SCHEDULING PARA MINIMIZAR EL TIEMPO DE TERMINACION:
ALGORITMOS DE APROXIMACION EN LINEA Y FUERA DE LINEA.

ALGORITMOS DE APROXIMACION-ρ PARA DIFERENTES MODELOS DE SCHEDULING.
(Leslie A. Hall)

- Asignar-por- $\overline{C}_j$
  - 1 | Prec | Σ wj Cj
  - 1 | rj, Prec | Σ wj Cj
  - 1 | rj | Σ wj Cj
- Asignar-premptivamenete-por- $\overline{C}_j$
  - 1 | rj, Prec, Pmtn | Σ wj Cj
  - 1 | rj, Prec, Pj=1 | Σ wj Cj
- Comienzo-de-trabajos-por- $\overline{C}_j$
  - P | rj | Σ wj Cj
- Lista-Scheduling-premptiva-por- $\overline{C}_j$
  - P | rj, Prec, Pmtn | Σ wj Cj
  - P | rj, Prec, Pj=1 | Σ wj Cj
  - p | Prec, Pmtn | Σ wj Cj
  - P | Prec, Pj=1 | Σ wj Cj
- Intervalo-asignado-por- $\overline{C}_j$
  - P | rj, Prec | Σ wj Cj
- Algoritmo de intervalo indicado-LP +Gloton-con-intervalo-LP
  - R | rij | Σ wj Cj
- Gloton-con-intervalo fuera-de-linea --> en-linea
  - 1 | rj | Σ wj Cj
  - P | rj | Σ wj Cj
  - R | rij | Σ wj Cj

Objetivo : Presentar la comprobación de los algoritmos de aproximacion-ρ Fuera-de-linea y En-linea propuestos para diversos modelos de Scheduling en la obtención de la minimizacion del peso total en tiempo de terminación.

Algunas definiciones importantes:

Se dice que un algoritmo A es un algoritmo de aproximacion-ρ si, para cada instancia de un modelo I, A(I) esta dentro de un factor de ρ del valor optimo. [Lenstra].

Algoritmo en Línea: Se construye la asignación de trabajos como un producto del tiempo, y no se conoce la existencia del trabajo hasta su tiempo tiempo de salida para ser procesado.

Algoritmo Fuera de Línea: Todos los trabajos se conocen al inicio de la asignación (tratados en este articulo en su mayoría).

Para obtener cotas superiores con respecto a la solución optima[Beasley]: Esencialmente es a través de métodos heurísticos (Simulated Annealing, Tabu Search, Ga, Artificial Neural Networks, etc.).

Para obtener cotas inferiores con respecto a la solución optima[Beasley]: La Relajación LP. Consiste en tomar un entero o mezclas de enteros en la formulacion de programación del problema para relajar los requerimientos de integridad de las variables. Esto da un programa lineal que se puede resolver por algún algoritmo lineal (ej. simpex). Ejemplo: Dado un problema P

P minimizar cx sujeto a Ax >=b Bx>=d x ε (0,1)	Para generar una cota inferior sobre la solución optima. Se reemplazan las restricciones de integridad por su relajación para obtener un programa lineal	P minimizar cx sujeto a Ax >=b Bx>=d 0<= x <= 1

NOMENCLATURA GENERAL.

Cj Denota el tiempo de terminación del trabajo j.

C*j Denota el tiempo de terminación del trabajo j en alguna asignación optima.

$\overline{C}$ Denota el tiempo de terminación del trabajo j asignado por la relajación.

$\tilde{C}j$ Denota el tiempo de terminación del trabajo j asignado por el algoritmo de aproximacion-ρ que se considera.

$j \prec k$ Es una restricción de precedencia que indica que el trabajo k no puede comenzar si el trabajo j no ha terminado.

NOMENCLATURA PARA LOS MODELOS.

Un problema de Scheduling se puede representar por un modelo a través de tres campos $\alpha \mid \beta \mid \gamma$ El campo α describe el ambiente de la maquina y contiene una única entrada. El campo β provee detalles de las características del procesamiento y puede permanecer sin nada, una o múltiples entradas. El campo γ contiene el objetivo a minimizado.

n Denota a el numero de trabajos.
j Se refiere a un trabajo.
m Denota el numero de maquinas.
i Se refiere a una maquina.

Para el campo a.

1Maquina única. El caso de maquina única es el mas simple de todos los posibles ambientes de maquinas.

P Maquinas idénticas en paralelo. Se tienen m maquinas idénticas en paralelo.
R Maquinas en paralelo no relacionadas. Este ambiente es una generalización del visto con anterioridad. Se tienen m maquinas en paralelo. La maquina i puede procesar el trabajo j a una velocidad vij. El tiempo pij que el trabajo j gastara sobre la maquina i es, asumiendo que este es procesado solamente sobre la maquina i, igual a pj/vij.

Para el campo b.

pj Tiempo de procesamiento. El pij representa el tiempo de procesamiento del trabajo j sobre la maquina.

rj Tiempos de salida. El trabajo j no puede empezar su procesamiento antes de su tiempo de descarga rj.

D Deadline. Cuando se tiene que convenir absolutamente en el tiempo estipulado, esto es conocido como un limite absoluto

wj Peso. El peso wj del trabajo j es básicamente un factor de prioridad, denotando la importancia del trabajo j relativa a los otros trabajos en el sistema.

pmtn Premptiones (Derechos de prioridad). Al scheduler se le es permitido interrumpir el procesamiento de un trabajo (asegura el derecho de prioridad "preempt") en algún tiempo y pone un trabajo diferente sobre la maquina. Cuando un trabajo con derecho de prioridad es regresado a la maquina (o sobre otra maquina, en el caso de maquinas en paralelo), este solo necesita la maquina para su restante (remaining) tiempo de procesamiento.

prec Restricciones de precedencia. Requiere que uno o mas trabajos tengan que ser completados antes de que a otros trabajos les sean permitidos comenzar su procesamiento.

Para el campo g

Σwjcj Peso total en el tiempo de terminación. Es la suma de los pesos de los tiempos de terminación de los n trabajos.

MODELOS DE ESTUDIO

Modelo	Conocido	Nuevo	Conocido	Nuevo
1 | prec | ΣwjCj	O(log n log log Σj wj)	2	--	--
1 | rj, prec, pj = 1| ΣwjCj	--	2	--	--
1 | rj, prec, pmtn | ΣwjCj	--	2	--	--
1 | rj | ΣwjCj	16 + ε	3	--	3 + ε
1 | rj, prec | ΣwjCj	--	3	--	--
P | prec, pj = 1 | ΣwjCj	--	3 - 1/m	--	--
P | prec, pmtn | ΣwjCj	--	3 - 1/m	--	--
P | rj, prec, pj = 1 | ΣwjCj	--	3	--	--
P | rj, prec, pmtn | ΣwjCj	--	3	--	--
P | rj | ΣwjCj	24 + ε	4 - 1/m	--	4 + ε
P | rj, prec | ΣwjCj	--	7	--	--
R | rij | ΣwjCj	$O(\log^2 n)$	16/3	--	8

Tabla 1. Un resumen de garantías de desempeño para la minimizacion del **peso total en tiempo de terminación.** Las columnas con etiqueta "Conocido" lista la mejor garantía de desempeño conocida previamente, mientras que las columnas con etiqueta de "Nuevo" lista los nuevos resultados de este articulo; "--" indica la ausencia de un resultado relevante, ε es una constante arbitrariamente pequeña, y m denota el numero de maquinas en paralelo. La mejor garantía de desempeño conocida previamente con restricciones de precedencia se debe a Even, Naor, Rao, y Schieber (1995). Las otras cotas se deben a Phillips, Stein, y Wein (1994, 1995).

Problemas de scheduling en maquinas simples.

Primero denotamos la entrada del conjunto de trabajos {1,...,n} como N, y, para cualquier subconjunto $S \subseteq N$, usamos la siguiente notación:

$$p(S) = \sum_{j \in S} pj, \quad p^2(S) = \sum_{j \in S} p^2 j, \qquad r_{min}(S) = \min_{j \in S} rj, \qquad y \qquad r_{max}(S) = \max_{j \in S} rj.$$

Las bases de nuestros algoritmos de aproximación es **una relajación de programación lineal** que usa como variables los tiempos de terminación Cj. Formulamos el problema **1 | rj, prec | ΣwjCj** en el siguiente camino , donde las restricciones aseguran que las variables C1,...,Cn especifican un conjunto factible de tiempos de terminación:

$$\text{minimizar} \sum_{j=1}^{n} wj\, Cj \tag{1}$$

Sujeto a

(tiempos de salida) $Cj \geq rj + pj, \qquad j = 1,...,n,$ (2)

(restricciones de precedencia) $Ck \geq Cj + pk, \qquad$ para cada par j, k tal que $j \prec k,$ (3)

$Ck \geq Cj + pk,$ o $Cj \geq Ck + pj, \qquad$ para cada par j, k. (4)

Las restricciones disyuntivas (4) no son desigualdades lineales y no pueden ser modeladas usando desigualdades lineales. En su lugar , usamos una clase de desigualdades validas , cuando no se tienen tiempos de salidas o restricciones de precedencia. En particular para la ordenación 1,...,n, si no se tiene tiempo de ocio en la asignación entonces

$Cj = \sum_{k=1}^{j} pk$; por lo tanto, para cualquier asignación podemos escribir la restricción valida

$$\sum_{j=1}^{n} pjCj \geq \sum_{j=1}^{n} pj\left(\sum_{k=1}^{j} pk\right) = \sum_{j=1}^{n}\sum_{k=1}^{j} pkpj = \tfrac{1}{2}\left(p^2(N) + p(N)^2\right), \tag{5}$$

Donde las desigualdades resultan de la posibilidad de tiempo desocupado en la asignación.

Considerar los tiempos de terminación para una asignación factibles. Cj, $j \in N$. Para cada subconjunto $S \subseteq N$, podemos considerar la instancia inducida por S; los tiempos de terminación inducidos Cj, $j \in S$, corresponden a una asignación factible para esta instancia mas pequeña. Por lo tanto podemos aplicar la desigualdad previa (5) para cada subconjunto y deriva la siguiente desigualdades validas:

$$\sum_{j \in S} pjCj \geq \tfrac{1}{2}\left(p^2(S) + p(S)^2\right), \quad \text{para cada } S \subseteq N. \tag{6}$$

si adicionamos restricciones tal como (2) y (3) que refuerzan los tiempos de salida y las restricciones de precedencia, respectivamente. Aunque no se tiene todavia la caracterizacion exacta para 1 | prec | ΣwjCj, 1 | rj | ΣwjCj, o 1 | rj, prec | ΣwjCj, se vera mas adelante que **las relajaciones lineales** pueden ser usadas para encontrar soluciones cerca de la optima para cada uno de esos problemas.

La llave para la calidad de aproximación derivadas de esas relajaciones es el lema siguiente.

Lema 2.1 Tomemos a C1,...Cn, que satisfacen (6), y asumir sin perdida de generalidad que $C1 \leq ... \leq Cn$. Entonces, para cada trabajo j = 1,...,n,

$$Cj \geq \tfrac{1}{2}\sum_{k=1}^{j} pk$$

Prueba: la desigualdad (6) para S = {1,2,...,j} implica que

$$\sum_{k=1}^{j} pkCk \geq \tfrac{1}{2}\left(p^2(S) + p(S)^2\right) \geq \tfrac{1}{2}\, p(S)^2 . \qquad (7)$$

Ya que Ck $\leq$ Cj, para cada k = 1,...,j, tenemos

$$Cj \cdot p(S) = Cj \sum_{k=1}^{j} pk \geq \sum_{k=1}^{j} pkCk \geq \tfrac{1}{2}\, p(S)^2 ,$$

o equivalentemente, Cj $\geq \sum_{k=1}^{j} pk / 2.$

El lema 2.1 satisfacen las restricciones (6) y es suficiente para obtener una relajación de este: Cj$\geq$(1/2) $\sum_{k=1}^{j} pk$, j=1,...n. Esta es la intuición que sirve de base a los algoritmos de aproximación para los modelos de maquinas simples que trata este articulo.

2.1 Asignación con restricciones de precedencia en maquinas simples.

Asignar-por-$\overline{C}j$ es un algoritmo de aproximaxion 2 para 1 | prec | ΣwjCj

Basado sobre la formulacion de programación lineal que minimiza $\sum_{j=1}^{n} wjCj$ sujeto a restricciones (3) y (6). La heurística para la producir una asignación: primero, obtenemos una solución optima para el programa lineal, $\overline{C}1,\ldots,\overline{C}n$; entonces asignamos los trabajos $\overline{C}j$ de forma creciente, donde atados son interrumpidos por la elección en un orden que es consistente con la relación de precedencia. (se asumen solamente tiempos de procesamiento positivos). Nos referimos a este algoritmo como **Asignar-por-$\overline{C}j$,** ya que los trabajos están ordenados de acuerdo a sus "tiempos de terminación" en la solución de programación lineal. Observe que las restricciones (3) aseguran que el resultado de asignar será consistente con las restricciones de precedencia.

Lema 2.2 Tomemos a C1*,...,Cn* que denotan los tiempo de terminación en alguna asignación optima, y $\tilde{C}1,\ldots,\tilde{C}n$ que denotan los tiempos de terminación en la asignación dada por **Asignar-por-$\overline{C}j$.** Entonces $\sum_j wj\tilde{C}j \leq 2\sum_j wjCj^*$.

Por simplicidad asumimos que los trabajos han sido renumerados así que $\overline{C}1 \leq \ldots \leq \overline{C}n$; por lo tanto, para S={1,...,j},

$$\tilde{C}j = p(S).$$

Por el lema 2.1 obtenemos inmediatamente $\tilde{C}j \leq 2\overline{C}j$. Ya que wj $\geq$0 para cada trabajo j = 1...,n, y $\sum_j wj\overline{C}j \leq \sum_j wjCj^*$.

2.2. Scheduling con restricciones de precedencia y tiempos de salida en maquinas simples.

Asignar-por-$\overline{C}j$ es un algoritmo de aproximacion-3 para 1 | rj, prec | ΣwjCj.

Considere el siguiente programa lineal dado por (1), (2), (3), y (6). Suponga que resolvemos el programa lineal para obtener una solución optima $\overline{C}1,\ldots,\overline{C}n$; por simplicidad asumimos, como antes, que $\overline{C}1 \leq \ldots \leq \overline{C}n$. dado el $\overline{C}j$, usamos la misma heurística, Asignar-por-$\overline{C}j$: Construir la mínima asignación factible en la cual los trabajos son ordenados de acuerdo a no decreciente (no disminución) $\overline{C}j$. En este caso, podemos introducir un tiempo ocioso antes de comienzo del trabajo j: si rj es mayor que el tiempo en el cual el trabajo j-1 se completa, entonces el trabajo j se comienza a procesar al tiempo rj.

Tomemos fijo a j y defina a S = {1,...j}. después de que no se introducen tiempos ociosos entre $r_{max}(S)$ y $\tilde{C}j$,

$$\tilde{C}j \leq r_{max}(S) + p(S).$$

Además, por (2) y la ordenación de los trabajos tenemos que $r_{max}(S) \leq \max_{k=1,...,j} \bar{C}k = \bar{C}j$, y así

$$\tilde{C}j \leq \bar{C}j + p(S).$$

Finalmente, por la aplicación del Lema 2.1, $p(S) = 2\bar{C}j$
obtenemos nuestro resultado.

$$\tilde{C}j \leq \bar{C}j + 2\bar{C}j$$
$$\tilde{C}j \leq 3\bar{C}j$$

$$\sum_{j \in S} pjCj \geq l(S), \text{ para cada } S \subseteq N, \qquad (14)$$

donde

$$l(S) = r_{min}(S)\,p(S) + \frac{1}{2}(p^2(S) + p(S)^2).$$

Las desigualdades validas en (14) son una variante fortalecida de (6) (ver. Queyranne y Schulz (1994)).

2.3. Scheduling con derecho de prioridad (preemption) en maquinas simples.

Asignar-Premptivamente-por-$\bar{C}j$ es un algoritmo de aproximacion-2 para 1 | rj,prec,pmtn | ΣwjCj.

Comenzamos por el procesamiento de los datos en la instancia de esta manera, para toda j, k con $j \prec k$, $rk \geq rj + pj$. En seguida consideramos la formulacion de programación lineal dada por (1), (2), (3), y (14), que también es una relajación valida para el problema premptivo. Suponga que obtenemos una solución optima de programación lineal para este sistema; llame a esta $\bar{C}1,\ldots,\bar{C}n$. Construimos una asignación premptiva de la solución LP como sigue. Consideramos los trabajos uno a un tiempo, en orden de sus valores $\bar{C}j$; note que el orden es consistente con las restricciones de precedencia, a causa de (3), y así, por el tiempo que consideramos para el trabajo j, cada un de sus predecesores han sido realmente asignados. Para asignar el trabajo j, encontramos el primer punto en el tiempo, en la asignación parcialmente construida, en la que cada uno de los predecesores de j han completado su procesamiento, o tiempo rj, cualquiera que sea mas grande. Después de este punto en el tiempo asignamos partes de j en cualquier tiempo ocioso en la asignación parcial, hasta que j se consigue asignar completamente. A este algoritmo le llamamos **Premptively-Schedule-by-$\bar{C}j$**.

Tomemos $\bar{C}1 \leq \ldots \leq \bar{C}n$ a ser una solución optima para el programa lineal definido por (1), (2), (3), y (14), y tomemos a $\tilde{C}1,\ldots,\tilde{C}n$ que denotan el tiempo de terminación en la asignación encontrada por **Preemptively-Schedule-by-$\bar{C}j$**.

Considere un trabajo j, cuyo tiempo de terminación en la asignación construida es $\tilde{C}j$, y considere la asignación parcial construida por el algoritmo para los trabajos 1,....,j. Tomemos a t a ser definido como el ultimo punto en el tiempo antes de $\tilde{C}j$ en el que se tiene un tiempo ocioso en esta parcial asignación (o, si no existe el tiempo ocioso antes de $\tilde{C}j$, t = 0). Tomemos a S para denotar el conjunto de trabajos que están parcialmente procesados en el intervalo [t,Cj], en la asignación parcial. Primero, observamos que los trabajos que no están en S se consiguen terminar antes en el tiempo t ; y los trabajos que están en el intervalo [t,Cj]

fueron asignados en ese mismo intervalo. Por lo tanto, $t = r_{min}(S)$, y puesto que no se tiene un tiempo ocioso entre t y $\tilde{C}j$,

$$\tilde{C}j \leq r_{min}(S) + p(S). \qquad (a)$$

Ahora regresamos a las desigualdades fortalecidas de (14). Recuerde que el conjunto S fue definido con referencia a la asignación parcial obtenida justo después que el trabajo j fue asignado; así, para todo. Este hecho, combinado con (14), implica que

de (14) $$\bar{C}j\, p(S) \geq \sum_{k \in S} pk\bar{C}k \geq r_{min}(S)\, p(S) + \frac{1}{2} p(S)^2 ,$$

$$\bar{C}j \geq r_{min}(S) + (1/2)p(S)$$

$$r_{min}(S) \leq \bar{C}j - (1/2)p(S)$$

sustituyendo en (a)

$$\tilde{C}j \leq \bar{C}j - (1/2)p(S) + p(S).$$

$$\tilde{C}j \leq \bar{C}j + (1/2)p(S)$$

del lema 2.1 $p(S) = 2\bar{C}j$ sustituyendo

$$\tilde{C}j \leq \bar{C}j + (1/2)\, 2\bar{C}j$$

el resultado es el siguiente

$$\tilde{C}j \leq 2\bar{C}j$$

Notar que el algoritmo **Asignar-Premptivamente-por-** $\bar{C}j$ **(Preemptively-Schedule-by-** $\bar{C}j$ **)** crea un derecho de prioridad solo cuando un trabajo es terminado. Consecuentemente el numero total de premptiones es menor que n. Además, el programa lineal fundamental (1), (2), (3), y (14) pueden ser resueltos en tiempo polinomial. La observación crucial precisa es que se tiene un algoritmo de separación en tiempo polinomial para las desigualdades en (14), ver Queyranne y Schulz (1995) o Goemans (1996). De aquí, en conjunto, tenemos el siguiente corolario.

Asignar-Premptivamente-por- $\bar{C}j$ es un algoritmo de aproximacion-2 para 1 | rj,prec,pmtn | ΣwjCj.

Debido a la suposición acerca de la integridad de los datos, cada premption introducida por **Asignar-Premptivamente-por-** $\bar{C}j$ ocurrirá a un punto entero en tiempo. Consecuentemente, si aplicamos **Asignar-Premptivamente-por-** $\bar{C}j$ a una instancia en la que pj = 1, para cada j = 1,...,n, entonces allí no será introducida cualquier premption. De aquí se obtenido una asignación nopremptiva que esta dentro de un factor de 2 del optimo premptivo.

La estructura general de los algoritmos en-línea.

Una técnica que produce un algoritmo en línea de aproximacion-4r para minimizar la suma de los pesos del tiempo de terminación en la función objetivo, donde r depende del ambiente del Scheduling; la postura esta en línea en el sentido que estamos construyendo la asignación como un producto del tiempo, y no conocemos la existencia del trabajo j hasta un tiempo rj. Si se asigna los trabajos a traves de intervalos de tiempo en una forma de glotón, uno puede obtener una buena garantía de desempeño. La técnica es totalmente general, y depende solo de la existencia de un algoritmo fuera de línea.para el siguiente problema en-linea.

La técnica utilizada, que es similar a una usada por Blum, Chalasani, Coppersmith, Pulleyblank, Raghavan, y Sudan (1994), es útil en el diseño de algoritmos en línea con garantías de desempeño que estan cerca de igualar a las obtenidas por los mejores algoritmos de aproximación fuera de línea.

La subrutina requerida es una generalización de una subrutina usada en el diseño de algoritmos de aproximación para minimizar la longitud de la asignación. Para varios de los modelos considerados en este articulo, el diseño de esta subrutina mas general es una clara extensión de técnicas concebidas para minimizar la longitud de la asignación (aunque no veamos todavía como construir esta subrutina para modelos de restricciones de precedencia). En adición a la simplicidad del enfoque, la garantía de desempeño puede ser totalmente buena. De hecho, para 1| rj | ΣwjCj y P | rj | ΣwjCj, respectivamente, este dirige a algoritmos de aproximación de -(3 + ε) y -(4 + ε) que igualan asintoticamente la garantía provista para modelos fuera de linea.

Estos resultados proveen un medio para convertir algoritmos scheduling fuera de línea dentro de algoritmos en línea.

En este caso, un algoritmo de aproximacion-r fuera de línea produce un algoritmo de aproximacion-2r en línea.