TP3 : Algorithmes gloutons

SETCOVER en dimension 2

L'objectif de ce TP est d'implanter l'algorithme glouton pour SETCOVER dans le plan vu en cours. On fournit pour cela :

- un fichier `tests.cc` qui contient le `main` ;
- deux fichiers `AffichageMaisons.h/cc` pour l'affichage graphique ;
- deux fichiers `SetCover.h/.cc` (`SetCover.cc` **est l'unique fichier à modifier**) ;
- un `Makefile`.

On représente les coordonnées des maisons par un tableau de couples `int coordMaisons[][2]`, chaque maison étant identifiée par son indice dans ce tableau. Pour représenter le choix de placement d'émetteurs, on utilise un tableau `bool emetteurs[]` qui contient `true` en case `i` si un émetteur est placé sur la maison d'indice `i`.

1. Compiler (`make`) et exécuter `tests` et observer le fichier `Maison.svg` produit.
2. Compléter la fonction `bool Couvre(int i, int j, const int coordMaisons[][2])` qui retourne vrai si, et seulement si, les maisons i et j se situent à moins de `dcouv` l'une de l'autre. *La valeur de la variable* `dcouv` *est fixée par* `#define dcouv 100` *dans le fichier* `TP3Exo2.cc`.
3. Compléter la fonction `int ChoixProchaineMaison(int n, const int coordMaisons[][2], const bool dejaCouvertes[])` qui retourne l'indice de la prochaine maison sur laquelle placer un émetteur. Le tableau `dejaCouvertes[]` indique quelles maisons sont déjà couvertes.
4. Compléter la fonction `void ChoixEmetteurs(int n, const int coordMaisons[][2], bool emetteurs[])` qui trouve un ensemble d'émetteurs qui couvre toutes les maisons, avec l'algorithme vu en cours. *Pour tester votre fonction, appeler le programme produit avec* `./tests <N>` *où* `<N>` *est un entier, pour générer N maisons aléatoires et obtenir une couverture. Les fichiers* `Maisons.svg` *et* `Emetteurs.svg` *produits doivent être semblables à ceux de la figure 1.*

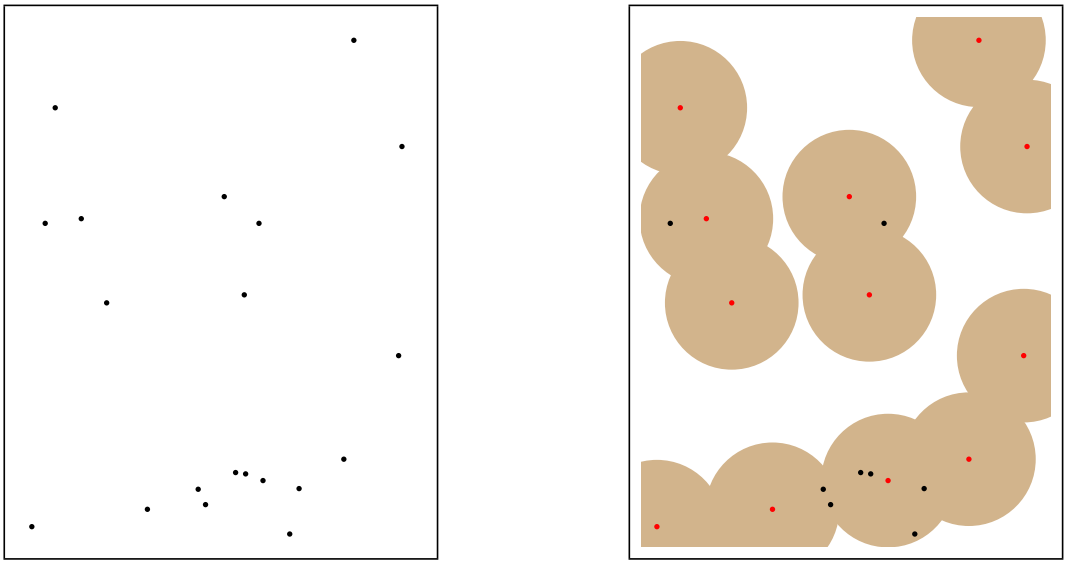

FIGURE 1 – À gauche, les maisons dans le plan, à droite les émetteurs placés (en rouge) et leur couverture.

5. Implanter la fonction `ChoixEmetteursOpt` qui effectue le même travail que `ChoixEmetteurs` mais avec un algorithme qui renvoie une solution optimale. *Indications. On pourra effectuer une recherche exhaustive, c'est-à-dire parcourir toutes les possibilités de placement d'émetteurs, et garder la meilleure. Ne pas hésiter à écrire une ou plusieurs fonctions auxiliaires pour faciliter la programmation !*

6. Rechercher un exemple où l'algorithme glouton donne un résultat non optimal.

7. *(bonus)* Améliorer la fonction `ChoixEmetteursOpt` afin de la rendre **la plus rapide possible**.