Chapitre 1

Two FPTAS for the problem $2\ |tree,\ merging|\ C_{max}$

1.1 DP algorithm

A state will have four components $[a, b, j, k]$:
— a : completion time on machine M1
— b : completion time on machine M2
— j : last tile on M1
— k : last tile on M2

Time complexity Let P be $\sum_{i=1}^{\mathcal{N}} p_i$. Thanks to the rule saying we keep only the state with the smallest b-value for each triplet (a, j, k), there is only one possible value for b while there are $P+1$ different values for a and m different values for j and k which implies that we generate at most $\mathcal{O}(m^2.P)$ different states for one iteration.
As there are m iterations in total, this gives us a time complexity of $\mathcal{O}(m^3.P)$.

Algorithm 1 The dynamic programming

$\chi_0 \leftarrow \{[0, 0, 0, 0]\}$
for i from 1 to m **do** ▷ m is the number of tiles.
  $\chi_i \leftarrow \emptyset$
  for all $[a, b, j, k]$ in χ_{i-1} **do**
    $\chi_i \leftarrow \chi_i \cup \{[a + M(j, i), b, i, k]\}$ ▷ We assign the tile i to $M1$
    $\chi_i \leftarrow \chi_i \cup \{[a, b + M(k, i), j, i]\}$ ▷ We assign the tile i to $M2$
  end for
  For every triplet (a, j, k), keep the one with the smallest b
  Remove χ_{i-1}
end for
return $\min_{[a,b,j,k]\in\chi_m}\{max\{a, b\}\}$

1.2 The FPTAS

Time complexity a and b vary between 0 and P, thus there are should be $P+1$ different possible values for each. But as we keep only one representative per interval and as each interval length is equal to δ there are in fact $\frac{P+1}{\delta}$ different possible values for a. As for b, as we only keep the state with the smallest b, there is only one possible value for it and so there are $\mathcal{O}(\frac{P}{\delta})$ value for the couple (a, b).
Furthermore, there are m tiles in total which means that for one iteration, there are at most $m^2 * (\frac{P}{\delta})^2$ generated states $[a, b, j, k]$.
As there are m iterations, we generate at most $m * (m^2 * (\frac{P}{\delta})^2) = \frac{m^3 P^2}{\delta^2} = \frac{4m^5}{\epsilon^2}$.
$\rightarrow$ The time complexity of the modified algorithm is $\mathcal{O}(\frac{m^5}{\epsilon^2})$.

Algorithm 2 FPTAS

$\chi_0^\# \leftarrow \{[0,0,0,0]\}$
for i from 1 to m **do** ▷ m is the number of tiles.
  $\chi_i^\# \leftarrow \emptyset$
  for all $[a^\#, b^\#, j, k]$ in $\chi_{i-1}^\#$ **do**
    $\chi_i^\# \leftarrow \chi_i^\# \cup \{[a^\# + M(j,i), b^\#, i, k]\}$ ▷ We assign tile i to $M1$
    $\chi_i^\# \leftarrow \chi_i^\# \cup \{[a^\#, b^\# + M(k,i), j, i]\}$ ▷ We assign tile i to $M2$
  end for
  For every $[I_z, I_{z'}, j, k]$, keep only one state $[a^\#, b^\#, j, k]$ such as $a^\# \in I_z$ and $b^\# \in I_{z'}$. $(z, z' \in \{0, 1, 2, ..., z_0\})$ If several states can be chosen, keep the one with the smallest a.
  Remove $\chi_{i-1}^\#$
end for
return $\min_{[a^\#, b^\#, j, k] \in \chi_m^\#} \{max\{a^\#, b^\#\}\}$

1.3 The improved FPTAS

The principle The FPTAS is still based on a dynamic programming algorithm. What changes from the previous FPTAS is the quadruplet chosen to represent a solution. The new state of the new dynamic programming algorithm is defined as follows $[a, b, k, \alpha]$ with :

- a : the completion time on machine M1
- b : the completion time on machine M2
- α : a boolean that says if the tile which was scheduled before was assigned to machine $M1$ or to machine $M2$.
- k : the index of the last tile assigned to the machine where the tile $i-1$ was not scheduled

Algorithm 3 The second dynamic programming algorithm

$\chi_0 \leftarrow \{[0,0,0,0]\}$ ▷ t_0 : artificial tile with $|t_0| = 0$
for all the tiles **do**
  $\chi_i \leftarrow \emptyset$
  for all $[a, b, k, \alpha]$ in χ_{i-1} **do**
    if $\alpha = 1$ **then** ▷ k was the last tile on $M2$
      $\chi_i \leftarrow \chi_i \cup \{[a + M(i, i-1), b, k, 1]\}$ ▷ We schedule t_i on $M1$
      $\chi_i \leftarrow \chi_i \cup \{[a, b + M(i, k), i-1, 0]\}$ ▷ We schedule t_i on $M2$
    else
      $\chi_i \leftarrow \chi_i \cup \{[a + M(i, k), b, i-1, 1]\}$ ▷ We schedule t_i on $M1$
      $\chi_i \leftarrow \chi_i \cup \{[a, b + M(i, i-1), k, 0]\}$ ▷ We schedule t_i on $M2$
    end if
  end for
  Dominance rule : for every triplets (a, j, k), keep only the one with the smallest b-value
  Delete χ_{i-1}.
end for
Return $\min_{[a,b,j,k] \in \chi_m} \{max\{a, b\}\}$

1.4 The matrix

The principle A cell in the matrix M (used in the algorithms) represents the number of symbols we need to add to a machine when we want to assign a tile to it. Of course, this number depends on the tile we are considering but it also depends on the tiles already scheduled on this machine.
Thanks to the hierarchy into the tiles and if the tiles are processed in a correct order (from left to right or from right to left in the tree), it is easy to see that we only need to know the last tile assigned to a machine in order to know how many symbols we will have to add to this machine to perform/schedule the new tile.

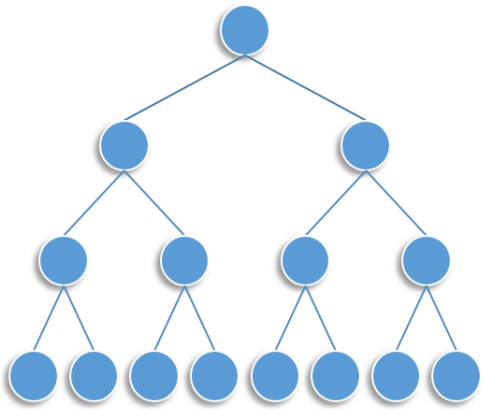

FIGURE 1.1 – More general example of the hierarchy with a example of a tile