

---

**Algorithm 1** Insertion tache

---

```
1: function ADD_TASK( $du\_t, cpu\_t$ ) ▷ duration, number of cpu
2:
3:   On recherche le premier emplacement possible pour la tache avec au minimum la durée  $du\_t$  et le nombre
   de cpu  $cpu\_t$ .
4:
5:   On récupère le temps de départ  $starting\_time\_min$  et la range de cpu  $processor\_range\_t$  de l'emplace-
   ment.
6:   On réduit la range de cpu  $processor\_range\_t$  du nombre de cpu  $cpu\_t$ .
7:   On récupère la liste de tous les emplacements potentiellement impactés : ceux dont le temps de départ est
   inférieur ou égale au temps de départ min  $starting\_time\_min$  + la durée  $du\_t$  .
8:
9:   On ne garde que ceux pour qui :
10:     l'intersection de la range de cpu avec  $processor\_range\_t$  est non vide
11:   and
12:     dont le temps de départ + la durée  $\geq starting\_time\_min$ 
13:
14:    $tab\_resultat \leftarrow$  vide
15:
16:   for tous les emplacements restant do
17:     On supprime l'emplacement de l'arbre des emplacements
18:      $tab\_resultat = tab\_resultat \cup cut\_freespace(emplacement, starting\_time\_min, processor\_range\_t)$ 
19:   end for
20:
21:    $tab\_final \leftarrow$  vide
22:
23:   for chaque élément de  $tab\_resultat$  do
24:     if  $is\_necessary\_freespace(element, tab\_resultat)$  then
25:        $tab\_final = tab\_final \cup$  élément
26:     end if
27:   end for
28:
29:   for chaque élément de  $tab\_final$  do
30:     Ajouter à l'arbre des emplacements  $element$ 
31:   end for
32:
33: end function
```

---

---

**Algorithm 2** Decoupage Freespace

---

```
1: function CUT_FREESPACE(freespace, start_time, duration, processor_range)
2:   new_emplacement  $\leftarrow$  vide
3:
4:   if temps de départ de freespace < start_time then
5:     On crée un nouvel emplacement left_freespace avec :
6:       Temps de départ : celui de freespace
7:       Durée : start_time - temps de départ de freespace
8:       Cpu : Cpu de freespace
9:     new_emplacement  $\leftarrow$  new_emplacement  $\cup$  left_freespace
10:  end if
11:
12:  if l'intersection de la range de cpu de freespace et processor_range < au nombre de processeurs de
    freespace then
13:    On crée un nouvel emplacement new_freespace avec :
14:      Temps de départ : celui de freespace
15:      Durée : durée de freespace
16:      Cpu : différence entre la range cpu de freespace et processor_range
17:    new_emplacement  $\leftarrow$  new_emplacement  $\cup$  new_freespace
18:  end if
19:
20:  On crée un nouvel emplacement right_freespace avec :
21:    Temps de départ : (start_time + duration)
22:    Durée : durée de freespace - (temps de départ de freespace - start_time + (start_time + duration))
23:    Cpu : Cpu de freespace
24:  new_emplacement  $\leftarrow$  new_emplacement  $\cup$  right_freespace
25:
26:  return new_emplacement
27: end function
```

---

---

**Algorithm 3** Suppression tache

---

```
1: function REMOVE_TASK( $st\_t$ ,  $d\_t$ ,  $cpu\_t$ ) ▷ Starting time, duration, cpu range
2:    $et\_t \leftarrow (st\_t + d\_t)$ 
3:
4:   On récupère la liste de tous les emplacements qui ont un temps de départ inférieur à  $(st\_t + d\_t)$ 
5:
6:   On ne garde que ceux qui :
7:     existent dans l'intervalle de temps  $st\_t$  à  $et\_t$ 
8:
9:   for tous les emplacements restant do
10:     On supprime l'emplacement de l'arbre des emplacements
11:   end for
12:
13:    $tab\_resultat \leftarrow$  vide
14:
15:    $tab\_resultat = \text{extend\_freespace}(\text{liste des emplacements} + \text{task})$ 
16:
17:    $tab\_final \leftarrow$  vide
18:
19:   for chaque élément de  $tab\_resultat$  do
20:     if  $\text{is\_necessary\_freespace}(\text{element}, tab\_resultat)$  then
21:        $tab\_final = tab\_final \cup \text{élément}$ 
22:     end if
23:   end for
24:
25:   for chaque élément de  $tab\_final$  do
26:     Ajouter à l'arbre des emplacements  $element$ 
27:   end for
28:
29: end function
```

---

---

**Algorithm 4** Augmentation Freespace

---

```
1: function EXTEND_FREESPACE(freespaces)
2:   new_location  $\leftarrow$  vide
3:   events  $\leftarrow$  vide
4:
5:    $\triangleright$  events est une table de hachage de temps
6:    $\triangleright$  chaque temps a deux listes de freespace, 'start' et 'end'
7:
8:   for chaque éléments f de freespaces do
9:     Ajoute à events au temps de départ de f, f dans la liste "start"
10:    Ajoute à events au temps final de f, f dans la liste "end"
11:  end for
12:
13:  for chaque clef de events do
14:    Ajoute clef à times
15:  end for
16:
17:  On trie par ordre croissant times
18:  time_init  $\leftarrow$  premier temps de times
19:  cpu_init  $\leftarrow$  premier range cpu au premier temps times
20:
21:  for chaque éléments time de times do
22:    cpu  $\leftarrow$  vide
23:
24:    for chaque éléments f de ev.time.start do
25:      Ajout à la liste ranges la range cpu de f
26:    end for
27:
28:    for chaque éléments f de ev.temps.end do
29:      Supprime de la liste ranges la range cpu de f
30:    end for
31:
32:    for chaque éléments c de ranges do
33:      cpu  $\rightarrow$  add(c)
34:    end for
35:
36:    if cpu_init  $\neq$  cpu and time_init  $\neq$  time then
37:      On crée un nouvel emplacement new_freespace avec :
38:      Temps de départ : time_init
39:      Durée : time - time_init
40:      Cpu : cpu_init
41:      new_location  $\leftarrow$  new_location  $\cup$  new_freespace
42:      time_init  $\leftarrow$  time
43:    end if
44:    last_cpu  $\leftarrow$  cpu_init
45:    cpu_init  $\leftarrow$  cpu
46:  end for
47:
48:  On crée un nouvel emplacement new_freespace avec :
49:  Temps de départ : time_init
50:  Durée : time - time_init
51:  Cpu : last_cpu
52:  new_location  $\leftarrow$  new_location  $\cup$  new_freespace
53:
54:  return new_location
55: end function
```

---

---

**Algorithm 5** Suppression des Freespaces inutiles

---

```
1: function IS_NECESSARY_FREESPACE(freespace, freespace_list)
2:
3:   for tous les éléments de freespace_list do
4:     if éléments  $\neq$  freespace then
5:       if temps de départ de freespace  $\geq$  temps de départ de space and temps final de freespace  $\leq$  temps
        final de space then
6:         if (range cpu de freespace  $\cap$  range cpu de space) = nombre de cpu de freespace then
7:           return 0
8:         end if
9:       end if
10:    end if
11:  end for
12:  return 1
13: end function
```

---