Génération de colonnes et recherche locale itérée dans un espace de permutations pour l'Arc-Routing

Daniel Porumbel*, Gilles Goncalvez**,
Tienté Hsu**, Hamid Allaoui**

*Conservatoire National des Arts et Métiers, CEDRIC

**Université d'Artois, LGI2A

Résumé

1. Recherche Locale et Génération de Colonnes
2. Arc-Routing
3. Tests et Conclusions

Contexte

- On minimise f dans un espace de permutations
- $f(s)$ est calculé par un décodeur `Arc-Routing`

Recherche Locale Itérée

Entrée solution candidate s

1. **Repeat**
   - $s \leftarrow$ `sol-voisine`(s)
   - **if** (optimum local)
     - $s \leftarrow$ `perturber`(s)
2. **Until** condition arrêt

Sortie la meilleure solution (permutation) visitée s'

Résumé

1. Recherche Locale et Génération de Colonnes
2. Arc-Routing
3. Tests et Conclusions

Contexte

- On minimise f dans un espace de permutations
- $f(s)$ est calculé par un décodeur `Arc-Routing`

Recherche Locale Itérée

Entrée solution candidate s

1. **Repeat**
   - $s \leftarrow$ `sol-voisine`(s)
   - **if** (optimum local)
     - $s \leftarrow$ `perturber`(s)
2. **Until** condition arrêt

Sortie la meilleure solution (permutation) visitée s'

Observation pratique

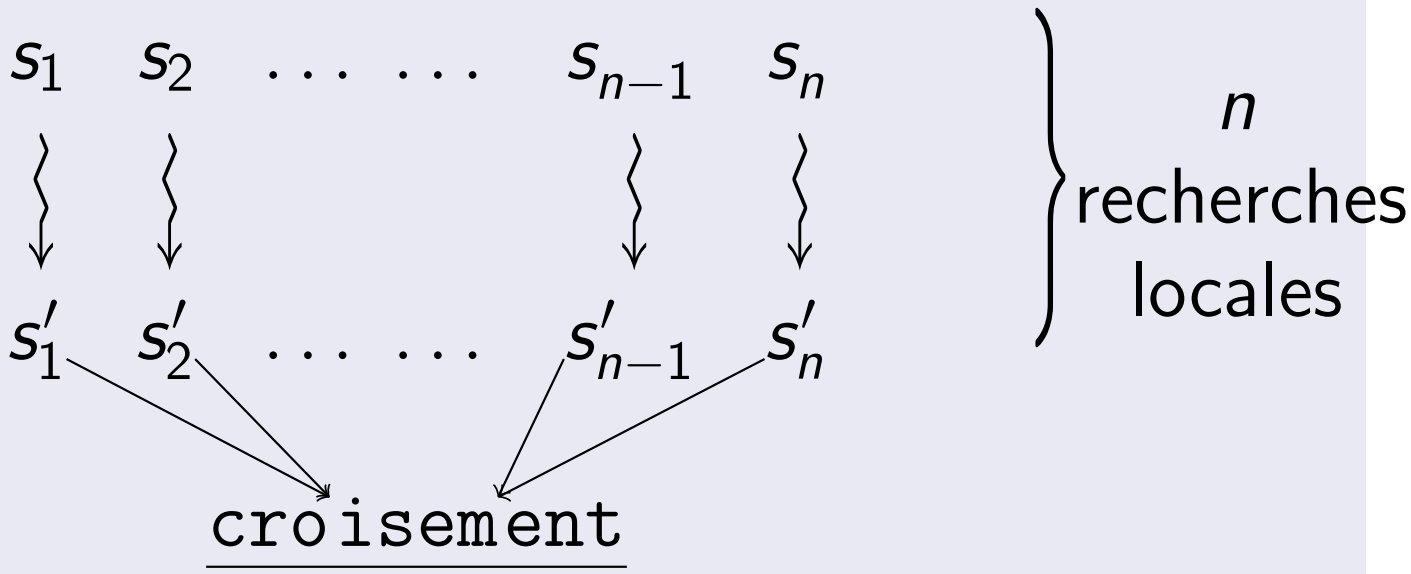
- recherche locale $\prec$ recherche locale $\oplus$ algo. génétique
- une recherche locale est souvent “boostée” par une approche mémétique:

Question : Peut-on obtenir la même valeur ajoutée sans utiliser de population?

- ni fourmi, ni abeille, ni guêpe

Observation pratique

- recherche locale $\prec$ recherche locale $\oplus$ algo. génétique
- une recherche locale est souvent "boostée" par une approche mémétique:

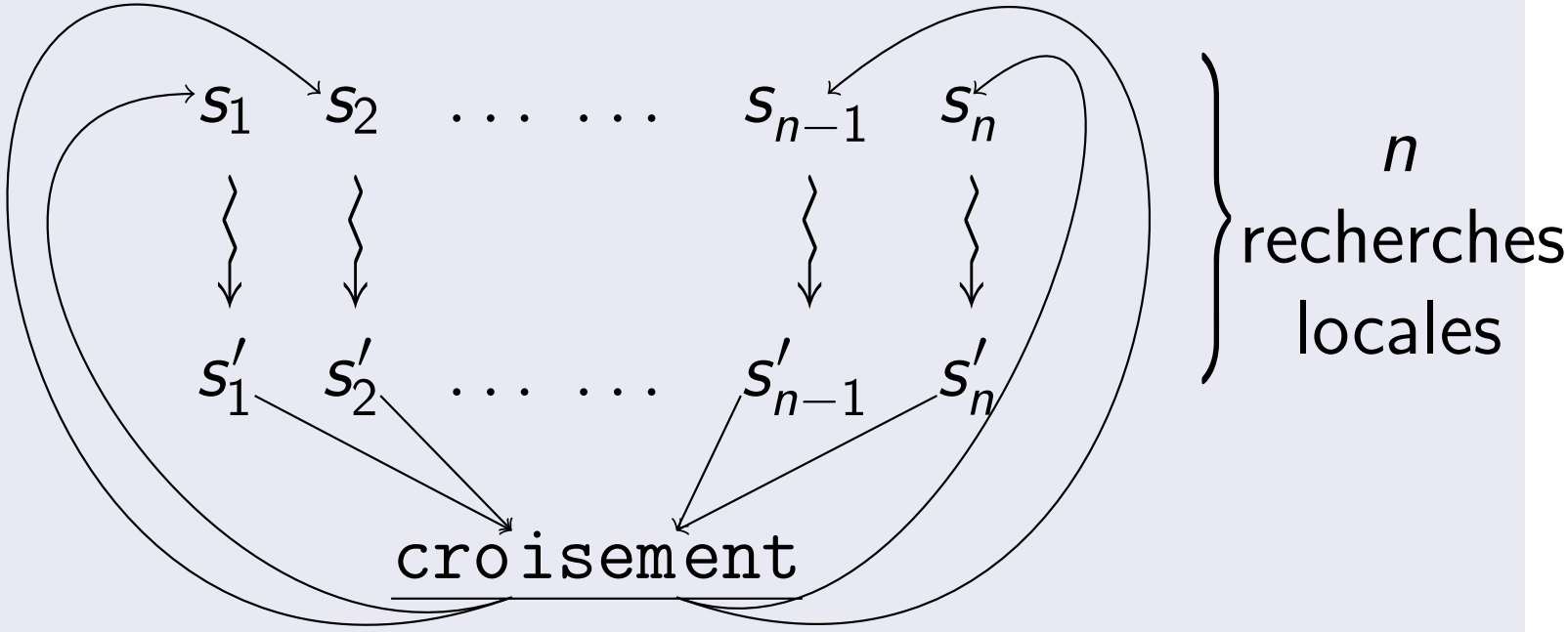


Question: Peut-on obtenir la même valeur ajoutée sans utiliser de population?

- ni fourmi, ni abeille, ni guêpe

Observation pratique

- recherche locale $\prec$ recherche locale $\oplus$ algo. génétique
- une recherche locale est souvent "boostée" par une approche mémétique:

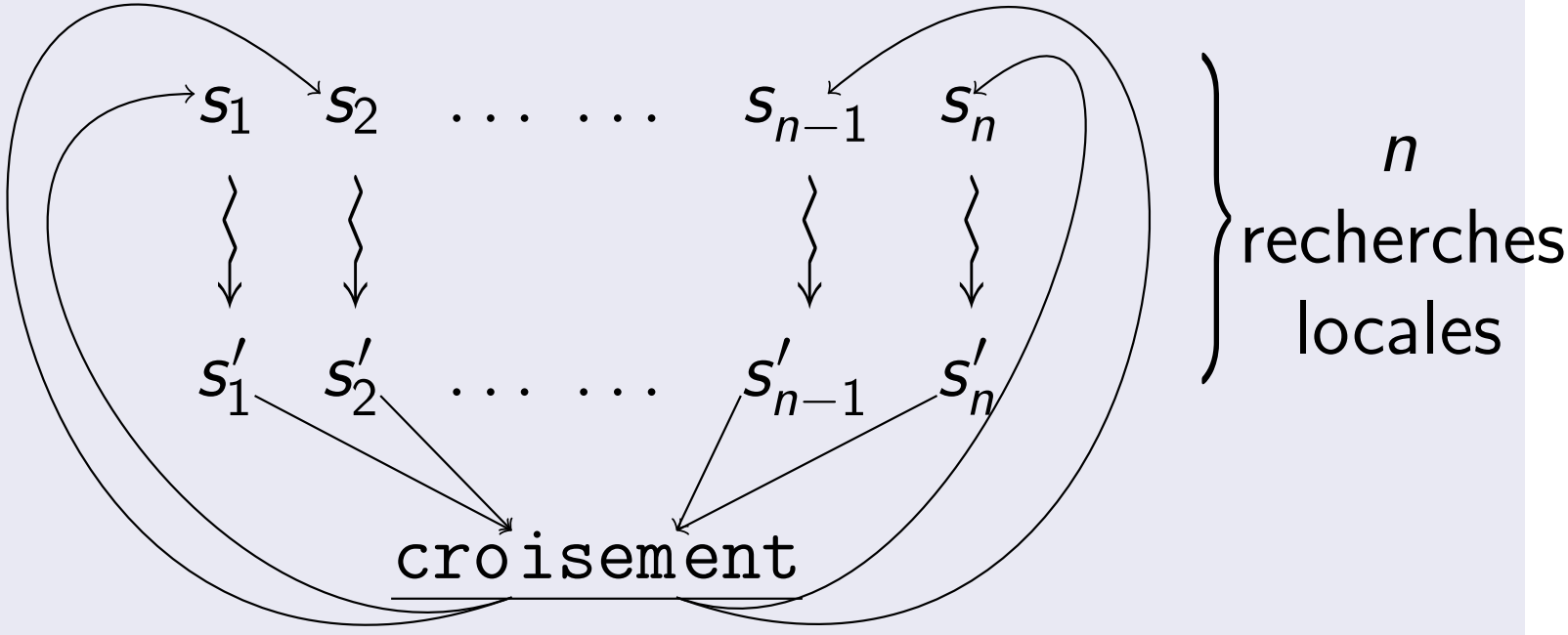


Question: Peut-on obtenir la même valeur ajoutée sans utiliser de population?

- ni fourmi, ni abeille, ni guêpe

Observation pratique

- recherche locale $\prec$ recherche locale $\oplus$ algo. génétique
- une recherche locale est souvent "boostée" par une approche mémétique:



Question: Peut-on obtenir la même valeur ajoutée sans utiliser de population?

- ni fourmi, ni abeille, ni guêpe

L'impact du croisement

Exemple:

$[1, 2, 3, 4, 5]$, $[2, 1, 5, 3, 4]$

$\rightarrow$

$[1, 2, 5, 3, 4]$

Valeur ajoutée aux recherches locales

- une nouvelle manière de générer des solutions
  - un bloc d'une solution peut être combiné avec un bloc d'une autre solution;
- solutions candidates plus dispersée dans l'espace

  $\Longrightarrow$ moins de blocages dans des optima locaux

Proposition

Recherche Locale Itérée

Repeat

$s \leftarrow$ `sol-voisine`(s)

if (optimum local)

  $s \leftarrow$ `perturbSimple`(s)

if (stagnation détectée)

  $s \leftarrow$ `perturbGenCol`(s)

Until condition arrêt

Perturbation Simple

- On insère au début de s un bloc (sous-permutation) stocké dans une archive de blocs de qualité
- ces blocs de qualité sont ajoutés à l'archive pendant la recherche

Perturbation par Génération de Colonnes

- On lance à partir de s quelques itérations de génération de colonnes

Proposition

Recherche Locale Itérée

Repeat

$s \leftarrow$ `sol-voisine`(s)

if (optimum local)

  $s \leftarrow$ `perturbSimple`(s)

if (stagnation détectée)

  $s \leftarrow$ `perturbGenCol`(s)

Until condition arrêt

Perturbation Simple

- On insère au début de s un bloc (sous-permutation) stocké dans une archive de blocs de qualité
- ces blocs de qualité sont ajoutés à l'archive pendant la recherche

Perturbation par Génération de Colonnes

- On lance à partir de s quelques itérations de génération de colonnes

Le modèle de génération de colonnes

Définition des colonnes (routes) pour le **Capacitated Arc Routing**

- $a_i = 1 \Leftrightarrow$ l'arrête i est servie par route $\mathbf{a}$ (vue comme un vecteur)
- c_a: coût du vecteur $\mathbf{a}$ (distance totale de la route $\mathbf{a}$)
- une route $\mathbf{a}$ satisfait une contrainte de capacité $\mathbf{w}^\top \mathbf{a} \leq C$

But: minimiser le coût total des colonnes sélectionnées

$$\min \sum_{\mathbf{a} \in \mathcal{R}} c_a x_a$$

$\mathcal{R}$: Ensemble de colonnes/routes

$$\sum_{\mathbf{a} \in \mathcal{R}} a_i x_a \geq 1, \quad \forall i \in [1..m]$$

Contraintes de couverture associées aux variables duales $\mathbf{y}$

$$x \in \{0,1\}^{|\mathcal{R}|}$$

Le modèle de génération de colonnes

Définition des colonnes (routes) pour le Capacitated Arc Routing

- $a_i = 1 \Leftrightarrow$ l'arrête i est servie par route $\mathbf{a}$ (vue comme un vecteur)
- c_a: coût du vecteur $\mathbf{a}$ (distance totale de la route $\mathbf{a}$)
- une route $\mathbf{a}$ satisfait une contrainte de capacité $\mathbf{w}^\top \mathbf{a} \leq C$

But: minimiser le coût total des colonnes sélectionnées

$$\min \sum_{\mathbf{a} \in \mathcal{R}} c_a x_a$$

$\mathcal{R}$: Ensemble de colonnes/routes

$$\sum_{\mathbf{a} \in \mathcal{R}} a_i x_a \geq 1, \quad \forall i \in [1..m]$$

$$x \in \{0,1\}^{|\mathcal{R}|}$$

Contraintes de couverture associées aux variables duales $\mathbf{y}$

Le dual:

$$\max y_1 + y_2 + \ldots y_m$$
$$\boxed{\mathbf{a}^\top \mathbf{y} \leq c_a}, \quad \forall a \in \mathcal{R}$$
$$y_i \geq 0, \qquad i \in [1..m]$$

Le sous-problème

À chaque itération, les valeur duales $\mathbf{y}$ sont connues:

- on cherche une route $\mathbf{a} \in \mathcal{R}$ qui minimise $a_c - \mathbf{a}^\top \mathbf{y}$
- si on trouve $\mathbf{a} \in K$ avec $a_c - \mathbf{a}^\top \mathbf{y} < 0$, on ajoute une nouvelle colonne (route). Sinon, on arrête.
- le pricing cherche une route/permutation, mais le modèle garde que le vecteur d'incidence.

Le dual:

$$\max y_1 + y_2 + \dots y_m$$
$$\boxed{\mathbf{a}^\top \mathbf{y} \leq c_a}, \quad \forall a \in \mathcal{R}$$
$$y_i \geq 0, \qquad i \in [1..m]$$

Le sous-problème

À chaque itération, les valeur duales $\mathbf{y}$ sont connues:

- on cherche une route $\mathbf{a} \in \mathcal{R}$ qui minimise $a_c - \mathbf{a}^\top \mathbf{y}$
- si on trouve $\mathbf{a} \in K$ avec $a_c - \mathbf{a}^\top \mathbf{y} < 0$, on ajoute une nouvelle colonne (route). Sinon, on arrête.
- le pricing cherche une route/permutation, mais le modèle garde que le vecteur d'incidence.

Génération de colonnes lancée à partir d'une solution primale

- La recherche locale trouve $[1, 2, 3, 4]$ de coût 64
  - route 1 de coût 46 (trois arêtes)
  - route 2 de coût 18 (une arrête)
- On génére le programme dual avec 4 variables positives

$$\max y_1 + y_2 + y_3 + y_4$$
$$y_1 + y_2 + y_3 \leq 46$$
$$y_4 \leq 18$$

- Simplex trouve $[y_1\ y_2\ y_3\ y_4] = [46\ 0\ 0\ 18]$ de coût 64

Génération de colonnes lancée à partir d'une solution primale

- La recherche locale trouve $[1, 2, 3, 4]$ de coût 64
  - route 1 de coût 46 (trois arêtes)
  - route 2 de coût 18 (une arrête)
- On génére le programme dual avec 4 variables positives

$$\max y_1 + y_2 + y_3 + y_4$$
$$y_1 + y_2 + y_3 \leq 46$$
$$y_4 \leq 18$$

- Simplex trouve $[y_1\ y_2\ y_3\ y_4] = [46\ 0\ 0\ 18]$ de coût 64

Génération de colonnes lancée à partir d'une solution primale

- La recherche locale trouve $[1, 2, 3, 4]$ de coût 64
  - route 1 de coût 46 (trois arêtes)
  - route 2 de coût 18 (une arrête)
- On génére le programme dual avec 4 variables positives

$$\max y_1 + y_2 + y_3 + y_4$$
$$y_1 + y_2 + y_3 \leq 46$$
$$y_4 \leq 18$$

- Simplex trouve $[y_1\ y_2\ y_3\ y_4] = [46\ 0\ 0\ 18]$ de coût 64

Si la génération de colonne fini dans l'état suivant:

$$\max y_1 + y_2 + y_3 + y_4$$
$$y_1 + y_2 + y_3 \leq 46$$
$$y_4 \leq 18$$
$$y_1 \leq 18$$
$$y_1 + y_3 \leq 20$$
$$y_2 + y_4 \leq 20$$

- Si les deux dernières contraintes ont une valeur master de 1, on renvoie la solution primale $[1, 3, 2, 4]$
- Si on obtient des variables master fractionnaires, on doit arrondir pour construire une solution primale entière.
  - À chaque itération, le pricing cherche une route de coût dual plus grand

Si la génération de colonne fini dans l'état suivant:

$$\begin{aligned} \max\, & y_1 + y_2 + y_3 + y_4 \\ & y_1 + y_2 + y_3 \leq 46 \\ & y_4 \leq 18 \\ & y_1 \leq 18 \\ & y_1 + y_3 \leq 20 \\ & y_2 + y_4 \leq 20 \end{aligned}$$

- Si les deux dernières contraintes ont une valeur master de 1, on renvoie la solution primale $[1, 3, 2, 4]$
- Si on obtient des variables master fractionnaires, on doit arrondir pour construire une solution primale entière.
- À chaque itération, le pricing cherche une route de coût dual plus grand

Si la génération de colonne fini dans l'état suivant:

$$\begin{aligned} \max\, & y_1 + y_2 + y_3 + y_4 \\ & y_1 + y_2 + y_3 \leq 46 \\ & y_4 \leq 18 \\ & y_1 \leq 18 \\ & y_1 + y_3 \leq 20 \\ & y_2 + y_4 \leq 20 \end{aligned}$$

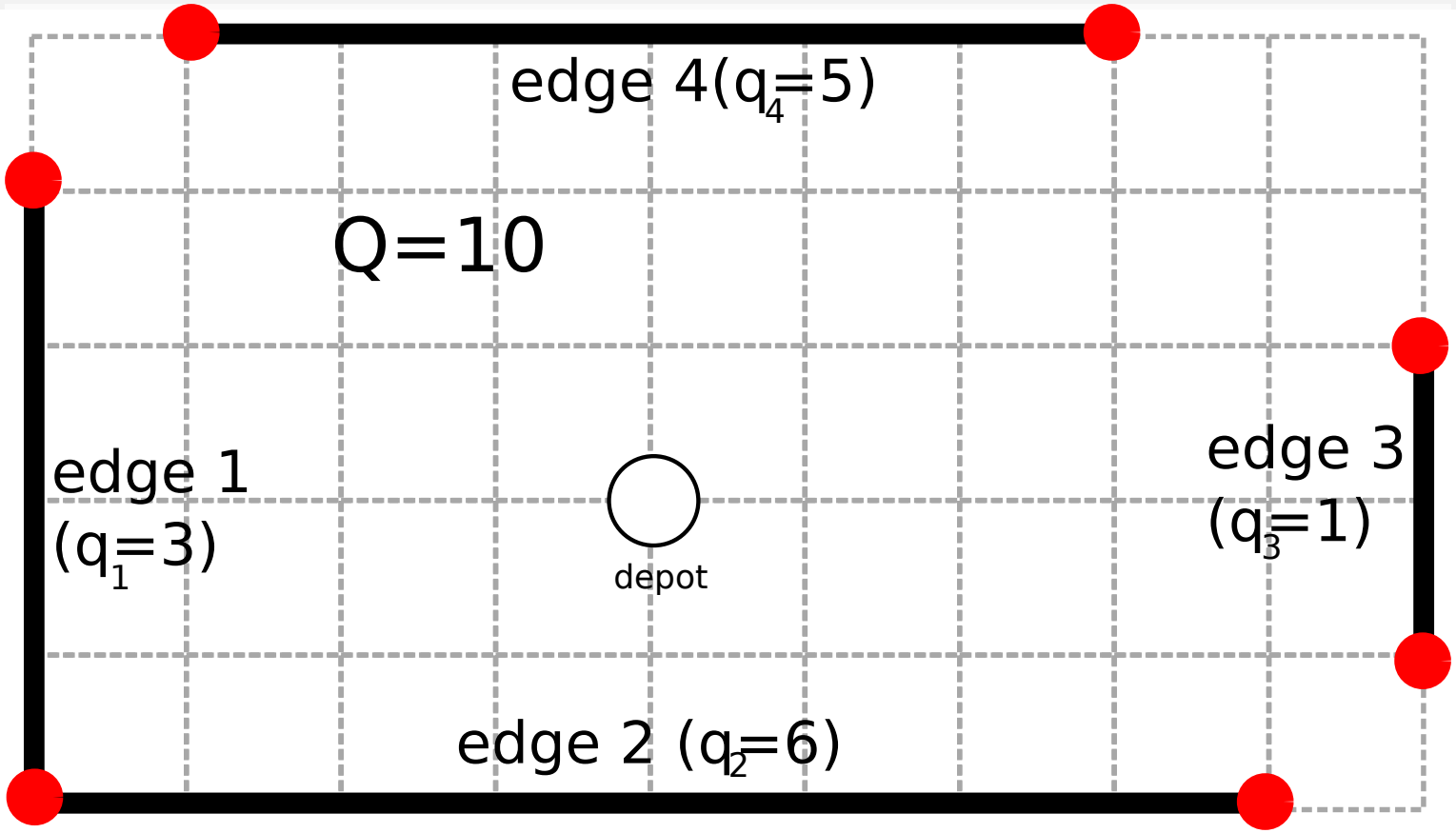
- Si les deux dernières contraintes ont une valeur master de 1, on renvoie la solution primale $[1, 3, 2, 4]$
- Si on obtient des variables master fractionnaires, on doit arrondir pour construire une solution primale entière.
- À chaque itération, le pricing cherche une route de coût dual plus grand

Si la génération de colonne fini dans l'état suivant:

$$\max y_1 + y_2 + y_3 + y_4$$
$$y_1 + y_2 + y_3 \leq 46$$
$$y_4 \leq 18$$
$$y_1 \leq 18$$
$$y_1 + y_3 \leq 20$$
$$y_2 + y_4 \leq 20$$

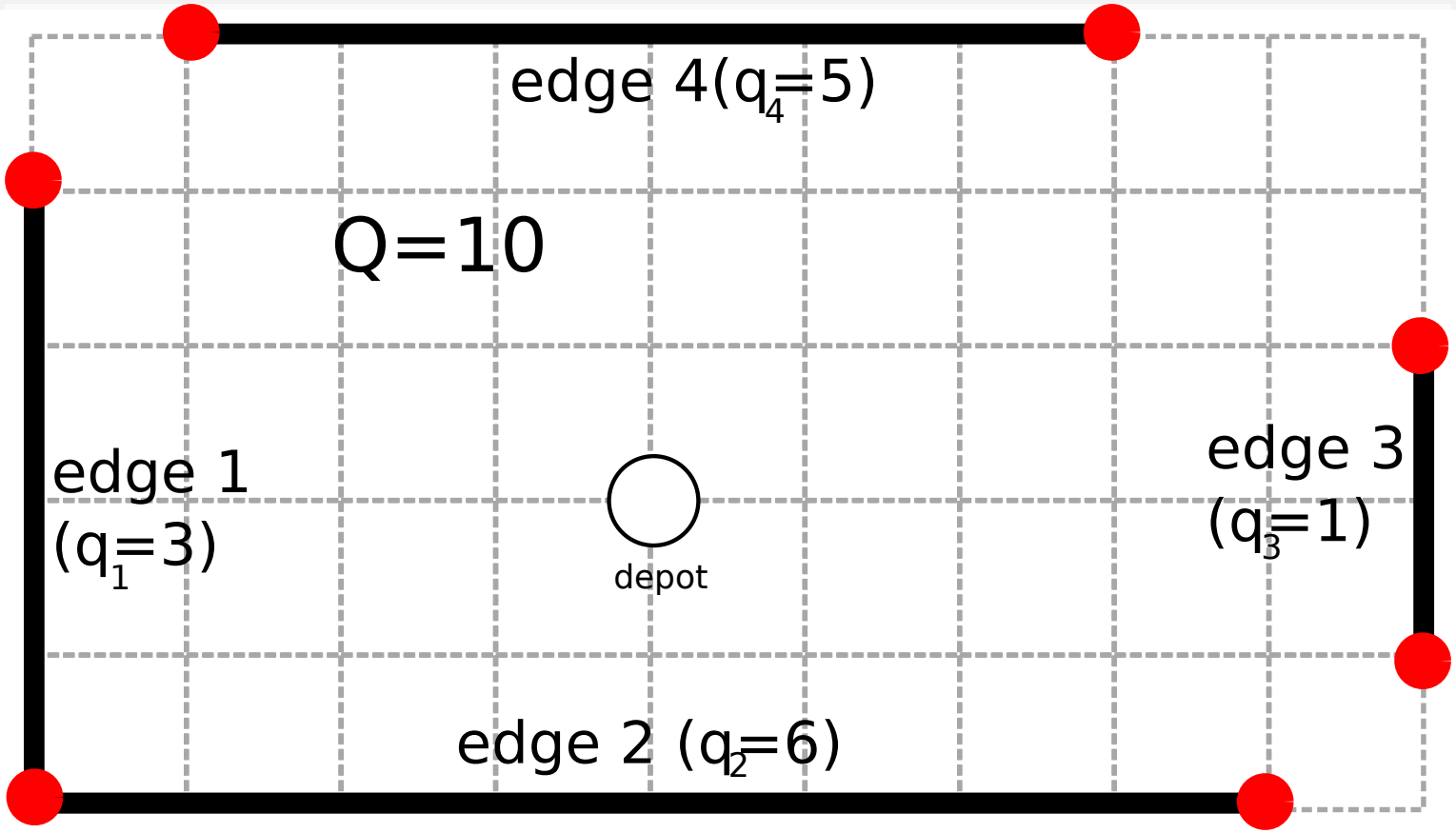
- Si les deux dernières contraintes ont une valeur master de 1, on renvoie la solution primale $[1, 3, 2, 4]$
- Si on obtient des variables master fractionnaires, on doit arrondir pour construire une solution primale entière.
- À chaque itération, le pricing cherche une route de coût dual plus grand

L'implémentation Arc-Routing



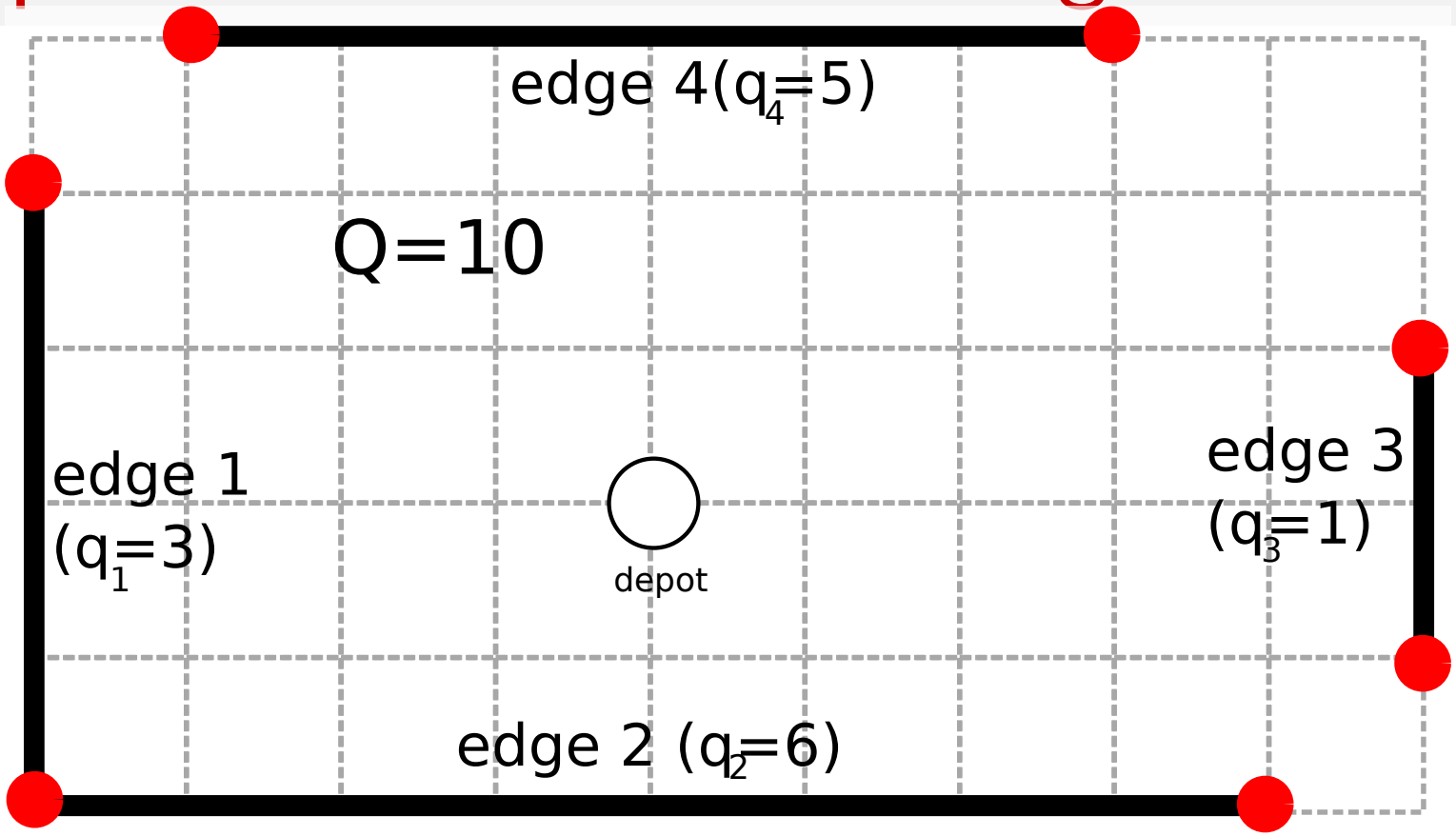
- 4 arrêtes à servir $\{1, 2, 3, 4\}$
  - demandes q_1, q_2, q_3, q_4
  - capacité 10
- un dépot et 7 sommets
- distance Manhatan

L'implémentation Arc-Routing



- 4 arrêtes à servir $\{1, 2, 3, 4\}$
  - demandes q_1, q_2, q_3, q_4
  - capacité 10
- un dépot et 7 sommets
- distance Manhatan

L'implémentation Arc-Routing



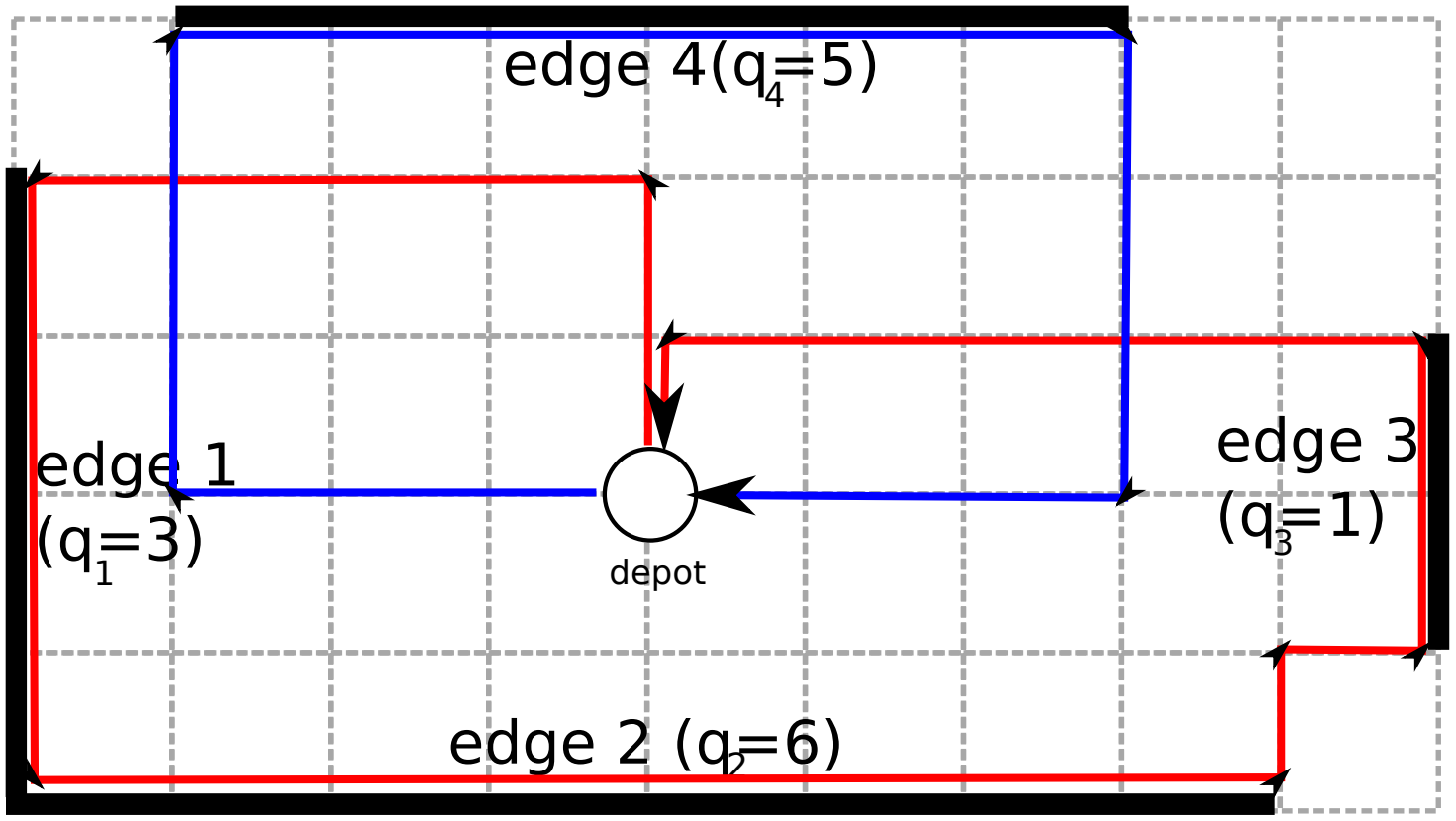
Comment coder une solution candidate

Une permutation de 4 arêtes:

example $[1, 2, 3, 4]$ veut dire "servir 1, puis 2, etc."

un décodeur trouve le plus petit coût pour faire cela

L'implementation d'Arc-Routing



- La solution primale [1, 2, 3, 4] comporte une route rouge et une route bleu $\Longrightarrow$ le coût total est 64
- Grâce à un décodeur, on peut résoudre le problème dans l'espace de permutations et blocs (routes)

Génération de colonnes et recherche locale

Valeurs ajoutées à la recherche locale par la génération de colonnes:

- une nouvelle manière de générer des routes grâce à des valeurs duales dans un modèle fractionnaire
- des routes diversifiées dans l'archive $\Longrightarrow$ moins de blocages dans des optima locaux
- si on tourne un processus de génération de colonnes jusqu'au bout, on obtient une borne duale!

Résultats **avec** ou **sans** génération de colonnes

Version d'algorithme	Benchmark	gdb	val	beullens	egl
Version **avec** génération de colonnes	**écart moyen** $(ub - opt)/opt$	**0.00%**	**0.28%**	**0.46%**	**0.79%**
	temps moyen rec. loc. [s]	13.00	89.59	138.81	1082.71
	écart moyen $(ub - lb)/ub$	4.63%	9.85%	6.99%	4.40%
Version **sans** génération de colonnes	**écart moyen** $(ub - opt)/opt$	**0.02%**	**1.48%**	**0.64%**	**1.15%**
	temps moyen rec. loc.[s]	38.96	103.76	149.56	1287.50

Trois nouvelles meilleurs bornes primales trouvées:

`egl-g1-C`:1274389, `egl-g2-A`:1115207, `egl-g2-C`:1376726

Conclusion

- La génération de colonnes peut *booster* la recherche locale, mais pas toujours plus que l'approche génétique
- L'approche génétique est plus facile à implémenter
- La génération de colonnes offre une borne duale!