Algorithm 1 Insertion tache

function ADD_TASK(du_t, cpu_t) ▷ duration, number of cpu

On recherche le premier emplacement possible pour la tache avec au minimum la durée du_t et le nombre de cpu cpu_t.

if on a trouvé un emplacement **then**

On récupère le temps de départ $starting\_time\_min$ et la range de cpu $processor\_range\_t$ de l'emplacement.

On soustrait à la range de cpu $processor\_range\_t$, le nombre de cpu cpu_t.

On récupère la liste de tous les emplacements potentiellement impactés dont le temps de départ est inférieur ou égale au temps de départ min $starting\_time\_min$ + la durée du_t .

On ne garde que ceux pour qui :

l'intersection de la range de cpu avec $processor\_range\_t$ est supérieur à 0

and

dont le temps de départ + la durée $\geq starting\_time\_min$

for tous les emplacements restant **do**

On appelle la fonction **cut_freespace**($emplacement, starting\_time\_min, processor\_range\_t$)

end for

for chaque élément unique de $tab\_emplacements$ **do**

Ajouter élément à l'arbre des emplacements.

end for

end if

end function

Algorithm 2 Decoupage Freespace

function CUT_FREESPACE($freespace, start\_time, duration, processor\_range$)
    On supprime $freespace$ de l'arbre des emplacements.

    if temps de départ de $freespace < start\_time$ **then**
        On crée un nouvel emplacement $left\_freespace$ avec :
            Temps de départ : celui de $freespace$
            Durée : $start\_time$ - temps de départ de $freespace$
            Cpu : Cpu de $freespace$
        Ajoute $left\_freespace$ à la table de hashage $tab\_emplacements$.
    end if

    if temps de départ de $freespace < (start\_time + duration)$ **then**
        On crée un nouvel emplacement $right\_freespace$ avec :
            Temps de départ : $(start\_time + duration)$
            Durée : durée de $freespace$ - $(start\_time + duration)$ - temps de départ de $freespace$
            Cpu : Cpu de $freespace$
        Ajoute $right\_freespace$ à la table de hashage $tab\_emplacements$.
    end if

    if l'intersection de la range de cpu de $freespace$ et $processor\_range <$ au nombre de processeurs de $freespace$ **then**
        On crée un nouvel emplacement $new\_freespace$ avec :
            Temps de départ : celui de $freespace$
            Durée : durée de $freespace$
            Cpu : différence entre la range cpu de $freespace$ et $processor\_range$
        Ajoute $new\_freespace$ à la table de hashage $tab\_emplacements$.
    end if

end function

Algorithm 3 Suppression tache

function REMOVE_TASK(st_t, d_t, cpu_t) ▷ Starting time, duration, cpu range
    On calcule le temps final et_t à partir de $(st\_t + d\_t)$

    On récupère la liste de tous les emplacements qui ont un temps de départ inférieur à $(st\_t + d\_t)$

    On ne garde que ceux qui :
        existent dans l'intervalle de temps st_t à et_t
    and
            ont au moins un cpu en commun avec cpu_t
        or
            ont au moins un cpu voisin avec cpu_t

    for tous les emplacements restant **do**
        On appelle la fonction **extend_freespace**($emplacement$)
    end for
end function

Algorithm 4 Augmentation Freespace

```
1: function EXTEND_FREESPACE(freespace)
2:
3:     if direction{freespace} = 1 then
4:         t_next ← (freespace → {starting_time} + freespace → {duration})
5:     else
6:         t_next ← freespace → {starting_time}
7:     end if
8:
9:     new_t ← t_next
10:    while (exist tab{new_t} and (tab{new_t} inter freespace → {cpu}) ≥ freespace → {cpu} → size()) do
11:        t_max ← new_t
12:        new_t ← (new_t + direction{freespace})
13:    end while
14:
15:    if direction{freespace} = 1 then
16:        freespace → {duration} ← (t_next − freespace → {starting_time})
17:    else
18:        freespace → {duration} ← (freespace → {starting_time} − new_t + freespace → {duration})
19:        freespace → {starting_time} ← new_t
20:    end if
21:
22:    TODO :VERIFIER SI L'ESPACE N'EST PAS COMPRIS DANS UN AUTRE
23: end function
```

Algorithm 5 Suppression des Freespaces inutiles

```
1: function REMOVE_UNNECESSARY_FREESPACE(freespace, freespace_list)
2:     for space ← freespace_list do
3:         if (freespace → {cpu} inter space → {cpu}) = (freespace → {cpu} → size) then
4:             if ((freespace → {starting_time} ≤ space → {starting_time}) and (freespace → {starting_time} + freespace → {duration} ≤ space → {starting_time} + space → {duration})) then
5:                 profile_tree → remove(freespace)
6:                 return
7:             end if
8:         end if
9:     end for
10: end function
```