

---

**Algorithm 1** Insertion tache

---

```
1: function ADD_TASK( $du\_t, cpu\_t$ ) ▷ duration, number of cpu
2:    $start\_key \leftarrow [0, du\_t, cpu\_t]$ 
3:    $end\_key \leftarrow [+∞, +∞, +∞]$ 
4:
5:    $profile\_tree \rightarrow \mathbf{node\_loop}(start\_key, end\_key, \{$ 
6:      $starting\_time\_min \leftarrow (freespace \rightarrow \{starting\_time\})$ 
7:      $processor\_range\_t \leftarrow (freespace \rightarrow \{cpu\})$ 
8:     return 1
9:    $\})$ 
10: ▷ Remove the number of cpu for the range
11:    $processor\_range\_t \leftarrow (processor\_range\_t \rightarrow reduce\_to\_basic(cpu\_t))$ 
12:    $start\_key \leftarrow [0, 0, 0]$ 
13:
14:    $end\_key \leftarrow [starting\_time\_min + du\_t, +∞, +∞]$ 
15:    $profile\_tree \rightarrow \mathbf{node\_loop}(start\_key, end\_key, \{$  ▷ Research with constraints
16:     push  $freespace\_impacted, freespace$ 
17:     return 0
18:    $\})$ 
19:
20:   for  $freespace \leftarrow freespace\_impacted$  do
21:     cut_freespace( $freespace, starting\_time\_min, du\_t, processor\_range\_t$ )
22:   end for
23: end function
```

---

---

**Algorithm 2** Decoupage Freespace

---

```
1: function CUT_FREESPACE(freespace, start_time, duration, processor_range)
2:   if (freespace → {cpu} inter processor_range) > 0 then
3:     profile_tree → remove(freespace)
4:
5:     if freespace → {starting_time} < start_time then
6:       new freespace left_freespace ← (freespace → {starting_time}, start_time − freespace →
       {starting_time}, freespace → {cpu})
7:       profile_tree → add(left_freespace)
8:     end if
9:
10:    if freespace → {starting_time} < (start_time + duration) then
11:      new freespace right_freespace ← ((start_time + duration), freespace → {duration} −
      (start_time + duration) − freespace → {starting_time}, freespace → {cpu})
12:      profile_tree → add(right_freespace)
13:    end if
14:
15:    range_test ← (freespace → {cpu} → copy())
16:    range_test → remove(processor_range)
17:
18:    if range_test → size() > 0 then
19:      new freespacenew_freespace ← (freespace → {starting_time}, freespace →
      {duration}, range_test)
20:      profile_tree → add(new_freespace)
21:    end if
22:
23:  end if
24:
25: end function
```

---

---

**Algorithm 3** Suppression tache

---

```
1: function REMOVE_TASK( $st\_t, d\_t, cpu\_t$ ) ▷ Starting time, duration, cpu range
2:    $et\_t \leftarrow (st\_t + d\_t)$  ▷ End time of job
3:
4:   for  $t \leftarrow st\_t$  to  $et\_t$  do ▷ Update cpu range for the job placement
5:     add  $tab\{t\} \leftarrow cpu\_t$ 
6:   end for
7:
8:    $start\_key \leftarrow [0, 0, 0]$ 
9:    $end\_key \leftarrow [st\_t + d\_t, +\infty, +\infty]$ 
10:
11:    $profile\_tree \rightarrow \mathbf{node\_loop}(start\_key, end\_key, \{$ 
12:
13:   if ( $cpu\_t$  inter  $freespace\{cpu\}) > 0$  then
14:     push  $freespace\_impacted, freespace$ 
15:      $st\_t2 \leftarrow (freespace \rightarrow \{starting\_time\})$ 
16:      $et\_t2 \leftarrow (freespace \rightarrow \{starting\_time\}) + freespace \rightarrow \{duration\})$ 
17:
18:     if ( $st\_t \leq st\_t2$  and  $st\_t2 \leq et\_t$ ) then
19:        $direction\{freespace\} \leftarrow -1$ 
20:     else if ( $st\_t2 \leq st\_t$  and  $st\_t \leq et\_t2$ ) then
21:        $direction\{freespace\} \leftarrow 1$ 
22:     end if
23:
24:     if exist  $direction\{freespace\}$  then
25:       for  $t \leftarrow st\_t2$  to  $et\_t2$  do ▷ Update cpu range for freespace placement
26:
27:         if exist  $tab\{t\}$  then
28:           add  $tab\{t\} \leftarrow (freespace \rightarrow \{cpu\} \mathbf{inter} tab\{t\})$ 
29:         else
30:           add  $tab\{t\} \leftarrow (freespace \rightarrow \{cpu\})$ 
31:         end if
32:
33:       end for
34:     end if
35:   end if
36: }
37:
38:   for  $freespace \leftarrow freespace\_impacted$  do
39:     extend\_freespace( $freespace$ )
40:   end for
41: end function
```

---

---

**Algorithm 4** Augmentation Freespace

---

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

---

---

**Algorithm 5** Suppression des Freespaces inutiles

---

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

---