Etude Expérimentale: Interface pour le problème du bin-packing

Maxime ROBACHE

Université de Technologie de Compiègne

Mardi 21 Mars 2006

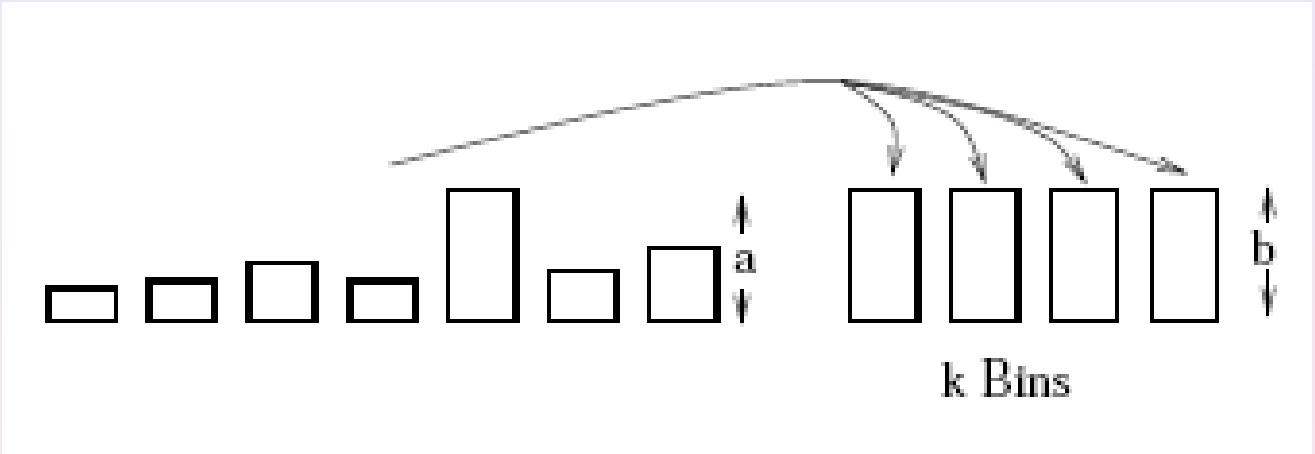
Plan

Plan

Plan

Plan

Bin-packing: Le bin-packing unidimensionnel



Formulation mathématique:

$$\exists ? f : [1, n] \rightarrow [1, k] \text{ tq } \forall j, \sum_{f(i)=j} a_i \leqslant b$$

Bin-packing: Le bin-packing unidimensionnel

Les applications

- Copie de fichiers sur un nombre minimum de supports
- Planification de tâches sur des processeurs
- Découpage de rectangles dans des plaques de bois ou d'autres matériaux
- Rangement avec contraintes de poids

Plan

Bin-packing: Le bin-packing en deux dimensions

Caractéristiques:

- Généralisation du problème 1D
- Peut-être décomposé en deux problèmes 1D

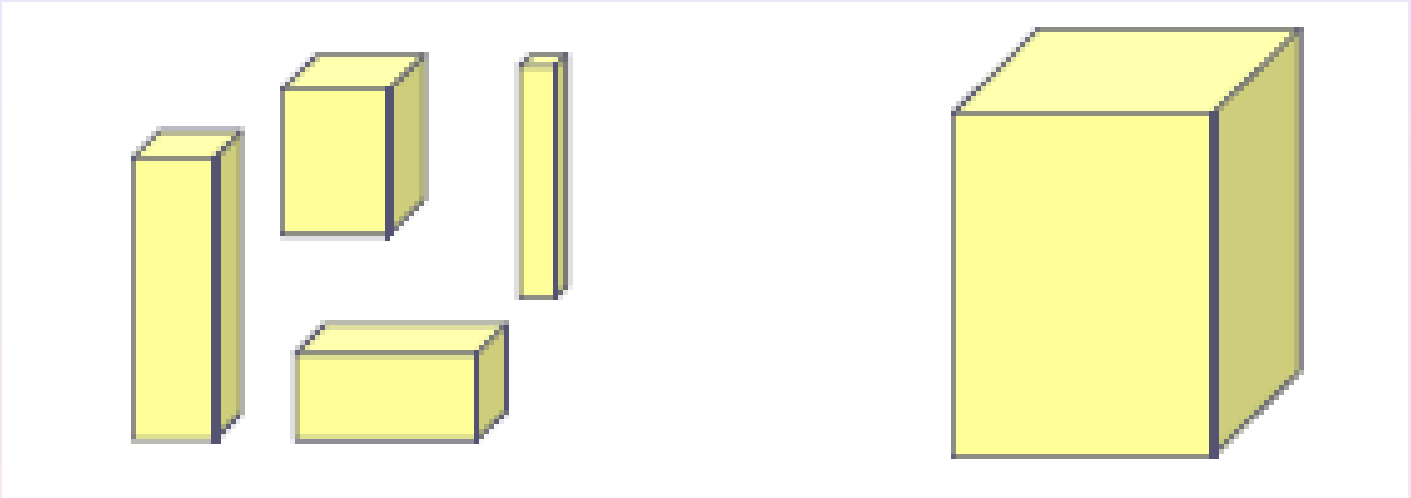
Bin-packing: Le bin-packing en deux dimensions

Les applications

- Problèmes d'optimisation de découpe dans l'industrie sur divers matériaux:
  - bois
  - tissus
  - métaux
  - etc...
- Planification de tâches avec contraintes de ressources

Plan

Bin-packing: Le bin-packing en trois dimensions



Caractéristiques:

- Généralisation du problème 2D
- Nombre donné d'objets tridimensionnels
- Bins tridimensionnels

Bin-packing: Le bin-packing en trois dimension

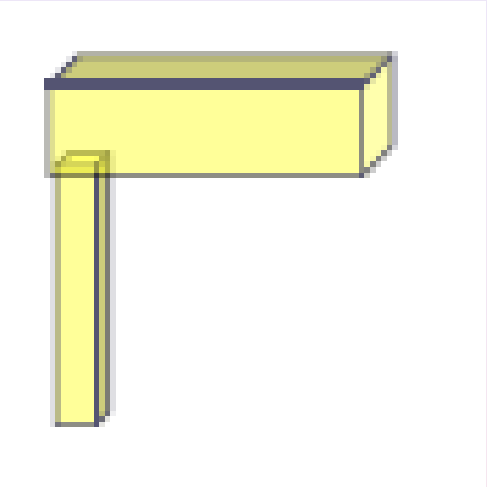
Les applications

- chargement de camions avec contraintes de volume
- remplissage de containers
- agencements des stocks
- remplissage de palettes

Les contraintes pratiques

- Les contraintes d'équilibre
- Rotation des objets
- Façon dont arrivent les objets

Bin-packing: Le bin-packing en trois dimension



Détails sur la façon dont arrivent les objets

- Les uns derrières les autres
- Tous disponibles: prétraitement

Plan

Pourquoi les heuristiques ?

Le Problème

Résolution exacte est un problème NP-difficile

La Solution

Utiliser des heuristiques: Règles empiriques

- simples et rapides
- analyse de situations
- objectif de résolution de problèmes
- domaine précis

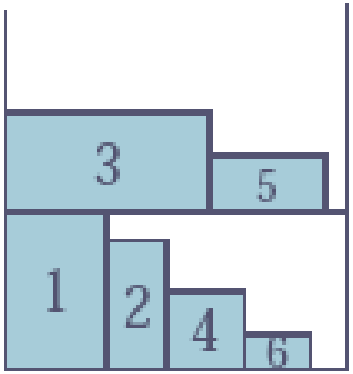
Offrent une résolution N-difficile du problème

Plan

Les heuristiques simples: Level Algorithms

Principe

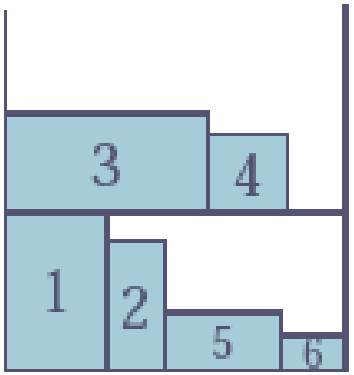
- (prétraitement des objets)
- Placement des objets un par un



Première méthode: First-Fit

Les objets sont placés dans le premier bin qui peut les acceuillir.

Les heuristiques simples: Level Algorithms



Deuxième méthode: Best-Fit

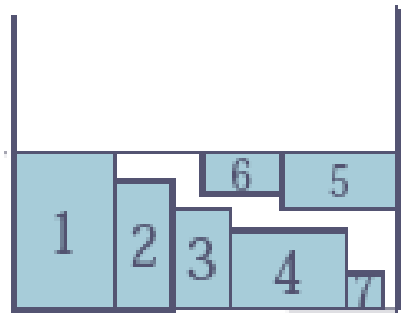
Les objets sont placés dans le bin qu'ils remplissent le mieux.

Plan

Une autre approche: le floor-ceiling

Détails

- LODI, 1999
- reprend FF et BF



Principe

- Floor-Ceiling = Sol-Plafond
- utiliser des espaces libres dans les bins

Plan

L'heuristique de BOSCHETTI et MINGOZZI

Détails

- BOSCHETTI et MINGOZZI, 2003
- possibilité de rotation des objets
- 2 étapes principales

Etape 1: tri des objets

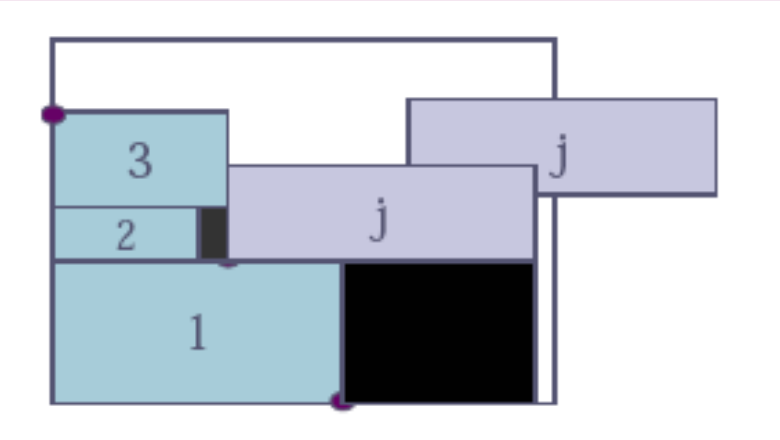
par:

- Surface décroissante
- Largeur décroissante
- Hauteur décroissante
- Périmètre décroissant

L'heuristique de BOSCHETTI et MINGOZZI

Etape 2: Remplissage bin par bin

- Recherche des positions envisageables
- Placement



Positions envisageables

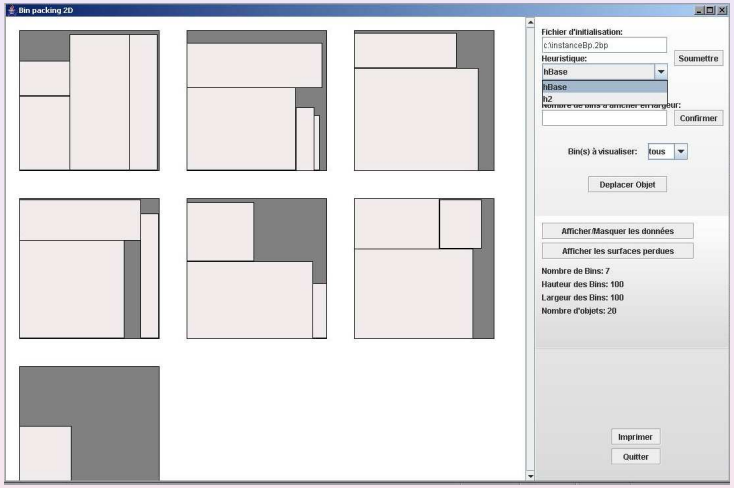
- pas de chevauchement
- pas de dépassement

Meilleure position

- la plus à gauche
- la plus basse

Plan

Pourquoi une interface ?



Objectifs

- visualisation immédiate
- validité et cohérence
- présentation

Definition

- **Instance:** Composée de données constituant un point de départ possible du problème. Dans le cas du bin-packing, il s'agit, du nombre de bins, d'objets, leurs dimensions etc...

Plan

Bin-Packing et Java

La classe ObjetSolution

- stockés dans un tableau

```
1  public class ObjetSolution
2  {
3      int noBin;
4      int posx;
5      int posy;
6      int width;
7      int height;
8      int verif;
9  }
```

Plan

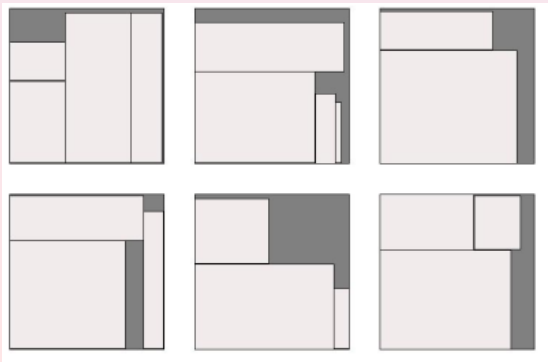
La représentation de la solution

Principe général

- D'après le tableau d'objets solution
- Simple boucle for

Les classes Java de dessin

- **fillRect()** qui dessine un rectangle plein
- **drawRect()** qui dessine un rectangle vide



Dessin

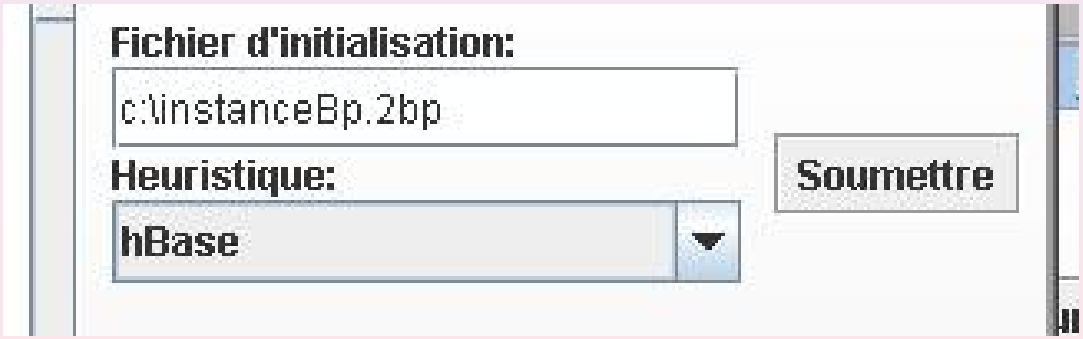
- Bins
- Objets

Plan

Choix de l'heuristique

Intêret

- Comparaison validité
- Comparaison efficacité



Détails de la solution

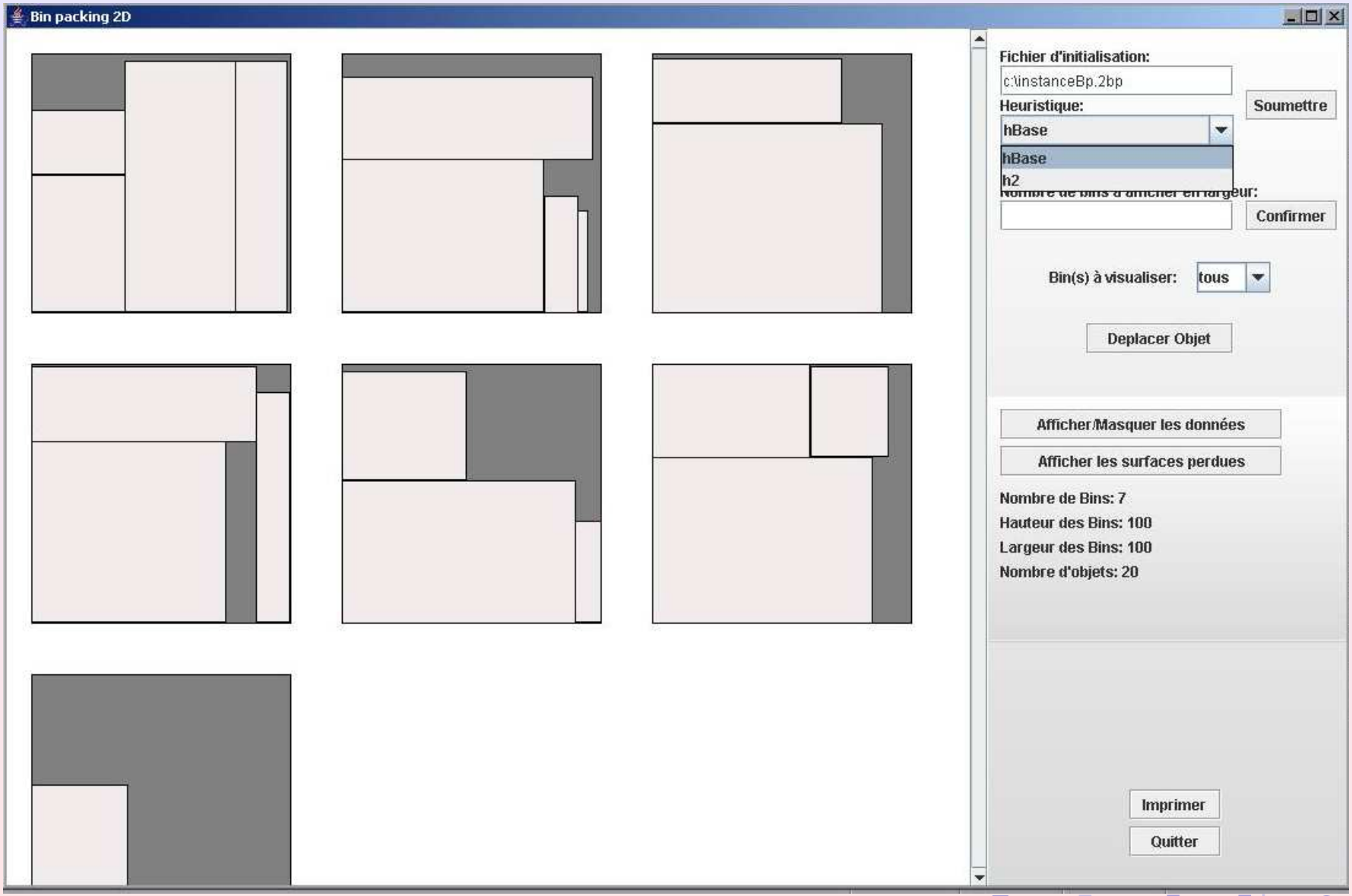
Détails affichables

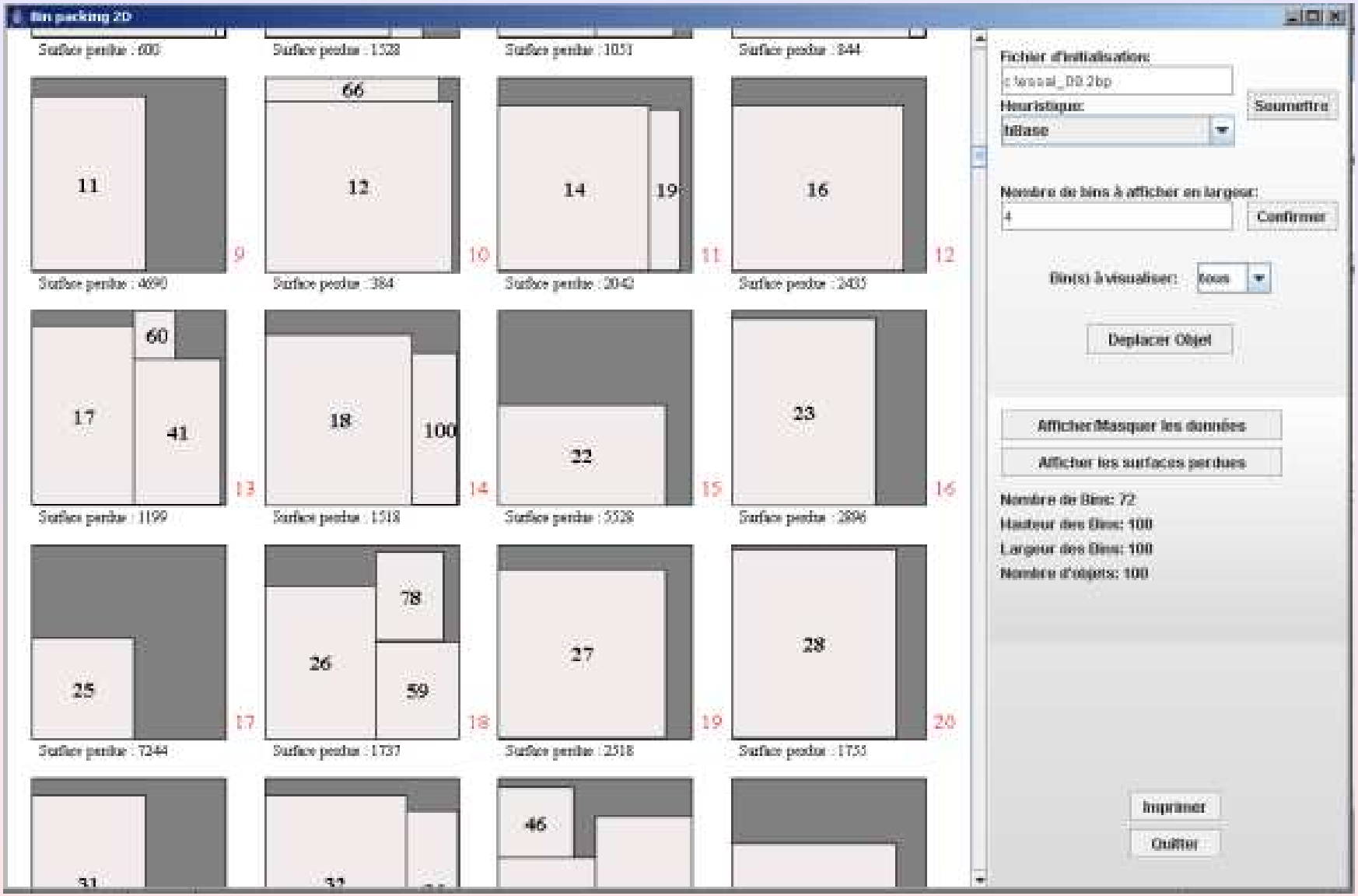
- Numéros d'objets et de bins
- Surfaces perdues
- positions des objets
- Résumé sur clic

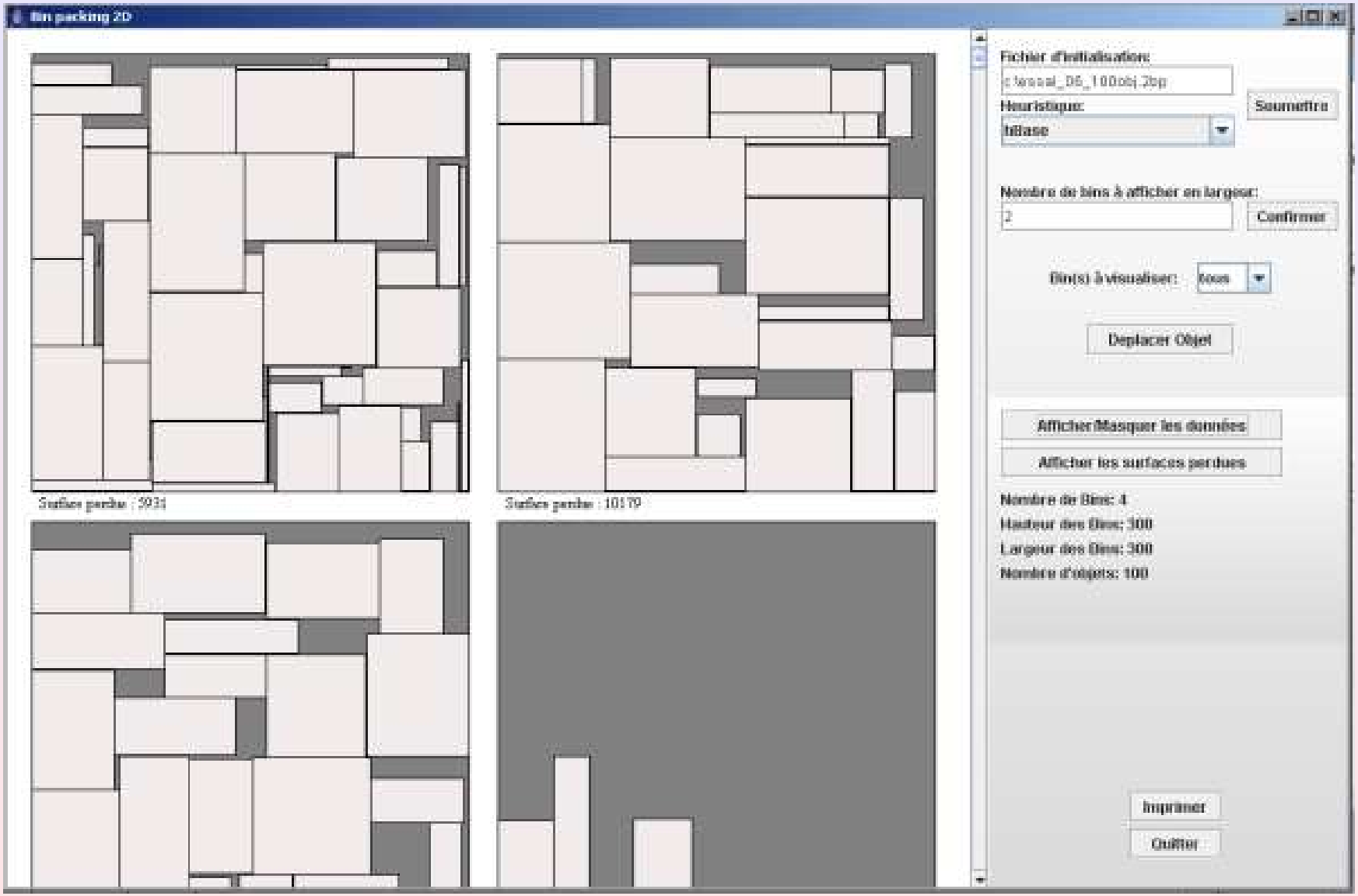
Déplacement des objets

Caractéristiques

- Interrogation de l'utilisateur
- Vérification de la validité de la nouvelle position
- déplacement ou message d'erreur


Bin packing 2D
Fichier d'initialisation:
c:\instanceBp.2bp
Soumettre
Heuristique:
hBase
hBase
h2
Confirmer
Bin(s) à visualiser:
tous
Deplacer Objet
Afficher/Masquer les données
Afficher les surfaces perdues
Nombre de Bins: 7
Hauteur des Bins: 100
Largeur des Bins: 100
Nombre d'objets: 20
Imprimer
Quitter


bin packing 2D
Surface perdue : 600
Surface perdue : 1528
Surface perdue : 1051
Surface perdue : 844
11
66
12
14
19
16
9
10
11
12
Surface perdue : 4690
Surface perdue : 384
Surface perdue : 2042
Surface perdue : 2403
17
60
41
18
100
22
23
13
14
15
16
Surface perdue : 1199
Surface perdue : 1518
Surface perdue : 5328
Surface perdue : 2896
25
78
26
59
27
28
17
18
19
20
Surface perdue : 7244
Surface perdue : 1737
Surface perdue : 2518
Surface perdue : 1733
46
Fichier d'initialisation:
Soumettre
Heuristique:
hBase
Nombre de bins à afficher en largeur:
4
Confirmer
Bin(s) à visualiser:
tous
Deplacer Objet
Afficher/Masquer les données
Afficher les surfaces perdues
Nombre de Bins: 72
Hauteur des Bins: 100
Largeur des Bins: 100
Nombre d'objets: 100
Imprimer
Quitter


bin packing 2D
Surface perdue : 5931
Surface perdue : 10179
Fichier d'initialisation:
Soumettre
Heuristique:
hBase
Nombre de bins à afficher en largeur:
2
Confirmer
Bin(s) à visualiser:
tous
Deplacer Objet
Afficher/Masquer les données
Afficher les surfaces perdues
Nombre de Bins: 4
Hauteur des Bins: 300
Largeur des Bins: 300
Nombre d'objets: 100
Imprimer
Quitter

Problème de bin packing

Le problème de **bin packing** relève de la recherche opérationnelle et de l'optimisation combinatoire. Il s'agit de trouver le rangement le plus économique possible pour un ensemble d'articles dans des boîtes. Le problème classique se définit en une dimension, mais il existe de nombreuses variantes en deux ou trois dimensions.

Applications pratiques

Le problème de bin packing peut être appliqué à un grand nombre de secteurs industriels ou informatiques.

Pour la version classique en une dimension :

- rangement de fichiers sur un support informatique ;
- découpe de câbles ou de barres ;
- remplissage de camions ou de containers avec comme seule contrainte le poids ou le volume des articles.

Pour la version en deux dimensions :

- découpe de matière première : tissue tôle
- placement de boîtes sur une palette (sans superposition de boîtes)
- placement dans un entrepôt (sans superposition de boîtes)

Pour la version en trois dimensions :

- rangement d'objets dans des boîtes, des camions, etc...(avec superposition de boîtes)

Formulation du problème

Dans le problème classique, les données sont :

- un nombre infini de boîtes de taille C
- une liste 1, 2, 3 …n d'articles i de taille c_i.

On cherche à trouver le rangement valide pour tous ces articles qui minimise le nombre de boîtes utilisées. Pour qu'un rangement soit valide, la somme des tailles des articles affectés à une boîte doit être inférieure ou égale à C.

Pour décrire une solution, on peut utiliser un codage binaire pour indiquer dans quelle boîte j chaque objet i est rangé.

- La variable x_{ij} vaudra 1 si l'article i est rangé dans la boîte j, et 0 sinon.
- La variable binaire y_j est égale à 1 si la boîte j est utilisée, 0 sinon.

On cherche donc à minimiser le nombre de boîtes utilisées

$$min \sum_{j=1}^{n} y_j$$

Sous les contraintes suivantes :

$$\sum_{i=1}^{n} c_i x_{ij} \leq C y_j, j = 1, \ldots, n$$
$$\sum_{j=1}^{n} x_{ij} = 1, i = 1, \ldots, n$$
$$x_{i,j} \in \{0, 1\}, i = 1, \ldots, n, j = 1, \ldots, n$$
$$y_j \in \{0, 1\}, j = 1, \ldots, n$$

La première inégalité signifie qu'on ne peut dépasser la taille d'une boîte pour un rangement. A noter que la partie droite de l'inégalité *oblige* y_j à prendre la valeur 1 dès qu'un article est rangé dans la boîte *j*. La deuxième inégalité impose à tous les objets d'être rangés dans une boîte et une seule. Toute solution pour laquelle la famille d'équations précédente est vérifiée est dite *réalisable*.

La modélisation décrite plus haut a été proposée par Leonid Kantorovich[1] en 1960. Il existe d'autres formulations linéaires pour ce problème, sous forme d'un problème de flot maximum dans un graphe, ou utilisant une décomposition de Dantzig et Wolfe.

Méthodes de résolution

Le problème de bin packing a été largement étudié dans la communauté de recherche opérationnelle. Il existe des heuristiques très efficaces pour le résoudre, et une modélisation très efficace utilisant la programmation linéaire.

Méthode heuristiques]

Pour résoudre le problème de bin packing, on utilise souvent des algorithmes simples comme *first-fit decreasing* (FFD) ou *best-fit decreasing* (BFD). Les deux méthodes fonctionnent suivant un principe similaire : on trie la liste d'articles par ordre décroissant de taille, puis on range chaque article dans l'ordre.

Voici deux algorithmes classiques utilisés pour 1*BP*. Ils seront aussi utilisés dans la suite pour la résolution de 2*BP*.

- L'algorithme *First Fit Decreasing* fonctionne de la manière suivante : les articles sont considérés dans l'ordre décroissant de leur longueur c_i. A chaque étape *i*, l'algorithme place l'article courant dans le premier bin qui peut l'accueillir. Si aucun bin ne peut l'accueillir, un nouveau bin est ouvert.
- L'algorithme *Best Fit Decreasing* fonctionne de manière similaire : la seule différence réside dans le choix du bin dans lequel l'article courant est placé. On utilise cette fois le bin le plus rempli qui puisse l'accueillir.

Ces algorithmes ne sont pas optimaux, mais ils permettent d'obtenir de très bons résultats en pratique.

Les algorithmes *Best Fit Decreasing* et *First Fit Decreasing* n'utilisent jamais plus de 11/9 **OPT** + 1 boîtes (où **OPT** est le nombre optimal de boîtes dans une solution optimale).La procédure de tri est la partie la plus coûteuse de l'algorithme, mais sans elle, la qualité de la méthode est beaucoup moins bonne. On obtient dans ce cas des solutions utilisant au pire 17/10 **OPT** + 2 boîtes.

Une version plus efficace de FFD utilise au plus 71/60 **OPT** + 1 boîtes.

Méthodes exactes

On utilise aujourd'hui essentiellement la programmation linéaire en nombres entiers pour résoudre ce problème. Lorsque l'instance traitée est de faible taille, la formulation de Kantorovich peut être utilisée. Lorsque le nombre d'articles est grand, on utilise plutôt une résolution par génération de colonnes utilisant le modèle de Gilmore et Gomory[4], ou des modèles reposant sur la résolution d'un problème de flot maximal. La grande qualité des méthodes obtenues est due à l'excellente *relaxation linéaire du modèle*. La qualité de cette relaxation faire d'ailleurs l'objet d'une conjecture, appelée *MIRUP* : si *L* est la la valeur de la solution de ce modèle en nombres réels et *OPT* la valeur de la solution en nombres entiers, alors on aurait $|L| + 1 \geq OPT$

LA BIOLOGIE AU SERVICE DE LA LOGISTIQUE

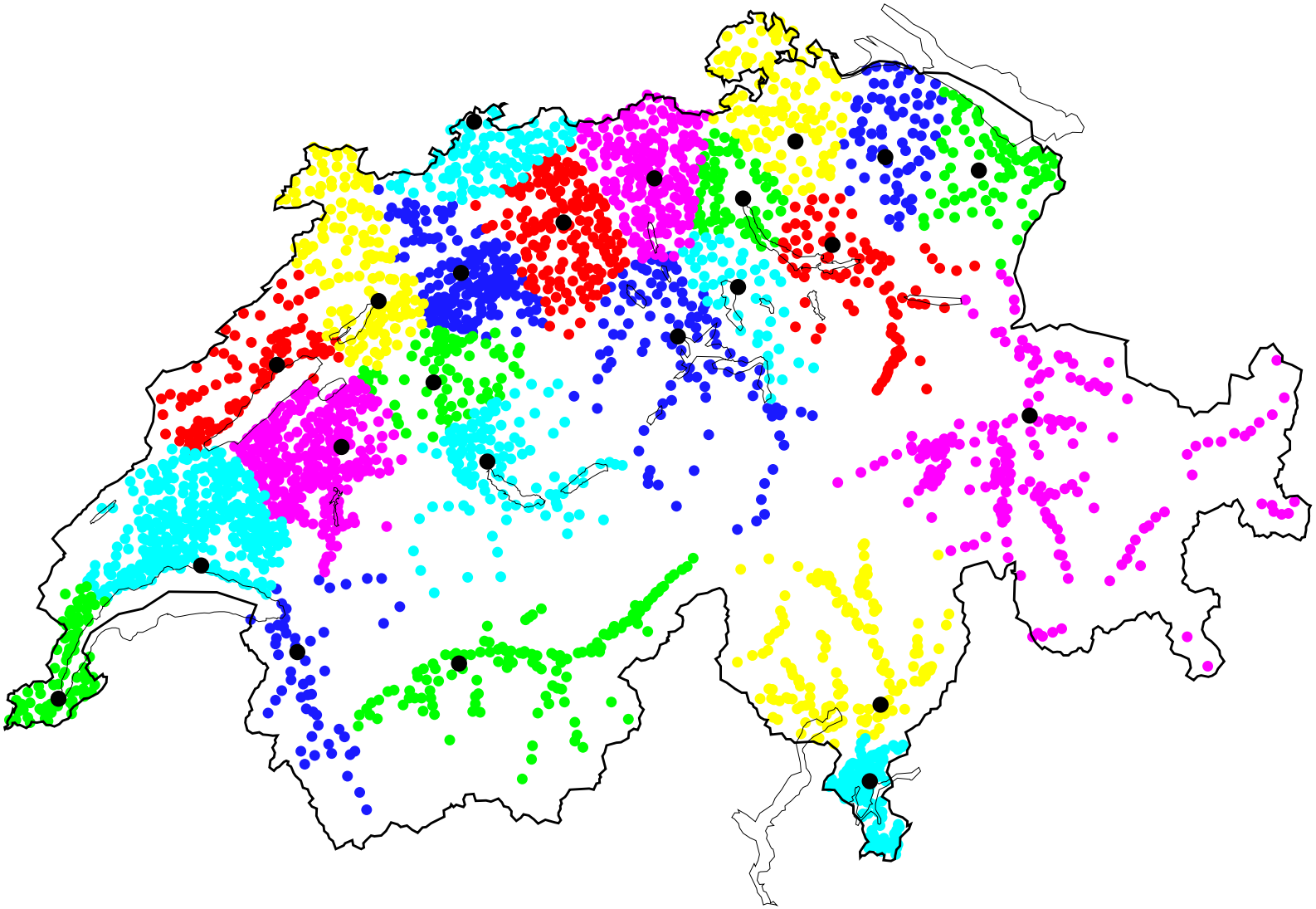
Éric Taillard

EIVD

Yverdon-les-Bains

PROBLÈMES DE PLACEMENT

Où placer des entrepôts, antennes, bureau de poste, stations de taxi



ALGORITHMES GÉNÉTIQUES

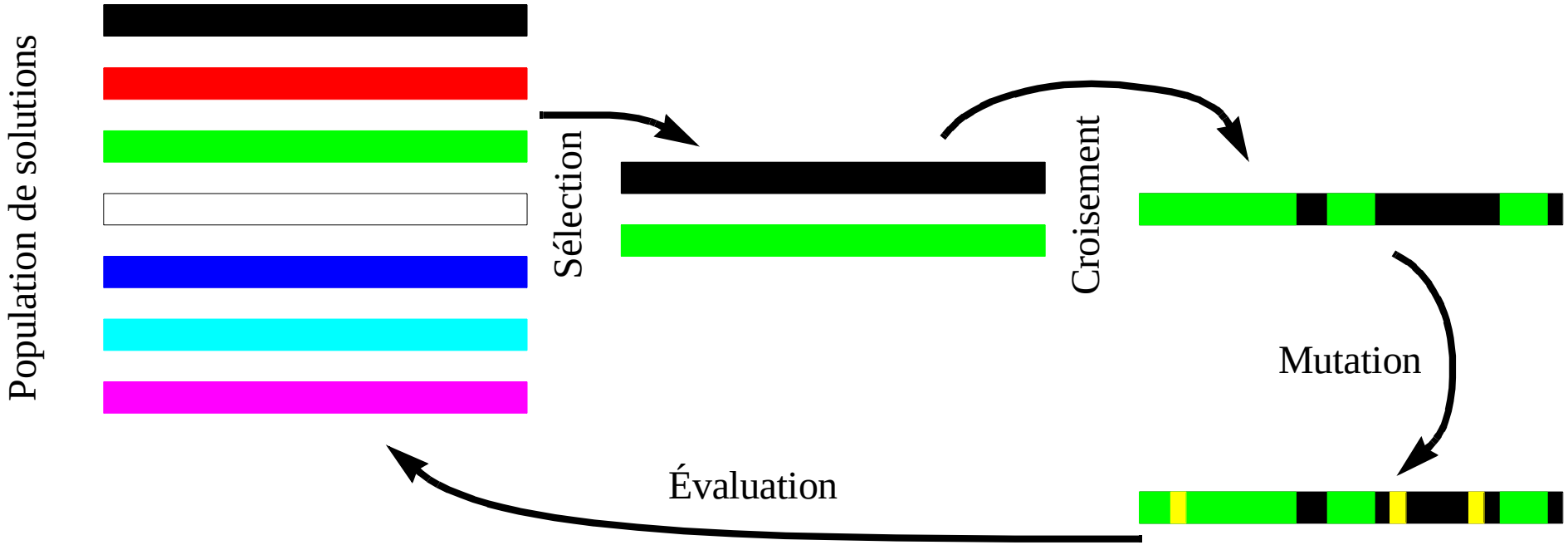
Processus d'évolution naturelle :

Sélection de deux individus dans une population.

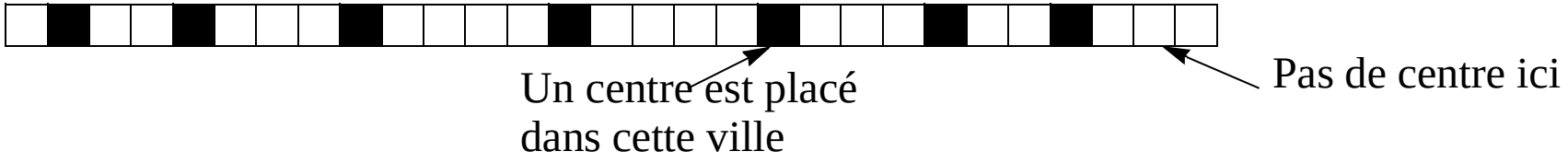
Croisement de ces deux individus, de façon aléatoire ; mutations aléatoires pour former un nouvel individu.

Incorporation du nouvel individu dans la population.

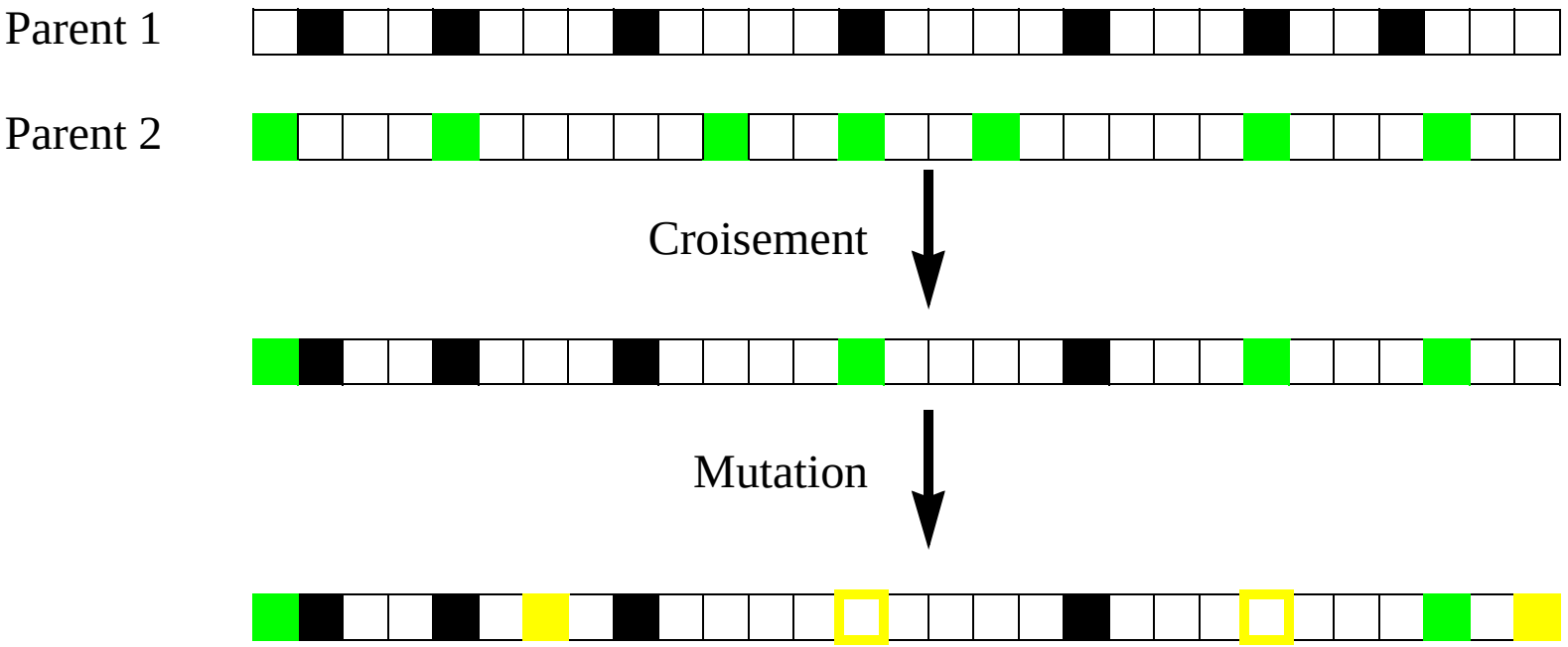
Élimination des individus trop faibles, âgés, malades.



Transcription du processus pour problème de localisation :

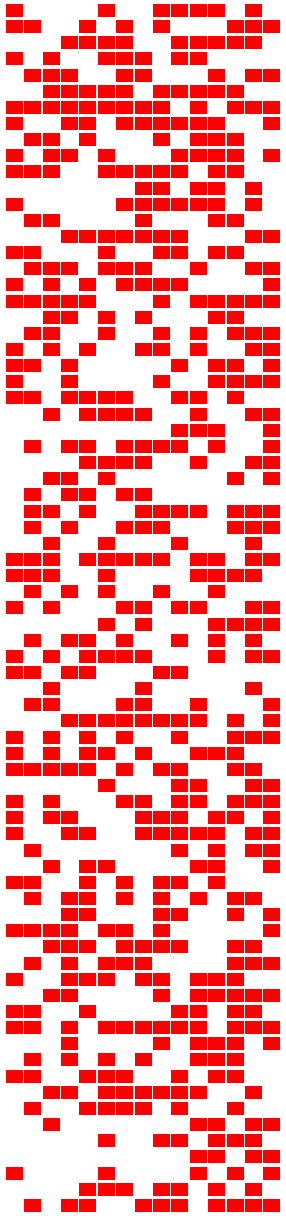


Représentation d'une solution

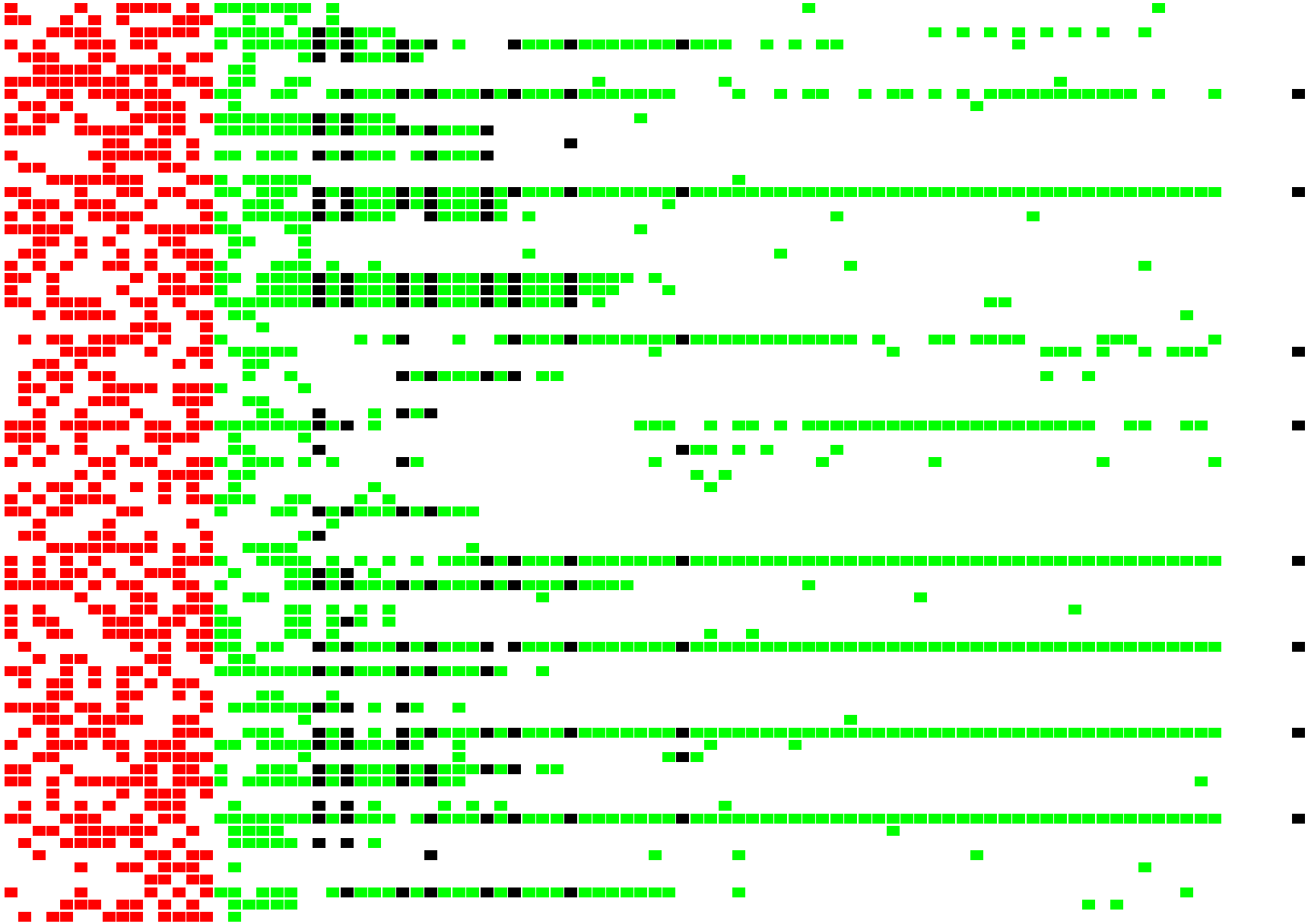


Simulation du processus de reproduction

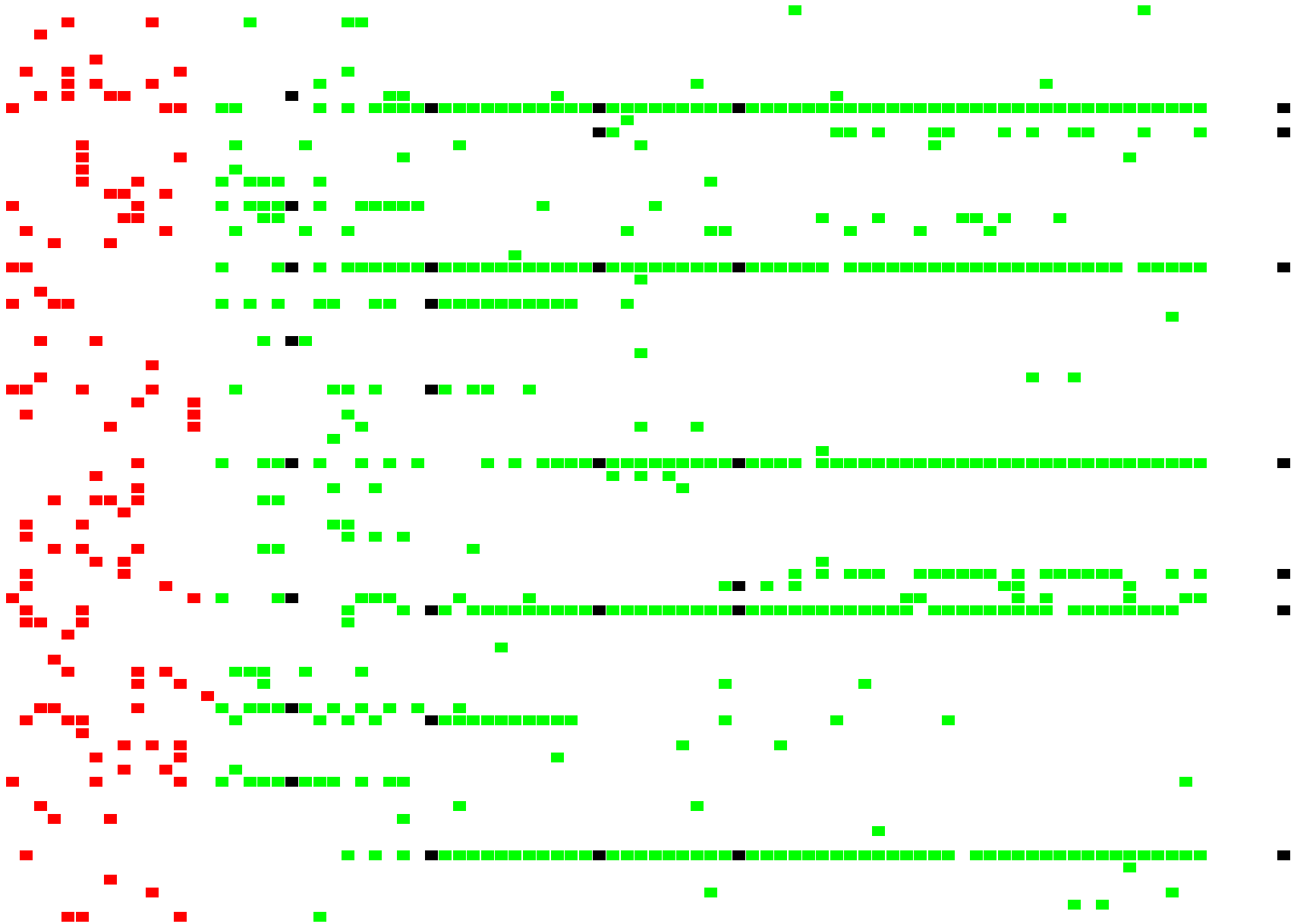
GÉNÉRATION D'UNE POPULATION INITIALE



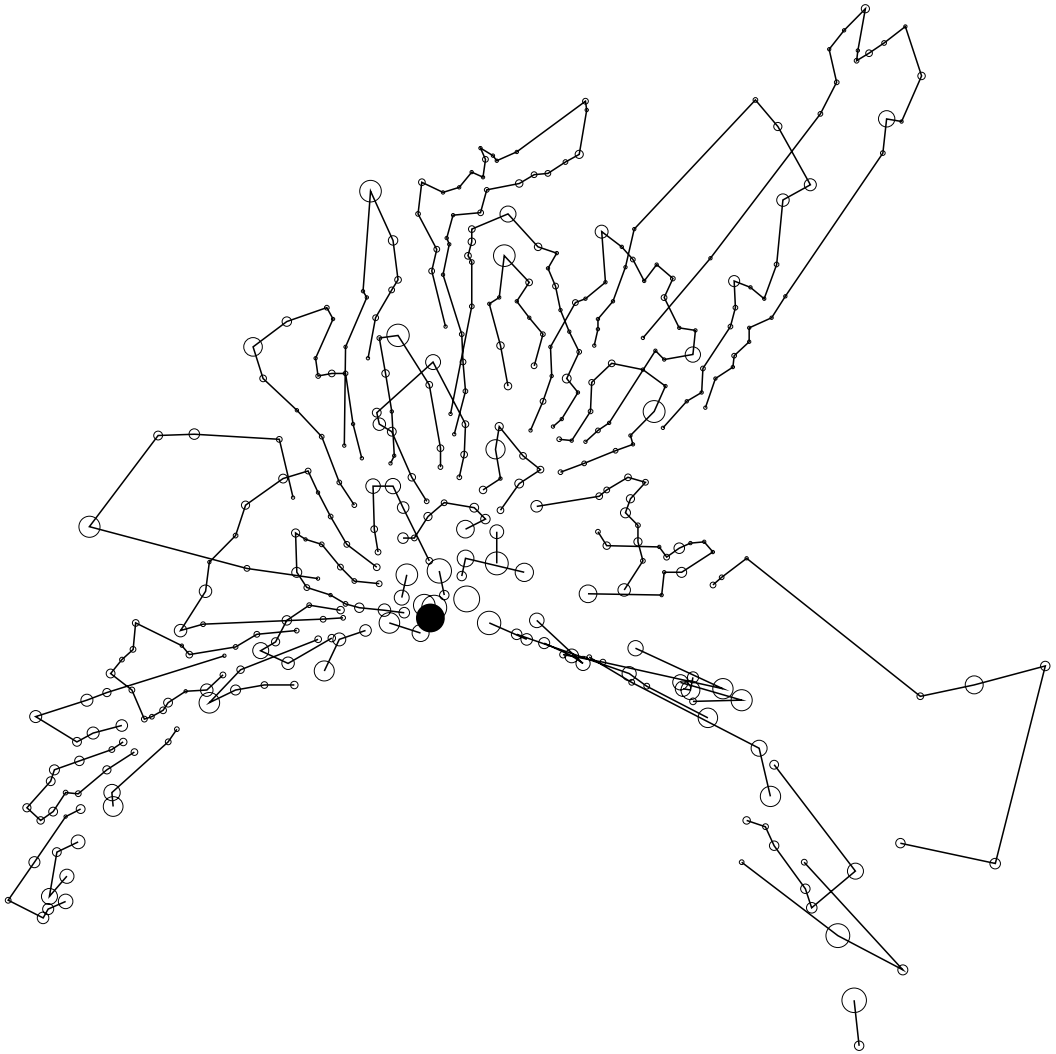
ÉVOLUTION DU PROCESSUS



MODIFICATION DE LA POPULATION INITIALE



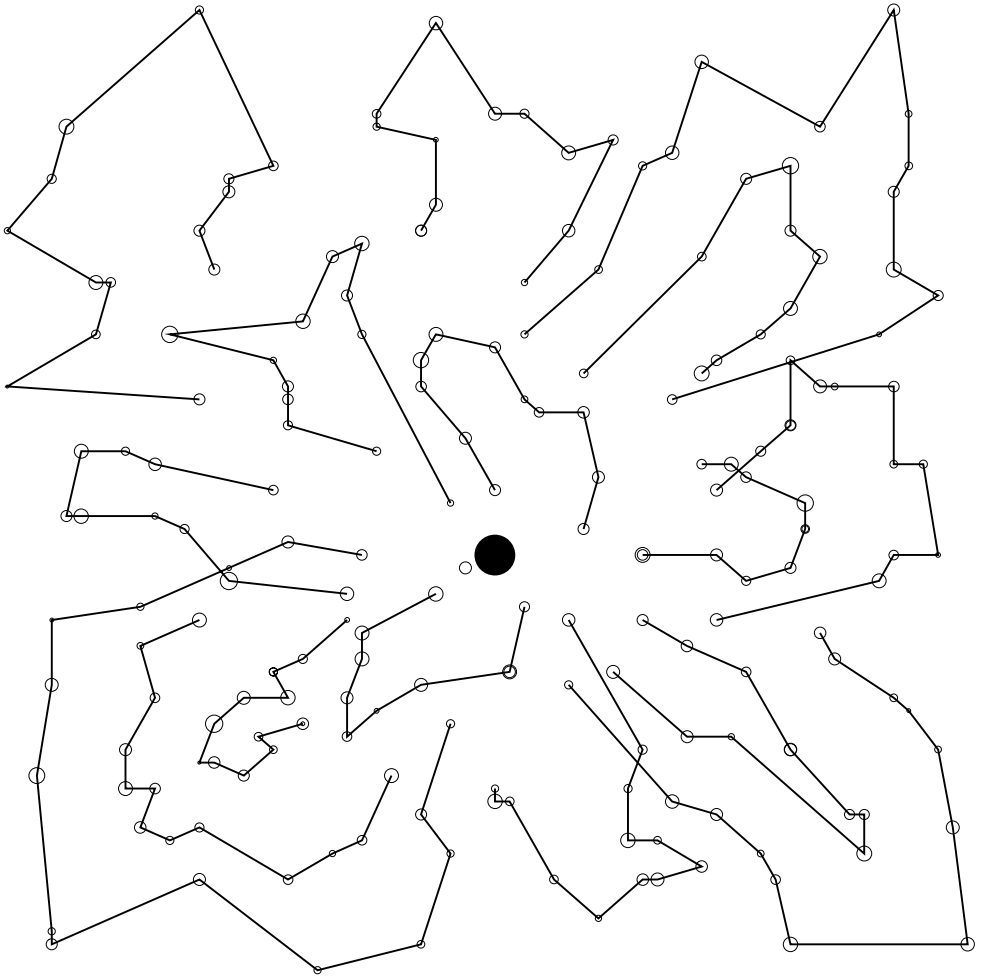
PROBLÈMES D'ÉLABORATION DE TOURNÉES



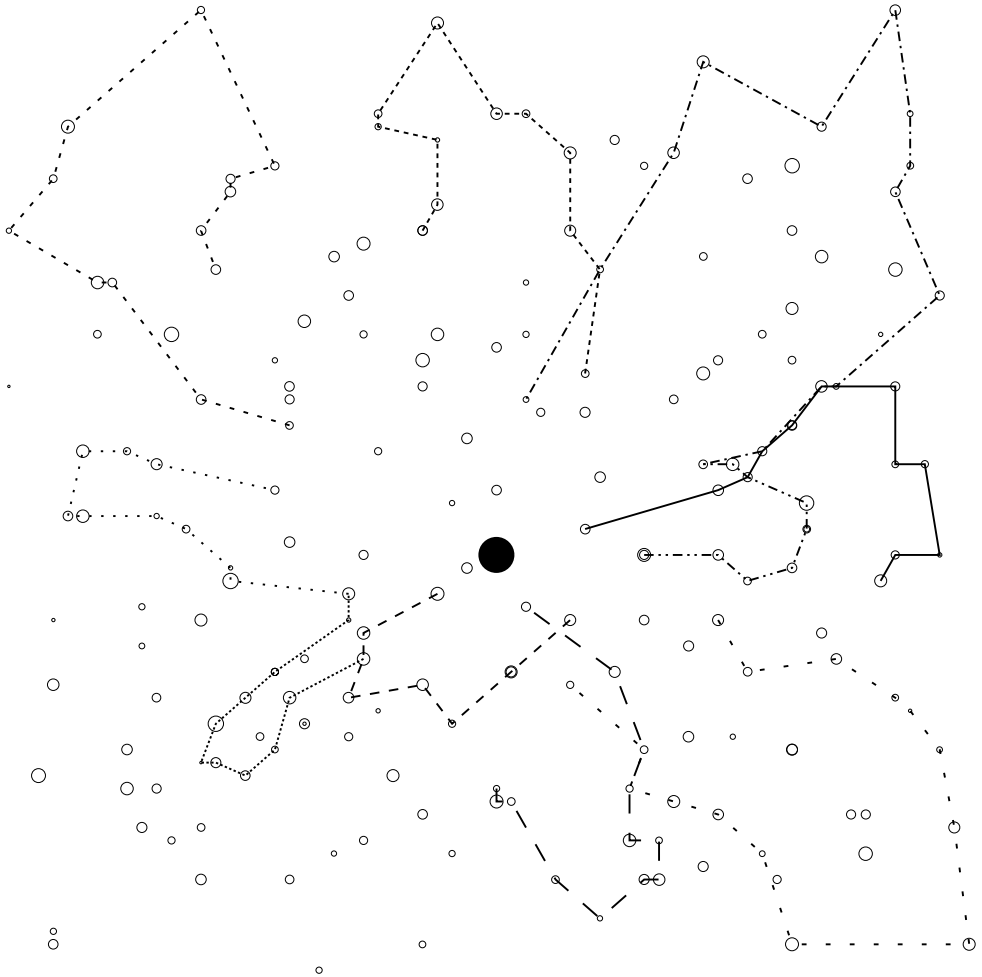
Quels clients placer sur quelle tournée, dans quel ordre desservir les clients ?

GÉNÉRALISATION DES ALGORITHMES GÉNÉTIQUES :

LA RECHERCHE PAR DISPERSION

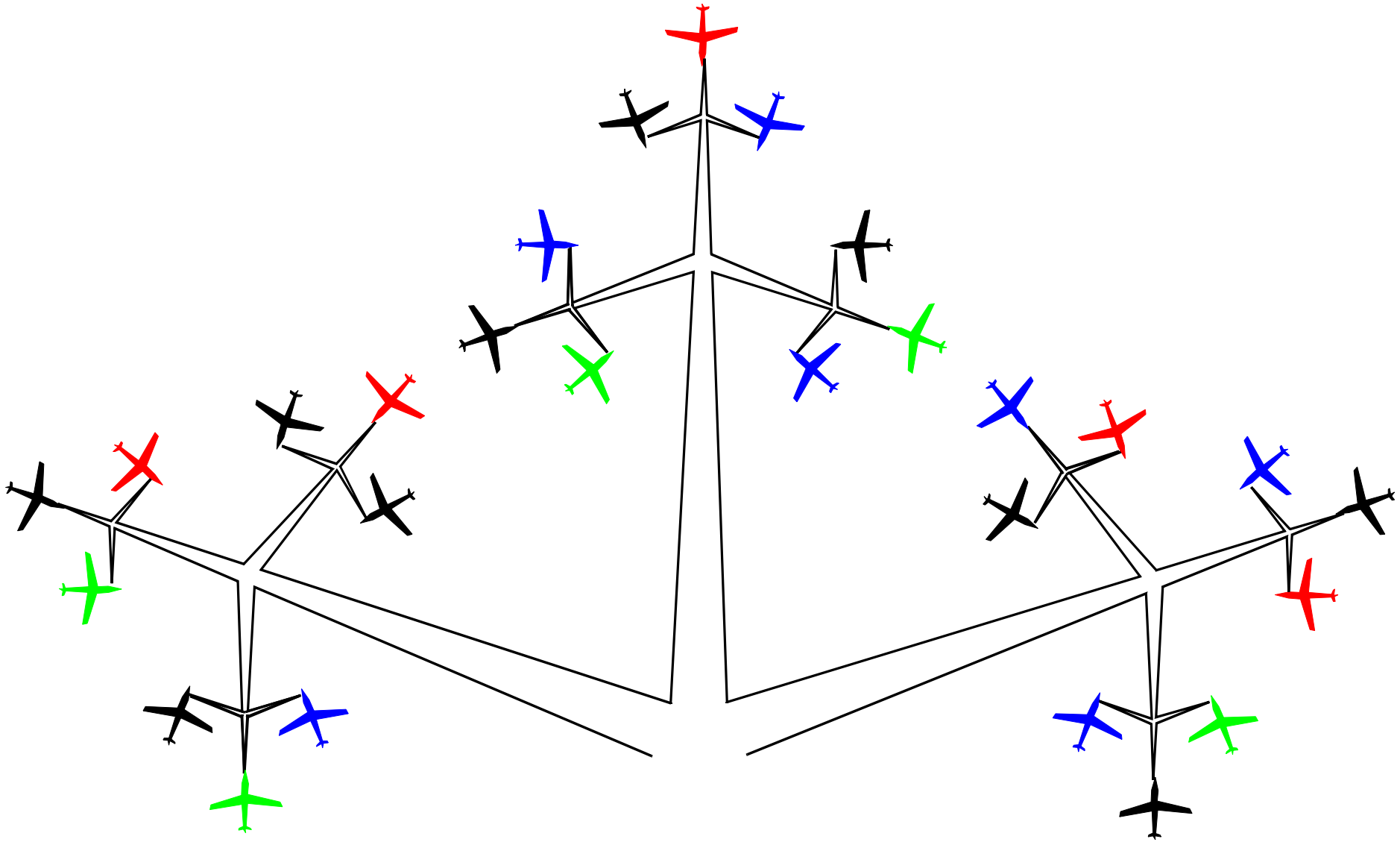


Meilleure solution connue d'un problème



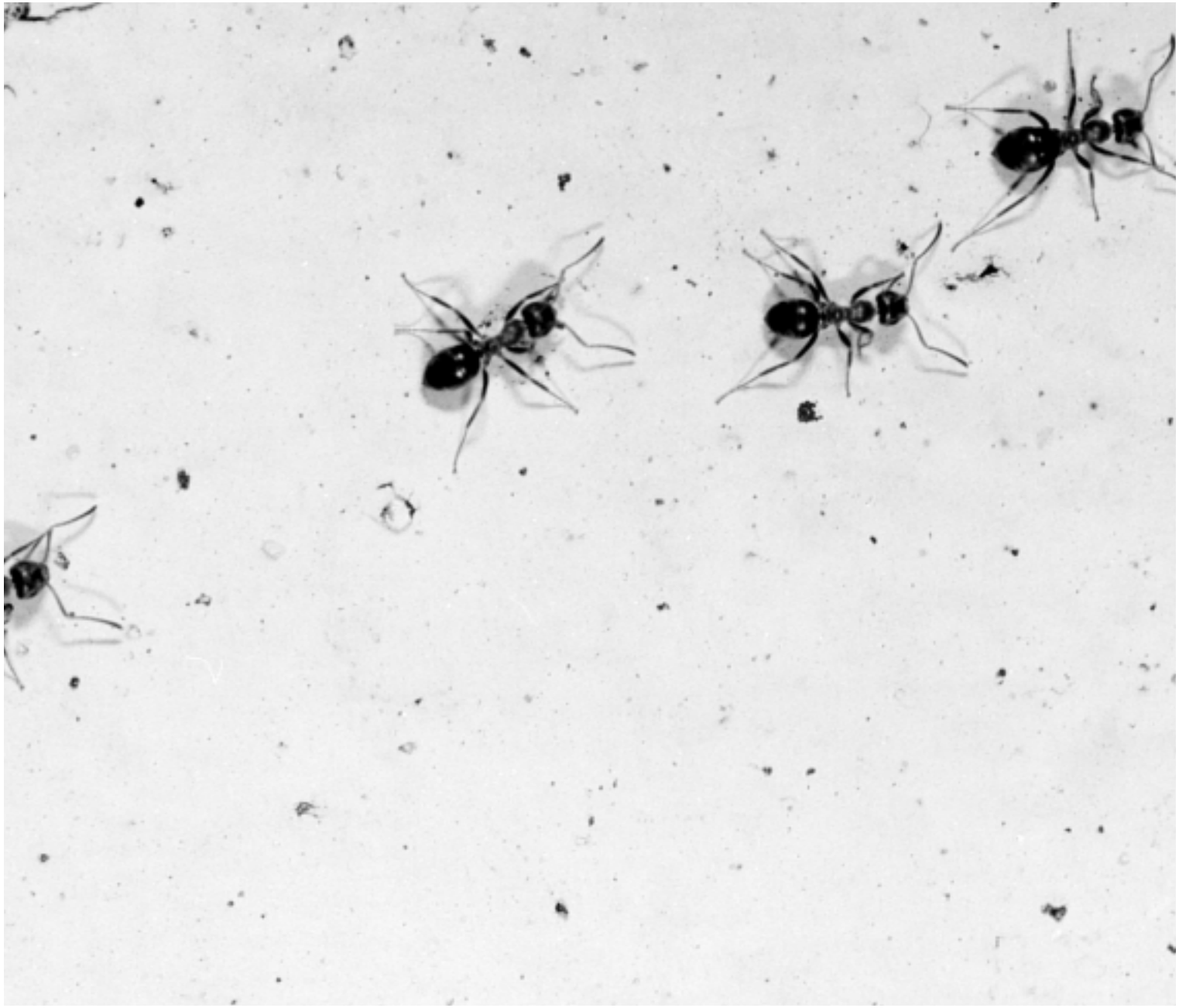
Quelques tournées trouvées après recherche sommaire avec une technique non déterministe

PROBLÈME D'AFFECTATION

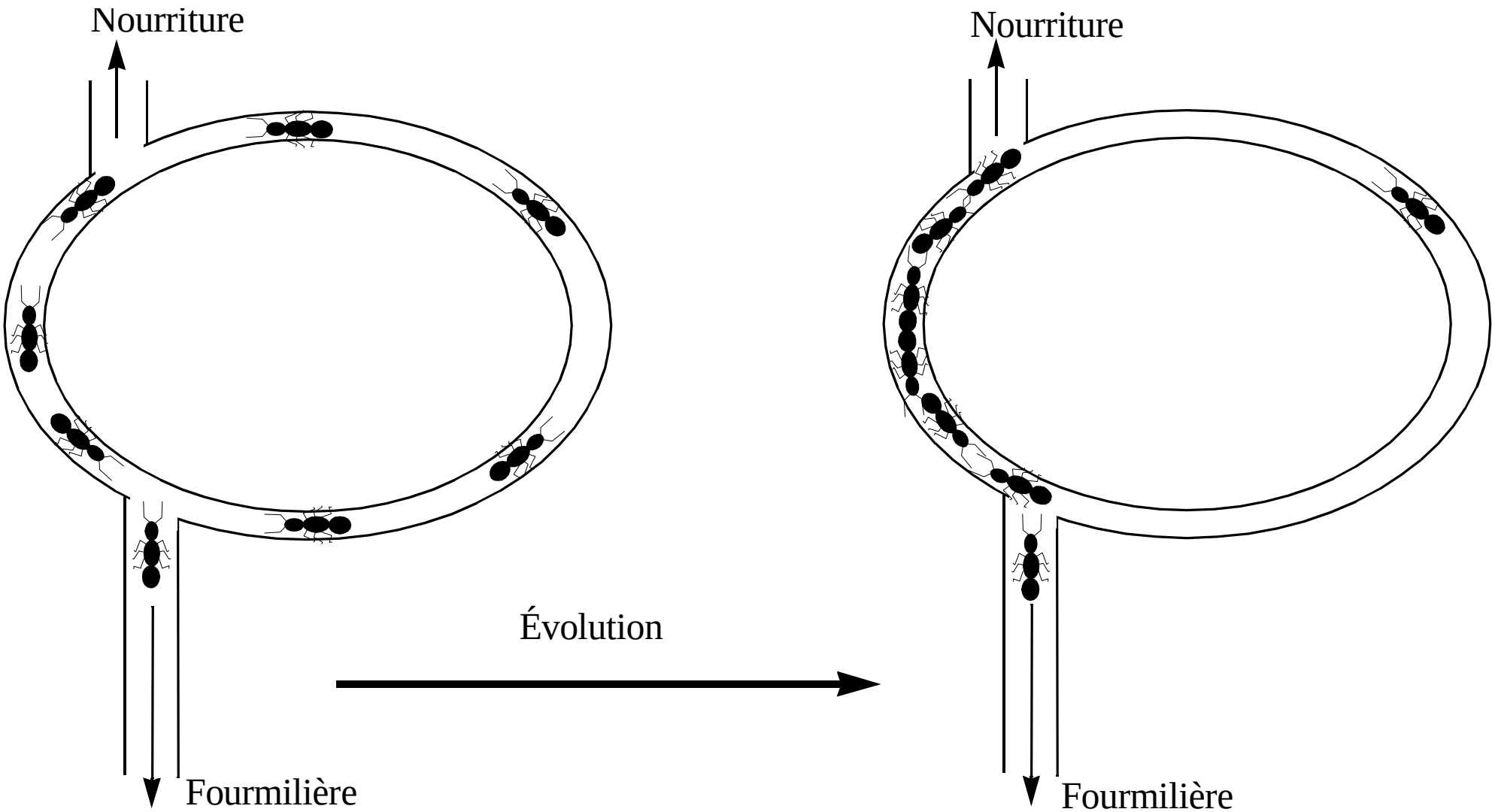


Quelle porte d'embarquement allouer à chaque avion ?

CHEMINS DE FOURMI



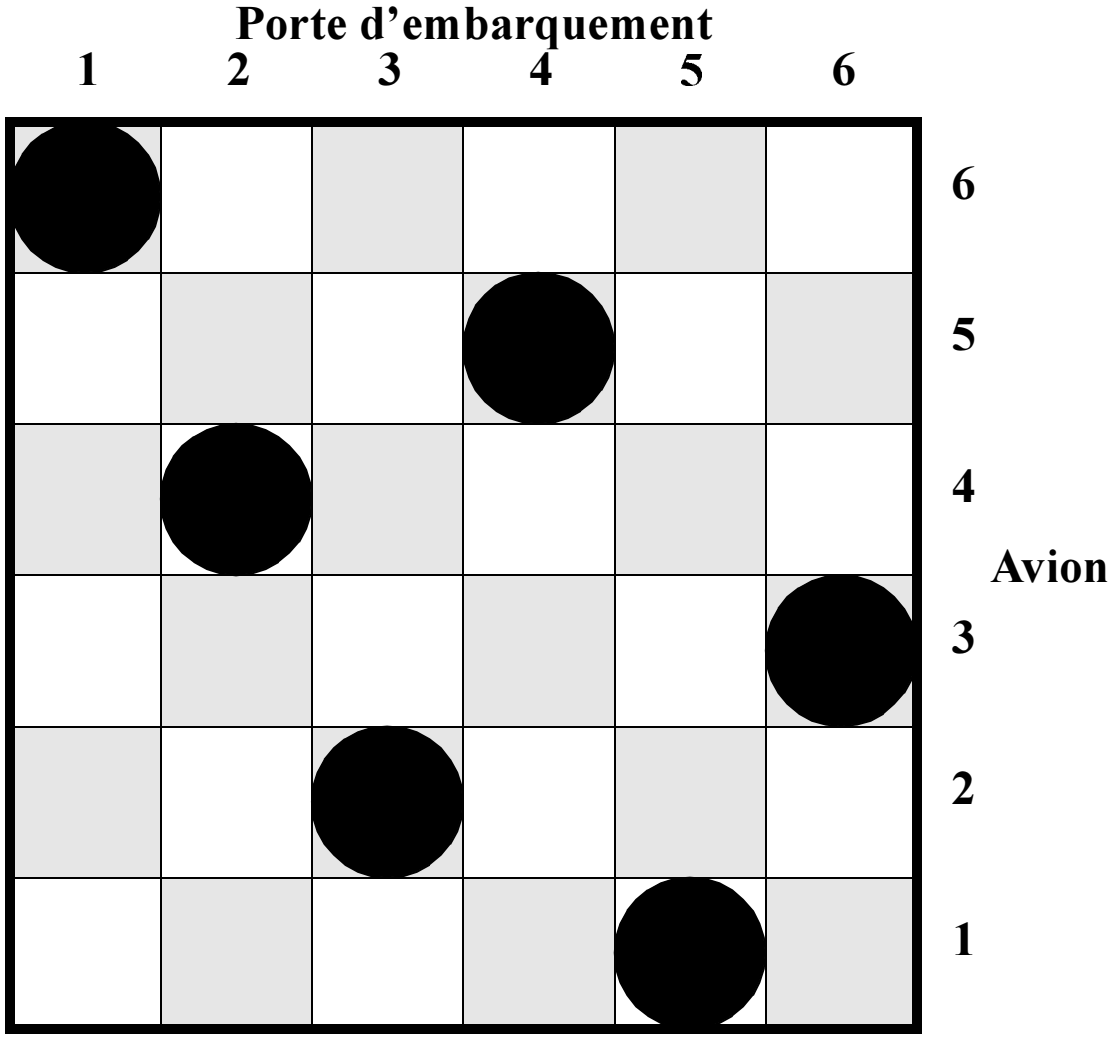
OPTIMISATION DE CHEMIN PAR LES FOURMIS



La fourmilière est séparée d'une source de nourriture par deux tubes différents. Après un certain temps, les fourmis empruntent le tube le plus court, les phéromones dans celui-ci augmentant plus rapidement.

Modélisation du problème de l'affectation des portes d'embarquement aux avions :

Placer des tours sur un échiquier de telle manière qu'elles ne s'attaquent pas.

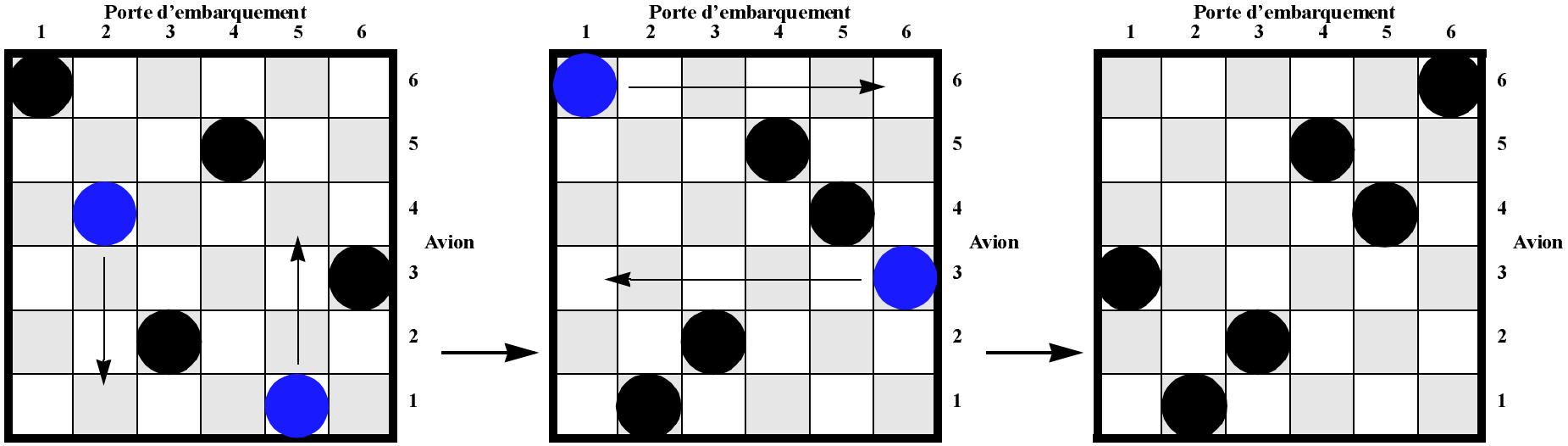


COLONIES DE FOURMI ARTIFICIELLES

Processus de construction d'une solution, à répéter un grand nombre de fois :

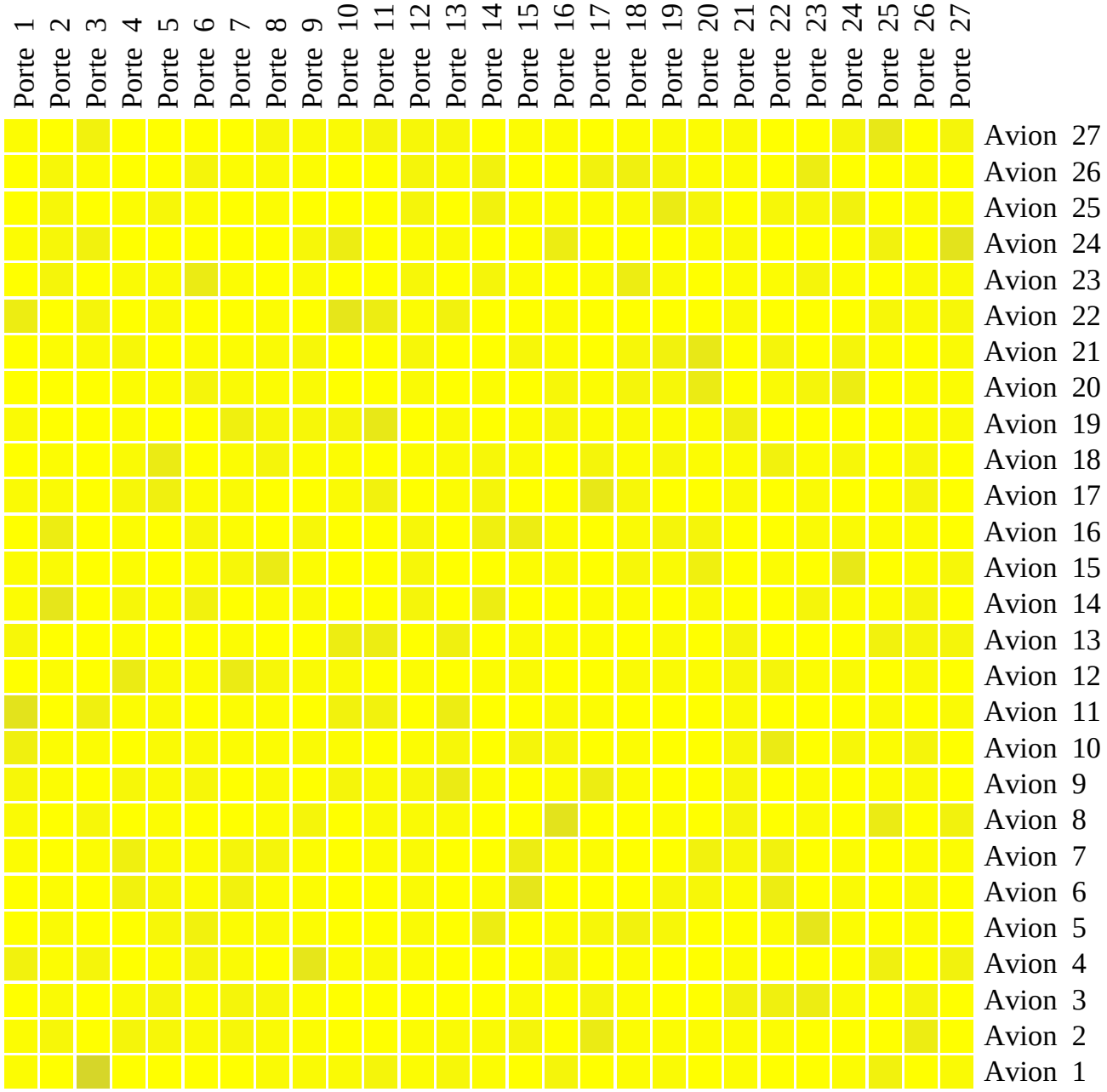
1. Répéter pour chaque tour:
   Choisir une case admissible avec une probabilité proportionnelle à la quantité de phéromone artificielle déposée sur la case.

2. Essayer d'améliorer la solution ainsi construite en déplaçant 2 tours à la fois:

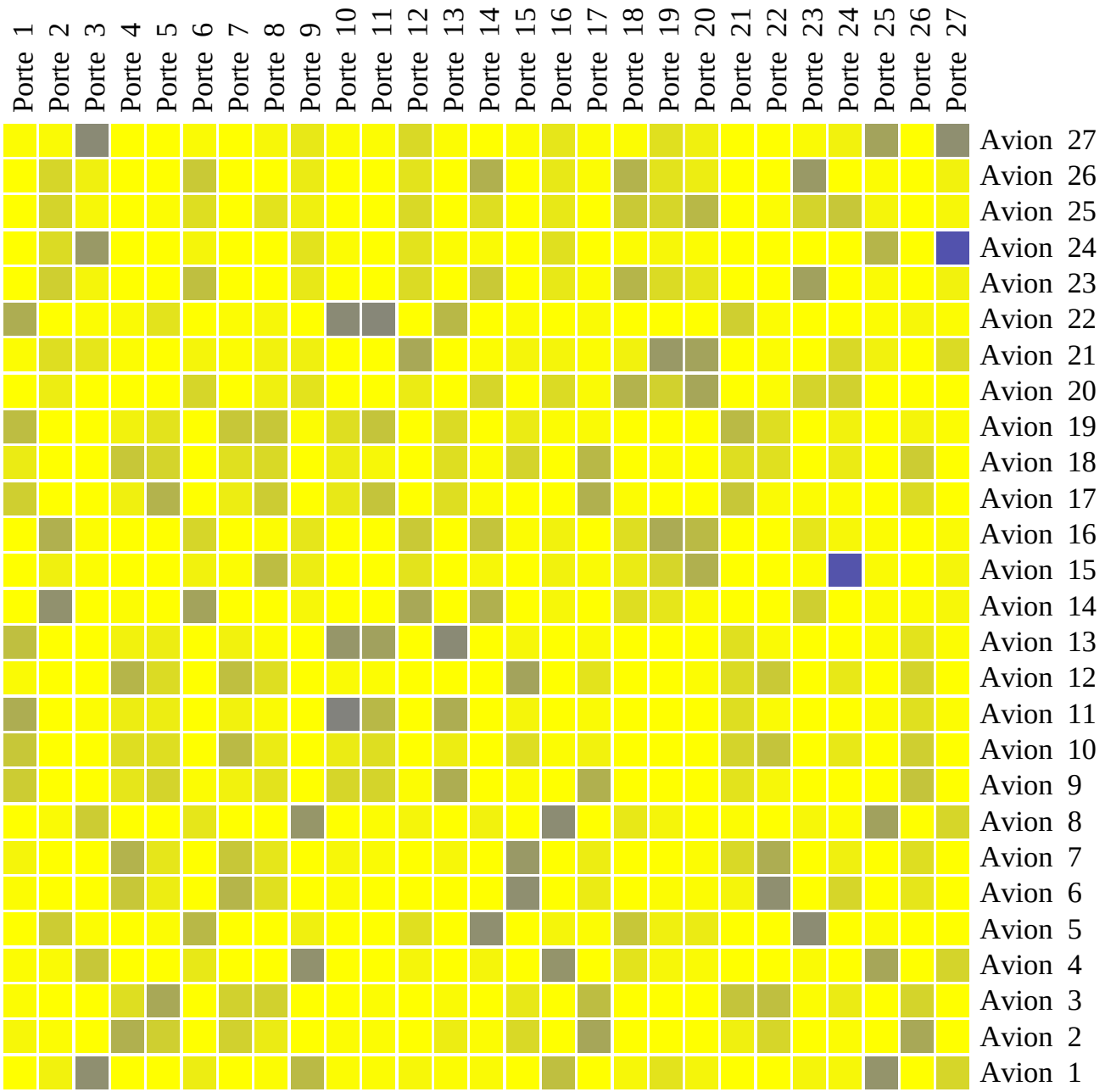


3. Déposer une certaine quantité de phéromone artificielle sur les cases occupées en dernier lieu par les tours.

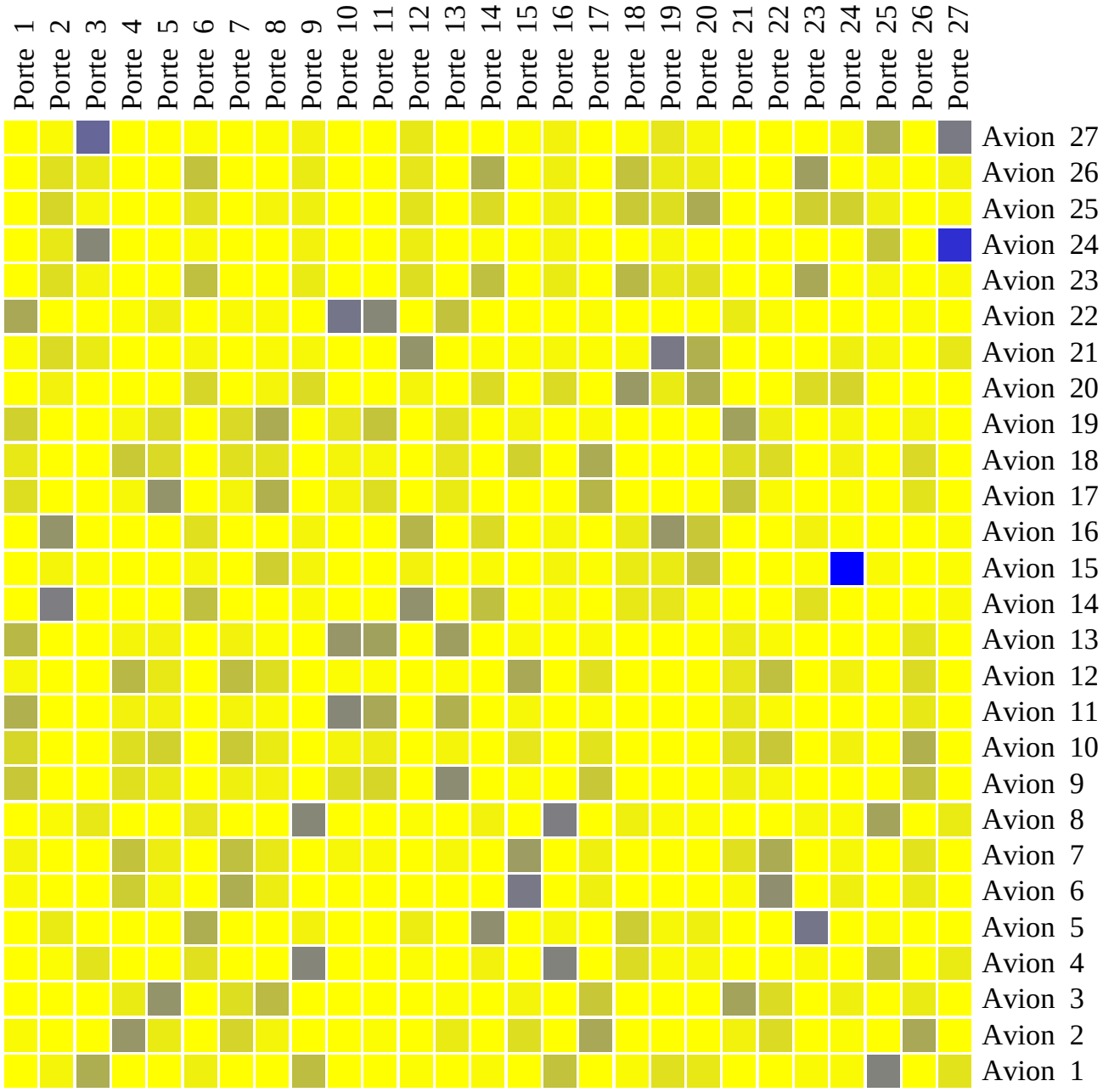
TRACES DE PHÉROMONES (BLEUES) AU DÉBUT



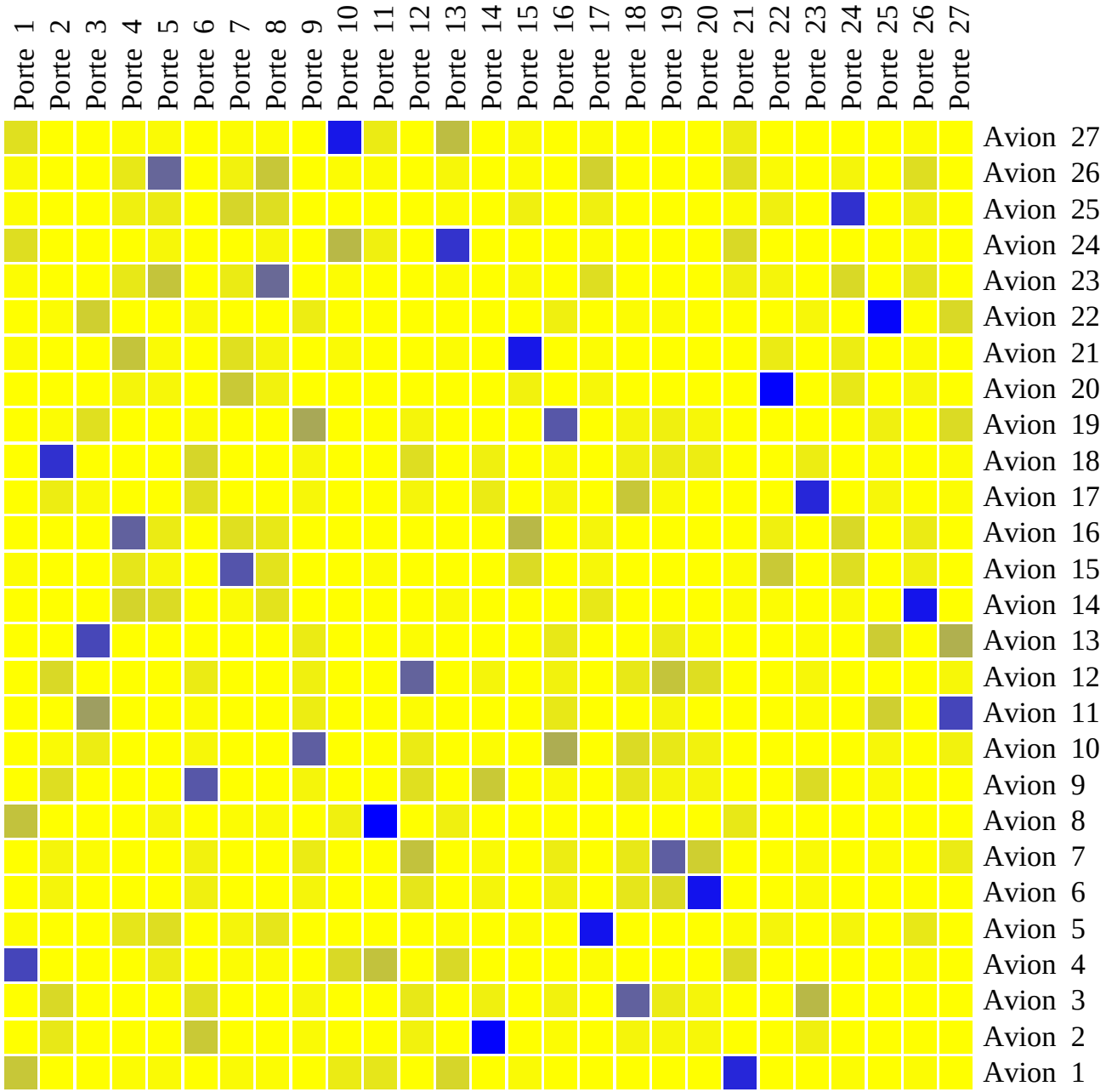
ÉVOLUTION DES TRACES AU COURS DU TEMPS



ÉVOLUTION DES TRACES AU COURS DU TEMPS (2)



TRACES EN FIN D'EXÉCUTION DU PROGRAMME



CONCLUSIONS

La nature peut nous inspirer des techniques d'optimisation très générales.

Le fonctionnement de ces techniques est un peu magique ; les connaissances théoriques sont encore très lacunaires.

La simplicité des méthodes d'optimisation en font un outil très flexible, en particulier pour aborder des problèmes pratiques présentant de multiples contraintes.

Pour de nombreux problèmes, les programmes mis au point en s'inspirant de ces techniques sont les plus efficaces à l'heure actuelle.

L'application de ces techniques reste encore très limitée dans l'industrie, même si des améliorations énormes pourraient être parfois obtenues.