
# Recherche de motifs dans des graphes colorés

Florian Sikora
(encadré par Guillaume Blin et Stéphane Vialette)

Université Paris-Est, LIGM - UMR CNRS 8049

Séminaire Symbiose – 11/02/2010

# Plan

# Plan

# Motivations

15000 30000 45000

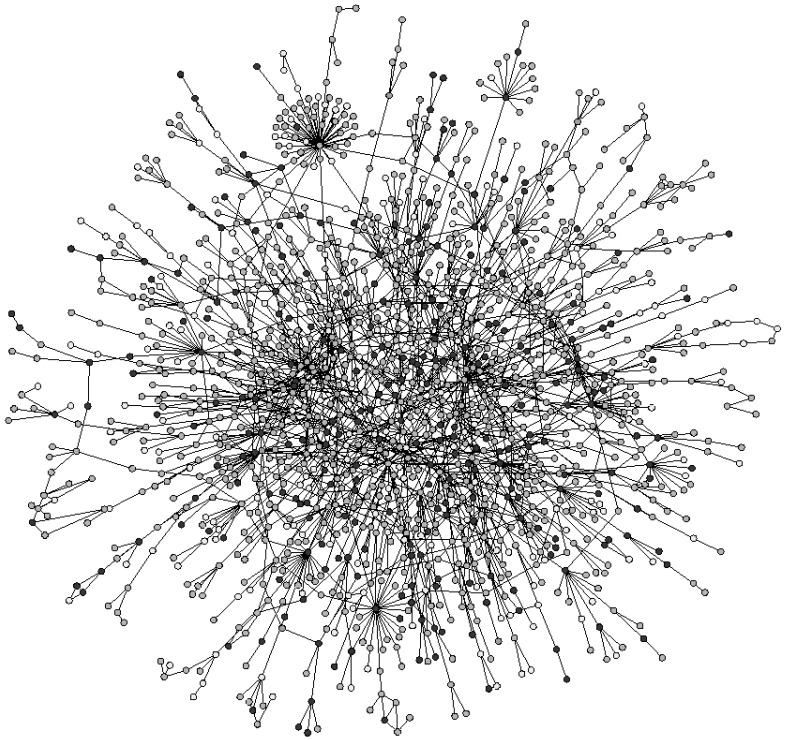
- Complexité de l'homme⇏ # de gènes ?
- Complexité de l'homme⇔ proteines ?

# Les protéines..

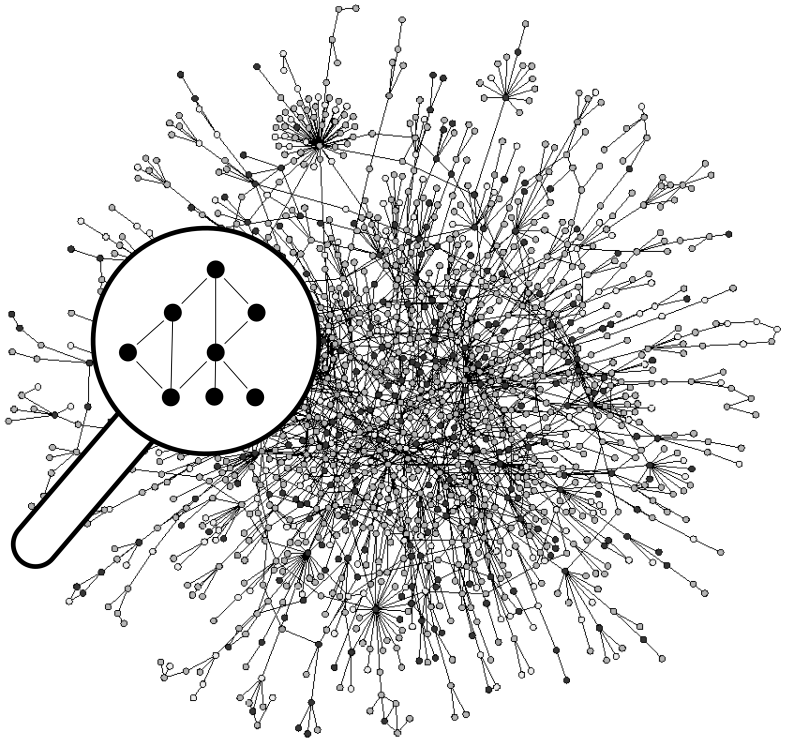
- Nouveaux interêts concernant les proteines...
- ... et sur leurs interactions: Protein-Protein Interactions (PPI)
- Obtenues biologiquement... avec beaucoup de bruit !

# Réseau de protéines

- Les protéines peuvent interagir avec d'autres protéines



# Réseau de protéines



- Modélisation par un graphe (éventuellement pondéré)
  - Les protéines sont représentées par les nœuds
  - Les interactions sont représentées par les arêtes
  - Les arêtes peuvent être pondérées par la probabilité de l'interaction

# Motivations

- Nouvelles techniques : l'information augmente très rapidement [SHARAN & IDEKER 2006]
  - 2001: quelques centaines d'interactions
  - 2006: plusieurs milliers
- Beaucoup de BDD

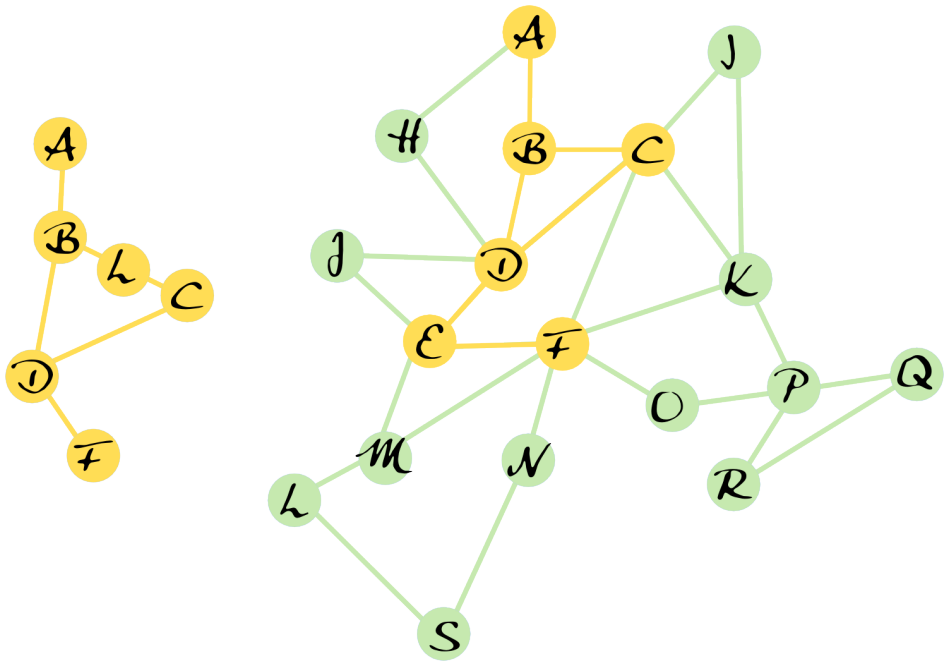
# Motivations

- Nouvelles techniques : l'information augmente très rapidement [SHARAN & IDEKER 2006]
  - 2001: quelques centaines d'interactions
  - 2006: plusieurs milliers
- Beaucoup de BDD
- Chercher des motifs pour retrouver des fonctions connues
- **Déduire** les informations d'espèces **peu connues** depuis des espèces **bien connues**

# Plan

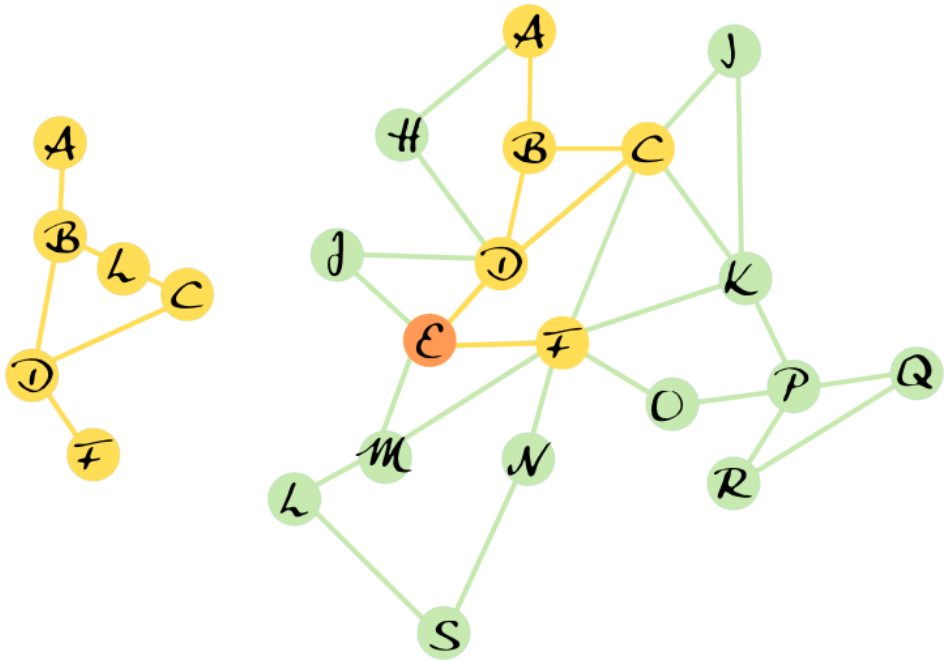
# Rechercher des motifs

- Rechercher des motifs (ici, ensemble de protéines avec une topologie) dans un réseau PPI
- Une protéine est dite **homologue** à une autre protéine selon une analyse de séquences (avec BLASTp)



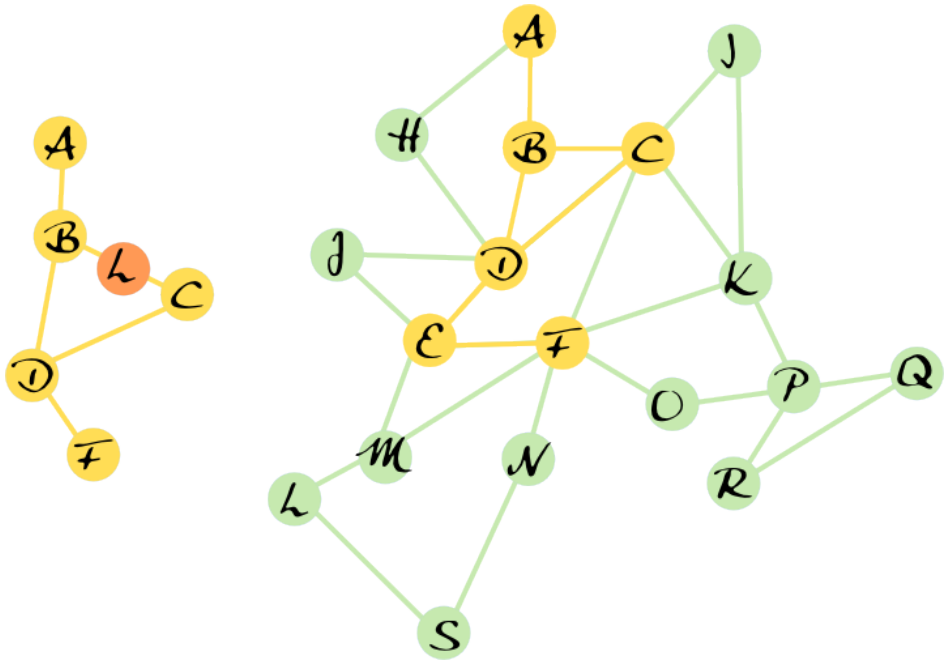
# Rechercher des motifs

- Rechercher des motifs (ici, ensemble de protéines avec une topologie) dans un réseau PPI
- Une protéine est dite **homologue** à une autre protéine selon une analyse de séquences (avec BLASTp)
- Un nombre borné d'**insertions** et deletions peut-être autorisé



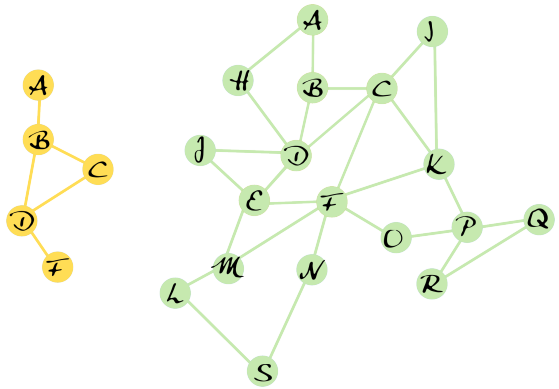
# Rechercher des motifs

- Rechercher des motifs (ici, ensemble de protéines avec une topologie) dans un réseau PPI
- Une protéine est dite **homologue** à une autre protéine selon une analyse de séquences (avec BLASTp)
- Un nombre borné d'insertions et **deletions** peut-être autorisé



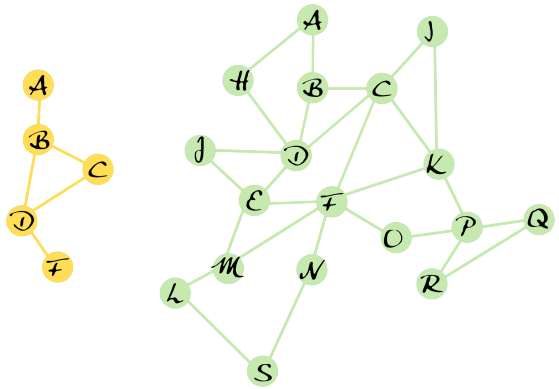
# Recherche avec topologie dans un réseau PPI

- Le motif peut être un **chemin**, un **arbre**, un **graphe**
- Problèmes **NP-difficiles** donc une solution exacte entraine une **complexité exponentielle**



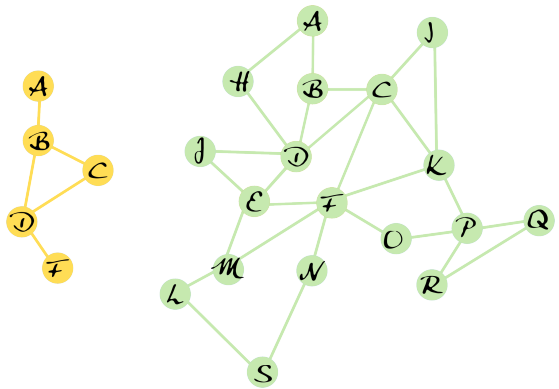
# Recherche avec topologie dans un réseau PPI

- Le motif peut être un **chemin**, un **arbre**, un **graphe**
- Problèmes **NP-difficiles** donc une solution exacte entraine une **complexité exponentielle**
- Mais tout n'est pas perdu !



# Recherche avec topologie dans un réseau PPI

- Idée: exploiter le fait que les **motifs sont plus petits** ($\sim 5 - 15$) que le réseau (e.g. $\sim 5.000$ pour la levure)
- Restreindre la partie exponentielle à $k$ (taille motif) au lieu de $n$ (taille du réseau): **complexité paramétrée**



# Algorithmes FPT

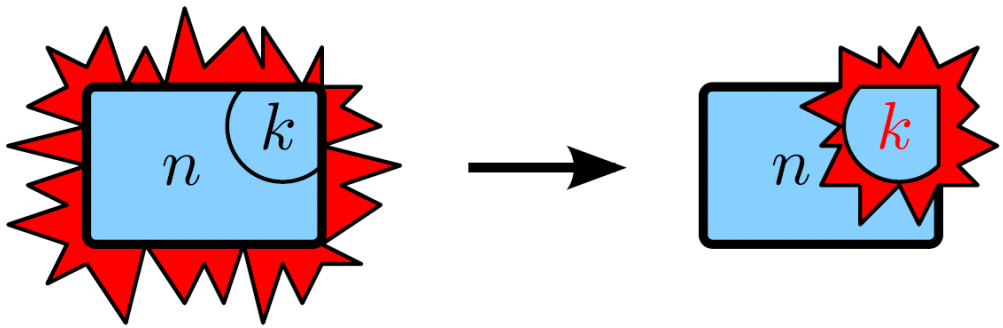
- Beaucoup de problèmes (*i.e.* problèmes paramétrés) sont de la forme :
  - Input : Un objet $X$, $|X| = n$, un entier $k$
  - Question : Est-ce que $X$ a une propriété dépendant de $k$ ?

# Algorithmes FPT

- Beaucoup de problèmes (*i.e.* problèmes paramétrés) sont de la forme :
  - Input : Un objet $X$, $|X| = n$, un entier $k$
  - Question : Est-ce que $X$ a une propriété dépendant de $k$ ?
- Exemples :
  - (VERTEX COVER) Le graphe $G = (V, E)$ contient-il un sous-ensemble $V'$ de sommets de taille $k$ t.q. pour chaque arête $(u, v)$ de $G$, soit $u$ soit $v$ est dans $V'$ ?
  - (LONGEST COMMON SUBSEQUENCE) Existe t-il une string de taille au moins $k$ qui soit une sous-séquence de $n$ strings ?
  - (SET COVER) Etant donné un ensemble $S$ de $n$ ensembles, existe-il un sous-ensemble $S' \subseteq S$ de $k$ ensembles t.q. chaque élément dans les ensembles de $S$ est dans un ensemble de $S'$ ?

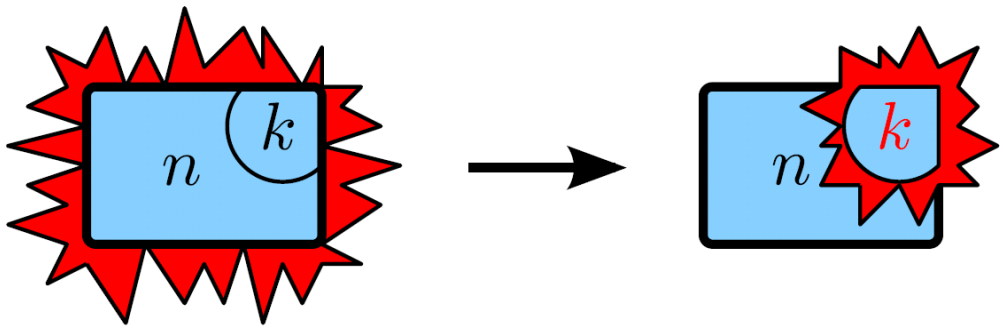
# Algorithmes FPT

- Un algorithme FPT [Downey & Fellows 1999]: algorithme exact **exponentiel** seulement en son **paramètre k** (et pas en la taille de l'entrée $n$)
- $f(k).n^c$, avec $c$ une constante, et $f$ n'importe quelle fonction
- L'algorithme devient souvent "praticable", car même si $f$ est exponentiel, $k$ est petit



# Algorithmes FPT

- Un algorithme FPT [DOWNEY & FELLOWS 1999]: algorithme exact **exponentiel** seulement en son **paramètre k** (et pas en la taille de l'entrée $n$)
- $f(k).n^c$, avec $c$ une constante, et $f$ n'importe quelle fonction
- L'algorithme devient souvent "praticable", car même si $f$ est exponentiel, $k$ est petit
- Attention, $2^{2^{2^{2^{2^{2^k}}}}}.n$ est FPT mais rédibitoire même pour $k = 1$



# Algorithmes FPT

- W-hierarchie :

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \cdots \subseteq W[P]$$

- $W[P]$ est “l’équivalent” de la classe $NP$ pour les problèmes paramétrés
- Probablement des inclusions strictes
- Si un problème est W[t]-difficile, $t \geq 1$, il y a peu de chance qu’un algorithme FPT existe pour ce problème
- Se prouve avec des réductions paramétrées (préservation de la taille de l’instance et du paramètre)

# Color-coding

- Les algorithmes donnant les meilleures complexités (à ma connaissance) utilisent le color-coding
- Technique randomisée initiée pour chercher un chemin de taille $k$ sans cycle dans un graphe

# Color-coding [ALON ET AL. 1995]

- Naïvement, tester tous les chemins possibles : $n^k$
- Color-coding :
  - Déterminer $k$ couleurs
  - Choisir aléatoirement 1 couleur parmi les $k$ pour chaque nœud du graphe
  - Chercher un chemin de taille $k$ contenant ces $k$ couleurs = maintenir une table de taille $2^k$
  - Un “bon chemin” obtient $k$ couleurs différentes avec une probabilité de $\frac{k!}{k^k}$
  - Relancer la procédure $|\ln(\epsilon)|e^k$ fois pour avoir une probabilité d’erreur de $\epsilon > 0$.
  - Exponentiel en $k$ seulement...

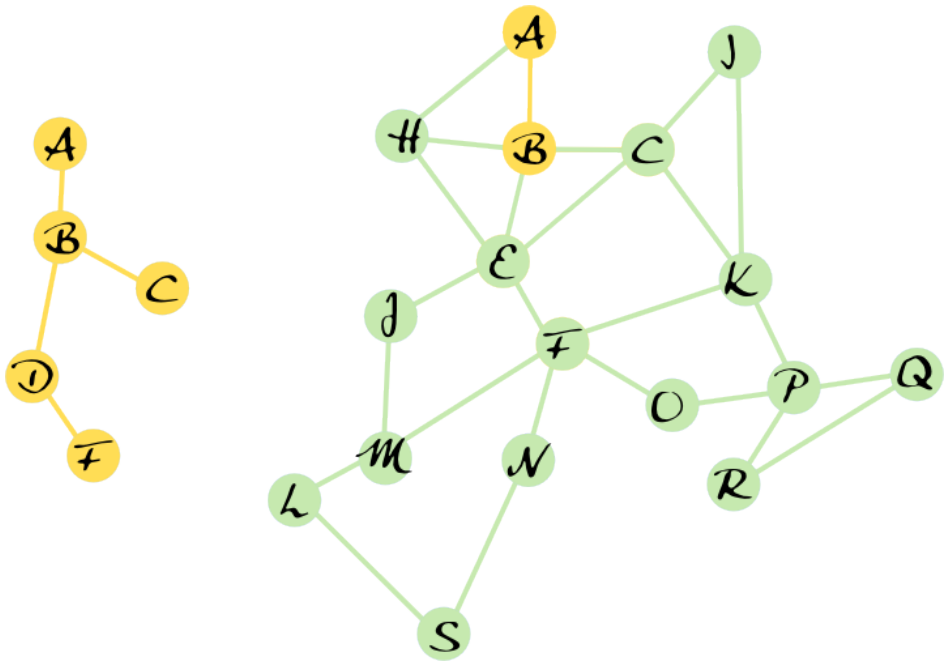
# Motif est un chemin

- QPath, $\mathcal{O}(2^k|E|)$ [Shlomi et al. 2006]
- Application directe du color-coding

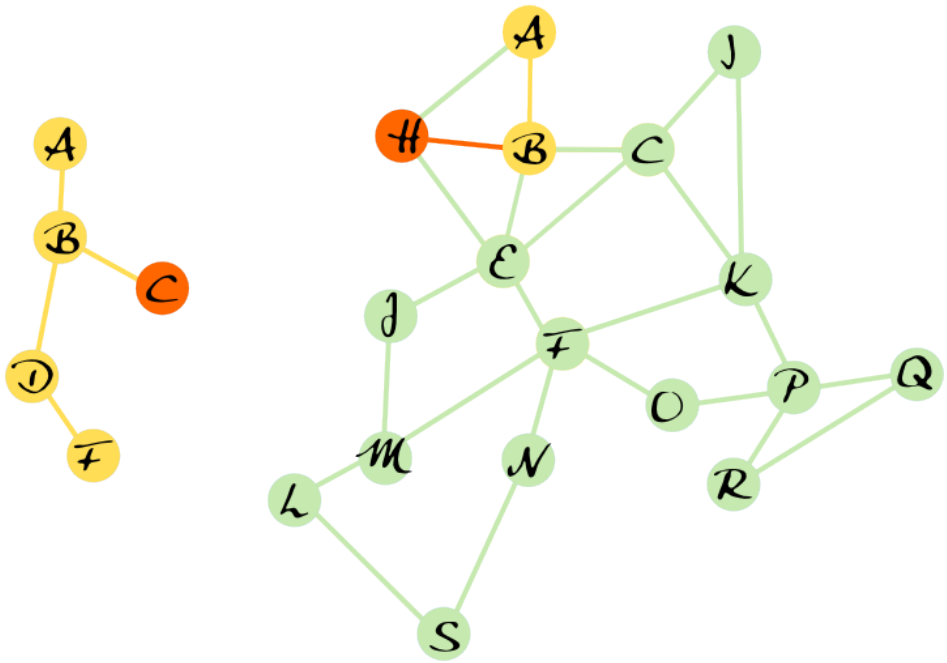
# Motif est un arbre

- Réseau doit être une forêt [PINTER ET AL. 2005]
- QNet, $\mathcal{O}^*(2^{\mathcal{O}(k)})$ [DOST ET AL. 2007]
  - Extension du color-coding (programmation dynamique sur l'arbre)

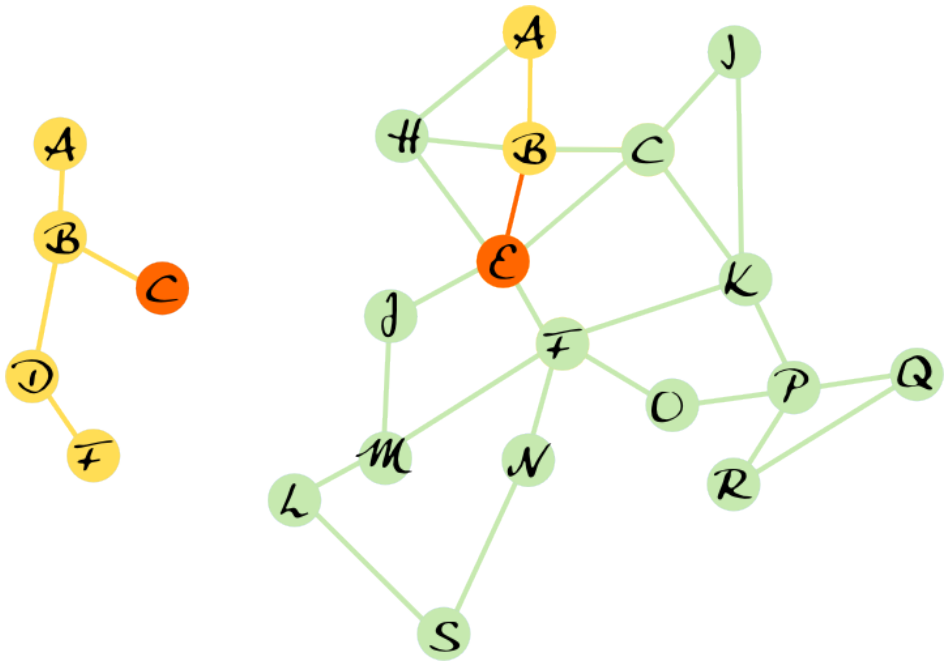
# Motif est un arbre – Idée de la programmation dynamique



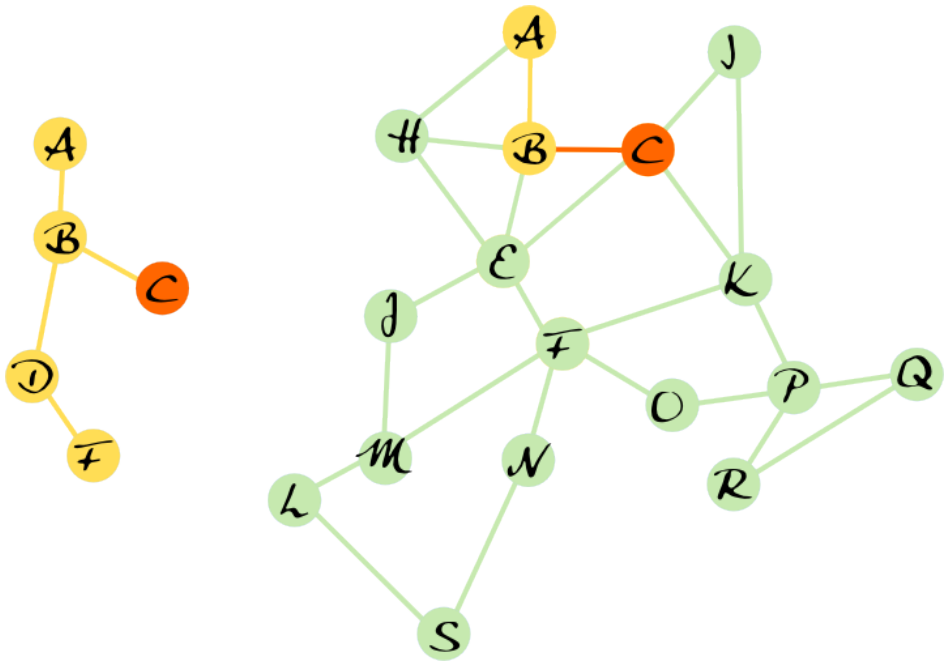
# Motif est un arbre – Idée de la programmation dynamique



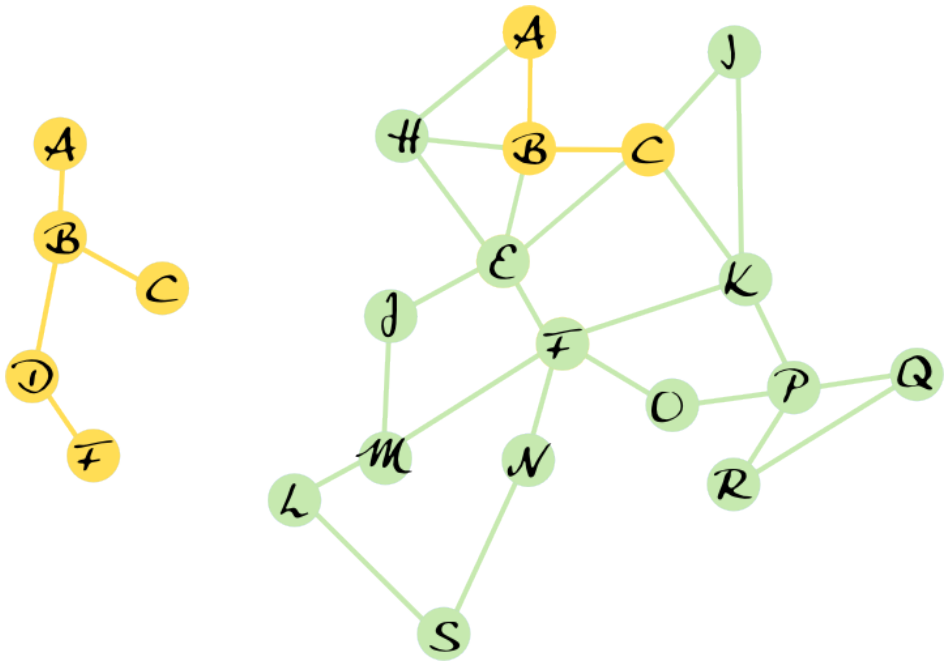
# Motif est un arbre – Idée de la programmation dynamique



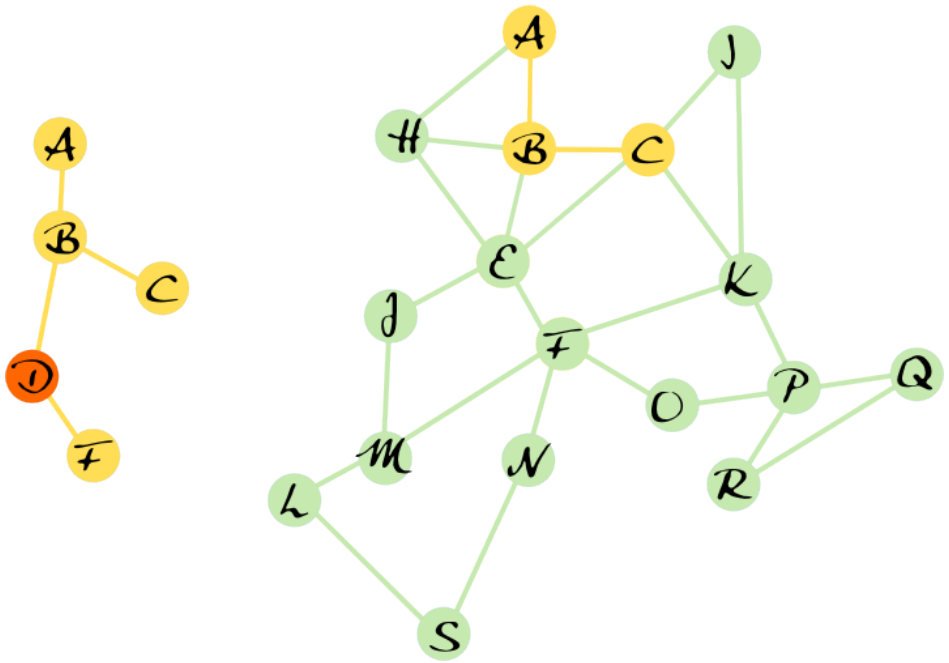
# Motif est un arbre – Idée de la programmation dynamique



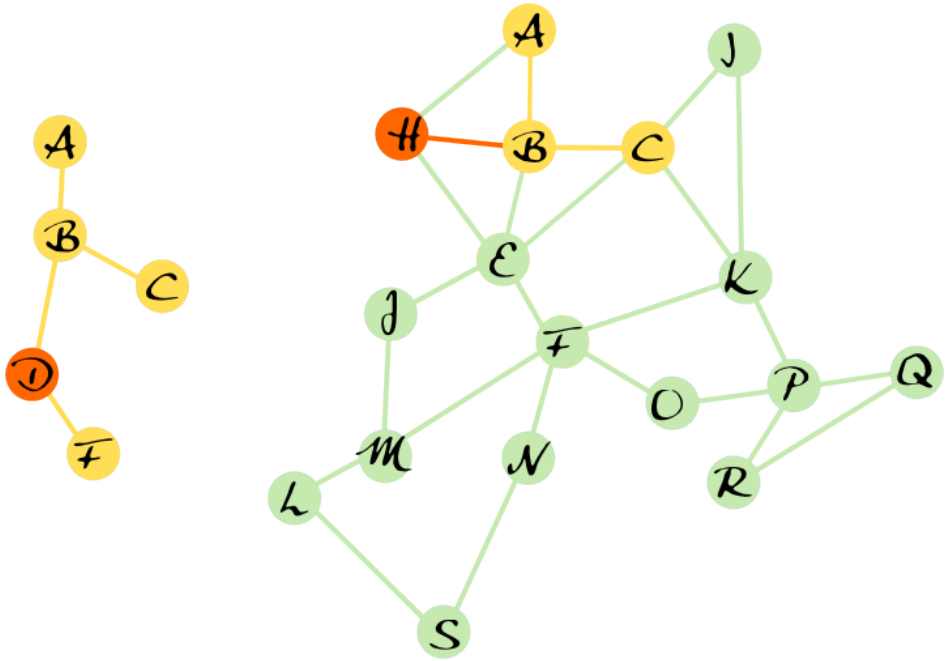
# Motif est un arbre – Idée de la programmation dynamique



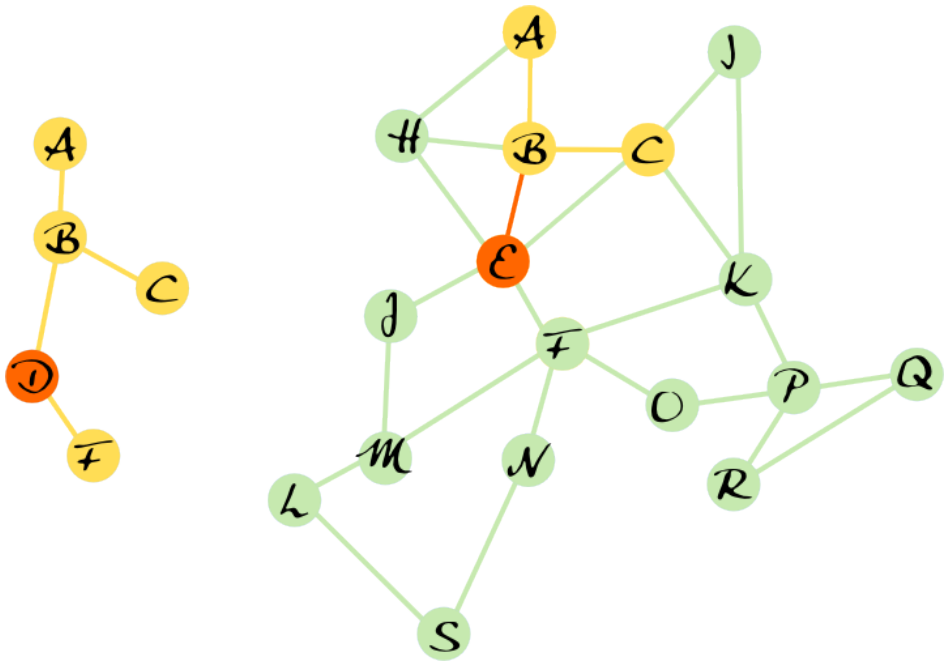
# Motif est un arbre – Idée de la programmation dynamique



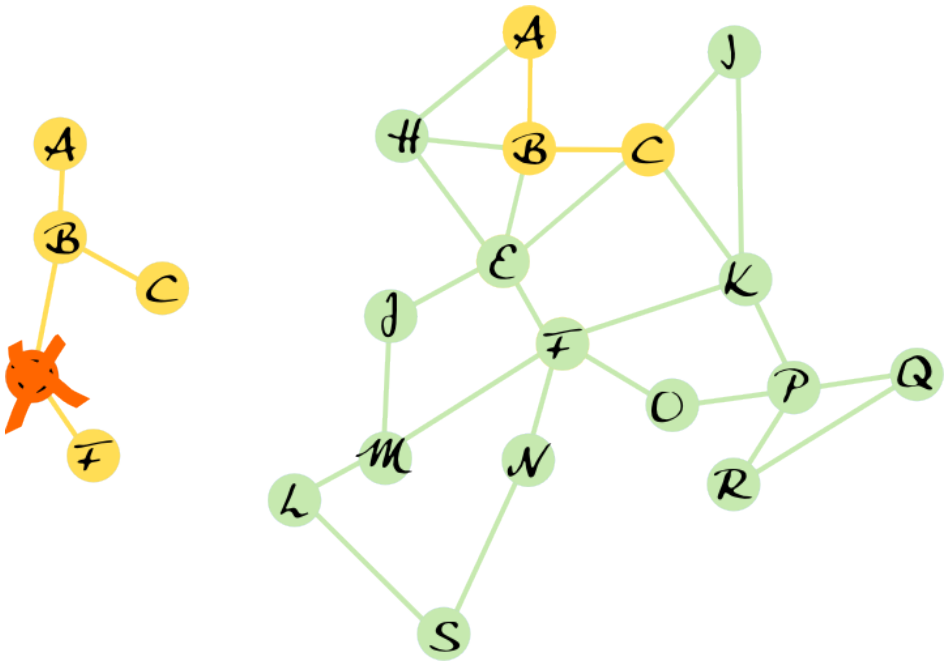
# Motif est un arbre – Idée de la programmation dynamique



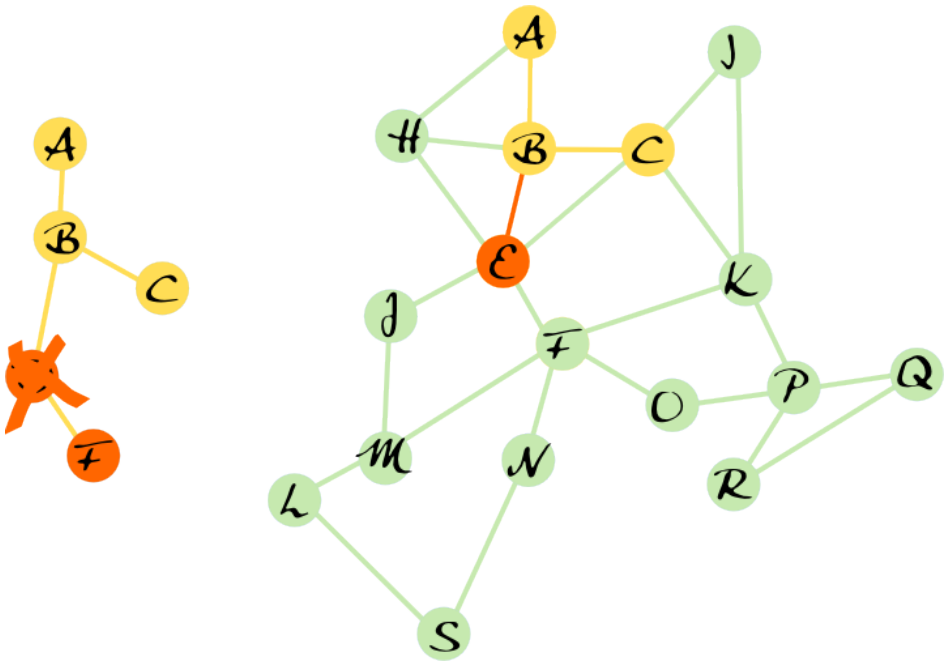
# Motif est un arbre – Idée de la programmation dynamique



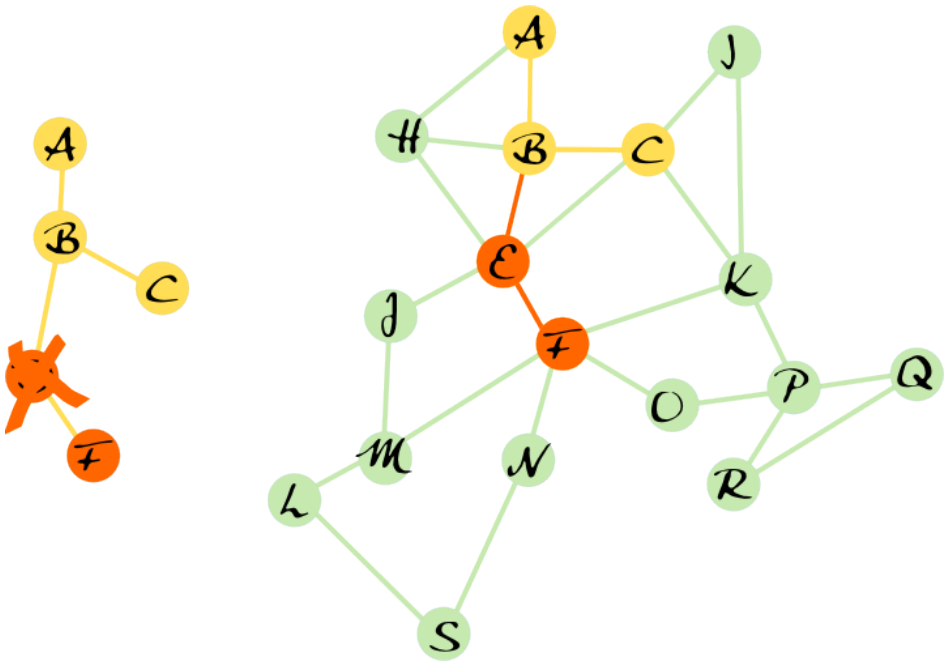
# Motif est un arbre – Idée de la programmation dynamique



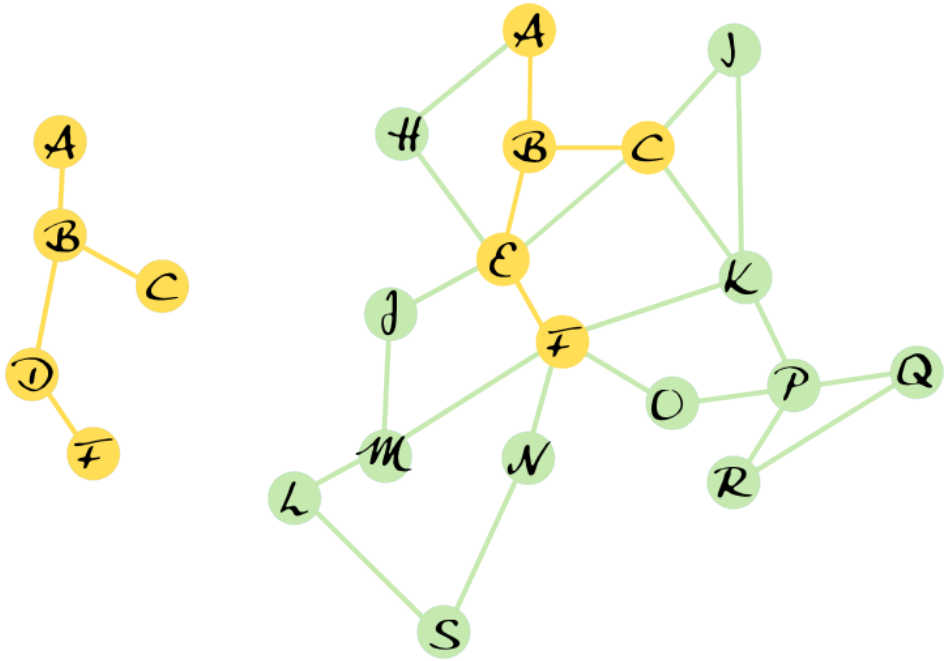
# Motif est un arbre – Idée de la programmation dynamique



# Motif est un arbre – Idée de la programmation dynamique



# Motif est un arbre – Idée de la programmation dynamique




- Initialisation effectuée entre la racine et chaque nœud homologue du réseau

# Motif est un graphe

- **W[1]-Dur** si paramétré par la taille du motif (si motif quelconque) [DOWNEY & FELLOWS 1999] (pas d'algo FPT possible)

# Motif est un graphe

- **W[1]-Dur** si paramétré par la taille du motif (si motif quelconque) [Downey & Fellows 1999] (pas d'algo FPT possible)
- FPT pour des graphes particuliers

## PADA1 [Blin et al. 2009]

- $\mathcal{O}(n^{|\mathbf{F}|}2^{\mathcal{O}(k)}.m)$
- Exponentiel en $|F|$, où $|F|$ est la taille du Vertex Feedback Set du motif (nombre de sommets à supprimer pour rendre le motif acyclique)
- Implémentation python

## QNet [Dost et al. 2007]

- $\mathcal{O}(n^{\mathbf{t+1}}2^{\mathcal{O}(k)})$
- Exponentiel en $t+1$, $t$ est la largeur arborescente du motif ("éloignement" d'un arbre)
- Résultat seulement théorique

# Motif est un graphe

- **W[1]-Dur** si paramétré par la taille du motif (si motif quelconque) [Downey & Fellows 1999] (pas d'algo FPT possible)
- FPT pour des graphes particuliers

## PADA1 [Blin et al. 2009]

- $\mathcal{O}(n^{|\mathbf{F}|}2^{\mathcal{O}(k)}.m)$
- Exponentiel en $|F|$, où $|F|$ est la taille du Vertex Feedback Set du motif (nombre de sommets à supprimer pour rendre le motif acyclique)
- Implémentation python

## QNet [Dost et al. 2007]

- $\mathcal{O}(n^{\mathbf{t+1}}2^{\mathcal{O}(k)})$
- Exponentiel en $t+1$, $t$ est la largeur arborescente du motif ("éloignement" d'un arbre)
- Résultat seulement théorique

- $TW \leq VFS + 1$ [Bodlaender et Koster 2007]

# Motif est un graphe – PADA1

- Adapter l'algorithme FPT existant pour motif arbre
- 2 étapes
  1. **Transformer** le motif en arbre sans perte (avec Vertex Feedback Set et duplication des nœuds)
  2. **Rechercher** une occurence de cet arbre dans le réseau par programmation dynamique

# Plan

# GRAPH MOTIF

- Constat : les données biologiques sont très bruitées
  - Manque des informations, faux négatifs. Estimé à 50% [GAVIN ET AL. 2002]
  - Informations erronées, faux positifs. Estimé à 65% [REGULY ET AL. 2006]
- La topologie du motif peut ne pas être connue *a priori*
- La topologie du motif peut ne pas être pertinente

- Chaque nœud du réseau est coloré par sa “fonction”
- Le motif sera juste un ensemble (ou multi ensemble) de couleurs à rechercher dans un réseau coloré [LACROIX ET AL. 2006]

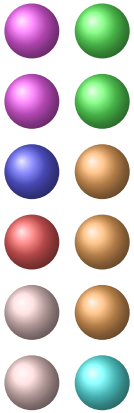
# GRAPH MOTIF

- S'applique à différent type de réseaux biologiques
  - Dans les réseaux PPI, chaque protéine du motif recoit une couleur
  - Le réseau est coloré selon les homologies avec les protéines du motif
- Selon [BETZLER ET AL. 2008], peut être utilisé pour réseaux sociaux ou autres réseaux complexes...

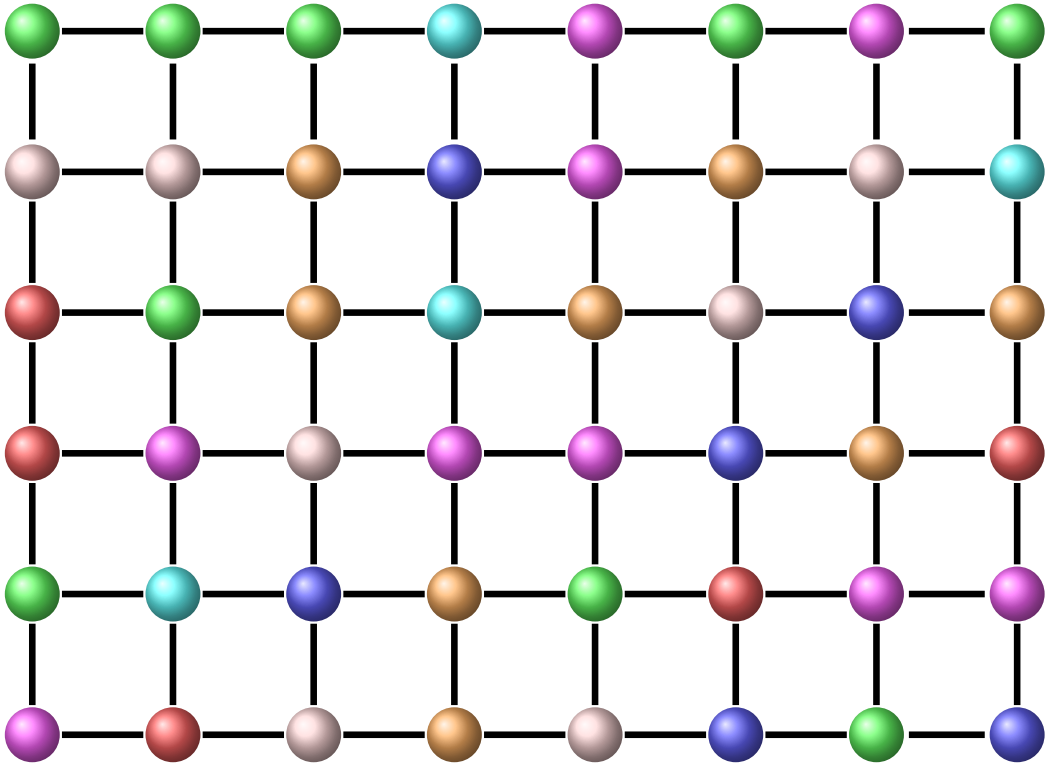
# GRAPH MOTIF [LACROIX ET AL. 2006]

- Etant donné un (multi)-ensemble de couleurs $M$ et un réseau coloré sur les nœuds $G = (V, E)$
- Rechercher un sous ensemble $V' \subseteq V$ t.q.
  - $G[V']$ est connexe
  - Les couleurs de $V'$ correspondent à $M$ (construction d'une bijection entre le motif et les couleurs de la solution)

# GRAPH MOTIF – Un exemple

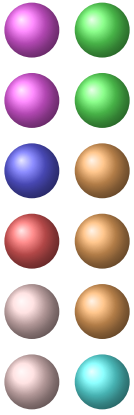


$M$

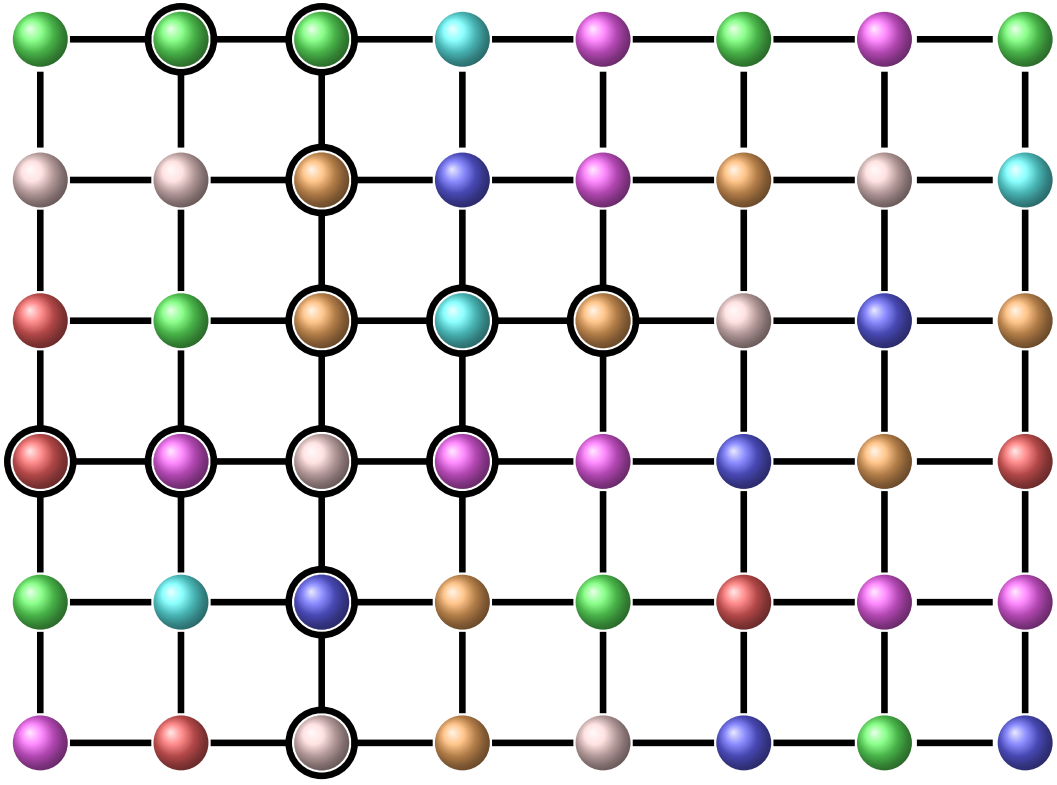


$G = (V, E)$

# GRAPH MOTIF – Un exemple

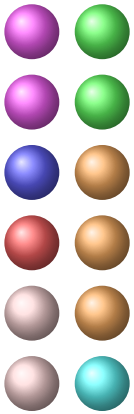


$M$

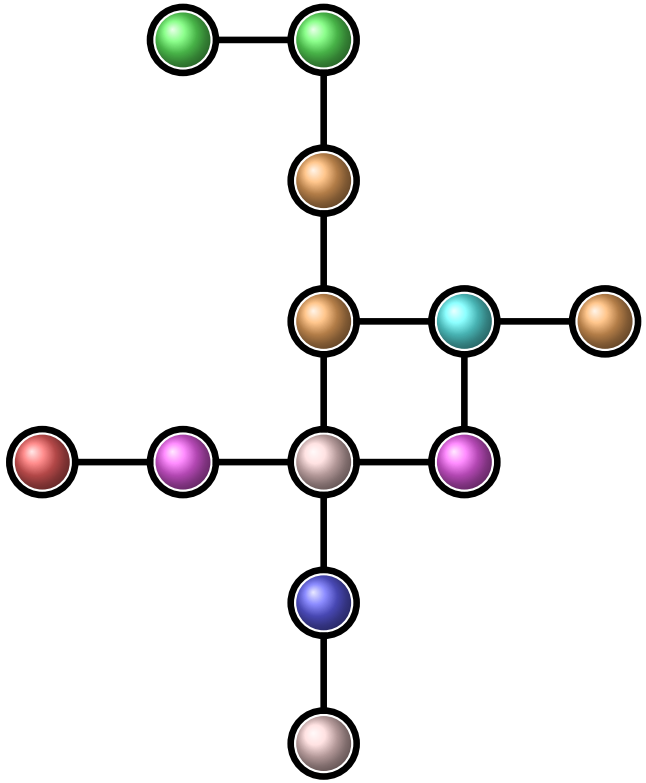


$G = (V, E)$
un $V' \subseteq V$ possible

# GRAPH MOTIF – Un exemple



$M$



$G[V']$

# GRAPH MOTIF – NP-C

- Le problème est NP-Complet, même si
  - Le réseau est un arbre [LACROIX ET AL. 2006]
  - Cet arbre est de degré max 3 et le motif est un ensemble simple [FELLOWS ET AL 2008]
  - Le motif n'est constitué que de 2 couleurs et le réseau est un graphe biparti de degré max 4 [FELLOWS ET AL 2008]

# GRAPH MOTIF – Complexité paramétré

- Le problème est FPT par la taille du motif [LACROIX ET AL. 2006]

# GRAPH MOTIF – Complexité paramétré

- Le problème est FPT par la taille du motif [LACROIX ET AL. 2006]
- Mais la sépération est fine : est W[1]-difficile si paramétré par le nombre de couleurs [FELLOWS ET AL 2008] (pas d'algo FPT possible)

# GRAPH MOTIF – FPT – Motif colorful

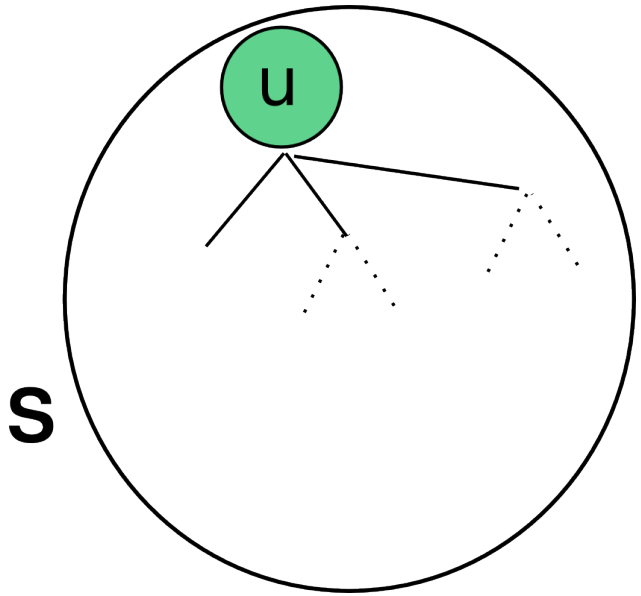
- Complexité en $\mathcal{O}^*(3^k)$ si le motif est un ensemble simple (colorful), avec $k$ taille du motif [BETZLER ET AL. 2008]

# GRAPH MOTIF – FPT – Motif colorful

- Complexité en $\mathcal{O}^*(3^k)$ si le motif est un ensemble simple (colorful), avec $k$ taille du motif [BETZLER ET AL. 2008]
- Idée :
  1. Solution connecté : on cherche un arbre
  2. Motif colorful : on cherche donc un arbre colorful
  3. $k$ couleurs différentes $\Rightarrow$ $k$ nœuds différents

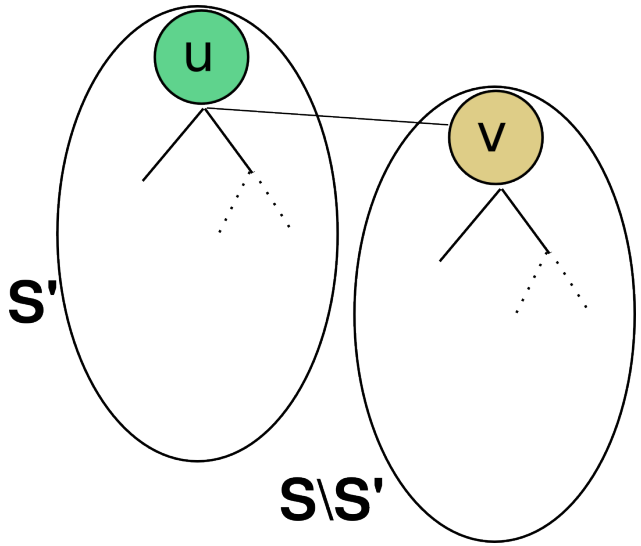
# GRAPH MOTIF – Motif colorful – Programmation dynamique

- But : trouver un arbre, enraciné en $u$, colorful sur $S$.
  $D(u, S) = True$ ?



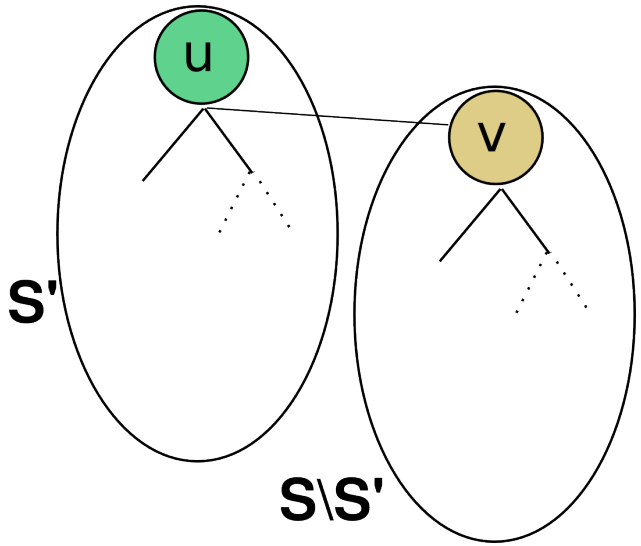
# GRAPH MOTIF – Motif colorful – Programmation dynamique

- But : trouver un arbre, enraciné en $u$, colorful sur $S$. $D(u, S) = True$ ?
- Oui si $u$ à un voisin $v$ t.q.
  1. $u$ enracine un arbre, colorful sur $S'$ couleurs
  2. $v$ enracine un arbre, colorful sur $S \setminus S'$ couleurs



# GRAPH MOTIF – Motif colorful – Programmation dynamique

- But : trouver un arbre, enraciné en $u$, colorful sur $S$. $D(u, S) = True$ ?
- Oui si $u$ à un voisin $v$ t.q.
  1. $u$ enracine un arbre, colorful sur $S'$ couleurs
  2. $v$ enracine un arbre, colorful sur $S \setminus S'$ couleurs
- Regarder pour chaque voisin $v$ et chaque $S' \subset S$
- $D(u, S) = D(u, S') \wedge D(v, S \setminus S')$



# GRAPH MOTIF – Motif colorful – Programmation dynamique

- But : trouver un arbre, enraciné en $u$, colorful sur $S$. $D(u, S) = True$ ?
- Oui si $u$ à un voisin $v$ t.q.
  1. $u$ enracine un arbre, colorful sur $S'$ couleurs
  2. $v$ enracine un arbre, colorful sur $S \setminus S'$ couleurs
- Regarder pour chaque voisin $v$ et chaque $S' \subset S$
- $D(u, S) = D(u, S') \wedge D(v, S \setminus S')$
- Recursivement...
- ...jusqu'à $S = col(u)$

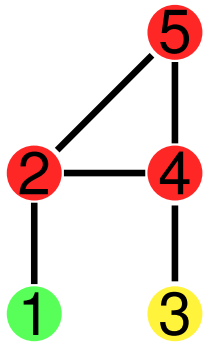
# GRAPH MOTIF – FPT – Multiset

- Programmation dynamique valable que si le motif est un simple ensemble, car $S$ et $S'$ doivent être distincts (sinon plus assuré d'avoir $k$ nœuds différents)
- On peut ramener le cas multi-ensemble au cas colorful avec le color-coding, en $\mathcal{O}^*(4.32^k)$ [BETZLER ET AL. 2008]

# GRAPH MOTIF – FPT – Multiset

- Pour chaque couleur $c$ du motif dont $occ_M(c) \geq 2$
- Créer $occ_M(c)$ nouvelles couleurs et remplacer $c$ dans le motif par ces nouvelles couleurs

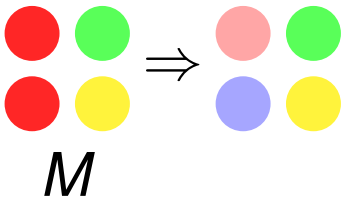
$M$



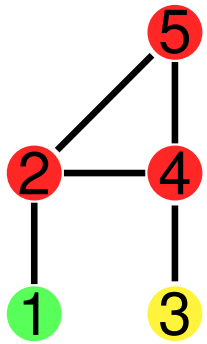

$G$

# GRAPH MOTIF – FPT – Multiset

- Pour chaque couleur $c$ du motif dont $occ_M(c) \geq 2$
- Créer $occ_M(c)$ nouvelles couleurs et remplacer $c$ dans le motif par ces nouvelles couleurs



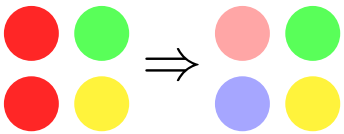
$M$



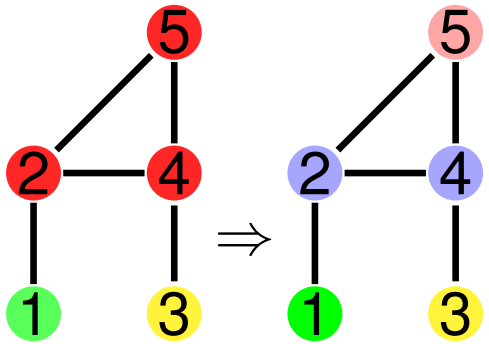

$G$

# GRAPH MOTIF – FPT – Multiset

- Pour chaque couleur $c$ du motif dont $occ_M(c) \geq 2$
- Créer $occ_M(c)$ nouvelles couleurs et remplacer $c$ dans le motif par ces nouvelles couleurs
- Recolorer aléatoirement les nœuds du réseau portant la couleur $c$ par une des nouvelles couleurs



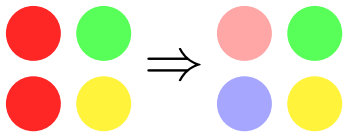
$M$



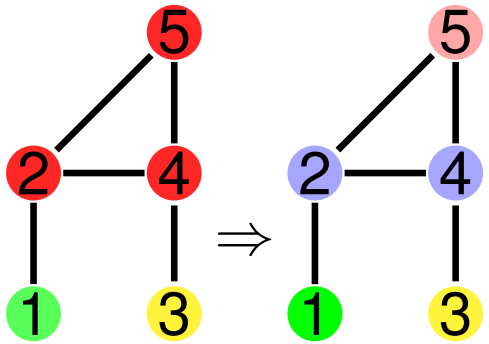

$G$

# GRAPH MOTIF – FPT – Multiset

- Pour chaque couleur $c$ du motif dont $occ_M(c) \geq 2$
- Créer $occ_M(c)$ nouvelles couleurs et remplacer $c$ dans le motif par ces nouvelles couleurs
- Recolorer aléatoirement les nœuds du réseau portant la couleur $c$ par une des nouvelles couleurs



$M$




$G$

- Finalement, le motif est colorful : en chercher une occurence dans le réseau modifié avec l'algo précédent

# GRAPH MOTIF – FPT – Multiset

- Pour chaque couleur $c$ du motif dont $occ_M(c) \geq 2$
- Créer $occ_M(c)$ nouvelles couleurs et remplacer $c$ dans le motif par ces nouvelles couleurs
- Recolorer aléatoirement les nœuds du réseau portant la couleur $c$ par une des nouvelles couleurs


- Finalement, le motif est colorful : en chercher une occurence dans le réseau modifié avec l'algo précédent
- Il faut répeter $|\ln(\epsilon)|e^k$ fois cette coloration aléatoire pour avoir une probabilité $1 - \epsilon$ de succès

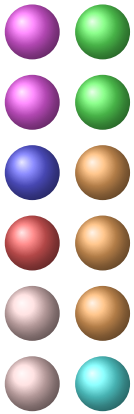
# Une variante de GRAPH MOTIF : MAX MOTIF

- Comme données bruitées, chercher une occurence exacte peut-être impossible
- Autoriser insertions et deletions

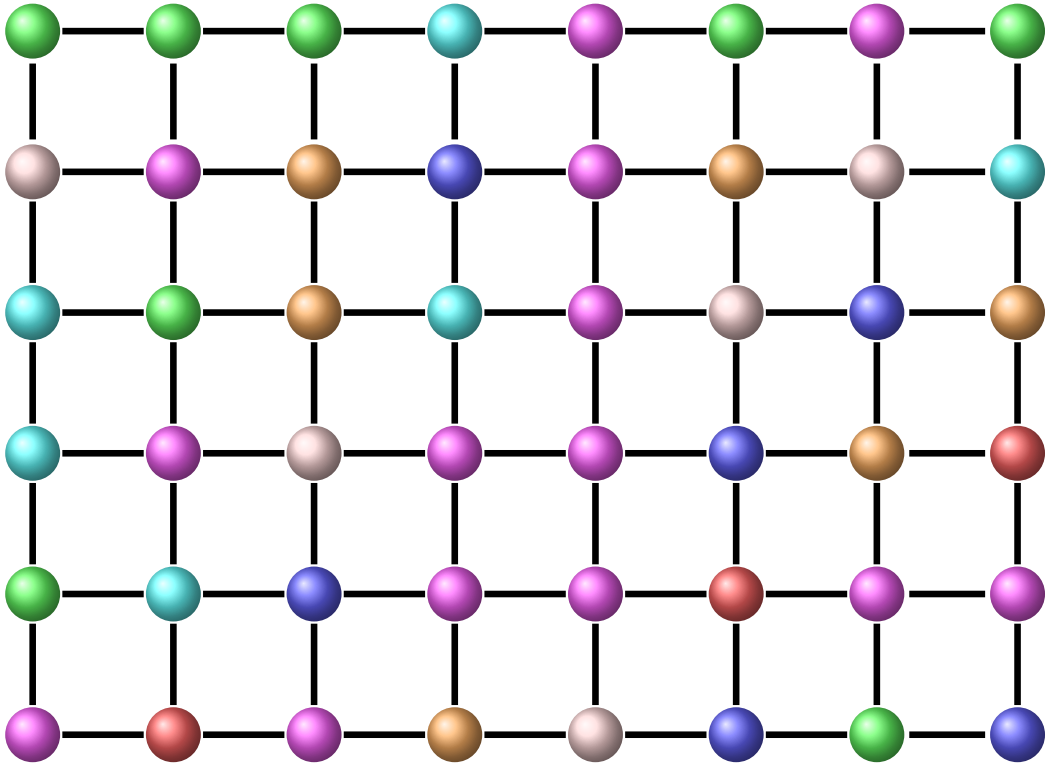
# Une variante de GRAPH MOTIF : MAX MOTIF

- Comme données bruitées, chercher une occurence exacte peut-être impossible
- Autoriser insertions et deletions
- Un exemple : MAX MOTIF
- Trouver une occurence qui matche ”le plus possible” de couleurs du motif

# Exemple MAX MOTIF

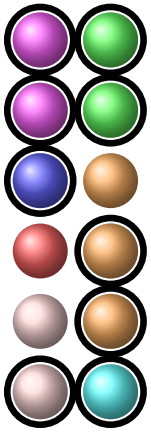


*M*

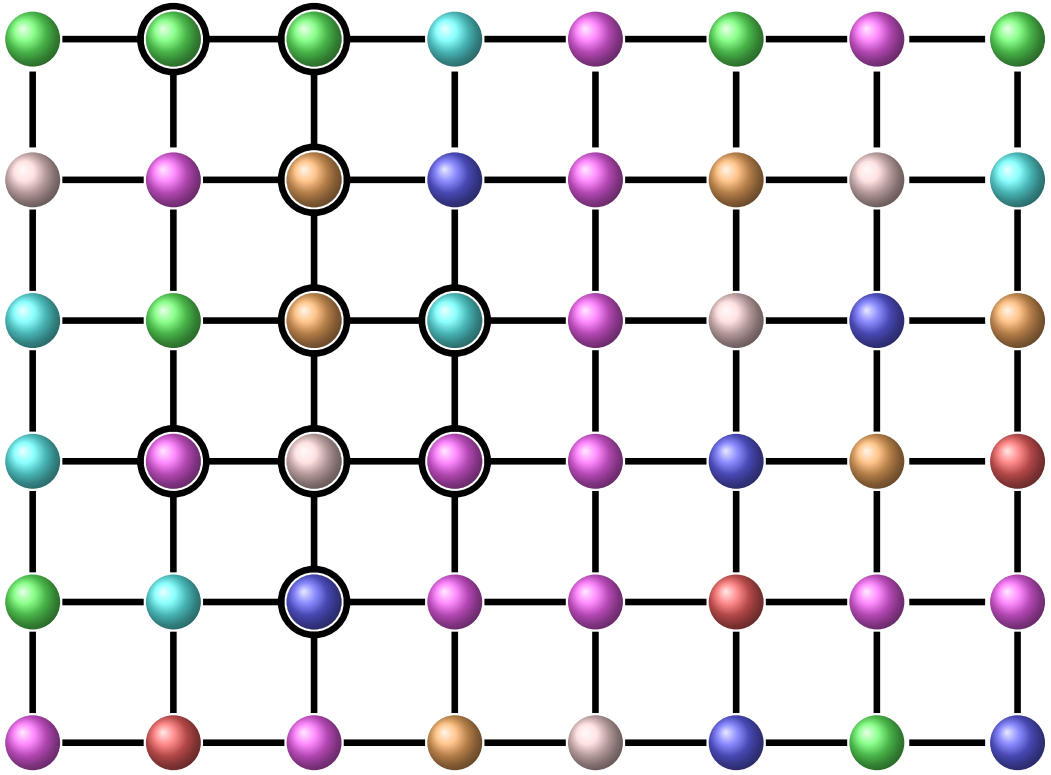


*G*

# Exemple MAX MOTIF

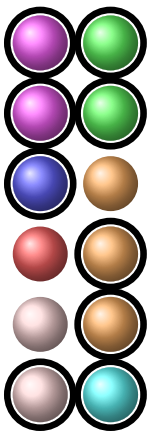


*M*

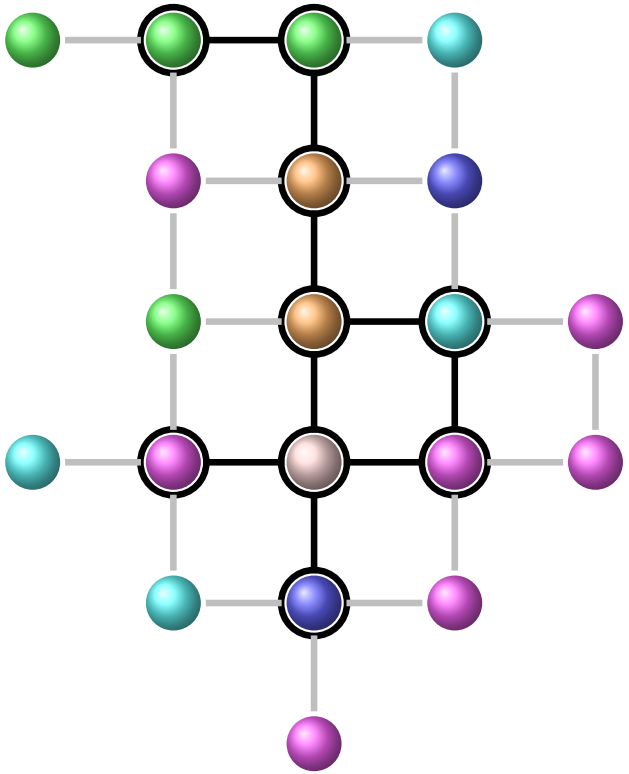


*G*

# Exemple MAX MOTIF

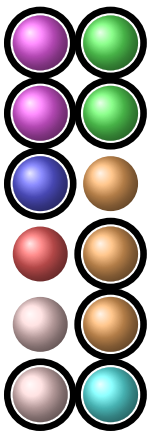


*M*

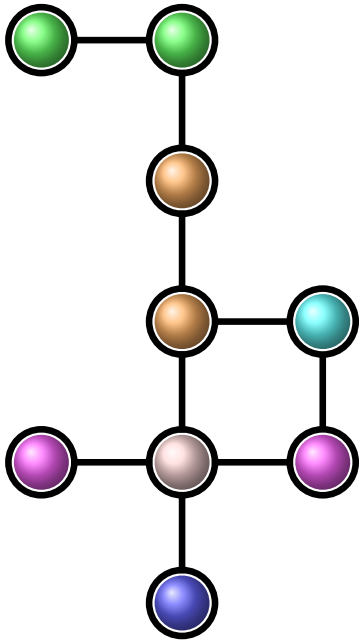


*G*

# Exemple MAX MOTIF



*M*



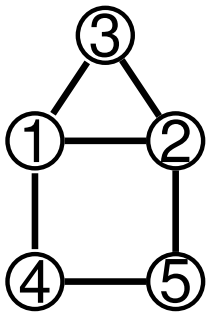
*G*

# MAX MOTIF – Innaproximabilité

- MAX MOTIF est aussi dur à approximer que INDEPENDANT SET , même si
  - Le réseau est un arbre
  - Le motif est un ensemble simple
  - Chaque couleur apparait au plus 2 fois dans le réseau
- INDEPENDANT SET : Dans un graphe $G = (V, E)$, trouver le plus grand $V' \subseteq V$ t.q. aucun nœud dans $V'$ est relié à un autre nœud de $V'$

# MAX MOTIF – Innaproximabilité

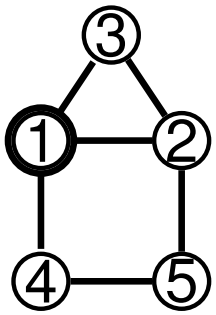
- A partir d'un graphe $G = (V_G, E_G)$, on construit un arbre



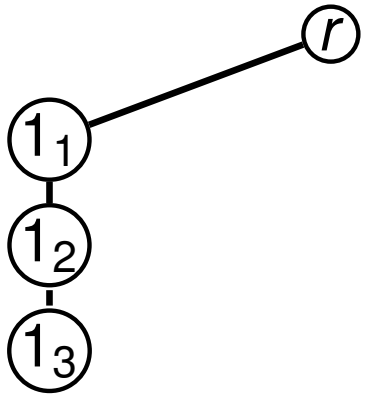

$G = (V_G, E_G)$

# MAX MOTIF – Innaproximabilité

- Chemins de taille $d(u)$, $\forall u \in V_G$



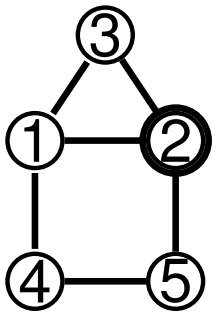

$G = (V_G, E_G)$



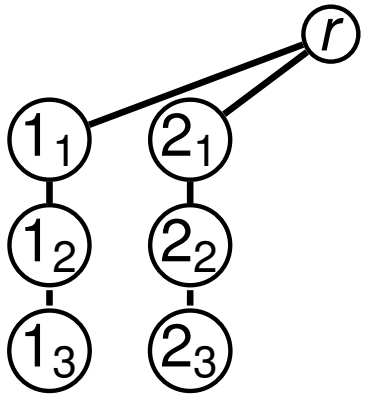

$T$

# MAX MOTIF – Innaproximabilité

- Chemins de taille $d(u)$, $\forall u \in V_G$



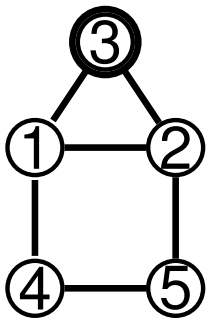

$G = (V_G, E_G)$



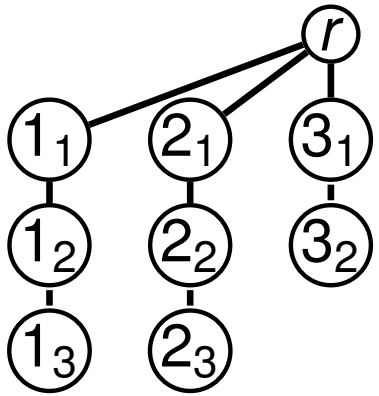

$T$

# MAX MOTIF – Innaproximabilité

- Chemins de taille $d(u)$, $\forall u \in V_G$



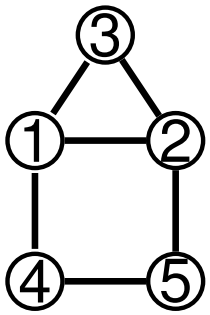

$G = (V_G, E_G)$



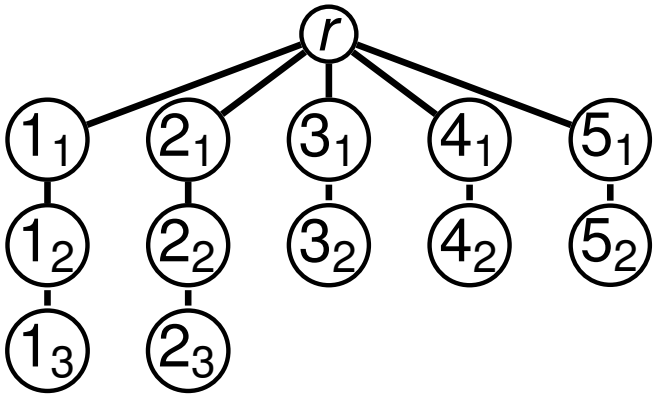

$T$

# MAX MOTIF – Innaproximabilité

- Chemins de taille $d(u)$, $\forall u \in V_G$



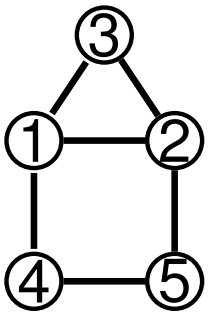

$G = (V_G, E_G)$



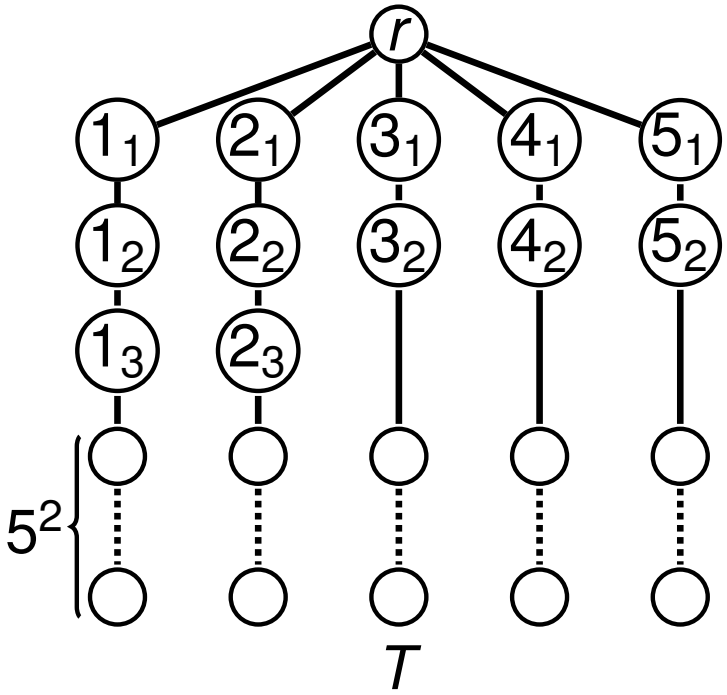

$T$

# MAX MOTIF – Innaproximabilité

- Ajout de $|V_G|^2$ nœuds à chaque chemin

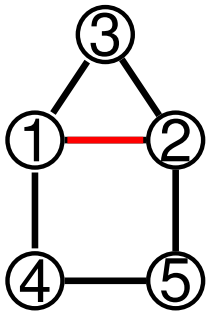



$G = (V_G, E_G)$



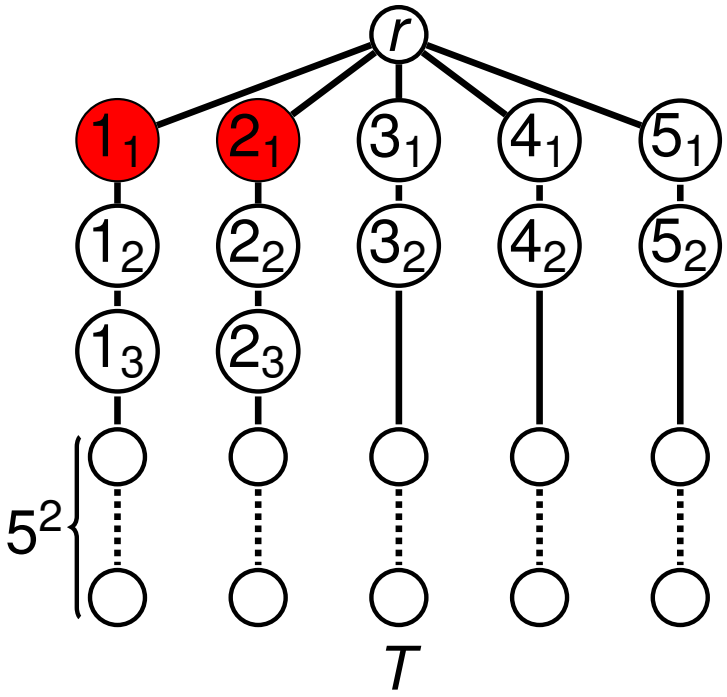
# MAX MOTIF – Innaproximabilité

- Coloration (2 occurences...) + motif (simple)




$G = (V_G, E_G)$

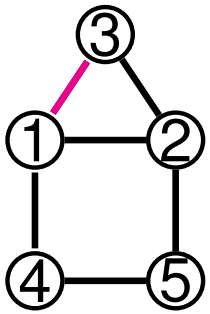
$M$




$T$

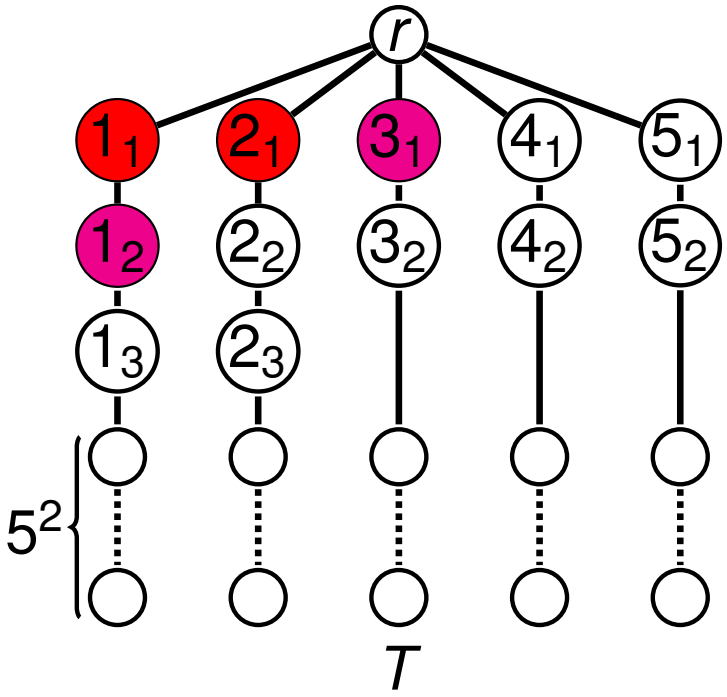
# MAX MOTIF – Innaproximabilité

- Coloration (2 occurences...) + motif (simple)




$G = (V_G, E_G)$

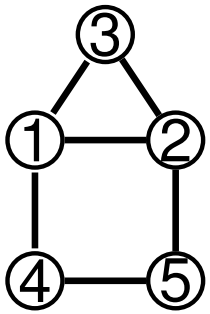
$M$



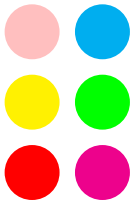

$T$

# MAX MOTIF – Innaproximabilité

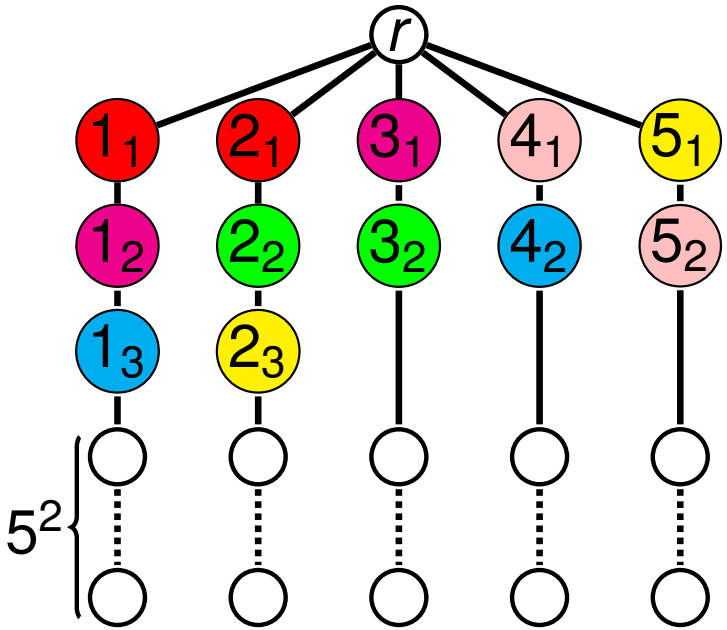
- Coloration (2 occurences...) + motif (simple)




$G = (V_G, E_G)$



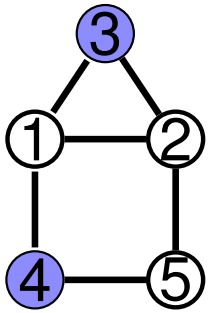
$M$



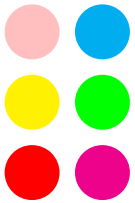

$T$

# MAX MOTIF – Innaproximabilité

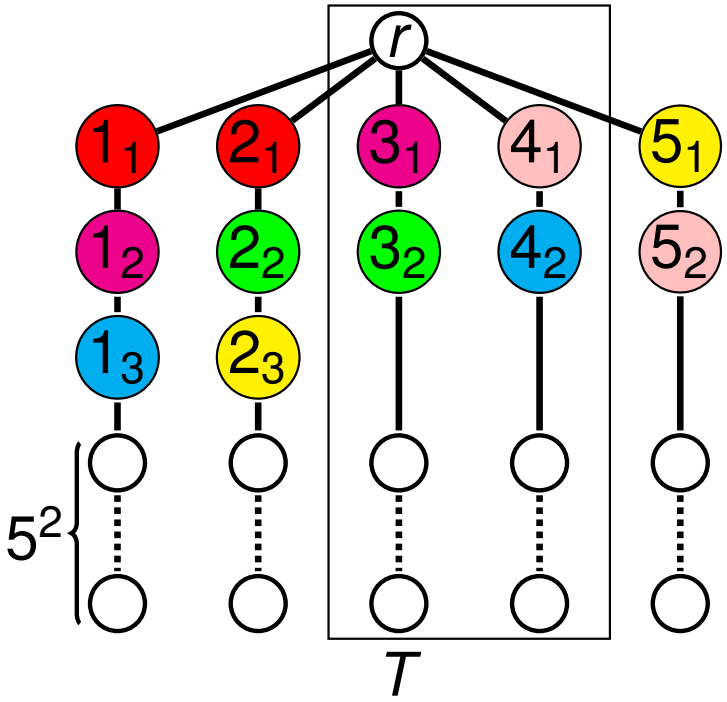
- INDEPENDANT SET $_{APX} \Leftrightarrow$ MAX MOTIF colorful $_{APX}$




$G = (V_G, E_G)$

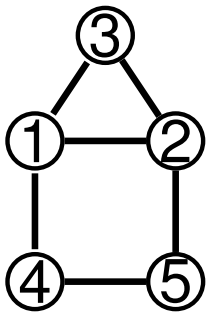


$M$

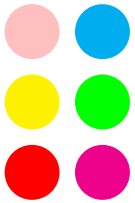


# MAX MOTIF – Innaproximabilité

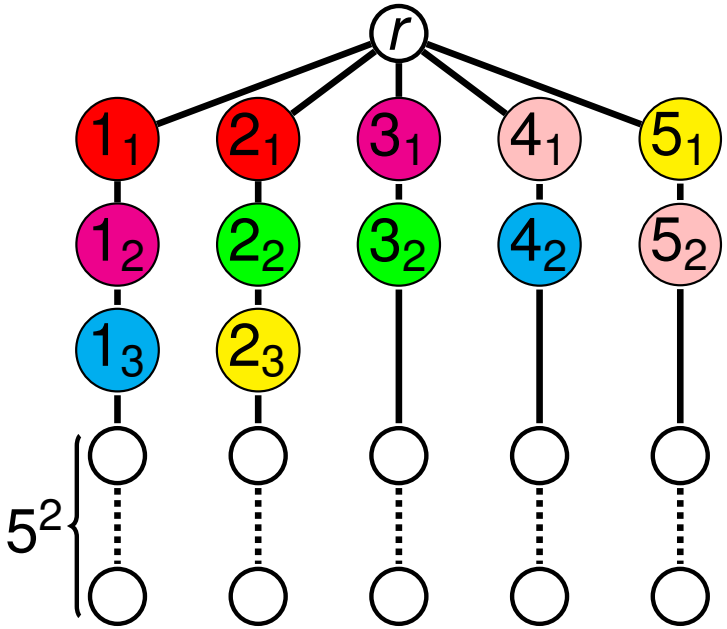
- On prouve que s'il existait un algo d'approx. pour MAX MOTIF , il y aurait un algo d'approx. pour INDEPENDANT SET




$G = (V_G, E_G)$



$M$




$T$

# Logiciels pour GRAPH MOTIF

- Torque [BRUCKNER ET AL. 2009] : un web service, qui ne gère que les motifs colorfuls
- GraMoFoNe [BLIN ET AL. 2010] : un plugin cytoscape

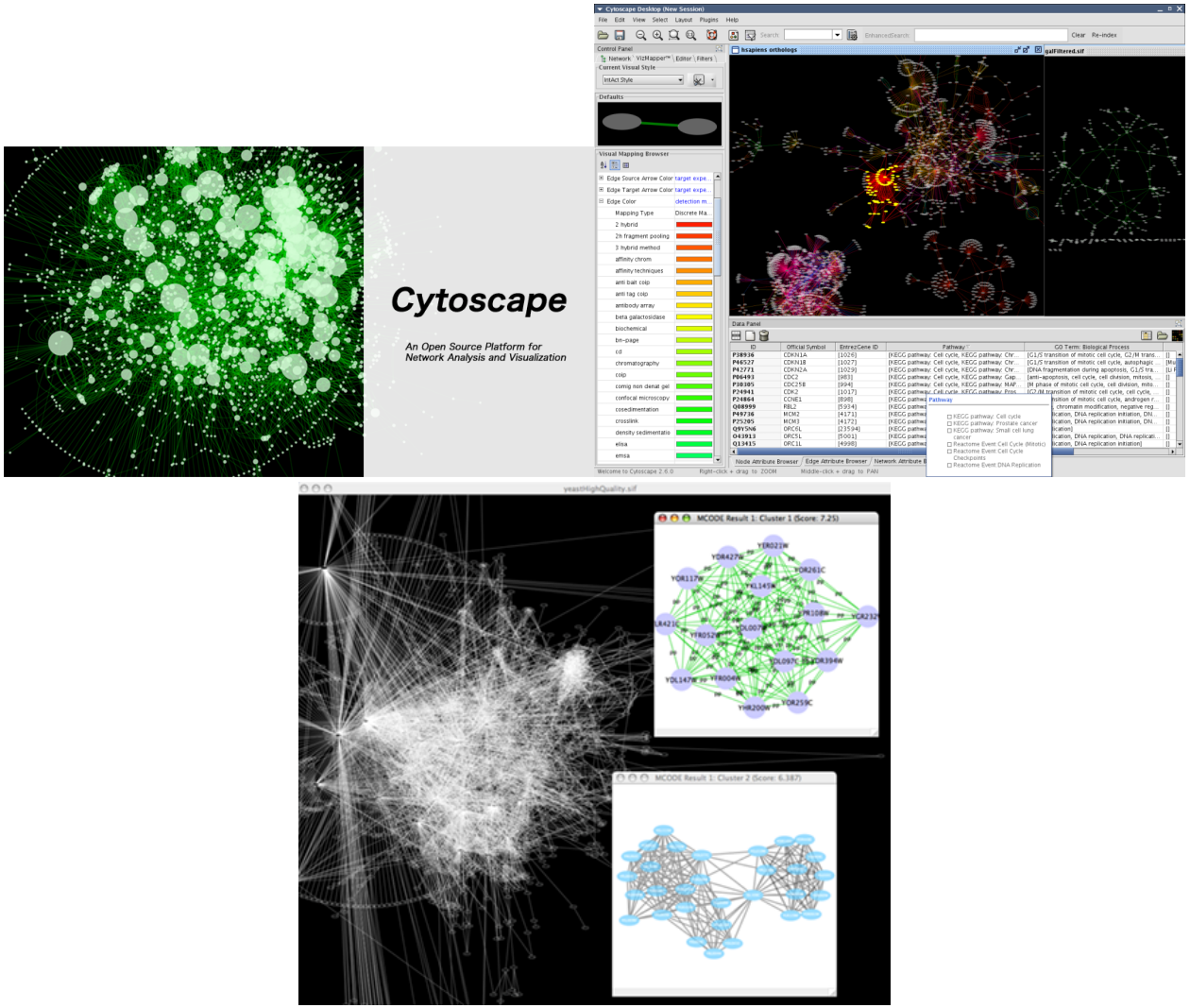
# GraMoFoNe – Cytoscape (2002)

- Plateforme java open-source, gratuite, permettant
  - l'importation / l'exportation nombreux formats, BDD,...
  - la visualisation et l'analyse de réseaux d'interactions
  - l'integration d'annotations à ces réseaux
- Largement utilisé ("des centaines" d'articles l'utilisent pour de l'analyse)
- Maintenu...

# GraMoFoNe – Cytoscape

- Principal avantage : plugins
- Ajout de fonctions :
  - Analyses
  - Nouveaux layouts
  - Support de fichiers
  - Connection avec des bases de données
  - ...

# GraMoFoNe – Cytoscape



# GraMoFoNe – PB

- On exprime GRAPH MOTIF avec de la programmation linéaire Pseudo-Booléenne
- *i.e.* de la programmation linéaire avec des variables booléennes

# GraMoFoNe – PB

- Trouver un assignement de variables satisfaisant les contraintes et maximisant l'objectif
- Un exemple simple :
  - **Variables** : $x_i \in \{0, 1\}, \forall i = 1, 2, 3$
  - **Objectif** : $\max x_1 + 2x_2 - x_3$
  - **Contraintes** :
    1. $x_1 - 2x_2 + 3x_3 \geq 1$
    2. $x_1 + x_2 + x_3 = 1$
    3. $2x_1 + x_2 + x_3 < 3$
- Solution : $x_1 = 1, x_2 = 0, x_3 = 0$

# GraMoFoNe – PB

- Donne une solution exacte
- Beaucoup de solvers efficaces existent
- Dont beaucoup de gratuits ($\neq$ CPLEX)

# GraMoFoNe – PB

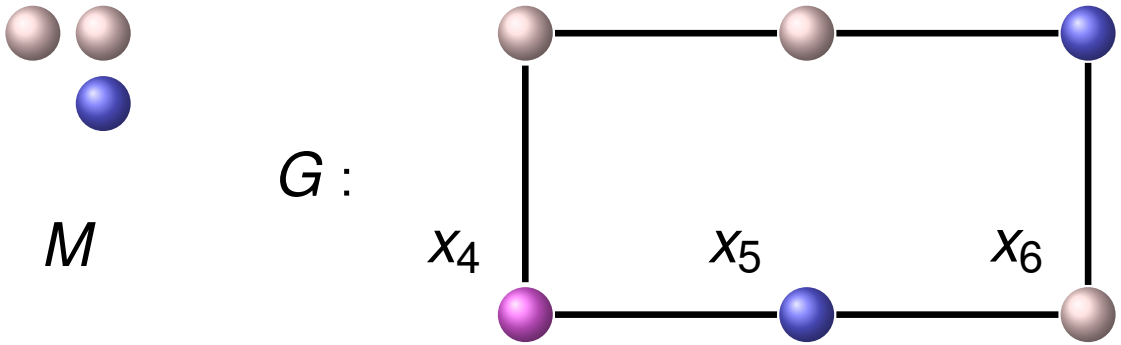
- Donne une solution exacte
- Beaucoup de solvers efficaces existent
- Dont beaucoup de gratuits ($\neq$ CPLEX)
- On utilise SAT4JPseudo [LE BERRE, PARRAIN 2007]
  - Gratuit
  - Maintenu
  - Bonnes performances lors des compétitions
  - En java

# GraMoFoNe – PB

- On utilise 23 contraintes et 9 domaines de variables
- On cherche une occurence d'un motif $M$ dans un graphe $G$
- A respecter :
  1. La taille de la solution
  2. La coloration de la solution par rapport au motif
  3. La connexité de la solution

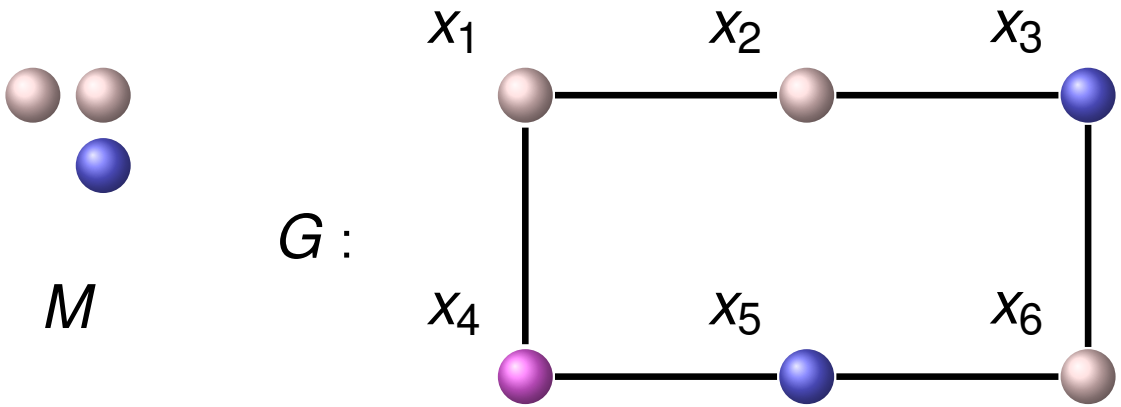
# GraMoFoNe – Variables

- Une variable $x$ pour chaque nœud



# GraMoFoNe – Variables

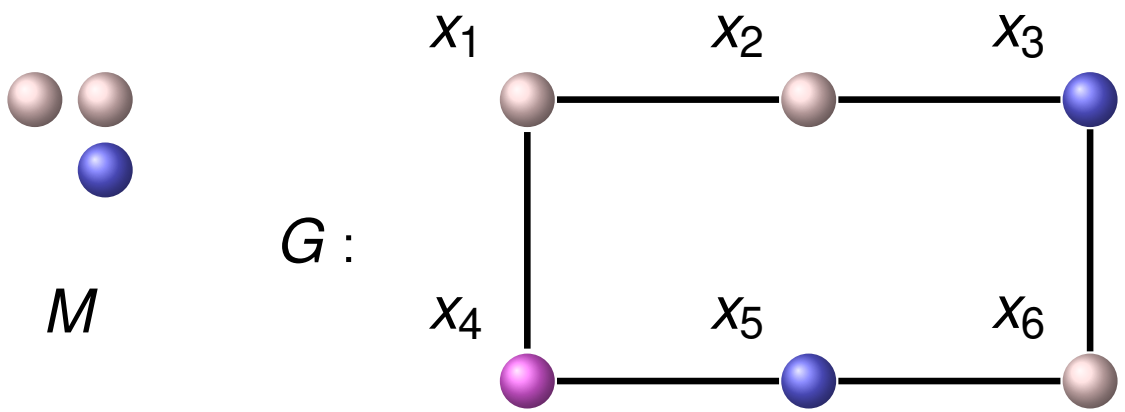
- Une variable $x$ pour chaque nœud




- Contrainte ”taille solution” : $\sum_{v \in V} x_v = |M|$ :
  - $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 3$

# GraMoFoNe – Variables

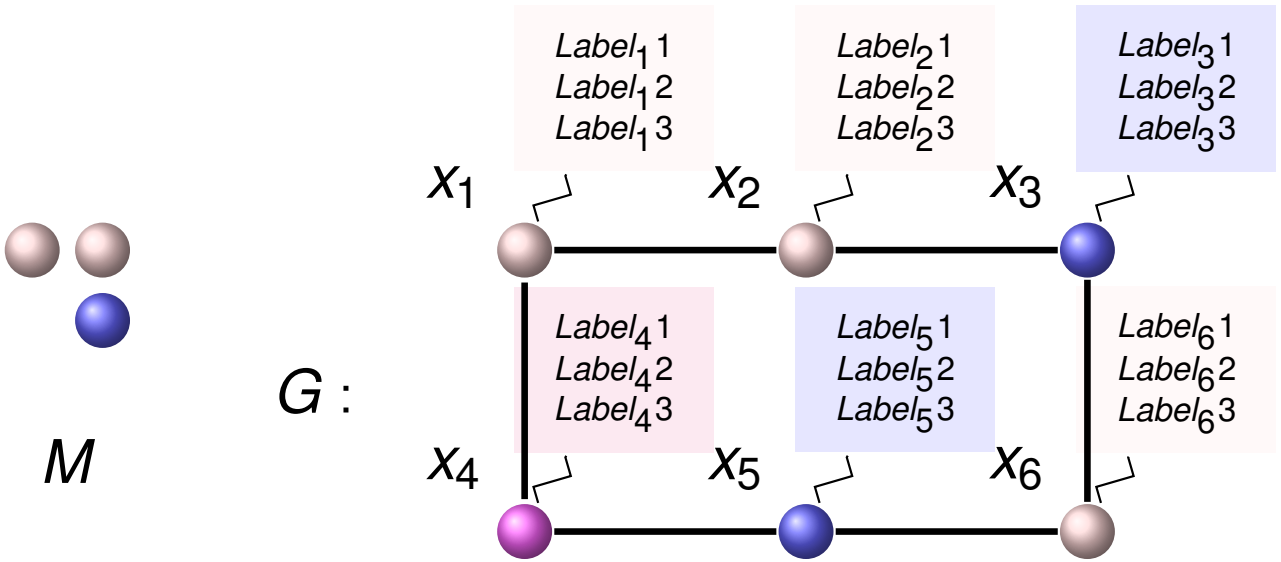
- Une variable $x$ pour chaque nœud




- Contrainte ”taille solution” : $\sum_{v \in V} x_v = |M|$ :
  - $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 3$
- Contraintes ”coloration” $\sum_{\substack{v \in V \\ c \in col(v)}} x_v = occ_M(c)$:
  - $x_1 + x_2 + x_6 = 2$ (gris)
  - $x_3 + x_5 = 1$ (bleu)
  - $x_4 = 0$ (rose)

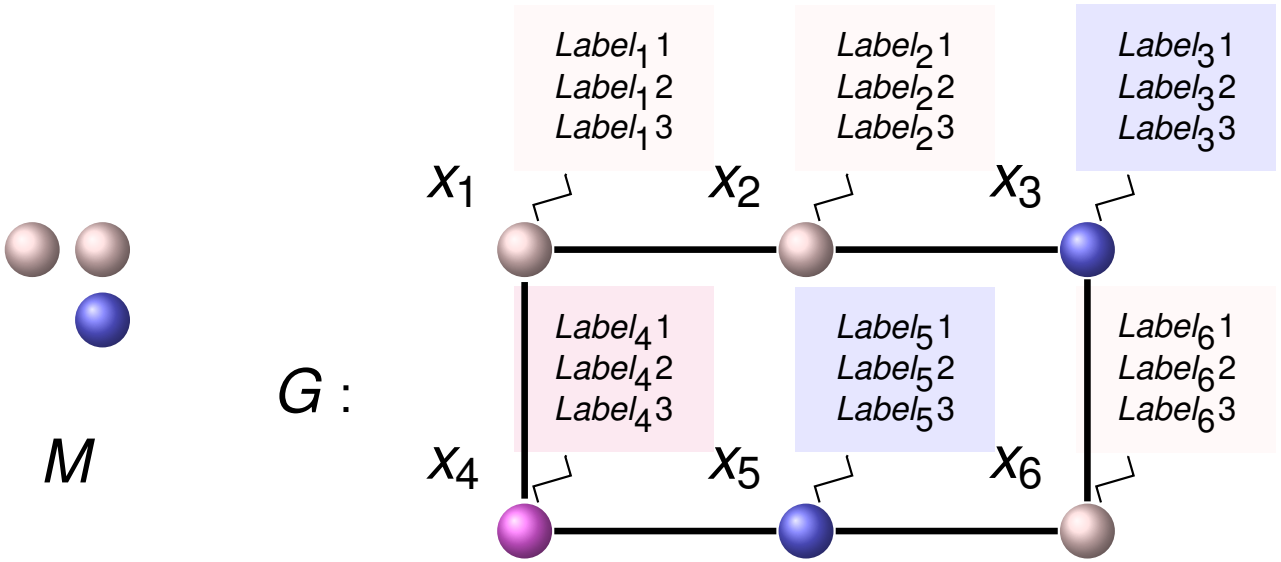
# GraMoFoNe – Variables

- Connexité : $|M|$ variables $Label_v$ pour chaque nœud $v$



# GraMoFoNe – Variables

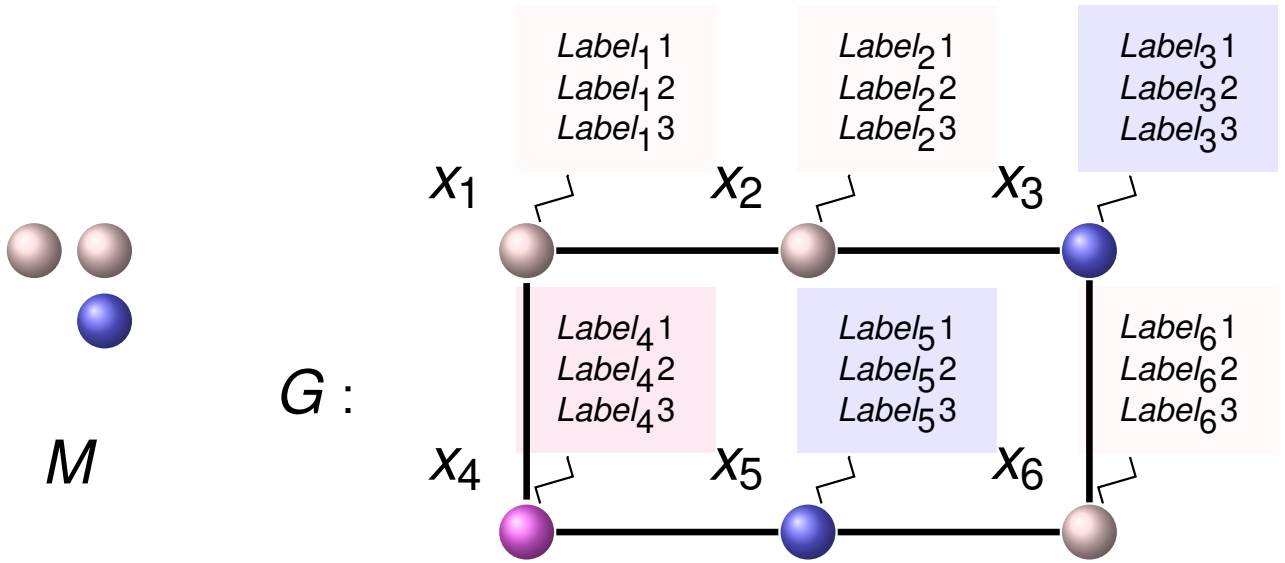
- Connexité : $|M|$ variables $Label_v$ pour chaque nœud $v$




- $|V|$ contraintes "un seul label par nœud dans la solution" :
  - Pour chaque $v$, $x_v \Rightarrow (\sum_{i=1}^{|M|} Label_v i = 1)$

# GraMoFoNe – Variables

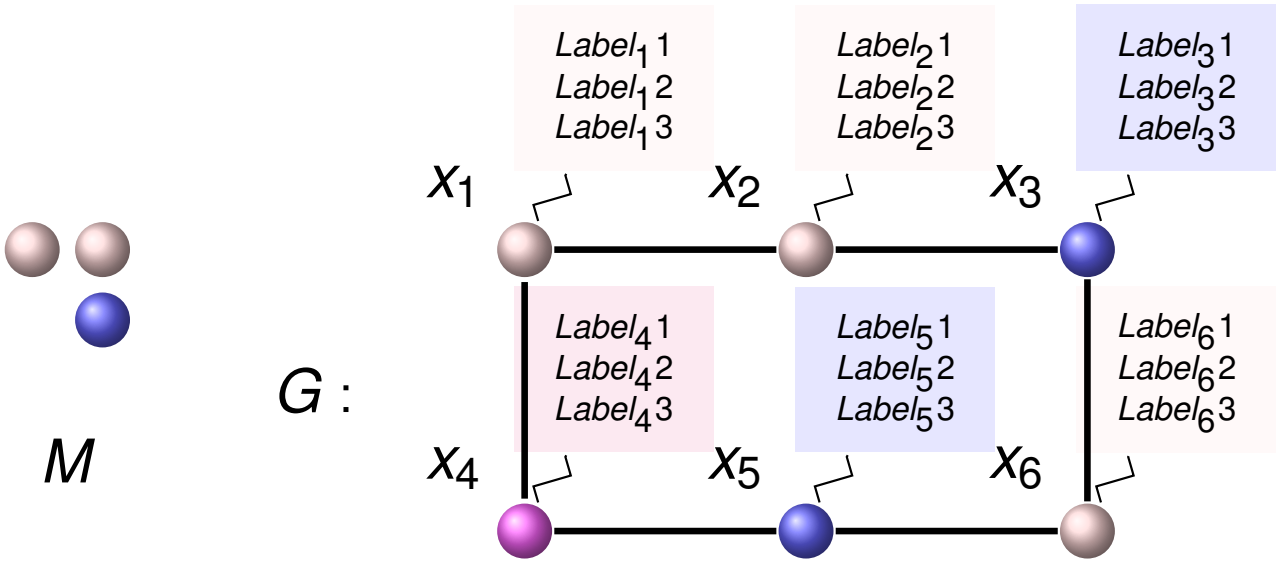
- Connexité : $|M|$ variables $Label_v$ pour chaque nœud $v$




- $|V|$ contraintes ”un seul label par nœud dans la solution” :
  - Pour chaque $v$, $x_v \Rightarrow (\sum_{i=1}^{|M|} Label_v i = 1)$
- $|M|$ contraintes “un seul nœud par label donné”
  - Pour un label $i$ donné, $\sum_{v \in V} Label_v i = 1$

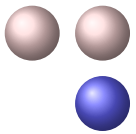
# GraMoFoNe – Variables

- Connexité : $|M|$ variables $Label_v$ pour chaque nœud $v$



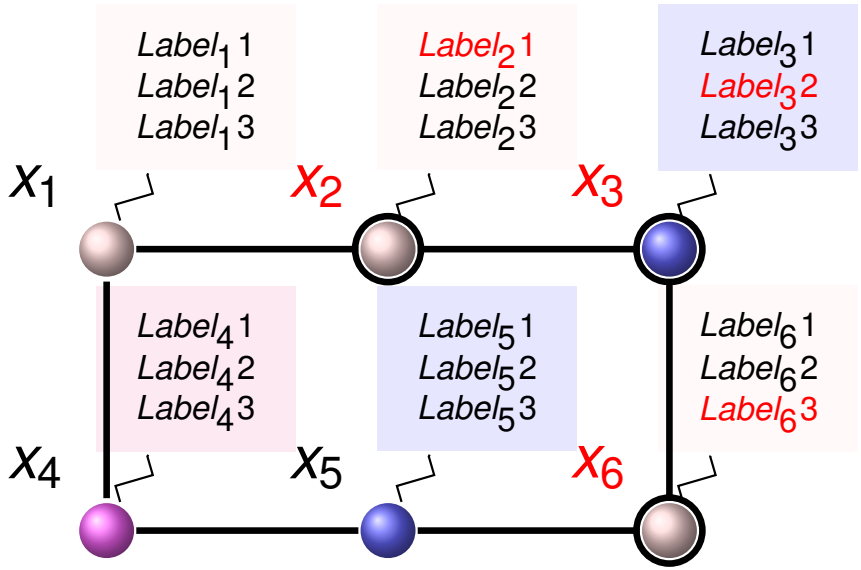

- $|V|$ contraintes "un seul label par nœud dans la solution" :
  - Pour chaque $v$, $x_v \Rightarrow (\sum_{i=1}^{|M|} Label_v i = 1)$
- $|M|$ contraintes "un seul nœud par label donné"
  - Pour un label $i$ donné, $\sum_{v \in V} Label_v i = 1$
- $|V|.|M|$ contraintes "un nœud avec un label a un voisin avec un label supérieur" (sauf le dernier)
  - $Label_v i \Rightarrow (\sum_{u \in N(v)} \sum_{j>i} Label_u j \geq 1)$

# GraMoFoNe – Une solution

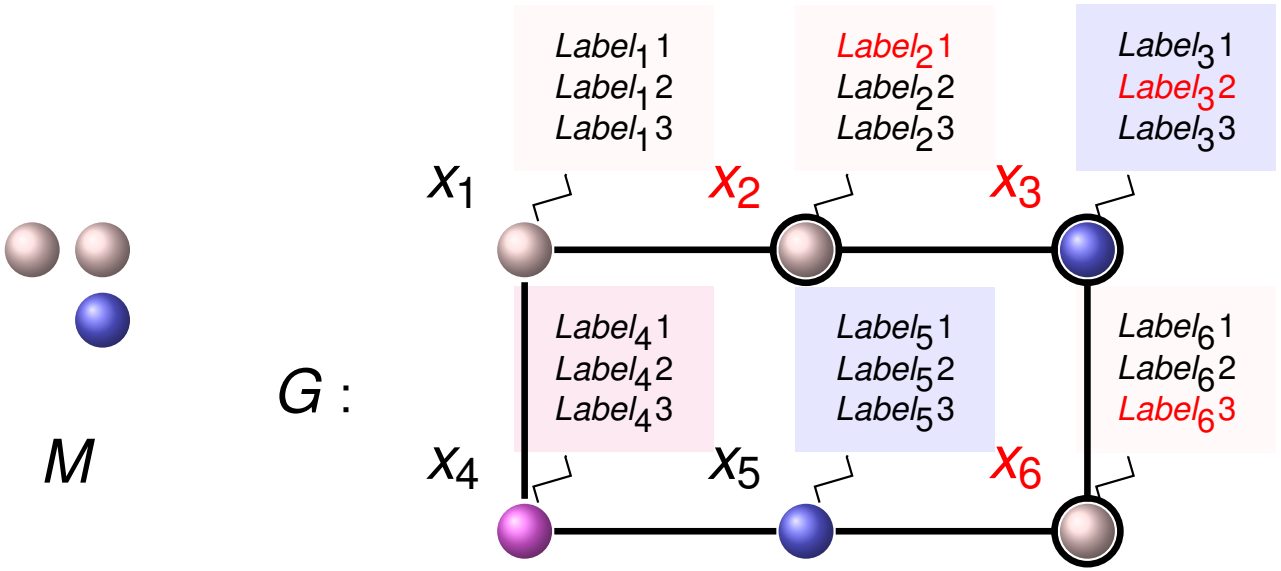


$M$

$G$ :

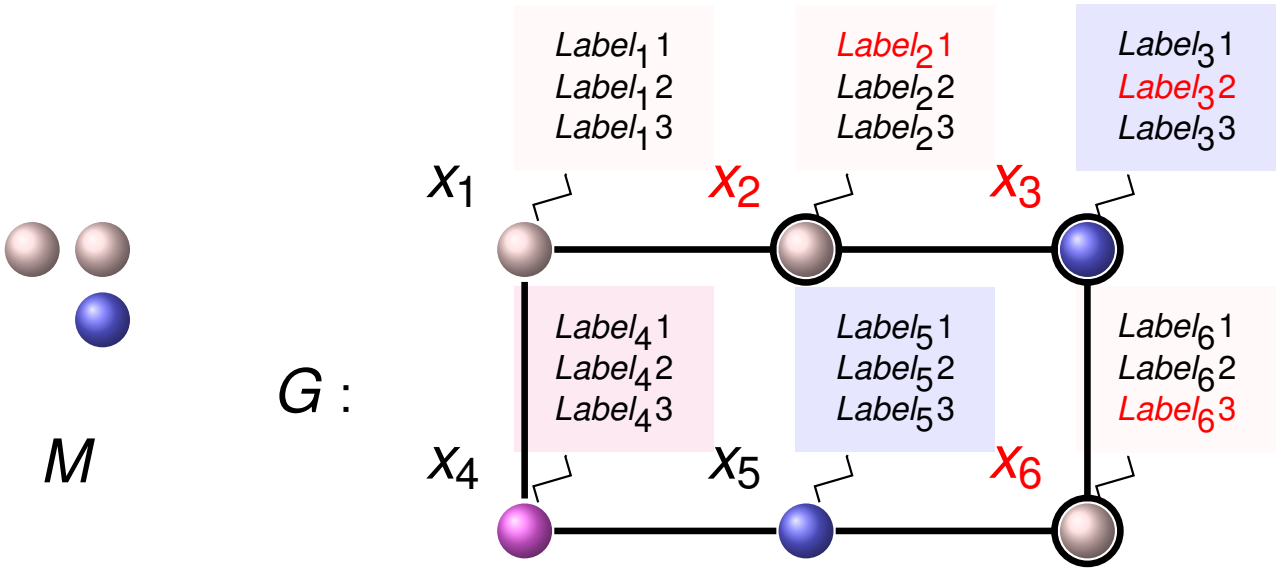


# GraMoFoNe – Une solution




- "taille" : $x_1 + x_2 + x_3 + x_4 + x_5 + x_6 = 3$
- "coloration":
  - $x_1 + x_2 + x_6 = 2$ (gris)
  - $x_3 + x_5 = 1$ (bleu)
  - $x_4 = 0$ (rose)

# GraMoFoNe – Une solution




- "un label par nœud" : $\forall v,\ x_v \Rightarrow (\sum_{i=1}^{|M|} Label_v i = 1)$
- "un nœud par label" : $\sum_{v \in V} Label_v i = 1$
- "voisin avec label supérieur" :
  $Label_v i \Rightarrow (\sum_{u \in N(v)} \sum_{j>i} Label_u j \geq 1)$

# GraMoFoNe – PB

- Gère GRAPH MOTIF ”classique” (colorful et multi-ensemble)
- PB permet d’avoir toutes les solutions possibles...

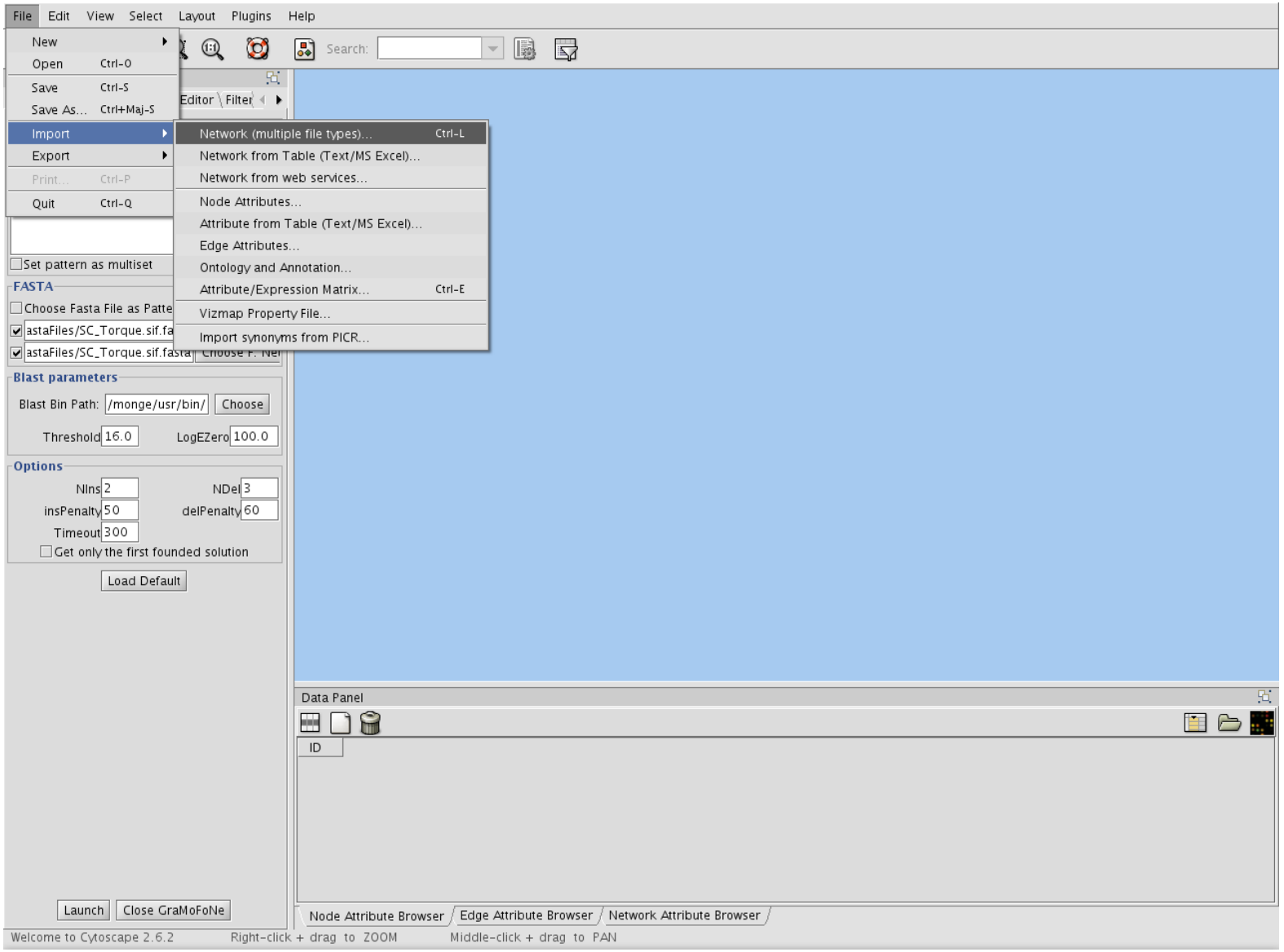
# GraMoFoNe – PB

- Gère GRAPH MOTIF ”classique” (colorful et multi-ensemble)
- PB permet d’avoir toutes les solutions possibles...
- Avec variables et contraintes supplémentaires, on peut gérer
  - Les insertions
  - Les deletions
  - Une liste de couleurs associée à chaque nœud du graphe

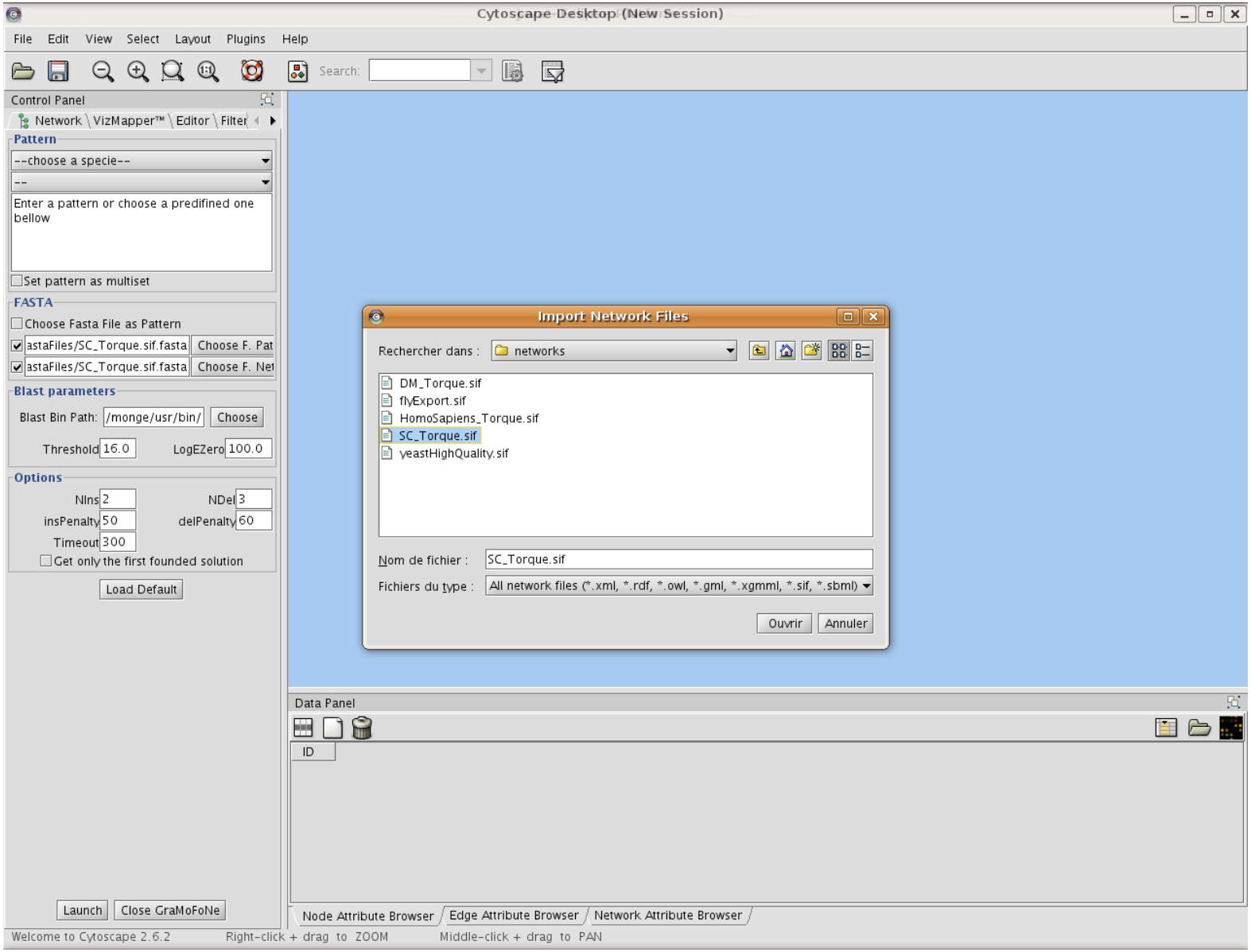
# GraMoFoNe – PB

- Gère GRAPH MOTIF ”classique” (colorful et multi-ensemble)
- PB permet d’avoir toutes les solutions possibles...
- Avec variables et contraintes supplémentaires, on peut gérer
  - Les insertions
  - Les deletions
  - Une liste de couleurs associée à chaque nœud du graphe
- Devient complexe (compensations “une deletion compense une insertion”, garder la bijection entre motif et graphe si plusieurs couleurs dans le graphe,...)

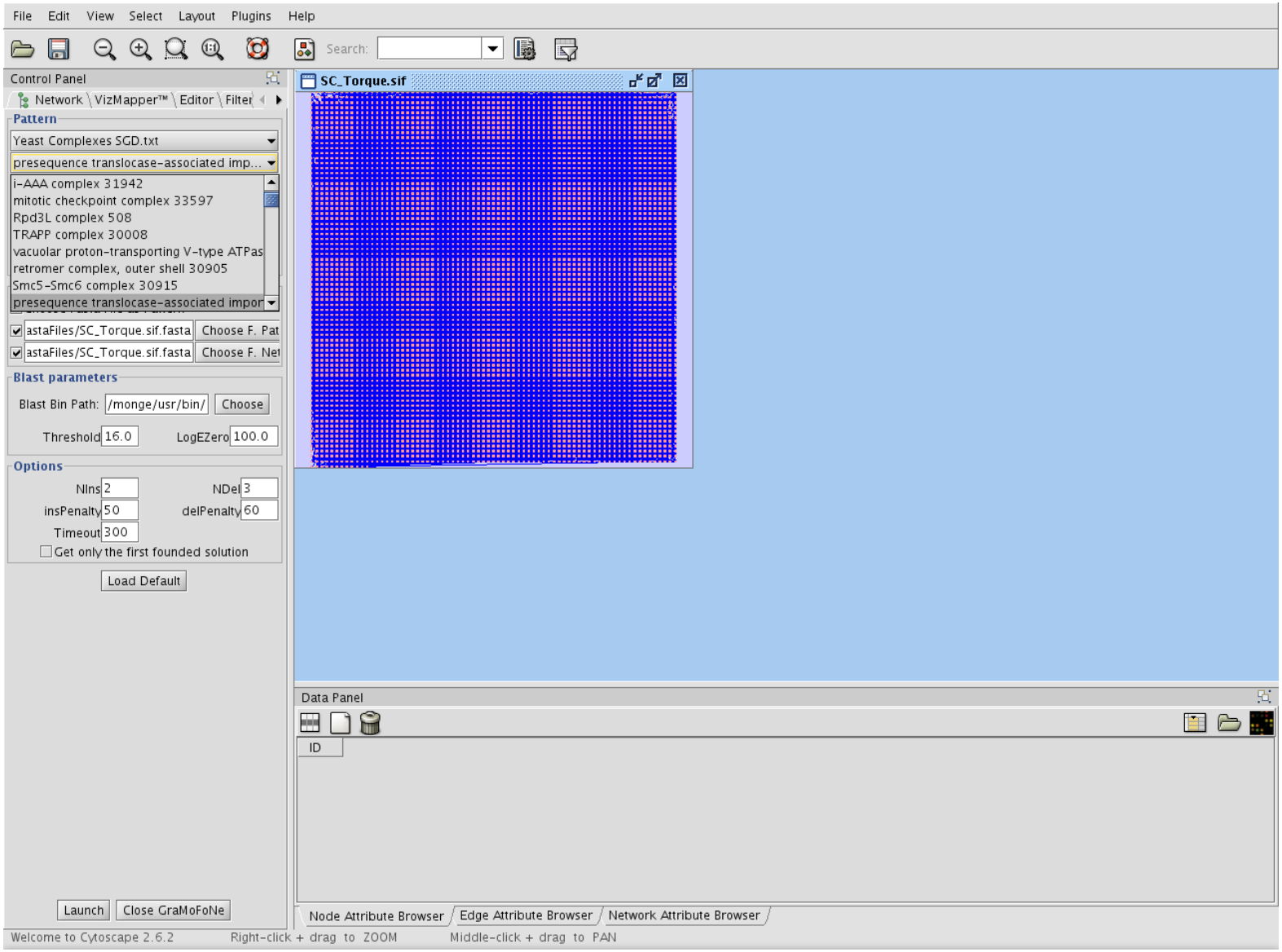
# GraMoFoNe – IG



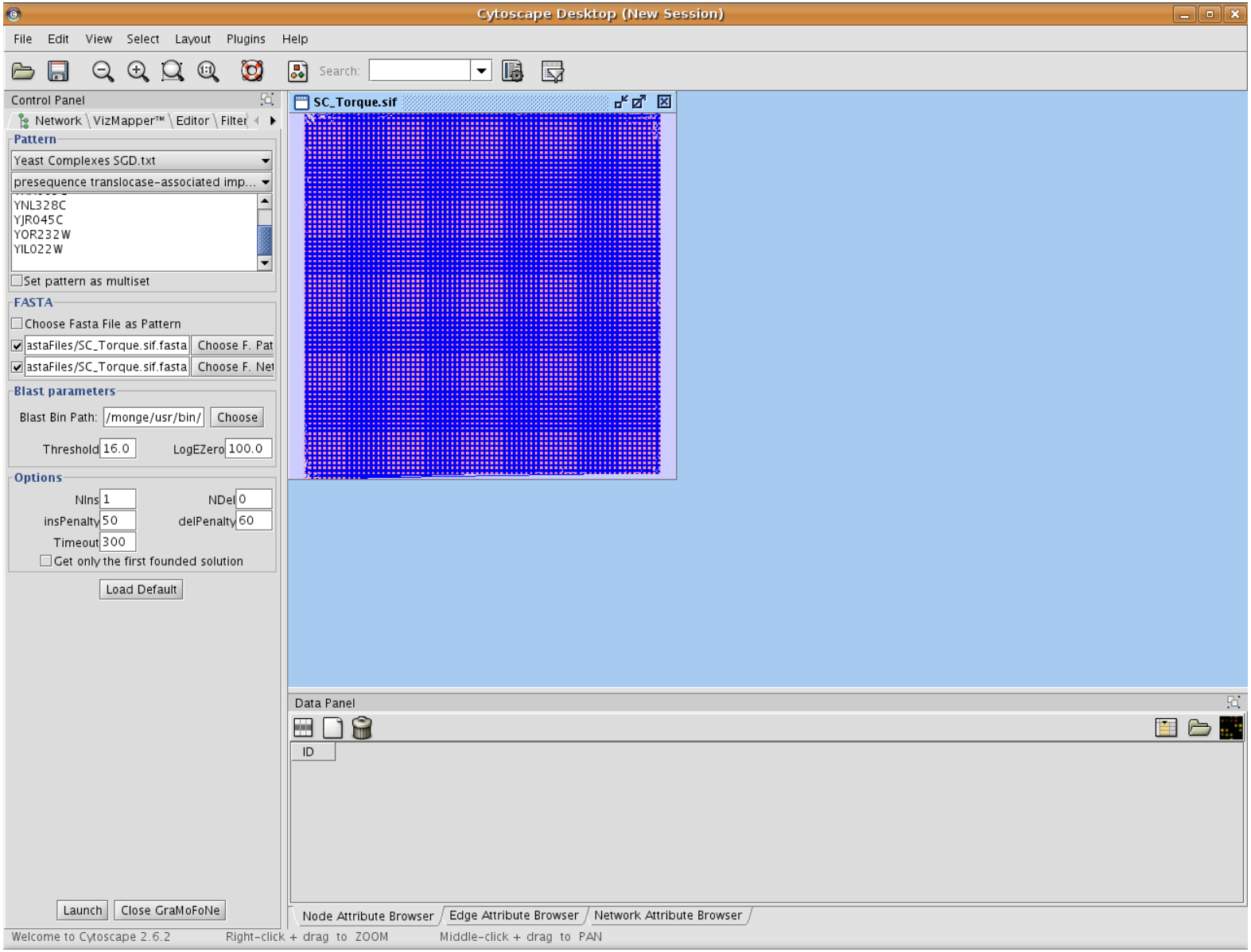
# GraMoFoNe – IG



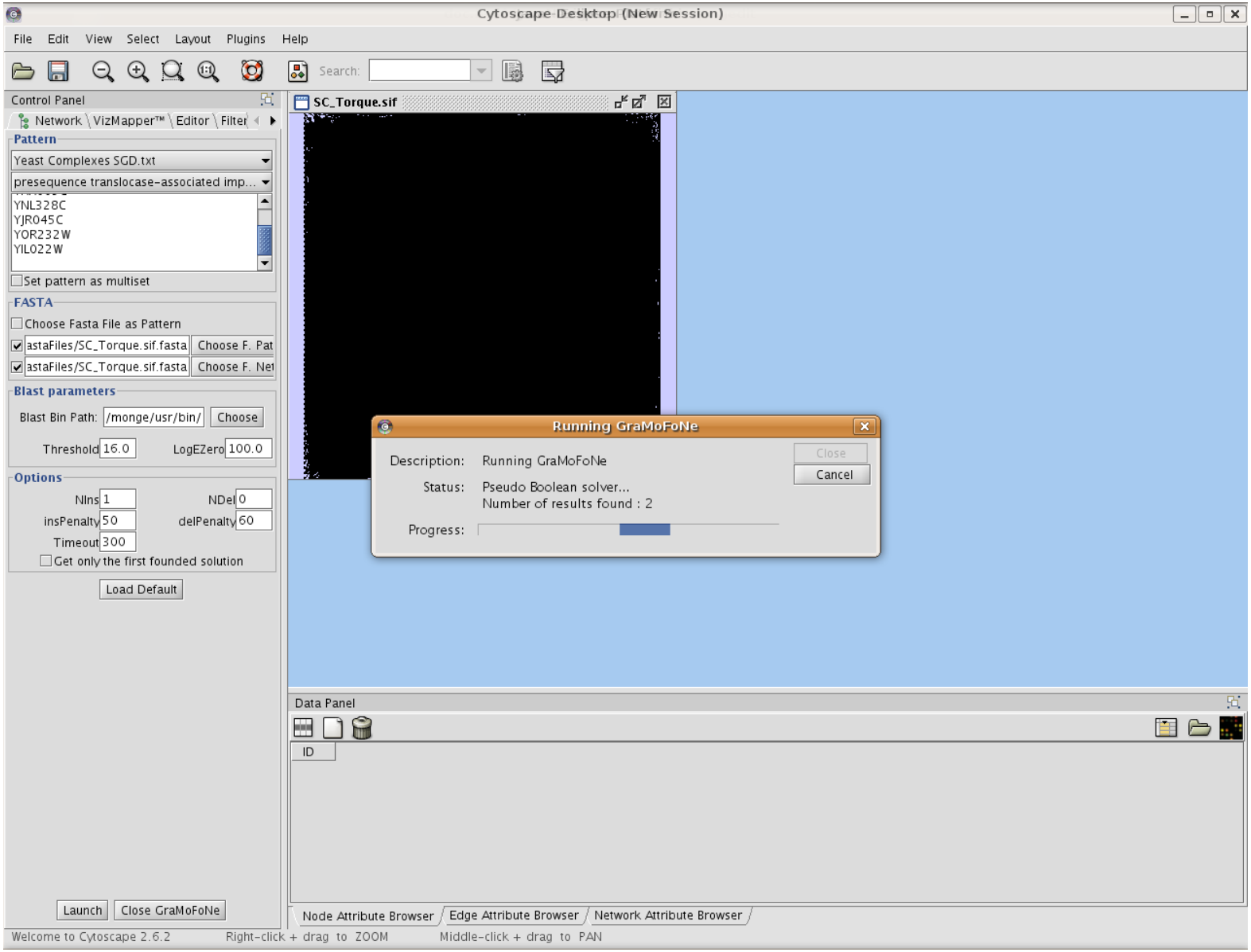
# GraMoFoNe – IG



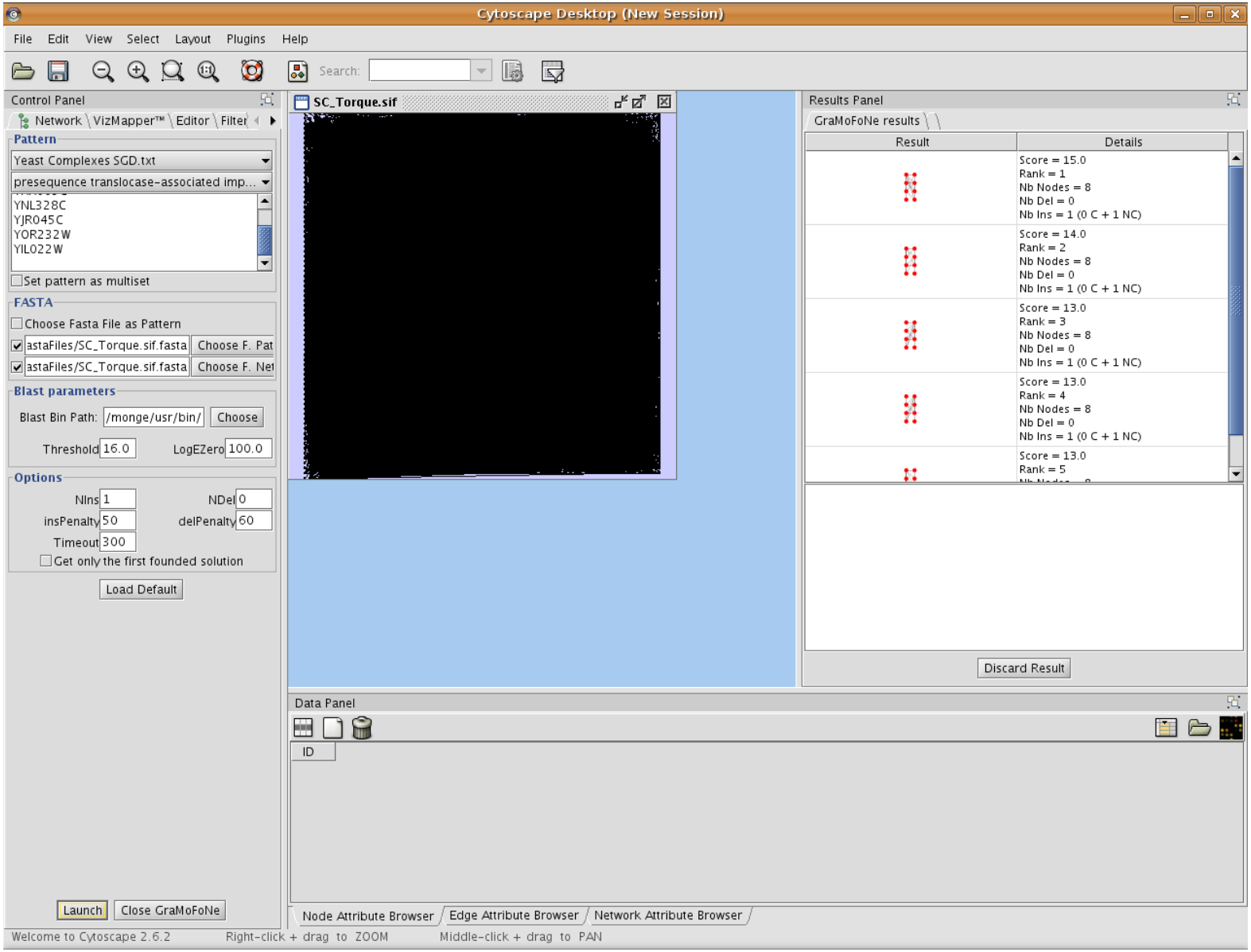
# GraMoFoNe – IG



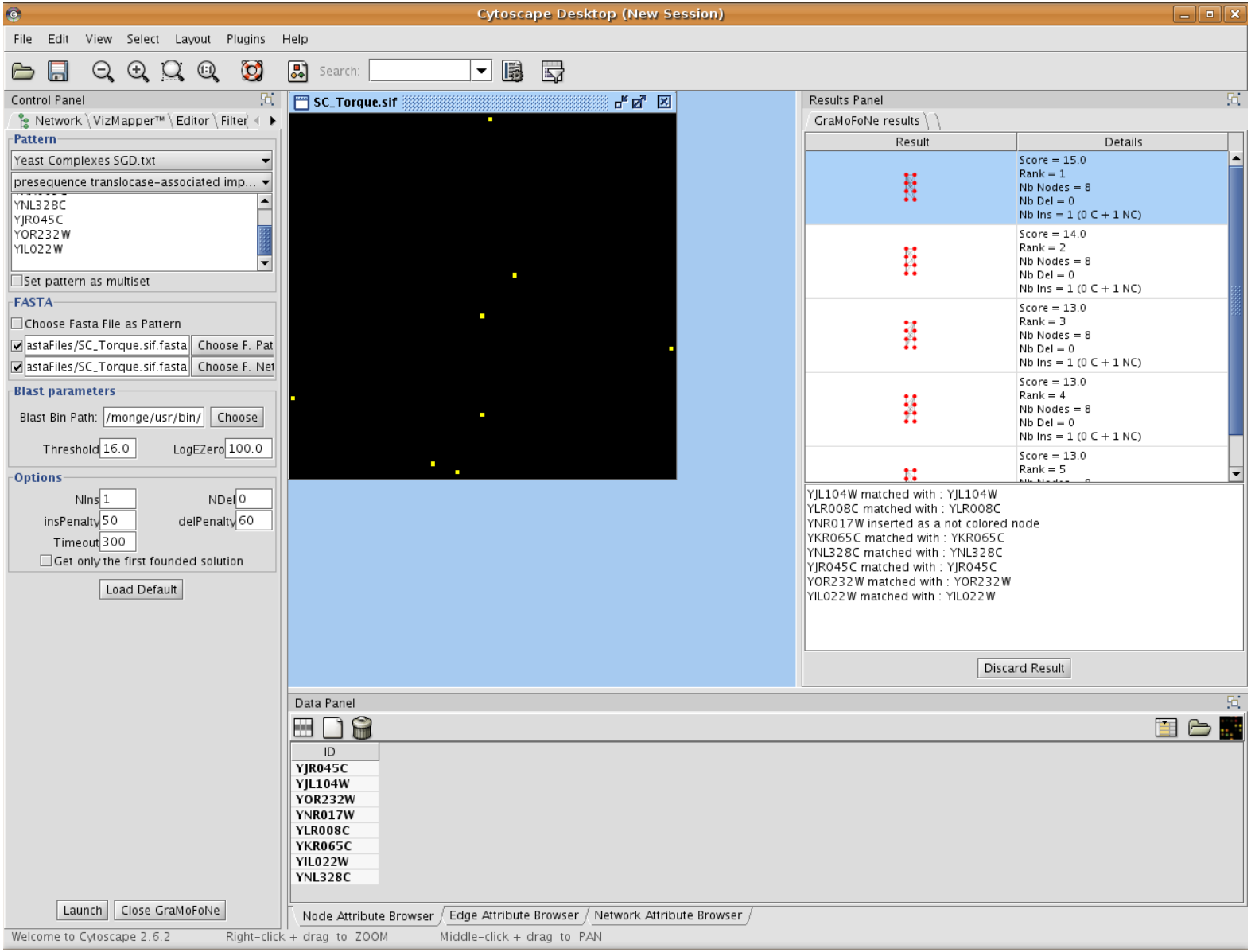
# GraMoFoNe – IG



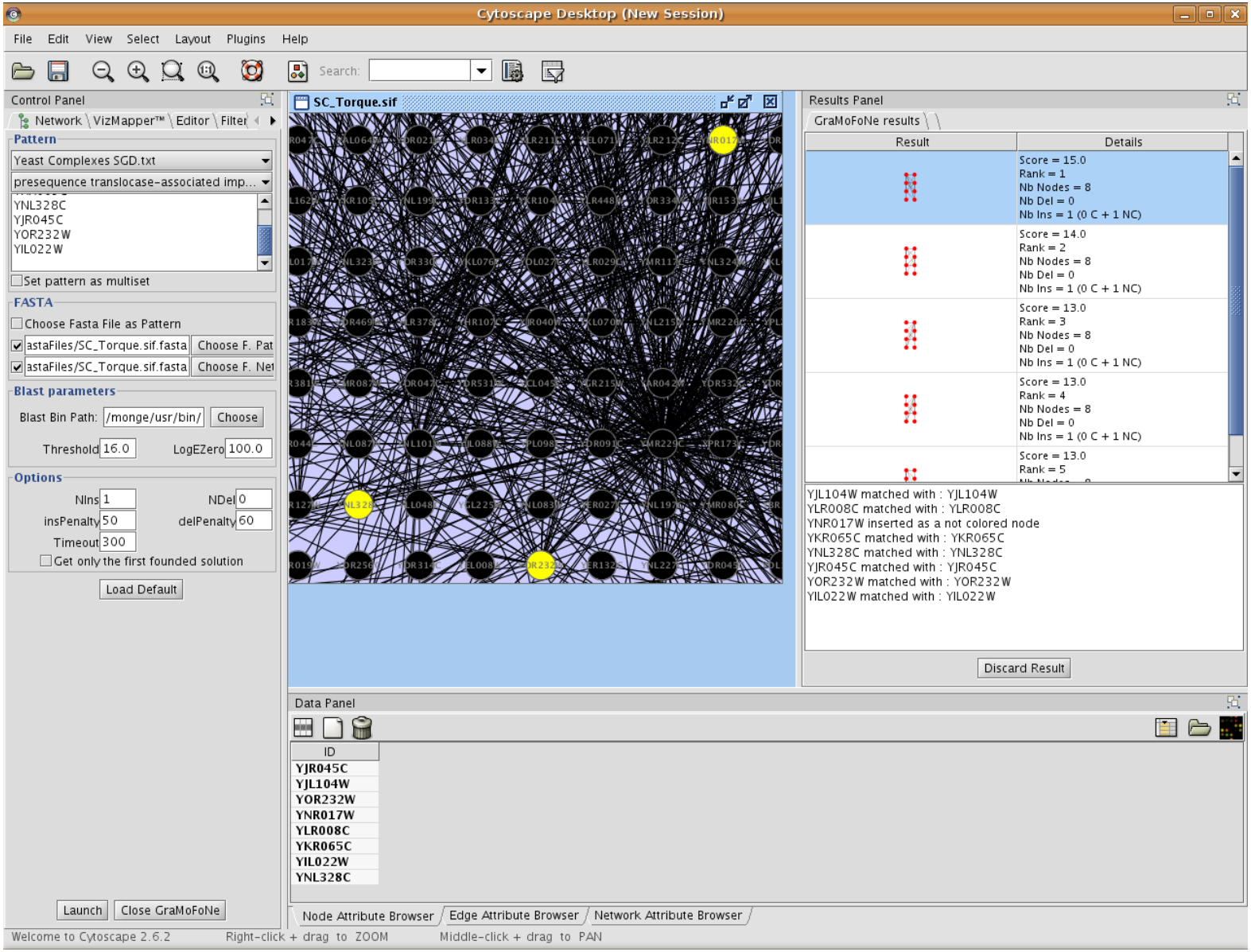
# GraMoFoNe – IG



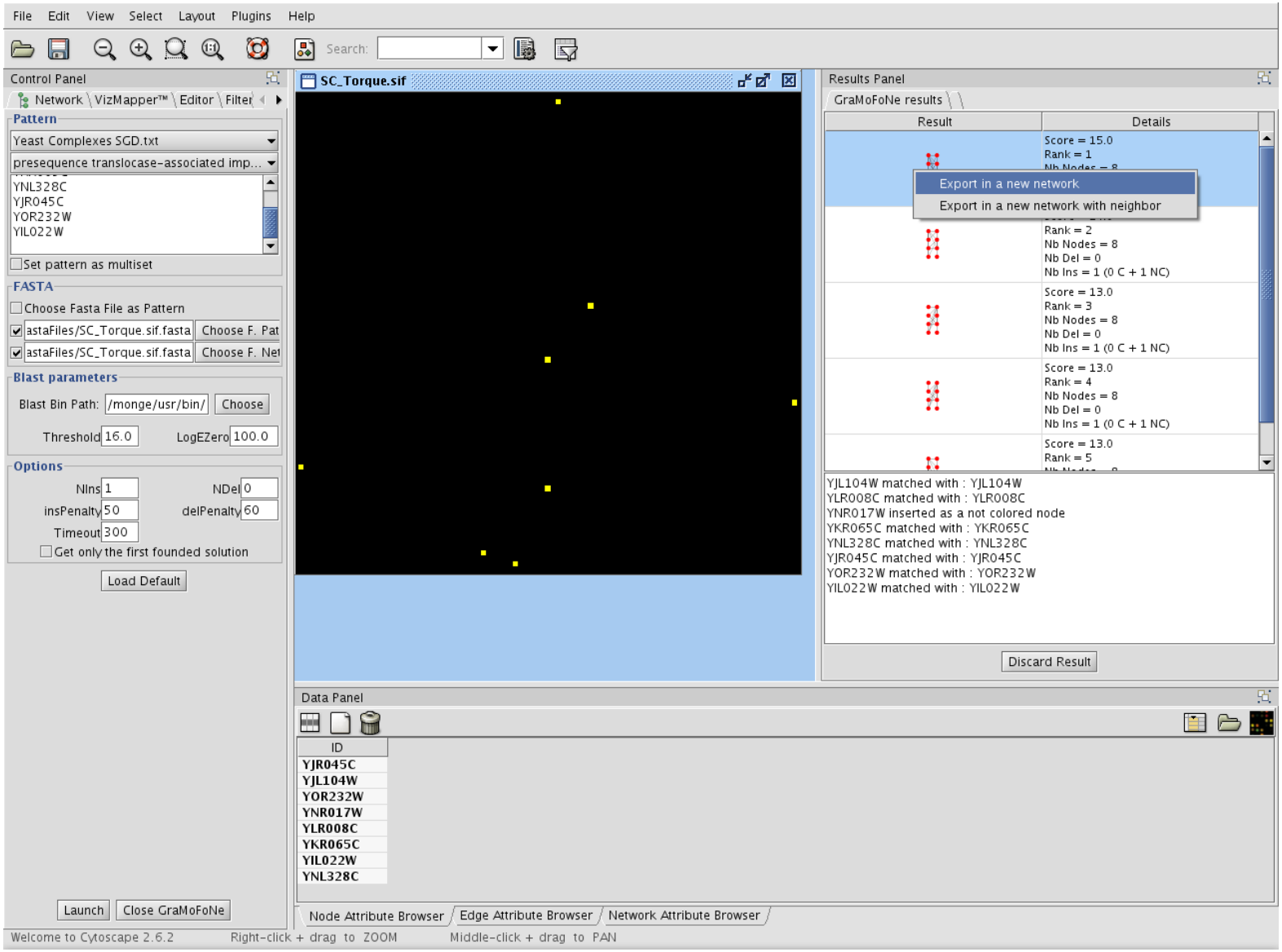
# GraMoFoNe – IG



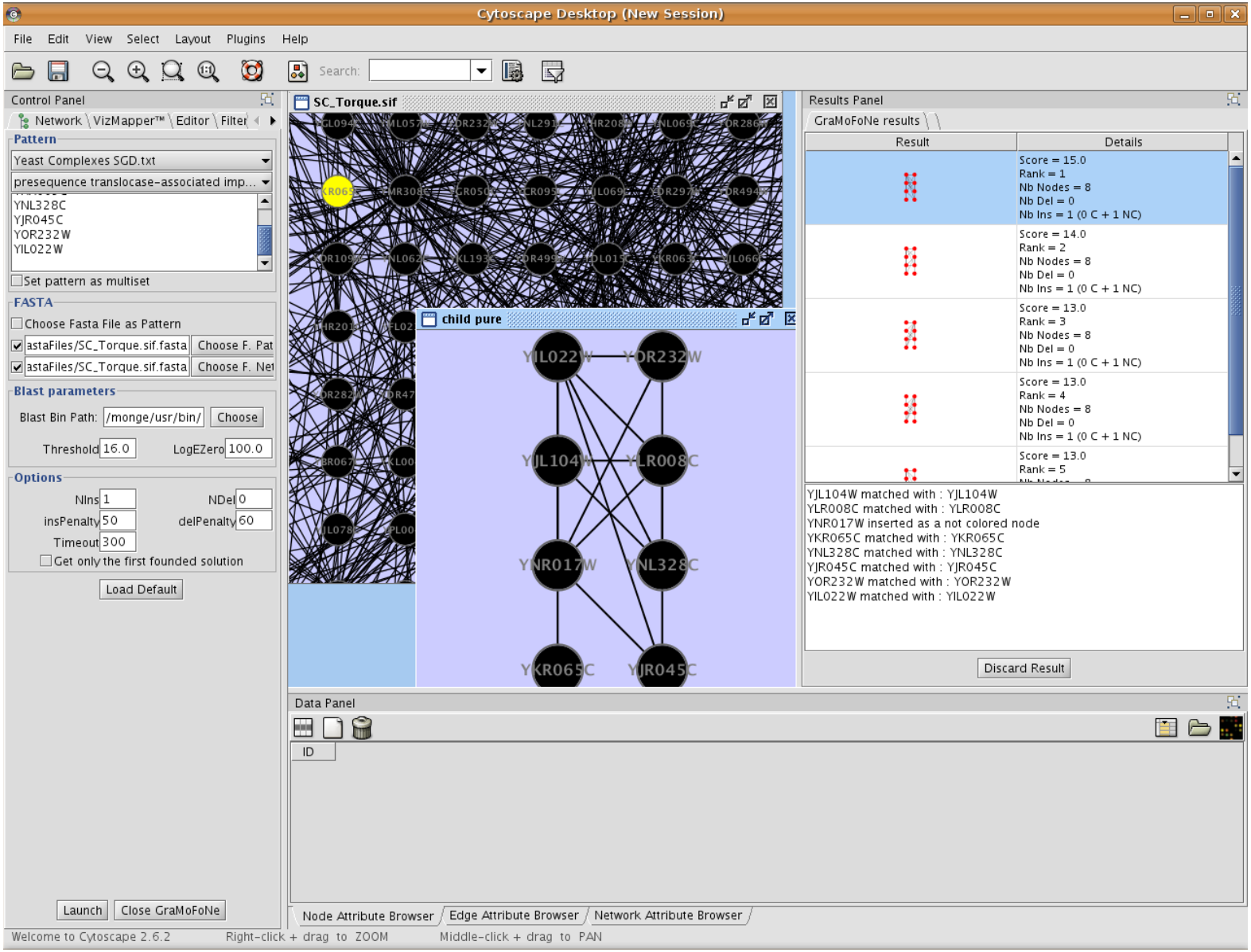
# GraMoFoNe – IG



# GraMoFoNe – IG



# GraMoFoNe – IG



# GraMoFoNe – “batch mode”

- Utilisé pour tests grande échelle
- Recherche de dizaines de complexes protéiques dans des réseaux d’autres espèces

# GraMoFoNe – “batch mode”

- Utilisé pour tests grande échelle
- Recherche de dizaines de complexes protéiques dans des réseaux d’autres espèces
- Données :
  - Motifs de 6 espèces (levure, drosophile, homo sapiens, souris, boeuf, rat)
  - Réseaux de 3 espèces (levure, drosophile, homo sapiens)
  - De 2 à 4 indels autorisés selon la taille du motif

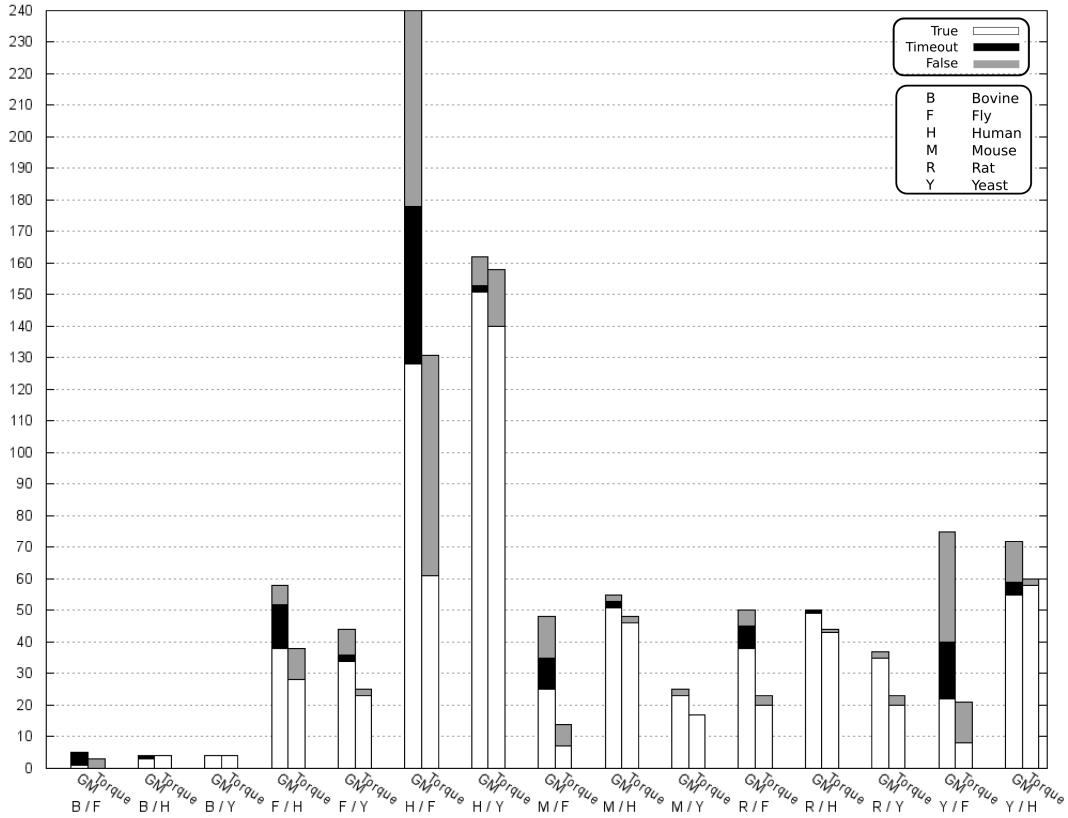
# GraMoFoNe – “batch mode”

- Preprocessing
  1. Les protéines du motif sans homologues sont “deleted” d’office
  2. Si un nœud coloré du réseau est “loin” d’un autre nœud coloré du réseau, on le supprime

# GraMoFoNe – “batch mode”

- Un motif est “faisable” si
  1. Taille 4-25
  2. Pas plus du nombre fixé de deletions de protéines sans homologues
  3. Une composante connexe avec assez de couleurs
- Chaque motif est ensuite
  1. Trouvé dans le temps imparti
  2. Marqué comme non trouvé dans le temps imparti
  3. Inconnu (temps imparti écoulé)

# GraMoFoNe – “batch mode”




- $\neq$ du au preprocessing ? A l’ajout d’information Fasta ?
- + de bruit pour la mouche (faux negatifs déconnectent la solution, faux positifs donnent des “mauvaises” solutions)
- 5-20s (petits *M*), 40-60s (grands). Mais PB “boite noire”

# Plan

# Conclusion

- Deux vues pour la recherche de motifs
  1. Motifs avec topologie
  2. Motifs sans topologie
- Problèmes difficiles donc :
  - de complexité paramétrée
  - d'approximation

# Conclusion

- Le bruit pousse à la recherche d'extensions plus souples
  - contrainte couleurs
  - contrainte connexité
    - Simple connexité : FPT
    - 2-connexité : W[1]-difficile
    - ?
  - ?
- Se passer du color-coding ?
  - Le derandomiser est non praticable (Hachage parfait : $88^k$)
  - La version randomisée demande un nombre exponentiel de relances

# Questions sur la recherche de motifs dans les graphes colorés ?

Florian Sikora
(encadré par Guillaume Blin et Stéphane Vialette)

Université Paris-Est, LIGM - UMR CNRS 8049

Séminaire Symbiose – 11/02/2010