Comment trouver des problèmes difficiles faciles ?

Florian Sikora

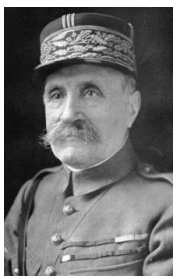
Plan

Difficulté

Théorème du doigt mouillé

La plupart des problèmes combinatoires intéressants sont NP-difficiles

- Approuvé par F. Foch : "*Ne me dites pas que ce problème est difficile. S'il n'était pas difficile, ce ne serait pas un problème.*"



Difficulté

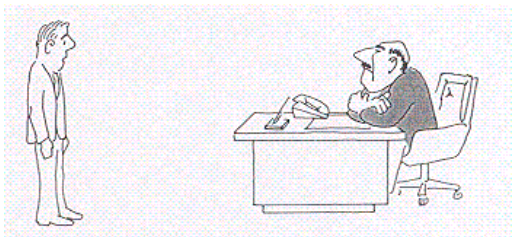
Théorème du doigt mouillé

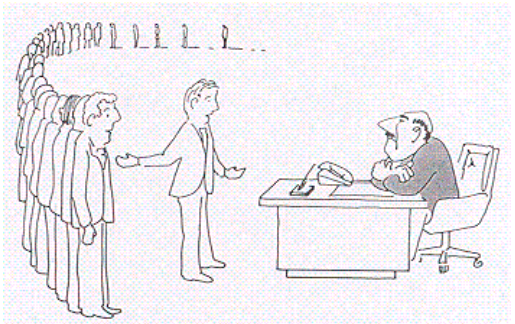
La plupart des problèmes combinatoires intéressants sont NP-difficiles

- Approuvé par F. Foch : "*Ne me dites pas que ce problème est difficile. S'il n'était pas difficile, ce ne serait pas un problème.*"
- Problèmes NP-**complets** :
  - N'existe probablement **pas d'algorithme exact** de complexité pire cas **polynomiale** (par ex. $\mathcal{O}(n^c)$).

Difficulté

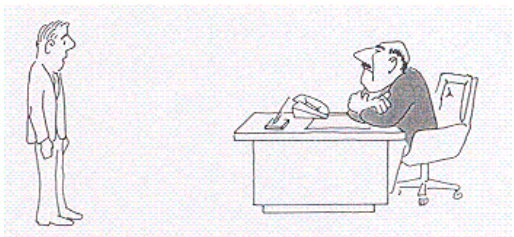
- Garey and Johnson 1979.
  - "Je suis trop bête pour trouver un algorithme efficace pour ce problème..."
  - "... comme toutes ces personnes reconnues !"

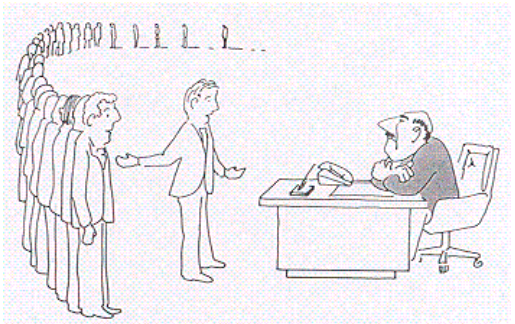




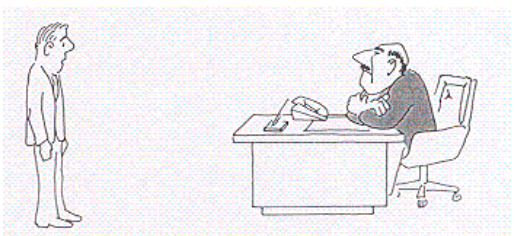
Difficulté

- Garey and Johnson 1979.
  - "Je suis trop bête pour trouver un algorithme efficace pour ce problème..."
  - "... comme toutes ces personnes reconnues !"





- Adaptation D. Hermelin 2009.
  - "Et alors ?! J'ai besoin d'une solution !"



Que faire ?

- Pour contourner la difficulté :
  - Trouver des instances plus simples.
  - Complexité paramétrée.
  - Algorithmes modérément exponentiels.
  - Approximation (polynomiale).
  - Approximation non polynomiale.
  - ILP.
  - ...

Complexité paramétrée

"*Mesurer la complexité seulement en fonction de la taille de la donnée signifie ignorer toute information structurelle sur l'instance donnée...*"

J. Flum et M. Grohe

"*Question : Quand un problème venant de la "vie réelle" n'a pas d'autre structure que sa taille ?*
Réponse : Jamais !"

R. Downey et M. Fellows

Complexité paramétrée

"*Mesurer la complexité seulement en fonction de la taille de la donnée signifie ignorer toute information structurelle sur l'instance donnée...*"

J. Flum et M. Grohe

"*Question : Quand un problème venant de la "vie réelle" n'a pas d'autre structure que sa taille ?*
Réponse : Jamais !"

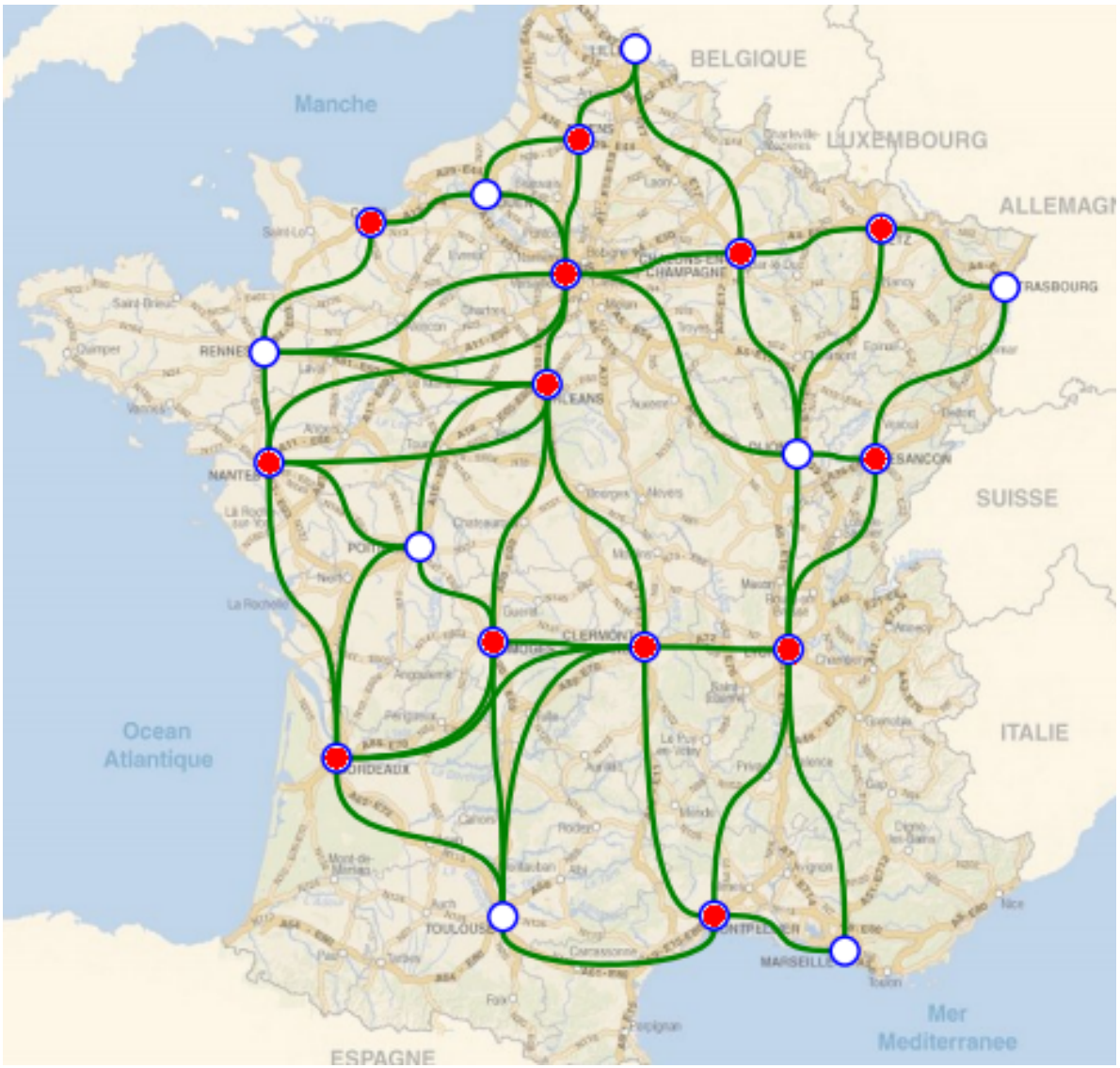
R. Downey et M. Fellows

"*L'idée fondamentale est de restreindre l'explosion combinatoire, semble-t-il inévitable, qui est responsable de la croissance exponentielle du temps de calcul, à un paramètre spécifique au problème...*"

R. Niedermeier

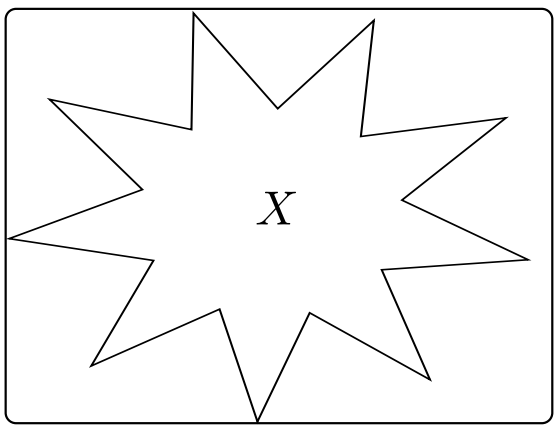
Complexité paramétrée : Exemple du Vertex Cover

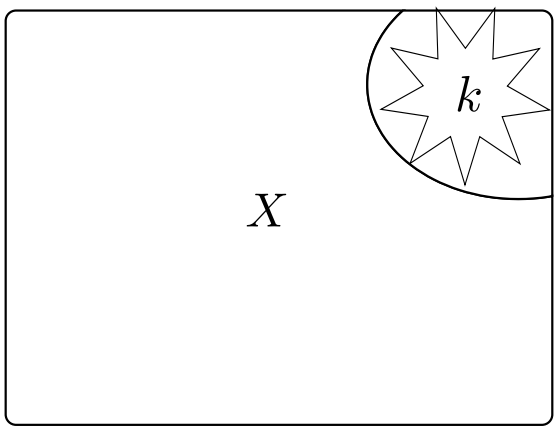
Complexité paramétrée : Exemple du Vertex Cover



Complexité paramétrée

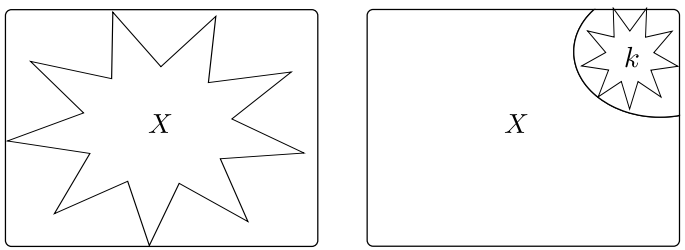
- Problème dans la classe FPT si algorithme **exact** avec complexité pire cas de la forme $f(k) \cdot |X|^c$.
  - Par ex. $O(2^k \cdot n^2)$ ou $O(2^{2^{2^{2^{2^k}}}} \cdot n)$.





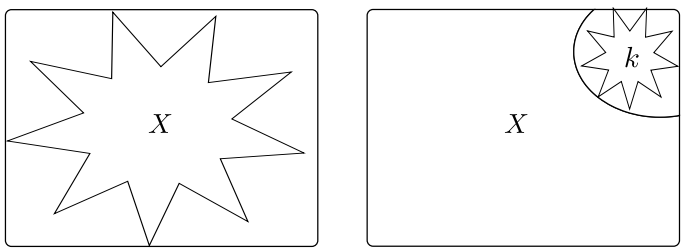
Complexité paramétrée - En pratique ?

- Souvent des paramètres naturels (petits) dans des applications.
  - Taille k de la requête dans une **base de données** de taille n.
  - k étapes dans un **jeu** à n mouvements possibles.
  - Aligner k **séquences d'ADN** de taille n.
  - Nombre k de sommets intéressants dans un **réseau social** de taille n.
  - Nombre n de **votants** parmi k candidats.
  - IA. **Strip Planning** : taille k du plan.
  - ...



Complexité paramétrée - En pratique ?

- Souvent des paramètres naturels (petits) dans des applications.
  - Taille k de la requête dans une **base de données** de taille n.
  - k étapes dans un **jeu** à n mouvements possibles.
  - Aligner k **séquences d'ADN** de taille n.
  - Nombre k de sommets intéressants dans un **réseau social** de taille n.
  - Nombre n de **votants** parmi k candidats.
  - IA. **Strip Planning** : taille k du plan.
  - ...



- $k \simeq 10000$ pour le problème Cluster Editing en quelques minutes.

Complexité paramétrée - Dark side

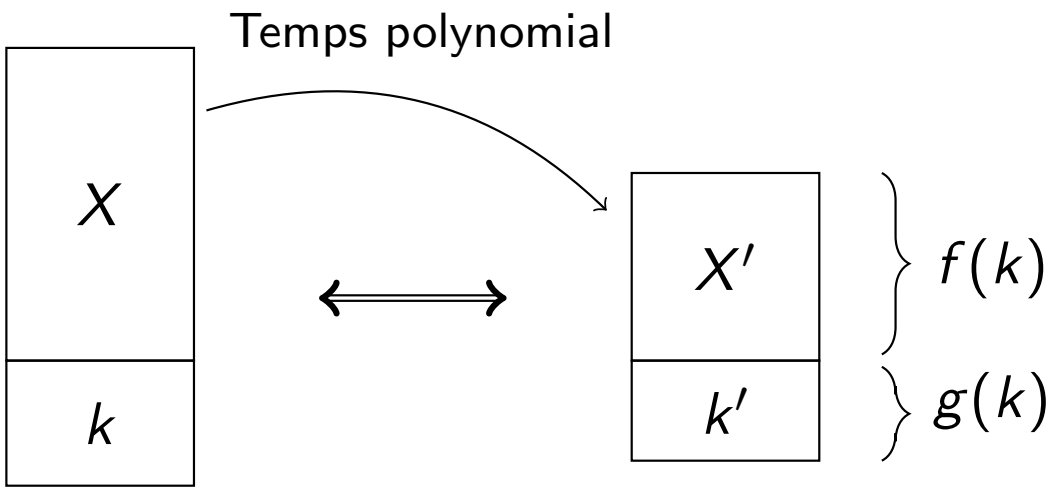
- Classes de complexités...
- **fpt-réductions** préservant le paramètre.
  - Transformer une instance (G, k) d'Independent Set en une instance $(G, n - k)$ de Vertex Cover **n'est pas** une fpt-réduction.
  - Transformer une instance (G, k) d'Independent Set en une instance $(\overline{G}, k)$ de Clique **est** une fpt-réduction.

Complexité paramétrée - Dark side

- Classes de complexités...
- **fpt-réductions** préservant le paramètre.
  - Transformer une instance (G, k) d'Independent Set en une instance $(G, n - k)$ de Vertex Cover **n'est pas** une fpt-réduction.
  - Transformer une instance (G, k) d'Independent Set en une instance $(\overline{G}, k)$ de Clique **est** une fpt-réduction.
- Existe une analogie avec le théorème de Cook-Levin pour "démarrer".
- Par ex., Independent Set pas dans FPT pour le paramètre taille de la solution.

Noyaux

- "Pre-processing" de l'instance.
- Obtenir en temps polynomial une **instance équivalente** de **taille bornée** par une fonction du paramètre.
- But : noyau de la plus petite taille possible (polynomial vs non-polynomial)



Noyaux - Max 3-Sat

Max 3-Sat

- Entrée : n variables, m clauses.
- Paramètre : un entier k.
- Question : Existe-t-il une affectation des variables pour satisfaire au moins k clauses ?

Noyaux - Max 3-Sat

Max 3-Sat

- Entrée : n variables, m clauses.
- Paramètre : un entier k.
- Question : Existe-t-il une affectation des variables pour satisfaire au moins k clauses ?

- Il existe une affectation pour satisfaire la **moitié des clauses**.
  - Prendre n'importe quelle affectation :
    - Si elle satisfait la moitié des clauses, OK.
    - Sinon, son complémentaire satisfait la moitié des clauses, OK.
- Si $k \leq m/2$: réduire à une instance OUI triviale (1 clause, $k = 1$).
- Sinon, $m \leq 2k$, donc $n \leq 6k$: instance bornée par une fonction (linéaire) de k.

Noyaux - Max 3-Sat

Max 3-Sat

- Entrée : n variables, m clauses.
- Paramètre : un entier k.
- Question : Existe-t-il une affectation des variables pour satisfaire au moins k clauses ?

- Il existe une affectation pour satisfaire la **moitié des clauses**.
  - Prendre n'importe quelle affectation :
    - Si elle satisfait la moitié des clauses, OK.
    - Sinon, son complémentaire satisfait la moitié des clauses, OK.
- Si $k \leq m/2$: réduire à une instance OUI triviale (1 clause, $k = 1$).
- Sinon, $m \leq 2k$, donc $n \leq 6k$: instance bornée par une fonction (linéaire) de k.

- Un noyau implique l'appartenance à la classe FPT pour ce paramètre (brute-force).

Algorithmes modérements exponentiels

- But : avoir un algorithme exact en $O(c^n)$, petit c !
  - Par ex., $O(1.8^n)$ à $O(1.7^n)$: facteur **multiplicatif** sur la taille des instances possibles.

Algorithmes modérements exponentiels

- Maximum Independent Set naïf :
  - Tester **tous les sous-ensembles** de sommets et renvoyer le plus grand.
  - $\Omega(2^n)$.

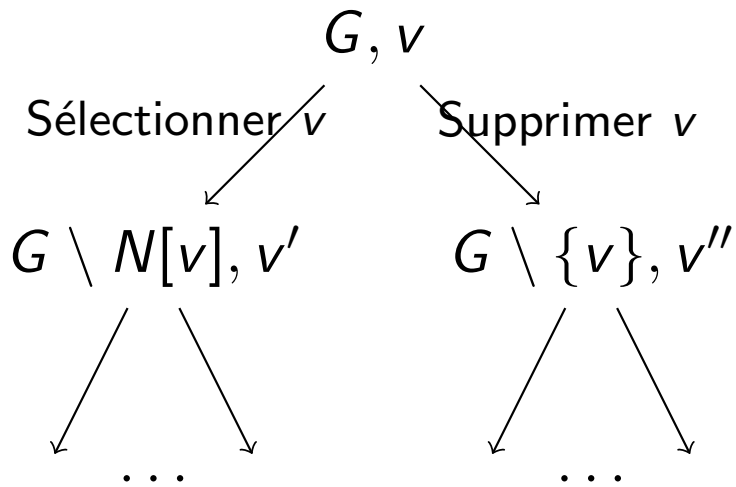
Branchements pour Maximum Independent Set

- Brancher sur deux sous-problèmes et résoudre récursivement.
- Pour chaque v, soit :
  - Sélectionner v (et supprimer son voisinage).
  - Supprimer v.

Branchements pour Maximum Independent Set

- Brancher sur deux sous-problèmes et résoudre récursivement.
- Pour chaque v, soit :
  - Sélectionner v (et supprimer son voisinage).
  - Supprimer v.

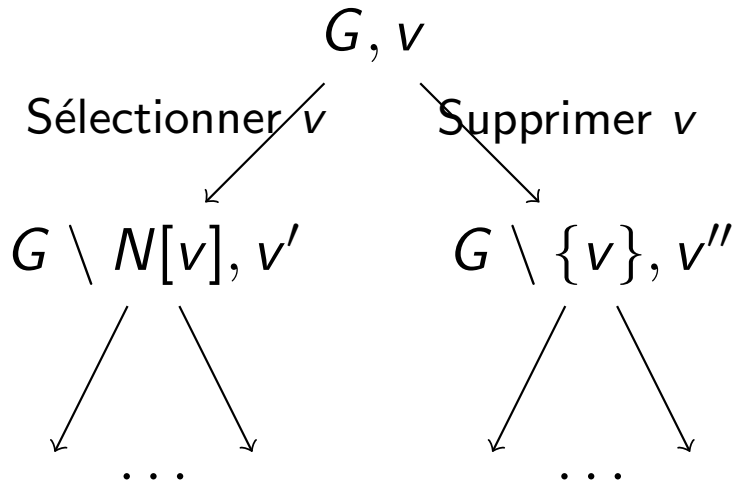
- Algo *MIS* récursif :
  - Si $deg(G) \leq 2$: polynomial.
  - Sinon,
    - Choisir un v.
    - Retourner $\max(1 + MIS(G \setminus N[v]), MIS(G \setminus \{v\})$.



Branchements pour Maximum Independent Set

- Brancher sur deux sous-problèmes et résoudre récursivement.
- Pour chaque v, soit :
  - Sélectionner v (et supprimer son voisinage).
  - Supprimer v.

- Algo *MIS* récursif :
  - Si $deg(G) \leq 2$: polynomial.
  - Sinon,
    - Choisir un v.
    - Retourner $\max(1 + MIS(G \setminus N[v]), MIS(G \setminus \{v\})$.



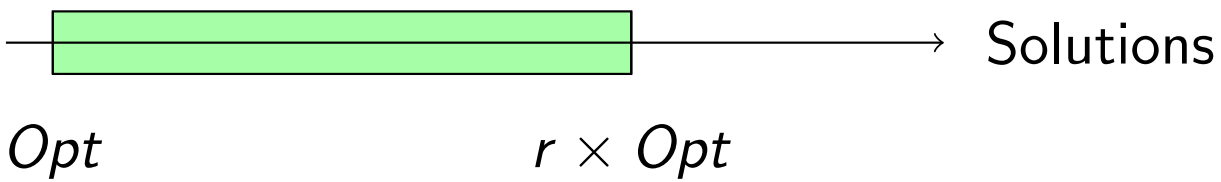
- $T(n) \leq T(n-1) + T(n-d(v)-1) \leq T(n-1) + T(n-4)$.
- Solution pour $x^n = x^{n-1} + x^{n-4}$.
- $O(1.3803^n \cdot poly(n)) = O^*(1.3803^n)$

Approximation

- Pour un problème d'optimisation NP-difficile, pas d'algorithme **polynomial** possible (sauf si P = NP)...
- ... si la solution doit être optimale !
- Sacrifier de l'exactitude pour obtenir un algorithme polynomial.

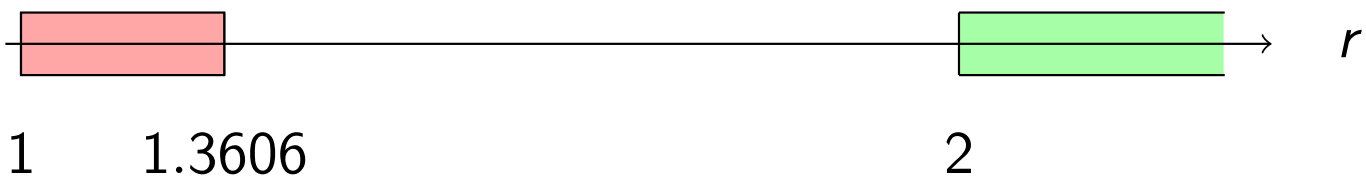
Approximation (problème de minimisation)

- On borne l'erreur.
- Un algorithme est de r-**approximation** s'il est polynomial et qu'il retourne des solutions dont le coût est au pire $r \times Opt$.



Approximation – Difficulté

- Possible de prouver qu'il n'existe pas d'algorithme d'approximation inférieur à un certain ratio.
- But : rapprocher les deux frontières.



Exemple pour Vertex Cover

Approximation en temps exponentiel

- Problème Max Independent Set.
- On sait le résoudre en $O^*(c^n)$.
- Propriété héréditaire
  - Vérifie la propriété sur G ssi tous ses sous-graphes la vérifient.

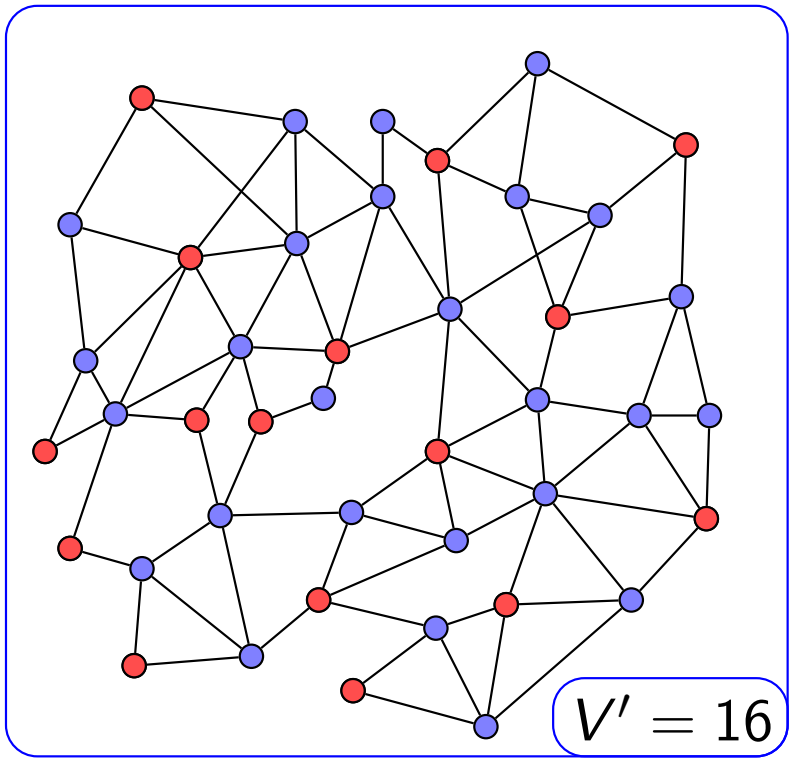


Figure E. Tourniaire

Approximation en temps exponentiel

- Problème Max Independent Set.
- On sait le résoudre en $O^*(c^n)$.
- Propriété héréditaire
  - Vérifie la propriété sur G ssi tous ses sous-graphes la vérifient.
- Partager l'instance en 2 parts égales.
- Résoudre sur chaque partie.
- Renvoyer le plus grand tel quel.
- Ratio $1/2$: $O^*(c^{n/2})$.

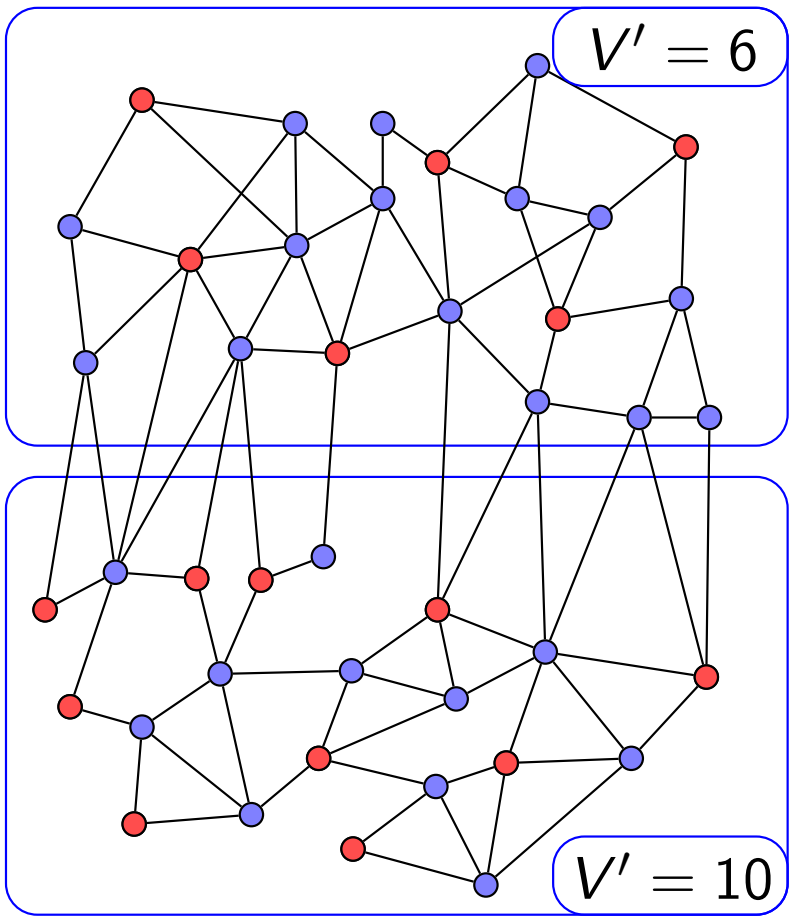


Figure E. Tourniaire

Approximation en temps exponentiel

- Problème Max Independent Set.
- On sait le résoudre en $O^*(c^n)$.
- Propriété héréditaire
  - Vérifie la propriété sur G ssi tous ses sous-graphes la vérifient.
- Partager l'instance en 2 parts égales.
- Résoudre sur chaque partie.
- Renvoyer le plus grand tel quel.
- Ratio 1/2 : $O^*(c^{n/2})$.
- Pas de ratio meilleur que $n^{1-\epsilon}$ en temps polynomial.

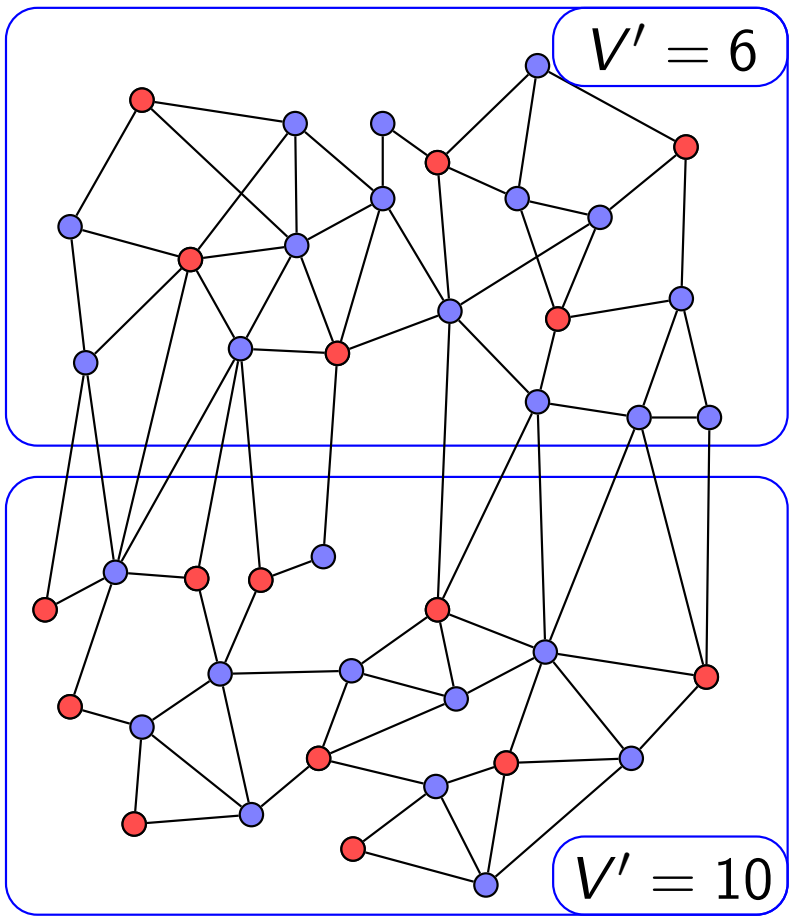
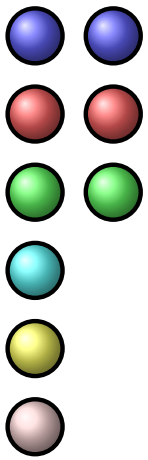


Figure E. Tourniaire

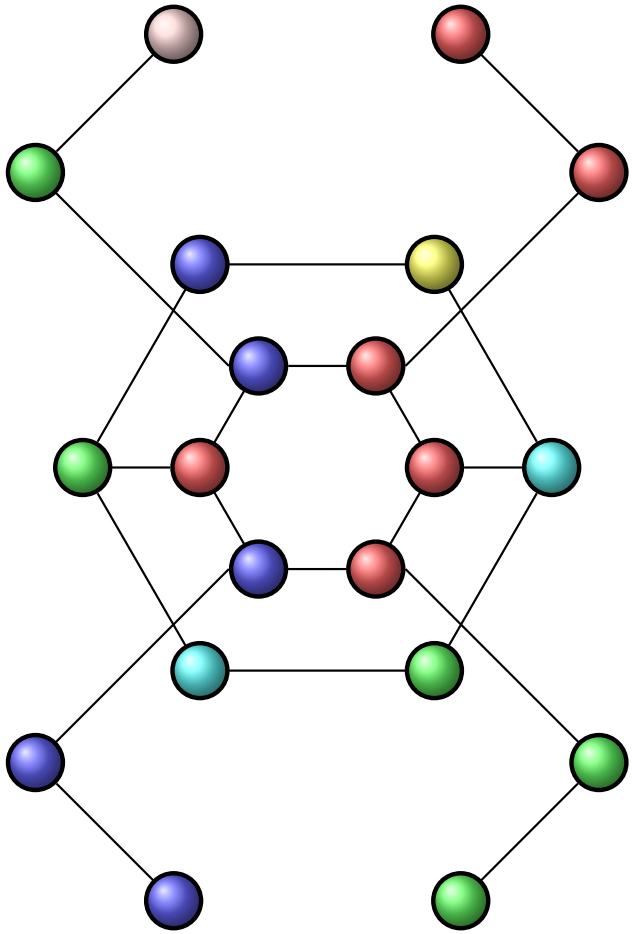
Plan

Graph Motif

M :



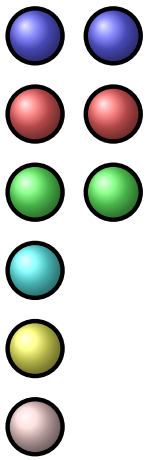
G :



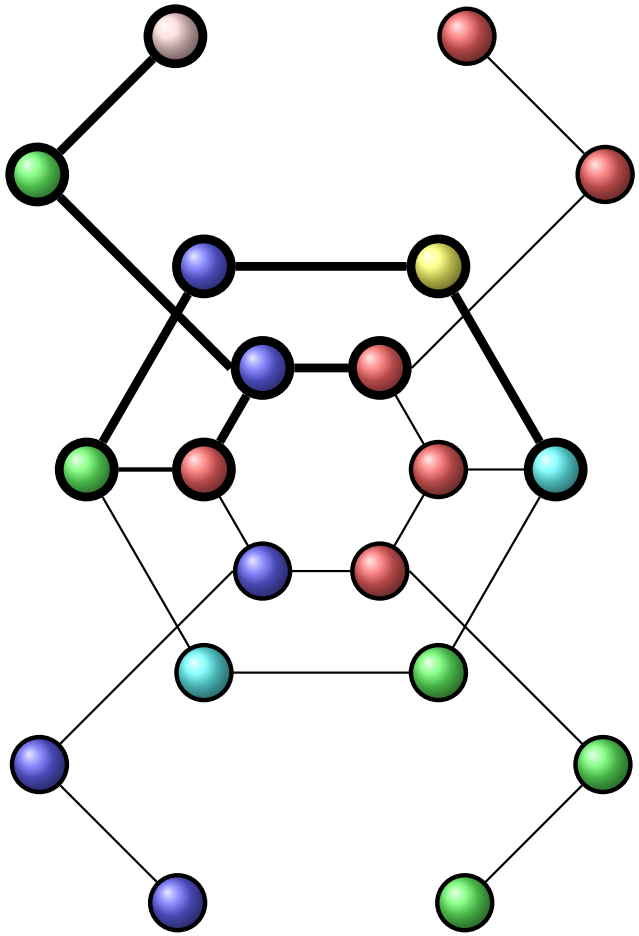
- Sortie : existe t-il un sous-graphe connexe contenant exactement toutes les couleurs de M ?

Graph Motif

M :

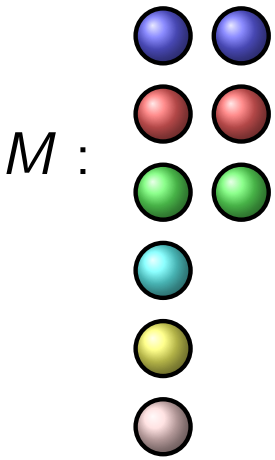


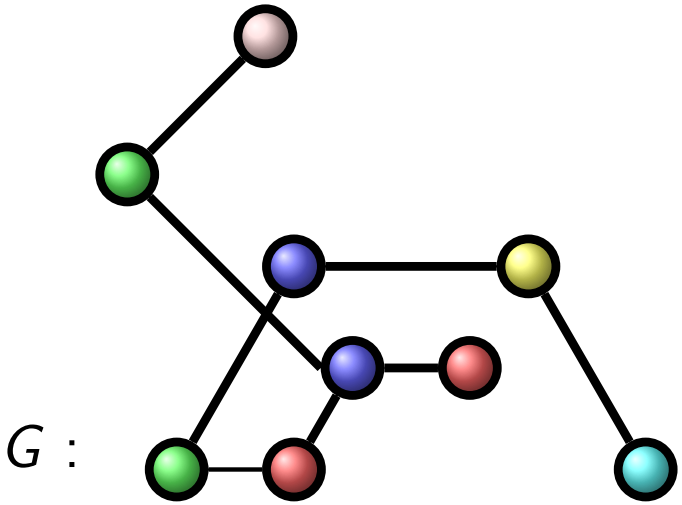
G :



- Sortie : existe t-il un sous-graphe connexe contenant exactement toutes les couleurs de M ?

Graph Motif





- Sortie : existe t-il un sous-graphe connexe contenant exactement toutes les couleurs de M ?

Graph Motif – Difficulté

- Le problème reste NP-complet, même si :
  - G est un arbre [Lacroix et al. 2006].
  - Cet **arbre est de degré maximum 3** et une occurrence par couleur dans M [Fellows et al. 2007].
  - L'**arbre est de profondeur 2** et une occurrence par couleur dans M [Ambalath et al. 2010].
  - **2 couleurs** différentes dans M et G est un graphe biparti de degré maximum 4 [Fellows et al. 2007].

Graph Motif – Difficulté

- Le problème reste NP-complet, même si :
  - G est un arbre [Lacroix et al. 2006].
  - Cet **arbre est de degré maximum 3** et une occurrence par couleur dans M [Fellows et al. 2007].
  - L'**arbre est de profondeur 2** et une occurrence par couleur dans M [Ambalath et al. 2010].
  - **2 couleurs** différentes dans M et G est un graphe biparti de degré maximum 4 [Fellows et al. 2007].
- Polynomial si :
  - G est un **caterpillar** (suppression des feuilles de l'arbre donne un chemin) [Ambalath et al. 2010].
    - Moins de n^2 sous-chemins possibles.
  - G est un arbre où chaque couleur **apparaît au plus deux fois**, une occurrence par couleur dans M [S. 2011].
    - 2 Sat

Graph Motif – Complexité paramétré

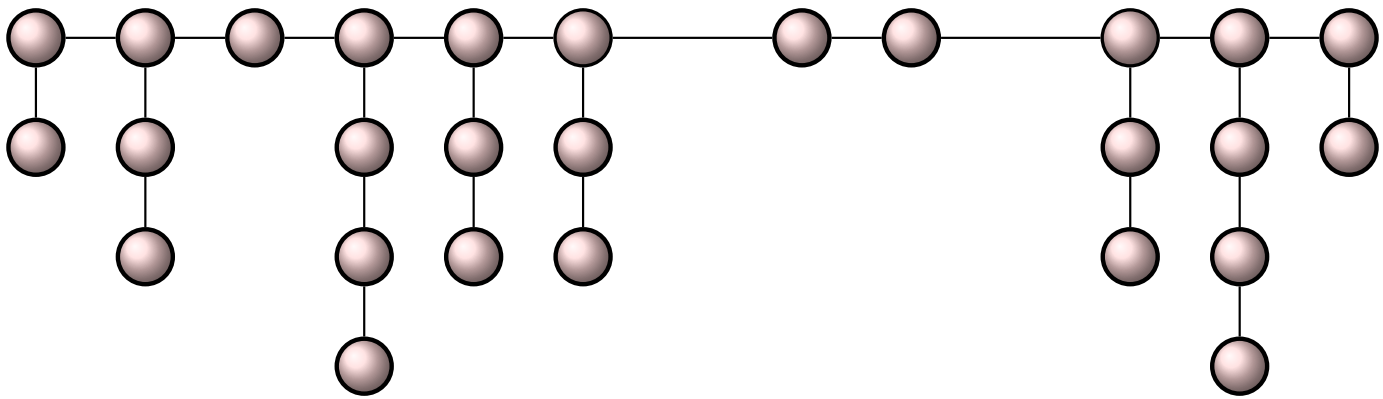
- Le problème est dans FPT avec le paramètre $k = |M|$.
  "Course" :
  - $O^*(87^k)$ [Fellows et al. 2007].
  - $O^*(4.3^k)$ (programmation dynamique et technique du color-coding) [Betzler et al. 2008].
  - $O^*(4^k)$ et espace polynomial (Recherche de monômes multilinéaires) [Guillemot et S. 2012].
  - $O^*(2.54^k)$ [Koutis 2012].
  - $O^*(2^k)$ et espace polynomial (Narrow sieves) [Bjorklund et al. 2013] [Pinter et al. 2013].
  - $O((2-\epsilon)^k)$ improbable [Bjorklund et al. 2013].

Graph Motif – Complexité paramétré

- Le problème est dans FPT avec le paramètre $k = |M|$.
  "Course" :
  - $O^*(87^k)$ [Fellows et al. 2007].
  - $O^*(4.3^k)$ (programmation dynamique et technique du color-coding) [Betzler et al. 2008].
  - $O^*(4^k)$ et espace polynomial (Recherche de monômes multilinéaires) [Guillemot et S. 2012].
  - $O^*(2.54^k)$ [Koutis 2012].
  - $O^*(2^k)$ et espace polynomial (Narrow sieves) [Bjorklund et al. 2013] [Pinter et al. 2013].
  - $O((2-\epsilon)^k)$ improbable [Bjorklund et al. 2013].
- Pas dans FPT pour le paramètre "nombre de couleurs différentes dans M" [Fellows et al. 2007].

Graph Motif – Noyaux

- Pas de noyau polynomial même si G est un comb graph [Ambalath et al. 2010].

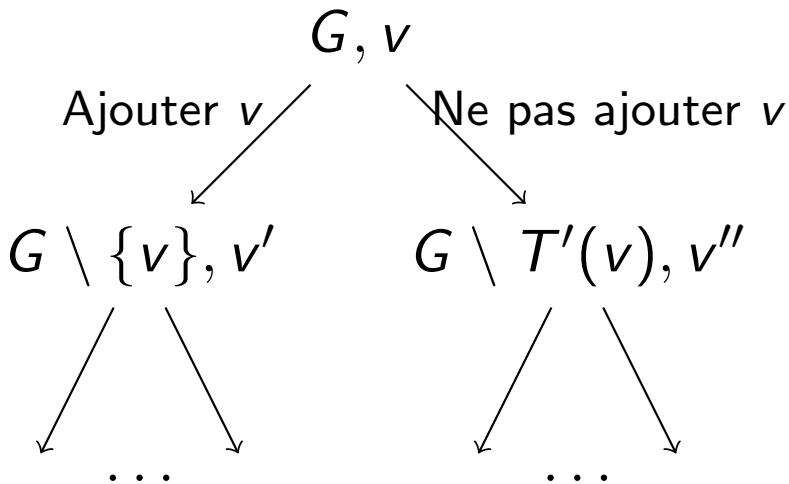


Graph Motif – Approximation

- Différentes versions d'optimisations associées.
- Par exemple, maximiser la taille de la solution.
  - Difficile à approcher.
    - APX-difficile, même pour G arbre de degré maximum 3 [Dondi et al. 2009].
    - Pas approximable à moins de $n^{1/3-\epsilon}$, même pour G arbre où chaque couleur apparait au plus deux fois [Rizzi et S. 2012].

Graph Motif – Algorithme modérement exponentiel

- Si G est un **arbre**, algorithme en $O^*(1.62^n)$ [Dondi et al. 2009].
- Suppose la racine dans la solution.
- Test de l'ajout de feuilles en temps polynomial.
- Branchement sur les sommets **internes**.
  1. L'ajouter dans la solution.
  2. Ne pas l'ajouter et supprimer son sous-arbre (donc au moins 2 sommets).

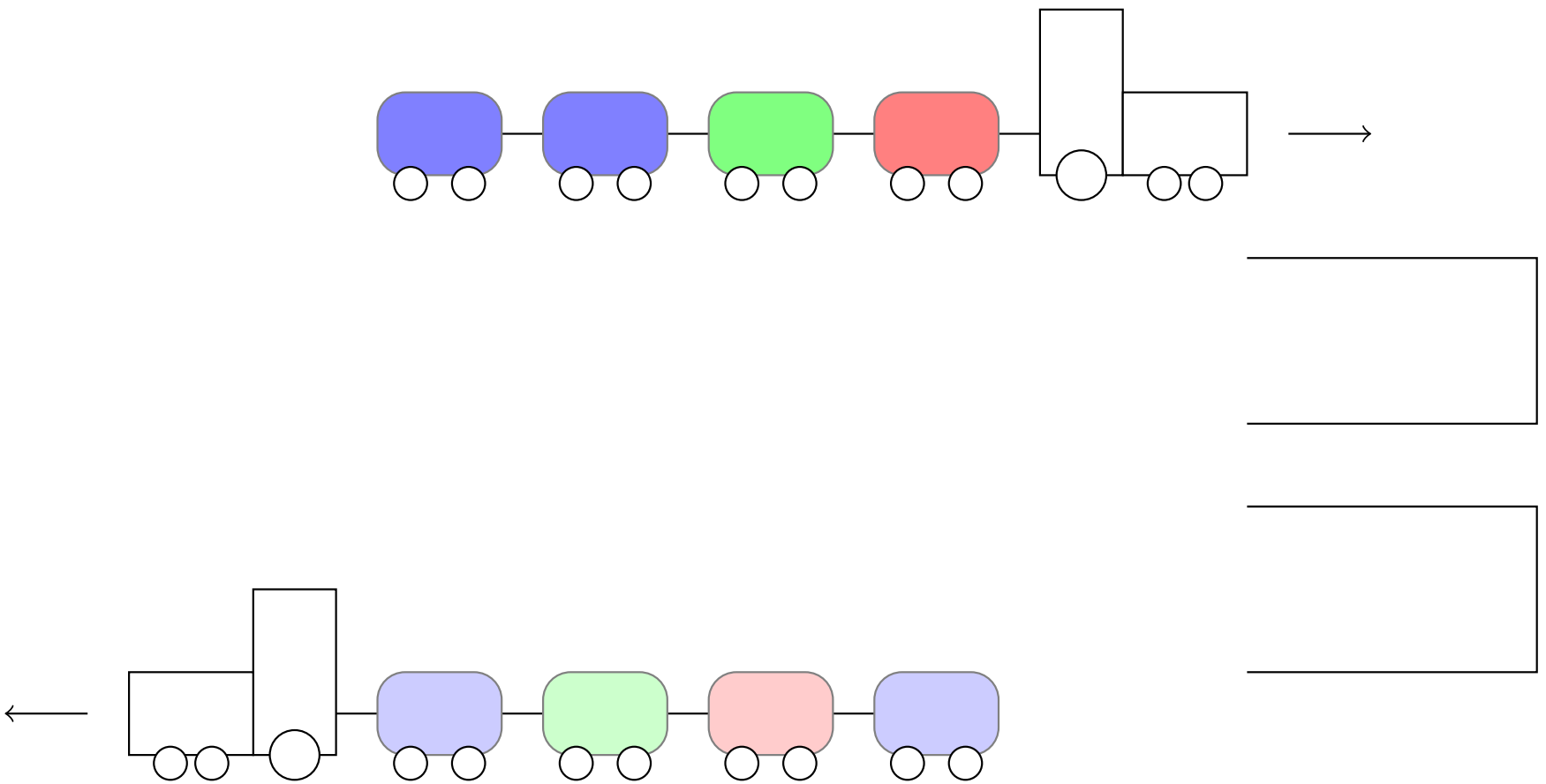


- $O^*(1.33^n)$ si chaque couleur n'apparait qu'au plus une fois dans M.

Plan

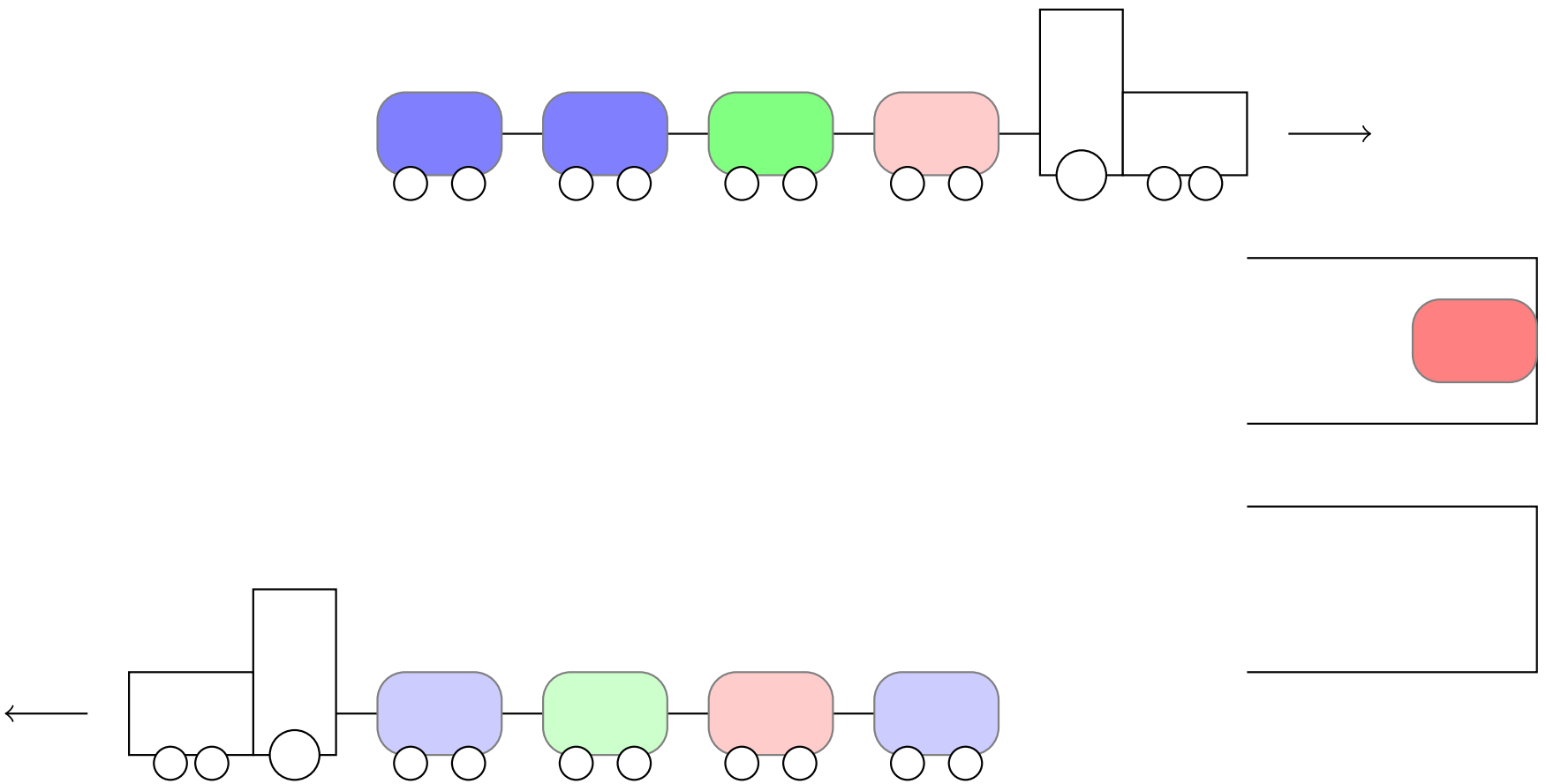
Trains [Woeginger et al. 2009]

- Train à stocker sur des tracks (lifo) de tailles bornées.
- Un train pour le lendemain.



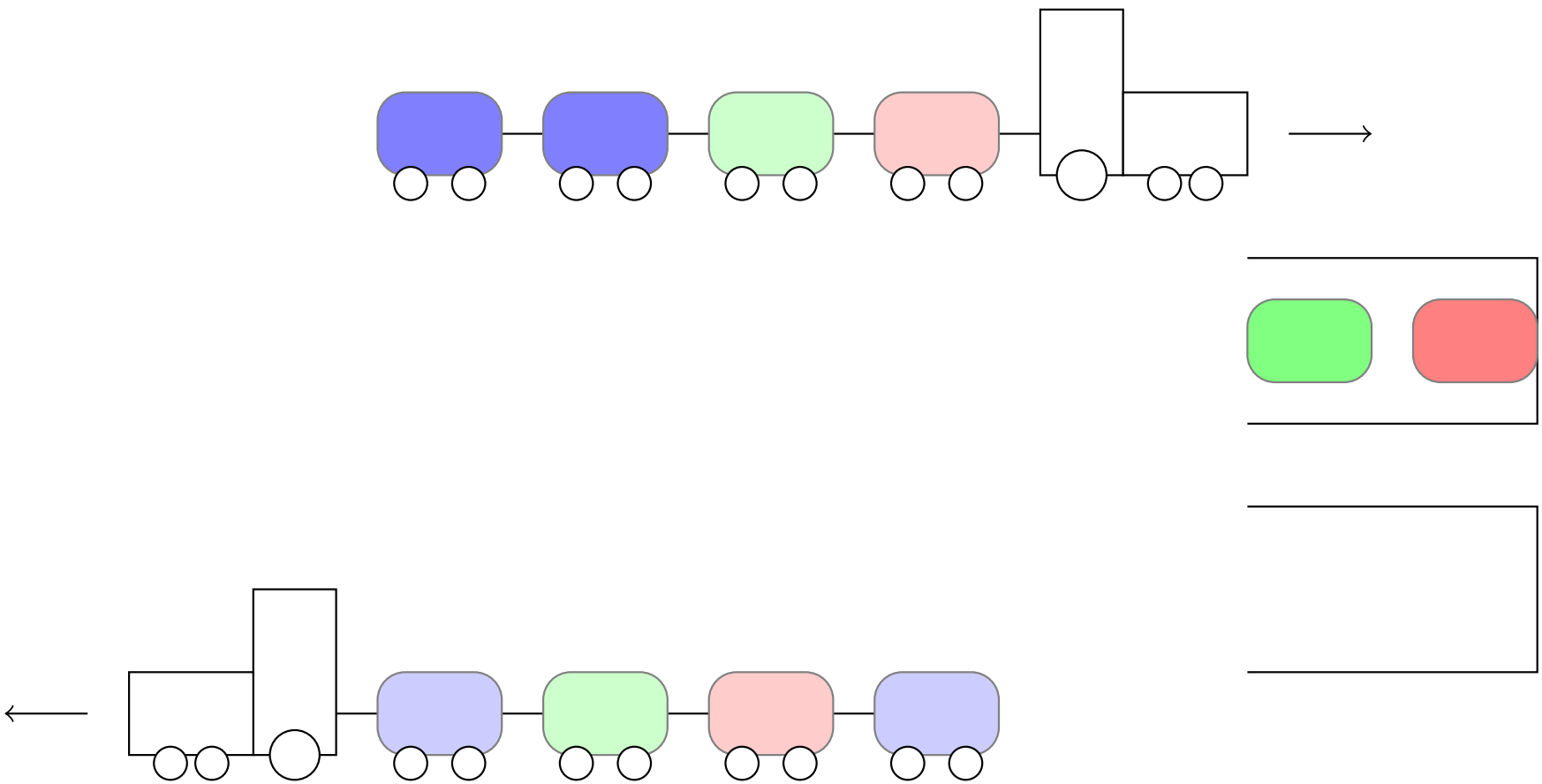
Trains [Woeginger et al. 2009]

- Train à stocker sur des tracks (lifo) de tailles bornées.
- Un train pour le lendemain.



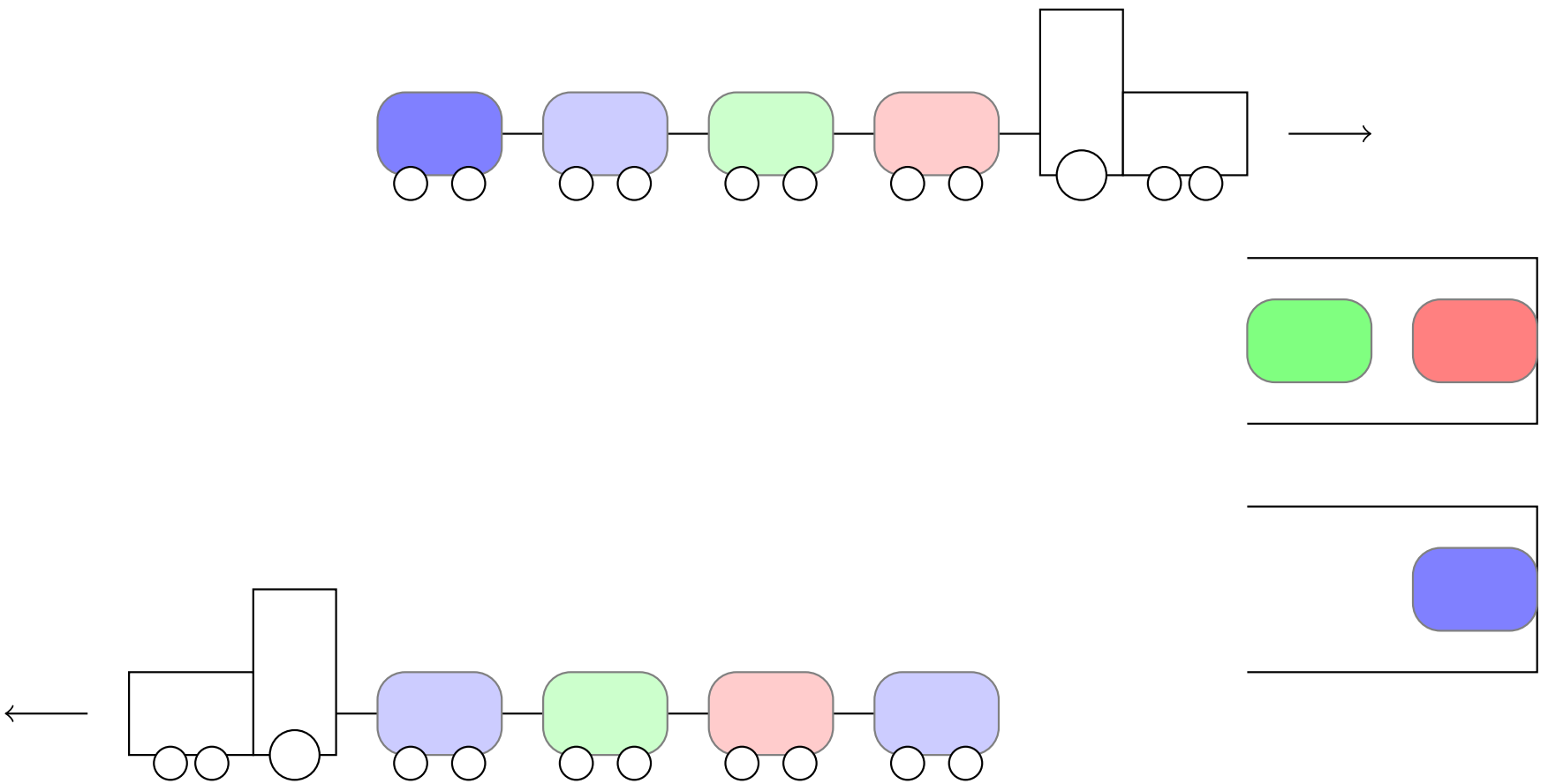
Trains [Woeginger et al. 2009]

- Train à stocker sur des tracks (lifo) de tailles bornées.
- Un train pour le lendemain.



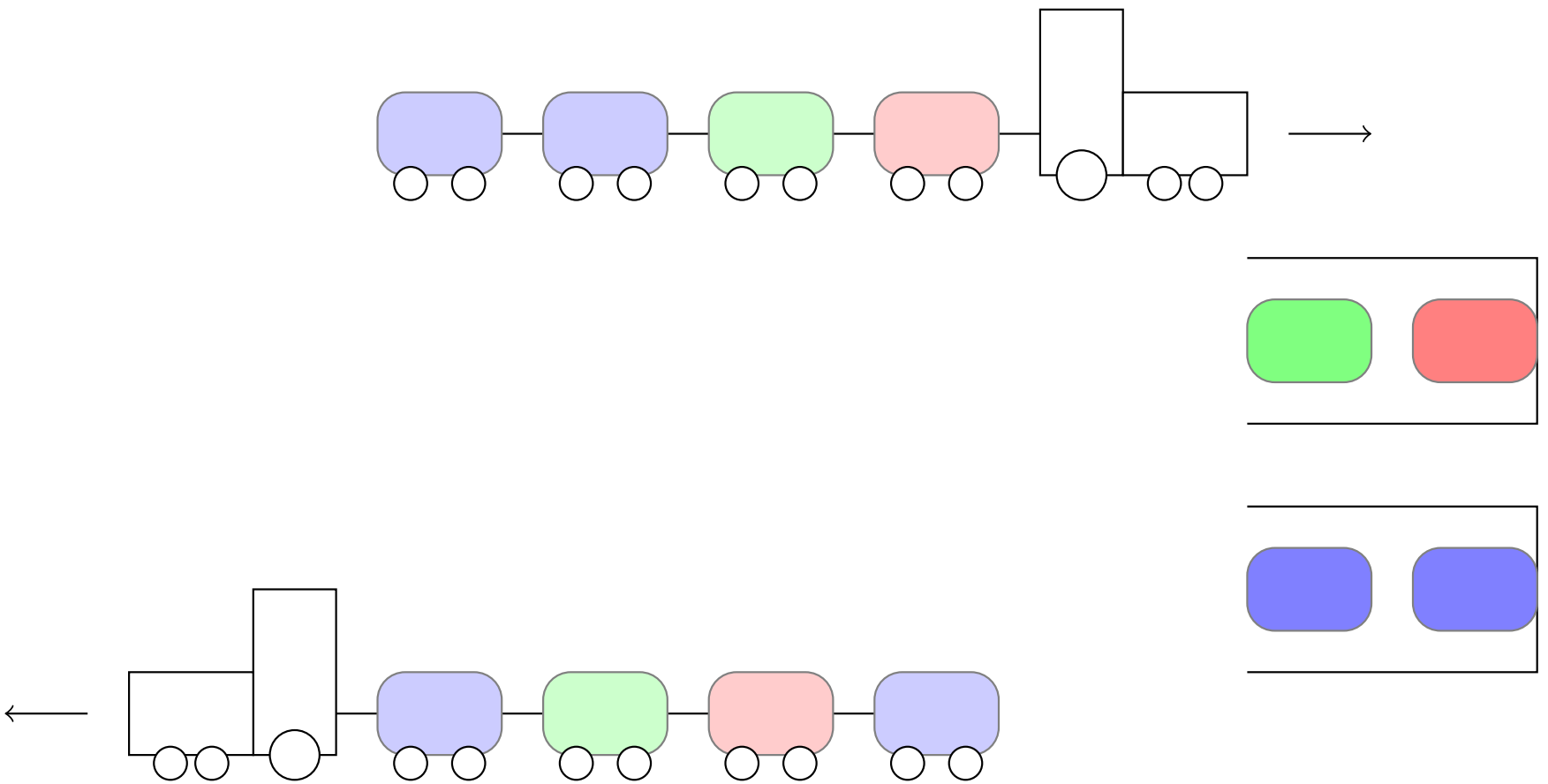
Trains [Woeginger et al. 2009]

- Train à stocker sur des tracks (lifo) de tailles bornées.
- Un train pour le lendemain.



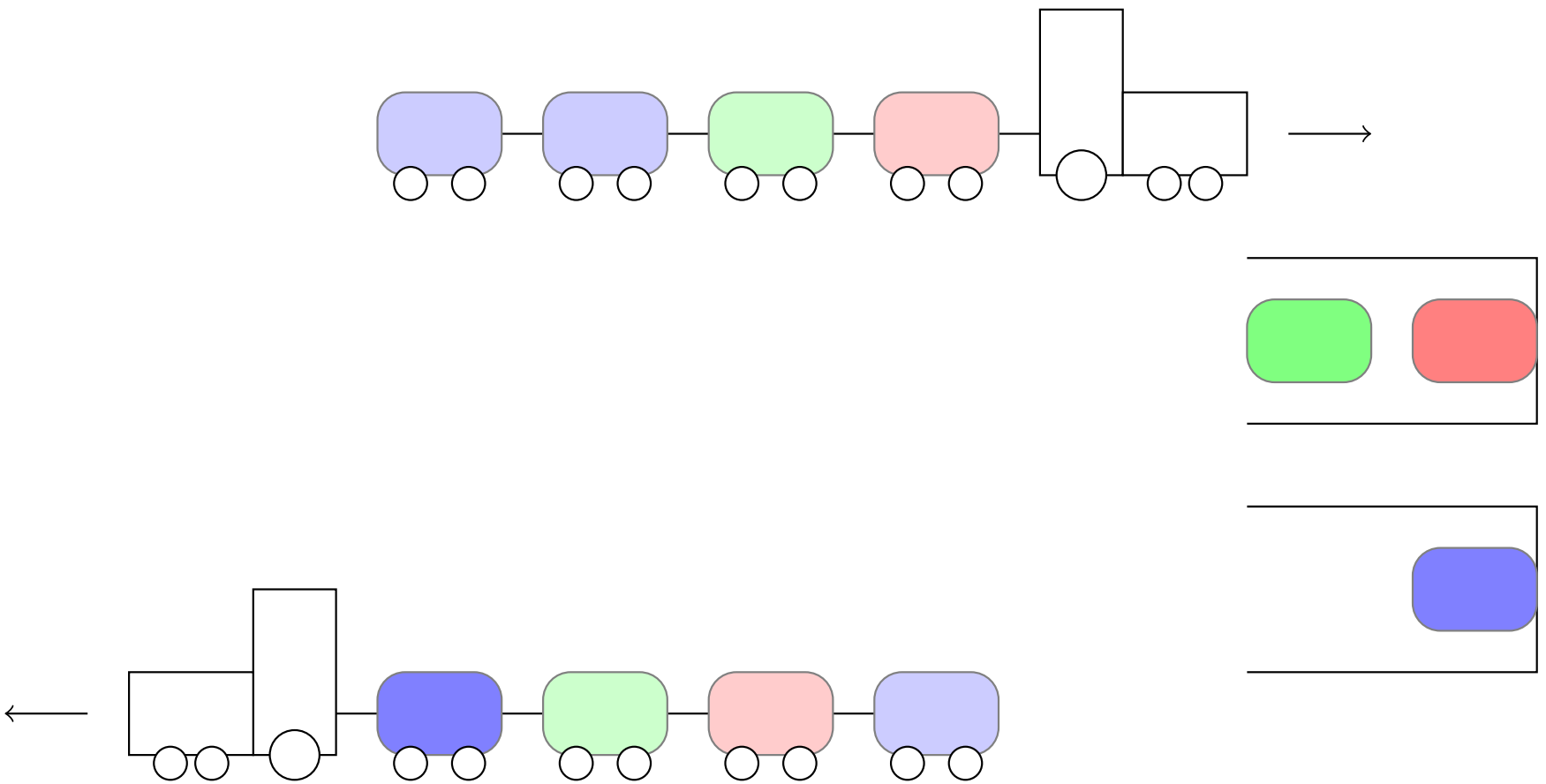
Trains [Woeginger et al. 2009]

- Train à stocker sur des tracks (lifo) de tailles bornées.
- Un train pour le lendemain.



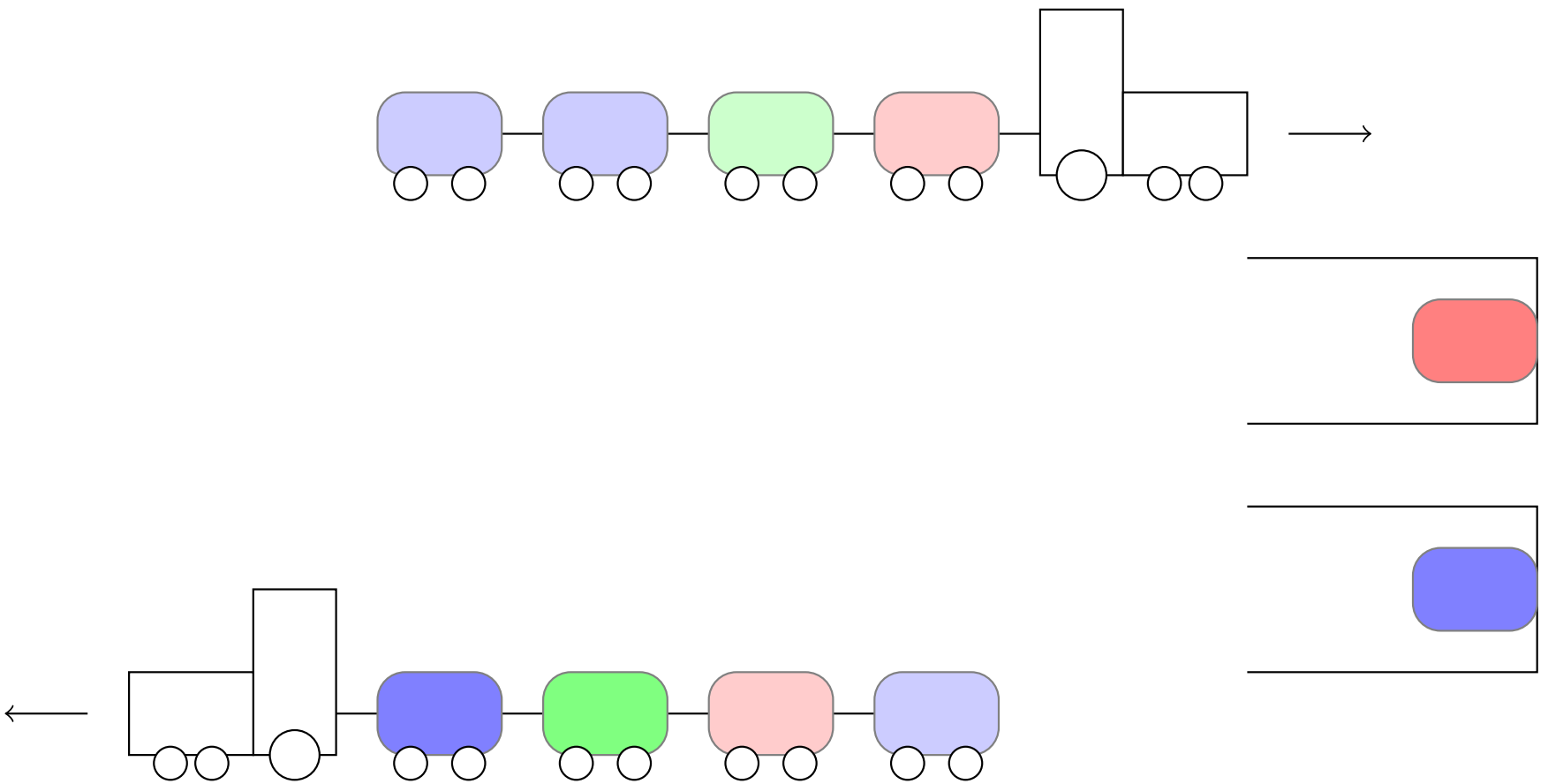
Trains [Woeginger et al. 2009]

- Train à stocker sur des tracks (lifo) de tailles bornées.
- Un train pour le lendemain.



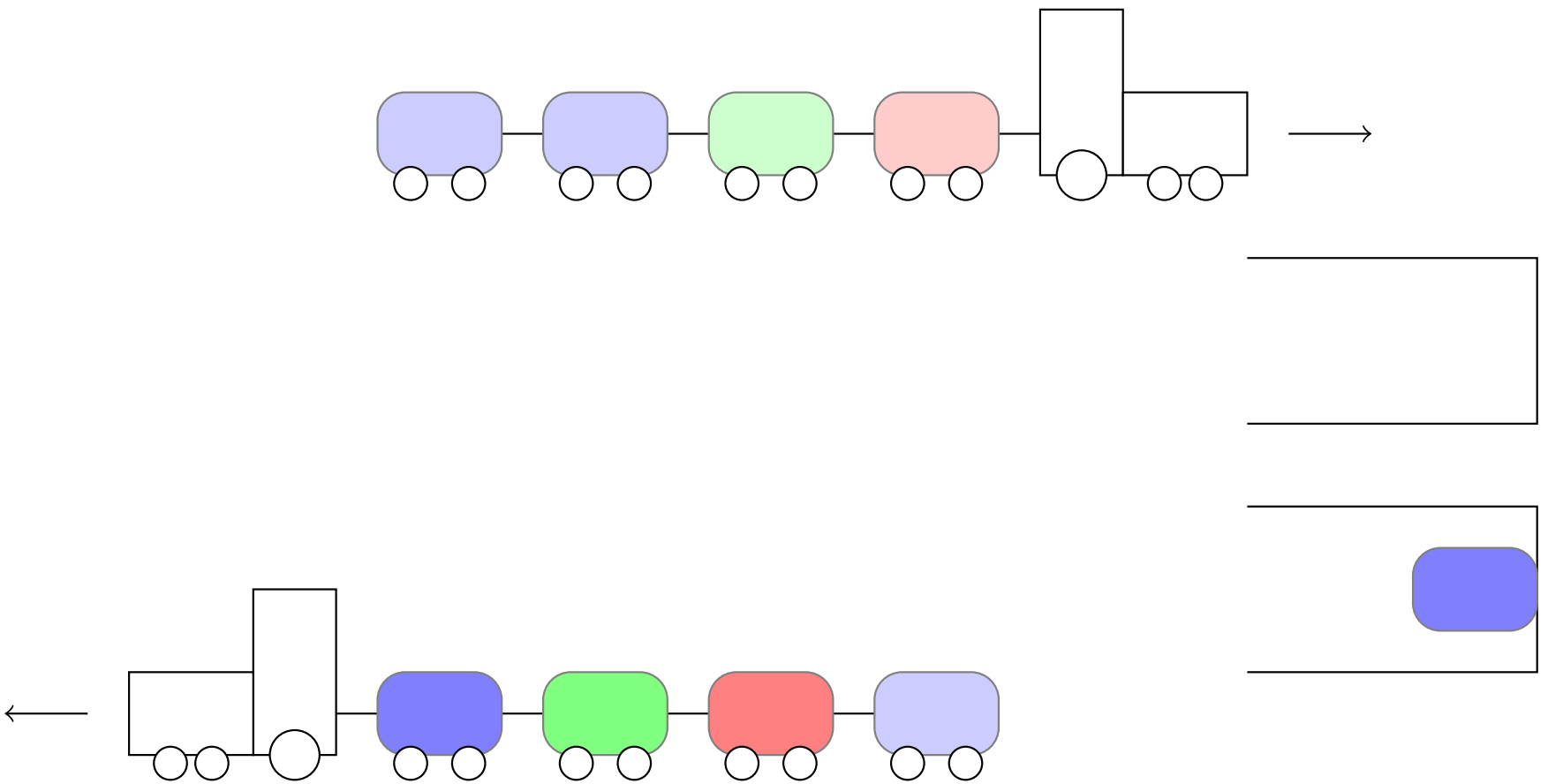
Trains [Woeginger et al. 2009]

- Train à stocker sur des tracks (lifo) de tailles bornées.
- Un train pour le lendemain.



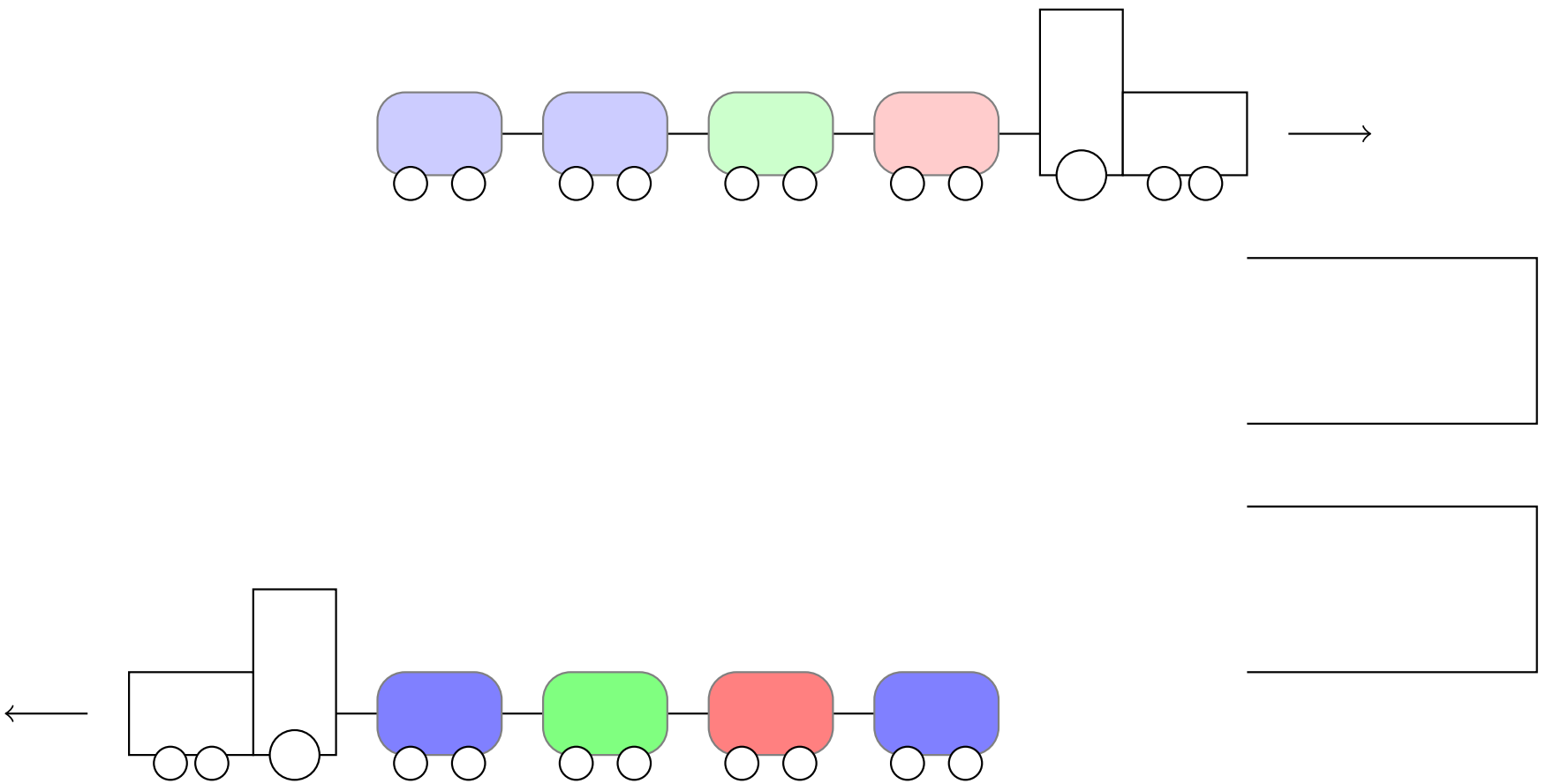
Trains [Woeginger et al. 2009]

- Train à stocker sur des tracks (lifo) de tailles bornées.
- Un train pour le lendemain.



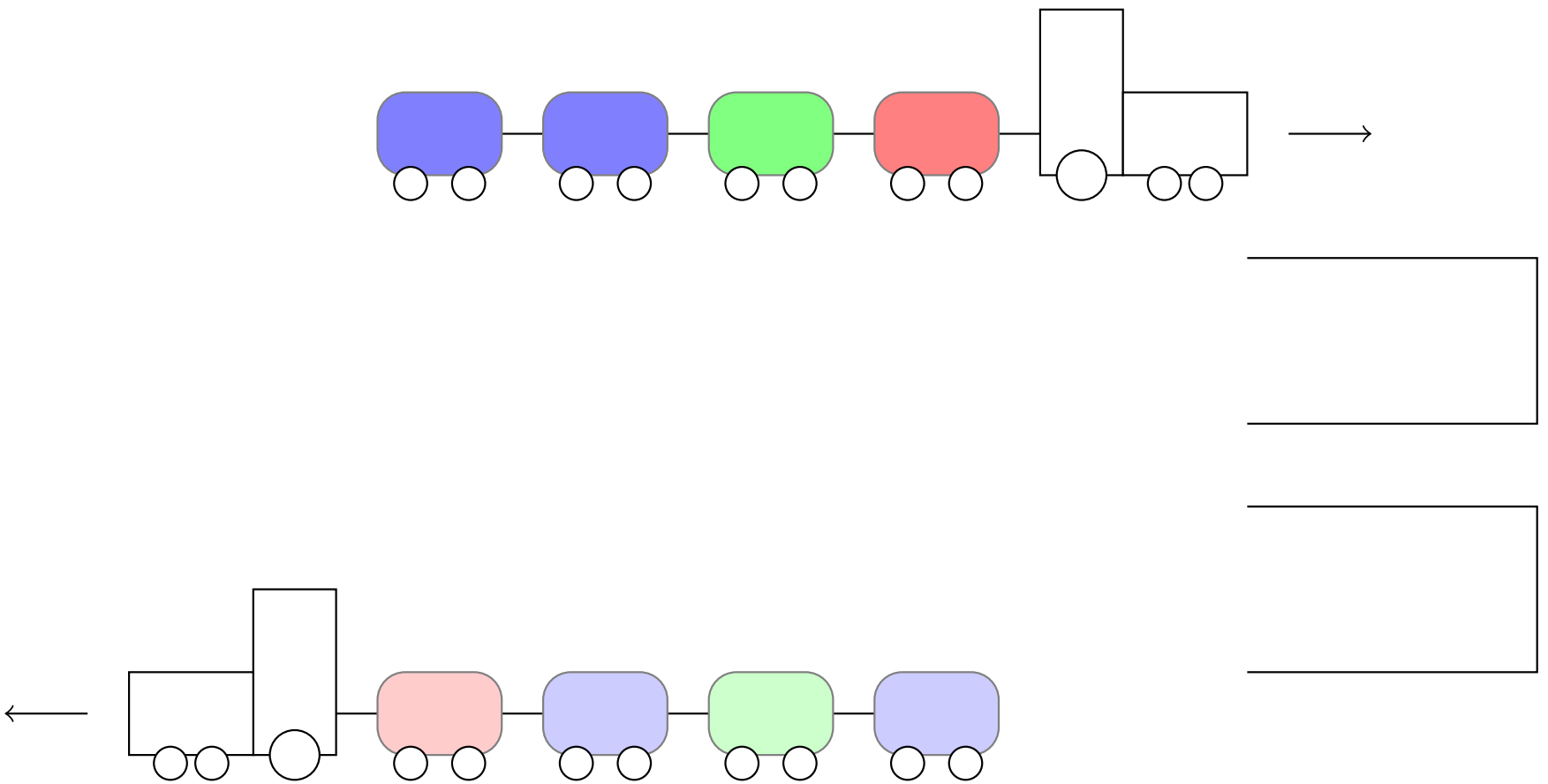
Trains [Woeginger et al. 2009]

- Train à stocker sur des tracks (lifo) de tailles bornées.
- Un train pour le lendemain.



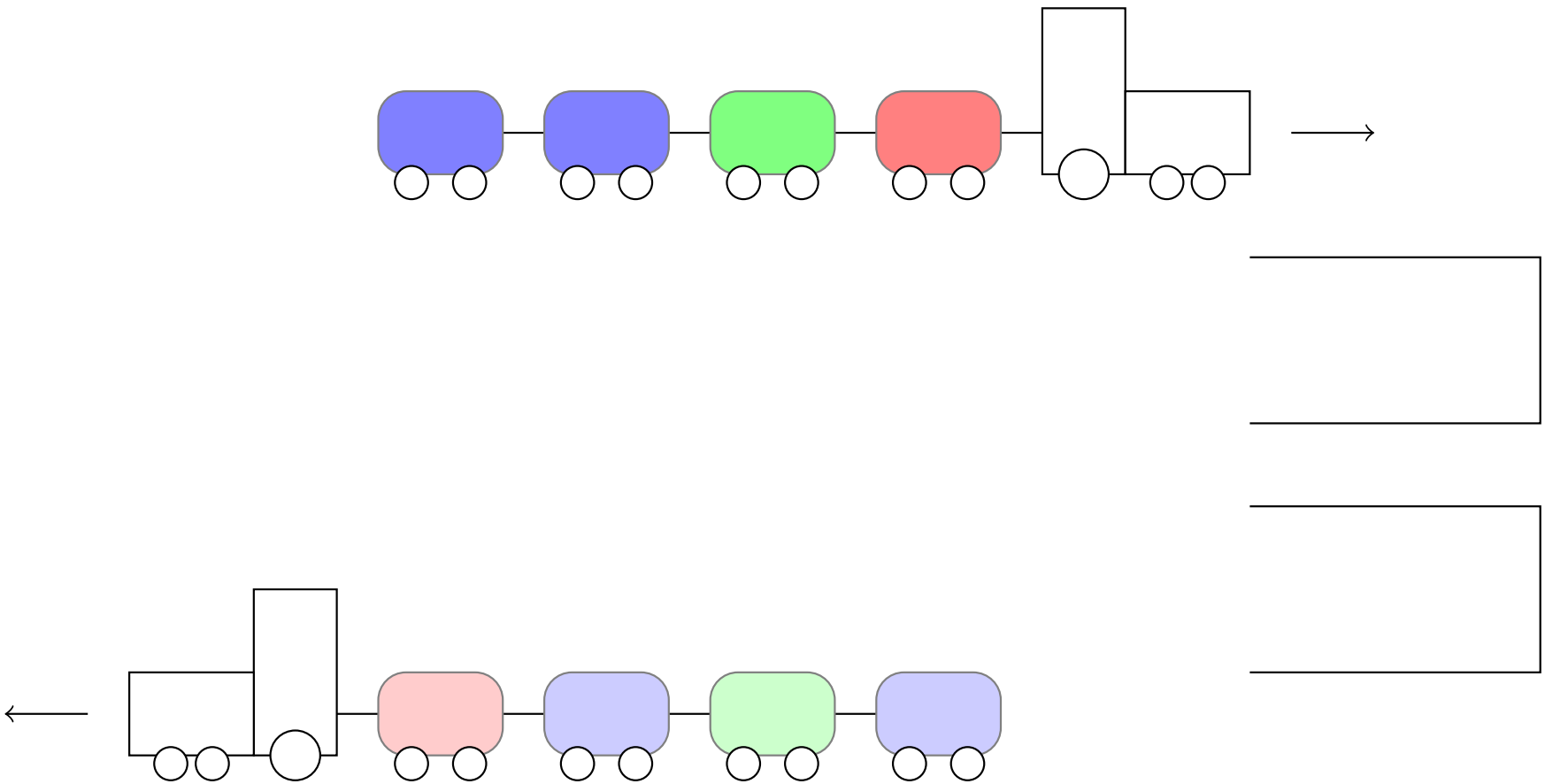
Trains [Woeginger et al. 2009]

- Train à stocker sur des tracks (lifo) de tailles bornées.
- Un train pour le lendemain.
- Parfois impossible.



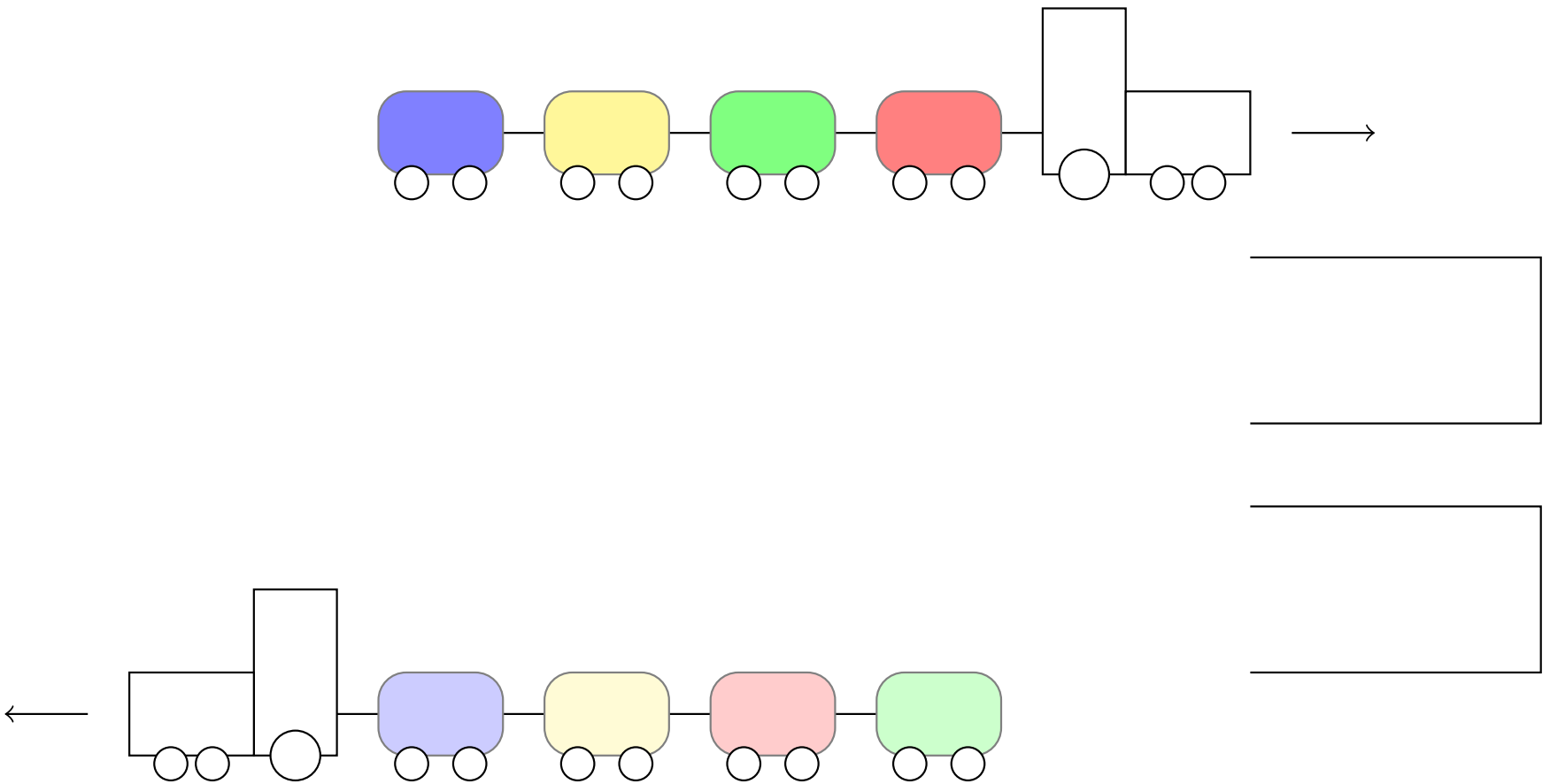
Trains [Woeginger et al. 2009]

- Tracks toutes de taille 3 : NP-complet.
  - Impossible d'être dans FPT pour le paramètre taille des tracks.
- Ouvert pour les tracks de taille 2.



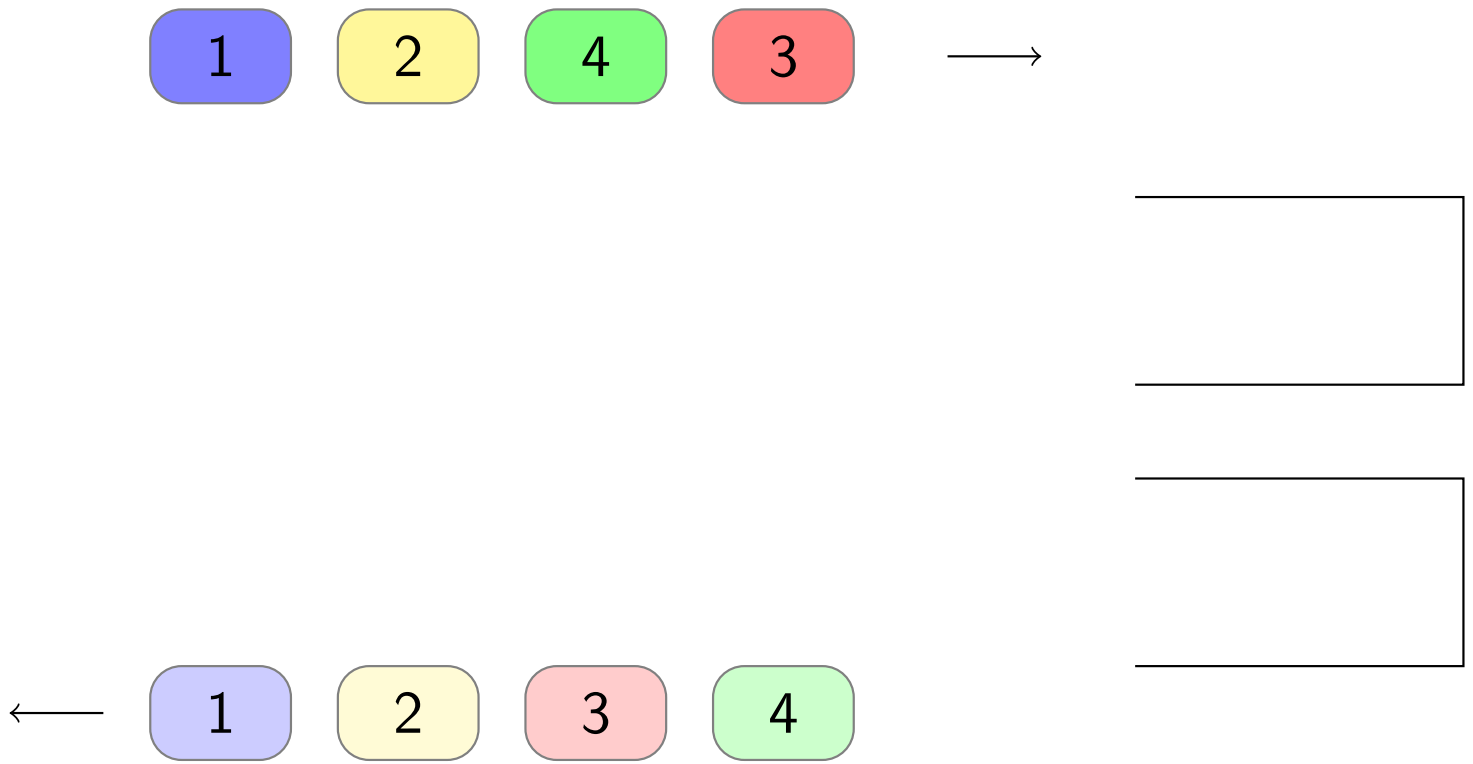
Trains - Permutations, tracks taille 2

- Au plus une occurrence par type.



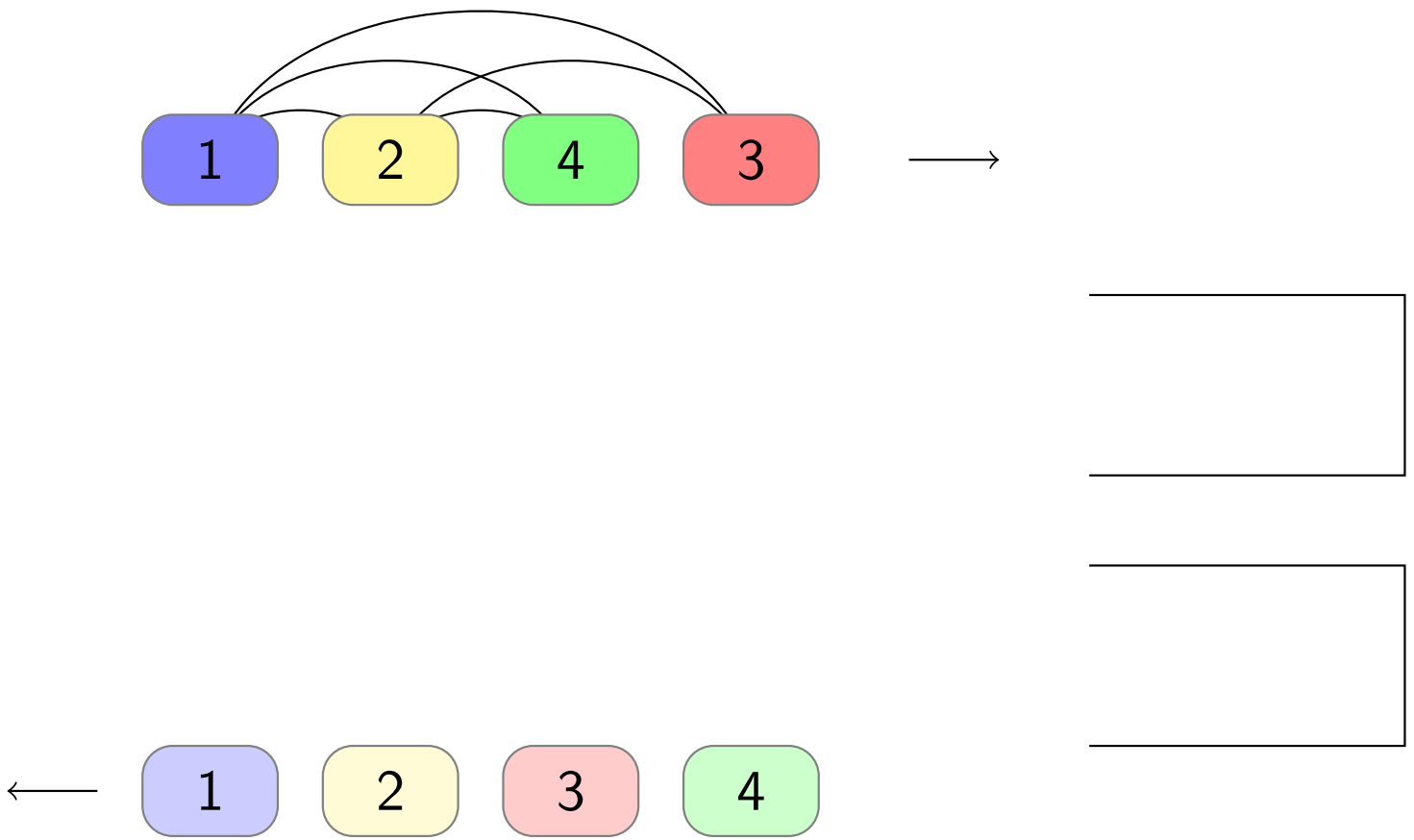
Trains - Permutations, tracks taille 2

- Permutations, wlog sortie identité.



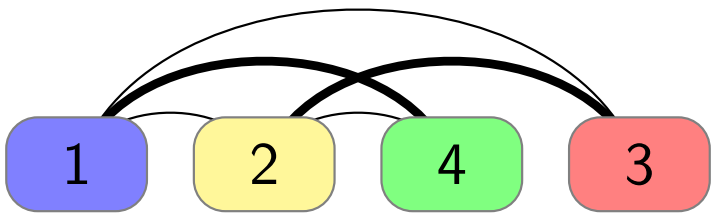
Trains - Permutations, tracks taille 2

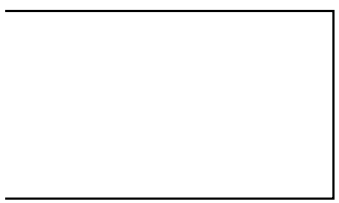
- Sommet : wagons.
- Arête : wagons pouvant être stockés sur la même track.
- Graphe de permutations.

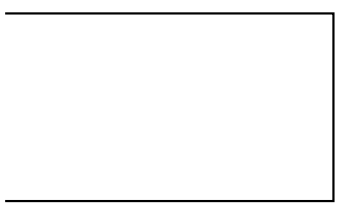


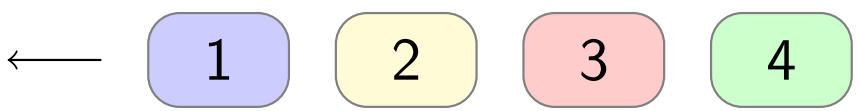
Trains - Permutations, tracks taille 2

- Sommet : wagons.
- Arête : wagons pouvant être stockés sur la même track.
- Graphe de permutations.
- Perfect matching.



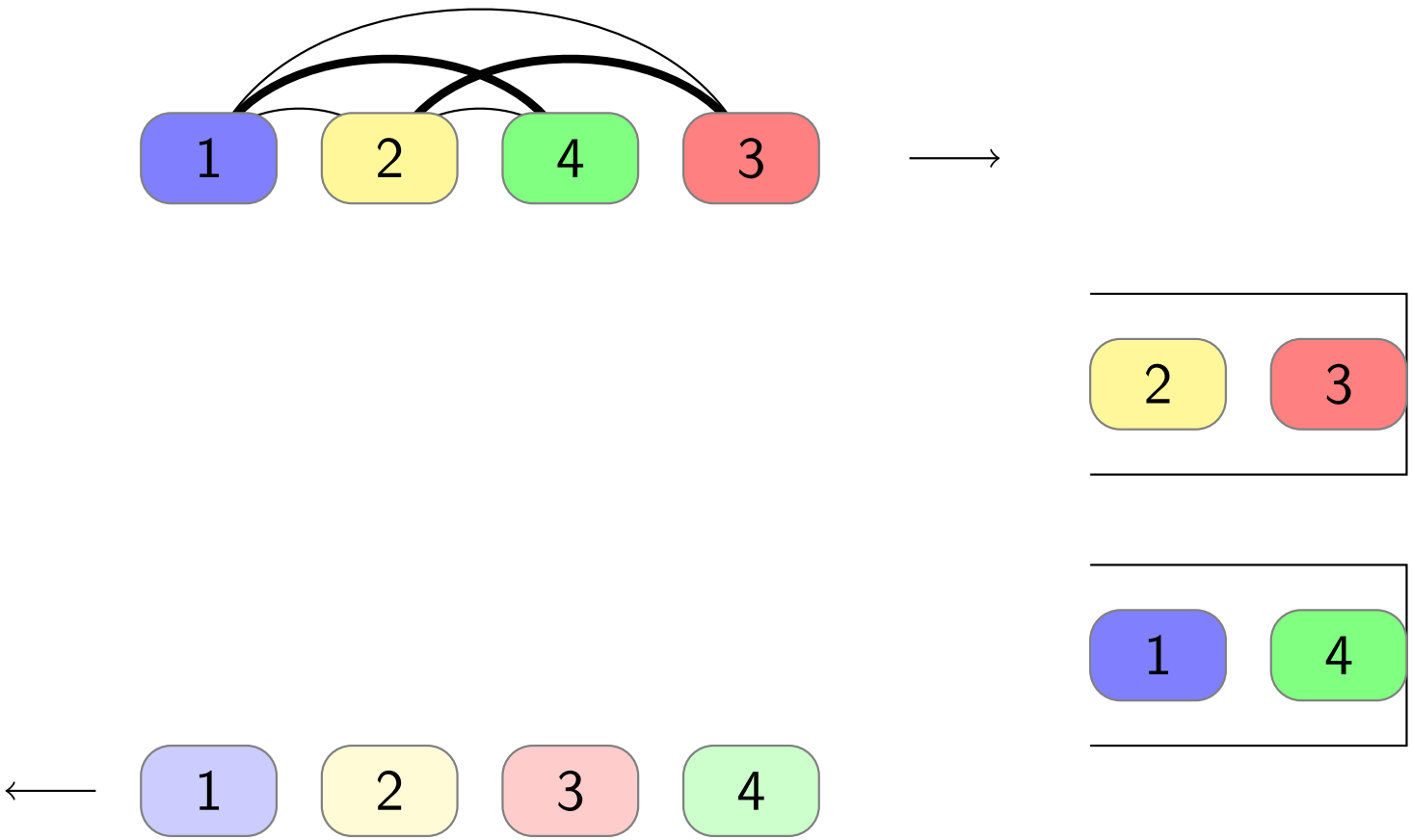






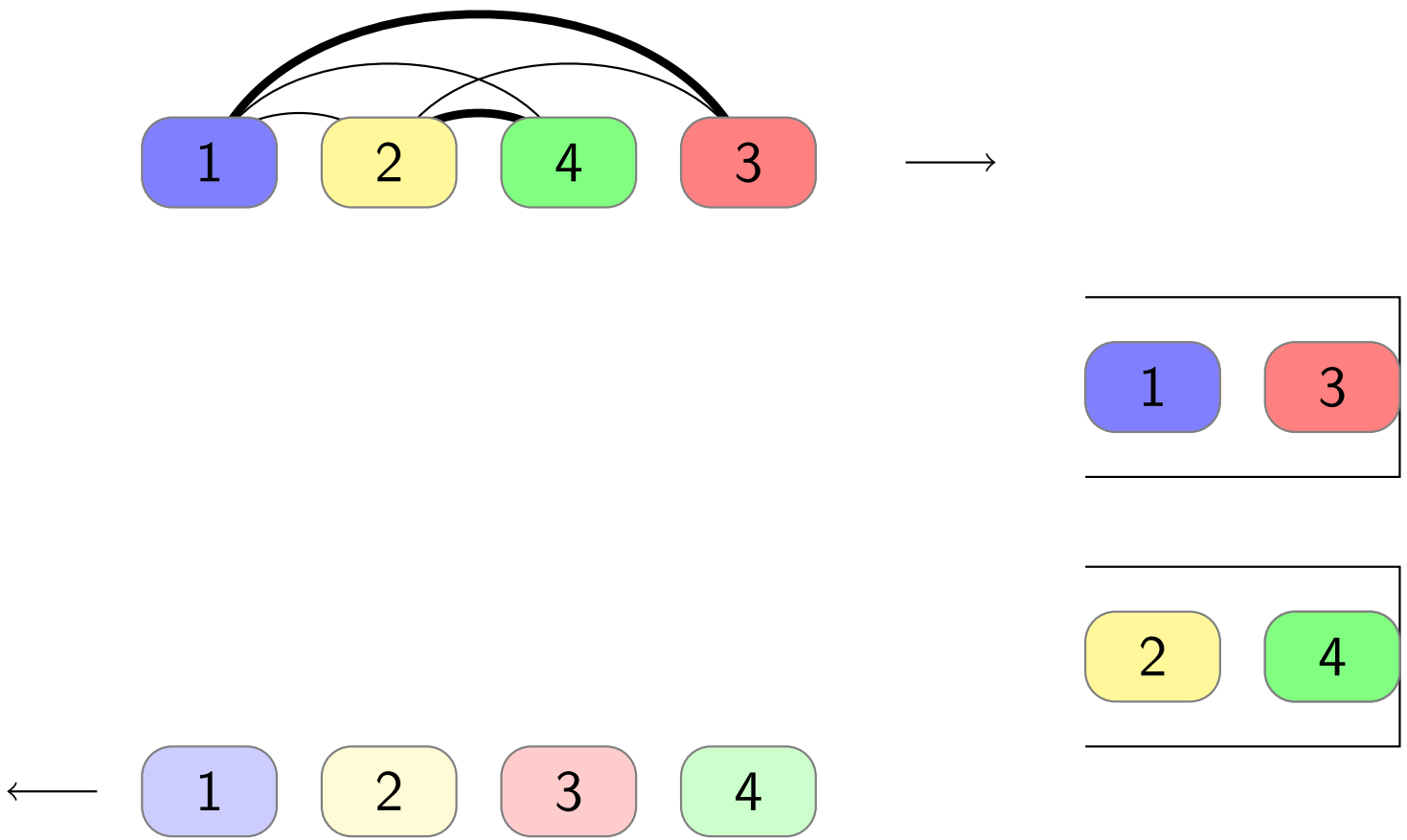
Trains - Permutations, tracks taille 2

- Sommet : wagons.
- Arête : wagons pouvant être stockés sur la même track.
- Graphe de permutations.
- Perfect matching.



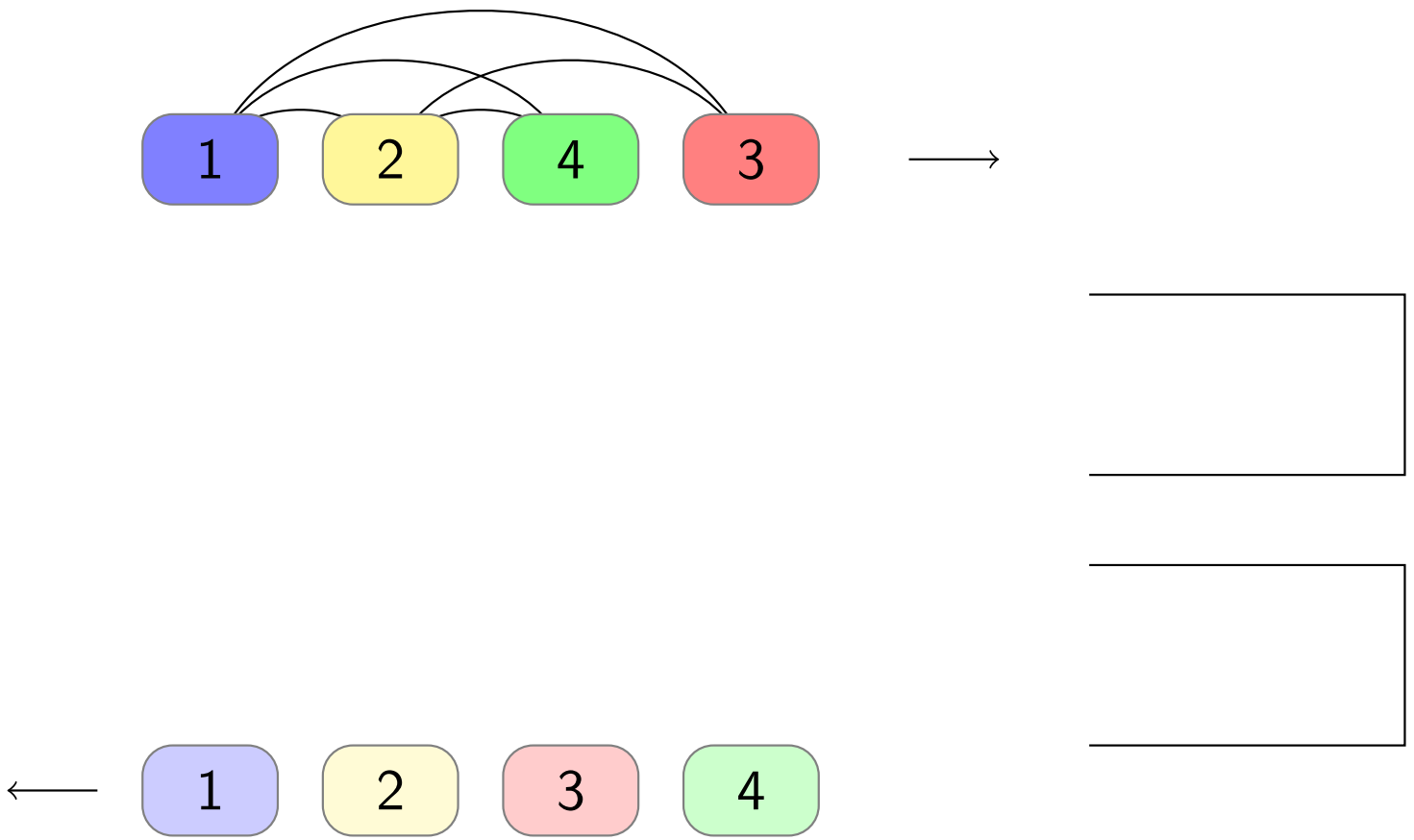
Trains - Permutations, tracks taille 2

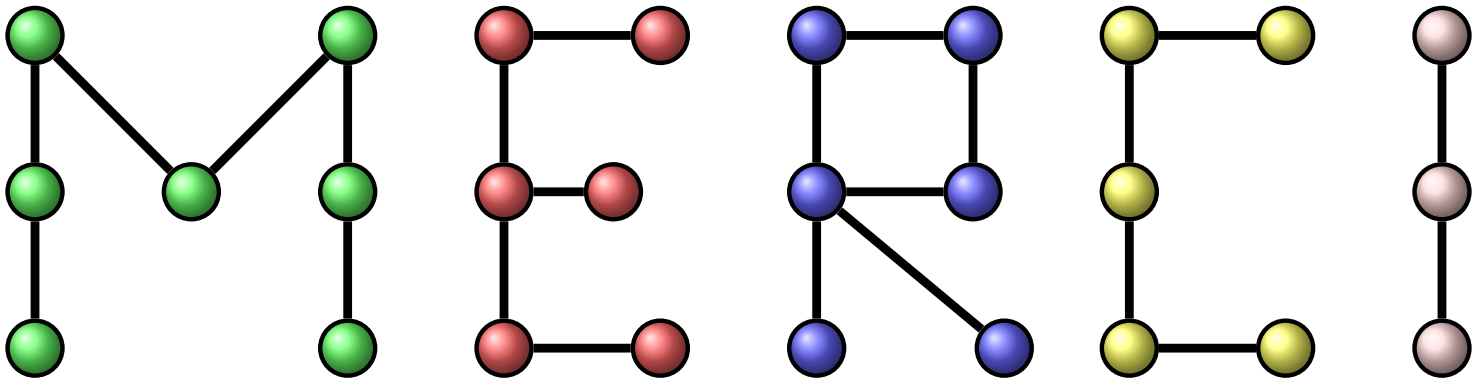
- Sommet : wagons.
- Arête : wagons pouvant être stockés sur la même track.
- Graphe de permutations.
- Perfect matching.



Trains - Permutations, tracks taille 2

- Si tracks de taille k : partitionner le graphe de permutation en chemins de taille k. Complexité ? FPT ?



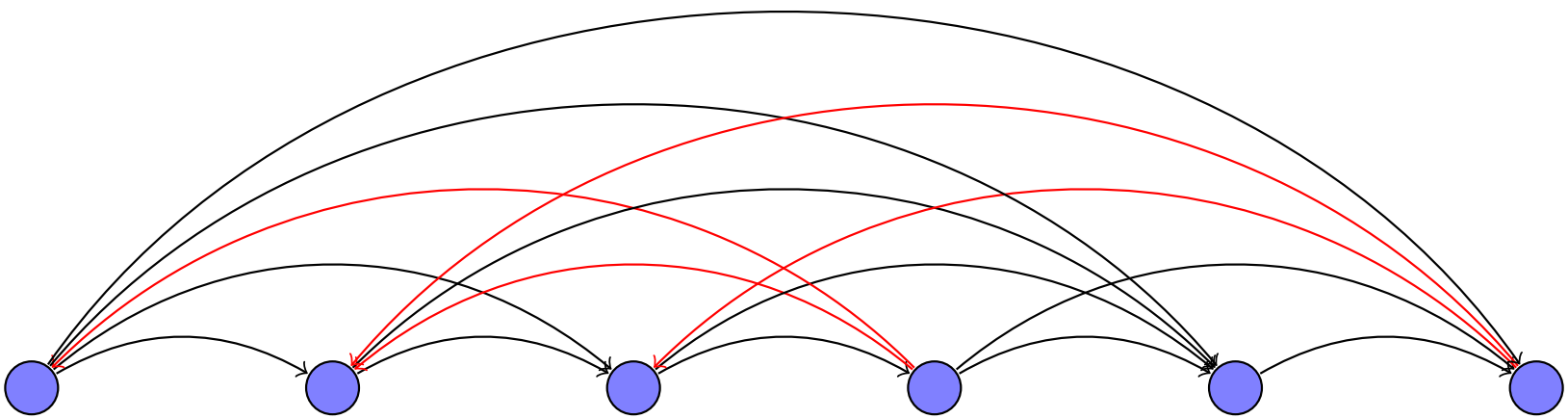

MERCI

Noyaux - FAST [Dom et al. 2006]

FAST (Feedback Arc Set in Tournament)

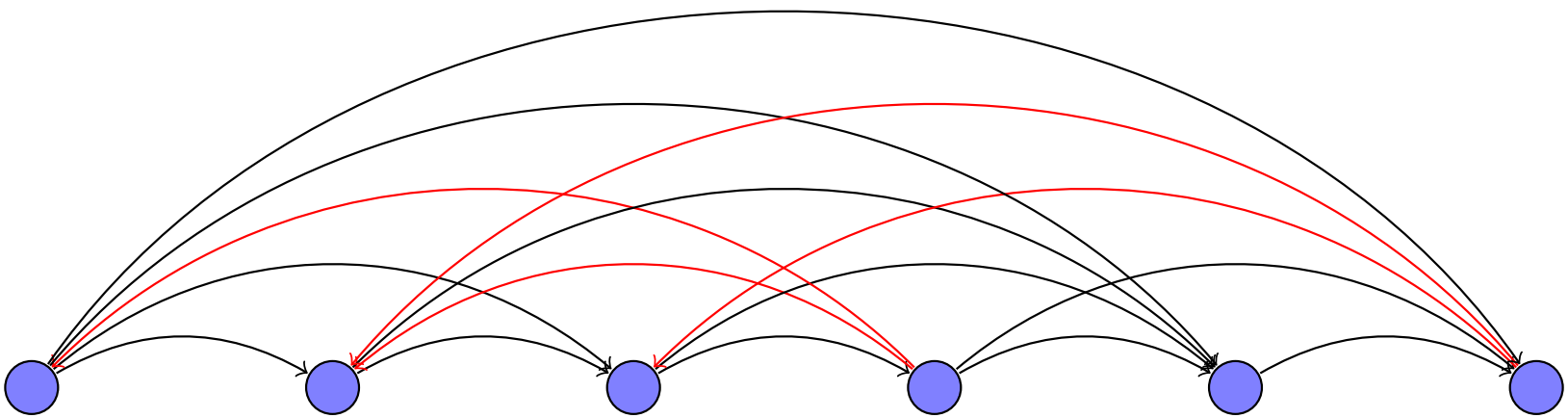
- Entrée : Un tournoi $T = (V, \overrightarrow{A})$.
- Paramètre : Un entier k.
- Question : Peut-on retourner au plus k arcs de $\overrightarrow{A}$ pour rentre T acyclique (= transitif) ?

- Tournoi : graphe orienté avec $\overrightarrow{uv}$ ou $\overrightarrow{vu}$ pour toute paire u, v.



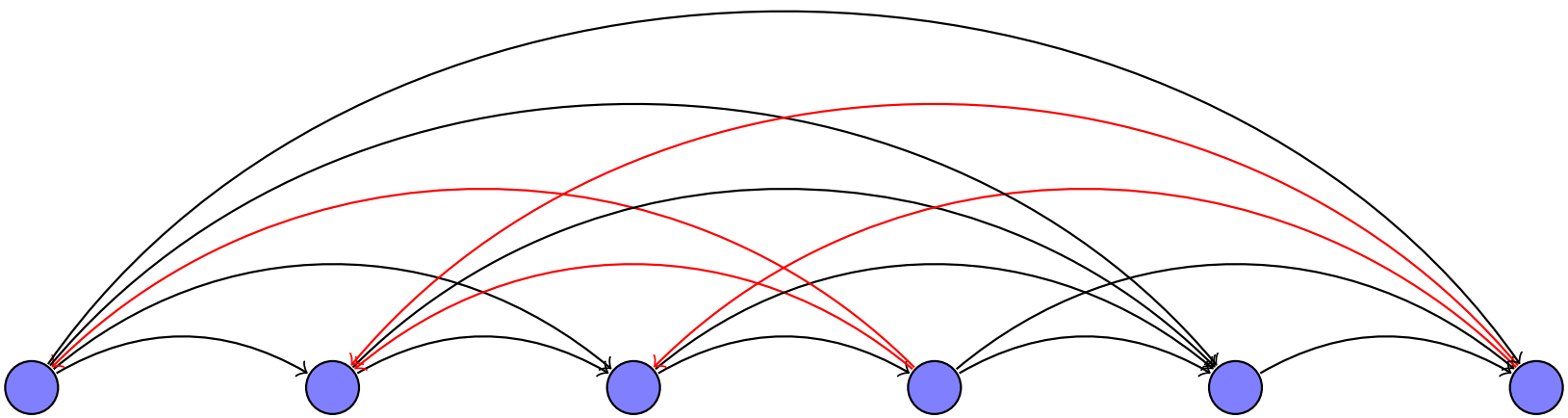
Noyaux - FAST [Dom et al. 2006]

- Deux règles de réduction polynomiales :
  1. Si un sommet n'est dans aucun triangle (orientés), le supprimer.
  2. Si un arc e est dans au moins $k+1$ triangles : inverser e et $k = k - 1$.



Noyaux - FAST [Dom et al. 2006]

- Deux règles de réduction polynomiales :
  1. Si un sommet n'est dans aucun triangle (orientés), le supprimer.
  2. Si un arc e est dans au moins $k+1$ triangles : inverser e et $k = k-1$.
     - Si la solution ne contient pas e : doit contenir un arc des $k+1$ triangles de e : trop grand.



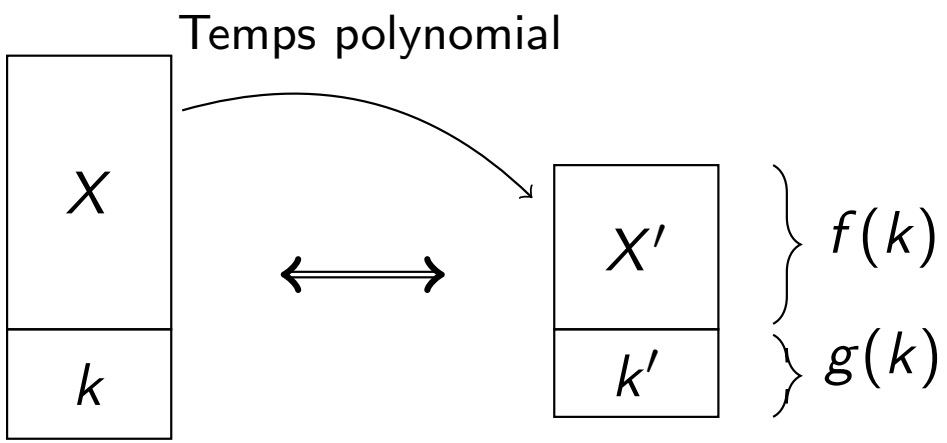
Noyaux - FAST [Dom et al. 2006]

- Règles :
  1. Si un sommet v n'est dans aucun triangle, supprimer v.
  2. Si un arc e est dans au moins $k+1$ triangles (orientés) : inverser e et $k = k-1$.

- Au plus k arcs dans la solution. Pour chacun :
  - En plus des deux sommets, voit au plus k autres sommets (sinon règle 2).
- Tout triangle de T contient un arc de la solution (définition) et tous les sommets de T sont dans un triangle (règle 1), donc au plus $k(k+2)$ sommets dans T (sinon, instance NON triviale).

Noyaux

- But : obtenir des noyaux de la **plus petite taille possible** (polynomiale).
- **Bornes inf.** : possible de prouver qu'un noyau polynomial est impossible à obtenir sauf si...
- **Bornes sup.** : méta-théorèmes si le graphe d'entrée à certaines propriétés.



Approximation en temps exponentiel

- Problème Max Independent Set.
- On sait le résoudre en $O^*(\gamma^n)$.
- Propriété héréditaire
  - Vérifie la propriété sur G ssi tous ses sous-graphes la vérifient.

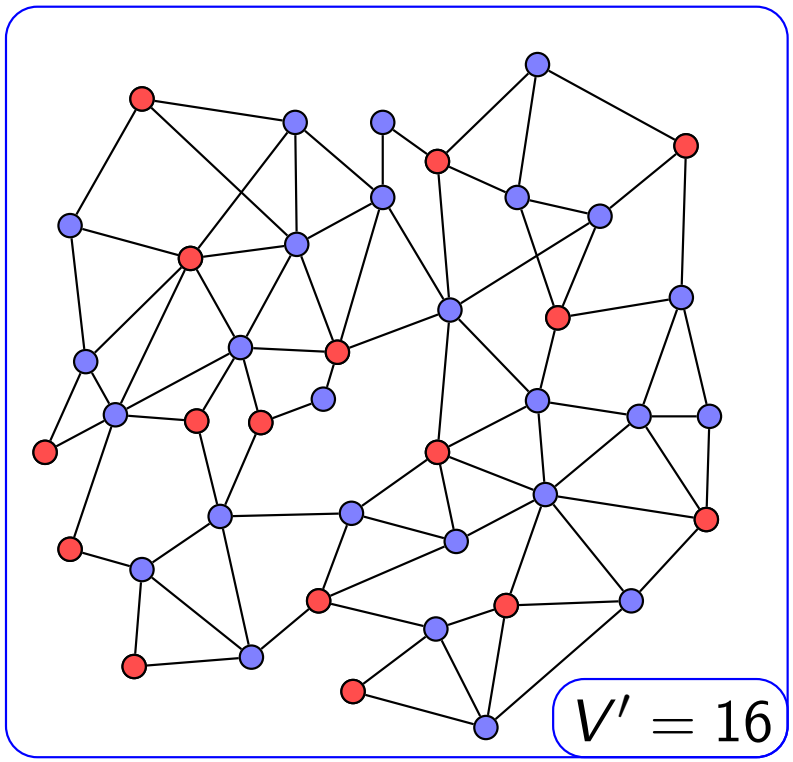


Figure E. Tourniaire

Approximation en temps exponentiel

- Problème Max Independent Set.
- On sait le résoudre en $O^*(\gamma^n)$.
- Propriété héréditaire
  - Vérifie la propriété sur G ssi tous ses sous-graphes la vérifient.
- Partager l'instance en 2 parts égales.
- Résoudre sur chaque partie.
- Renvoyer le plus grand tel quel.
- Ratio $1/2$: $O^*(\gamma^{n/2})$.

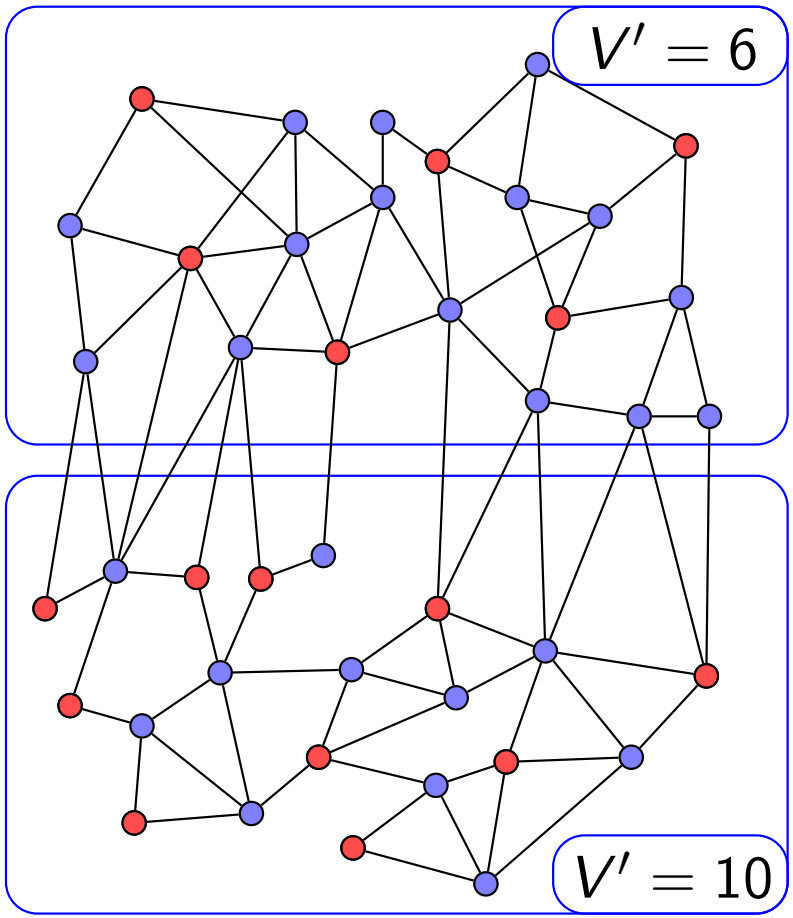


Figure E. Tourniaire

Approximation en temps exponentiel

- Problème Max Independent Set.
- On sait le résoudre en $O^*(\gamma^n)$.
- Propriété héréditaire
  - Vérifie la propriété sur G ssi tous ses sous-graphes la vérifient.
- Partager l'instance en 2 parts égales.
- Résoudre sur chaque partie.
- Renvoyer le plus grand tel quel.
- Ratio $1/2$: $O^*(\gamma^{n/2})$.
- Ratio ρ : $O^*(\gamma^{\rho n})$.

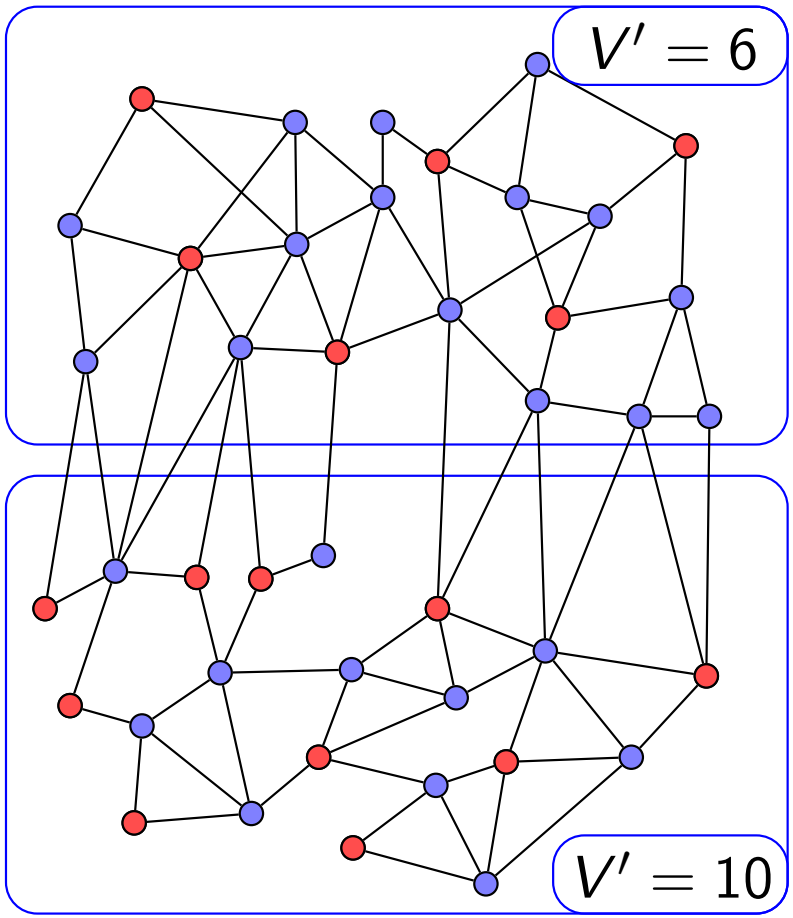


Figure E. Tourniaire

Approximation en temps exponentiel

- Problème Max Independent Set.
- Exact en $O^*(1.2127^n)$ [Bourgeois et al.].
- Pas d'approx à moins de $n^{1-\epsilon}, \forall \epsilon > 0$ en temps polynomial.

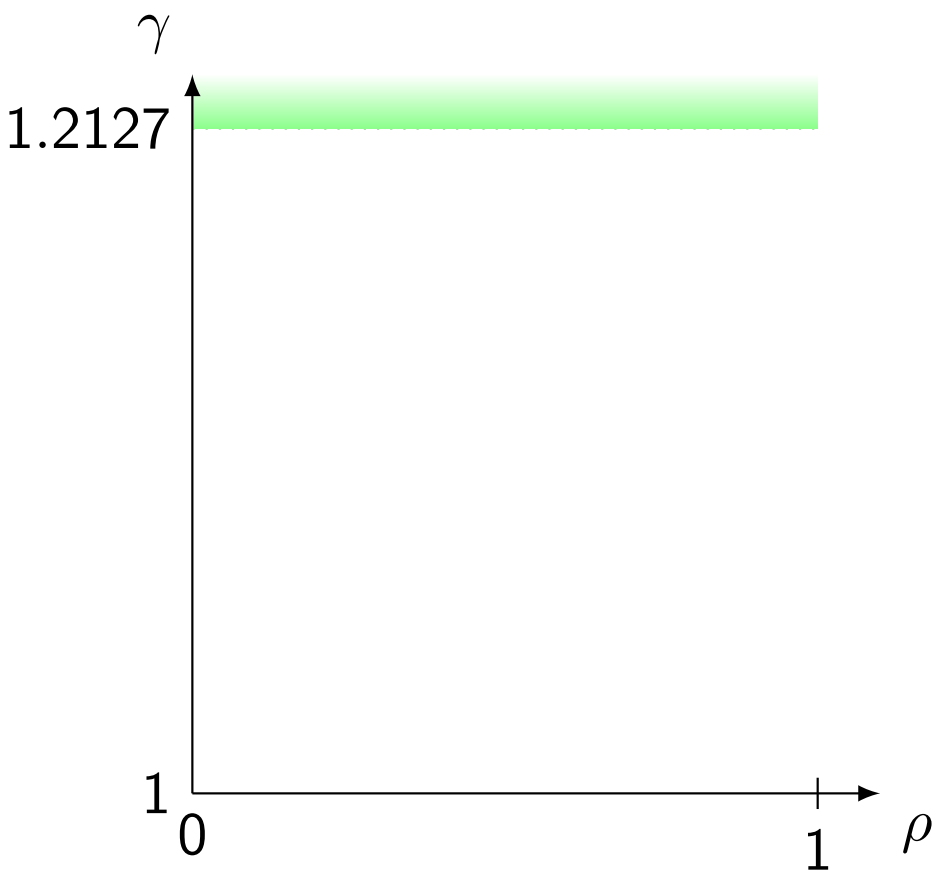


Figure E. Tourniaire

Approximation en temps exponentiel

- Problème Max Independent Set.
- Exact en $O^*(1.2127^n)$ [Bourgeois et al.].
- Pas d'approx à moins de $n^{1-\epsilon}, \forall \epsilon > 0$ en temps polynomial.

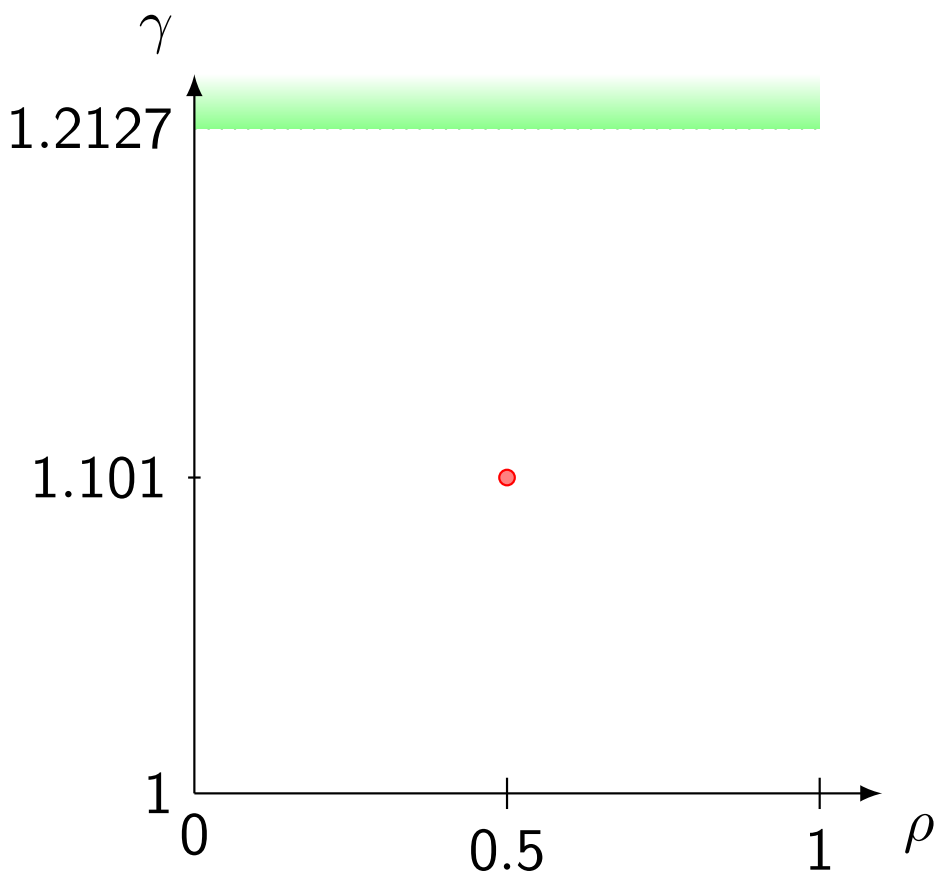


Figure E. Tourniaire

Approximation en temps exponentiel

- Problème Max Independent Set.
- Exact en $O^*(1.2127^n)$ [Bourgeois et al.].
- Pas d'approx à moins de $n^{1-\epsilon}, \forall \epsilon > 0$ en temps polynomial.
- Compromis ratio / temps.

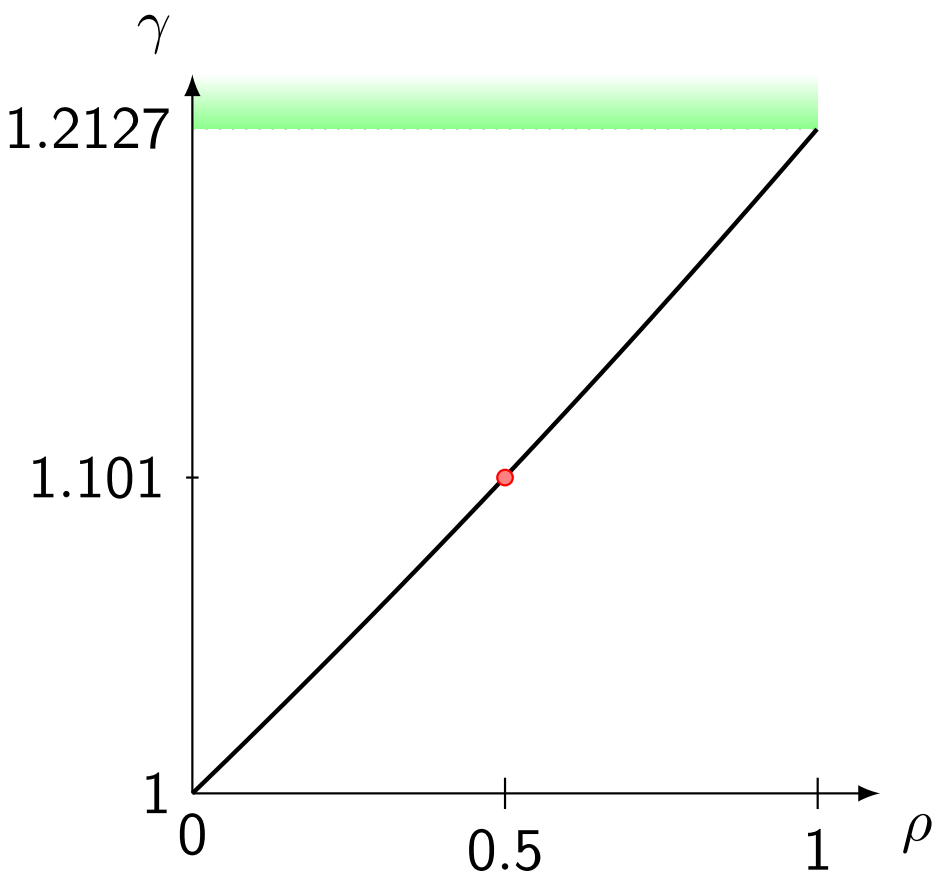


Figure E. Tourniaire