Algorithme 1 : Division de l'espace en sous-groupes indépendants

```
Entrées:
          C_{num siege, coordX, coordY};
                                                     ▷ Une matrice de n lignes
          d;
                                                      ▷ Une distance en mètres
          M_{i,j};
                                           ▷ Une matrice de dist. eucl. vide
   Sorties:
          C*_{numsiege,coordX,coordY,\mathbf{numgroupe}}
 1 Calcul des distances euclidiennes ;
 2 pour chaque i \in C faire
        pour chaque i \in C faire
          M_{i,j}=distance euclidienne i, j
 5
      _{\rm fin}
 6 fin
 7 Assignation des groupes de chaises ;
                             ▷ Initialiser le compteur de groupe à 0
 9 pour chaque i \in C faire
        i_{numgroupe} = 0
11 fin
12 pour chaque n \in C faire
        si n_{numgroupe} = \theta alors
13
          G+=1;
14
          n_{numgroupe} = G;
                                     ▷ Assigner la chaise au groupe G
15
                                   ▷ Créer une liste avec l'élément n
          L=[n];
16
          pour chaque i \in L faire
17
              pour chaque j \in C faire
18
                 \mathbf{si} \ i! = j \ ET \ M_{i,j} < d \ \mathbf{alors}
19
                     j_{numgroupe} = G;
                                                     ▷ Ajouter le voisin
20
                     non-distancé au même groupe
                                        ▷ L'ajouter à la liste L pour
                     L.append(j);
21
                      itérer à travers ses voisins
              _{
m fin}
22
          fin
23
24 fin
```

Algorithme 2 : Allocation des sièges

```
Entrées:
         C_{numsiege,coordX,coordY};
                                                 ▷ Une matrice de n lignes
         F_{numsiege,coordX,coordY,assignation}; \triangleright Une matrice de n lignes, vide
         ou remplie de zéros
                                                            ▷ Une liste vide
         L;
         d;
                                                   ▷ Une distance en mètres
         iter;
                         ▷ Un nombre maximal d'itérations de stagnation
   Sorties:
         F_{numsiege,coordX,coordY,assignation} ; \triangleright Une matrice de n lignes avec
         l'assignation des chaises
 1 Allocation des chaises;
 2 compteur = 0;
                                        ▷ Compteur des boucles while
 3 dernierchangement = 0; ▷ Le numéro de boucle avec la dernière
    amélioration
 4 solutionoptimale = 0;
                                              ▷ La solution initiale
 5 tant que (compteur - dernierchangement) > iter faire
      tant que C > 0 faire
         Choisir numsiege au hasard entre 0, n i=numsiege F_i = C_i;
 7
          ▷ Copier la ligne i dans F à la même position
                                         ▷ Assigner l'occupation à 1
          F_{i,assignation} = 1;
 8
         pour chaque j \in C faire
 9
             dist = distance euclidienne C_{i,j};
10
             \mathbf{si} \ dist > d \ ET \ dist > 0 \ \mathbf{alors}
11
                              F_j = C_j;
12
                 position
                F_{j,assignation} = 1 \; ; \;\; 
ightharpoons {
m Assigner occupation ~ \^a ~ 0 ~ (trop ~ )}
13
                 proche)
                                                        ▷ Retirer de C
                C_i=0;
14
         _{\rm fin}
15
         C_i=0;
                                                        ▷ Retirer de C
16
17
      fin
      S=somme(F_{assignation});
18
      si S > solution optimale alors
             ▷ Si la solution act. est meilleure que la solution
          opt.
                                     ▷ Affecter sol. actuelle comme
         solutionoptimale = S;
20
           solution opt.
         dernierechangement=compteur; ▷ Enregistrer l'itération
21
          avec l'amélioration
      Compteur+=1
22
23 fin
```