Un problème de tournée pour les sytèmes de transport en libre-service

Travail commun avec D. Chemla, R. Wolfler Calvo, et divers étudiants en stage

Redistribuer les vélos

Dans la gestion d'un système de vélos en libre-service (penser Vélib'), la redistribution des vélos en fin de la nuit est essentielle pour la qualité de service.

Cette redistribution prépare le rush du matin.

On peut supposer en première approximation que les vélos ne bougent pas.

Sur une zone, on a un camion ; pour toute station v, on a x_v vélos, et on en veut y_v (déterminé par ailleurs).

Quel trajet faire suivre au camion et quelles opérations lui faire faire afin d'atteindre cet état-cible, et ce au coût minimum ?

Formalisation : problème de tournée sur un graphe

Données Un graphe $G = (V, E)$ (V=stations, E=rues) ; $\boldsymbol{d} \in \mathbb{R}^E$ une distance ; $\boldsymbol{x} \in \mathbb{Z}_+^V$ un remplissage initial ; $\boldsymbol{y} \in \mathbb{Z}_+^V$ un remplissage cible ; un camion de capacité K.

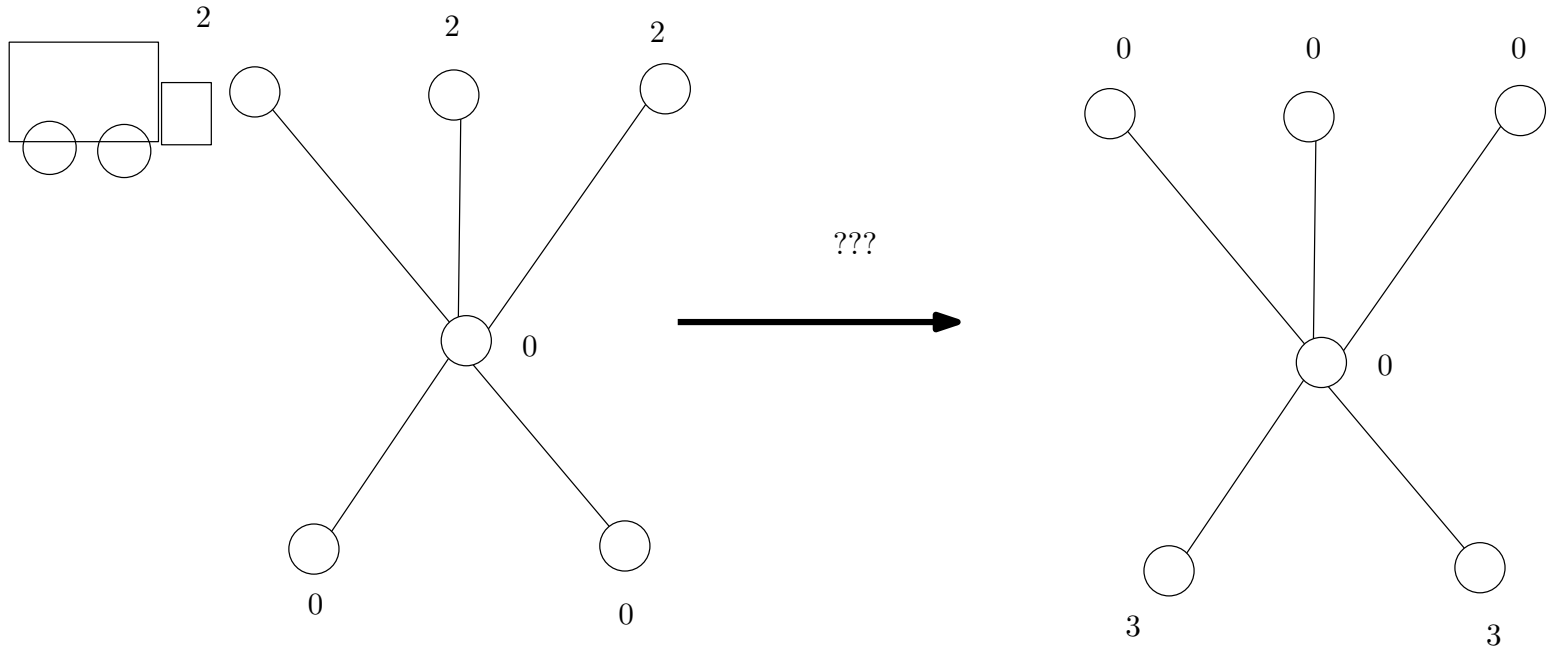
Tâche Trouver la tournée la plus courte permettant de faire passer les stations de $\boldsymbol{x}$ à $\boldsymbol{y}$.

Remarque : on suppose que $x(V) = y(V)$.

On a le droit de poser, puis reprendre des vélos

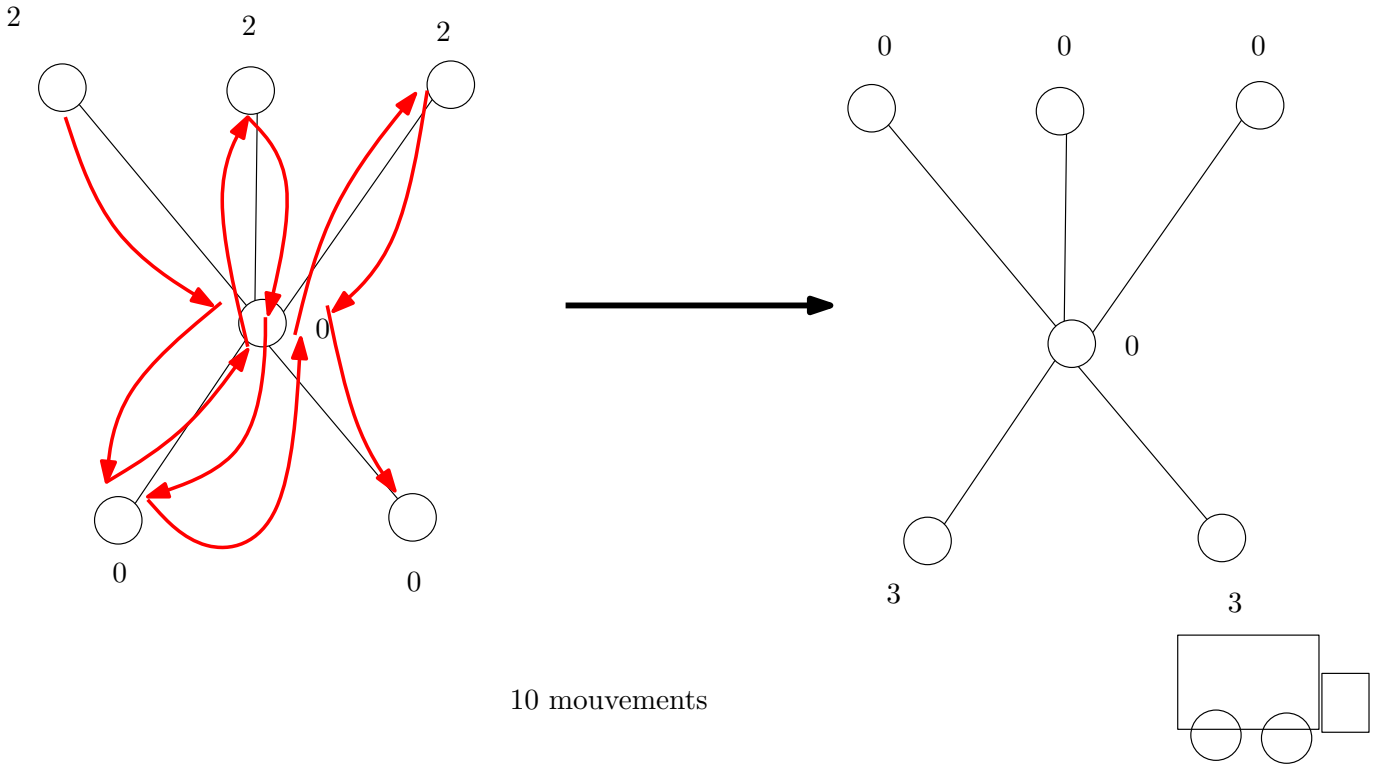
On se met dans un cas préemptif.

$K = 3$



Sans utiliser ce droit

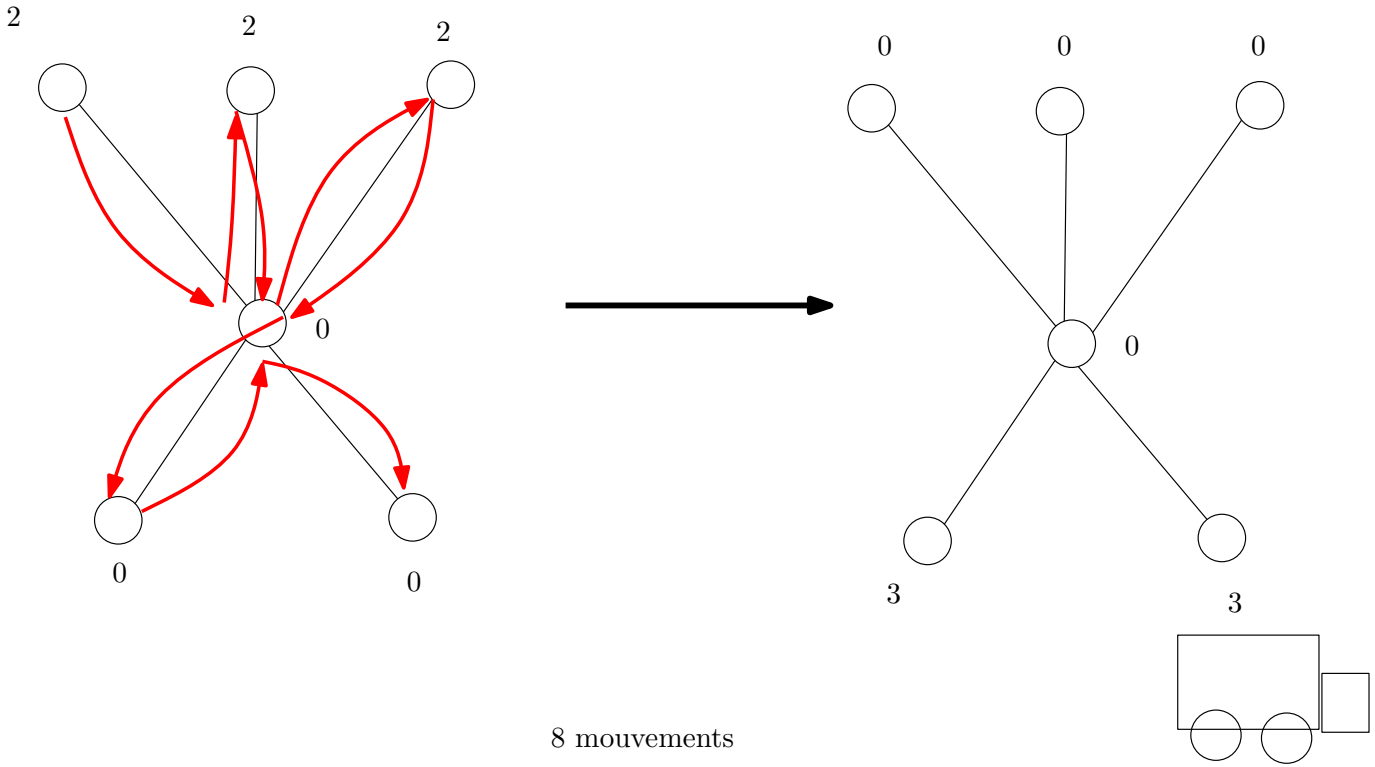
$K = 3$



10 mouvements

En l'utilisant : c'est mieux

$K = 3$



8 mouvements

C'est un problème d'optimisation difficile

Le problème précédent est NP-dur, même si $K = 1$

il contient le voyageur de commerce, 2-partition, split delivery, ...

Il existe quelques problèmes proches dans la littérature

Hernadez Pérez et Salazar Gonzáles (2004) : le 1-PDTSP – one-commodity pickup and delivery problem – quasiment notre problème, à ceci près que la solution doit nécessairement être un cycle hamiltonien.

Annily et Hassin (1992) : le Swapping Problem, quasiment notre problème, sauf que les offres et les demandes sont toutes unitaires, mais il peut y avoir des biens de types variés.

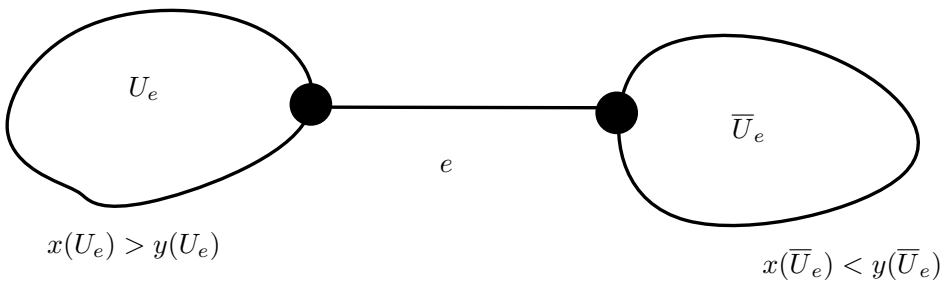
Un cas particulier polynomial : arbre

Théorème

Le problème peut être résolu en temps polynomial si le graphe est un arbre.

On a même : le camion passe exactement par toute arête $e \in E$ un nombre de fois

$$\simeq 2 \left\lceil \frac{x(U_e) - y(U_e)}{K} \right\rceil$$



Un algorithme glouton

Tant qu'il existe des stations qui ne sont pas à leur état-cible, répéter (v position courante du camion ; $\tilde{\boldsymbol{x}}$ répartition courante des vélos).

1. considérer $Q_1, Q_2, \ldots, Q_s$ les composantes connexes de $G \setminus \{v\}$
2. S'il y a un composante en excès entrer dans une composante Q_i en excès.
3. S'il n'y a pas de composante en excès
   - choisir un Q_i en défaut,
   - amener le camion à $\min(K, y(Q_i) - \tilde{x}(Q_i))$ vélos,
   - entrer dans Q_i.

Un autre cas polynomial ?

Question ouverte

Y a-t-il un algorithme polynomial qui résout ce problème si le graphe G est un cycle ?

Un algorithme 9.5-approximatif

Théorème

Il existe un algorithme 9.5-approximatif pour le problème.

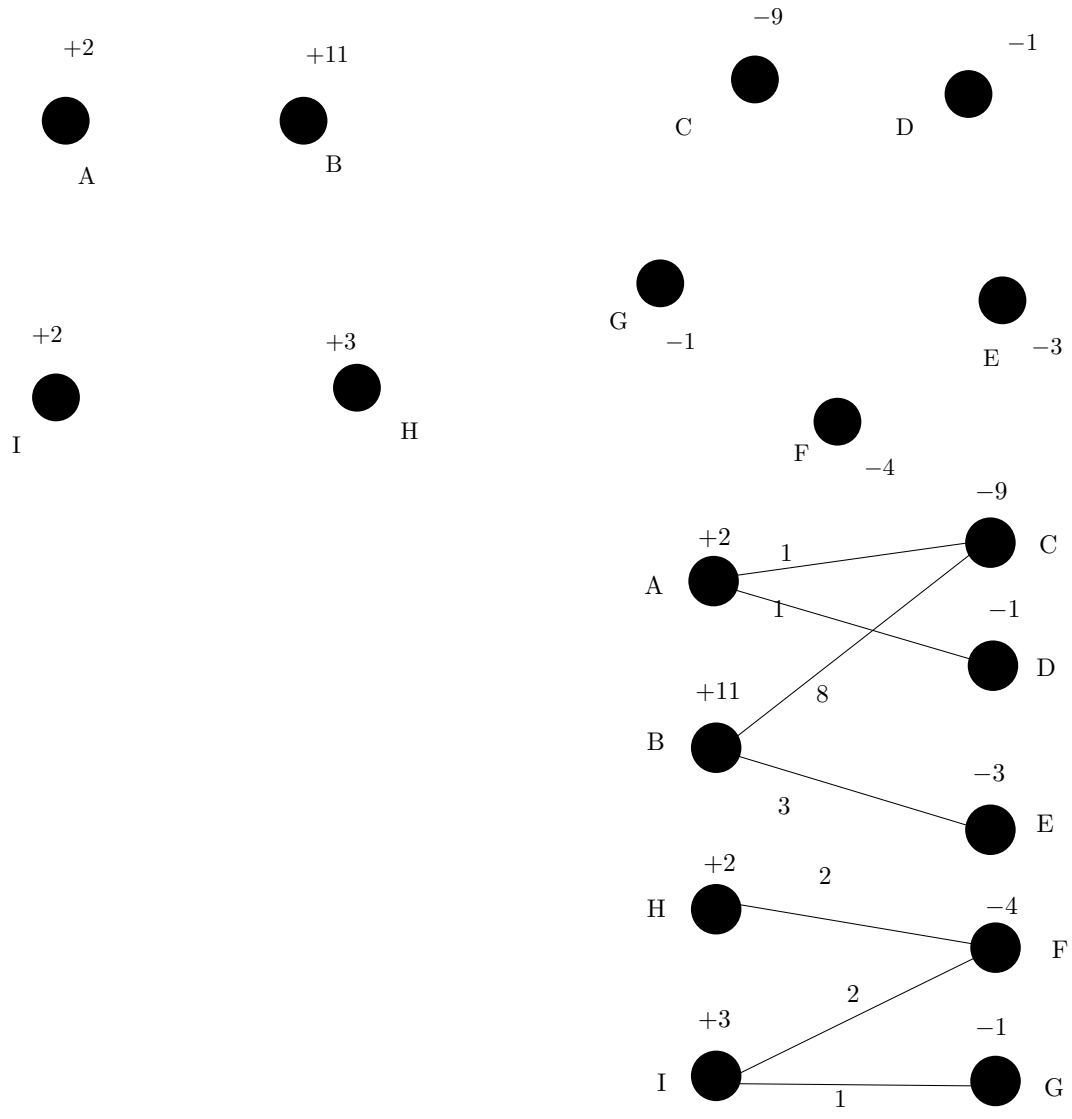
On s'inspire d'un alghorithme de Chalasani et Motwani pour le Swapping Problem avec un seul type d'objet.

Les étapes de cet algorithme sont...

Etapes de l'algorithme :

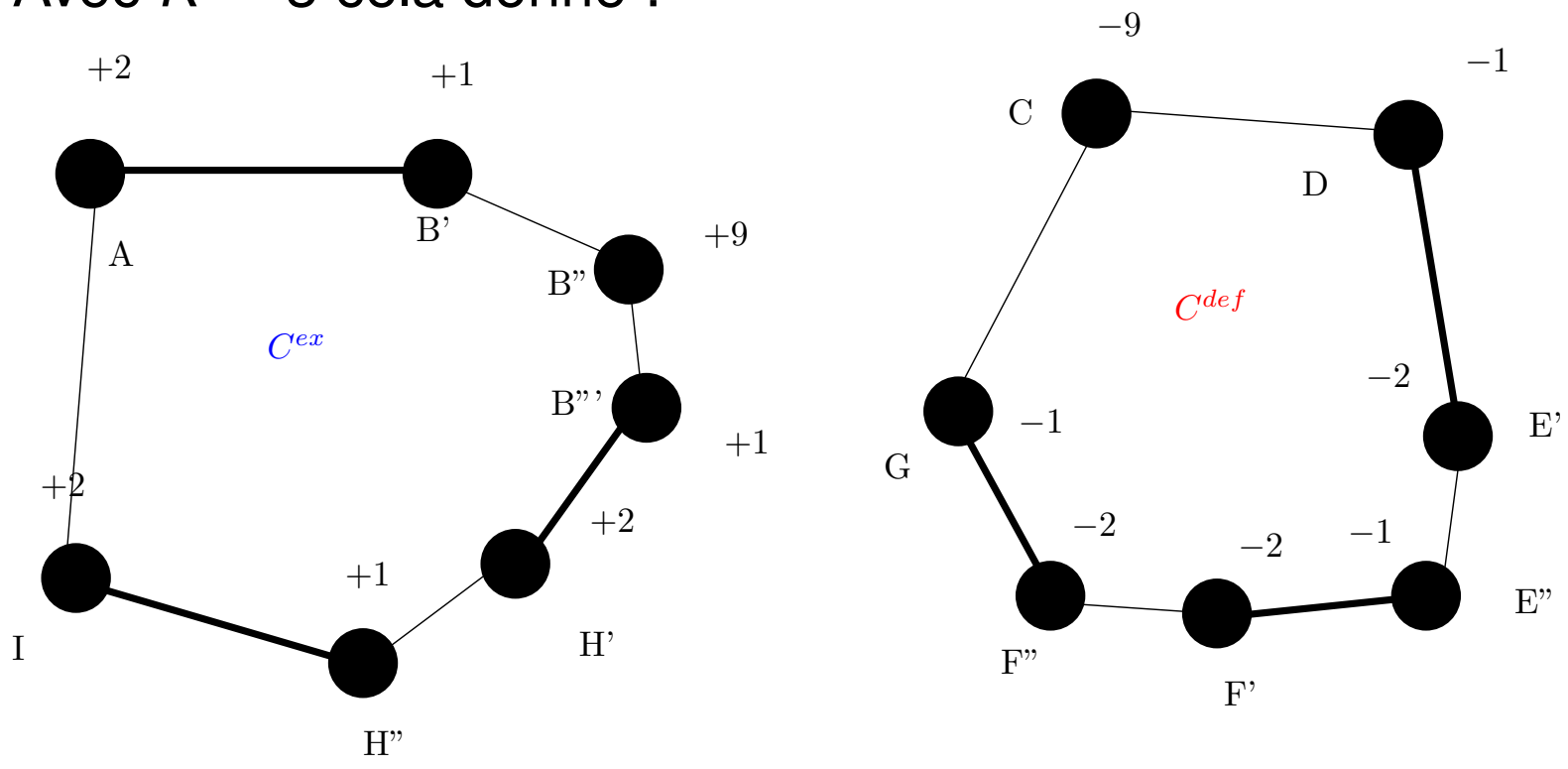
- chercher un b-couplage parfait Mo de coût minimum entre les sommets en excès et les sommets en défaut
- calculer un tour C^{ex} passant par tous les sommets en excès
- calculer un tour C^{def} passant par tous les sommets en défaut
- faire sur chacun de ces tours des "paquets" d'excès et de défaut un multiple de K
- utiliser Mo pour transvaser les vélos des paquets en excès vers les paquets en défaut

b-couplage entre sommets en excès et sommets en défaut



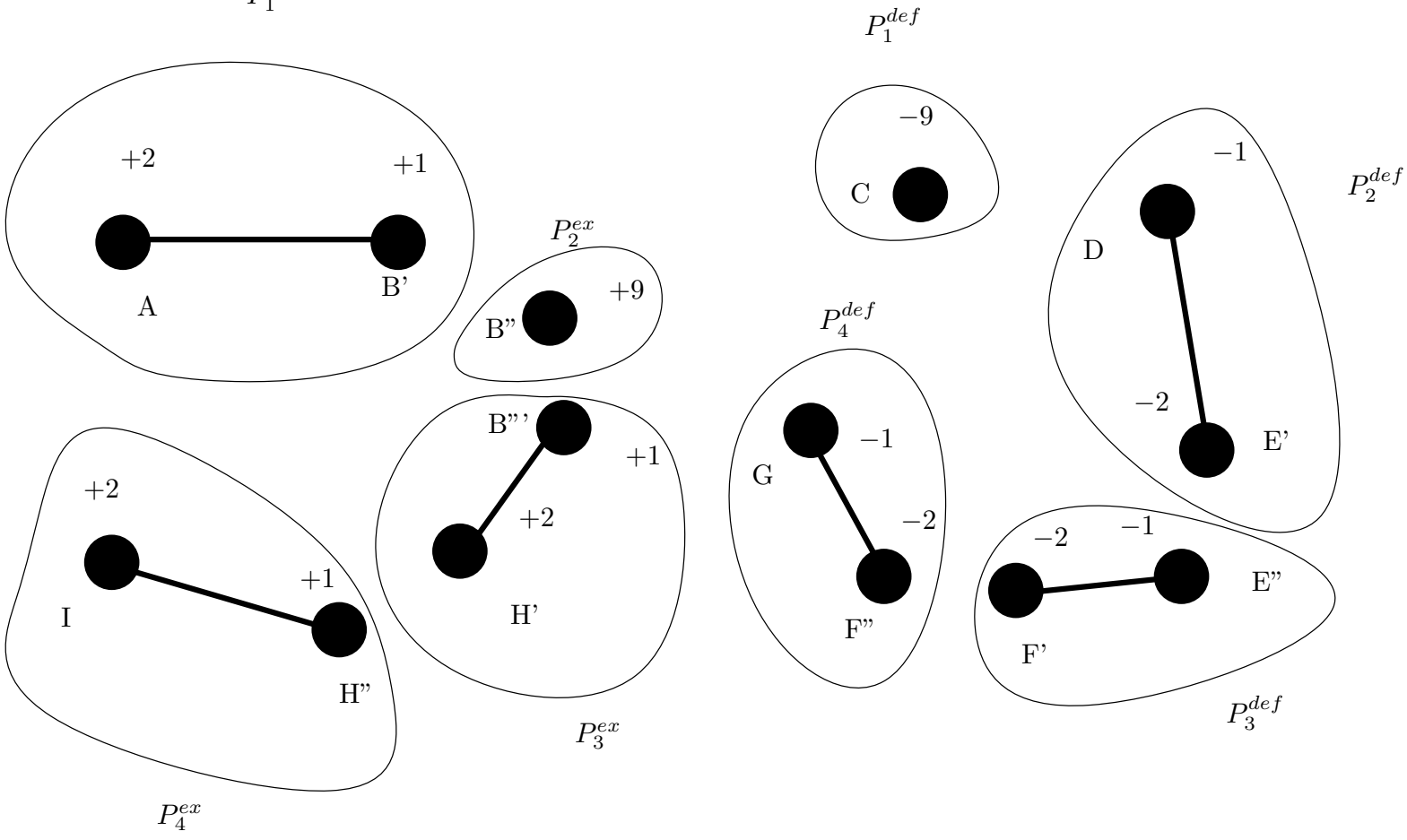
Tour sur les sommets en excès et tour sur ceux en défaut

Avec $K = 3$ cela donne :

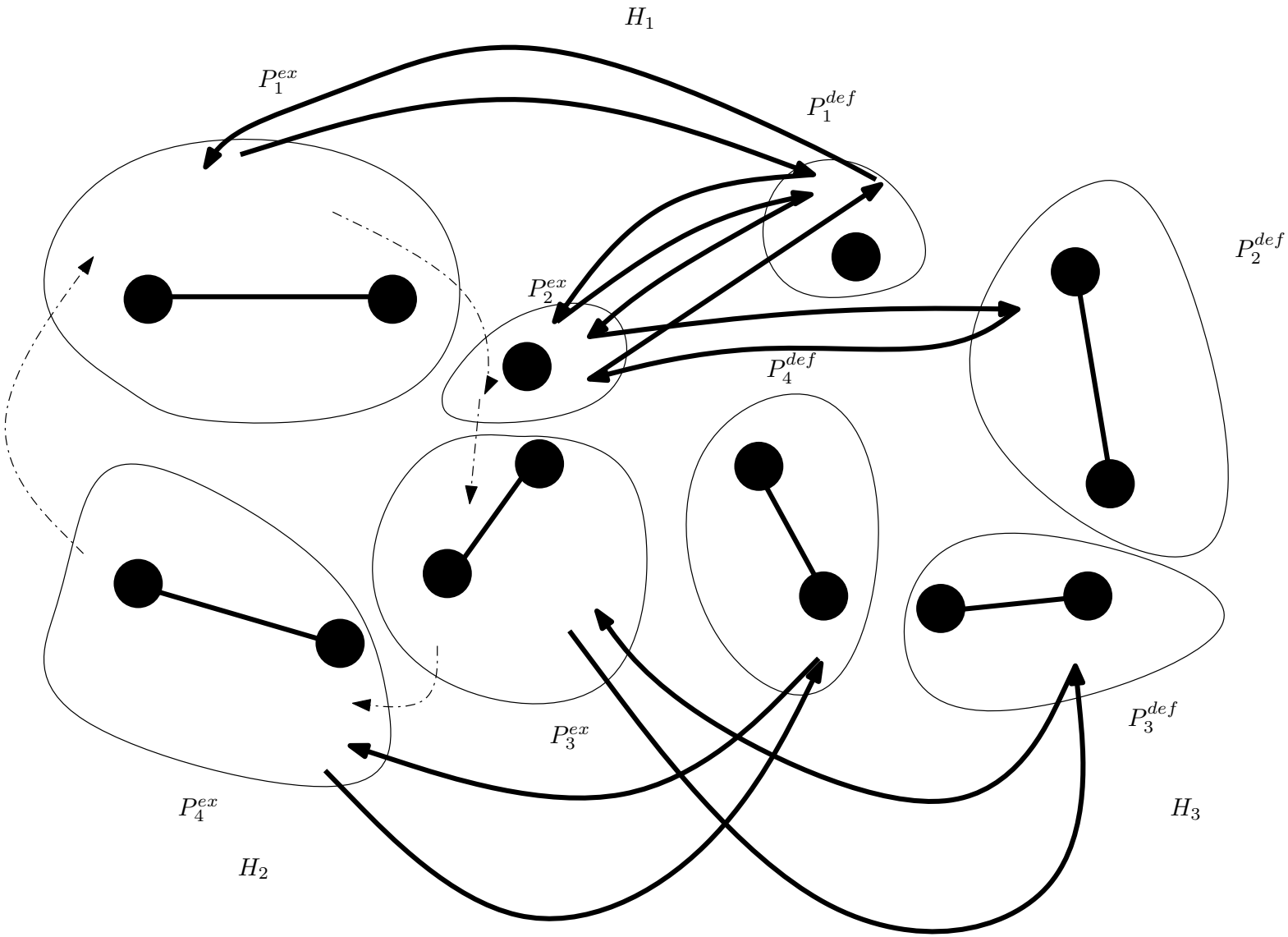


Construction de paquets de αK vélos

Avec $K = 3$ cela donne :



Tout ensemble



On a bien un 9.5 approximation

$$\begin{aligned} SOL &\leq 2C^{ex} + 2C^{def} + 2/KMo + C^{ex} \\ &\leq 4.5OPT + 3OPT + 2OPT \\ &= 9.5OPT \end{aligned}$$

avec l'aide Christofidès et de König (version coloration).

Un codage combinatoire des solutions optimales

Proposition

Soit $s_1, s_2, \ldots, s_k$ une suite de stations. Il existe un algorithme polynomial (très rapide) qui régule au mieux pour cette suite de stations visitées par le camion.

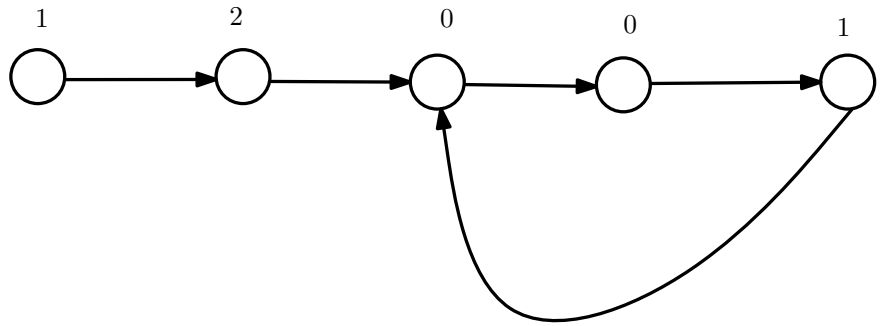
Qui régule au mieux :
Trouve $\boldsymbol{x}'$ et $\boldsymbol{y}'$ tels que

- $x'(V) = y'(V)$
- $x'_v \leq x_v$ et $y'_v \leq y_v$ pour tout $v \in V$
- maximise $x'(V)$.

Curieusement, cela résout aussi

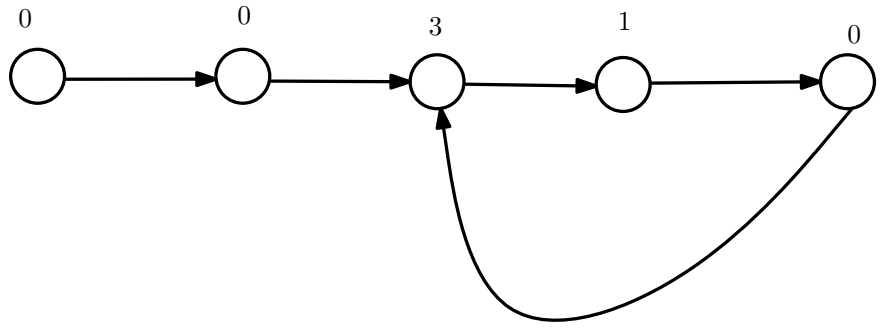
- $x'(V) = y'(V)$
- $x'_v = x_v$ pour tout $v \in V$
- minimise $\sum_{v \in V} |y_v - y'_v|$.

$x =$



???

$y =$



Une recherche locale

D'après la proposition précédente, on peut donc limiter l'exploration à des suites de stations.

On définit des transformations possibles...

- 2-OPT
- suppression de sommet
- ajout de sommet
- ...

Itérer de telles transformations $\rightarrow$ recherche locale

Une borne inférieure par la programmation linéaire

Une borne inférieure au problème est donnée par

$$\min \quad \sum_{u,v \in V} d_{uv} z_{uv}$$

$$\text{s.c.} \quad \sum_{u \in V} z_{uv} = \sum_{w \in V} z_{vw} \qquad \text{pour tout } v \in V$$

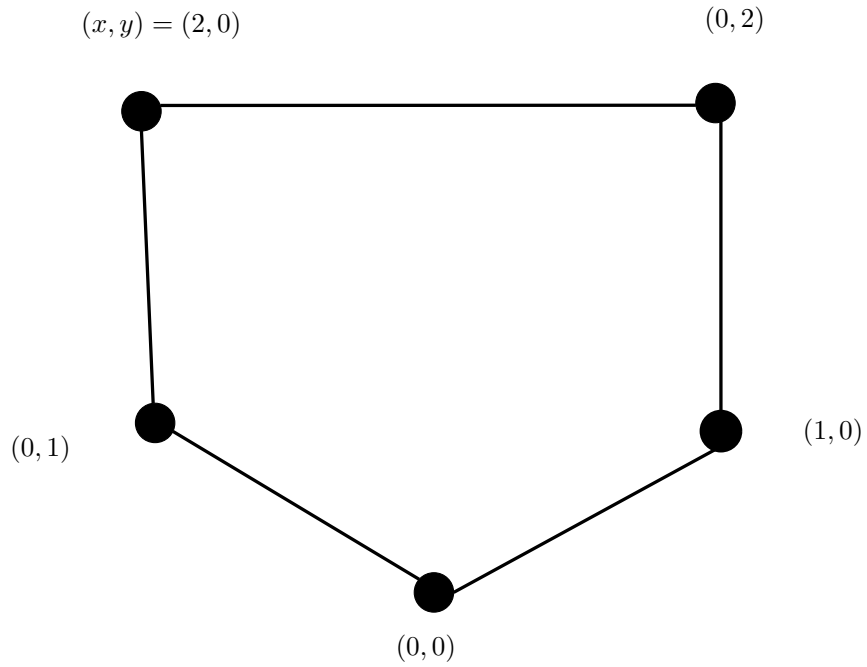
$$\sum_{u \in X,\, v \notin X} z_{uv} \geq \left\lceil \frac{|x(X) - y(X)|}{K} \right\rceil \qquad \text{pour tout } X \subseteq V \setminus \{0\}$$

$$z_{uv} \in \mathbb{Z}_+ \qquad \text{pour tout } u, v \in V$$

Cela se résout par une méthode de branch-and-cut.

Ce n'est qu'une borne inférieure

$K = 2$



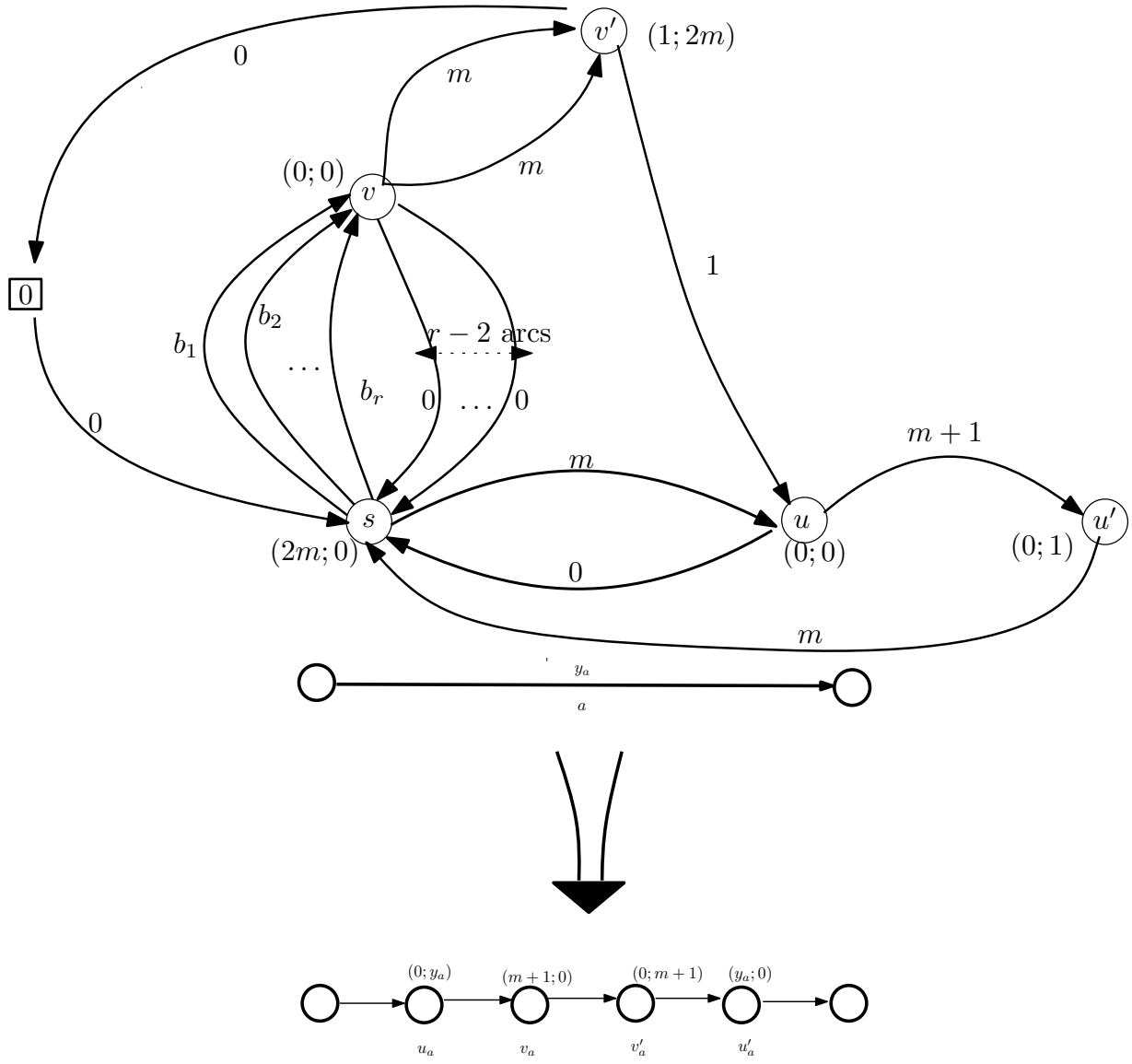
qui peut être optimale sans qu'on puisse le prouver

Proposition

Décider si une solution (réalisable ou optimale) du programme linéaire précédent est réalisable pour notre problème est NP-complet.

Dit autrement : même si on décrit une solution par les arêtes par lesquelles passe le camion et par leur sens de parcours, on ne sait pas si c'est une solution réalisable ou non.

car ça contient 2-partition



Performances combinées du branch-and-cut et de la recherche locale

Instance name	**n**	**Q**	**UB**	**Time**	**LB**	**Gap %**
n20A	20	10	4702	7	4702.00	0.00
n20C	20	10	6013	14	6012.00	0.02
n20B	20	10	4769	8	4769.00	0.00
n20A	20	30	3583	4	3583.00	0.00
n20E	20	30	4556	5	4299.00	5.98
n20F	20	30	4108	5	4108.00	0.00
n40E	40	10	6424	2253	6424.00	0.00
n40F	40	10	7095	10509	6760.00	4.96
n40J	40	10	6268	10067	6267.00	0.02
n40A	40	30	4949	178	4949.00	0.00
n40C	40	30	4692	450	4644.00	1.03
n40B	40	30	5110	301	5110.00	0.00
n60H	60	10	8208	11328	7707.44	6.49
n60B	60	10	8723	11312	7508.53	16.17
n60A	60	10	8010	11349	7276.80	10.08
n60G	60	30	6360	1264	6360.00	0.00
n60I	60	30	6766	8234	6390.00	5.88
n60H	60	30	6081	1835	5992.00	1.49

Résultats pour une moyenne de 10 vélos par station.

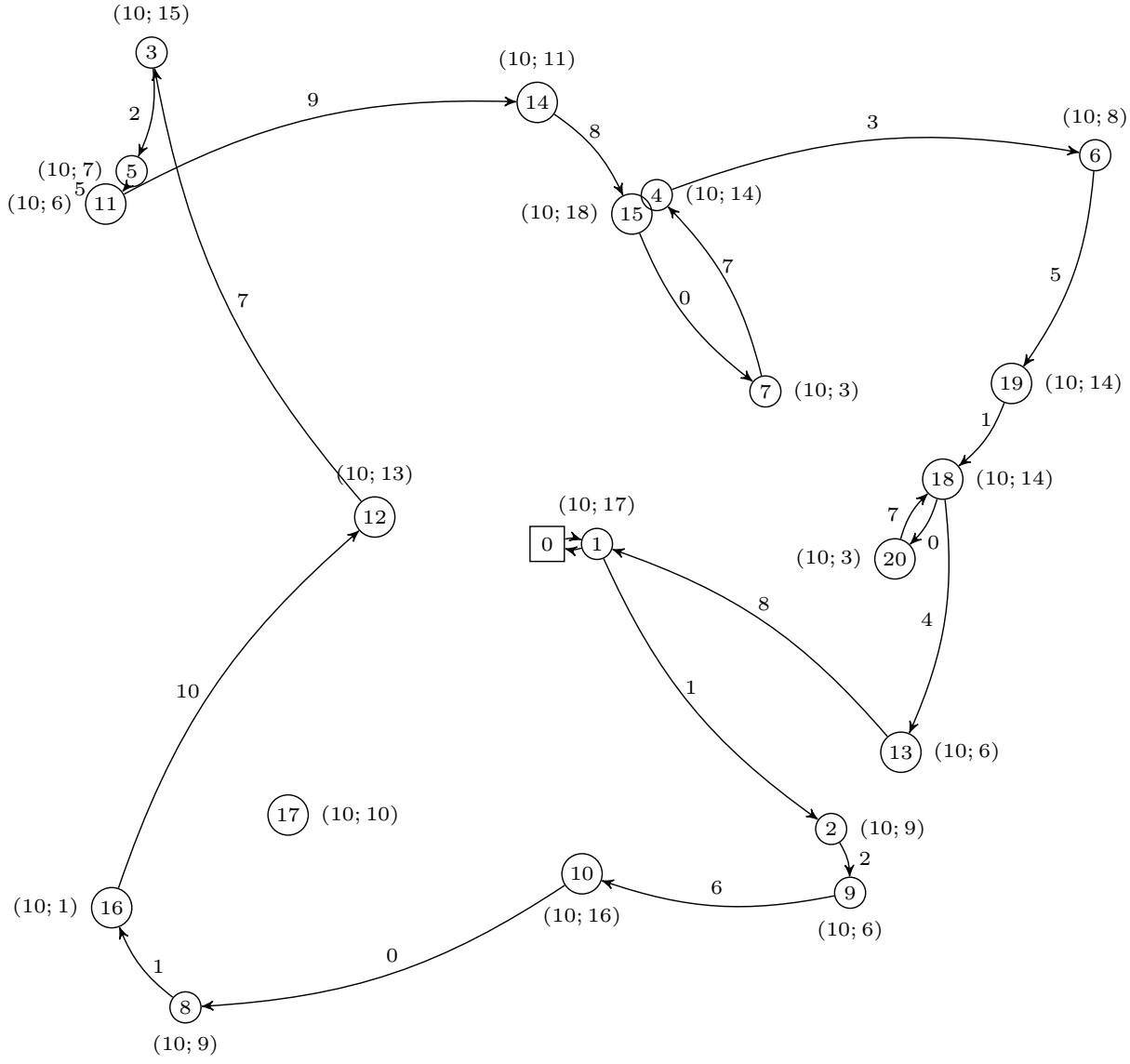


FIGURE: Une solution optimale pour une instance avec $n = 20$, $Q = 10$ et $\frac{1}{20}x(V) = 30$

Performances combinées du branch-and-cut et de la recherche locale

Instance name	**n**	**Q**	**UB**	**Time**	**LB**	**Gap %**
n20B	20	10	9883	71	9883.00	0.00
n20C	20	10	14040	137	14039.00	0.01
n20D	20	10	14925	247	14925.00	0.00
n20B	20	30	4769	16	4769.00	0.00
n20C	20	30	6013	23	6012.00	0.02
n20D	20	30	5989	16	5989.00	0.00
n40E	40	10	13159	1786	13159.00	0.00
n40F	40	10	15410	11309	14456.90	6.59
n40I	40	10	14849	2531	14849.00	0.00
n40E	40	30	6424	1024	6424.00	0.00
n40F	40	30	7240	10239	6571.83	10.17
n40I	40	30	6901	2144	6901.00	0.00
n60F	60	10	17696	11414	16925.71	4.55
n60A	60	10	18755	11075	15789.56	18.78
n60J	60	10	17462	11136	15774.62	10.70
n60H	60	30	8120	11334	7608.96	6.72
n60C	60	30	9818	11227	8313.06	18.10
n60J	60	30	8407	11357	7642.33	10.01

Résultats pour une moyenne de 30 vélos par station.

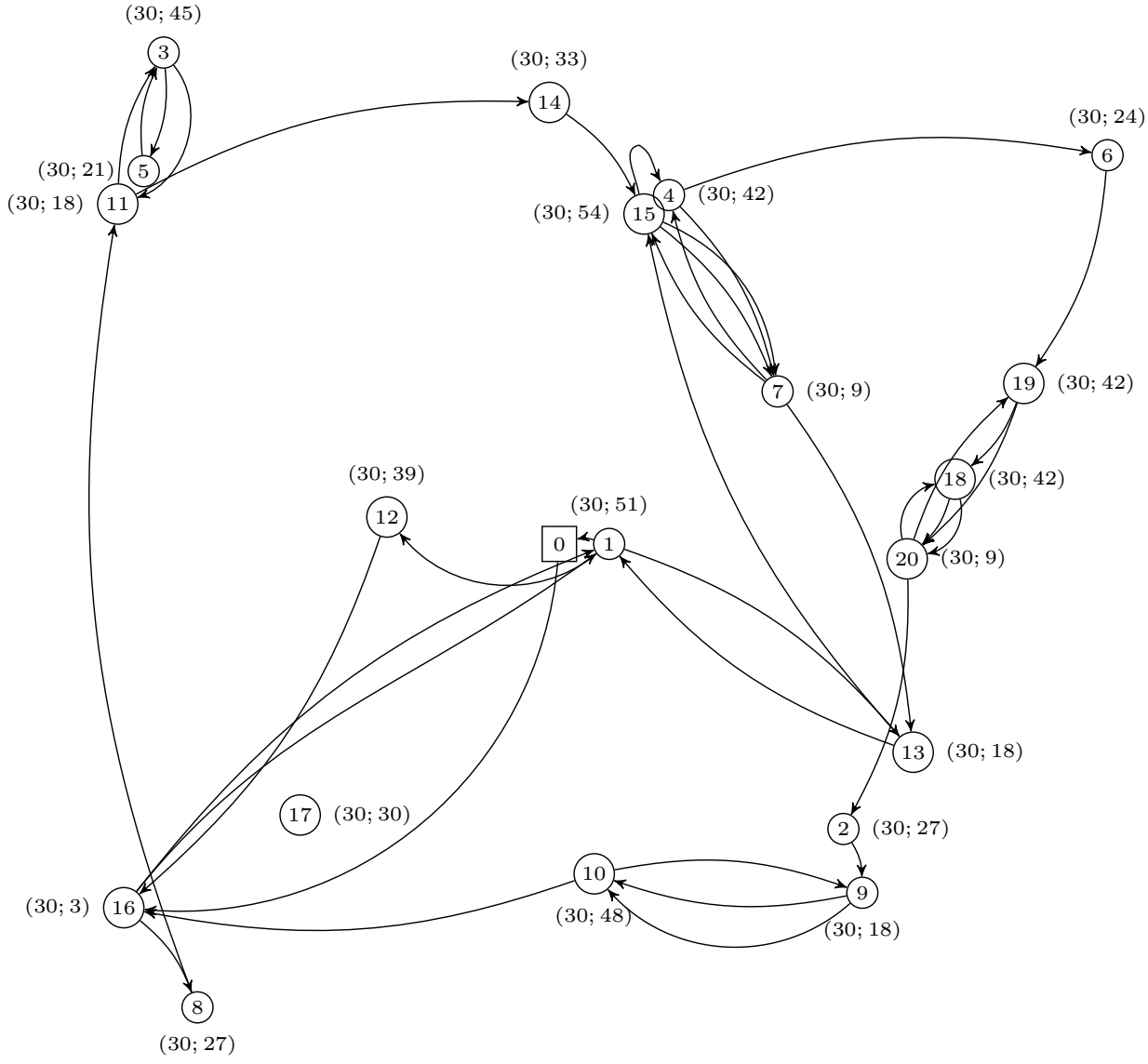


FIGURE: Une solution optimale pour une instance avec $n = 20$, $Q = 10$ et $\frac{1}{20}x(V) = 30$

Un algorithme exact ?

L'existence d'un algorithme exact efficace est une question ouverte.

Une question liée reste ouverte :

Peut-on coder de manière polynomiale toute solution optimale du problème ?

MERCI