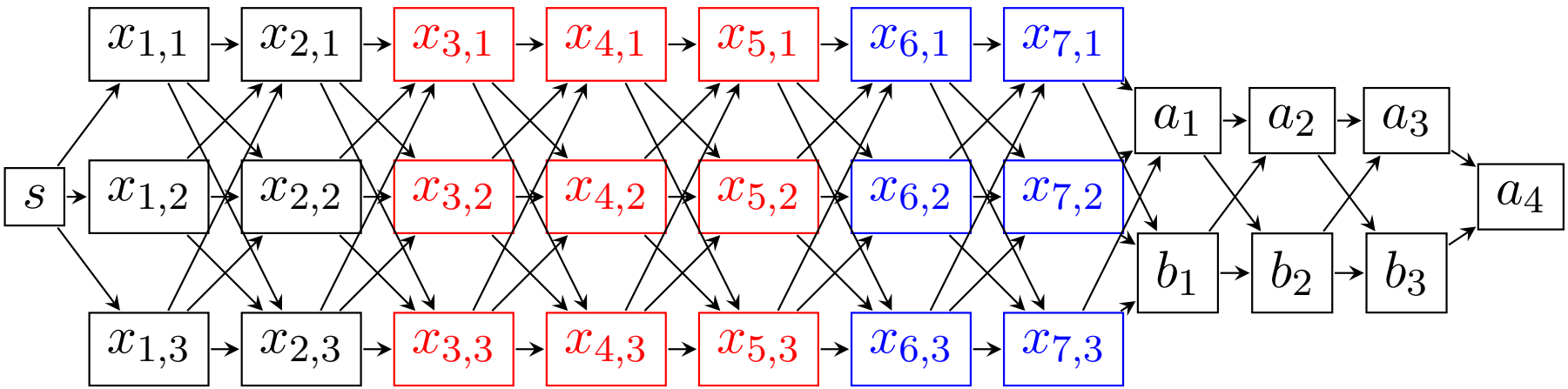
G est construit en $P + N + 1$ niveaux, de 0 à $P + N$.

- niveau 0 : un nœud s.
- niveau i, $1 \leq i \leq P$: 3 nœuds $x_{i,1}, x_{i,2}, x_{i,3}$ correspondent aux variables de clause C_i.
- niveau $P + j$, $1 \leq j \leq N - 1$: 2 nœuds a_j, b_j.
- level $P + N$: un nœud a_N.

On met tous les arcs possibles d'un niveau i au niveau $i + 1$, pour tout $0 \leq i \leq P + N - 1$.

Exemple de construction de G

$L = \{l_1, l_2, l_3\}$, $F = (l_1 \vee \bar{l_2} \vee l_3) \wedge (l_1 \vee \bar{l_2} \vee \bar{l_3})$
$F' = (l_1 \vee \bar{l_2} \vee l_3) \wedge (l_1 \vee \bar{l_2} \vee \bar{l_3}) \wedge$
$(l_1 \vee l_1 \vee \bar{l_1}) \wedge (l_2 \vee l_2 \vee \bar{l_2}) \wedge (l_3 \vee l_3 \vee \bar{l_3}) \wedge$
$(l_4 \vee l_4 \vee l_4) \wedge (l_4 \vee l_4 \vee l_4)$



G

Réduction de *3-SAT* : construction du graphe T

- T est un arbre de même nœuds que G et de racine s
- Il y a une arrête entre s et tous les a_i et b_i, pour tout $1 \leq i \leq N$
- I y a une arrête entre a_i et tous les nœuds $x_{j,k}$ qui correspondent au littéral l_i, pour tout $1 \leq i \leq N$, $1 \leq j \leq P$ et $1 \leq k \leq 3$
- Il y a une arrête entre b_i et tous les nœuds $x_{j,k}$ qui correspondent au littéral $\bar{l_i}$, pour tout $1 \leq i \leq N$, $1 \leq j \leq P$ et $1 \leq k \leq 3$

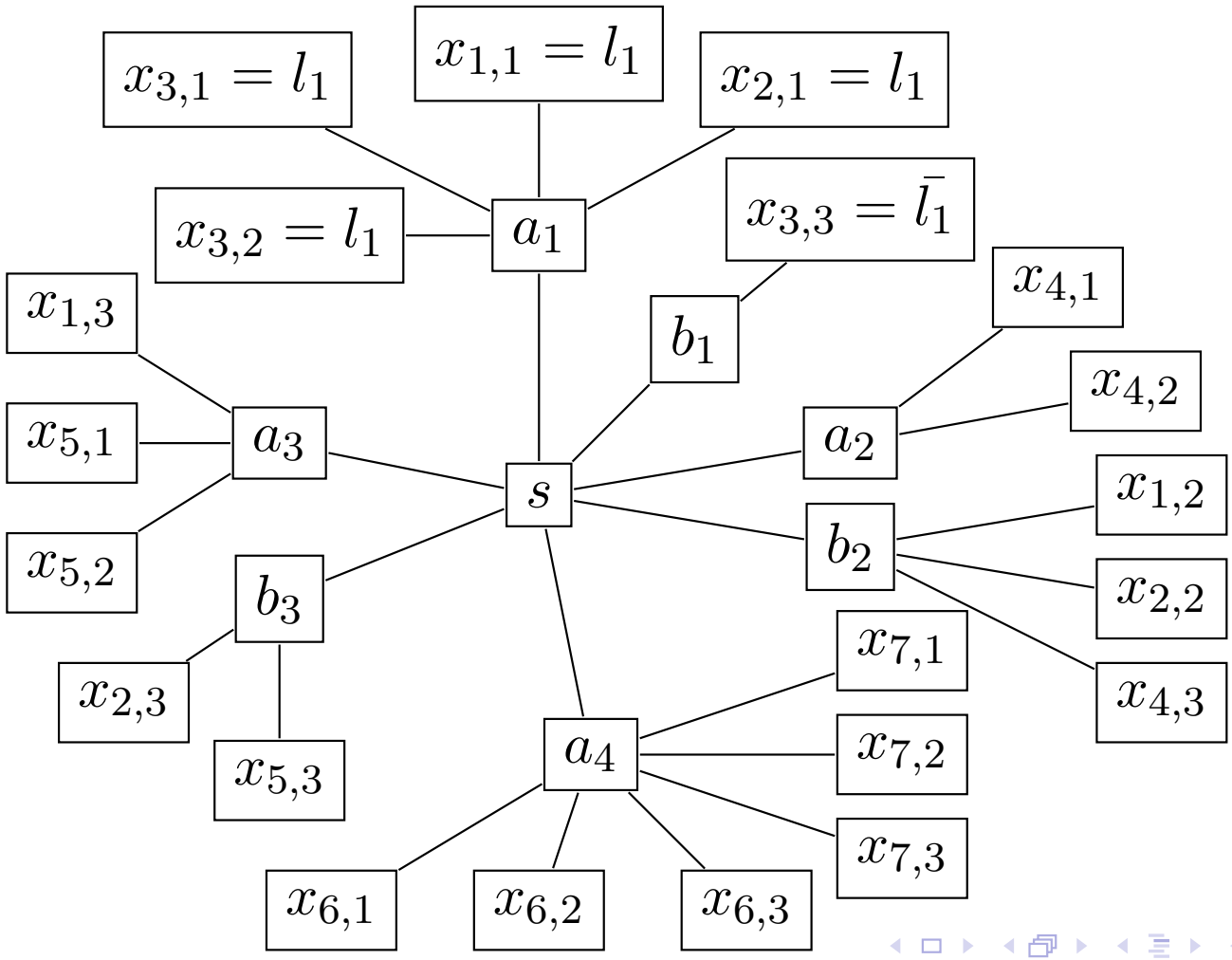
Exemple de construction de T

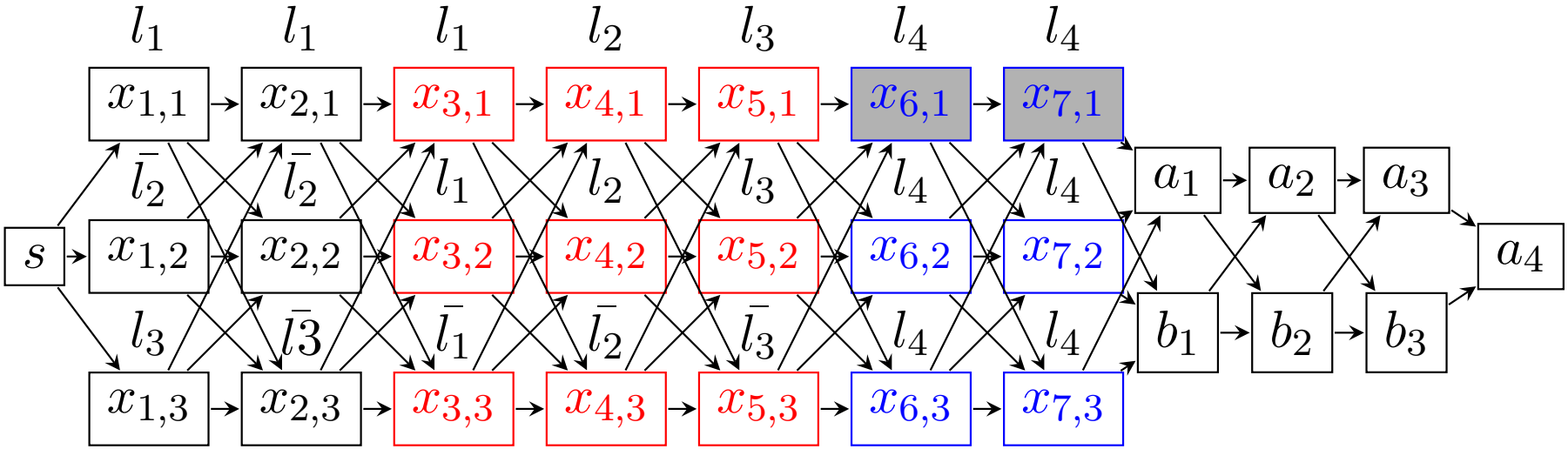
$L = \{l_1, l_2, l_3\}$, $F = (l_1 \vee \bar{l_2} \vee l_3) \wedge (l_1 \vee \bar{l_2} \vee \bar{l_3})$

$F' = (l_1 \vee \bar{l_2} \vee l_3) \wedge (l_1 \vee \bar{l_2} \vee \bar{l_3}) \wedge$

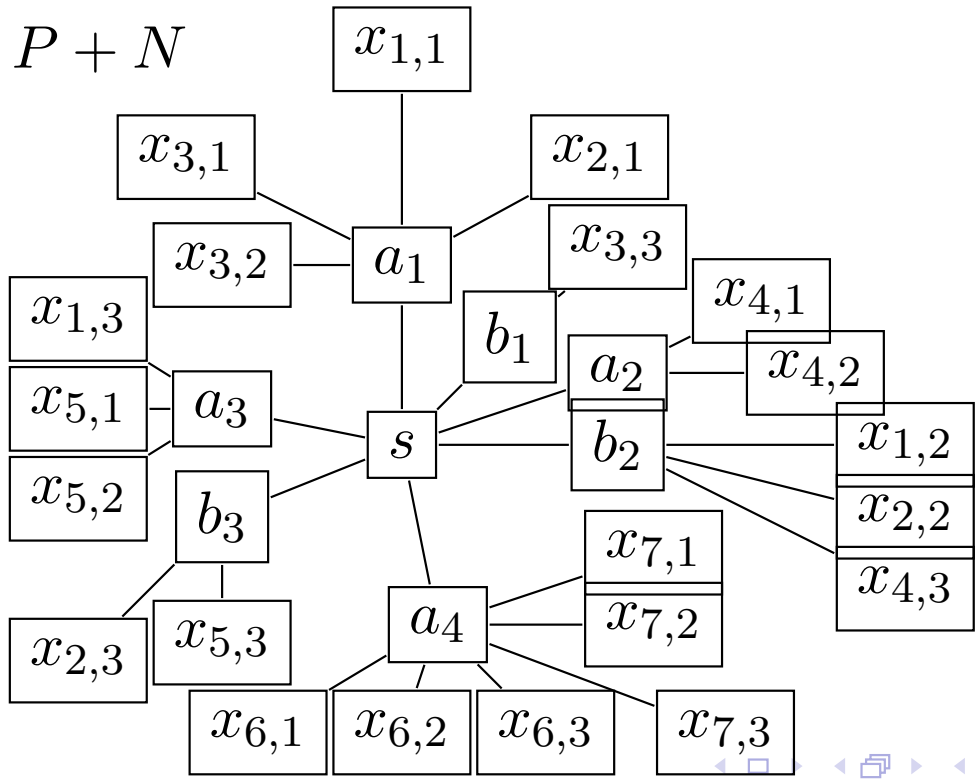
$(l_1 \vee l_1 \vee \bar{l_1}) \wedge (l_2 \vee l_2 \vee \bar{l_2}) \wedge (l_3 \vee l_3 \vee \bar{l_3}) \wedge$

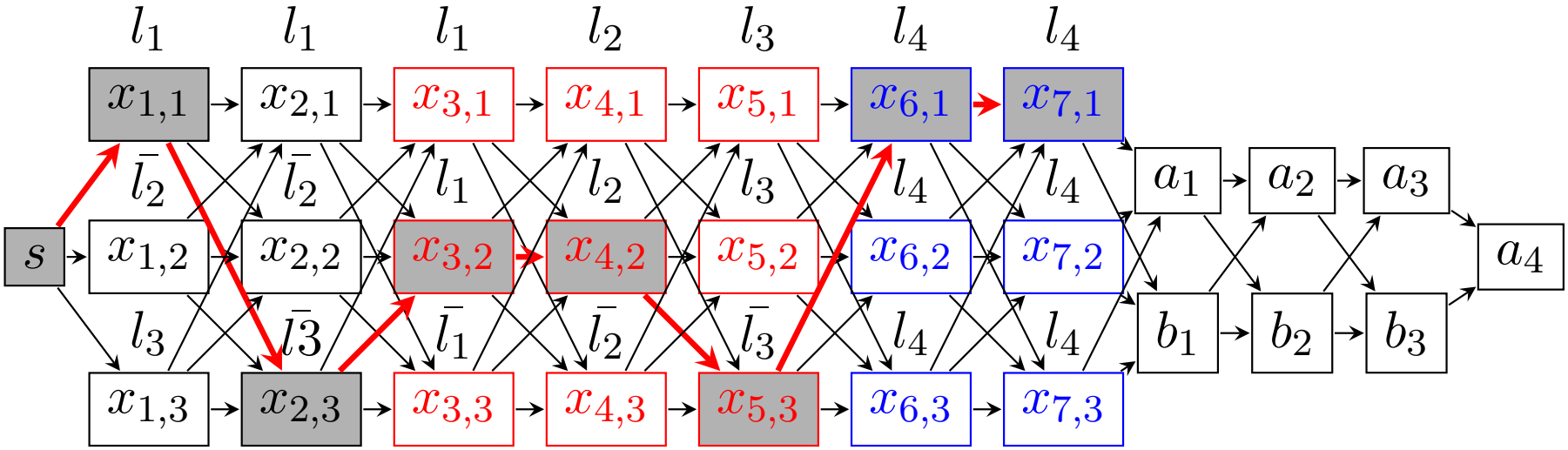
$(l_4 \vee l_4 \vee l_4) \wedge (l_4 \vee l_4 \vee l_4)$



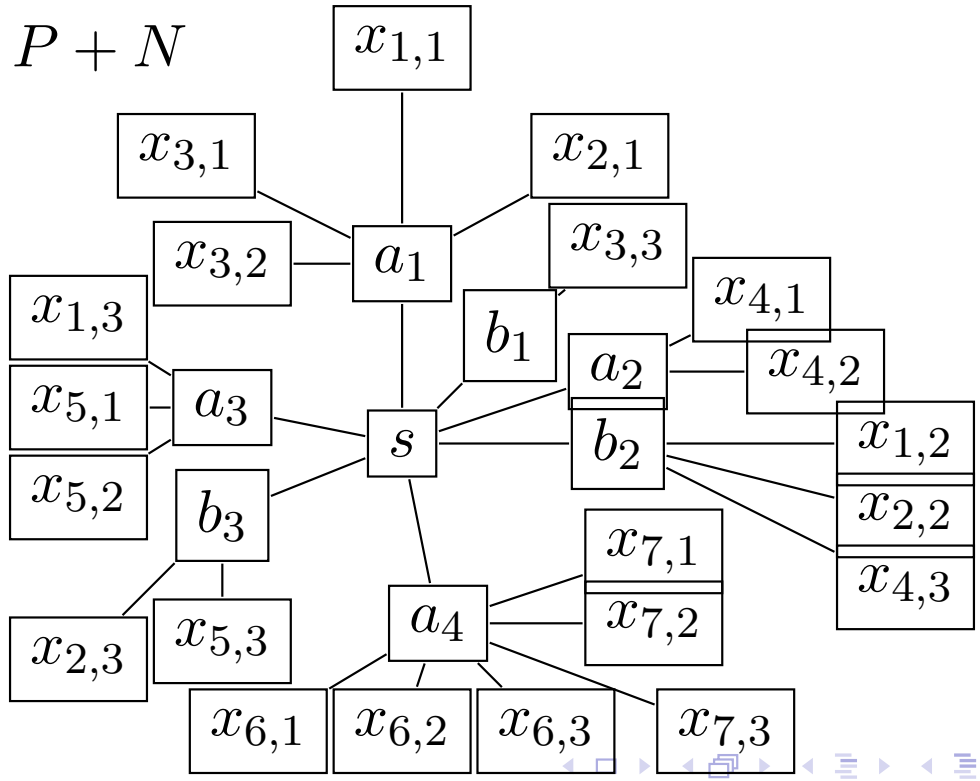

l_1 l_1 l_1 l_2 l_3 l_4 l_4
$x_{1,1}$ $x_{2,1}$ $x_{3,1}$ $x_{4,1}$ $x_{5,1}$ $x_{6,1}$ $x_{7,1}$
$\bar{l}_2$ $\bar{l}_2$ l_1 l_2 l_3 l_4 l_4
s $x_{1,2}$ $x_{2,2}$ $x_{3,2}$ $x_{4,2}$ $x_{5,2}$ $x_{6,2}$ $x_{7,2}$
l_3 $\bar{l}3$ $\bar{l}_1$ $\bar{l}_2$ $\bar{l}_3$ l_4 l_4
$x_{1,3}$ $x_{2,3}$ $x_{3,3}$ $x_{4,3}$ $x_{5,3}$ $x_{6,3}$ $x_{7,3}$
a_1 a_2 a_3 a_4
b_1 b_2 b_3

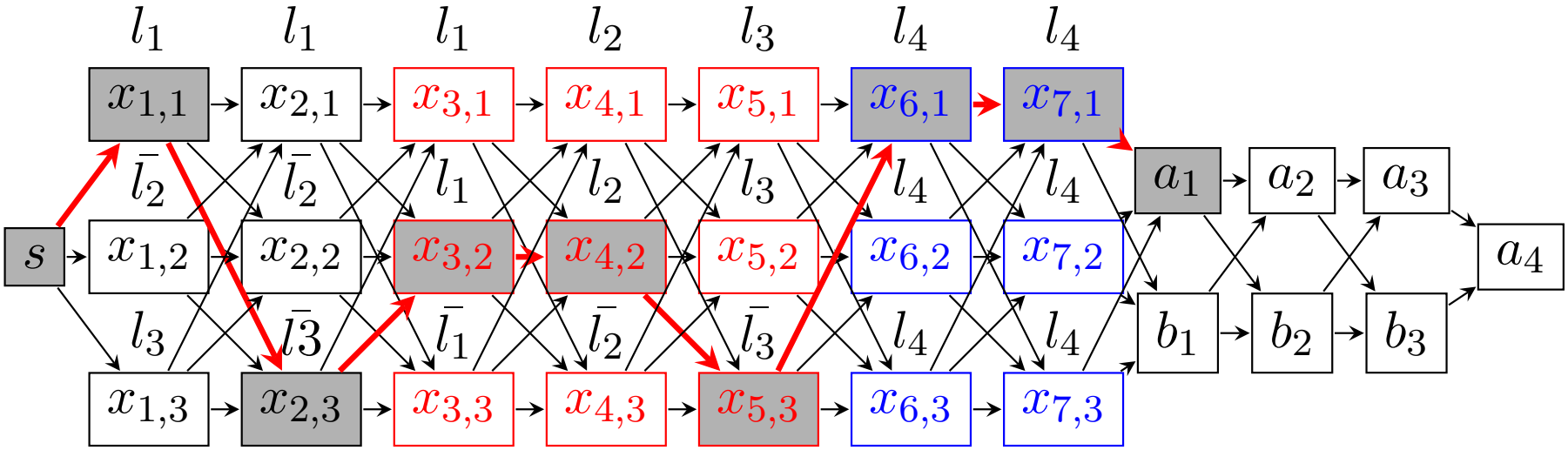
$k = P + N$


$x_{1,1}$
$x_{3,1}$ $x_{2,1}$
$x_{3,2}$ a_1 $x_{3,3}$
$x_{1,3}$ b_1 $x_{4,1}$
$x_{5,1}$ a_3 a_2 $x_{4,2}$
s b_2 $x_{1,2}$
$x_{5,2}$ b_3 $x_{2,2}$
$x_{7,1}$ $x_{4,3}$
$x_{2,3}$ $x_{5,3}$ a_4 $x_{7,2}$
$x_{6,1}$ $x_{6,2}$ $x_{6,3}$ $x_{7,3}$

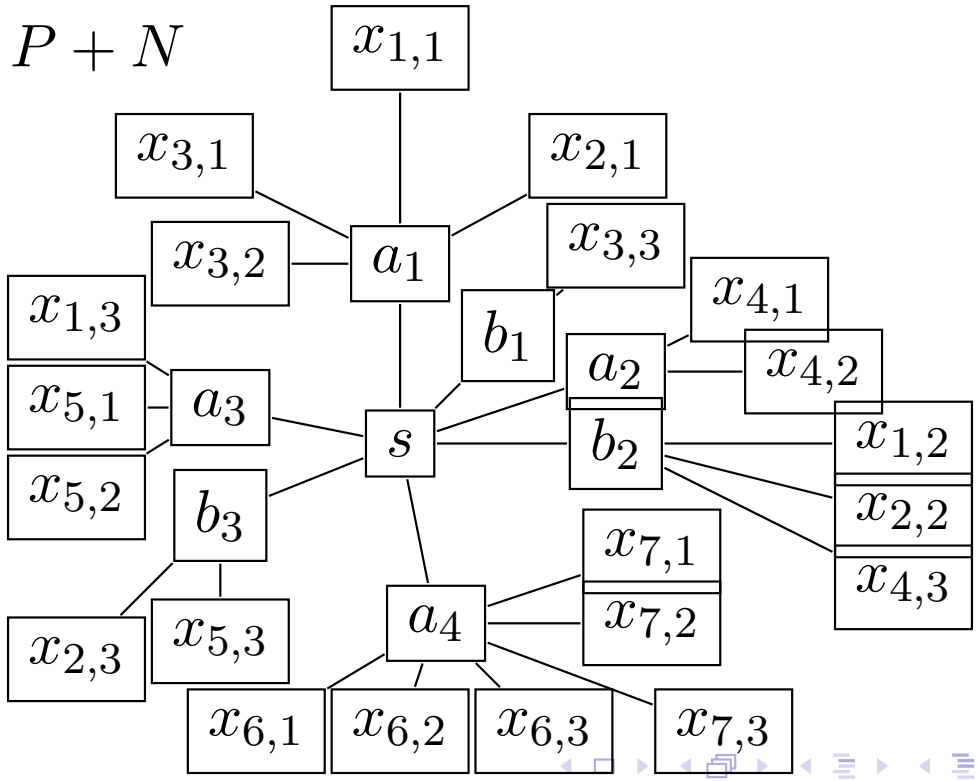


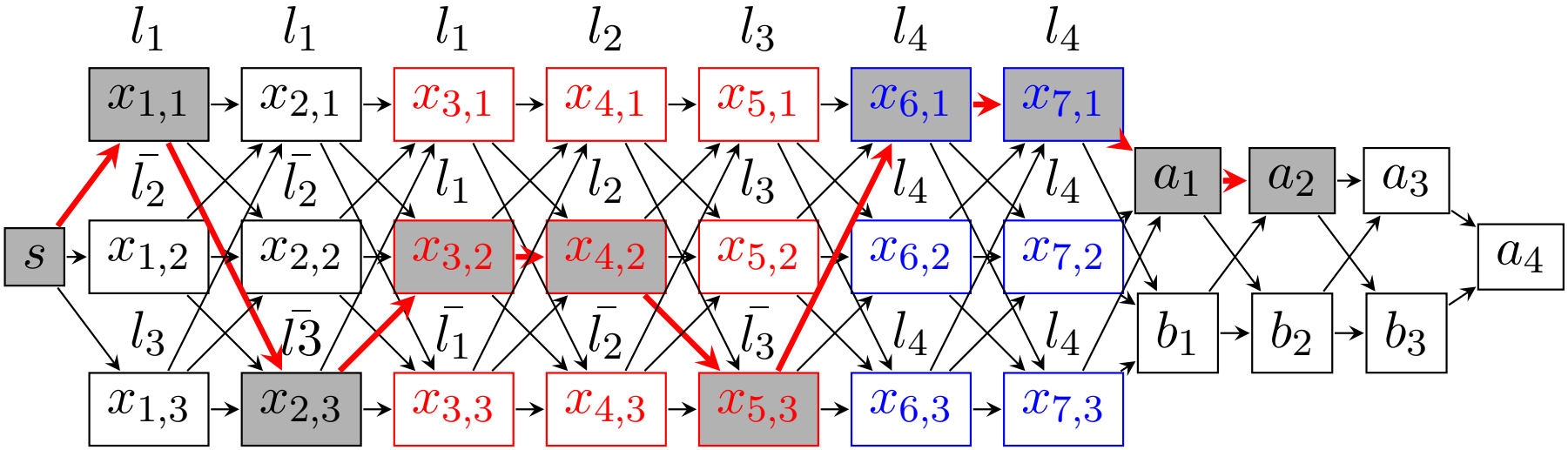
$(\Rightarrow)$ $\quad k = P + N$



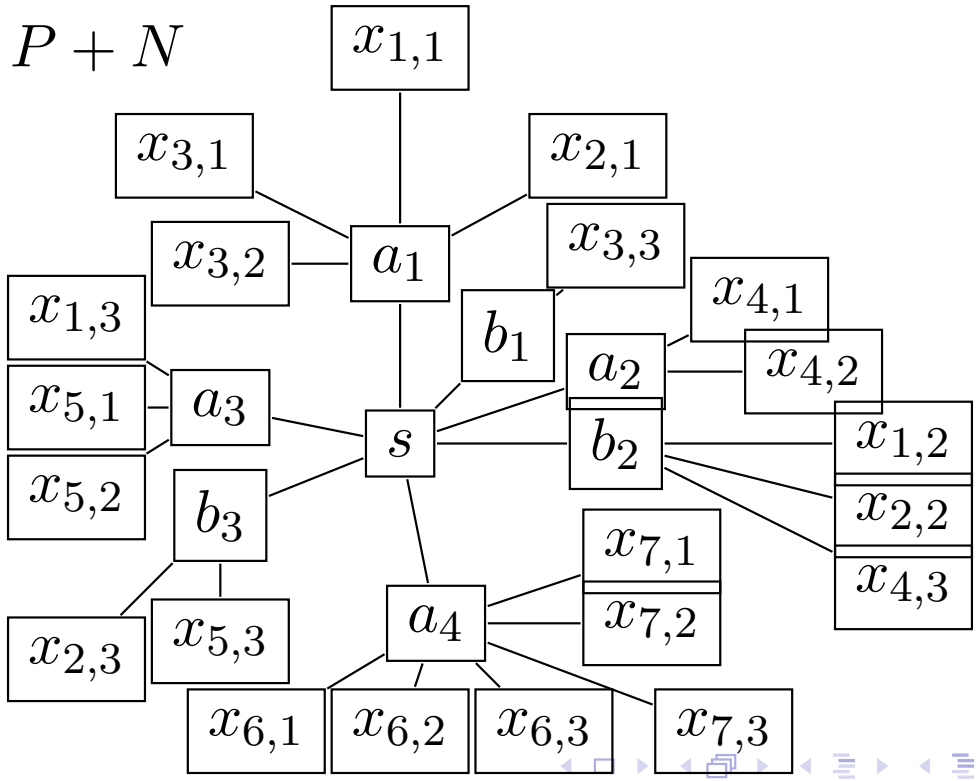


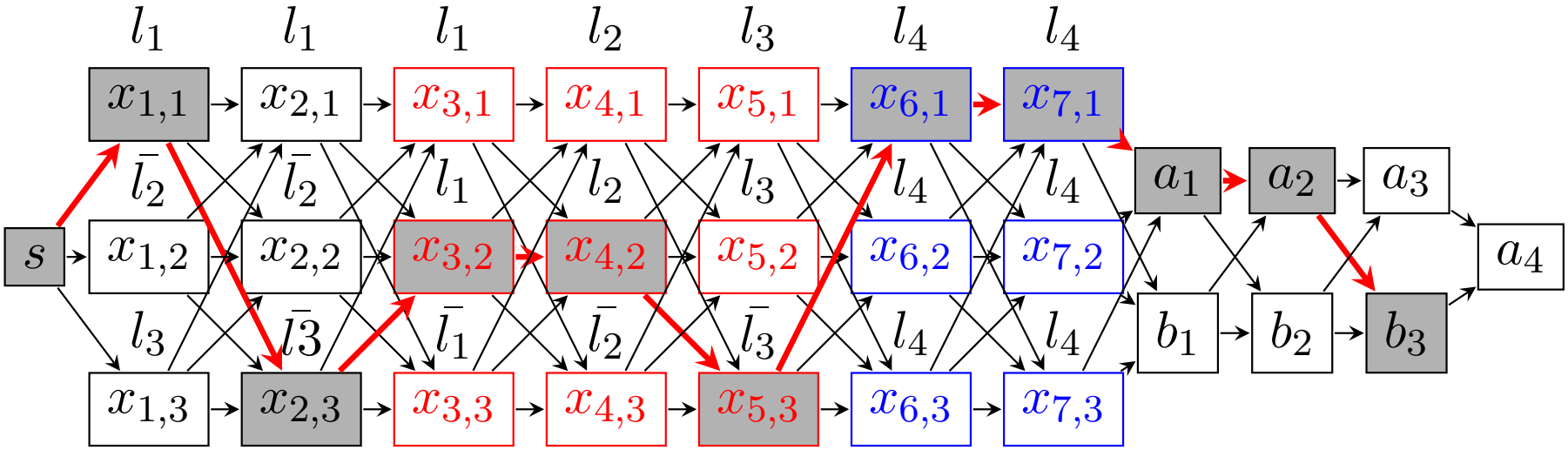
$(\Rightarrow)$ $\quad k = P + N$



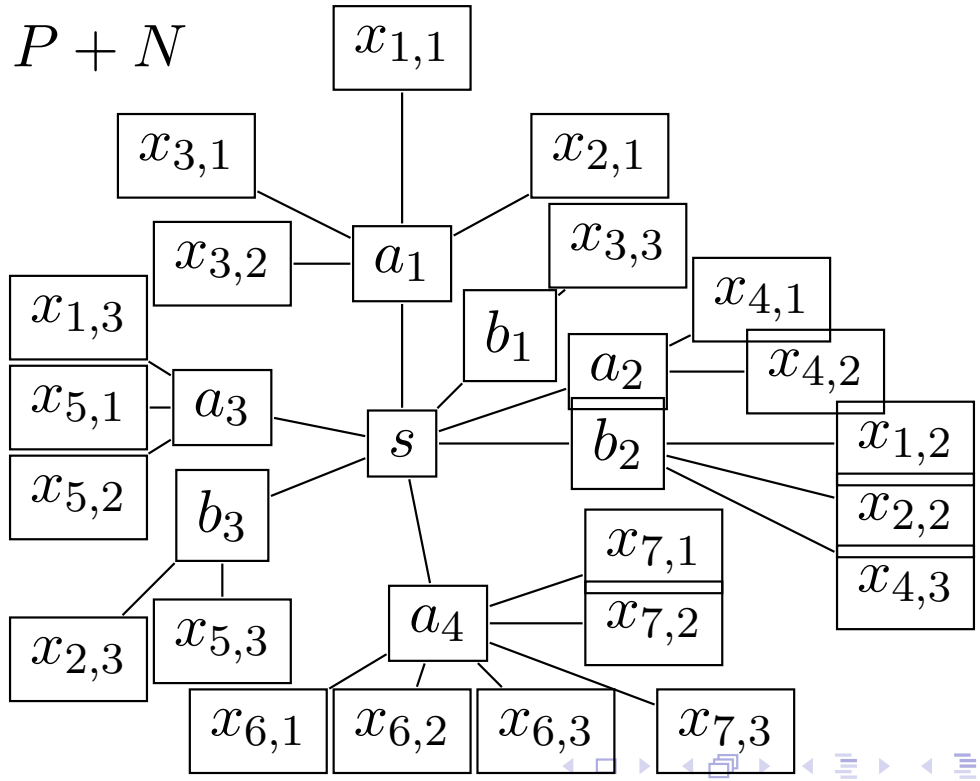


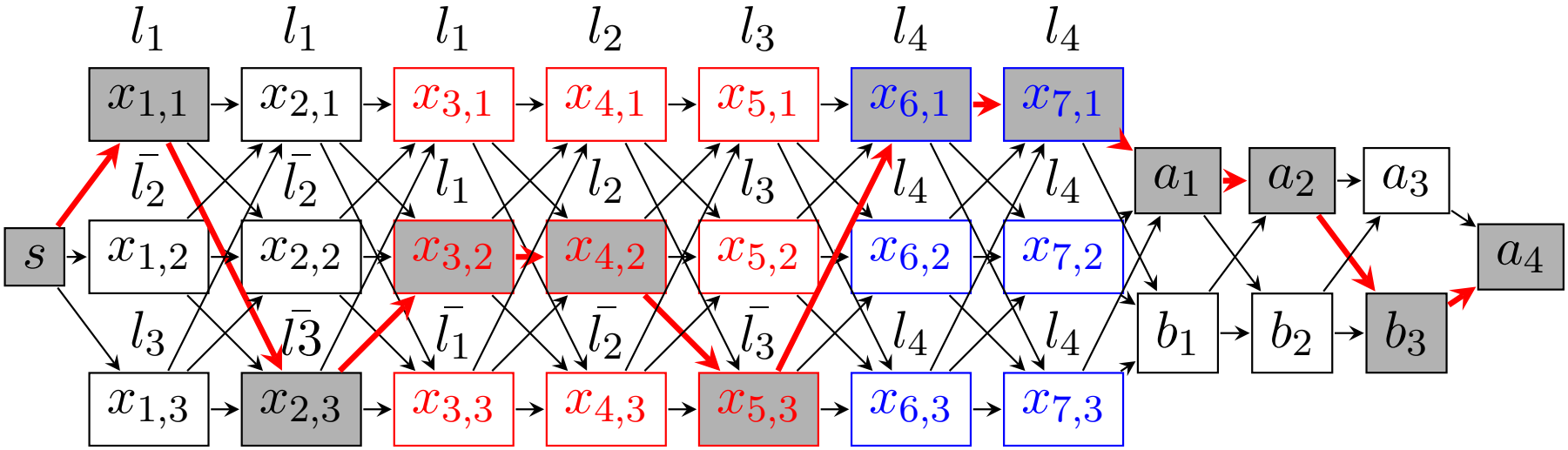
$(\Rightarrow)$ $\quad k = P + N$



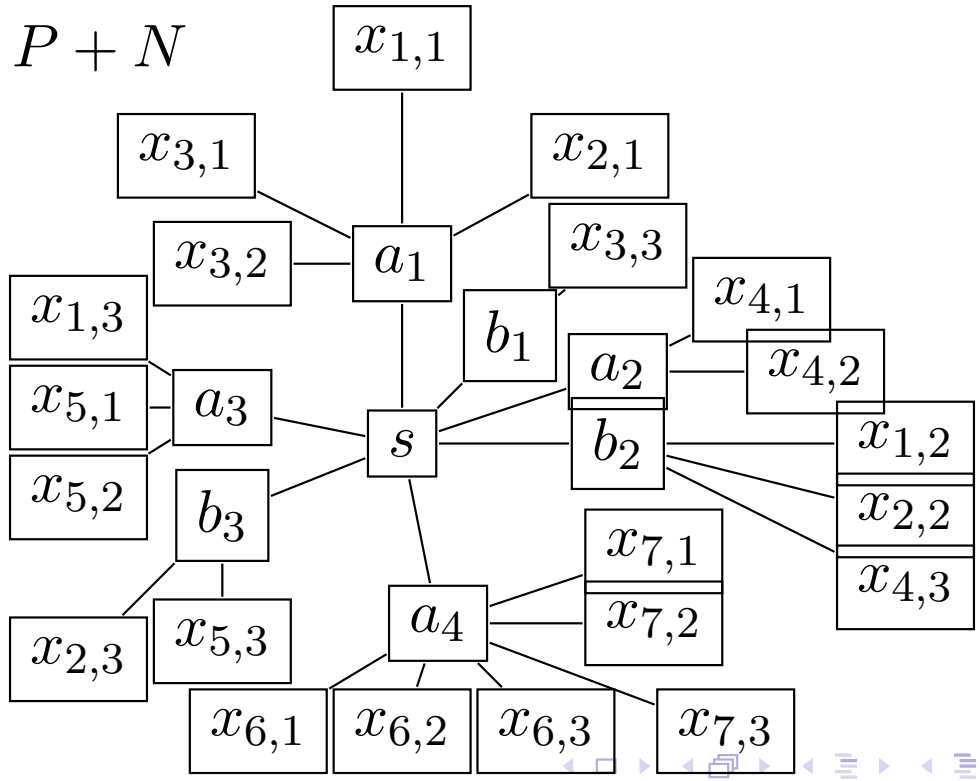


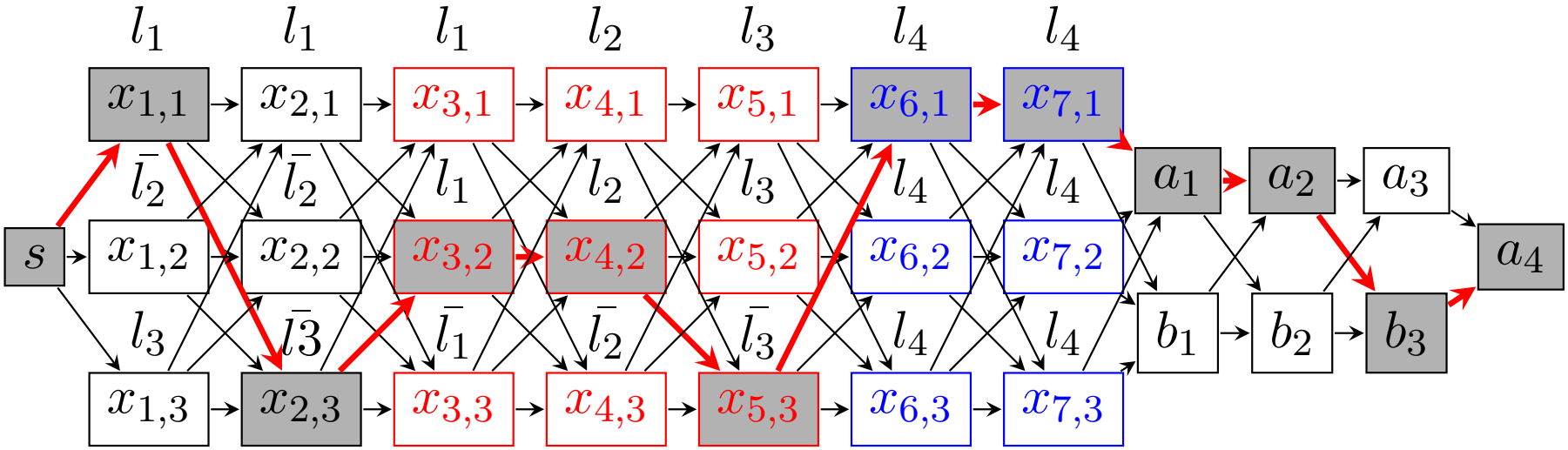
$(\Rightarrow)$ $\quad k = P + N$



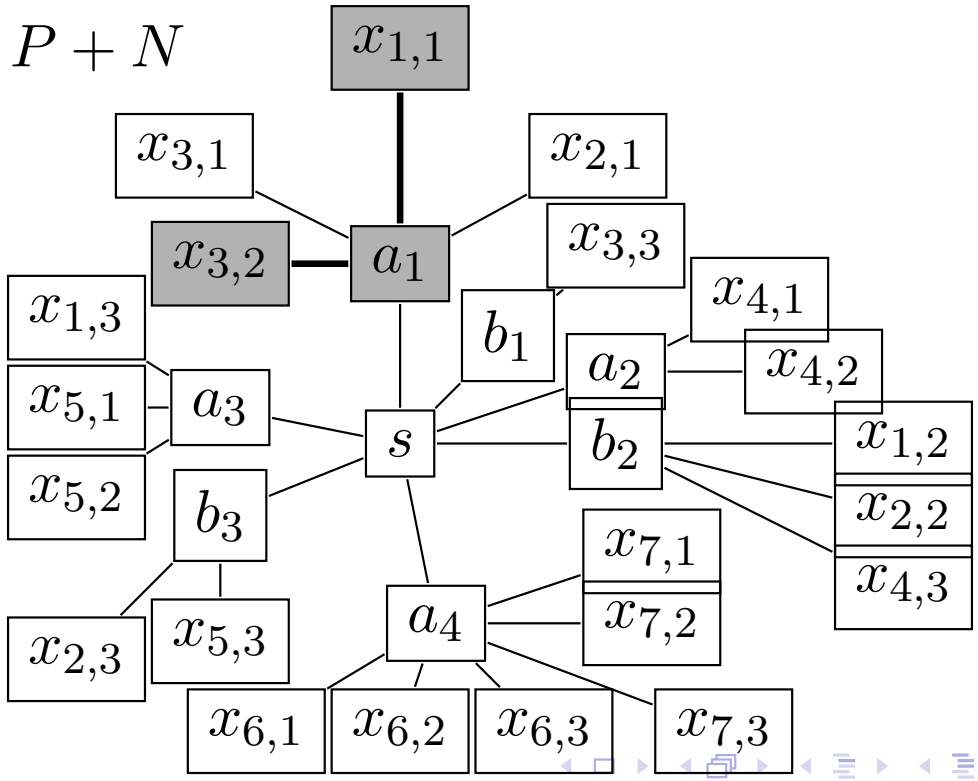


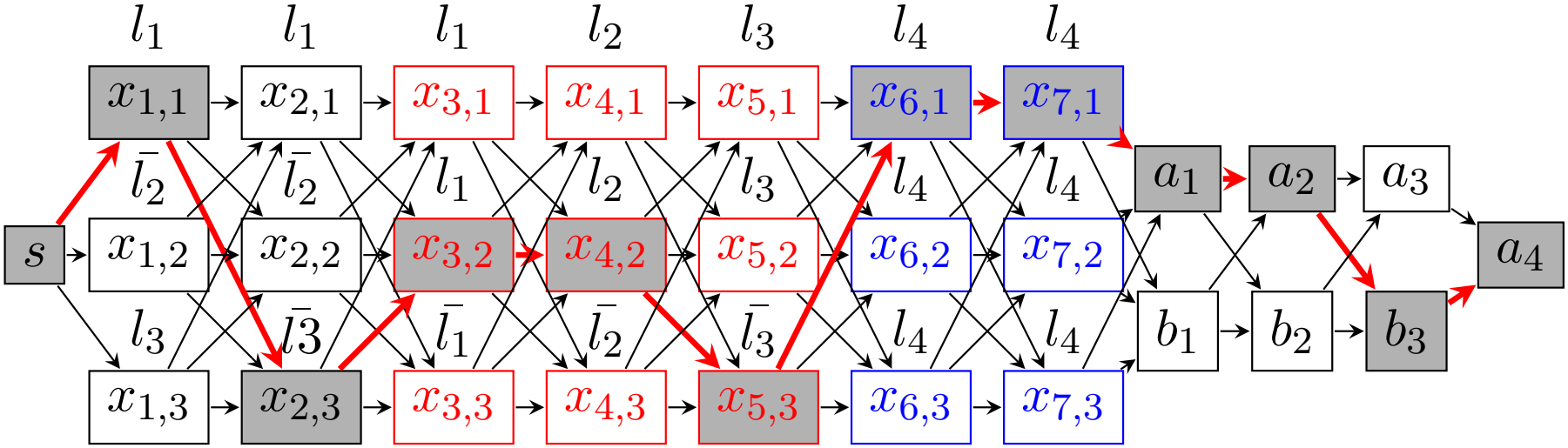
$(\Rightarrow)$ $\quad k = P + N$



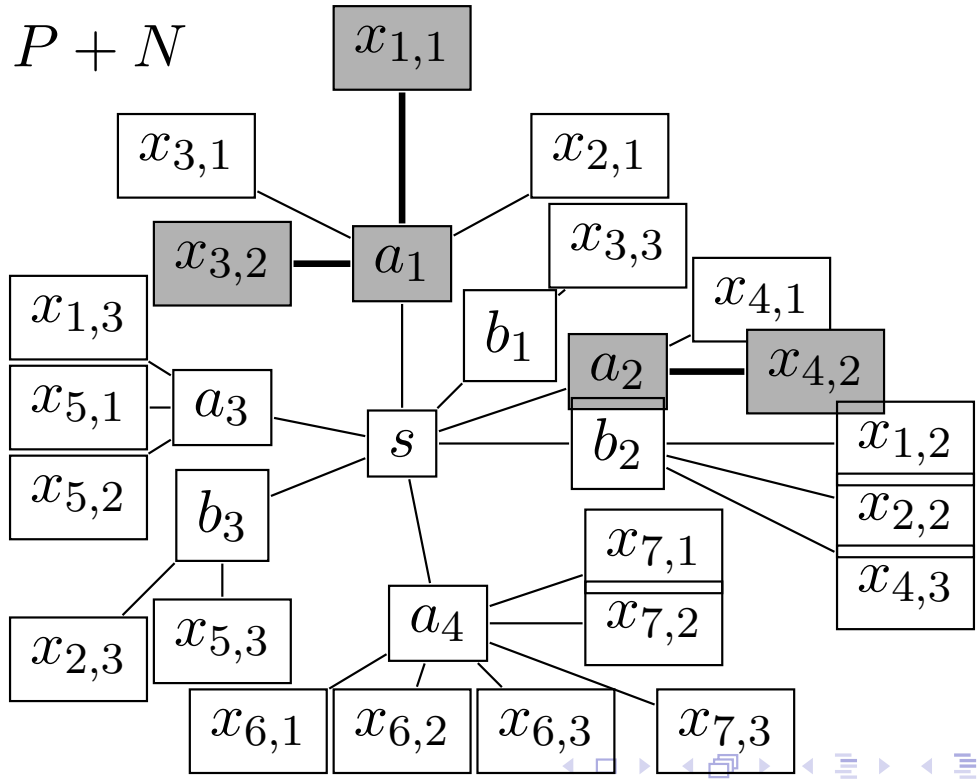


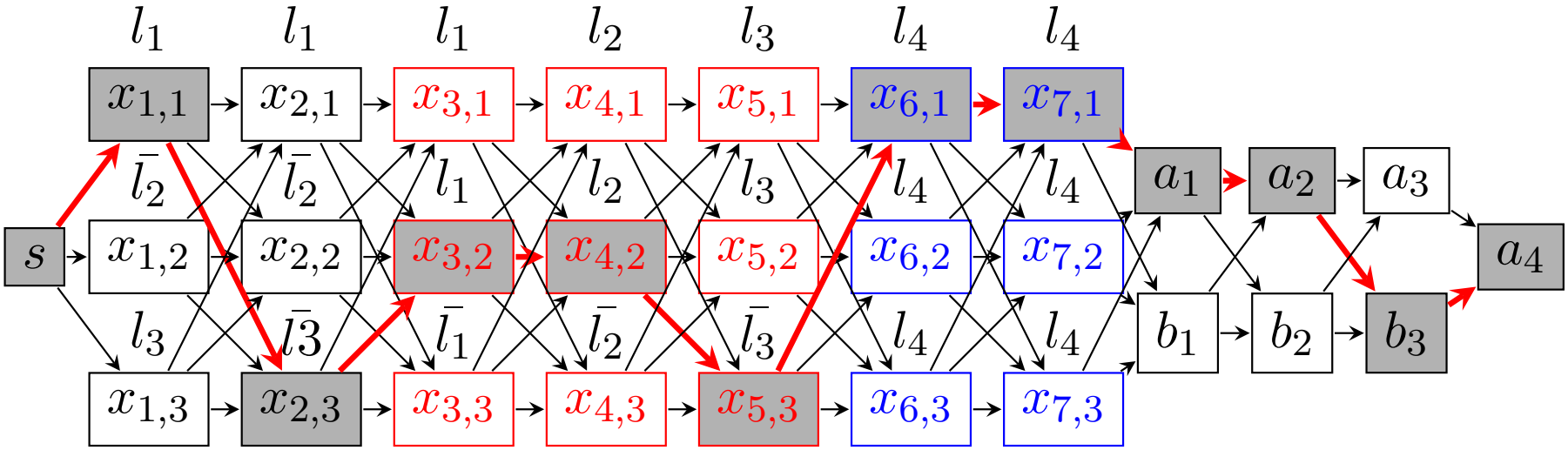
$(\Rightarrow)$ $\quad k = P + N$



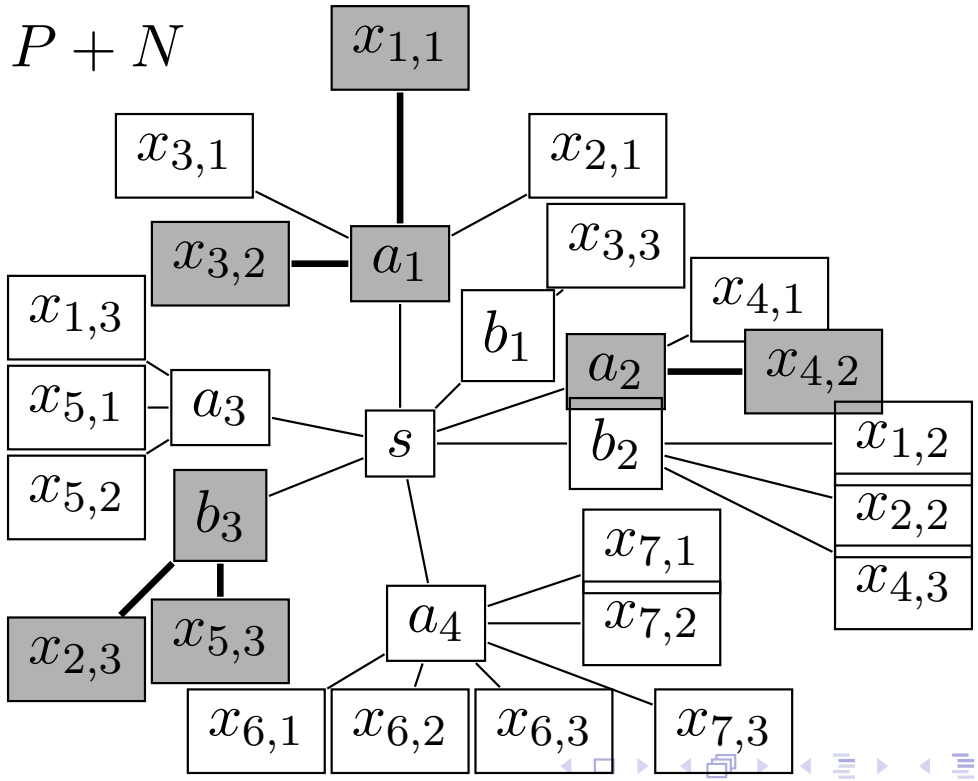


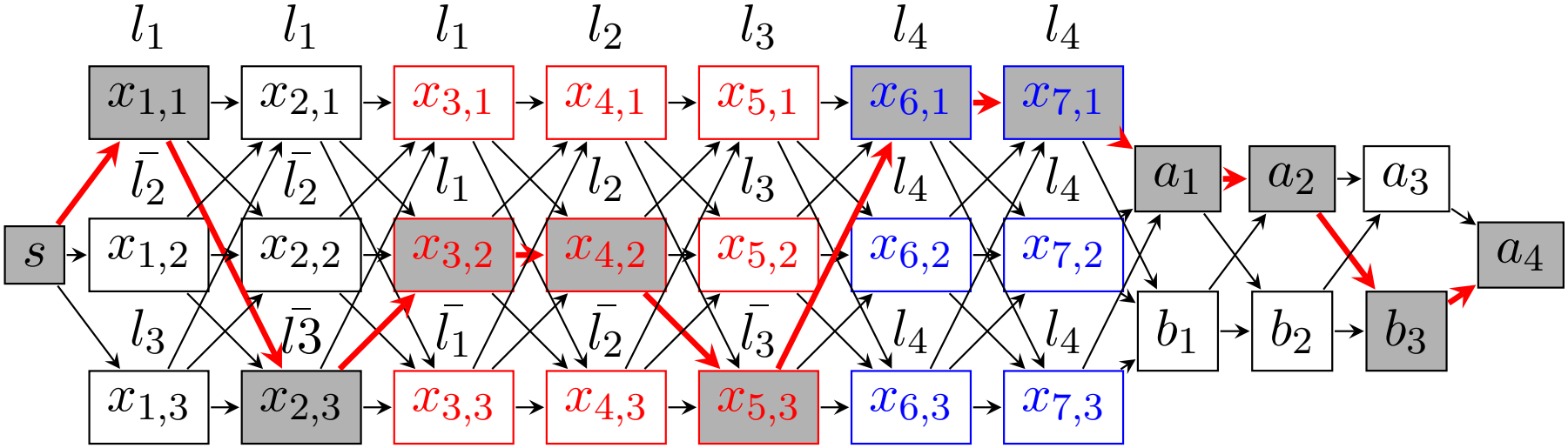
$(\Rightarrow)$ $\quad k = P + N$



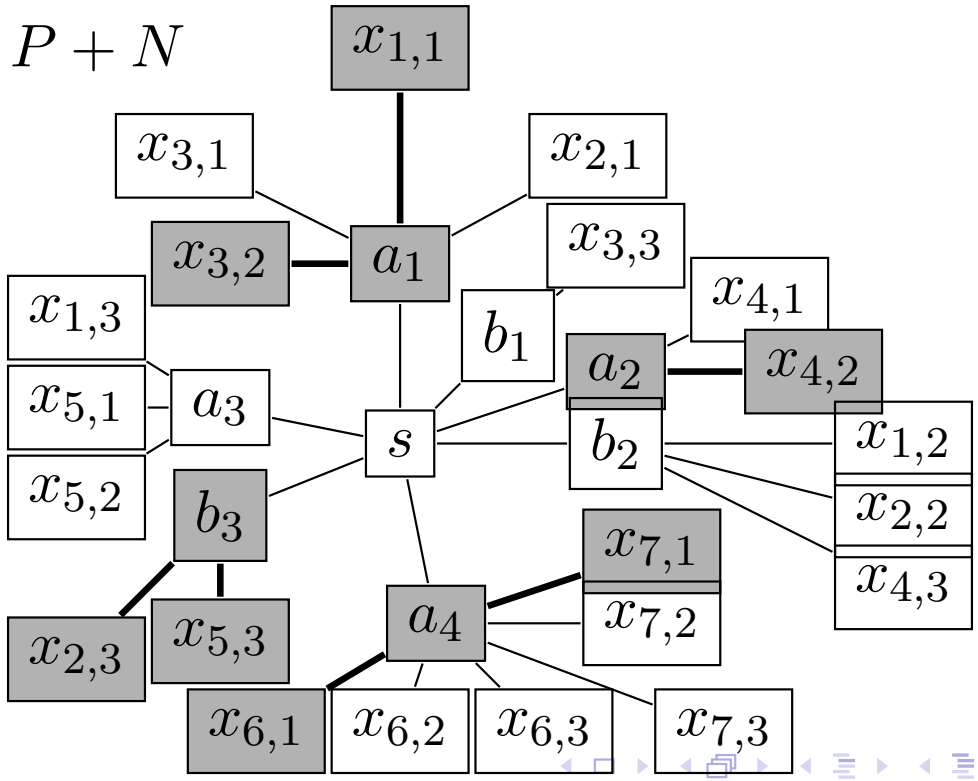


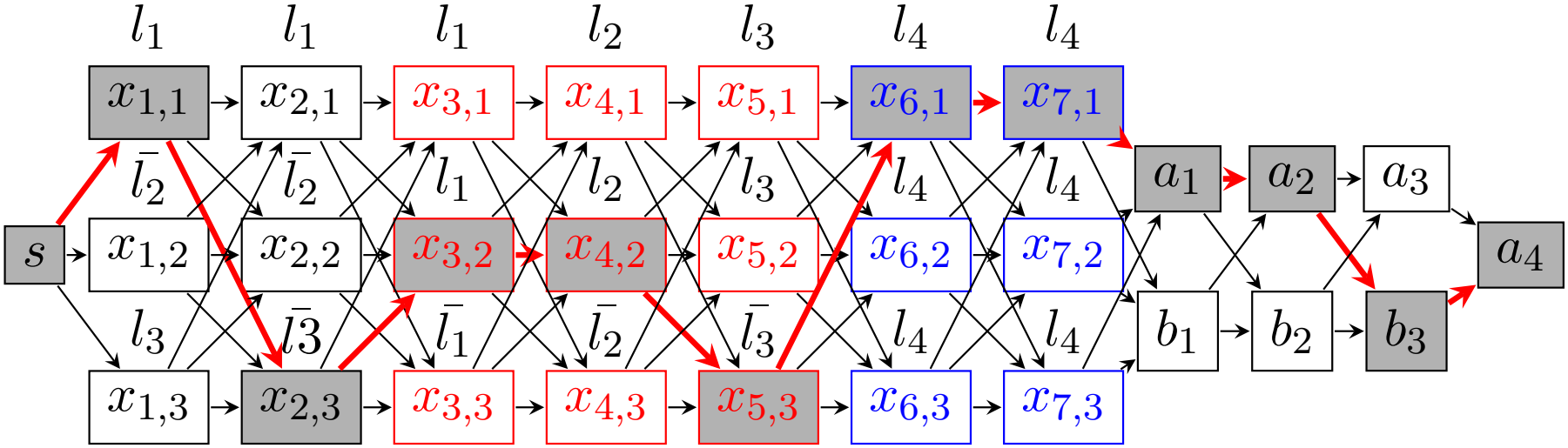
$(\Rightarrow)$ $\quad k = P + N$



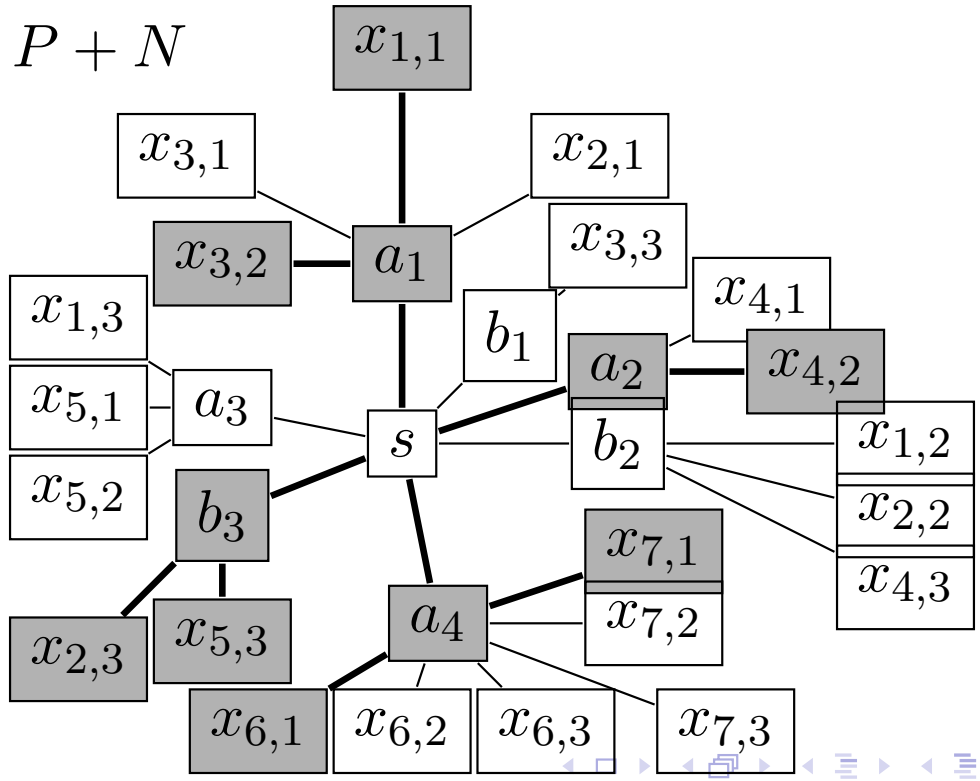


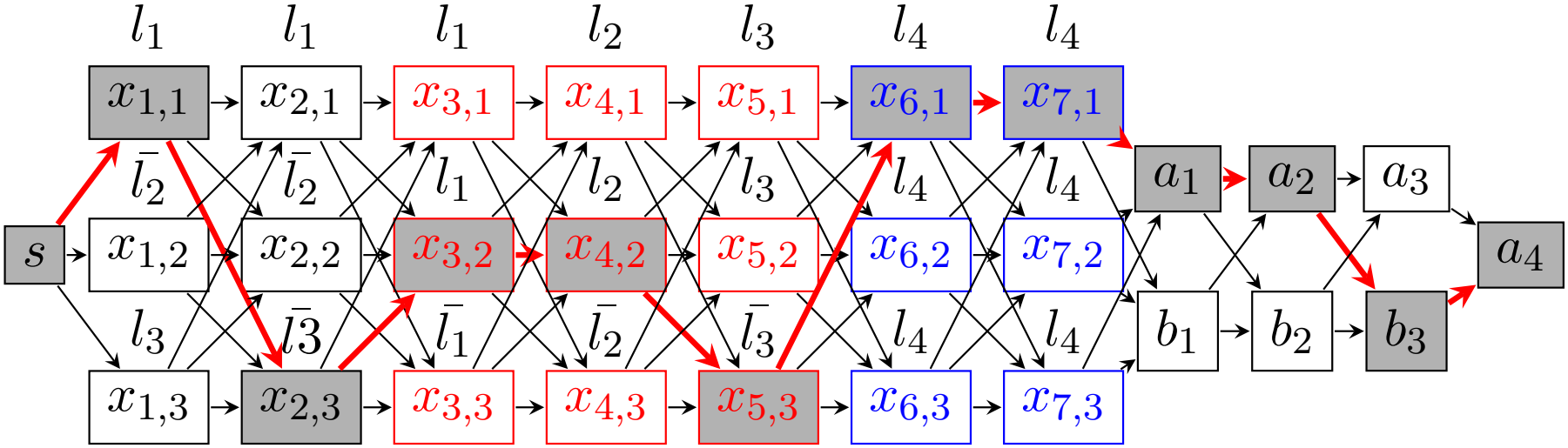
$(\Rightarrow)$ $\quad k = P + N$



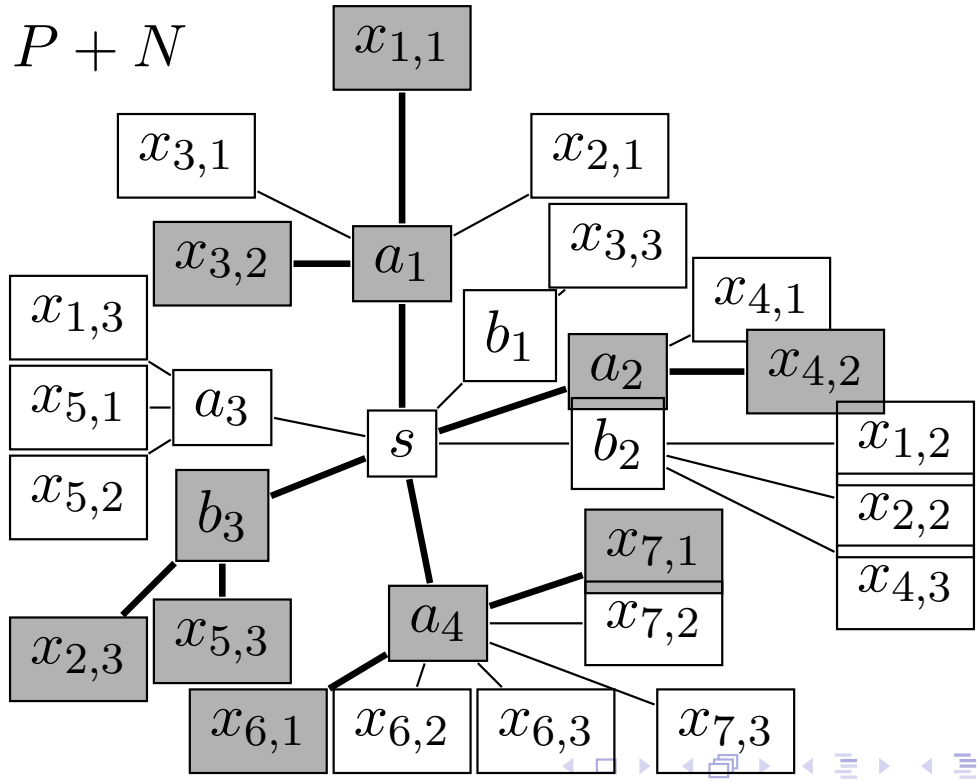


$(\Rightarrow)$ $\quad k = P + N$





$(\Leftarrow)$ $k = P + N$

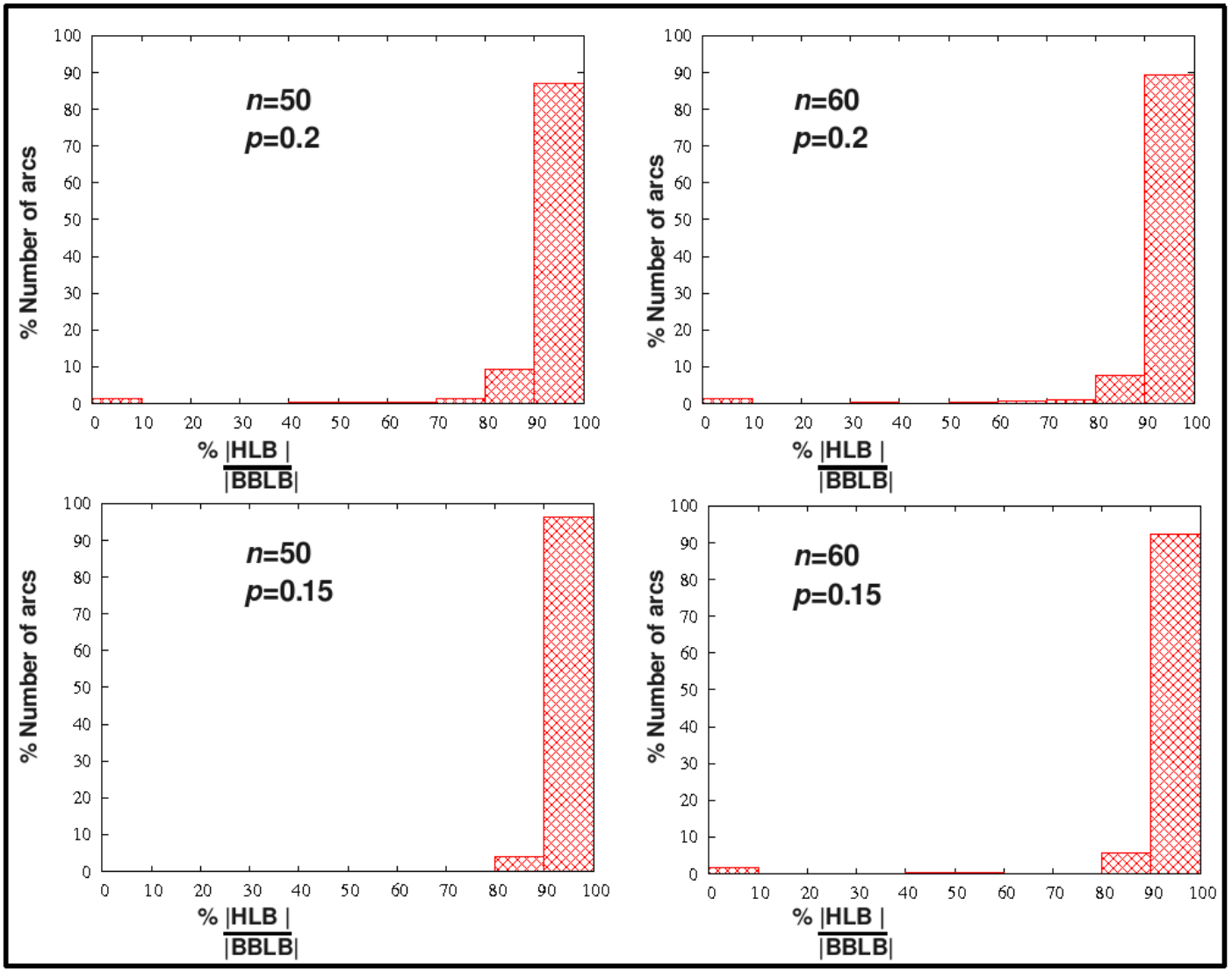


LPDAG-BIJ : résultats théoriques

Si $P \neq NP$ alors le problème $LPDAG - BIJ$ n'est pas f-approximable quelque soit la fonction f.

- Bonne heuristique : HLB.
- Algorithme exact (Branch & Bound) : $BBLB$.

l'heuristique HLB $vs.$ l'algorithme exact $BBLB$



Temps de calcul : 56 minutes $vs.$ 10 heures et 20 minutes.

Conclusion et perspectives

- Nouvelle approche pour comparer des réseaux biologiques
- Bonne heuristique pour le problème LPDAG-BIJ
- Étude la complexité paramétré du problème LPDAG-BIJ
- Trouver des classes des graphes où le problème devient facile
- Généralisation de notre approche pour comparer plus de deux réseaux biologiques