
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 appelle la fonction extend_freespace(emplacement)
11:   end for
12: end function
```

Algorithm 4 Augmentation Freespace

```
1: function EXTEND_FREESPACE(freespace, task)
2:   new_emplacement  $\leftarrow$  vide
3:
4:   if temps de départ de freespace = temps final de task then
5:     if les cpu de freespace et task sont les mêmes then
6:       On modifie l'emplacement freespace avec :
7:       Temps de départ : celui de task
8:       Durée : Durée de task + durée de freespace
9:       Cpu : Cpu de freespace
10:      new_emplacement  $\leftarrow$  new_emplacement  $\cup$  freespace
11:    else
12:      On crée un nouvel emplacement left_freespace avec :
13:      Temps de départ : celui de task
14:      Durée : Durée de task + durée de freespace
15:      Cpu : Cpu de freespace  $\cap$  cpu de task
16:      new_emplacement  $\leftarrow$  new_emplacement  $\cup$  left_freespace  $\cup$  freespace
17:    end if
18:  else if temps de départ de task = temps final de freespace then
19:    if les cpu de freespace et task sont les mêmes then
20:      On modifie l'emplacement freespace avec :
21:      Temps de départ : celui de freespace
22:      Durée : Durée de task + durée de freespace
23:      Cpu : Cpu de freespace
24:      new_emplacement  $\leftarrow$  new_emplacement  $\cup$  freespace
25:    else
26:      On crée un nouvel emplacement right_freespace avec :
27:      Temps de départ : celui de freespace
28:      Durée : Durée de task + durée de freespace
29:      Cpu : Cpu de freespace  $\cap$  cpu de task
30:      new_emplacement  $\leftarrow$  new_emplacement  $\cup$  right_freespace  $\cup$  freespace
31:    end if
```

Algorithm 5 Suite

```
32:  else
33:    if temps de départ de freespace < temps de départ de task then
34:      if temps final de freespace ≤ temps final de task then
35:        On crée un nouvel emplacement new_freespace avec :
36:        Temps de départ : celui de task
37:        Durée : (temps de départ de task - temps de départ de freespace) + durée de freespace
38:        Cpu : Cpu de freespace ∪ cpu de task
39:      else
40:        On crée un nouvel emplacement new_freespace avec :
41:        Temps de départ : celui de task
42:        Durée : durée de task
43:        Cpu : Cpu de freespace ∪ cpu de task
44:      end if
45:      new_emplacement ← new_emplacement ∪ new_freespace ∪ freespace
46:    else if temps de départ de task < temps de départ de freespace then
47:      if temps final de freespace ≤ temps final de task then
48:        On crée un nouvel emplacement new_freespace avec :
49:        Temps de départ : celui de freespace
50:        Durée : durée de freespace
51:        Cpu : Cpu de freespace ∪ cpu de task
52:      else
53:        On crée un nouvel emplacement new_freespace avec :
54:        Temps de départ : celui de freespace
55:        Durée : (temps de départ de freespace - temps de départ de task) + durée de task
56:        Cpu : Cpu de freespace ∪ cpu de task
57:      end if
58:      new_emplacement ← new_emplacement ∪ new_freespace ∪ freespace
59:    else
60:      if temps final de freespace < temps final de task then
61:        On crée un nouvel emplacement new_freespace avec :
62:        Temps de départ : celui de freespace
63:        Durée : durée de freespace
64:        Cpu : Cpu de freespace ∪ cpu de task
65:        new_emplacement ← new_emplacement ∪ new_freespace ∪ freespace
66:      else if temps final de task < temps final de freespace then
67:        On crée un nouvel emplacement new_freespace avec :
68:        Temps de départ : celui de freespace
69:        Durée : durée de task
70:        Cpu : Cpu de freespace ∪ cpu de task
71:        new_emplacement ← new_emplacement ∪ new_freespace ∪ freespace
72:      else
73:        On modifie l'emplacement freespace avec :
74:        Temps de départ : celui de freespace
75:        Durée : Durée de freespace
76:        Cpu : Cpu de freespace ∪ cpu de task
77:        new_emplacement ← new_emplacement ∪ freespace
78:      end if
79:    end if
80:  end if
81:
82:  return new_emplacement
83: end function
```

Algorithm 6 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
```
