5. *Planificació.*

Tenim un conjunt de n tasques. La tasca i es descriu per un parell $T_i = (s_i, d_i)$ on s_i es el seu temps de disponibilitat i d_i la seva duració. L'objectiu es planificar totes les tasques en un processador de manera que: (a) cap tasca s'executa abans del seu temps de disponibilitat, (b) les tasques s'executen sense interrumció, i (d) el temps de finalització del procesament de totes les tasques sigui mínim.

Se sap que, quan és possible interrompre qualsevol tasca en execució i reiniciar-le més tard des del punt d'aturada, l'algorisme voraç que executa en cada punt de temps la tasca (disponible) més propera al seu temps de finalització, aconsegueix el desitjat mínim. Anomenarem a aquest algorisme Voraç1.

Considereu el procediment següent:

(a) Executa Voraç1.

(b) Ordena les tasques en l'ordre ascendent del seu temps de finalització d'acord amb la solució proporcionada per versió Voraç1.

(c) Les tasues s'executeu en aquest ordre i sense interrupció, afegint el temps d'espera necesari fins que la tasca estigui disponible.

Demostreu que aquest algorisme es una 2-aproximació pel problema plantejat.

Sea OPT la solución óptima donde las tareas estan ordenadas en orden creciente de su tiempo de finalización. También, sea A la solución producida por el algoritmo propuesto. Si t_i es el tiempo de finalización y s_i es el tiempo de inicio de la tarea i en A, entonces la duración de la tarea i en A es d_i = t_i - s_i.

Observamos que el ordenamiento de las tareas de A en orden creciente de su tiempo de finalización, que se realiza en el paso (b), implica que t_i ≤ t_{i+1} para todo i = 1, 2, ..., n-1. Esto se debe a que el algoritmo Voraç1 garantiza que la tarea i termina antes que la tarea i+1 en A, y por lo tanto, t_i ≤ t_{i+1}.

A continuación, consideramos la tarea i en A. Si s_i ≥ f_{i-1}, entonces la tarea i no entra en conflicto con la tarea i-1, y por lo tanto, tenemos d_i = t_i - s_i = d_i, lo que implica que la duración de la tarea i en A es la misma que su duración en OPT.

Si s_i < d_{i-1}, entonces la tarea i entra en conflicto con la tarea i-1 en A. En este caso, podemos retrasar la tarea i hasta que la tarea i-1 haya terminado en A. Por lo tanto, asignamos s_i = d_{i-1}, y luego, t_i = s_i + d_i. Esto implica que d_i ≤ t_i - s_i ≤ d_{i-1} + d_i, lo que significa que la duración de la tarea i en A es como máximo dos veces su duración en OPT.

Finalmente, como la duración de cada tarea en A es como máximo dos veces su duración en OPT, la duración total de A es como máximo dos veces la duración total de OPT. Como A cumple con todas las restricciones del problema, es una solución factible, lo que significa que el algoritmo propuesto es una 2-aproximación para el problema de planificación dado.