Scheduling

Jesús García López de Lacalle

Ejemplo 1: Considerar la CPU de un ordenador que debe procesar una secuencia de trabajos que llegan en el tiempo.

¿En qué orden debe procesarlos para minimizar, en promedio, el tiempo de espera de los trabajos desde su llegada a su compleción?

Ejemplos

Ejemplo 2: Considerar un equipo de astronautas preparando la reentrada de su nave a la atmósfera. Existe un conjunto de tareas que deben realizar antes de la reentrada, de forma que cada una es realizada exactamente por un astronauta y hay ciertas tareas que no pueden empezarse antes de haber completado otras.

¿Qué tareas debe realizar cada astronauta y en qué orden, de manera que se completen en el menor tiempo posible?

Ejemplos

Ejemplo 3: Considerar una fábrica que produce diferentes tipos de dispositivos que deben procesarse en las máquinas 1, 2, …, m. La fábrica recibe pedidos de lotes de dispositivos con fecha de entrega.

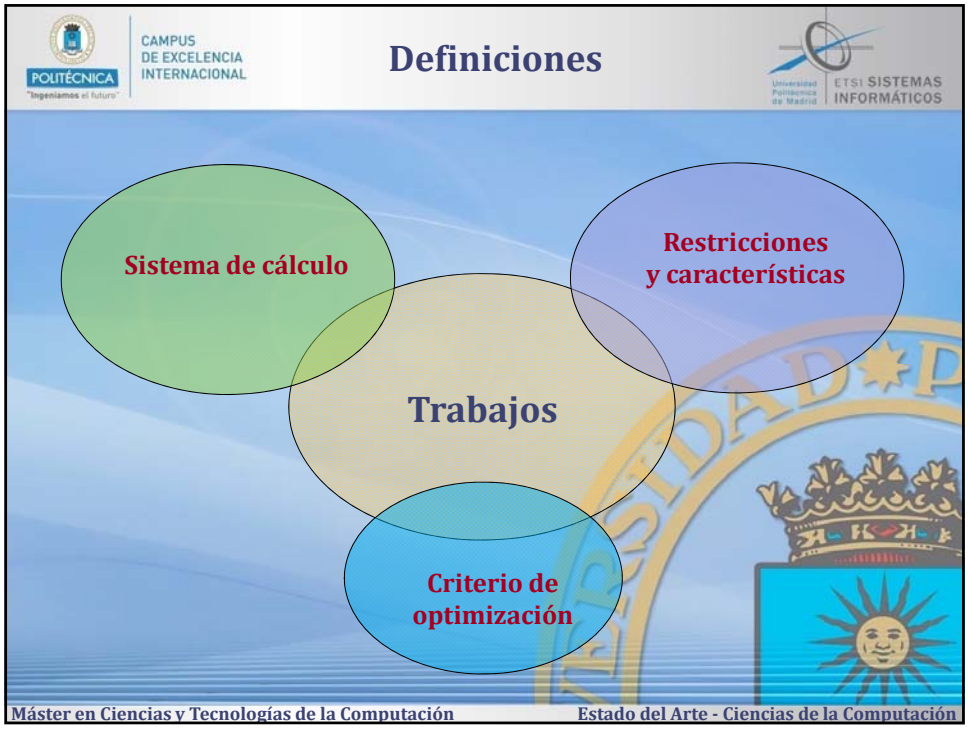
¿En qué orden deben trabajar las máquinas y sobre qué dispositivos para entregar el mayor número de pedidos a tiempo?

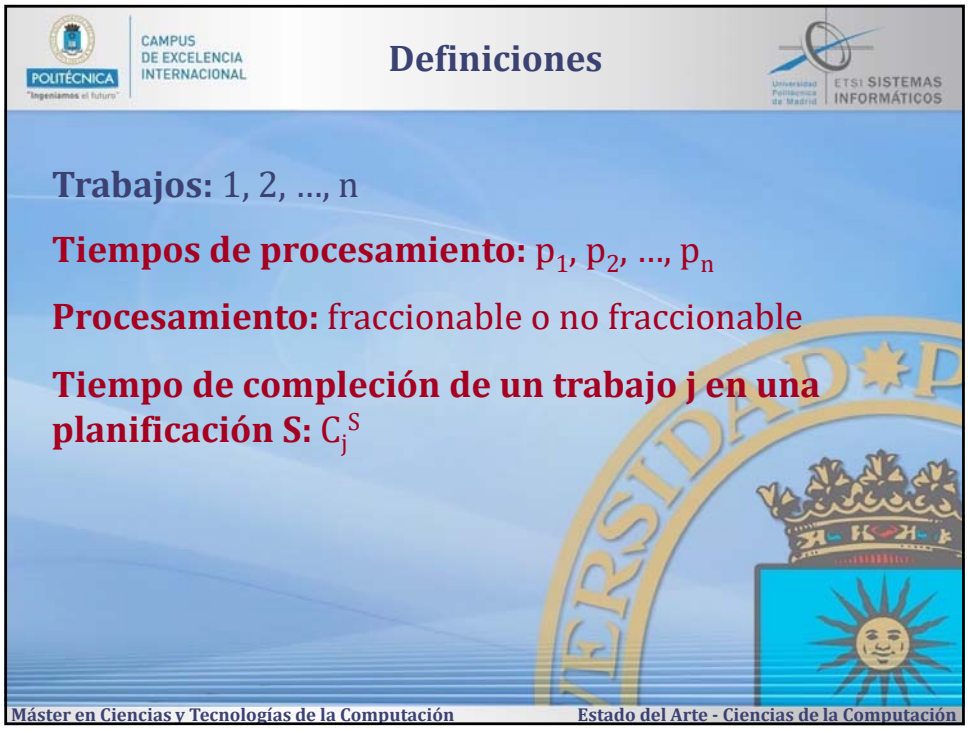
Aplicaciones

- Sistemas Operativos
- Programación Concurrente
- Sistemas Distribuidos
- Comunicaciones en Redes
- Bases de Datos
- Logística: transporte, sistemas productivos, almacenamiento…

Índice

- Definiciones
- Notación
- Reglas de prioridad: problemas
  - $1 \mid\mid \Sigma C_j$, $1 \mid\mid \Sigma w_j C_j$, $1 \mid r_j, frac \mid \Sigma C_j$
  - $1 \mid\mid L_{max}$, $1 \mid r_j, frac \mid L_{max}$
- Más definiciones
- Reglas de prioridad: problema $F2 \mid\mid C_{max}$
- Máquinas paralelas idénticas:
  - Algoritmos aproximados: $P \mid\mid C_{max}$, $P \mid prec \mid C_{max}$
  - Problema $P \mid prec \mid C_{max}$


CAMPUS
DE EXCELENCIA
INTERNACIONAL
Definiciones
ETSI SISTEMAS
INFORMÁTICOS
Sistema de cálculo
Restricciones
y características
Trabajos
Criterio de
optimización
Máster en Ciencias y Tecnologías de la Computación
Estado del Arte - Ciencias de la Computación


CAMPUS
DE EXCELENCIA
INTERNACIONAL
Definiciones
ETSI SISTEMAS
INFORMÁTICOS
Trabajos: 1, 2, …, n
Tiempos de procesamiento: p_1, p_2, …, p_n
Procesamiento: fraccionable o no fraccionable
Tiempo de compleción de un trabajo j en una planificación S: C_j^S
Máster en Ciencias y Tecnologías de la Computación
Estado del Arte - Ciencias de la Computación

Definiciones

Sistemas de cálculo: 1, 2, …, m

Máquinas paralelas idénticas:

→ tiempo del trabajo j = p_j para cualquier máquina

Máquinas uniformemente relacionadas:

→ tiempo del trabajo j en la máquina i = p_j / s_i

Máquinas paralelas no relacionadas:

→ tiempo del trabajo j en la máquina i = p_j / s_{ij}

…

Definiciones

Restricciones y características

Procesamiento: fraccionable o no fraccionable

Tiempo de llegada de los trabajos: $r_1, r_2, \ldots, r_n$

Tiempo de entrega de los trabajos: $d_1, d_2, \ldots, d_n$

Relaciones de precedencia entre los trabajos:

el trabajo j debe realizarse antes que el i (j→i)

Definiciones

Criterio de optimización

Tiempo máximo: $C_{max}^S = \max_j C_j^S \rightarrow C_{min} = \min_S C_{max}^S$

Tiempo promedio: $(\Sigma_j C_j^S)/n \rightarrow \min_S (\Sigma_j C_j^S)$

Tiempo promedio con pesos: $(\Sigma_j w_j C_j^S)/n \rightarrow \min_S (\Sigma_j w_j C_j^S)$

Máxima demora: $L_{max}^S = \max_j L_j^S$ donde $L_j^S = C_j^S - d_j$

Número de demoras: $\Sigma_j U_j^S$

donde $U_j^S = 0$ si $C_j^S \leq d_j$ y $U_j^S = 1$ en otro caso

Número de demoras con pesos: $\Sigma_j w_j U_j^S$

Notación

Problema de optimización-Scheduling

$$\alpha \mid \beta \mid \gamma$$

α: sistemas de cálculo

β: restricciones y características

γ: criterio de optimización

Notación

Ejemplo 1: Considerar la CPU de un ordenador que debe procesar una secuencia de trabajos que llegan en el tiempo.

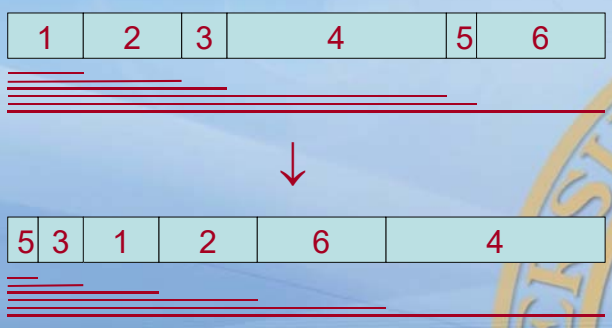
¿En qué orden debe procesarlos para minimizar, en promedio, el tiempo de espera de los trabajos desde su llegada a su compleción?

$$1 \mid r_j \mid \Sigma C_j$$

Reglas de prioridad

La prioridad de un trabajo no depende del resto

Problema $1 \mid \mid \Sigma C_j$



Reglas de prioridad

Problema 1 | | ΣC_j

Algoritmo: menor tiempo de procesamiento (SPT)

Teorema 1: SPT da una solución óptima para 1 | | ΣC_j

Demostración:

Supongamos, por reducción al absurdo, que existe una planificación óptima P en la que los trabajos no están ordenados de menor a mayor tiempo de procesamiento.

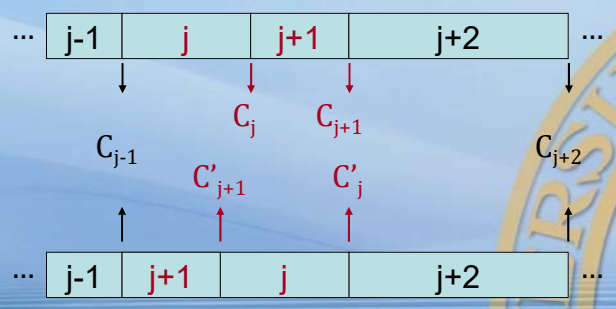
Supongamos, renumerando los trabajos si es necesario, que el orden de procesamiento de los trabajos en dicha planificación es 1, 2, … n.

Reglas de prioridad

Entonces existe un j, con $1 \le j < n$, tal que $p_j > p_{j+1}$.

Si permutamos los trabajos j y j+1 obtenemos una planificación P' en la que solo cambian los tiempos de dichos trabajos:

$C'_j = C_{j+1}$ y $C'_{j+1} = C_j - (p_j - p_{j+1}) < C_j$



Reglas de prioridad

Entonces existe un i, con $1 \leq i < n$, tal que $p_j > p_{j+1}$.

Si permutamos los trabajos j y j+1 obtenemos una planificación P' en la que solo cambian los tiempos de dichos trabajos:

$C'_j = C_{j+1}$ y $C'_{j+1} = C_j - (p_j - p_{j+1}) < C_j$

Por tanto, la suma de tiempos de los trabajos en la planificación P' verifica que:

$\Sigma_k C'_k = \Sigma_k C_k - (p_j - p_{j+1}) < \Sigma_k C_k$

Este resultado contradice el hecho de que la planificación P es óptima y concluye la demostración. ■

Reglas de prioridad

Problema 1 | | $\Sigma w_j C_j$

Algoritmo: mayor valor de w_j / p_j

Teorema 2: el algoritmo anterior da una solución óptima para el problema 1 | | $\Sigma w_j C_j$

Demostración: análoga a la del problema 1 | | ΣC_j

Reglas de prioridad

Problema 1 | | L_{max}

Algoritmo: menor tiempo de entrega d_j

Teorema 3: el algoritmo anterior da una solución óptima para el problema 1 | | L_{max}

Demostración:

Supongamos, renumerando los trabajos si es preciso, que $d_1 \leq d_2 \leq \ldots \leq d_n$

Consideramos de entre todas las planificaciones óptimas la que tiene menos inversiones, P, donde una inversión es un par de trabajos consecutivos k y j tales que j < k.

Reglas de prioridad

Supongamos, por reducción al absurdo, que en P los trabajos no están ordenados de menor a mayor tiempo de entrega.

Esto significa que existen k y j consecutivos en P tales que $d_j < d_k$.

Consideramos la planificación P' que se obtiene permutando los trabajos k y j. Entonces se cumple:

$$\max(L_j, L_k) = C_j - d_j > \max(C_j - d_k, C'_j - d_j) = \max(L'_k, L'_j)$$

Si $L'_{max} < L_{max}$ obtenemos la contradicción de que P no es óptima.

Si $L'_{max} = L_{max}$ obtenemos la contradicción de que P no es la solución óptima con menos inversiones. ■

Reglas de prioridad

Problema 1 | r_j, frac | ΣC_j

Algoritmo: menor tiempo pendiente de procesamiento (SRPT)

Teorema 4: el algoritmo anterior da una solución óptima para el problema 1 | r_j, frac | ΣC_j

Demostración: análoga a la del problema 1 | | ΣC_j

Reglas de prioridad

Problema 1 | r_j, frac | L_{max}

Algoritmo: menor tiempo pendiente de entrega d_j

Teorema 5: el algoritmo anterior da una solución óptima para el problema 1 | r_j, frac | L_{max}

Demostración: análoga a la del problema 1 | | L_{max}

Ejercicio

¿Se pueden adaptar las reglas de prioridad para resolver los siguientes problemas? ¿Por qué?

- $1 \mid r_j \mid \Sigma C_j$
- $1 \mid r_j \mid L_{max}$
- $1 \mid r_j, frac \mid \Sigma w_j C_j$

Más definiciones

Sistemas de cálculo: 1, 2, ..., m

Máquinas paralelas idénticas (P):

→ tiempo del trabajo j = p_j para cualquier máquina

Máquinas uniformemente relacionadas (Q):

→ tiempo del trabajo j en la máquina i = p_j / s_i

Máquinas paralelas no relacionadas (R):

→ tiempo del trabajo j en la máquina i = p_j / s_{ij}

Más definiciones

Sistemas de cálculo: 1, 2, …, m

Cada trabajo debe procesarse en todas las máquinas

Los trabajos no pueden procesarse en varias máquinas a la vez

Cadenas libres (O):

→ el orden de procesamiento de cada trabajo es libre

Cadenas independientes (J):

→ el orden de procesamiento de cada trabajo está determinado

Cadenas idénticas (F):

→ el orden de procesamiento de todos los trabajos es el mismo

Reglas de prioridad

Problema F2 | | C_{max}

Tiempos de procesamiento del trabajo j: (a_j, b_j)

Regla de prioridad (Johnson):

Clasificamos los trabajos en $\mathbf{A=\{j \mid a_j \leq b_j\}}$ y $\mathbf{B=\{j \mid a_j > b_j\}}$

Orden de los trabajos para las dos máquinas:

A orden creciente de a_j – B orden decreciente de b_j

Definición: Una planificación con el mismo orden de los trabajos en todas las máquinas se llama **permutación**

Reglas de prioridad

Lema 1: F2 | | C_{max} tiene como solución una permutación

Demostración:

Consideramos, renumerando los trabajos si es necesario, que 1, 2, …, n es el orden de procesamiento de la máquina 1

Supongamos que el k precede inmediatamente a j en la máquina 2, pero j < k.

Sea t el tiempo en el que se inicia k en la máquina 2.

Tanto j como k se han procesado completamente en la máquina 1 en tiempo t.

Por tanto podemos cambiar el orden de k y j en la máquina 2, manteniendo el mismo valor de C_{max}.

Repetimos el proceso hasta conseguir una permutación óptima. ■

Reglas de prioridad

Teorema 6: La regla de Johnson de una solución óptima al problema F2 | | C_{max}

Demostración: es fácil usando el lema anterior

Máquinas paralelas idénticas

Problema $P||C_{max}$

Muchos problemas que con 1 máquina se resuelven mediante reglas de prioridad pasan a ser NP-duros

El objetivo es aproximar $\min_P C^P_{max} = \mathbf{C^*_{max}}$

Por tanto, nos centramos en **algoritmos aproximados**:

Encontrar P tal que $C^*_{max} \leq C^P_{max} \leq \alpha C^*_{max}$

P es una α-aproximación (α > 1)

Cuando α tiende a 1 la complejidad para calcular la planificación P tenderá a ser exponencial.

Máquinas paralelas idénticas

Estretegia para obtener α-aproximaciones

1- Encontrar una cota inferior C de C^*_{max}

$C = (\Sigma p_j)/m \leq C^*_{max}$

Para todo j, $C = p_j \leq C^*_{max}$

2- Encontrar un algoritmo que obtenga P tal que:

$C^P_{max} \leq \alpha C$

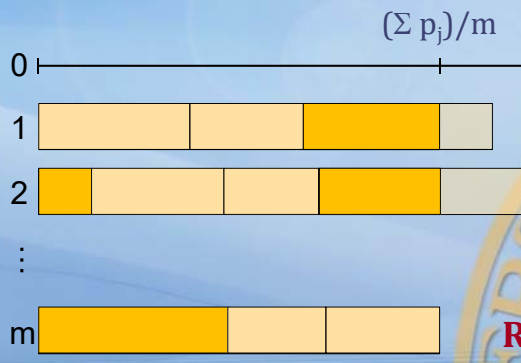
Entonces se verifica

$C^*_{max} \leq C^P_{max} \leq \alpha C \leq \alpha C^*_{max}$

Máquinas paralelas idénticas

Problema P | frac | C_{max}

¿Se puede resolver este problema de forma exacta?



Sí se puede

Fragmentando a lo sumo m-1 trabajos

Regla de McNaughton

CAMPUS DE EXCELENCIA INTERNACIONAL
POLITÉCNICA "Ingeniamos el futuro"
ETSI SISTEMAS INFORMÁTICOS

Máquinas paralelas idénticas

Problema P | | C_{max}

Regla de priorización: cualquier trabajo (LS).

Combinamos las reglas de priorización con una estrategia *greedy* en la que se asigna el siguiente trabajo pendiente a la primera máquina que está libre.

Teorema 7: LS es una 2-aproximación de P | | C_{max}

Demostración:

Sea j el último trabajo finalizado y s_j el momento en el que se inicia dicho trabajo.

Entonces se cumple que $C_{max} = s_j + p_j$

Máster en Ciencias y Tecnologías de la Computación
Estado del Arte - Ciencias de la Computación

Máquinas paralelas idénticas

Todas las máquinas están ocupadas hasta s_j

El tiempo máximo en el que todas las máquinas están ocupadas es $(\Sigma p_j)/m$

Por tanto, $s_j \leq (\Sigma p_j)/m$

Finalmente obtenemos

$$C_{max} = s_j + p_j \leq (\Sigma p_j)/m + p_j$$
$$\leq C^*_{max} + C^*_{max} = 2C^*_{max}$$

■

Máquinas paralelas idénticas

Problema $P||C_{max}$

Regla de priorización: trabajo más largo (LPT).

Teorema 8: LPT es una 4/3-aproximación de $P||C_{max}$

Demostración:

Suponemos que el último trabajo en terminar, j, es el último en empezar y llamamos s_j al momento en que se inicia el trabajo j.

Si no es así suprimimos todos los trabajos que empiezan después de s_j.

Máquinas paralelas idénticas

Se cumple que $C'_{max} = C_{max}$.

Por tanto, basta demostrar el teorema suponiendo que j es el último trabajo en empezar.

Obviamente se cumple que $C_{max} = s_j + p_j$.

Por otro lado, todos los equipos trabajan al menos hasta s_j y entonces se verifica que $s_j \leq C^*_{max}$.

Dada la regla de priorización utilizada también se cumple que $p_j = p_{min}$.

Por tanto, $C_{max} \leq C^*_{max} + p_{min}$.

Máquinas paralelas idénticas

Si $p_{min} \leq C^*_{max}/3$:

$C_{max} \leq C^*_{max} + p_{min} \leq C^*_{max} + C^*_{max}/3 = (4/3)\, C^*_{max}$

Si $p_{min} > C^*_{max}/3$: entonces para todo k $p_k > C^*_{max}/3$

Por tanto, en una planificación óptima cada máquina realiza como mucho dos trabajos

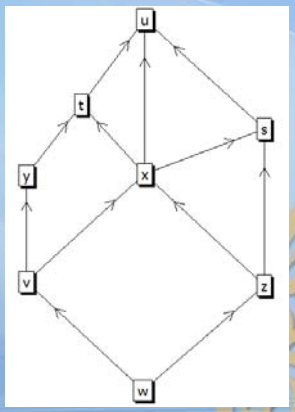
Si $n \leq m$ la planificación obtenida es óptima, es decir, $C_{max} = C^*_{max} \leq (4/3)\, C^*_{max}$

Si $m < n \leq 2m$ la planificación obtenida también es óptima. ■

Máquinas paralelas idénticas

Problema P | prec | C_{max}

Ejemplo:



Máquinas paralelas idénticas

Problema P | prec | C_{max}

Regla de priorización: cualquier trabajo (LS).

Cota inferior: si $j_1 \rightarrow j_2 \rightarrow \ldots \rightarrow j_k$ es una cadena entonces:

$$\Sigma_i \, p_{j_i} \leq C^*_{max}$$

Teorema 9: LS es una 2-aproximación de P | prec | C_{max}

Demostración:

Sea j_1 el último trabajo en terminar, j_2 el último predecesor de j_1 en terminar y así sucesivamente j_{s+1} el último predecesor de j_s en terminar.

Máquinas paralelas idénticas

Sea E={ j_1 j_2, … j_k} el conjunto de trabajos obtenido.

Dividimos el intervalo de tiempo I=[0, C_{max}] en dos conjuntos:

A={t∈I | en t un trabajo de E se está ejecutando} y
B={t∈I | t∉A}.

Si t∈B en t todas las máquinas están trabajando.

$\mathbf{C_{max}} = |A|+|B| \leq \Sigma_{j \in E} p_j + (\Sigma p_j)/m \leq \mathbf{2\ C^*_{max}}$

Máquinas paralelas idénticas

Problema P | prec | C_{max}

Si todos los tiempos de procesamiento p_j=1:

El problema es polinomial si m=2

El problema es NP-completo si m ≥ 3

Es el problema más famoso de Scheduling

¿Para qué grafos el problema es polinomial?