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.

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

On ne garde que ceux pour qui :
    l'intersection de la range de cpu avec $processor\_range\_t$ est non vide
and
    dont le temps de départ + la durée $\geq starting\_time\_min$

$tab\_resultat \leftarrow$ vide

for tous les emplacements restant **do**
    On supprime l'emplacement de l'arbre des emplacements
    $tab\_resultat = tab\_resultat \cup$ **cut_freespace**($emplacement, starting\_time\_min, processor\_range\_t$)
end for

$tab\_final \leftarrow$ vide

for chaque élément de $tab\_resultat$ **do**
    if is_necessary_freespace($element, tab\_resultat$) **then**
        $tab\_final = tab\_final \cup$ élément
    end if
end for

for chaque élément de tab_final **do**
    Ajouter à l'arbre des emplacements $element$
end for

end function

Algorithm 2 Decoupage Freespace

function CUT_FREESPACE($freespace, start\_time, duration, processor\_range$)
    $new\_emplacement \leftarrow$ vide

    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 $freespace$
        $new\_emplacement \leftarrow new\_emplacement \cup left\_freespace$
    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 $processor\_range$
        $new\_emplacement \leftarrow new\_emplacement \cup new\_freespace$
    end if

    On crée un nouvel emplacement $right\_freespace$ avec :
        Temps de départ : $(start\_time + duration)$
        Durée : durée de $freespace$ - (temps de départ de $freespace$ - $start\_time + (start\_time + duration)$)
        Cpu : Cpu de $freespace$
    $new\_emplacement \leftarrow new\_emplacement \cup right\_freespace$

    return $new\_emplacement$
end function

Algorithm 3 Suppression tache

function REMOVE_TASK(st_t, d_t, cpu_t) ▷ Starting time, duration, cpu range
    $et\_t \leftarrow (st\_t + d\_t)$

    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

    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 IS_NECESSARY_FREESPACE(freespace, freespace_list)
2:
3:     for tous les éléments de freespace_list do
4:         if éléments != freespace then
5:             if temps de départ de freespace ≥ temps de départ de space and temps final de freespace ≤ temps final de space then
6:                 if (range cpu de freespace ∩ 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
```