Édition 2021

KI022 G. Dalle J.G. Fages B. Houzel A. Parmentier F. Ramond

1 Le mot de la SNCF

Le groupe SNCF est l'un des leaders mondiaux de la mobilité. Il est confronté chaque jour à de multiples problématiques industrielles, et les choix effectués ont des répercussions directes et importantes sur les coûts de production, les recettes et la qualité du service fourni à ses clients. Ces décisions concernent plusieurs horizons de temps, qui vont de la conception plusieurs mois ou années avant les circulations (horaires des trains, rotations des rames, planification des journées de service des agents, ...) jusqu'à la gestion opérationnelle le jour J (régulation du trafic, adaptations de l'offre en cas de perturbations, ...). Il est nécessaire de prendre les meilleures décisions pour optimiser les performances et offrir à nos clients le meilleur service au meilleur coût, d'où l'importance de la Recherche Opérationnelle. La conception des Graphiques d'Occupation des Voies, qui fait l'objet de ce challenge et qui est décrite dans les sections suivantes, en est un parfait exemple : on souhaite pouvoir accueillir tous les trains dans les gares tout en garantissant une bonne robustesse le jour J en cas de petits retards. Cette robustesse sera assurée par des incompatibilités entre les itinéraires utilisés à des horaires trop proches.

Nous espérons que ce challenge vous donnera un bon aperçu du type de problèmes auxquels est confrontée la SNCF et que vous prendrez plaisir à y répondre avec vos algorithmes les plus performants.

Bon challenge !

2 Problématique métier

Un train qui circule sur le réseau est sur ce que l'on appelle une *voie en ligne*, tandis qu'un train arrêté en gare est sur ce que l'on appelle une *voie à quai*. Lorsqu'un train arrive en gare, il doit passer d'une voie en ligne à une voie à quai. Et lorsqu'un train repart, il doit passer d'une voie à quai à une voie en ligne. Plusieurs itinéraires sont possibles pour passer d'une voie en ligne à une voie à quai. Comme deux trains ne peuvent pas passer sur la même voie au même moment, certains itinéraires sont incompatibles. L'objectif de ce challenge est d'optimiser la gestion des trains en gare. Plus précisément, pour un ensemble de trains dont on connait les voies en ligne, on souhaite choisir à quelle voie à

quai sera affecté chaque train, et quels itinéraires ils empruntent en arrivant en gare et en repartant de la gare, de manière à minimiser les incompatibilités.

3 Modélisation du problème

Une gare est composée d'un ensemble de voies en lignes $\mathcal{L}$ et de voies à quai Q.

Itinéraires Un *itinéraire d'arrivée* en gare i permet de rejoindre une voie à quai q_i depuis une voie en ligne ℓ_i. Un *itinéraire de départ* de la gare i permet de rejoindre une voie en ligne ℓ_i depuis une voie à quai q_i. En pratique, pour rejoindre une voie depuis une autre, il y a généralement beaucoup d'options possibles passant par différents tronçons intermédiaires. On note $\mathcal{I}^{\mathrm{a}}$ l'ensemble des itinéraires d'arrivée, $\mathcal{I}^{\mathrm{d}}$ l'ensemble des itinéraires de départ, et $\mathcal{I} = \mathcal{I}^{\mathrm{a}} \cup \mathcal{I}^{\mathrm{d}}$ l'ensemble des itinéraires.

Trains. Sur la période considérée, un ensemble $\mathcal{T}$ de *trains* t arrivent et repartent de la gare. Ce que l'on appelle train n'est pas le véhicule, mais la mission entre une origine et une destination. Le véhicule en tant qu'objet physique est appelé rame. L'ensemble des trains qui arrivent en gare est noté $\mathcal{T}^{\mathrm{a}}$ et l'ensemble des trains qui repartent de la gare est noté $\mathcal{T}^{\mathrm{d}}$. À chaque train t correspondent une voie en ligne $\ell_t \in \mathcal{L}$, une voie à quai $q_t \in \mathcal{Q}$, et un itinéraire $i_t \in I$. Si t part de la gare, i.e. $t \in \mathcal{T}^{\mathrm{d}}$, il rejoint ℓ_t depuis q_t en passant par $i_t \in \mathcal{I}^{\mathrm{d}}$. S'il arrive en gare, i.e. $t \in \mathcal{T}^{\mathrm{a}}$, il rejoint q_t depuis ℓ_t en passant par $i_t \in \mathcal{I}^{\mathrm{a}}$.

La voie en ligne ℓ_t est une donnée du problème, tandis que la voie à quai q_t et l'itinéraire i_t sont des variables du problème. À titre indicatif, une voie à quai est suggérée pour chaque train de l'instance, mais celle-ci pourra être changée pour obtenir un meilleur objectif. Rien ne garantit d'ailleurs que ces suggestions combinées fournissent une solution réalisable.

Pour faciliter le problème, on suppose qu'un train n'est pas nécessairement affecté à un quai (autrement dit, l'algorithme peut laisser le soin aux opérateurs de l'insérer manuellement par la suite). Pour indiquer qu'un train n'est pas affecté, on rajoute un quai fictif $q^{\emptyset}$ et un itinéraire fictif $i^{\emptyset}$. Ne pas affecter un train coûte $c^{\emptyset}$.

Résoudre le problème consiste à choisir le quai et l'itinéraire de chaque train. On note

$$\mathcal{S} = \left(\mathcal{Q} \cup \{q^{\emptyset}\}, \mathcal{I}^{\mathrm{d}} \cup \{i^{\emptyset}\}\right)^{\mathcal{T}^{\mathrm{d}}} \times \left(\mathcal{Q} \cup \{q^{\emptyset}\}, \mathcal{I}^{\mathrm{a}} \cup \{i^{\emptyset}\}\right)^{\mathcal{T}^{\mathrm{a}}}$$

l'ensemble des solutions du problème. Une solution du problème est donc un vecteur

$$\boldsymbol{s} \in \mathcal{S} \quad \text{avec} \quad \boldsymbol{s} = (q_t, i_t)_{t \in \mathcal{T}} \quad \text{tel que} \quad q_t \in \mathcal{Q} \cup \{q^{\emptyset}\} \quad \text{et} \quad \begin{cases} i_t \in \mathcal{I}^{\mathrm{d}} \cup \{i^{\emptyset}\} & \text{si } t \in \mathcal{T}^{\mathrm{d}}, \\ i_t \in \mathcal{I}^{\mathrm{a}} \cup \{i^{\emptyset}\} & \text{si } t \in \mathcal{T}^{\mathrm{a}}. \end{cases}$$

Une solution doit satisfaire certaines contraintes pour être réalisable. Tout d'abord, il faut que l'itinéraire emprunté par un train coïncide avec sa voie en ligne et sa voie à quai.

Pour tout $t \in \mathcal{T}$, on a la contrainte

$$\left\{ \begin{array}{lll} & q_t & = q^\emptyset \\ \text{et} & i_t & = i^\emptyset \end{array} \right. \quad \text{ou} \quad \left\{ \begin{array}{lll} & q_t & = q_{i_t} \\ \text{et} & \ell_t & = \ell_{i_t}. \end{array} \right. \tag{1}$$

On note $\mathbb{1}_{q^\emptyset} : \mathcal{Q} \cup \{q^\emptyset\} \to \{0,1\}$ la fonction indicatrice de $q^\emptyset$, définie par

$$\mathbb{1}_{q^\emptyset}(q) = \begin{cases} 1 & \text{if } q = q^\emptyset \\ 0 & \text{sinon.} \end{cases}$$

Groupes de trains et voies à quai. L'ensemble des trains $\mathcal{T}$ est partitionné en groupes de trains G. On note $\mathcal{G}$ l'ensemble des groupes :

$$\mathcal{T} = \bigcup_{G \in \mathcal{G}} G \quad \text{et} \quad G \neq G' \implies G \cap G' = \emptyset$$

L'affectation des voies à quai aux trains doit satisfaire la contrainte suivante : *tous les trains d'un même groupe doivent être affectés à la même voie à quai*

$$q_t = q_{t'}, \quad \forall (t,t') \in G^2, \forall G \in \mathcal{G}. \tag{2}$$

En pratique, un groupe de train contient le plus souvent deux trains : l'un qui correspond à l'arrivée de la rame en gare et l'autre au départ de la même rame depuis la gare. Il peut également contenir un seul train, si la rame qui réalise un train ne repart pas sur un autre mission. De manière générale, un groupe G peut contenir jusqu'à cinq trains si plusieurs rames venant de destination différentes sont assemblées (ou désassemblées) en gare avant de repartir pour leur mission suivante.

Types de circulation et types de matériels Les trains qui passent dans la gare peuvent correspondre à différents types de circulation (TGV, TER, etc.) et à différents types de matériels (caractérisant la locomotive, les wagons, les rames, etc. qui composent le train). On note respectivement $\mathcal{E}$ et $\mathcal{M}$ l'ensemble des types de circulations et de matériels. Chaque train t appartient à un type de circulation $e_t \in \mathcal{E}$ et possède un ensemble de types de matériels $M_t \subseteq \mathcal{M}$. Par exemple, un train peut-être composé de plusieurs unités aux caractéristiques légèrement différentes. Ces types sont des données du problème.

Certains matériels ou certaines circulations sont incompatibles avec certains quais, pour des raisons techniques (e.g. le mode d'alimentation électrique) ou pour des raisons de service (e.g. pour gérer le flot des passagers). Une *contrainte de quai* est notée f, et on note $\mathcal{F}$ l'ensemble des *contraintes de quais*. Chaque contrainte f de F concerne un ensemble de quais $\mathcal{Q}_f \subsetneq \mathcal{Q}$, un ensemble de voies en ligne $\mathcal{L}_f$, un ensemble de type de circulation $\mathcal{E}_f$ et un ensemble de types de matériels $\mathcal{M}_f$. La contrainte s'exprime de la manière suivante : un train t tel que $\ell_t \in \mathcal{L}_f$, $M_t \cap \mathcal{M}_f \neq \emptyset$ ou $e_t \in \mathcal{E}_f$ ne peut être assigné à un quai $q \in \mathcal{Q}_f$.

$$\left. \begin{array}{ll} & \ell_t \in \mathcal{L}_f \\ \text{ou} & M_t \cap \mathcal{M}_f \neq \emptyset \\ \text{ou} & e_t \in \mathcal{E}_f \end{array} \right\} \implies q_t \notin \mathcal{Q}_f, \quad \forall f \in \mathcal{F} \tag{3}$$

Itinéraires incompatibles Comme certains couples d'itinéraires traversent les mêmes tronçons de voie. Si deux trains empruntent ces itinéraires au même moment, ils vont se gêner mutuellement, ce qui risque de créer des retards car l'un des deux devra patienter. Ces incompatibilités sont indexées par j, et on note $\mathcal{J}$ l'ensemble des incompatibilités. Pour chaque incompatibilité j, on a un quintuplet $(t_{j,1}, i_{j,1}, t_{j,2}, i_{j,2}, c_j)$ qui est interprété de la manière suivante :

$$\text{si } t_{j,1} \text{ emprunte } i_{j,1} \text{ et } t_{j,2} \text{ emprunte } i_{j,2} \text{ alors on paie un coût } c_j.$$

Étant données une solution $\boldsymbol{s} = (q_t, i_t)_{t\in\mathcal{T}} \in \mathcal{S}$ et une incompatibilité $j \in \mathcal{J}$, on définit la fonction $\mathbb{1}_j : \mathcal{S} \to \{0, 1\}$ par

$$\mathbb{1}_j(\boldsymbol{s}) = \begin{cases} 1 & \text{si } i_{t_{j,1}} = i_{j,1} \text{ et } i_{t_{j,2}} = i_{j,2} \\ 0 & \text{sinon.} \end{cases}$$

C'est aussi via ces contraintes que l'on modélise le fait que deux trains ne doivent pas utiliser le même quai au même moment (rappelons que l'itinéraire d'un train détermine en particulier sa voie à quai).

Informations additionnelles sur les trains. Les incompatibilités sont causées par des trains qui empruntent de façon rapprochée des itinéraires qui s'intersectent, créant ainsi un conflit pour l'usage de l'infrastructure. À titre indicatif, on fournit donc pour chaque train t la date et l'heure d_t du départ (si le train entre en gare) ou de l'arrivée (si le train quitte la gare). Ces informations ne sont pas nécessaires pour formuler le problème, mais peuvent éventuellement être utilisées pour guider les algorithmes.

Position du problème. Le problème peut donc être résumé par

$$\min_{\boldsymbol{s}\in\mathcal{S}} \sum_{t\in\mathcal{T}} c^{\emptyset} \mathbb{1}_{q^{\emptyset}}(q_t) + \sum_{j\in\mathcal{J}} c_j \mathbb{1}_j(\boldsymbol{s})$$

tel que $\boldsymbol{s}$ satisfait les contraintes (1), (2) et (3)

4 Format des instances et des solutions

Les instances sont fournies au format `.json`. Pour toutes les instances, le coût $c^{\emptyset}$ vaut 2000. Voici un exemple ci-dessous.

```
{
  "trains": [
    [
      {
        "id": 0,
        "sensDepart": true,
        "voieEnLigne": "V2_N",
        "voieAQuai": "V2",
```

```
        "typeCirculation": "FRET",
        "dateHeure": "2021-03-30T20:35",
        "typesMateriels": ["1"]
      }
    ],
    [
      {
        "id": 1,
        "sensDepart": false,
        "voieEnLigne": "VU2_S",
        "voieAQuai": "11",
        "typeCirculation": "TER",
        "dateHeure": "2021-03-30T21:36",
        "typesMateriels": ["0"]
      }
    ]
  ],
  "itineraires": [
    {
      "id": 0,
      "sensDepart": true,
      "voieEnLigne": "VE_N",
      "voieAQuai": "V2"
    },
    {
      "id": 1,
      "sensDepart": false,
      "voieEnLigne": "VU2_S",
      "voieAQuai": "11"
    },
    {
      "id": 2,
      "sensDepart": false,
      "voieEnLigne": "VU2_S",
      "voieAQuai": "11"
    }
  ],
  "voiesAQuai": ["V2", "V4", "11"],
  "voiesEnLigne": ["V2_N", "VU2_S"],
  "interdictionsQuais": [
    {
      "voiesAQuaiInterdites": ["V2", "V4"],
      "voiesEnLigne": [],
      "typesMateriels": ["0", "2"],
      "typesCirculation": []
    },
    {
      "voiesAQuaiInterdites": ["11"],
      "voiesEnLigne": ["VE_N"],
```

```
58        "typesMateriels": [],
59        "typesCirculation": []
60      }
61    ],
62    "contraintes": [
63      [0, 0, 1, 2, 10000],
64      [0, 0, 1, 1, 15]
65    ]
66  }
```

- L'attribut `trains` contient les *groupes de trains* de $\mathcal{T}$. C'est donc une liste de groupes de trains. Chaque groupe de trains contient la liste des trains dans le groupe. Chaque train possède les attributs suivants :
  - `id` contient l'id de t (un entier ≥ 0)
  - `sensDepart` contient un booléen égal à `true` si $t \in \mathcal{T}^{\mathrm{d}}$ et `false` si $t \in \mathcal{T}^{\mathrm{a}}$
  - `voieEnLigne` contient ℓ_t
  - `voieAQuai` contient une proposition pour q_t (comme q_t est une variable, celle-ci peut bien entendu être remise en cause)
  - `typeCirculation` contient e_t
  - `typesMateriels` contient la liste M_t
  - `dateHeure` contient d_t sous forme de `string`
- L'attribut `itineraires` contient les itinéraires de $\mathcal{I}$. Pour chacun d'eux
  - `id` contient l'id de i (un entier ≥ 0)
  - `sensDepart` un booléen égal à `true` si $i \in \mathcal{I}^{\mathrm{d}}$ et `false` si $i \in \mathcal{I}^{\mathrm{a}}$
  - `voieEnLigne` contient ℓ_i
  - `voieAQuai` contient q_i.
- L'attribut `voiesAQuai` contient la liste des voies à quai.
- L'attribut `voiesEnLigne` contient la liste des voies en ligne.
- L'attribut `interdictionsQuais` contient les contraintes de quais de $\mathcal{F}$. Pour chacune,
  - `voiesAQuaiInterdites` contient la liste $\mathcal{Q}_f$
  - `voiesEnLigne` contient la liste $\mathcal{L}_f$
  - `typesMateriels` contient la liste $\mathcal{M}_f$
  - `typesCirculation` contient la liste $\mathcal{E}_f$
- L'attribut `contraintes` contient les itinéraires incompatibles de $\mathcal{J}$. C'est une liste dont chaque élément correspond à un j et vient sous la forme d'un quintuplet d'entiers `[a,b,c,d,e]` où

- `a` contient l'`id` de $t_{1,j}$
- `b` contient l'`id` de $i_{1,j}$
- `c` contient l'`id` de $t_{2,j}$
- `d` contient l'`id` de $i_{2,j}$
- `e` contient c_j

Les solutions doivent être retournées au format `.json`. Ce fichier doit contenir pour chaque train t de $\mathcal{T}$ un attribut (au format `string`) correspondant à l'`id` de t (pensez à convertir du format `int` au format `string`). Et sous cet attribut

- Un attribut `voieAQuai` (au format `string`) contenant q_t
- Un attribut `itineraire` (au format `string`) contenant i_t (là aussi, pensez à convertir du format `int` au format `string`)

Si un train n'est pas affecté, les deux attributs `voieAQuai` et `itineraire` doivent être égaux à `"notAffected"`.

Voici un premier exemple ou aucun train n'est affecté – une telle solution est toujours possible, mais généralement avec un coût très mauvais.

```
{
  "0": {
    "voieAQuai": "notAffected",
    "itineraire": "notAffected"
  },
  "1": {
    "voieAQuai": "notAffected",
    "itineraire": "notAffected"
  }
}
```

Voici un second exemple plus naturel avec un coût de 15.

```
{
  "0": {
    "voieAQuai": "V2",
    "itineraire": "0"
  },
  "1": {
    "voieAQuai": "11",
    "itineraire": "1"
  }
}
```

5 Comment sera établi le classement

Cinq instances vous sont fournies :

- `A.json`
- `NS.json`
- `NE.json`
- `PMP.json`
- `PE.json`

Le score d'une équipe est la somme des coûts des solutions proposées pour chaque instance, sans normalisation (autrement dit, les grosses instances pèseront plus lourd et c'est normal). L'équipe avec le meilleur score, c'est-à-dire le score le plus petit, gagne.

6 Quelques conseils

Six heures, c'est très court. L'équipe qui gagnera sera celle qui arrivera à produire le plus vite des solutions décentes.

1. Partagez-vous les tâches
2. Commencez immédiatement à coder les outils pour parser un fichier d'entrée et créer un fichier de sortie
3. Faites simple : il n'est pas difficile de construire une solution admissible. Commencez par coder des méthodes simples vous donnant des solutions plutôt bonnes et évitez de faire un algorithme très puissant que vous n'arriverez pas à coder en 6 heures.