UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA MATEMÁTICA

OPTIMIZACIÓN DE GANANCIAS CÓNCAVAS EN MÁQUINAS PARALELAS

TESIS PARA OPTAR AL GRADO DE
MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS
MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO

JUAN PABLO DONOSO MERLET

PROFESOR GUÍA:
JOSÉ SOTO SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
MARTÍN MATAMALA VÁSQUEZ
JOSÉ VERSCHAE TANNENBAUM

Este trabajo ha sido parcialmente financiado por CMM ANID BASAL FB210005

SANTIAGO DE CHILE
2022

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICA
POR: JUAN PABLO DONOSO MERLET
FECHA: 2022
PROF. GUÍA: JOSÉ SOTO SAN MARTÍN

OPTIMIZACIÓN DE GANANCIAS CÓNCAVAS EN MÁQUINAS PARALELAS

El problema de Optimización de Ganancias Cóncavas en Máquinas Paralelas consiste en, dados conjuntos J de trabajos, I de máquinas que pueden procesar dichos trabajos y $\{f_j\}_{j\in J}$ de funciones de ganancias cóncavas asociados a cada trabajo, encontrar una función $A : I \times J \to \mathbb{Q}_+$ que *asigna* a cada trabajo j un tiempo de proceso A_{ij} en la máquina i (si $A_{ij} = 0$, entonces j no es procesado en la máquina i), de modo que A maximiza la ganancia obtenida por el procesamiento de dichos trabajos y al mismo tiempo satisface la capacidad que tienen las máquinas para procesar dichos trabajos. En este trabajo se tratan dos variantes de este problema, las cuales se llaman GCMP *con transferencia* y *sin transferencia*, donde la primera consiste en el caso en que se permite que un mismo trabajo pueda ser procesado en más de una máquina (sin ser simultáneo), y en la segunda no.

Se demuestra primero que el caso de GCMP con transferencia se puede resolver en tiempo polinomial, usando para esto la solución de knapsack fraccional. Mostramos luego que GCMP sin transferencia es *NP-completo*, probando para esto que existe una reduccón del problema 3–Partición al problema de decisión asociado a GCMP sin transferencia. Lo anterior nos lleva a demostrar como resultado principal de este trabajo que GCMP sin transferencia admite un PTAS, lo cual se realiza en tres etapas: se ve el caso de GCMP sin transferencia solo trabajos cortos; solo trabajos largos; y de largos arbitrario.

El presente trabajo va dedicado a mi madre, la persona que más admiro en mi vida.

Agradecimientos

Ante todo, quiero agradecer a mi familia. Son el pilar que sostiene mi vida y mi ser, y la razón misma de mi alegría. Sin ustedes habría sido imposible completar esta etapa. Agradezco también a mi querida esposa Mariana. Tu apoyo incondicional y amor fueron un motor fundamental a lo largo de todo este proceso. Agradezco a mi profesor guía, José Soto San Martín. El honor de trabajar con usted ha sido tremendo. Muchas gracias por su apoyo constante y paciencia infinita todo este tiempo. Sus excelentes explicaciones fueron fundamentales para poder llegar hasta aquí, y su rigurosidad es una inspiración que me llevaré para toda la vida. Muchas gracias también a mi comisión, profesores José Verschae y Martín Matamala, por aceptar ser parte de este proceso. Finalmente, quiero agradecer al soporte brindado por el proyecto Fondecyt 1181180.

Tabla de Contenido

Introducción **1**

1. Preliminares **5**
1.1. Problema de las ganancias cóncavas en máquinas paralelas (GCMP) 5
1.1.1. GCMP sin transferencia 6
1.1.2. GCMP con transferencia 7
1.1.3. Funciones cóncavas lineales por trozos. 7
1.2. Técnicas preliminares 8
1.2.1. Repartir de una máquina a múltiples máquinas mediante round-robin. 8
1.2.2. Contracción de una asignación 9

2. GCMP con transferencia **11**

3. GCMP sin transferencia: Complejidad **18**

4. GCMP sin transferencia: solo trabajos cortos **21**

5. GCMP sin transferencia: solo trabajos largos **25**

6. GCMP sin transferencia: Trabajos de largo arbitrario **32**
6.1. Problema Redondeado 32
6.2. Programa lineal entero asociado a una configuración de máquinas κ, a una eficiencia $\alpha > 0$ y una capacidad $T > 0$ 34
6.3. PTAS para GCMP sin Transferencia 36
6.3.1. Demostración del Teorema 6.1 49

Conclusión **51**

Bibliography **52**

Índice de Ilustraciones

1. Pilas de lixiviación, Minera Centinela 1
2. Función de ganancia asociada a un trabajo j y una representación del conjunto de máquinas $I = \{i_u\}_{u=1}^{m}$ 2

1.1. Representación de asignación factible para GCMP sin transferencia. 7
1.2. Representación de asignación factible para GCMP con transferencia. 7
1.3. Representación final para el procesamiento de la asignación de A 9

2.1. Para cada función de ganancia lineal por trozos f_j de un trabajo j cualquiera, esta se puede descomponer en los subtrabajos g que la conforman. 12

3.1. A la izquierda, un ejemplo de función de ganancia definida a partir del elemento $x_j \in S$. A la derecha, se representa con el tamaño de los bloques el tiempo de proceso x_τ/K de cada trabajo asignado, cuando $\mathrm{Val}(A) = m$. 19

5.1. Para el caso $k = 2$, $J = \{j_1, j_2, j_3\}$, el conjunto de largos $\mathcal{L}$ está dado por $\mathcal{L} := \{1/2, 3/4, 1\}$. En la imagen se ven dos configuraciones c_1 y c_2, y se muestra como dos máquinas distintas utilizan dichas configuraciones. c_1 utiliza dos trabajos de largo $1/2$, j_1 y j_2. Notar que la configuración solo indica el tiempo de proceso de los trabajos largos a utilizar por una máquina arbitraria, mas no especifica *qué* trabajos. Por otro lado, la configuración c_2 indica que una máquina dada usará un único trabajo de largo $3/4$ (donde se puede ver en la imagen que se ha utilizado nuevamente el trabajo j_2), lo cual es factible dado que no supera la capacidad de dicha máquina dada por 1. 27
5.2. Grafo bipartito completo $G(\kappa)$ definido a partir de κ. Para cada largo ℓ presente en la configuración de máquinas, se agregan tantos nodos como indiquen el total de configuraciones de κ. Todos los trabajos son conectados a cada uno de estos nodos, y si un arco conecta un nodo j y un largo ℓ, su peso está dado por $g_e := f_j(\ell)$, es decir por la ganancia de procesar dicho trabajo j por un tiempo ℓ. 28
5.3. En la imagen, la configuración de máquinas definida anteriormente determina dos máquinas utilizando la configuración c_1 y otra utilizando la configuración c_2. Como $b_{1/3} = \sum_{c \in \mathcal{C}} \kappa_{\mathcal{C}} \cdot c_{1/3} = 2 \cdot 3 = 6$ (i.e., los seis casilleros/espacios de las máquinas i_1 e i_2), entonces existen 6 nodos de R, 3 asociados a i_1 y otros 3 asociados a i_2. 29

6.1. Ejemplo de configuración de máquinas para el caso $k = 2$. Dicho k implica que el conjunto de largos $\mathcal{L}$ está dado por $\mathcal{L} = \{1/2, 3/4, 1\}$. Luego la configuración $c_1 = (0, 1, 0)$ significa que se usan cero trabajos de largo $1/2$, un trabajo de largo $3/4$ y cero trabajos de largo 1. c_2 indica que se utilizan dos trabajos de largo $1/2$, y ninguno del resto, y finalmente la configuración c_3 solo utiliza un trabajo de largo 1 35

Introducción

Motivación del Problema

El presente trabajo de tesis tiene su motivación en el proceso de lixiviación del cobre. En la minería del cobre, el proceso comienza con la extracción del recurso mineral desde el *pit* o *rajo*, el cual después debe ser *chancado* para lograr la disminución granulométrica necesaria para los procesos siguientes. Una vez *chancado* el mineral, éste se *aglomera* con distintos reactivos que serán claves para que la reacción química de *lixiviación* tome lugar. Luego se procede a formar las llamadas *pilas de lixiviación*, las cuales constituyen una disposición del material en *pads* o *pilas*, los cuales serán regados con una sustancia ácida y así poder extraer el cobre del mineral apilado.

Figura 1: Pilas de lixiviación, Minera Centinela

En el contexto anterior, uno de los problemas que deben ser resueltos al momento de llevar a cabo la lixiviación es determinar qué material es procesado de modo de recuperar la mayor cantidad de cobre desde el mineral apilado. Para fijar ideas, considere que cada pila de lixiviación se compone de distintos *bloques* de mineral, y que cada bloque de mineral es un cubo de un volumen dado. Además, cada uno de estos bloques tienen asociado una *cinética de recuperación*, que consiste en una función cóncava $f : [0, 1] \to \mathbb{R}_+$, no-negativa y creciente

que explica la recuperación de cobre en función del tiempo. Así, el problema de qué bloques son lixiviados en una ventana de tiempo dada con el objetivo de maximizar la recuperación, lleva a la siguiente abstracción. Considere un conjunto de trabajos J, donde cada trabajo j tiene asociado una función de ganancia $f_j : [0,1] \to \mathbb{Q}_+$, cóncava, creciente y no-negativa. Además, considere un conjunto de máquinas I, las cuales tienen asociada una capacidad de procesamiento de 1. Así, lo que se busca es determinar la función $A : I \times J \to \mathbb{Q}_+$ que asigna trabajos a máquinas un *tiempo de proceso* de modo que la cantidad

$$\text{val}(A) := \sum_{j \in J} f_j \left(\sum_{i \in I} A_{ij} \right)$$

llamada *valor de la asignación* A,sea máxima.

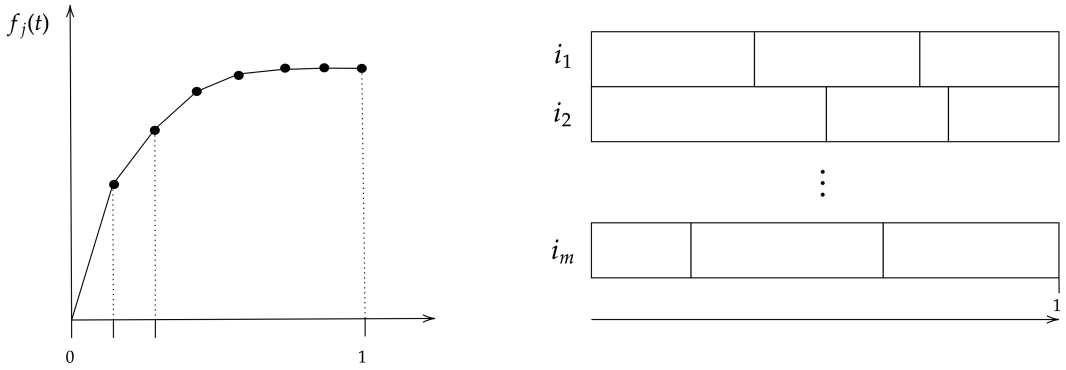


Figura 2: Función de ganancia asociada a un trabajo j y una representación del conjunto de máquinas $I = \{i_u\}_{u=1}^m$

El problema descrito en el párrafo anterior será llamado *Optimización de Ganancias Cóncavas en Máquinas Paralelas, o GCMP*. En el presente trabajo, se estudiarán dos variantes de este problema:

1. GCMP *con transferencia*. El término transferencia hace referencia a que un trabajo pueda ser procesado en más de una máquina. Es decir puede ocurrir que para dos máquinas distintas $i_1, i_2 \in I$ se tenga que $A_{i_1 j} > 0$ y $A_{i_2 j} > 0$ para un mismo trabajo j.
2. GCMP *sin transferencia*. En este caso, no se permite el punto anterior, es decir los trabajos j son procesados en *a lo más* una máquina.

El resultado principal de la presente tesis es el siguiente. Sea $k \in \mathbb{N}$ dado:

Teorema 0.1 *GCMP sin transferencia admite un PTAS, y existe un polinomio $p : \mathbb{R}^3 \to \mathbb{R}$ tal que la complejidad está dada por $\mathcal{O}(p(n,m,k)m^{k^{k^2}})$.*

A continuación se mostrarán problemas similares.

Trabajo Relacionado

$0-1$ Multiple Knapsack

Un problema similar es una variante del problema de la mochila (o *knapsack*). Dado un conjunto de objetos $\mathcal{O}$, cada uno con un peso w_o y un valor v_o asociados, y una mochila de capacidad $W > 0$, el problema consiste en determinar que objetos deben ser seleccionados de modo de no superar la capacidad de la mochila y al mismo tiempo maximizar el valor total que esta lleva. El problema knapsack se puede formular mediante el siguiente programa lineal entero.

$$\max_{x} \quad \sum_{o \in \mathcal{O}} v_o x_o \tag{1a}$$

$$\text{s.t.} \quad \sum_{o \in \mathcal{O}} x_o w_o \leq W \tag{1b}$$

$$x \in \{0,1\}^{\mathcal{O}} \tag{1c}$$

Una variante natural de este problema consiste en el caso en que se tienen múltiples mochilas, llamado variante $0-1$ Multiple Knapsack.

$$\max_{x} \quad \sum_{i \in I} \sum_{o \in \mathcal{O}} v_o x_{oi} \tag{2a}$$

$$\text{s.t.} \quad \sum_{o \in \mathcal{O}} x_{oi} w_o \leq W_i \quad \forall i \in I \tag{2b}$$

$$\sum_{i \in I} \sum_{o \in \mathcal{O}} x_{oi} \leq 1 \tag{2c}$$

$$x \in \{0,1\}^{\mathcal{O} \times I} \tag{2d}$$

Sabemos que ambas variantes son NP-completos [L Z86], y que admiten un PTAS [CK05].

Makespan Minimization

Consideremos el siguiente problema, según la definición de [WS11]: Supongamos que tenemos un conjunto de m máquinas y un conjunto de n trabajos, cada uno con un tiempo de proceso asociado para ser completado. Cada trabajo $j = 1, \ldots, n$ debe ser procesado en alguna de las máquinas por un tiempo p_j sin interrupción. Cada máquina puede procesar a lo más un trabajo a la vez. Con esto, el objetivo es completar todos los trabajos lo antes posible, es decir si el trabajo j se completa en el instante C_j (suponiendo que el procesamiento comienza en el instante 0), entonces queremos minimizar $C_{max} = \text{máx}_{j=1,\ldots,n} C_j$, lo que es llamado en general el *makespan* o largo del agendamiento.

El problema recién enunciado es NP-difícil [GJ79a] y en Teorema 3.7 de [WS11] se demuestra que admite un PTAS.

Fractional Knapsack

Fractional Knapsack consiste en una variante de Knapsack donde la restricción (1c) es reemplazada por:

$$x \in [0,1]^{\mathcal{O}}$$

Es decir, está permitido agregar a la mochila *una fracción* de los objetos, obteniendo consecuentemente una fracción del valor de dicho objeto. Este problema admite una solución glotona de complejidad $\mathcal{O}(n \log n)$, con n el cardinal del conjunto $\mathcal{O}$ de objetos.

Knapsack acotado

En [EW19] Eisenbrand y Weismantel consideran programas lineales enteros en la forma estándar

$$\text{máx}\{c^T x : Ax = b,\ x \geq 0,\ x\ in\mathbb{Z}^n\}$$

y demuestran, utilizando el lema de Steinitz, que la distancia en norma ℓ_1 entre una solución óptima entera y fraccional, suponiendo que las variables están acotadas superiormente, está acotada por $m \cdot (2 \cdot m \cdot \Delta + 1)^m$. Sea el problema de *kanpsack acotado* el siguiente:

$$\text{máx}\{c^T x : a^T x = \beta,\ 0 \leq x \leq u,\ x \in \mathbb{Z}^n\}$$

donde $c, a, u \in \mathbb{Z}^n_{>0}$ y $\beta \in \mathbb{Z}_{>0}$. Cuando la cota superior $u = \beta \cdot \mathbf{1}$, entonces el problema de knapsack se dirá *no acotado*. Sea también Δ_a una cota superior de las entradas de a. Utilizando la cota mencionada en el párrafo anterior, logran demostrar lo siguiente:

Teorema 0.2 *El problema de knapsack no acotado puede ser resuelto en tiempo* $\mathcal{O}(n \cdot \Delta_a^2)$

Nuestro Resultado

En el capítulo 1 se realizan las definiciones elementales para el trabajo de Tesis, tales como las definiciones de GCMP en con/sin transferencia y *asignación*, trabajos activos, *valor de una asignación*, entre otras. Además, se especificará las funciones cóncavas que se considerarán. Por último, se detallarán las técnicas de *round-robin* y *contracción de una asignación*, que serán utilizadas constantemente. En el capítulo 2, se resuelve el caso de GCMP con transferencia, de manera que se define un Algoritmo glotón para resolver el problema en tiempo polinomial. Además, en dicha resolución se define el concepto de *subtrabajo* y *eficiencia* de un subtrabajo, los cuales serán utilizados frecuentemente en los siguientes capítulos. En el capítulo 3 se aborda la complejidad de GCMP sin transferencia, demostrando que es un problema *fuertemente NP-completo*, demostrando que 3−partición se puede reducir a GCMP sin transferencia. Lo anterior justifica que el objetivo del presente trabajo de tesis sea la demostración de la existencia de un PTAS para GCMP sin transferencia. En el capítulo 4 se demuestra la existencia de un PTAS para GCMP sin transferencia para el caso en que todos los trabajos en la asignación óptima deben ser procesados en a lo más $1/k$ tiempo de proceso, con $k \in \mathbb{N}$ dado. En el capítulo 5, de manera análoga al capítulo 4, para $k \in \mathbb{N}$ dado, se demuestra que GCMP sin transferencia admite un PTAS en el caso que se exige, además, que las asignaciones deban procesar sus trabajos en un tiempo no menor a $1/k$. Finalmente, en el capítulo 6 se demuestra que GCMP sin transferencia admite, para $k \in \mathbb{N}$ dado, un $(1 - 1/k)^2$-PTAS, sin poner restricciones a si los trabajos deban ser procesados en tiempos largos o cortos, mediante un algoritmo de complejidad $\mathcal{O}(p(n, m, k) \cdot m^{k^{k^2}})$, para un polinomio $p : \mathbb{R}^3 \to \mathbb{R}$.

Capítulo 1

Preliminares

En este capítulo se definirán variantes del problema de asignación de trabajos con ganancias cóncavas en máquinas paralelas (GCMP).

1.1. Problema de las ganancias cóncavas en máquinas paralelas (GCMP)

Para fijar ideas, primero se definen aspectos que serán comunes a todas las variantes de GCMP. En cada una de estas variantes, se tiene un conjunto de *trabajos* J y un conjunto de *máquinas* I, tales que $|J| = n$ y $|I| = m$, con $n, m \in \mathbb{N}$. Cada máquina tendrá una *capacidad de proceso* igual a 1. Cada trabajo $j \in J$ está asociado a una *función de ganancia* $f_j : [0,1] \to \mathbb{Q}_+$, cóncava, creciente, y con $f_j(0) = 0$, de modo que la ganancia por procesar un trabajo j un tiempo total $t \in [0,1]$ (posiblemente en múltiples máquinas) es $f_j(t)$. Ganancias Cóncavas en Máquinas Paralelas busca encontrar la *asignación* de trabajos a máquinas (i.e. qué trabajos serán procesados y en qué máquinas, y por cuánto tiempo), de manera que se obtenga la máxima ganancia, sujeto a que la asignación satisfaga ciertas restricciones, las cuales serán detalladas más adelante. A continuación, se define una *asignación* y el *valor* asociada a la misma.

Definición 1.1 (Asignación) *Una asignación es una función $A : I \times J \to \mathbb{Q}_+$, de modo que para $i \in I, j \in J$, $A(i,j)$ representa el tiempo que la máquina i procesa el trabajo j. Para hacer la lectura más liviana, se escribirá A_{ij} en vez de $A(i,j)$.*

Definición 1.2 (Trabajo activo) *Dada una asignación A, se dirá que un trabajo j* está activo *en la máquina i si y solo si $A_{ij} > 0$. Decimos que j* es activo *si está activo en alguna máquina, de lo contrario j* es inactivo.

Definición 1.3 (Valor de una asignación) *El valor de una asignación A se define de la*

siguiente forma:

$$Val(A) := \sum_{j \in J} f_j \left(\sum_{i \in I} A_{ij} \right) \tag{1.1}$$

Es importante notar que, respecto al valor de una asignación, no es relevante qué máquina realiza qué trabajo, pues la función de ganancia de cada trabajo no depende del conjunto de máquinas.

Finalmente, se definirá lo que es un *esquema de aproximación en tiempo polinomial*, o *PTAS* por sus siglas en inglés:

Definición 1.4 (PTAS) *Un esquema de aproximación en tiempo polinomial es una familia de algoritmos $\{A_\varepsilon\}$ donde, para cada $\varepsilon > 0$ existe un algoritmo A_ε tal que A_ε entrega una solución de a lo menos $(1 - \varepsilon) \cdot OPT$, donde OPT es el valor del óptimo del problema en cuestión.*

1.1.1. GCMP sin transferencia

Con lo anterior, ya se tiene lo necesario para definir en detalle GCMP sin transferencia.

Definición 1.5 (GCMP sin transferencia) *Sean I, J conjuntos de máquinas y trabajos, respectivamente. Para cada trabajo $j \in J$, se asocia una función de ganancia $f_j : [0,1] \to \mathbb{Q}_+$, cóncava y creciente, con $f_j(0) = 0$. GCMP sin transferencia busca determinar la asignación de trabajos a máquinas $A : I \times J \to \mathbb{Q}_+$ de modo tal que cada trabajo esté activo en a lo más una máquina y tal que maximice la ganancia de procesar dichos trabajos, en otras palabras, se busca una solución para el problema lineal entero siguiente:*

$$\max_A \quad val(A) \tag{1.2a}$$

$$s.t. \quad A_{ij} \leq y_{ij} \quad \forall i \in I,\ j \in J \tag{1.2b}$$

$$\sum_{i \in I} y_{ij} = 1 \quad \forall j \in J \tag{1.2c}$$

$$\sum_{j \in J} A_{ij} \leq 1 \quad \forall i \in I \tag{1.2d}$$

$$A_{ij} \in \mathbb{Q}_+, y_{ij} \in \{0,1\} \quad \forall i \in I,\ j \in J \tag{1.2e}$$

La variable y_{ij} indica si j está activo en i. Se hará referencia a al problema recién descrito como GCMP sin transferencia (ya que no existe transferencia de trabajos entre máquinas).

La restricción 1.2d es para que cada máquina no pueda procesar más tiempo del que tiene disponible (en este caso 1 unidad). Las restricciones 1.2c y 1.2b establecen que, para que una asignación sea factible, cada trabajo está activo en a lo más una máquina. La mayor parte de este trabajo consistirá en el estudio del problema recién definido.

1.1.2. GCMP con transferencia

Definición 1.6 (GCMP con transferencia) *GCMP con transferencia consiste en el mismo problema descrito en 1.5, pero con las siguientes modificaciones:*

1. *Cada trabajo se puede procesar en más de una máquina.*
2. *Cada trabajo puede estar activo por a lo más un tiempo* 1 *en total.*
3. *Cada trabajo puede estar activo en una máquina por a lo más* 1 *intervalo consecutivo de tiempo.*
4. *Ningún trabajo puede procesarse de manera simultánea*

En la sección 1.2.1 veremos que para resolver el problema, solo es necesario determinar el valor $A_{i,j}$ de cada máquina, pues la decisión de como se reparte temporalmente el trabajo en cada máquina se puede determinar vía un postproceso.

Dicho esto, la siguiente es una formulación del problema GCMP con transferencia.

$$\max_{A} \quad val(A) \tag{1.3a}$$

$$\text{s.t.} \quad \sum_{i \in I} A_{ij} \leq 1 \quad \forall j \in J \tag{1.3b}$$

$$\sum_{j \in J} A_{ij} \leq 1 \quad \forall i \in I \tag{1.3c}$$

$$A_{ij} \in \mathbb{Q}_+ \quad \forall i \in I, j \in J \tag{1.3d}$$

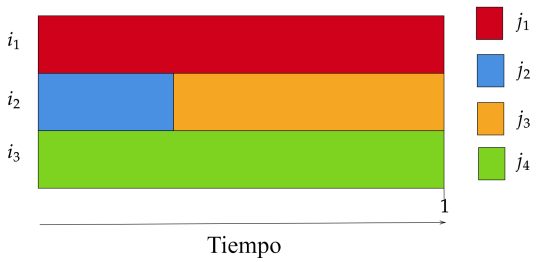


Figura 1.1: Representación de asignación factible para GCMP sin transferencia.

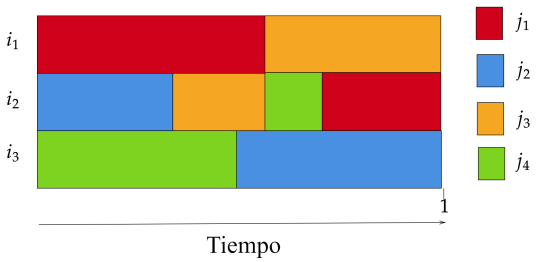


Figura 1.2: Representación de asignación factible para GCMP con transferencia.

1.1.3. Funciones cóncavas lineales por trozos.

Hasta el momento, de las funciones de ganancia $f_j : [0,1] \to \mathbb{Q}_+$ solo se ha especificado que son cóncavas, crecientes y que $f_j(0) = 0$. En esta tesis solo se considerarán funciones de ganancia lineales por trozos, de modo que cada f_j tenga un número finito de quiebres en el intervalo $[0,1]$.

1.2. Técnicas preliminares

1.2.1. Repartir de una máquina a múltiples máquinas mediante round-robin.

Como establecen las Definiciones 1.5 y 1.6, para resolver cualquiera de las variantes de GCMP basta encontrar la función asignación $A : I \times J \to \mathbb{Q}_+$, satisfaciendo las respectivas restricciones. Pero para poder determinar *cómo* cada máquina procesa los trabajos en función del tiempo, va a depender de la variante la forma de determinar dicho procesamiento. El caso de GCMP sin transferencia es más sencillo, ya que cada trabajo está activo en a lo más a una máquina, luego los trabajos asignados en una máquina fija se pueden procesar como intervalos, en cualquier orden. En cambio, para el caso de GCMP con transferencia, es necesario tener precaución al momento de determinar cómo se procesan los trabajos en todas las máquinas en función del tiempo, ya que dada una asignación A, un trabajo activo en varias máquinas podría terminar siendo procesado simultáneamente en distintas máquinas, lo cual no está permitido. Para este último caso se utilizará la asignación de trabajos a máquinas dada por la técnica *round-robin*, que se describe más abajo. Veremos esta técnica en mayor generalidad: Sea $A : I \times J \to \mathbb{Q}$ una asignación tal que cada trabajo j tiene tiempo total de proceso $\sum_{i \in I} A_{ij} \leq 1$:

1. Todos los trabajos activos (i.e., los j tales que $A_{ij} > 0$ para alguna máquina $i \in I$), se ponen en una máquina ficticia, denotada por $i^\star \notin I$, de capacidad m, de modo que dichos trabajos se procesen de manera ininterrumpida, eligiendo un orden cualquiera entre trabajos. Dicha situación se puede ver a la izquierda de la Figura 1.3.
2. Para asignar cada trabajo de $i^\star$ a las máquinas de I, basta cortar $i^\star$ en m máquinas de largo 1. Lo que se obtiene finalmente son los *espacios* que usarán cada uno de los trabajos activos en cada una de las máquinas, como se muestra a la derecha en la Figura 1.3.

Observación: Los espacios asociados a un mismo trabajo en distintas máquinas no son simultáneos por el hecho que, de serlos, significa que dicho trabajo se procesó una cantidad de tiempo superior a 1, lo cual es una contradicción. Es importante notar que, en el caso de GCMP con transferencia, una vez obtenida la asignación óptima A_{OPT}, round-robin permite directamente obtener cómo se procesa cada uno de los trabajos asignados, esto es:

1. Qué máquina(s) procesa(n) cada trabajo j activo.
2. El tiempo de proceso para cada trabajo activo.

Como round-robin será utilizado más adelante, a continuación se explicita el pseudo-código para futuras referencias:

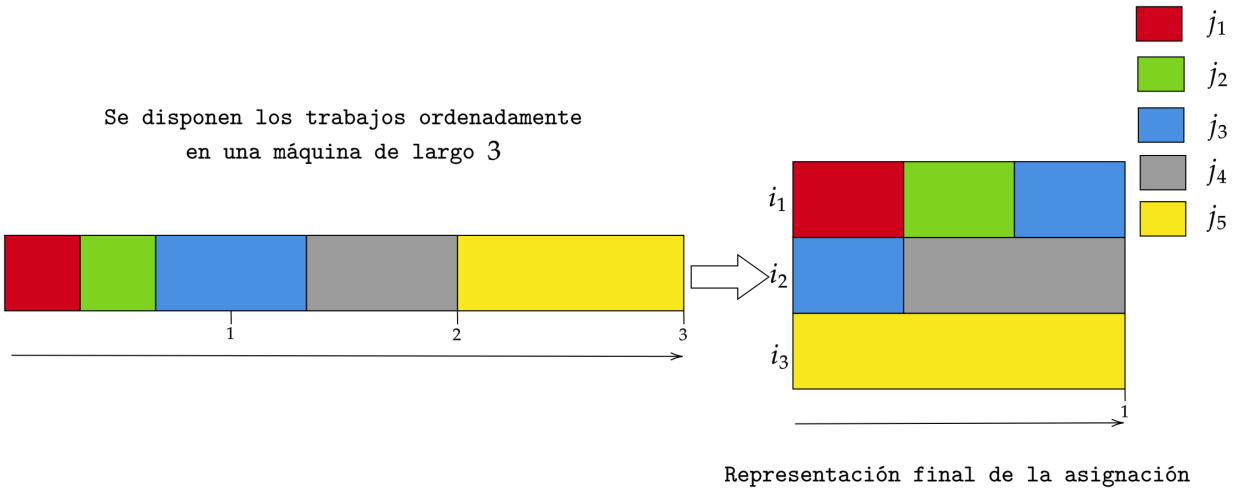


Figura 1.3: Representación final para el procesamiento de la asignación de A

.

Algorithm 1 `round-robin`

1: **Inputs:** $A : J \to \mathbb{Q}_+, I = \{1, \ldots, m\},\ J = \{1, \ldots, n\}$
2: $A^r_{ij} \leftarrow 0 \quad \forall i \in I,\ j \in J \cup \{0\}$
3: $i \leftarrow 1$
4: **for** $j = 1, \ldots, n$ **do**
5: $\quad A^r_{ij} \leftarrow \min(A_j, 1 - \sum_{z=1}^{j} A^r_{i,z-1})$
6: $\quad$ **if** $A^r_{ij} < A_j$ **or** $A_j = \sum_{z=1}^{j} A^r_{i,z-1}$ **then**
7: $\quad\quad i \leftarrow i + 1$
8: $\quad\quad A^r_{ij} \leftarrow A_j - A^r_{i-1,j}$
9: $\quad$ **end**
10: **end**
11: **Return:** $A^r : I \times J \to \mathbb{Q}_+$

Observación El algoritmo 1 round-robin tiene complejidad $\mathcal{O}(n)$, pues en el paso 8 recorre todos los trabajos de J.

1.2.2. Contracción de una asignación

Con *contracción de una asignación* se hace referencia a la acción de, dada una asignación $A : I \times J \to \mathbb{Q}_+$, disminuir todos los tiempos de proceso de los trabajos por un factor $\alpha \in (0, 1)$. La contracción de A se define como la asignación $A^s : I \times J \to \mathbb{Q}_+$ dada por:

$$A^s_{ij} = \alpha A_{ij} \tag{1.4}$$

Lema 1.7 *Siempre se tiene que:*

$$val(A^s) \geq \alpha val(A) \tag{1.5}$$

Demostración. En efecto, sea A una asignación

$$\begin{aligned}
\text{val}(A^s) &= \sum_{j \in J} f_j \left(\sum_{i \in I} A^s_{ij} \right) \\
&= \sum_{j \in J} f_j \left(\sum_{i \in I} \alpha A_{ij} \right) \\
&\geq \sum_{j \in J} \alpha f_j \left(\sum_{i \in I} A_{ij} \right) \qquad (\star) \\
&= \alpha \text{val}(A)
\end{aligned}$$

Donde el paso $(\star)$ viene de la concavidad de las f_j, i.e. para $\alpha \in (0,1)$ y $t \in [0,1]$, $f_j(\alpha t) \geq \alpha f_j(t)$ y que $f_j(0) = 0$. □

Capítulo 2

GCMP con transferencia

Como se comentaba anteriormente, esta es la variante más simple y por ello será la primera en ser resuelta. Se verá que GCMP con transferencia es reductible a Knapsack Fraccional (en adelante KF). Para fijar ideas, se demostrará una caracterización de la solución óptima, tanto de GCMP con o sin transferencia. Un primer resultado que será de utilidad para poder encontrar una solución para el caso con transferencia es que basta poder resolver el problema en 1 máquina.

Teorema 2.1 *Sean I, J, y $\{f_j\}_{j\in J}$ una instancia de GCMP con transferencia. Entonces dicha instancia puede ser resuelta reemplazando el conjunto de máquinas I por una única máquina de capacidad m, y luego aplicando round-robin.*

Demostración. Llame P a una instancia de GCMP con transferencia donde el conjunto de máquinas es I, y de manera análoga llame P' a la instancia que tiene solo una máquina de capacidad m. Sea $A' : J \to \mathbb{Q}_+$ una asignación óptima para P', y sea $A : I \times J \to \mathbb{Q}_+$ una asignación óptima para P. Veamos primero que A induce una solución factible para el problema P' con igual valor, específicamente tomando $B'_j = \sum_{i\in I} A_{ij}$, se obtiene que B' es factible para P' y que

$$\mathrm{val}(A)_P = \mathrm{val}(B')_{P'} \leq \mathrm{val}(A')_{P'}$$

Por otro lado, A' induce también una solución factible para P. Para ello se puede utilizar la técnica introducida en la sección 1.2.1, i.e. se repartirá la asignación A' en m máquinas de capacidad 1. Sea $A^r : I \times J \to \mathbb{Q}_+$ el resultado de aplicar dicha técnica sobre A'. Entonces A^r es factible y:

$$\mathrm{val}(A')_{P'} = \mathrm{val}(A^r)_P \leq \mathrm{val}(A)_P \leq \mathrm{val}(A')_{P'}$$

Luego $\mathrm{val}(A')_{P'} = \mathrm{val}(A)_P$, y se concluye la demostración. □

Gracias al resultado anterior, en adelante se considerará que siempre se está resolviendo GCMP con transferencia con m máquinas en una única máquina de capacidad m (notar que GCMP con transferencia es equivalente a GCMP sin transferencia en el caso de una única máquina). Un concepto que será de utilidad es el de *subtrabajo*.

Definición 2.2 (subtrabajo) *Sea f_j una función de ganancia y $T_j \subset [0,1]$ el conjunto de puntos de quiebre para f_j. Para dos puntos de quiebre sucesivos $t_1, t_2 \in T_j$, se define al subtrabajo de f_j como la función $g : [t_1, t_2] \to \mathbb{Q}_+$, que a cada $t \in [t_1, t_2]$ le asigna el valor $f_j(t)$*

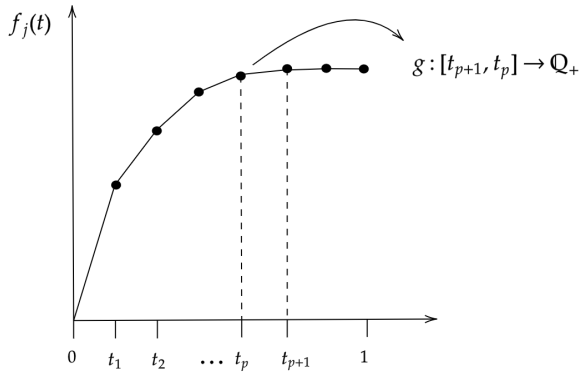


Figura 2.1: Para cada función de ganancia lineal por trozos f_j de un trabajo j cualquiera, esta se puede descomponer en los subtrabajos g que la conforman.

Dada la formulación de GCMP con transferencia, se puede pensar que una forma de encontrar la solución óptima para GCMP con transferencia es demostrando que es *reducible a Knapsack Fraccional*. Antes de demostrar dicha reducción, se realizan las definiciones de *valor de un subtrabajo*, *eficiencia de un subtrabajo* y *largo de un subtrabajo*, las cuales serán de utilidad más adelante.

Definición 2.3 (valor de un subtrabajo) *Dado un subtrabajo $g : [t_1, t_2] \to \mathbb{Q}$ de algún trabajo $j \in J$, su valor está dado por la siguiente cantidad:*

$$val(g) := g(t_2) - g(t_1)$$

Definición 2.4 (eficiencia de un subtrabajo) *La* eficiencia *de un subtrabajo $g : [t_1, t_2] \to \mathbb{Q}_+$, denotada por $eff(g^j)$, está dada por su pendiente:*

$$eff(g) := \frac{g(t_2) - g(t_1)}{t_2 - t_1}$$

Definición 2.5 (Largo de un subtrabajo) *Dada un conjunto de funciones de ganancias lineales por trozo $\{f_j\}_{j=1}^n$, un trabajo $j \in J$ cualquiera y un subtrabajo $g : [t_1, t_2] \to \mathbb{Q}_+$ de el, se define el largo ℓ_g de g de la siguiente forma:*

$$\ell_g := t_2 - t_1$$

Definición 2.6 (subtrabajo activo) *Dadas una asignación $A : J \to \mathbb{Q}_+$, un trabajo j y un subtrabajo $g : [t_1, t_2] \to \mathbb{Q}_+$ de j, se dirá que g es activo si $A_j > t_1$.*

La siguiente proposición caracteriza la solución óptima, tanto para GCMP con y sin transferencia.

Proposición 2.7 (Caracterización de Asignación óptima) *Sean J y $\{f_j\}_{j\in J}$ una instancia de GCMP en una única máquina de capacidad arbitraria, y sea $A : J \to \mathbb{Q}_+$ una asignación óptima. Entonces, los subtrabajos g, activos según A, son al menos tan eficientes como los subtrabajos inactivos g':*

$$\mathit{eff}(g) \geq \mathit{eff}(g') \quad \forall g \text{ activo, } g' \text{ inactivo} \tag{2.1}$$

Demostración. Por contradicción, sean g, g' subtrabajos activo e inactivo, respectivamente, y tales que:

$$\text{eff}(g') > \text{eff}(g) \tag{2.2}$$

Notar que g y g' provienen de trabajos j y j', respectivamente, con $j \neq j'$ (Esto ya que, si $j = j'$ entonces, por concavidad de f_j, si g' es inactivo también lo es g). Pueden ocurrir dos casos:

1. $g : [t_1, t_2] \to \mathbb{Q}_+$ es el último subtrabajo activo del trabajo j del cual proviene. Sea $0 < \varepsilon \leq A_j - t_1$. Se define una nueva asignación $\tilde{A}$ de la siguiente forma:

$$\tilde{A}_k := \begin{cases} A_{j'} + \varepsilon & \text{si } k = j' \\ A_j - \varepsilon & \text{si } k = j \\ A_j & \sim \end{cases} \tag{2.3}$$

   Entonces $\text{val}(\tilde{A}) > \text{val}(A)$, pues

   $$\text{val}(\tilde{A}) - \text{val}(A) = f_{j'}(A_{j'} + \varepsilon) - f_{j'}(A_{j'}) - [f_j(A_j) - f_j(A_j - \varepsilon)] \geq \varepsilon \cdot \text{eff}(g') - \varepsilon \cdot \text{eff}(g) > 0$$

2. En caso de que no sea el último subtrabajo activo de j, sin perdida de generalidad se puede considerar como tal y repetir la misma demostración que en 1, ya que las funciones de ganancia son decrecientes.

En cualquiera de los casos, a partir de la existencia de g y g' se puede definir una nueva asignación $\tilde{A}$ que genera ganancia estrictamente mayor a A, lo cual es una contradicción, y entonces concluye la demostración. □

Con los conceptos recién definidos, se está en condiciones de definir Knapsack Fraccional y demostrar la reductibilidad.

Definición 2.8 (Knapsack Fraccional) *Sea una mochila de capacidad $W \in \mathbb{R}_+$, un conjunto de n items $\{b_p\}_{p=1}^n$, un conjunto de pesos para cada uno de ellos dado por $\{w_p\}_{p=1}^n$, y un conjunto de valores para cada uno de estos items $\{v_p\}_{p=1}^n$.* Knapsack Fraccional *consiste en encontrar un vector $x \in [0,1]^n$ que define la proporción de cada item $\{b_p\}_{p=1}^n$ a cargar en una mochila de capacidad W, de modo de maximizar el valor total dado por $\sum_{p=1}^n x_p v_p$.*

$$\max_{x} \quad \sum_{p=1}^{n} x_p v_p \tag{2.4a}$$

$$\text{s.t.} \quad \sum_{p=1}^{n} x_p w_p \leq W \tag{2.4b}$$

$$x \in [0,1]^n \tag{2.4c}$$

Teorema 2.9 *GCMP con transferencia se puede reducir a Knapsack Fraccional (KF).*

Demostración. En efecto, considera la siguiente instancia de KF definida partir de una instancia arbitraria de GCMP con transferencia:

- Sea $\mathcal{S}$ el conjunto de los subtrabajos de J. Se define entonces un item b_g para cada subtrabajo $g \in \mathcal{S}$, con $g : [t_1^g, t_2^g] \to \mathbb{Q}_+$.
- Para cada item b_g, sea su valor $v_g := \text{val}(g)$, y su peso $w_g = \ell_g$, es decir el largo del subtrabajo g.
- Por último considere que la mochila tiene una capacidad de $W := m$

Con ello, sea el programa lineal 2.4 con la instancia recién definida para KF. Ahora se mostrará que toda asignación factible para GCMP con transferencia induce una solución factible para 2.4. Sea $N = |\mathcal{S}|$, y A una asignación factible arbitraria para GCMP con transferencia. Se define $x \in [0,1]^N$ de la siguiente forma: Sea $g : [t_1^g, t_2^g] \to \mathbb{Q}_+$ un subtrabajo arbitrario en $\mathcal{S}$ y j el trabajo asociado

$$x_g := \begin{cases} 1 & \text{si } A_j > t_2^g \\ 0 & \text{si } A_j \leq t_1^g \\ \dfrac{A_j - t_1^g}{\ell_g} & \sim \end{cases}$$

Luego $x \in [0,1]$, ya que cuando $A_j \in [0,1] \setminus (t_1^g, t_2^g]$ por definición $A_j \in \{0,1\}$, y cuando $A_j \in (t_1^g, t_2^g]$, entonces

$$\frac{A_j - t_1^g}{\ell_g} \geq \frac{t_1^g - t_1^g}{\ell_g} \geq 0 \quad \wedge \quad \frac{A_j - t_1^g}{\ell_g} \leq \frac{t_2^g - t_1^g}{\ell_g} \leq 1$$

Por otro lado, x satisface 2.4b. Considere para cada j el conjunto $\mathcal{S}_j$ de subtrabajos de j:

$$\begin{aligned} \sum_{g \in \mathcal{S}} x_g w_g &= \sum_{j \in J} \sum_{g \in \mathcal{S}_j} x_g w_g \\ &= \sum_{j \in J} \left(\sum_{\substack{g \in \mathcal{S}_j \\ A_j > t_2^g}} x_g w_g + \sum_{\substack{g \in \mathcal{S}_j \\ t_1^g < A_j \leq t_2^g}} x_g w_g \right) \\ &= \sum_{j \in J} \left(\sum_{\substack{g \in \mathcal{S}_j \\ A_j > t_2^g}} \ell_g + \sum_{\substack{g \in \mathcal{S}_j \\ t_1^g < A_j \leq t_2^g}} A_j - t_2^g \right) \end{aligned}$$

Dado un trabajo $j \in J$, existe a lo más un subtrabajo $g^* \in \mathcal{S}_j$ tal que $t_1^g < A_j \leq t_2^g$, y entonces la sumatoria de la derecha es:

$$\sum_{\substack{g \in \mathcal{S}_j \\ A_j \leq t_2^g}} A_j - t_2^g = A_j - t_2^{g^*}$$

Además, la suma $\sum_{g \in \mathcal{S}_j,\ A_j > t_2^g} \ell_g$ es telescópica, pues $\ell_g = t_2^g - t_1^g$, donde $[t_1^g, t_2^g]$ son intervalos sucesivos. Luego:

$$\sum_{\substack{g \in \mathcal{S}_j \\ A_j > t_2^g}} \ell_g = \sum_{\substack{g \in \mathcal{S}_j \\ A_j > t_2^g}} t_2^g - t_1^g = t_2^{g^*} - 0 = t_2^{g^*}$$

Entonces

$$\sum_{g \in \mathcal{S}} x_g w_g = \sum_{j \in J} \left(t_2^{g^*} + A_j - t_2^{g^*} \right) = \sum_{j \in J} A_j \leq W$$

Así, x es efectivamente una solución factible para KF. Para concluir, basta demostrar que existe una solución óptima de KF que induce una asignación factible para GCMP con transferencia del mismo valor. Se sabe que una solución óptima para KF se obtiene de la siguiente forma:

1. Ordenar los items decrecientemente según la cantidad $\text{eff}(g) = v_g / w_g$.
2. De manera glotona, agregar los items b_g completamente, es decir definir $x_g := 1$ en el orden descrito anteriormente, hasta que no se puedan seguir agregando items de manera completa, i.e. hasta que ocurra $\sum_g x_g w_g > W$.
3. Agregar el siguiente item $b_{g'}$ tanto como quede de capacidad disponible, i.e. $x_{g'} := \ell_g^{-1} \max\{0, W - \sum_{g \in \mathcal{S},\ x_g = 1} w_g\}$

Se sabe que el procedimiento anterior entrega una solución de valor máximo para KF. Queda demostrar entonces que dicha solución x *induce* una solución $A : J \to \mathbb{Q}_+$. Sea $\mathcal{P} := \{g \in \mathcal{S} : x_g = 1\} \cup \{g'\}$, con g' el subtrabajo del trabajo que se agrega en 3 y j' el trabajo asociado a dicho subtrabajo g'. Para ello basta considerar que, al momento de definir cada item b_g, este tiene asociados un subtrabajo $g_p : [t_1, t_2] \to \mathbb{Q}_+$ y un trabajo $j_p \in J$. Luego una forma de definir la asignación A inducida por x es la siguiente:

$$A_j := \sum_{g \in \mathcal{S}_j} x_g \ell_g \quad \forall j \in J$$

Sea $\text{val}(x) := \sum_{g \in \mathcal{P}} v_g x_g$. Notar que para todo $j \neq j'$ se tiene que f_j es cóncava, luego $\mathcal{P} \cap \mathcal{S}_j$ es un conjunto de subtrabajos consecutivos, y entonces $\{[t_2^g, t_1^g]\}_{g \in \mathcal{S}_j \cap \mathcal{P}}$ es un conjunto de intervalos consecutivos. Lo anterior implica que:

$$\sum_{g \in \mathcal{S}_j \cap \mathcal{P}} \ell_g = t_2^{g^j}$$

Con $g^j : [g_1^j, g_2^j] :\to \mathbb{Q}_+$ el último subtrabajo activo asociado a j. Es decir, la sumatoria de la izquierda es una suma telescópica, y entonces queda el último término, denotado por $t_2^{g^j}$, y el inicial que es nulo. Dado lo anterior, se tiene que si $j \neq j'$:

$$f_j(A_j) = f_j\left(\sum_{g \in \mathcal{S}_j} x_g \ell_g \right) = f_j\left(\sum_{g \in \mathcal{S}_j \cap \mathcal{P}} \ell_g \right) = f_j(t_2^{g^j})$$

Análogamente a que $\sum_{g\in\mathcal{S}_j\cap\mathcal{P}} \ell_g$ es una suma telescópica, $\sum_{g\in\mathcal{S}_j\cap\mathcal{P}} v_g$ también lo es (pues $v_g = g(t_2^g) - g(t_1^g)$). Como $g(0) = 0$, se tiene que

$$f_j(t_2^{g^j}) = \sum_{g\in\mathcal{S}_j\cap\mathcal{P}} v_g$$

Y entonces:

$$f_j(A_j) = \sum_{g\in\mathcal{S}_j} v_g x_g$$

La ecuación anterior también se satisface para j'. En efecto, para j' se tiene que g' es el último subtrabajo procesado de j', así:

$$\begin{aligned} f_{j'}(A_{j'}) &= f_{j'}\left(\sum_{g\in\mathcal{S}_{j'}} \ell_g x_g\right) \\ &= f_{j'}\left(\sum_{\substack{g\in\mathcal{S}_{j'} \\ g\neq g'}} \ell_g x_g + \ell_{g'} x_{g'}\right) \\ &= f_{j'}(t_1^{g'} + x_{g'}\ell_{g'}) \\ &= f_{j'}(t_1^{g'}) + x_{g'} v_{g'} \end{aligned}$$

donde la última desigualdad viene del hecho que $t_1^{g'}$ es un punto de quiebre y $f_{j'}$ es lineal por trozos. Análogamente al caso $j \neq j'$, $f(t_1^{g'}) = \sum_{g\in\mathcal{S}_{j'},\ g\neq g'} v_g x_g$, entonces:

$$f(A_{j'}) = f_{j'}(t_1^{g'}) + x_{g'} v_{g'} = \sum_{\substack{g\in\mathcal{S}_{j'} \\ g\neq g'}} v_g x_g + x_{g'} v_{g'} = \sum_{g\in\mathcal{S}_{j'}} v_g x_g$$

Con lo anterior se puede mostrar que $\mathrm{val}(A) = \mathrm{val}(x)$:

$$\mathrm{val}(A) = \sum_{j\in J} f_j(A_j) = \sum_{j\in J}\sum_{g\in\mathcal{S}_j} x_g v_g = \sum_{g\in\mathcal{S}} x_g v_g = \mathrm{val}(x)$$

Por último, A satisface la restricción de capacidad para una máquina de capacidad m:

$$\sum_{j\in J} A_j = \sum_{j\in J}\sum_{g\in\mathcal{S}_j} x_g \ell_g = \sum_{g\in\mathcal{S}} x_g \ell_g \leq W$$

Donde la última desigualdad viene de los pasos 2 y 3 para la construcción de x. Luego A es una asignación factible para GCMP con transferencia y tal que $\mathrm{val}(A) = \mathrm{val}(x)$, y entonces GCMP con transferencia se puede reducir a KF. □

El siguiente algoritmo *Glotón GCMP con transferencia* detalla los pasos para poder obtener la asignación óptima en dicho caso:

Algorithm 2 Glotón GCMP con transferencia

1: **Inputs:** $J, \{f_j\}_{j\in J}$
2: **for** $j \in J$ **do**
3: $\quad \mathcal{S}_j \leftarrow \{g : [t_1, t_2] :\to \mathbb{Q}_+ : g$ es subtrabajo de $f_j\}$ ▷ definir el conjunto de subtrabajos para cada $j \in J$
4: **end**
5: $S \leftarrow \bigcup_{j\in J} \mathcal{S}_j$ y ordenar de forma decreciente según $\text{val}(g)/(t_2 - t_1)$ $= \text{eff}(g)$
6: **for** $j \in J$ **do**
7: $\quad A_j \leftarrow 0$ ▷ incializar los largos de los subtrabajos como 0
8: **while** $\sum_{j\in J} A_j \leq 1$ **do**
9: $\quad g \leftarrow$ primer subtrabajo de S
10: $\quad S \leftarrow S \setminus \{g\}$
11: $\quad A_j \leftarrow A_j + t_2 - t_1$ con j el trabajo del cual proviene g
12: **end**
13: $A_j \leftarrow 1 - \sum_{\substack{j'\in J \\ j'\neq j}} A_{j'}$ ▷ el último trabajo se le quita el tiempo que sobra.
14: **Return:** A

Finalmente, veremos la complejidad del algoritmo 2.

Corolario 2.10 *La complejidad del algoritmo 2* Glotón GCMP con transferencia *es* $\mathcal{O}(n^2)$

Demostración. La mayor cantidad de iteraciones se dan entre los pasos 8 – 12, donde en el peor de los casos se pasa por todos los subtrabajos. Como existen n trabajos con $\mathcal{O}(n)$ subtrabajos, el algoritmo 2 es $\mathcal{O}(n^2)$ en el peor caso. □

Capítulo 3

GCMP sin transferencia: Complejidad

En este capítulo se demuestra que el problema de decisión asociado a GCMP sin transferencia es fuertemente NP-completo. Esto justifica la búsqueda de algoritmos de aproximación.

Definición 3.1 (GCMP sin transferencia, versión de decisión) *Dado un número racional $M > 0$, la versión de decisión para GCMP sin transferencia consiste en determinar si existe $A: I \times J \to \mathbb{Q}_+$ factible tal que la ganancia val(A) es al menos M.*

Luego, esta sección se centrará en demostrar el siguiente teorema.

Teorema 3.2 *GCMP sin transferencia es fuertemente NP-completo.*

Para demostrar lo anterior, se mostrará que 3-partición puede ser reducido a GCMP sin transferencia. Para ello, recordar de la definición de 3-partición.

Definición 3.3 (3-partición) *Sea $S = \{x_\tau\}_{\tau \in [3m]} \subset \mathbb{N}$ tal que $|S| = 3m$, para algún $m \in \mathbb{N}$. 3-partición consiste en decidir si es posible encontrar una partición del conjunto S en m 3-tuplas, de manera que cada tupla sume lo mismo.*

Con lo anterior, se está en condiciones para dar la demostración del Teorema 3.2.

Demostración. Sean $m \in \mathbb{N}, S \subset \mathbb{N}$ tal que $|S| = 3m$ y sea $K = \frac{\sum S}{m}$. Considere además que, para cada $x \in S$, se cumplen el siguiente par de desigualdades:

$$\frac{1}{4} < \frac{x}{K} < \frac{1}{2} \tag{3.1}$$

Se sabe [GJ79b] que 3-partición es fuertemente NP-completo, incluso si se restringe a instancias que satisfacen 3.1. Para hacer la reducción, se define la siguiente instancia de GCMP a partir de la instancia de 3-partición descrita arriba.

- Sea el conjunto de trabajos $J_{3P} := \{1, 2, \ldots, 3m\}$. Para cada trabajo $j \in J_{3P}$, se define la siguiente función de ganancia f_j:

$$f_j(t) = \begin{cases} t & \text{si } t \in [0, x_j/K] \\ x_j/K & \text{si } t > x_j/K \end{cases} \tag{3.2}$$

  Es decir, f_j tiene largo x_j/K y pendiente 1 antes de su quiebre. Ver Figura 3.1.
- Sea $I_{3P} := \{i_1, i_2, \ldots, i_m\}$ el conjunto de máquinas, cada una con capacidad 1.

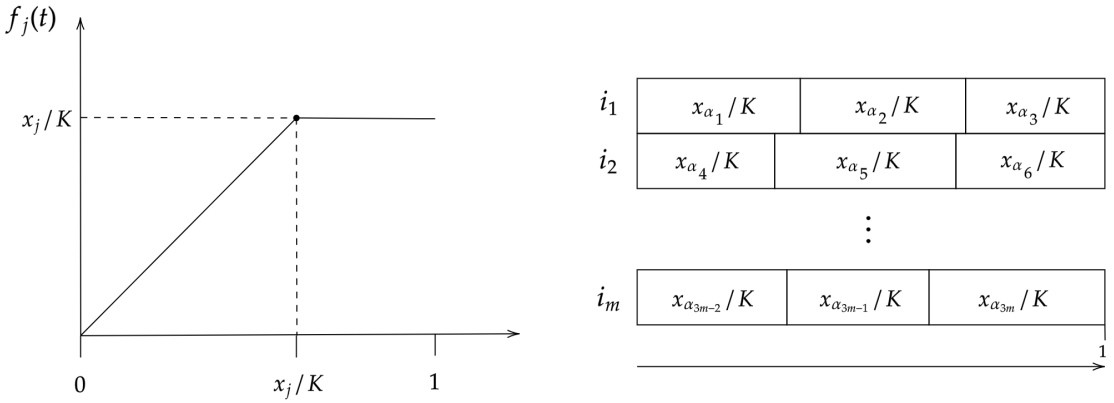


Figura 3.1: A la izquierda, un ejemplo de función de ganancia definida a partir del elemento $x_j \in S$. A la derecha, se representa con el tamaño de los bloques el tiempo de proceso x_τ/K de cada trabajo asignado, cuando $\text{Val}(A) = m$.

Por último, se cumple la siguiente equivalencia: Existe una 3–partición para S si y solo si para J_{3P}, I_{3P} y $\{f_j\}_{j \in J_{3P}}$ existe una asignación óptima A tal que $\text{Val}(A) \geq m$. Se demostrará esta equivalencia. Considere la implicancia directa. Sean $\mathrm{T}_1, \ldots, \mathrm{T}_m$ una partición de los índices $[3m]$ tal que

$$[3m] = \bigcup_{i \in [3m]}^{\cdot} \mathrm{T}_i, \ \sum_{j \in T_i} x_j = K \quad \forall i \in [3m]$$

La idea es que cada T_i se puede asociar con una de las máquinas de I_{3P}. En efecto, basta definir la siguiente asignación $A : I_{3P} \times J_{3P} \to \mathbb{Q}_+$

$$A_{ij} := \begin{cases} x_j/K & \text{si } j \in \mathrm{T}_i \\ 0 & \sim \end{cases}, \quad \forall i \in [3m], \ j \in [m] \tag{3.3}$$

Luego, A satisface todas las restricciones para ser una asignación factible de GCMP sin transferencia. En efecto, para todo $i \in I_{3P}$, $\sum_{j \in J_{3P}} A_{ij} = \sum_{j \in \mathrm{T}_i} A_{ij} = \sum_{j \in \mathrm{T}_i} x_j/K = 1$, luego cada máquina respeta la restricción de capacidad. Por otro lado, como $\{T_i\}_{i \in [3m]}$ es 3-partición, cada uno de los trabajos $j \in J_{3P}$ está únicamente asignado a una máquina. Finalmente, por construcción de las funciones de ganancia $\{f_j\}_{j \in J_{3P}}$ y usando que cada trabajo j es asignado

a exactamente una máquina con un largo $\frac{x_j}{K}$:

$$\begin{aligned}\text{val}(A) &= \sum_{j \in J_{3P}} f_j \left(\sum_{i \in [3m]} A_{ij} \right) \\ &= \sum_{j \in J_{3P}} f_j \left(\frac{x_j}{K} \right) \\ &= \sum_{j \in J_{3P}} \frac{x_j}{K} \\ &= m\end{aligned}$$

Lo que concluye la implicancia directa. Para la implicancia reversa, considere una asignación A con $\text{val}(A) \geq m$. Se cumplen los siguientes hechos:

1. Cada máquina debe ocupar toda su capacidad 1 de proceso y $\text{val}(A) = m$. Esto ya que, para cada $j \in J_{3P}$, la ganancia que estos generan es menor o igual al tiempo de proceso, como se establece en la definición de f_j en 3.2. Como solo existe m tiempo de proceso disponible entre todas las m máquinas, cada una debe estar completamente ocupada para satisfacer $\text{Val}(A) \geq m$. Lo anterior además implica que $\text{val}(A) = m$ y todos los trabajas en J_{3P} están activos.
2. Todos los trabajos $j \in J_{3P}$ tienen que ser procesados en x_j/K unidades de tiempos, i.e. $t_j = x_j/K$, pues si existiese algún $j' \in J_{3P}$ tal que $t_{j'} < x_{j'}/K$, entonces $f_{j'}(t_{j'}) < x_{j'}/K$, y dada la definición de los f_j, siempre se cumple que $f_j(t_j) \leq x_j/K$ para el resto de trabajos, se tendría que $\text{Val}(A) < m$, lo que es una contradicción. Luego $t_j \geq x_j/K$ para todo j. Además, si existiese otro trabajo $j' \in J_{3P}$ tal que $t_{j'} > x_{j'}/K$, entonces el tiempo de proceso de los trabajos activos sería mayor que la capacidad total, i.e.

$$\sum_{j \in J_{3P}} t_j > \sum_{j \in J_{3P}} \frac{x_j}{K} = m \qquad (\star)$$

   Como todas las m máquinas cuentan con capacidad 1 de tiempo de proceso, y dado el punto 1., el tiempo de proceso de los trabajos activos es exactamente m, lo cual es una contradicción con $(\star)$.
3. Gracias a la condición (3.1) y el punto 2., cada máquina está ocupada por exactamente 3 trabajos, pues como cada máquina ocupa su capacidad total 1 de tiempo de proceso, no pueden existir máquinas con dos trabajos asignados, mientras que tampoco puede tener más de tres trabajos sin superar la capacidad de 1 de la máquina.

Así, cada máquina completa exactamente 3 trabajos en sus respectivos tiempos de proceso $t_j = x_j/K$, lo cual define exactamente una 3–partición, cada una sumando un total de 1, lo cual concluye la demostración. □

Capítulo 4

GCMP sin transferencia: solo trabajos cortos

Dado $k \in \mathbb{N}$ fijo, se describirá en este capítulo una $(1+1/k)$-aproximación polinomial para GCMP sin transferencia en el caso que todos los trabajos asignados son cortos respecto a $1/k$, es decir se considera la restricción adicional de que para que una asignación A sea factible es necesario que $A_{ij} \leq 1/k \ \forall i \in I, \ j \in J$.

Basándose en el resultado de eficiencias de la proposición 2.7 y en el Teorema 2.1 para GCMP con transferencia, se plantea utilizar la misma solución de Knapsack Fraccional, i.e. utilizar el algoritmo 2 con la salvedad de que ahora se tiene que satisfacer, para cada trabajo $j \in J$, las restricciones 1.2c y 1.2b, que aseguran que ningún trabajo es procesado en más de una máquina. Se llamará *Glotón trabajos cortos* al Algoritmo 5 que consiste en la rutina para KF y una subrutina para evitar transferencia. Este algoritmo se describe a continuación.

Sean I, J y $\{f_j\}_{j\in J}$ una instancia de GCMP sin transferencia:

1. Se considera J y $\{f_j\}$ como una instancia de GCMP sin transferencia en una única máquina, llamada i^*, de capacidad m, la cual se resuelve utilizando el Algoritmo 2 GCMP con transferencia, pero añadiendo una modificación para que se satisfaga que $A_j \leq 1/k$. Dicho algoritmo se llamará *Knapsack - Trabajos cortos* (ver Algoritmo 3). Sea $A^* : I \to \mathbb{Q}_+$ la asignación resultante.
2. Se utiliza round-robin para separar los trabajos de i^* en las m máquinas de capacidad 1 unidad (sección 1.2.1). Sea A^r la asignación resultante de round-robin. Finalmente, y para evitar que exista la *transferencia* de trabajos entre máquinas distintas, se utiliza la siguiente subrutina:
   (a) Partiendo de la máquina i_1, considere j_u el último trabajo procesado en dicha máquina. Si j_u es procesado en 2 máquinas i_1 e i_2, entonces aumentar el valor de $A^r_{i_1 j_u}$ a $1/k$, i.e. procesarlo completamente en i_1. Al mismo tiempo, eliminar ese tiempo de proceso en i_2, i.e. $A^r_{i_2 j_u} \leftarrow 0$. Repetir esto para todas las máquinas donde puede haberse cortado un trabajo, es decir, para $i = i_1, \ldots, i_{m-1}$ (ver Algoritmo 4 *subrutina para eliminar transferencia de trabajos*). Sea $A^t : I \times J \to \mathbb{Q}_+$ la asignación resultante.

(b) Finalmente, definir una contracción a partir de A^t, denotada por A^G, como sigue:

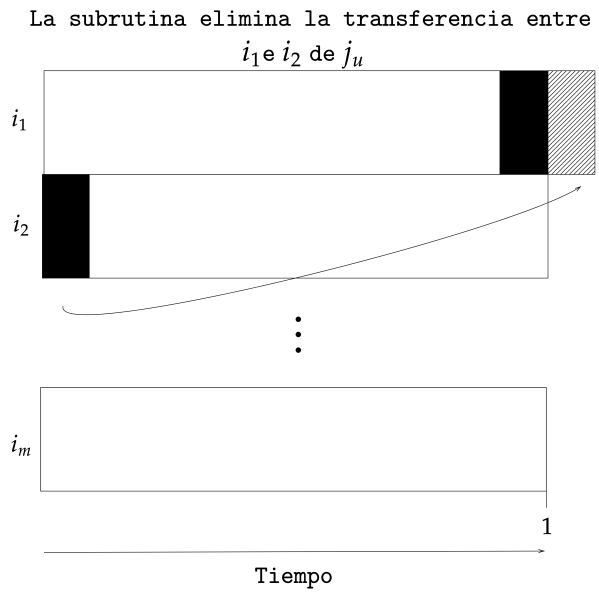
$$A^G = \frac{1}{1+1/k} A^t \tag{4.1}$$

Algorithm 3 Knapsack - Trabajos cortos

Inputs: $J, \{f_j\}_{j\in J}$
for $j \in J$ **do**
    $\mathcal{S}_j \leftarrow \{g : [t_1, t_2] :\to \mathbb{Q}_+ : g$ es subtrabajo de $f_j\}$ ▷ definir el conjunto de subtrabajos para cada $j \in J$
end
$S \leftarrow \bigcup_{j\in J} \mathcal{S}_j$ y ordenar de forma decreciente según $\text{val}(g)/(t_2 - t_1)$
for $j \in J$ **do**
    $A_j \leftarrow 0$ ▷ incializar los largos de los subtrabajos como 0
while $\sum_{j\in J} A_j \le m$ **do**
    $g : [t_1, t_2] \to \mathbb{Q}_+ \leftarrow$ primer subtrabajo de S
    $j \leftarrow$ trabajo asociado a g
    $S \leftarrow S \setminus \{g\}$
    if $A_j + t_2 - t_1 > 1/k$ **then** ▷ verificar si agregando el subtrabajo g
        $A_j \leftarrow 1/k$
    else
        $A_j \leftarrow A_j + t_2 - t_1$
    end
end
$A_j \leftarrow m - \sum_{\substack{j'\in J \\ j'\neq j}} A_{j'}$ ▷ el último trabajo se le asigna el tiempo que sobra.
Return: A

Algorithm 4 Eliminar transferencias

Inputs: A^r
for $p \in [m-1]$ **do**
    for $j \in J$ **do**
        if $A^r_{i_p j} > 0$ **and** $A^r_{i_{p+1} j} > 0$ **then**
            $A^r_{i_p j} \leftarrow 1/k$
            $A^r_{i_{p+1} j} \leftarrow 0$
        end
    end
End
Return A^r



El pseudocódigo del algoritmo 5, llamado *Glotón trabajos cortos*, se incluye a continuación.

Algorithm 5 GCMP - trabajos cortos

1: **Inputs:** $J, \{f_j\}_{j\in J}, I$
2: $A^* \leftarrow$ Algoritmo 3 Knapsack - Trabajos cortos $(J, \{f_j\}_{j\in J})$
3: $A^r \leftarrow$ Algoritmo 1 Round-Robin(A^*, I)
4: $A^t \leftarrow$ Algoritmo 4 Eliminar transferencias(A^r)
5: $A^G \leftarrow \dfrac{1}{1+1/k} A^t$
6: **Return:** A^G

Queda demostrar entonces que GCMP glotón efectivamente obtiene la asignación A^G buscada.

Proposición 4.1 *Dado $k \in \mathbb{N}$ fijo, el Algoritmo 5* GCMP - Trabajos cortos *entrega una $(1+1/k)$-aproximación para GCMP sin transferencia cuando todos los trabajos son cortos.*

Demostración. Como se demostró en el Teorema 2.9, el Algoritmo 3 resuelve de manera exacta el caso GCMP sin transferencia en una máquina (notar que si bien en el capítulo 2 se revisó el caso con transferencia, ambos casos de GCMP coinciden para el caso de una máquina), y entonces A^* es óptimo para la instancia de la máquina i^*. Por otro lado, sea A^{OPT} la asignación óptima de GCMP sin transferencia con trabajos cortos en las máquinas de I. A^{OPT} induce directamente una asignación factible para el caso de la máquina i^*, llamada $A' : J \to \mathbb{Q}_+$ dada por

$$A'_j := \sum_{i\in I} A^{\mathrm{OPT}}_{ij} \qquad \forall j \in J$$

Como para cada trabajo j su tiempo de proceso no cambia entre las asignaciones A^* y A^r al aplicar round-robin sobre la primera asignación, es claro que $\mathrm{val}(A') = \mathrm{val}(A^{\mathrm{OPT}})$. De la factibilidad de A^{OPT} para el caso de las m máquinas de I de capacidad 1 unidad se sigue la factibilidad de A' para la máquina i^*, pues $A'_j = \sum_{i\in I} A^{\mathrm{OPT}}_{ij} \leq 1$, donde la última desigualdad es la restricción **??**, y $\sum_{j\in J} A'_j = \sum_{j\in J}\sum_{i\in I} A^{\mathrm{OPT}}_{ij} = \sum_{i\in I}\sum_{j\in J} A^{\mathrm{OPT}}_{ij} \leq \sum_{i\in I} 1 = m$, y como m es la capacidad de i^*, se concluye que A' es una asignación de igual valor que A^{OPT} para el caso de i^*. Como A^* es óptima para dicho caso por lo explicado en el párrafo anterior, se cumple que:

$$\mathrm{val}(A^{\mathrm{OPT}}) = \mathrm{val}(A') \leq \mathrm{val}(A^*)$$

Ahora se verá que A^r logra ser una asignación de GCMP sin transferencia en las m máquinas de I que mantiene el valor de A^*. En efecto, como round-robin no altera los tiempos de proceso de la asignación A^*, se sigue que $\mathrm{val}(A^*) = \mathrm{val}(A^r)$. Respecto de la eliminación de transferencias en la asignación A^r, esta se basa en dos hechos:

1. El Algoritmo 1 round-robin, de generar transferencia de trabajos, lo hace entre máquinas sucesivas (ver paso 11)

2. Dado lo anterior, como la subrutina 4 verifica si hubo transferencia para cada uno de los trabajos de J entre máquinas sucesivas, se cumple que la asignación resultante A^t no presenta transferencias. Además, al aplicar el Algoritmo 3 obtenemos una asignación A^t tal que para cada máquina $i \in I$ se usa una capacidad de a lo más $1 + 1/k$:

$$\begin{aligned}\sum_{j \in J} A^t_{ij} &\leq \left[\sum_{j \in J} A^r_{ij}\right] + \frac{1}{k} \\ &\leq 1 + \frac{1}{k}\end{aligned}$$

Finalmente, como A^G es una contracción por un factor $(1 + 1/k)^{-1}$, se tiene que:

$$\text{val}(A^G) \geq \frac{1}{1 + 1/k}\text{val}(A^t) = \frac{1}{1 + 1/k}\text{val}(A^*) \geq \frac{1}{1 + 1/k}\text{val}(A^{\text{OPT}}) \tag{4.2}$$

donde la primera desigualdad viene de la concavidad y el hecho que A^G sea una contracción (ver Lema 1.7), mientras que la igualdad viene del hecho de que round-robin y la subrutina no modifican los tiempos de proceso de la asignación A^*. Finalmente, la factibilidad sale del hecho que al contraer:

$$\sum_{j \in J} A^G_{ij} = \frac{1}{1 + 1/k} \cdot \sum_{j \in J} A^t_{ij} \leq 1$$

□

Corolario 4.2 (Complejidad Algoritmo 5) *La complejidad del algoritmo 5 es* $\mathcal{O}(nm + n^2)$

Demostración. En el paso 2, el algoritmo 5 es $\mathcal{O}(n^2)$, lo cual se vio en el Corolario 2.10. El algoritmo 1 round-robin es $\mathcal{O}(n)$, pues en el peor caso recorre todos los trabajos, y el algoritmo 4 para eliminar las transferencias entre máquinas, al recorrer para cada máquina todos los trabajos, es $\mathcal{O}(nm)$. En conclusión, el algoritmo 5 es $\mathcal{O}(nm + n^2)$ □

Capítulo 5

GCMP sin transferencia: solo trabajos largos

Sea $k \in \mathbb{N}$ fijo. En este capítulo se estudiará el siguiente problema: Encontrar la mejor solución factible A para GCMP sin transferencia tal que para cada trabajo activo j, su asignación a la única máquina en que es asignado, es de largo al menos $1/k$. Se dice entonces que todos los trabajos activos j tienen una asignación larga respecto a k. Para fijar ideas, considere la siguiente definición de *GCMP sin transferencia con trabajos largos*:

Definición 5.1 (GCMP sin transferencia - Trabajos largos) *Sean I, J y $\{f_j\}_{j\in J}$ una instancia arbitraria de GCMP sin transferencia.* GCMP sin transferencia - Trabajos largos *considera el GCMP sin transferencia de la definición 1.5, con la restricción adicional de que todo trabajo j activo en alguna máquina i debe satisfacer que $A_{ij} \geq 1/k$.*

Además, para poder determinar el algoritmo que resuelva este caso, se considerará trabajar con asignaciones aún más restringidas, las que llamaremos asignaciones k-redondeadas.

Definición 5.2 (Asignación k-redondeada) *Dado $k \in \mathbb{N}$, se dirá que una asignación es k-redondeada siempre que A tome valores en el siguiente conjunto $\mathcal{L}$:*

$$\mathcal{L} := \{\ell \in \mathbb{Q}^+ : \ell = z/k^2,\ z = k, k+1, \ldots, k^2\} \cup \{0\}$$

Notar que si A toma valores en $\mathcal{L}$, necesariamente los trabajos activos según A son largos.

El primer resultado es que no se pierde mucho por el hecho de solo usar trabajos redondeados.

Proposición 5.3 *Considere por un momento una asignación óptima A^{OPT} del problema GCMP trabajos largos, y sea $\tilde{A}^{OPT}$ la asignación obtenida de redondear los tiempos de proceso*

de los trabajos A^{OPT}-activos al mayor múltiplo entero de $1/k^2$ menor a él. Entonces:

$$\left(1 - \frac{1}{k+1}\right) val(A^{OPT}) \leq val(\tilde{A}^{OPT}) \tag{5.1}$$

Demostración. En efecto, lo anterior viene del hecho de que se puede pensar en $\tilde{A}^{\mathrm{OPT}}$ como una *contracción* de A^{OPT}, solo que se aplica un factor diferente por trabajo. De hecho, llame α a un factor arbitrario para redondear un trabajo j activo según A^{OPT}. Se tiene que:

$$\alpha \geq \frac{k}{k+1}$$

Luego, considere δA^{OPT} con $\delta = \dfrac{k}{k+1}$. Como esta contracción es más restrictiva que el redondeo $\tilde{A}^{\mathrm{OPT}}$, se cumple que:

$$\mathrm{val}(\tilde{A}^{\mathrm{OPT}}) \geq \mathrm{val}(\delta A^{\mathrm{OPT}}) \geq \left(1 - \frac{1}{k+1}\right) \mathrm{val}(A^{\mathrm{OPT}})$$

lo que concluye la demostración. □

Dado que redondear permitirá encontrar una asignación que aproxime el valor óptimo, salvo por un factor de $1 + 1/k$, se supondrá en adelante que todas las asignaciones factibles toman valores en $\mathcal{L}$, i.e. $A : I \times J \to \mathcal{L}$.

Para poder resolver el presente caso se resolverá un *matching de peso máximo*. Para definirlo se usarán los conceptos de *configuración* y *configuración de máquinas*.

Definición 5.4 (Configuración) *Sean $J, I, \{f_j\}_{j=1}^n$ una instancia arbitraria de GCMP sin transferencia. Dada una máquina arbitraria $i \in I$, una* configuración $c \in \mathbb{N}^{\mathcal{L}}$ *de i es un vector tal que la coordenada $\ell-$ésima c_ℓ indica el número de trabajos de largo ℓ activos en dicha máquina. El conjunto de todas las configuraciones será denotado por $\mathcal{C}$ (ver Fig. 5).*

Notar que las configuraciones c son vectores que permiten determinar como está ocupado el tiempo en una máquina dada, i.e. cuántos trabajos procesa y en cuánto tiempo es procesado cada uno de ellos.

Observación: Notar que las configuraciones solo especifican cuantos trabajos utiliza una máquina para cada largo en $\mathcal{L}$. Como todos los trabajos activos tienen tiempo de proceso $\geq 1/k$, entonces no puede haber más de k trabajos asignados por máquina.

Definición 5.5 (Configuración de máquinas) *Sean $J, I, \{f_j\}_{j=1}^n$ una instancia arbitraria de GCMP sin transferencia. Considere un vector $\kappa \in \mathbb{N}^{\mathcal{C}}$ tal que, para $c \in \mathcal{C}$, si $\kappa_c = b \in \mathbb{N}$, entonces significa que hay b máquinas de I que ocupan la configuración c. Dicho vector κ se llamará una* configuración de máquinas. *Se dirá además que una configuración de máquinas es* factible *si satisface lo siguiente:*

$$\sum_{c \in \mathcal{C}} \kappa_c = m \tag{5.2}$$

Es decir, para que una configuración de máquinas sea factible *es necesario que se utilicen exactamente m máquinas.*

Un lema que será importante a la hora de evaluar la complejidad del Algoritmo para resolver GCMP trabajos largos es el siguiente:

Lema 5.6 (Cota número de configuraciones de máquinas factibles) *El número de configuraciones de máquinas factibles es a lo más $(m+1)^{|\mathcal{C}|}$, con $|\mathcal{C}| \leq (k+1)^{k^2}$, esto es $\mathcal{O}(m^{(k+1)^{k^2}})$*

Demostración. En efecto, lo anterior viene del hecho que cada configuración debe escoger entre las m máquinas de I, o no ser utilizada en ninguna máquina, lo que es un total de $(m+1)^{|\mathcal{C}|}$ formas de realizar esta selección. Por otro lado, una configuración $c \in \mathcal{C}$ se define, dada una máquina arbitraria, escogiendo para cada largo $\ell \in \mathcal{L}$ la cantidad de veces que dicho largo será ocupado en dicha máquina. Como cada largo ℓ puede ser escogido a lo más k o ninguna, y el conjunto de largos está acotado por k^2, entonces las configuraciones posibles son a lo más $(k+1)^{k^2}$, y entonces hay a lo más $m^{O(k+1)^{k^2}}$. □

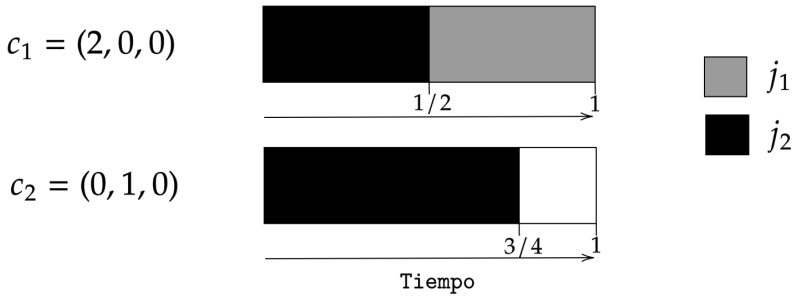


Figura 5.1: Para el caso $k = 2$, $J = \{j_1, j_2, j_3\}$, el conjunto de largos $\mathcal{L}$ está dado por $\mathcal{L} := \{1/2, 3/4, 1\}$. En la imagen se ven dos configuraciones c_1 y c_2, y se muestra como dos máquinas distintas utilizan dichas configuraciones. c_1 utiliza dos trabajos de largo $1/2$, j_1 y j_2. Notar que la configuración solo indica el tiempo de proceso de los trabajos largos a utilizar por una máquina arbitraria, mas no especifica *qué* trabajos. Por otro lado, la configuración c_2 indica que una máquina dada usará un único trabajo de largo $3/4$ (donde se puede ver en la imagen que se ha utilizado nuevamente el trabajo j_2), lo cual es factible dado que no supera la capacidad de dicha máquina dada por 1.

La idea principal detrás de las configuraciones de máquinas es que se puede iterar sobre ellas, y en cada iteración resolver un problema de matching de peso máximo para determinar la asignación de ganancia máxima para GCMP sin transferencia con tiempos de proceso en $\mathcal{L}$. En efecto, considere $\kappa \in \mathbb{N}^{\mathcal{C}}$ una configuración de máquinas factible, y sea $G = (L \cup R, E)$ el siguiente grafo bipartito completo:

Definición Grafo(κ):

1. Sea J el conjunto de nodos izquierdo, i.e. cada trabajo $j_1, j_2, \ldots, j_n$ tiene asociado un nodo.

2. La configuración de máquinas κ especifica para cada una de las configuraciones de $\mathcal{C}$ cuántas máquinas lo están utilizando. Como cada configuración de $\mathcal{C}$ indica cuántos tiempos de proceso de un determinado largo $\ell \in \mathcal{L}$ hay presentes, de κ se infiere cuántos "casilleros" en total hay presentes de largo ℓ entre todas las máquinas. Luego existirán para este largo ℓ tantos nodos $b_\ell := \sum_{c \in \mathcal{C}} \kappa_c \cdot c_\ell$ en R como casilleros del mismo largo haya entre todas las máquinas según κ (ver Fig. 5).
3. Sea $E = J \times R$ el conjunto de aristas, de modo que $G(\kappa)$ es un grafo bipartito completo.
4. Para una arista $e = jr \in E$, considere $\ell_r \in \mathcal{L}$ como el largo asociado al nodo $r \in R$. La *ganancia* de procesar el trabajo j en dicho tiempo de proceso será denotada por $g_e := f_j(\ell_r)$.

Observación: Es importante destacar que, para toda configuración de máquinas factible, el número de nodos del conjunto R está acotado por la cantidad mk. En efecto, como se están considerando asignaciones factibles únicamente las que tomen valores en $\mathcal{L}$, entonces por máquina no puede haber más de k trabajos largos activos, pues la capacidad de cada máquina es 1, y como son m máquinas se sigue que no hay más de mk trabajos activos. Con lo anterior se puede establecer el siguiente resultado.

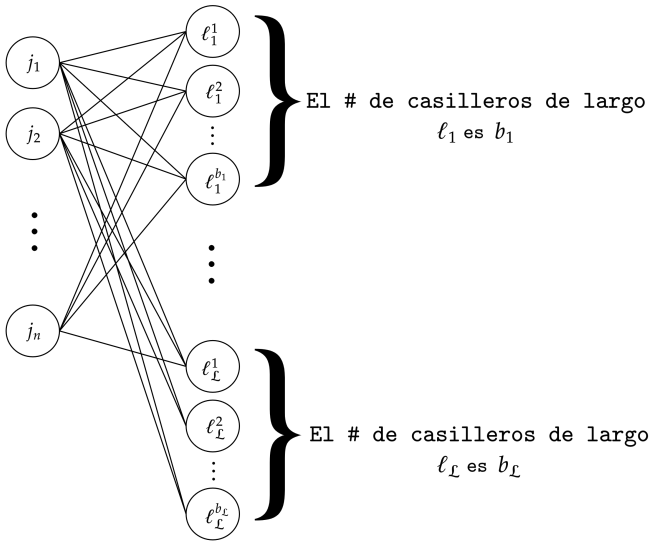


Figura 5.2: Grafo bipartito completo $G(\kappa)$ definido a partir de κ. Para cada largo ℓ presente en la configuración de máquinas, se agregan tantos nodos como indiquen el total de configuraciones de κ. Todos los trabajos son conectados a cada uno de estos nodos, y si un arco conecta un nodo j y un largo ℓ, su peso está dado por $g_e := f_j(\ell)$, es decir por la ganancia de procesar dicho trabajo j por un tiempo ℓ.

Proposición 5.7 *Sea A^{OPT} una asignación óptima para el presente caso de GCMP sin transferencia - solo trabajos largos en $\mathcal{L}$, sea κ^{OPT} la configuración de máquinas utilizada por A^{OPT}, y sea $G = G(\kappa^{opt})$ el grafo definido a partir de κ^{OPT}. Considere el siguiente programa*

lineal entero para matching de peso máximo*, denotado por* PLE_κ

$$\max_{x} \quad \sum_{e\in E} g_e x_e \tag{5.3a}$$

$$\text{s.t.} \quad \sum_{j\in J} x_{jr} = 1 \quad \forall r \in R \tag{5.3b}$$

$$\sum_{r\in R} x_{jr} \leq 1 \quad \forall j \in J \tag{5.3c}$$

$$x \in \{0,1\}^E \tag{5.3d}$$

Entonces se puede definir una asignación A^G *a partir de la solución de* PLE_κ*, denotada por* x^**, tal que* $val(A^G) = val(A^{OPT})$*.*

Demostración. En efecto, considere x^* una solución óptima de PLE_κ. Para poder explicar el algoritmo que permitirá obtener una asignación A^G inducida a partir de x^*, es necesario explicar como pasar de una configuración de máquinas κ a una lista de m máquinas y sus respectivos casilleros, los cuales serán finalmente los nodos que constituyan R. Para fijar ideas, sea $k = 3$, entonces el conjunto de largos $\mathcal{L}$ está dado por $\mathcal{L} = \{1/3, 2/3, 1\}$. Sean entonces las configuraciones $c_1 = (3,0,0)$ y $c_2 = (2,0,0)$, y sea κ definida de la siguiente forma:

$$\kappa_{c_1} = 2,\ \kappa_{c_2} = 1,\ \kappa_c = 0\ \forall c \in \mathcal{C} \setminus \{c_1, c_2\}$$

Es decir κ dice que 2 máquinas utilizan la configuración c_1, y 1 tercera máquina utiliza la configuración c_2. Ahora que cada máquina tiene una configuración, entonces se puede notar que R consiste de 6 nodos asociados a un largo 1/3 (que se corresponden con los casilleros azules), y un nodo de largo 2/3 (que se corresponde con el casillero de color verde en la máquina i_3)

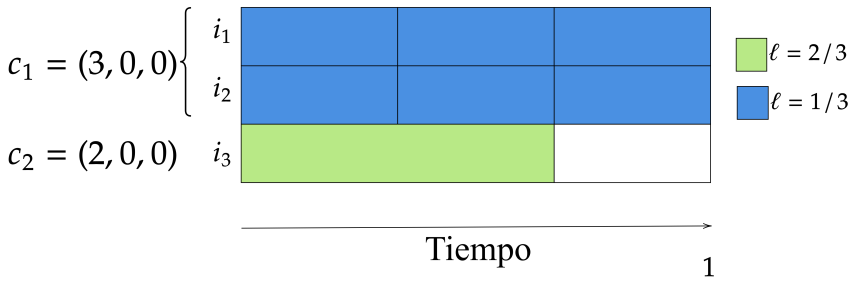


Figura 5.3: En la imagen, la configuración de máquinas definida anteriormente determina dos máquinas utilizando la configuración c_1 y otra utilizando la configuración c_2. Como $b_{1/3} = \sum_{c\in\mathcal{C}} \kappa_{\mathcal{C}} \cdot c_{1/3} = 2 \cdot 3 = 6$ (i.e., los seis casilleros/espacios de las máquinas i_1 e i_2), entonces existen 6 nodos de R, 3 asociados a i_1 y otros 3 asociados a i_2.

Considere el siguiente algoritmo para definir a A^G partir de dicha solución:

Algorithm 6 `asignación trabajos largos`

1: **Inputs:** $J,\ I,\ \kappa^O,\ x^*,\ d$
2: **for** $r \in R$ **do**
3: $\quad i \leftarrow d[r]$ ▷ obtener maquina desde el diccionario d
4: $\quad j_r \leftarrow j$ tal que $x^*_{jr} = 1$
5: $\quad A^G_{i_r j_r} \leftarrow \ell_r$
End
6: **Return** A^G

Notar primero que la complejidad del algoritmo 6 está determinada principalmente por el `for` del paso 2, el cual itera sobre el conjunto R de casilleros. Como solo puede haber a lo más k, y existen m máquinas en total, entonces no hay más de mk iteraciones sobre R, y entonces el algoritmo 6 tiene complejidad $\mathcal{O}(mk)$. Además, dicha asignación es k-redondeada, pues en el paso 6 solo se asignan largos de $\mathcal{L}$.

Veamos que la asignación A^G tiene el mismo valor que el peso del matching x^*, en efecto:

$$\text{val}(A^G) = \sum_{j\in J} f_j\left(\sum_{i\in I} A^G_{ij}\right) = \sum_{j\in J}\sum_{\substack{e\in\delta(j)\\ e=jr}} f_j(\ell_r)x^*_e = \sum_{e\in E} g_e x^*_e = \text{val}(x^*).$$

Lo que hace el algoritmo es, para cada casillero $r \in R$, recuperar la configuración c_r que dio origen a dicho casillero r (esto se puede guardar al momento de la construcción del grafo $G(\kappa^{\text{opt}})$); la máquina i_r que utiliza dicha configuración; y el trabajo j_r que utiliza dicho casillero en la máquina i_r según la solución x^*. Como la asignación A^{OPT} *induce* una solución x' factible para PLE_κ de igual valor (i.e. $\text{val}(x') = \text{val}(A^{\text{OPT}})$), se tiene que:

$$\text{val}(A^{\text{OPT}}) = \sum_{e\in E} g_e x'_e =\leq \text{val}(x^*) = \sum_{e\in E} g_e x^*_e = \text{val}(A^G)$$

Por otro lado, por ser A^{OPT} una asignación óptima para GCMP sin transferencia - solo trabajos largos, entonces $\text{val}(A^G) \leq \text{val}(A^{\text{OPT}})$. De las dos últimas desigualdades se deduce que $\text{val}(A^G) = \text{val}(A^{\text{OPT}})$, lo que concluye la demostración. □

Finalmente, se escribe el Algoritmo en su totalidad para el GCMP sin transferencia - solo trabajos largos:

Algorithm 7 GCMP sin transferencia - solo trabajos largos

Inputs: $J,\ I,\ \{f_j\}_{j\in J}$
$A : I \times J \to \mathbb{Q}_+ \leftarrow$ variable para guardar la mejor asignación
$\text{val}(A) \leftarrow 0$
for κ una configuración de máquinas factibles **do**
    $G = (L \cup R, E) \leftarrow$ el grafo asociado a la configuración de máquinas κ (fig. 5)
    $x^* \leftarrow$ solución del problema de matching de peso máximo PLE_κ
    $A^G \leftarrow$ Algoritmo 6 `asignación trabajos largos`
    if $\text{val}(A) > \text{val}(A^G)$ **then**
        $A \leftarrow A^G$
        $\text{val}(A) \leftarrow \text{val}(A^G)$
    end
end
Return: A

Corolario 5.8 *El algoritmo 7 entrega una asignación k-redondeada óptima para GCMP sin transferencia y trabajos largos. Este algoritmo tiene complejidad $\mathcal{O}(m^{k^k}(mk+n)^3)$.*

Demostración. Primero vemos que la asignación A del algoritmo 7 es k-redondeada, lo cual viene del hecho que se obtiene del algoritmo 6. Por otro lado, gracias a la proposición 5.7, se sabe que existe alguna iteración del `for` del algoritmo 7 tal que se utilizan la configuración de máquinas de alguna asignación óptima A^{OPT}, y entonces la asignación A entregada por el algoritmo 7 satisface $\text{val}(A) \geq \text{val}(A^{\text{OPT}})$, y luego es óptima.

Notar que el `for` itera sobre las configuraciones de máquinas, el cual está acotado por $\mathcal{O}(m^{k^k})$. Por otro lado, dentro del `for` se resuelve el PLE de matching de peso máximo, el cual se puede resolver en $\mathcal{O}((mk+n)^3)$, donde el sumando mk viene del hecho que no existen más de k trabajos largos por máquina. Luego la complejidad está dada por $\mathcal{O}(m^{k^k}(mk+n)^3)$ □

Capítulo 6

GCMP sin transferencia: Trabajos de largo arbitrario

En el presente capítulo se probará el siguiente resultado:

Teorema 6.1 *Para $0 \leq \varepsilon < 1$ y $k = k(\varepsilon) \in \mathbb{N}$, GCMP sin transferencia admite un $(1-\varepsilon)$-PTAS, y existe un polinomio $p : \mathbb{R}^3 \to \mathbb{R}$ tal que la complejidad está dada por $\mathcal{O}(p(n, m, k) \cdot m^{k^2})$.*

El procedimiento para llegar al resultado anterior será el siguiente:

1. Dado $k \in \mathbb{N}$, en vez de resolver el GCMP sin transferencia de la definición 1.5, se resolverá la siguiente relajación: nos restringiremos a buscar soluciones $A^k : I \times J \to \mathbb{Q}_+$ tales que los trabajos largos sean $k-$redondeados (ver definición 5.2). A este problema restringido lo llamaremos *problema redondeado*.
2. Encontrar una solución del problema redondeado puede ser difícil, entonces buscaremos una *relajación* de dicho problema. Además, esta estará dada por una colección de programas lineales enteros. Dichos PLEs serán una *relajación* en el sentido de que, si A^k fuese la solución óptima del problema redondeado, entonces A^k *induce* una solución x de igual valor que $\text{val}(A^k)$ y factible para alguno de los PLEs.
3. Finalmente, estos PLEs no podremos resolverlos de manera exacta en tiempo polinomial, por lo que se buscará una solución aproximada resolviendo una *relajación lineal* de dichos PLEs.

6.1. Problema Redondeado

Un primer lema que será de utilidad para relacionar los valores óptimos de GCMP sin transferencia y el problema redondeado es el siguiente.

Lema 6.2 *Sea I, J, $\{f_j\}_j$ una instancia de GCMP sin transferencia, y sea $k \in \mathbb{N}$ fijo. Si se denota por A^k a una asignación óptima del problema redondeado y A^o a una asignación*

óptima de GCMP sin transferencia, se tiene que:

$$val(A^k) \geq \left(\frac{k}{k+1}\right) val(A^o)$$

Demostración. Esta demostración es muy similar a la de la proposición 5.3, donde solo tratábamos el caso de solo trabajos largos. En este lema necesitaremos además considerar lo que sucede con los trabajos cortos. En efecto, basta considerar la siguiente asignación $\tilde{A}$ definida a partir de A^o: Sean i y j arbitrarios

$$\tilde{A}_{ij} := \begin{cases} q/k^2 & \text{si } A^o_{ij} > 1/k \text{ y } q = \max\{h \in \mathbb{N} : h/k^2 \leq A^o_{ij}\} \\ A^o_{ij} & \text{si } A^o_{ij} \leq 1/k \end{cases}$$

Es decir, $\tilde{A}$ consiste en la asignación A^o *redondeada*, de modo que los trabajos largos son procesados por $\tilde{A}$ en un tiempo tal que es múltiplo entero más grande de $1/k^2$ tal que no supera el tiempo de la asignación óptima A^o para procesar dicho trabajo largo, y los trabajos cortos son procesados en el mismo tiempo que A^o. Como A^o es una asignación factible para GCMP sin transferencia, entonces $\tilde{A}$ es factible para el problema redondeado, pues en la definición de $\tilde{A}$ se procesan los mismos trabajos de J, cada uno en la misma máquina, y por a lo más el mismo tiempo de proceso, lo que implica que no se supera la capacidad 1 de cada máquina y no hay transferencia de trabajos. Finalmente:

$$\begin{aligned} \text{val}(\tilde{A}) &= \sum_{j\in J}\sum_{i\in I} f_j(\tilde{A}_{ij}) \\ &= \sum_{\substack{j\in J \\ j \text{ es largo}}} \sum_{i\in I} f_j(\tilde{A}_{ij}) + \sum_{\substack{j\in J \\ j \text{ es corto}}} \sum_{i\in I} f_j(\tilde{A}_{ij}) \\ &\geq \sum_{\substack{j\in J \\ j \text{ es largo}}} \sum_{i\in I} \left(\frac{k}{k+1}\right) f_j(A^o_{ij}) + \sum_{\substack{j\in J \\ j \text{ es corto}}} \sum_{i\in I} f_j(A^o_{ij}) \\ &= \left(\frac{k}{k+1}\right) \text{val}(A^o) \end{aligned}$$

donde la desigualdad viene del hecho que las f_j son funciones cóncavas crecientes, y la definición de $\tilde{A}$ para trabajos largos. En efecto, si j es un trabajo largo procesado en una máquina i, entonces $A^o_{ij} > 1/k$, y de la definición de $\tilde{A}$ se sigue que:

$$\tilde{A}_{ij} \geq \left(\frac{k}{k+1}\right) A^o_{ij}$$

Como f_j es una función creciente, $f_j(\tilde{A}_{ij}) \geq f_j(k/(k+1))A^o_{ij})$, y de la concavidad y el hecho que $f_j(0) = 0$ se tiene que $f_j(k/(k+1))A^o_{ij}) \geq (k/(k+1))f_j(A^o_{ij})$, de donde se justifica la desigualdad del párrafo anterior y en consecuencia:

$$\text{val}(\tilde{A}) \geq \left(\frac{k}{k+1}\right) \text{val}(A^o)$$

□

6.2. Programa lineal entero asociado a una configuración de máquinas κ, a una eficiencia $\alpha > 0$ y una capacidad $T > 0$

El programa lineal entero tiene por objetivo tratar el *problema redondeado*. Una primera idea sería considerar un PLE que contemple un conjunto de variables $x_{ji\ell}$ con $j \in J$, $i \in I$ y ℓ un tiempo de proceso (o largo) posible para procesar dicho trabajo. Sin embargo, se advierte el problema respecto de que, dados $j \in J$ e i, dicho trabajo tiene la posibilidad de ser procesado en i en varios tiempos de proceso, de modo que para formular que i procese a j en un tiempo determinado, habría que darle la posibilidad a ℓ de tomar todos los valores posibles de tiempo de proceso, los cuales no estarían acotados polinomialmente. Una forma en la que se enfrentó este problema en el capítulo 5 fue a través del concepto de *configuración de máquinas* (definición 5.5). En efecto, como estamos considerando el problema redondeado, el hecho de que los trabajos largos tienen que ser procesados en largos de $\mathcal{L}$, implica que no puede haber más de k trabajos largos por máquina. A continuación recordamos este concepto. Cada configuración de máquinas $\kappa \in \mathbb{N}^{\mathcal{C}}$ especifica cuántas máquinas usarán cada configuración $c \in \mathcal{C}$. Como a su vez cada configuración determina cuántos trabajos largos utilizan una máquina, es fácil ver que *si se supone conocida* una configuración de máquinas utilizada por la asignación solución, entonces se sabe cuántos trabajos largos se procesarán. Digamos entonces que la configuración de máquinas define cuántos *casilleros* hay para procesar trabajos largos, y que para cada configuración de máquinas se definirá un PLE como relajación del *problema redondeado*. Si se denota por r a uno de estos casilleros, se considerará x_{jr} como 1 cuando j es procesado en un casillero de largo $\ell_r \in \mathcal{L}$ (0 en caso contrario), y entonces se elimina el problema de considerar el tiempo de proceso en las mismas variables $x_{ji\ell}$.

Respecto de los trabajos cortos, es importante notar que no es posible hacer el mismo razonamiento a partir del problema redondeado, pues la cantidad de largos posibles en los que pueden ser procesados no está determinada a priori, incluso suponiendo conocida alguna configuración de máquinas. La manera en que se enfrentará esto es considerar una máquina auxiliar r^* que procesará exclusivamente trabajos cortos:

1. Sean $J, \{f_j\}_{j\in J}$ una instancia de GCMP sin transferencia, y consideremos el caso de 1 sola máquina y una asignación A óptima. Además, solamente se pueden procesar los trabajos hasta un tiempo $1/k$. Sea α la eficiencia tal que satisface lo siguiente:

   $$d^- f_j(A_j) \geq \alpha \quad \forall j \in J,\ j \text{ activo}$$

   donde $d^- f_j(A_j)$ es la derivada por la izquierda en el instante A_j de la función cóncava f_j. La idea detrás de α es que permitirá encontrar los *tiempos de proceso* de los trabajos que sean procesados en la máquina auxiliar r^*.
2. Para cada $j \in J$, sea t_j la estimación del tiempo de proceso de un trabajo corto en r^*, definida como:

   $$t_j := \min\{1/k, \max\{t \in \mathcal{Q} : d^- f_j(t) \geq \alpha\}\} \tag{6.1}$$

   donde $\mathcal{Q}$ es el conjunto de puntos de quiebre. Así, t_j siempre satisface que $t_j \leq 1/k$, y es lo más grande posible, de modo que j es procesado sin que su pendiente sea inferior a α. Siempre que un trabajo j sea asignado a r^*, i.e. $x_{jr^*} = 1$, j será procesado por un tiempo t_j, y por ende tendrá una ganancia asociada $b_{jr^*} := f_j(t_j)$

Luego, suponiendo conocido α, es posible tener para cada j el tiempo en que se procesaría este trabajo de ser corto en la máquina auxiliar r^*, acotando de esta manera los posibles largos para trabajos cortos. Se supondrá finalmente que dicha máquina r^* tiene una capacidad $T > 0$.

Con lo anterior ya se está en condiciones de escribir la relajación del problema redondeado a través de un PLE. Para esto, se construirá un grafo bipartito completo $G_\kappa = (J \cup (R \cup \{r^*\}), E)$ de la siguiente forma:

1. Un lado de la bipartición será el conjunto de trabajos J.
2. El conjunto R será el conjunto de casilleros para trabajos largos. La configuración de máquinas κ determina, para cada configuración $c \in \mathcal{C}$ definida en 5.4 cuántas máquinas utilizan c. Como además cada configuración c de trabajos indica cuántos casilleros de un determinado largo $\ell \in \mathcal{L}$ hay presentes, de κ se deduce cuántos casilleros en total hay presentes de largo ℓ entre todas las máquinas. Para cada largo ℓ se crean tantos nodos en R como casilleros de dicho largo haya entre todas las máquinas. Por último, considere $r^* \notin R$ un nodo extra, que jugará el rol de una *máquina auxiliar*, con capacidad $T > 0$ para procesar los trabajos cortos.

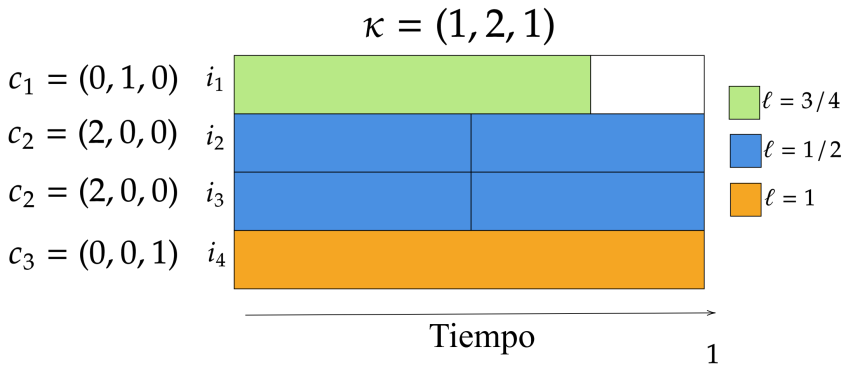


Figura 6.1: Ejemplo de configuración de máquinas para el caso $k = 2$. Dicho k implica que el conjunto de largos $\mathcal{L}$ está dado por $\mathcal{L} = \{1/2, 3/4, 1\}$. Luego la configuración $c_1 = (0, 1, 0)$ significa que se usan cero trabajos de largo 1/2, un trabajo de largo 3/4 y cero trabajos de largo 1. c_2 indica que se utilizan dos trabajos de largo 1/2, y ninguno del resto, y finalmente la configuración c_3 solo utiliza un trabajo de largo 1

Con lo anterior, a continuación se define PLE(κ, α, T) que servirá de *relajación* del problema redondeado.

Definición 6.3 (PLE para el problema redondeado) *Sea I, J, $\{f_j\}_{j \in J}$ una instancia de GCMP sin transferencia. Sea κ una configuración de máquinas dada, una eficiencia $\alpha > 0$ y una capacidad $T > 0$, y sean $G_\kappa = (J \cup (R \cup r^*), E)$ el grafo bipartito completo recién descrito.*

Se define el $PLE(\kappa, \alpha, T)$ para el problema redondeado de GCMP sin transferencia:

$PLE(\kappa, \alpha, T)$

$$\max_{x} \quad \sum_{e \in E} b_e x_e \tag{6.2a}$$

$$\text{s.t.} \quad \sum_{j \in J} x_{jr} = 1 \quad \forall r \in R \tag{6.2b}$$

$$\sum_{r \in R} x_{jr} = 1 \quad \forall j \in J \tag{6.2c}$$

$$\sum_{j \in J} x_{jr^*} t_j \leq T \tag{6.2d}$$

$$x \in \{0,1\}^E \tag{6.2e}$$

donde los $\{b_e\}_{e \in E}$ corresponden a los pesos asociados para cada para arista e de la siguiente forma. Sea $j \in J$ arbitrario:

1. *Para $e = jr$ con $r \neq r^*$, $b_e = f_j(\ell_r)$, es decir, el peso de e estará dado por la ganancia de procesar j por un tiempo de proceso ℓ_r, siendo ℓ_r el largo asociado al casillero r.*
2. *Para $r = r^*$, $b_e = f_j(t_j)$, donde t_j es el valor definido en 6.1 que representa el tiempo de proceso si j es un trabajo corto.*

Finalmente, llamamos $PL(\kappa, \alpha, T)$ a la relajación lineal de *$PLE(\kappa, \alpha, T)$, obtenida al cambiar 6.2e por no negatividad.*

Con lo anterior ya estamos en condiciones de poder exponer los lemas que permitirán demostrar el Teorema 6.1.

6.3. PTAS para GCMP sin Transferencia

Definición 6.4 *Llamamos valor del $PLE(\kappa, \alpha, T)$ a su ganancia asociada a la solución óptima x. Es decir,*

$$val(PLE(\kappa, \alpha, T)) = \sum_{e \in E} b_e x_e$$

Se define $PL(\kappa, \alpha, T)$ de manera análoga pero para la solución óptima del PL.

Una última consideración será que, para cada configuración de máquinas κ, se definirá la cantidad $T(\kappa) > 0$ como el tiempo disponible para procesar trabajos cortos según κ. En el ejemplo de la figura 6.1, $T(\kappa)$ corresponderá al espacio en blanco, de largo 1/4. En adelante, siempre que se omita explícitamente T en $\text{PLE}(\kappa, \alpha)$, se entenderá que la capacidad de la máquina auxiliar r^* es $T(\kappa)$.

Proposición 6.5 *Sea $I, J, \{f_j\}_{j \in J}$ una instancia de GCMP sin transferencia, y sea A^k una asignación óptima k-redondeada para dicha instancia. Sea κ la configuración de máquinas utilizada por A^k, y $T(\kappa)$ el tiempo total utilizado por los trabajos cortos en A^k. Defina $\alpha > 0$*

de la siguiente forma:

$$\alpha := \text{máx}\left\{ d^{-} f_j \left(\sum_{i \in I} A^k_{ij} \right) : j \ \textit{activo} \ \wedge \ \sum_{i \in I} A^k_{ij} \leq 1/k \right\}$$

Entonces

$$val(PL(\kappa, \alpha, T(\kappa))) \geq val(A^k)$$

Demostración. Para demostrar esto, bastará definir una solución x factible para $\text{PL}(\kappa, \alpha, T(\kappa))$, y tal que:

$$\text{val}(x) := \sum_{e \in E} b_e x_e \geq \text{val}(A^k)$$

Considere $\rho : R \to I$ la función que asocia cada nodo $r \in R$ con una máquina de I de la cual proviene (notar que es posible guardar la información que permite definir dicha función ρ en la construcción del conjunto R a partir de la configuración de máquinas κ, según lo revisado en el paso 2 de la definición de G). Mediante el siguiente algoritmo 8 `recuperar` x se define la solución x buscada. Sea J_L el conjunto de trabajos largos, i.e. tales que $\sum_{i \in I} A^k_{ij} \geq 1/k$, y sea $J_C := \{j \in J \setminus J_L : j \text{ es activo}\}$:

Algorithm 8 recuperar x

1: **Inputs:** $J,\ R,\ A^k : I \times J \to \mathbb{Q}_+,\ \rho : R \to I$
2: $x_{rj} \leftarrow 0 \qquad \forall r \in R,\ \forall j \in J$
3: **for** $r \in R$ **do**
4: **for** $j \in J$ **do**
5: **if** $A^k_{\rho(r)j} = \ell_r$ **and** $\sum_{r' \in R,\ r' \neq r} x_{rj} = 0$ **then**
6: $x_{rj} \leftarrow 1$
7: **break** j ▷ pasar al siguiente r
8: **end**
9: **end**
10: **end**
11: $x_{r^* j} \leftarrow \sum_{i \in I} A^k_{ij}/t_j \quad \forall j \in J_C$
12: **Return:** x

A continuación se describe brevemente el pseudo código. El algoritmo 8 `recuperar` x realiza lo siguiente: Entre los pasos 3 – 10, se recorren todas las combinaciones entre casilleros y trabajos, de modo que, en el paso 5 se realizan las siguientes validaciones para saber cómo proceder:

1. $A^k_{\rho(r)j} = \ell_r$: Esta condición verifica si j es procesado en la máquina asociada al casillero r, $\rho(r)$, en un tiempo de proceso ℓ_r asociado a dicho casillero. Esto permitirá que x, el resultado del algoritmo 8, satisfaga que procesa los mismos trabajos largos de A^k en el mismo tiempo de proceso.
2. $\sum_{r' \in R,\ r' \neq r} x_{rj} = 0$: Aquí se verifica si el trabajo j ya está asignado a otro casillero r. Con esto se satisface la restricción 6.2c.

Dado lo anterior, de satisfacerse ambas condiciones del paso 5, entonces en el paso 6 se asigna $x_{rj} \leftarrow 1$, de modo que j sea procesado en un tiempo $A^k_{\rho(r)j}$. Como j ya fue asignado a un

casillero, el paso 7 permite que x no asocie más de un j, de modo que se pase a asignar el siguiente r. En el paso 11 se realiza la asignación de trabajos cortos, de modo que los trabajos cortos de J_C se les asigna un tiempo de proceso tal que se satisfaga la restricción 6.2d.

A continuación se verificará las dos cosas restantes: Que el valor de la solución x es al menos el de la asignación A^k, y que además dicho x es solución factible para $\text{PLE}(\kappa, \alpha, T(\kappa))$.

1. Sea $j' \in J_L$. Por definición de R y $\{\ell_r\}_{r\in R}$, necesariamente existe un vértice $r \in R$ tal que $\ell_r = A^k_{\rho(r)j'}$. Luego, sea r^o el primer vértice de R procesado por el algoritmo 8 tal que $x_{r^o j'} = 1$. Entonces:

$$f_{j'}\left(\sum_{i\in I} A^k_{ij}\right) = f_{j'}(A^k_{\rho(r^o)j'}) = f_{j'}(\ell_{r^o}) = b_{r^o j'} = \sum_{r\in R} b_{rj'}x_{rj'}$$

   Donde la primera igualdad viene del hecho que A^k es una asignación (redondeada) factible y luego cada trabajo j en A^k es asignado a una sola máquina $\rho(r^o)$. Luego x obtiene la misma ganancia para los trabajos largos.
2. Sea $j' \in J_C$. Sea $i' \in I$ la máquina donde j' es procesado. Por la definición de α y $t_{j'}$ se tiene que $\sum_{i\in I} A^k_{ij'} = A^k_{i'j'} \le t_{j'}$, y entonces $f_{j'}(A^k_{i'j'}) \le f_{j'}(t_{j'}) = b_{r^* j'}$. Como el algoritmo 8 asigna todos los trabajos cortos de J_C a r^*, entonces:

$$\sum_{j\in J_C} f_j\left(\sum_{i\in I} A^k_{ij}\right) \le \sum_{j\in J_C} b_{r^* j}x_{r^* j}$$

Luego, los puntos 1 y 2 implican que x efectivamente obtiene al menos la ganancia de A^k:

$$\begin{aligned}
\text{val}(x) &= \sum_{r\in R}\sum_{j\in J_L} b_{rj}x_{rj} + \sum_{j\in J_C} b_{r^* j}x_{r^* j} \\
&\ge \sum_{j\in J_L}\sum_{r\in R} b_{rj}x_{rj} + \sum_{j\in J_C} f_j\left(\sum_{i\in I} A^k_{ij}\right) \\
&= \sum_{j\in J_L} f_j\left(\sum_{i\in I} A^k_{ij}\right) + \sum_{j\in J_C} f_j\left(\sum_{i\in I} A^k_{ij}\right) \\
&= \sum_{j\in J} f_j\left(\sum_{i\in I} A^k_{ij}\right)
\end{aligned}$$

y entonces:

$$\text{val}(x) \ge \sum_{j\in J} f_j\left(\sum_{i\in I} A^k_{ij}\right)$$

Veamos ahora que x es factible para $\text{PL}(\kappa, \alpha, T(\kappa))$. En efecto, para la restricción 6.2b los pasos $3-7$ aseguran que, si j es procesado en un tiempo ℓ_r en la máquina $\rho(r)$, se ejecuta un `break` en el paso 7 tal que no se asignan más trabajos al vértice r. Para la restricción 6.2c, notar que en el `If` del paso 4 se verifica si j ya fue asignado en otro casillero de R, mediante la condición $\sum_{r'\in R,\ r'\neq r} x_{rj} = 0$. Con ello se asegura que cada trabajo es asignado a lo más a un vértice de R, satisfaciéndose entonces 6.2c. Para la restricción relajada de 6.2e (recordar que se está trabajando con el PL), i.e. $x_e \ge 0 \quad \forall e \in E$, basta notar que todos los valores involucrados en la definición de x a lo largo del algoritmo 8 son no negativos. La restricción

6.2d se satisface, pues de la definición de α dada en el enunciado y de t_j en la definición 6.1, se tiene que:

$$t_j \leq \sum_{i \in I} A^k_{ij}$$

Luego:

$$\sum_{j \in J} t_j x_{r^* j} \leq \sum_{j \in J_C} \sum_{i \in I} A^k_{ij} = T(\kappa)$$

Y entonces x es factible para $\mathrm{PL}(\kappa, \alpha, T(\kappa))$, lo que concluye la demostración. □

La proposición 6.5 permite ver que la relajación lineal de 6.3 entrega una cota superior para el problema redondeado. Dado lo anterior, se mostrará que, a partir de una solución de $\mathrm{PL}(\kappa, \alpha, T(\kappa))$ se puede encontrar una solución factible de $\mathrm{PLE}(\kappa, \alpha, T(\kappa) + 1/k)$ de un valor no tan distinto al de $\mathrm{PL}(\kappa, \alpha, T(\kappa))$. Para lo anterior, será útil el siguiente lema:

Lema 6.6 *Sea x^* solución de $PL(\alpha, \kappa, T)$ que es punto extremo. Si x^* es fraccional, y se denota por F al soporte fraccional de x^*,i.e.*

$$F := \{ij \in E : 0 < x^*_{rj} < 1\}$$

entonces F es un ciclo que pasa por r^*.

Demostración. En efecto, sea P el poliedro factible de $\mathrm{PL}(\alpha, \kappa, T)$. Luego,

$$P = \{x \in \mathbb{R}^E : A_I x = \mathbf{1},\ \sum_{j \in J} x_{r^* j} t_j \leq T\ , x \geq 0\}$$

donde A_I es la matriz de incidencia de un grafo bipartito, correspondiente a las restricciones (6.2b) y (6.2c) de $\mathrm{PL}(\alpha, \kappa, T)$. Si x^* satisface de manera estricta la restricción (6.2d), esto es, si

$$\sum_{j \in J} x^*_{r^* j} t_j < T$$

se deduce de la definición de P que x^* es un punto extremo de un poliedro definido a partir de la matriz de incidencia A_I de un grafo bipartito, i.e. es punto extremo de un poliedro integral, y entonces x^* debe ser integral. Luego, como se está suponiendo x^* fraccional, necesariamente la desigualdad anterior es igualdad, i.e.

$$\sum_{j \in J} x^*_{r^* j} t_j = T$$

Considere además el siguiente grafo bipartito H definido a partir del soporte de fraccional de x^*:

$$\begin{aligned} J' &:= V(F) \cap J \\ R' &:= V(F) \cap R \\ H &:= (J' \cup R' \cup \{r^*\}, F) \\ \beta &:= \sum_{j \in J : j r^* \in F} t_j x^*_{r^* j} \end{aligned}$$

A continuación se demuestra que existe un ciclo $C \subseteq F$. En efecto, sea $j \in J'$, por definición de J' necesariamente existe $r' \in R'$ tal que $x^*_{r'j} \in (0,1)$. Como x^* satisface la restricción (6.2c), tenemos $x^*(\delta(j)) = 1$ y luego necesariamente existe $r'' \in R', r'' \neq r'$ tal que $x^*_{r''j} \in (0,1)$, y entonces $\deg(j) \geq 2$. El razonamiento es análogo para el caso en que $r \in R'$, pues también ocurre que x^* debe satisfacer la restricción 6.2b y luego $x^*(\delta(r)) = 1$. Se tiene entonces que para cada vértice $v \in J' \cup R'$ $\deg_F(r^*) \geq 2$. Si H fuese un grafo acíclico, necesariamente debiese tener al menos dos hojas, pero por lo recién demostrado el único vértice que podría ser una hoja es r^*. Entonces, H contiene al menos un ciclo $C \subseteq F$.

Por otro lado, dicho ciclo C debe pasar por el vértice r^*. En efecto, suponga por contradicción que no pasa por r^*, y defina los siguientes puntos de $\mathbb{R}^E$: Sea $e \in E$ arbitrario y denote $C = e_1 e_2 \cdots e_{|C|}$:

$$x^1_e := \begin{cases} x^*_e & e \in E \setminus C \\ x^*_e + \varepsilon & \text{si } e = e_i \ i \text{ par} \\ x^*_e - \varepsilon & \text{si } e = e_i \ i \text{ impar} \end{cases}, \quad x^2_e := \begin{cases} x^*_e & e \in E \setminus C \\ x^*_e - \varepsilon & \text{si } e = e_i \ i \text{ par} \\ x^*_e + \varepsilon & \text{si } e = e_i \ i \text{ impar} \end{cases} \tag{6.3}$$

donde $\varepsilon > 0$ es tal que $0 < x^*_e - \varepsilon$ y $x^*_e + \varepsilon < 1$ para todo $e \in C$. Se puede ver que ambos puntos x^1, x^2 están en P. En efecto, defina $C_+ := \{e \in C : e = e_i \text{ con } i \text{ par}\}$, y $C_- := C \setminus C_+$. Entonces para $r \in R$:

$$\begin{aligned} \sum_{j \in J} x^1_{rj} &= \sum_{e \in \delta(r)} x^1_e \\ &= \sum_{e \in \delta(r) \cap C_+} x^1_e + \sum_{e \in \delta(r) \cap C_-} x^1_e + \sum_{e \in \delta(r) \setminus C} x^1_e \\ &= \sum_{e \in \delta(r) \cap C_+} (x^*_e + \varepsilon) + \sum_{e \in \delta(r) \cap C_-} (x^*_e - \varepsilon) + \sum_{e \in \delta(r) \setminus C} x^*_e \\ &= \varepsilon(|\delta(r) \cap C_+| - |\delta(r) \cap C_-|) + \sum_{e \in \delta(r)} x^*_e \end{aligned}$$

Como G es un grafo bipartito, $|\delta(r) \cap C_+| = |\delta(r) \cap C_-|$, pues el largo de C es par, luego:

$$\sum_{j \in J} x^1_{rj} = \sum_{e \in \delta(r)} x^* = 1$$

Luego x^1 satisface (6.2b). La demostración para 6.2c es análoga. Entonces, si $r^* \notin C$, entonces $\sum_{j \in J} x^1_{r^* j} t_j = \sum_{j \in J} x^*_{r^* j} \leq T$, i.e. x^1 satisface 6.2d. Lo anterior es análogo para el punto x^2. Luego, ambos puntos $x^1, x^2 \in P$. Como x^* es el promedio de x^1 y x^2, se contradice que x^* sea punto extremo de P. Luego, de existir un ciclo $C \subseteq S$, necesariamente este tiene que pasar por r^*.

A continuación, se demostrará que dicho ciclo C es exactamente F. Ya se sabe que $C \subseteq F$, falta probar que $F \subseteq C$. Para esto, considere el siguiente poliedro restringido P':

$$P' := \{x \in \mathbb{R}^F : x(\delta_F(j')) = 1 \ \forall j' \in J'; \ x(\delta_F(r')) = 1 \ \forall r' \in R'; \sum_{j' \in J' : r^* j' \in F} t_{j'} x_{r^* j'} = \beta; \ 0 \leq x \leq 1\}$$

y llame $x' = x^*_F \in \mathbb{R}^F$, $x'' = x^*_{E \setminus F} \in \mathbb{R}^{E \setminus F}$ a los vectores obtenidos al restringir x^* a las aristas de F y $E \setminus F$, respectivamente, de modo que $x^* = (x', x'')$. Se demostrará que $x' \in P'$. Para ello, sea S el soporte de x^* y note que para todo $j' \in J'$, $S \cap \delta(j') \subset F$. En efecto, por definición

de J', existe al menos un r' tal que $r'j' \in F$ y $x^*_{r'j'} > 0$. Si ocurriese que $S \cap \delta(j') \notin F$, entonces existe r'' tal que $x^*_{r''j'} = 1$, lo cual es una contradicción para la factibilidad de x^* respecto de la restricción (6.2b) pues $\sum_{r \in R} x^*_{rj'} \geq x^*_{r'j'} + x^*_{r''j'} > 1$. Luego se cumple que $S \cap \delta(j') \subset F$ para todo $j' \in J'$. La inclusión anterior implica que $\delta_S(j') = \delta_F(j')$ para todo $j' \in J'$. Así:

$$1 = \sum_{r \in R} x^*_{rj'} = x^*(\delta_S(j')) = x^*(\delta_F(j')) = x'(\delta_F(j'))$$

El razonamiento anterior es análogo para $r' \in R'$, y entonces $x'(\delta_F(r')) = 1$. La desigualdad que involucra a β viene de la misma definición de β. Luego $x' \in P'$. Más aún, si se toma $y' \in P'$, entonces $x^y = (y', x'') \in P$. En efecto, se verá por casos.

1. Si $j \notin J'$, entonces $\delta(j) = \delta_{E \setminus F}(j)$ (por definición de J'). Luego:
$$x^y(\delta(j)) = x^y(\delta_{E \setminus F}(j)) = \sum_{\substack{e \in \delta(j) \\ e \in E \setminus F}} x^y_e = \sum_{\substack{e \in \delta(j) \\ e \in E \setminus F}} x^*_e = x^*(\delta_{E \setminus F}(j)) = x^*(\delta(j)) = 1$$

2. Si $j \in J'$, se tiene que $\delta_{E \setminus F}(j) \cap S = \varnothing$. En efecto, como $j \in J'$, entonces existe $e \in F$ tal que $x^*_e \in (0, 1)$. Por otro lado, si existiese e' tal que $e' \in \delta_{E \setminus F}(j) \cap S$, como en particular e' está en S, entonces $x^*_{e'} > 0$, pero como $\delta_{E \setminus F}(j) \subset E \setminus F$, entonces $x^*_{e'} = 1$, lo cual es una contradicción, pues $e, e' \in \delta(j)$ y tendríamos que $x^*(\delta(j)) > 1$. Así, cuando $j \in J'$, $\delta_{E \setminus F}(j) \cap S = \varnothing$, luego $x^*(\delta_{E \setminus F}(j)) = 0$. Entonces:
$$x^y(\delta(j)) = x^y(\delta_F(j)) + x^y(\delta_{E \setminus F}(j)) = y'(\delta_F(j)) + x^*(\delta_{E \setminus F}(j)) = y'(\delta_F(j)) = 1$$

Con esto, para todo $j \in J$, necesariamente x^y satisface (6.2b). De manera análoga se demuestra que x^y satisface (6.2c). Para ver que satisface (6.2d) basta notar que:

$$\sum_{j \in J} x^y_{r^*j} t_j = \sum_{r^*j \in \delta_F(r^*)} x^y_{r^*j} t_j + \sum_{r^*j \in \delta_{E \setminus F}(r^*)} x^y_{r^*j} t_j = \beta + \sum_{r^*j \in \delta_{E \setminus F}(r^*)} x^*_{r^*j} t_j = \sum_{j \in J} x^*_{r^*j} t_j \leq T$$

Luego, se tiene que para todo $y' \in P'$, $(y', x'') \in P$. Por otro lado, $x' \in P'$ es punto extremo, pues en caso contrario, significa que existen $x^1, x^2 \in P'$ tales que $x' = x^1/2 + x^2/2$, y por lo demostrado en el párrafo anterior existen dos puntos $(x^1, x''), (x^2, x'') \in P$ tales que $x^* = \frac{1}{2}(x^1, x'') + \frac{1}{2}(x^2, x'')$, contradiciendo que x^* es punto extremo de P. Ahora llame $\mathcal{O}$ al conjunto de restricciones activas de x'. Como $0 < x' < 1$, entonces de la definición de P' se tiene que $|\mathcal{O}| \leq |J'| + |R'| + 1$. Por otro lado, como x' tiene $|F|$ variables y es punto extremo, entonces $\mathcal{O}$ contiene al menos $|F|$ restricciones linealmente independientes, se tiene que $|F| \leq |\mathcal{O}|$. Luego:

$$|F| \leq |J \cap R'\{r^*\}| = |J'| + |R'| + 1$$

Luego,

$$|J'| + |R'| + 1 \geq |F| = \frac{1}{2} \sum_{v \in J' \cup R' \cup \{r^*\}} \deg_F(v) \geq \frac{1}{2} \sum_{v \in J' \cup R' \cup \{r^*\}} 2 = |J'| + |R'| + 1$$

donde la igualdad viene de doble conteo de las aristas de |F| dado que la suma es sobre todos los vértices de H, y la segunda desigualdad del hecho que $\deg_F(v) \geq 2$ para todo $v \in J' \cup R' \cup \{r^*\}$. Como el comienzo y el final de la secuencia de desigualdades son la misma cantidad, entonces todas son igualdades, en particular se tiene $\deg_F(v) = 2$ para $v \in J' \cup R' \cup \{r^*\}$. Como el grafo H es 2-regular, se deduce que $|F|$ es unión vértice-disjunta de ciclos. Como antes ya se demostró que de existir un ciclo $C \subseteq F$ este debía pasar por r^*, y $F \neq \varnothing$, entonces F es el conjunto de aristas de un único ciclo. □

El Lema 6.6 permitirá escribir un algoritmo para encontrar una solución x^M que es *casi* factible para GCMP sin transferencia (esto se precisará en el lema 6.7) pero que aproxima a la mejor solución redondeada. La idea central son los siguientes pasos:

1. Sean $\mathrm{A} := \{\alpha > 0 : \exists g_j \text{ subtrabajo}, \ j \in J, \mathrm{eff}(g_j) = \alpha\}$ el *conjunto de eficiencias* y $\mathrm{K} \subset \mathbb{N}^{\mathcal{C}}$ el conjunto de todas las *configuraciones de máquinas* factibles
2. De manera iterativa, fijar $\alpha \in \mathrm{A}$, $\kappa \in \mathrm{K}$
3. Para cada uno de los elementos anteriores, definir $\mathrm{PL}(\alpha, \kappa, T(\kappa))$ y el grafo G asociado. Por último definir $M = 0$ y $x^M = 0 \in \mathbb{R}^E$. M será la ganancia del mejor vector x^M considerado hasta el momento, y x^M la solución del $\mathrm{PL}(\alpha, \kappa, T(\kappa))$ asociado.
4. Resolver $\mathrm{PL}(\alpha, \kappa, T(\kappa))$. Sea x^* una solución punto extremo.
5. A continuación se procede según si x^* es entera o no:
   (a) x^* es una solución entera, luego:
      i. Si $\mathrm{val}(x^*) > M$, entonces:

$$x^M \leftarrow x^*$$
$$M \leftarrow \mathrm{val}(x_M)$$

      y se pasa a la siguiente iteración desde el paso (a).
      ii. En caso contrario, continuar a la siguiente iteración desde el paso (a).
   (b) Si x^* es fraccional, gracias al Lema 6.6 se sabe que el soporte fraccional dado por F consiste en un ciclo en G' tal que pasa por r^*. Como F es un ciclo en un grafo bipartito, entonces es unión de dos matchings arista-disjuntos, M_1 y M_2:

$$F = M_1 \uplus M_2$$

   luego, sin perdida de generalidad, suponga que el peso asociado a las asignaciones de M_1 es mayor que el de M_2, es decir:

$$\sum_{e \in M_1} b_e \geq \sum_{e \in M_2} b_e$$

   Si los trabajos de J' fuesen asignados a los casilleros de R según las aristas de M_1, entonces la ganancia aportada por dichos trabajos sería mayor que si se hiciera según las aristas de M_2. Considerando lo anterior, sea x' definido a partir de x^* de la siguiente forma:

$$x'_e := \begin{cases} x^*_e & \text{si } e \in E \setminus F \\ 1 & \text{si } e \in M_1 \\ 0 & \sim \end{cases}, \qquad M' := \mathrm{val}(x^M)$$

   Así, x' tiene todas sus entradas enteras, y entonces el algoritmo repite el paso 5 (a) usando x'.
6. Una vez terminadas las iteraciones, retornar x^M

A continuación se demuestra que el procedimiento anterior permite construir un algoritmo que encuentra una *aproximación* al problema redondeado.

Lema 6.7 *Sea $J,\ I,\ \{f_j\}_{j\in J}$ una instancia de GCMP sin transferencia y $k \in \mathbb{N}$. Se puede construir un algoritmo que, en tiempo polinomial en n y m, entrega x^M tal que, si A^k es una solución óptima asociada al problema redondeado:*

1. *Existen $\alpha \in \mathrm{A}$ y $\kappa \in \mathrm{K}$ tales que x^M es punto entero factible de $PLE(\alpha, \kappa, T(\kappa) + 1/k)$.*
2. *$val(x^M) \geq val(A^\kappa)$*
3. *Existe un polinomio $p : \mathbb{R}^3 \to \mathbb{R}$ tal que la complejidad del algoritmo es $\mathcal{O}(p(n^2, m, k) \cdot m^{k^{k^2}})$*

Demostración. La idea del siguiente algoritmo es poner en pseudo código los pasos 1−6 recién vistos. Sea A el conjunto de eficiencias asociado a las funciones de $\{f_j\}$ y K el conjunto de configuraciones de máquinas factibles:

Algorithm 9 obtener mejor solución PLs - x^M

Inputs: $J,\ I,\ \{f_j\}_{j\in J},\ \mathrm{A},\ \mathrm{K}$
$x^M \leftarrow 0 \in \mathbb{R}^E$
$M \leftarrow 0 \in \mathbb{R}$
$G^M \leftarrow (\varnothing, \varnothing)$
$\kappa^M \leftarrow 0 \in \mathbb{N}^{\mathcal{C}}$
for $\alpha \in \mathrm{A},\ \kappa \in \mathrm{K}$ **do**
    Definir $\mathrm{PL}(\alpha, \kappa, T(\kappa))$ y $G := (J \cup (R \cup \{r^*\}),\ E)$ bipartito asociado a dicho PL
    Resolver $\mathrm{PL}(\alpha, \kappa, T(\kappa))$. Sea x^* solución extrema
    if x^* es entero **then**
        if $\mathrm{val}(x^*) > M$ **then**
            $x^M \leftarrow x^*$
            $M \leftarrow \mathrm{val}(x^M)$
            $G^M \leftarrow G$
            $\kappa^M \leftarrow \kappa$
        else
            Continuar a la siguiente iteración de α y κ
        end
    else
        $F \leftarrow \{e \in E : 0 < x^*_e < 1\}$
        $M_1, M_2 \leftarrow F = M_1 \cup M_2$, con M_1 y M_2 artista-disjuntos t.q. $\sum_{e\in M_1} b_e \geq \sum_{e\in M_2} b_e$
        $x'_e \leftarrow \begin{cases} x^*_e & \text{si } e \in E \setminus F \\ 1 & \text{si } e \in M_1 \\ 0 & \text{si } \sim \end{cases}$
        volver al paso 9 usando $x^* \leftarrow x'$
    end
end
Return: $(x^M,\ G^M,\ \kappa^M)$

Lo que hace el algoritmo 9 es, primero, inicializar x^M en $0 \in \mathbb{R}^E$; su ganancia asociada M en $0 \in \mathbb{R}$; el grafo G^M del cual proviene x^M como el grafo vacío y finalmente la configuración de máquinas κ^M de la cual proviene x^M. Luego, para cada eficiencia $\alpha \in \mathrm{A}$ y configuración de

máquinas $\kappa \in \mathrm{K}$, se resuelve la *relajación lineal* del PLE asociado al problema redondeado, i.e. $\mathrm{PL}(\alpha, \kappa, T(\kappa))$ (ver pasos 7 y 8), denotando por x^* dicha solución. En los pasos siguientes se actualiza x^M siempre que:

1. x^* sea una solución entera (`if` del paso 9)
2. $\mathrm{val}(x^*) > M$.

Si ambas condiciones se satisfacen, se ejecutan los pasos 9−14 donde se hacen las actualizaciones correspondientes de M, G^M y κ^M. En caso de que x^* no sea entero, se pasa directamente a los pasos 19 − 22, donde se define x' a partir de x^M de modo que x' sea entera. Gracias al lema 6.6, de ser x^* una solución fraccional, el soporte fraccional F de x^* es un ciclo en un grafo bipartito, entonces se puede escribir $F = M_1 \cup M_2$, i.e. como unión de dos matchings arista-disjuntos, con M_1 tal que:

$$\sum_{e \in M_1} b_e \geq \sum_{e \in M_2} b_e$$

Así, en el paso 21 se obtiene x' reemplazando las entradas e de x^* que están en M_2 por 0, y por 1 las que se encuentran en M_1. Lo anterior, más la concavidad de los $\{f_j\}$, implican que:

$$\mathrm{val}(x') \geq \mathrm{val}(x^*)$$

Finalmente, el algoritmo vuelve al paso 9 tomando $x^* \leftarrow x'$, para actualizar el valor x^M, G^M y κ^M en caso de que $\mathrm{val}(x^*) \geq \mathrm{val}(x^M)$.

A continuación veremos la complejidad del algoritmo 9. Primero, gran parte de dicho algoritmo consiste en ejecutar iteraciones para $\alpha \in \mathrm{A}$, $\kappa \in \mathrm{K}$. Suponemos en general que las funciones de ganancia tienen una cantidad de eficiencias acotada por $\mathcal{O}(n)$, y entonces el cardinal de A está acotado por $\mathcal{O}(n^2)$. Recordar que las configuraciones de K máquinas son arreglos indexados en $\mathcal{C}$, el conjunto de configuraciones de la definición 5.4, y que toma valores en $\{0, 1, \ldots, m\}$. El conjunto $\mathcal{C}$ consiste en arreglos indexados en $\mathcal{L}$, que tiene cardinal $\mathcal{O}(k^2)$, y donde cada entrada toma valores en $0, \ldots, k$. Luego $|\mathcal{C}|$ es $\mathcal{O}(k^{k^2})$. Como el conjunto sobre el que están indexadas las configuraciones de máquinas es $\mathcal{O}(k^{k^2})$, entonces K es $\mathcal{O}(m^{k^{k^2}})$. En síntesis, el `for` del paso 6 ejecuta $\mathcal{O}(n^2 m^{k^{k^2}})$ iteraciones. Dentro del `for` la complejidad está dominada por la resolución de $\mathrm{PL}(\kappa, \alpha, T(\kappa))$. A partir del Teorema 2.1.6 de [CM11], se sabe que existe un algoritmo que encuentra un punto extremo óptimo para un programa lineal dado, y dicho algoritmo es polinomial en el número de nodos de trabajos y casilleros. El número de trabajos es n, mientras que el número de casilleros está acotado superiormente por mk, lo cual viene del hecho que todo casillero es de largo al menos $1/k$, y entonces no puede haber más de k casilleros por máquina, y como existen m máquinas, se deduce la cota superior. Así, el tamaño de $\mathrm{PL}(\kappa, \alpha, T(\kappa))$ está determinado por las variables n, m y k, y entonces se tiene que existe un polinomio $q(n, m, k) : \mathbb{R}^3 \to \mathbb{R}$ tal que la complejidad de resolver $\mathrm{PL}(\kappa, \alpha, T(\kappa))$ es $\mathcal{O}(q(n, m, k))$. Tomando el p polinomio tal que $p(n, m, k) = q(n, m, k) \cdot n^2$, se deduce que la complejidad que la complejidad del algoritmo 9 es $\mathcal{O}(p(n, m, k) \cdot m^{k^{k^2}})$. Así, para cada k *fijo*, el algoritmo 9 efectivamente es polinomial en n y m.

Por otro lado, x^M es punto factible de $\mathrm{PLE}(\alpha, \kappa, T(\kappa) + 1/k)$: Sea una iteración cualquiera y α, κ asociados a ella

- Si x^* es solución entera y $\mathrm{val}(x^*) > M$, entonces $x^M \leftarrow x^*$ en paso 11. Como x^* es

solución entera de $\mathrm{PL}(\alpha, \kappa, T(\kappa))$, entonces es punto factible de $\mathrm{PLE}(\alpha, \kappa, T(\kappa))$, y en particular del $\mathrm{PLE}(\alpha, \kappa, T(\kappa) + 1/k)$

- si x^* no es entera, entonces $x^* \leftarrow x'$. Si $\mathrm{val}(x^*) = \mathrm{val}(x') > M$, entonces $x^M \leftarrow x'$. De la definición de x' en el paso 21 se tiene que ella es solución factible de $\mathrm{PLE}(\alpha, \kappa, T(\kappa) + 1/k)$. En efecto, lo anterior viene del hecho que x' viene de x^* factible para $\mathrm{PL}(\alpha, \kappa, T(\kappa))$, en el paso 21 solo se modifican los valores de x^* a lo largo de un ciclo de modo que se siguen satisfaciendo las restricciones (6.2b) y (6.2c). Para la restricción (6.2d) es importante notar que al pasar a 1 $e = j'r^* \in M_1 \cap \delta_F(r^*)$, entonces la suma $\sum_{j \in J} x'_{jr^*} t_j$ cambia el valor de x^*_e a 1 $= x'_e$, y entonces la suma aumenta en a lo más $t_{j'} \leq 1/k$. Así, x^M es solución entera de $\mathrm{PLE}(\alpha, \kappa, T(\kappa) + 1/k)$.

Dado lo anterior, x^M siempre es solución de algún $\mathrm{PLE}(\alpha, \kappa, T(\kappa) + 1/k)$, donde α y κ son los asociados a las iteraciones, lo cual permite concluir el punto 1 del enunciado. Respecto de 2, sea $\kappa' \in \mathrm{K}$ la configuración de máquinas asociada A^k. Gracias a la proposición 6.5, tomando α' de la definición (6.3) se tiene que el x^* solución del paso 8 satisface:

$$\mathrm{val}(x^*) \geq \mathrm{val}(A^k)$$

y como x^M es el punto de mayor valor entre las iteraciones, entonces:

$$\mathrm{val}(x^M) \geq \mathrm{val}(A^k)$$

Lo cual demuestra el punto 2 del enunciado, y entonces concluye la demostración. □

A partir del resultado del algoritmo 9, se puede construir una asignación A^M que aproxime el valor de la solución óptima del problema redondeado A^k.

Lema 6.8 *Continuando con el enunciado del lema 6.7, se puede construir un algoritmo que a partir de x^M, obtiene una asignación, denotada por A^M, y tal que:*

$$val(A^M) \geq val(A^k)$$

Además, A^M es una asignación tal que:

1. *No transfiere trabajos entre máquinas*
2. *El tiempo de proceso utilizado por máquina está acotado por $1 + 1/k$*
3. *El algoritmo tiene complejidad $\mathcal{O}(n \cdot mk)$*

Demostración. Sea x^M el resultado del algoritmo 9. A continuación se utilizarán resultados de secciones/capítulos anteriores para obtener una asignación $A^M : I \times J \to \mathbb{Q}$ a partir de x^M, y tal que:

$$\mathrm{val}(A^M) = \mathrm{val}(x^M) \geq \mathrm{val}(A^\kappa)$$

En efecto, en el capítulo 5 vimos que mediante el algoritmo 6 es posible recuperar una asignación, que se denotará por A^L, que asigna exclusivamente a los trabajos largos a partir de la solución al problema de peso máximo utilizado para resolver el caso de GCMP sin transferencia - solo trabajos largos. Sea x^L definido a partir de x^M de la siguiente forma:

$$x^L_e := \begin{cases} x^M_e & \text{si } e = jr \text{ con } r \in R,\ j \in J \\ 0 & \text{si } e = jr^*,\ j \in J \end{cases}$$

Es decir, x^L toma los valores de x^M asociados a los casilleros de R para los trabajos largos. Aplicando el algoritmo 6 `asignación trabajos largos` al vector x^L se logra obtener la asignación $A^L : I \times J \to \mathbb{Q}$ tal que:

$$\text{val}(x^L) = \text{val}(A^L)$$

Por otro lado, basado en el algoritmo 1 `round-robin`, este se modificará ligeramente en un nuevo algoritmo 10 `round-robin con aumento` para asignar los trabajos cortos en las máquinas de I. Sea $J^C = \{j \in J : x^M_{r^*j} = 1\}$. Consideremos la asignación $A^* : J^C \to \mathbb{Q}_+$ siguiente:

$$A^*_j := x^M_{r^*j} t_j$$

Es decir, $A^* : J^C \to \mathbb{Q}_+$ es la asignación que procesa los trabajos j asignados a la máquina ficticia r^* un tiempo t_j. El algoritmo 10 `round-robin con aumento` es esencialmente el algoritmo 1 `ronund-robin`, pero con dos consideraciones distintas del algoritmo 1 `round-robin`:

1. Considera el caso en que las máquinas tienen capacidades distintas, dadas por el conjunto $\{T_i\}_{i=1}^m$
2. Para toda máquina $i \in I$, si el último trabajo asignado supera la capacidad T_i, este no se corta (de ahí el *aumento de recurso*)

Sin pérdida de generalidad, supongamos que los trabajos de J^C están numerados de 1 hasta n, y que las máquinas en I están numeradas de 1 hasta m. Con esto podemos definir el algoritmo 10 `round-robin con aumento`:

Algorithm 10 `round-robin con aumento`

1: **Inputs:** $\{T_i\}_{i=1}^m, A^* : J^C \to \mathbb{Q}_+$
2: **for** $i \in I$ **do**
3: **for** $j \in J^C \cup \{0\}$ **do**
4: $A^c_{ij} \leftarrow 0$
5: **while** $j \leq n$ **do**
6: $A^c_{ij} \leftarrow A^*_j$
7: **if** $T_i - \sum_{z=1}^{j} A^c_{iz-1} < A^*_j \ \wedge \ i \leq m-1$ **then**
8: $i \leftarrow i+1$
9: $j \leftarrow j+1$
10: **Return** A^c

El algoritmo 10 `round-robin con aumento` termina, pues existen n trabajos. Veamos ahora que el algoritmo 10 `round-robin con aumento` procesa todos los trabajos activos de A^* en el mismo tiempo utilizado por dicha asignación, es decir:

$$\sum_{i \epsilon I} A^c_{ij} = A^*_j \qquad \forall j \in J^C$$

En efecto, ya se sabe que el algoritmo 10 asigna todos los trabajos de J^C. Por otro lado, los trabajos son asignados a una sola máquina, pues en cada iteración del `while` se asigna un trabajo en el paso 6, y al final de cada iteración se pasa al siguiente trabajo en el paso 9.

Luego, dado $j \in J^C$, se tiene que existe un único i' con $\sum_{i \in I} A^c_{ij} = A^c_{i'j}$. Como solo se asigna una única vez tiempo de proceso a j durante la ejecución del algoritmo 10 en el paso 6, necesariamente:

$$A^c_{i'j} = A^*_j$$

Y entonces se concluye $\sum_{i \in I} A^c_{ij} = A^*_j$ para todo $j \in J^C$. Por otro lado, A^c satisface que para toda máquina $i \in I$, los trabajos activos según A^c no superan la capacidad de i más allá de un tiempo $1/k$, es decir:

$$\sum_{j \in J} A^c_{ij} \leq T_i + 1/k \qquad \forall i \in I$$

Sea entonces $i \leq m-1$. En el `while` del paso 9 se tiene que solo se pasa a la siguiente máquina una vez que se satisface $T_i - \sum_{z=1}^{j} A^c_{iz-1} < A^*_j$, es decir una vez que el último trabajo asignado, digamos j_i, satisface que su tiempo de proceso en A^* es superior a la capacidad disponible en esta iteración, dado por la cantidad $T_i - \sum_{z=1}^{j_i} A^c_{iz-1}$. Entonces se tiene que:

$$\sum_{z=1}^{j_i - 1} A^c_{iz} \leq T_i$$

Como $A^*_{j_i} \leq 1/k$, entonces:

$$\sum_{z=1}^{j_i} A^c_{iz} \leq T_i + 1/k$$

Dado que j_i es el último trabajo asignado a i antes de pasar a la siguiente máquina $i+1$ y para los trabajos $j \geq j_i$ se inicializaron con tiempo de proceso $A^c_{ij} = 0$ en los pasos 2 a 4, entonces:

$$\sum_{j \in J} A^c_{ij} \leq T_i + 1/k$$

Para el caso $i = m$, supongamos por contradicción que:

$$\sum_{j \in J} A^c_{mj} > T_m + 1/k$$

Como se llegó a asignar trabajos en la máquina m, necesariamente para cada $i \leq m-1$ se tiene que:

$$\sum_{j \in J} A^c_{ij} > T_i$$

Luego, sumando sobre i:

$$\sum_{i,j} A^c_{ij} = \sum_{j \in J} A^*_j = \sum_{j \in J} x^M_{jr^*} t_j > T(\kappa) + 1/k$$

lo cual contradice el hecho de que x^M sea solución factible de $\text{PLE}(\alpha, \kappa, T(\kappa) + 1/k)$. Luego, para toda máquina $i \in I$.

$$\sum_{j \in J} A^c_{ij} \leq T_i + 1/k$$

Y entonces el algoritmo 10 `round-robin con aumento` permite obtener una asignación A^M que satisface los puntos 1 y 2 del enunciado

Se define entonces $A^M : I \times J \to \mathbb{Q}$ como la siguiente asignación:

$$A^M_{ij} := A^L_{ij} + A^c_{ij} \qquad \forall i,\ j$$

Veremos que A^M satisface el punto 2 del enunciado, es decir dada una máquina $i \in I$ arbitraria

$$\sum_{j \in J} A^M_{ij} \leq 1 + 1/k$$

La asignación A^L no procesa más tiempo de proceso para los trabajos largos que el definido por los casilleros r que están asociados a dicha máquina i, lo cual queda determinado cuando se define el $\mathrm{PL}(\alpha, \kappa, T(\kappa) + 1/k)$ asociado a x^M. Dado lo anterior, y recordando que T_i es el tiempo disponible en la máquina i para procesar trabajos cortos, se tiene que:

$$\sum_{j \in J} A^L_{ij} \leq 1 - T_i$$

Por otro lado, por definición de A^c, a partir del Algoritmo 10 se tiene que:

$$\sum_{j \in J} A^c_{ij} \leq T_i + 1/k$$

entonces, para $i \in I$ arbitrario:

$$\sum_{j \in I} A^M_{ij} = \sum_{j \in J} A^L_{ij} + \sum_{j \in J} A^c_{ij} \leq 1 - T_i + T_i + 1/k = 1 + 1/k$$

Es decir, A^M utiliza máquinas con *aumento de capacidad* en $1/k$. Respecto a la transferencia de trabajos, sea j un trabajo activo según A^M, pueden ocurrir dos casos:

1. Si j es un trabajo largo, entonces debe provenir de la asignación A^L y en último término de x^L. Es decir, j es un trabajo que fue asignado a un casillero $r \neq r^*$. Como x^L asigna j a un único casillero $r \neq r^*$, y r está asociado a una única máquina $i \in I$, entonces j es procesado completamente en i.
2. Si j es un trabajo corto, entonces debe provenir de la asignación A^c, la cual en último término viene del Algoritmo 10 `round-robin con aumento`. Como se vio anteriormente, dado $j \in J$, en cada iteración del `while` del paso 5 el algoritmo 10 solo asigna a j una única vez, pues:
   (a) En cada iteración se realiza una sola asignación de j a la máquina i
   (b) Al final de cada iteración siempre se pasa al siguiente trabajo $j + 1$

Dado lo anterior, para j un trabajo activo corto o largo, no existe transferencia entre máquinas, y entonces se tienen los puntos 1 y 2. Por último, como $\mathrm{val}(A^M) = \mathrm{val}(x^M)$, satisface que:

$$\mathrm{val}(A^M) \geq \mathrm{val}(A^k)$$

Para finalizar, se escribe el algoritmo `obtener asignación` A^M `desde` x^M, que resume lo hecho en esta demostración.

Algorithm 11 obtener asignación A^M desde x^M

1: **Inputs:** $x^M,\ \kappa^M$
2: $x_e^L \leftarrow \begin{cases} x_e^M & \text{si } e = rj \text{ con } j \in J,\ r \in R \\ 0 & \text{si } e = jr^*,\ j \in J \end{cases}, \quad \forall e \in E^M$
3: $A^L : I \times J \to \mathbb{Q}_+ \leftarrow$ Algoritmo 6 `asignación de trabajos largos` con (J, I, κ^M, x^M)
4: $A_j^* \leftarrow x_{r^*j}^M t_j, \quad \forall j \in J$
5: $A^c : I \times J \to \mathbb{Q}_+ \leftarrow$ Algoritmo 10 `round-robin con aumento` en $(\{T_i\}_{i=1}^m, A^*)$
6: $A^M \leftarrow (A^L + A^c)$
7: **Return:** A^M

El algoritmo 11 `obtener asignación` A^M `desde` x^M tiene como subrutinas a los algoritmos 6 y 10, con complejidades de $\mathcal{O}(mk)$ y $\mathcal{O}(m+n)$. Por último, en el paso 2 se define x^L, donde itera sobre el conjunto $E^M = J \times R$, y entonces dicho paso toma $\mathcal{O}(n \cdot mk)$. Luego el algoritmo 11 tiene una complejidad $\mathcal{O}(n \cdot mk)$. □

6.3.1. Demostración del Teorema 6.1

Gracias al lema 6.8, ahora se está en condiciones de demostrar el Teorema 6.1 de la existencia de un PTAS para GCMP sin transferencia.

Demostración. (demostración del Teorema 6.1) Sea $A^M : I \times J \to \mathbb{Q}_+$ la asignación resultante del lema 6.8. Primero veremos que a partir de A^M basta aplicar una contracción para terminar de construir un PTAS de GCMP sin transferencia. Por lo demostrado en dicho lema, se tiene que A^M no transfiere trabajos entre máquinas y no supera la capacidad de las mismas más allá de $1+1/k$. Luego, para que A^M sea una asignación factible para GCMP sin transferencia, basta que satisfaga la restricción de capacidad 1 en cada máquina de I. Para ello se aplicará una contracción a A^M por un factor de $1/(1+1/k)$, y llamemos a dicha contracción A^G:

$$A^G := \frac{1}{1+1/k} A^M$$

Por el resultado del lema 1.7, se tiene que:

$$\text{val}(A^G) \geq \frac{1}{1+1/k} \text{val}(A^M)$$

Por otro lado, se sabe que A^M genera al menos la ganancia del problema redondeado de valor $\text{val}(A^k)$, luego:

$$\text{val}(A^G) \geq \frac{1}{1+1/k} \text{val}(A^k)$$

Además, denotando por A^o una asignación óptima para GCMP sin transferencia, sabemos que la solución del problema redondeado A^k no pierde más que un factor $1-1/k$ del valor de la solución óptima (ver lema 6.2), entonces:

$$\text{val}(A^k) \geq \left(1 - \frac{1}{k}\right) \text{val}(A^o)$$

Juntando ambas desigualdades se obtiene que:

$$\mathrm{val}(A^G) \geq \left(1 - \frac{1}{k}\right)^2 \mathrm{val}(A^o)$$

Por último, tomando $k \in \mathbb{N}$ tal que:

$$k :=\leq \left\lfloor \frac{1}{1-(1-\varepsilon)^{1/2}} \right\rfloor$$

podemos ver que

$$\mathrm{val}(A^G) \geq (1-\varepsilon)\,\mathrm{val}(A^o)$$

Es decir, A^G es una asignación factible para GCMP sin transferencia y tal que es una $(1-\varepsilon)$-aproximación del óptimo. A continuación se resume el PTAS que permitió llegar a dicha asignación y que consiste en juntar los algoritmos utilizados en los lemas anteriores:

Algorithm 12 PTAS: GCMP sin transferencia

1: **Inputs:** $I,\ J,\ \{f_j\}_{j=1}^n,\ k \in \mathbb{N},\ \mathrm{A},\ \mathrm{K}$
2: $(x^M, \kappa^M, G^M) \leftarrow$ algortimo 9 `obtener mejor solución PLs -` x^M
3: $A^M : I \times J \to \mathbb{Q}_+ \leftarrow$ algoritmo 11 `obtener` A^M `desde` x^M
4: $A^G \leftarrow \left(\frac{1}{1+1/k}\right) A^M$ ▷ Aplicamos una contracción a A^M
5: **Return:** A^G

Para analizar la complejidad del algoritmo 12 `PTAS: GCMP sin transferencia`, del lema 6.7 sabemos que obtener x^M a través del algoritmo 9 toma $\mathcal{O}(p(n,m,k)\cdot m^{k^{k^2}})$; del lema 6.8 que el algoritmo 11 toma $\mathcal{O}(n{\cdot}mk)$. Luego la complejidad del algoritmo 12 está dominada por la ejecución del algoritmo 9, y entonces tiene también complejidad $\mathcal{O}(p(n,m,k)\cdot m^{k^{k^2}})$. □

Conclusión

El problema de maximización de GCMP sin transferencia es el objeto de estudio central en esta tesis. La principal contribución hecha consiste en la construcción de distintos algoritmos polinomiales para la resolución de las variantes de este problema, los cuales permiten en su conjunto la construcción de un PTAS para GCMP sin transferencia.

Para el caso en que se permite transferencia de trabajos entre las distintas máquinas de I, se demostró que dicho problema puede ser resuelto mediante el algoritmo 2, de complejidad $\mathcal{O}(n^2)$, lo cual se resume en el Teorema 2.9, donde se establece que GCMP sin transferencia se puede reducir a Knapsack Fraccional.

El siguiente resultado es el Teorema 3.2 donde se demuestra que GCMP sin transferencia es NP-Completo, lo cual justificó la búsqueda de un PTAS para dicho caso. A continuación, se propone una forma de abordar el problema en la que se divide este en tres distintos casos, inspirándonos en lo hecho para *makespan* en [WS11]:

- GCMP sin transferencia, solo trabajos cortos. Se construye el algoritmo 5 de complejidad $\mathcal{O}(nm + n^2)$, usando la misma estrategia glotona utilizada para resolver knapsack.
- GCMP sin transferencia, solo trabajos largos. Se definen los conceptos de *asignación $k-$redondeada, configuración y configuración de máquinas*, mediante los cuales se define un grafo bipartito que conecta el conjunto de trabajos J con los casilleros o tiempos de procesos para trabajos largos. Resolviendo dicho matching para cada configuración de máquinas, se define el algoritmo 7 `GCMP sin transferencia - solo trabajos largos`, que constituye un PTAS de complejidad $\mathcal{O}(m^{k^{k^2}}(mk+n)^3)$.
- El resultado principal de este trabajo, el Teorema 6.1, demostrando que GCMP sin transferencia admite un PTAS de complejidad $\mathcal{O}(p(n,m,k)\cdot n^2 m^{k^{k^2}})$, con $p:\mathbb{R}^3\to\mathbb{R}$ un polinomio adecuado.

Por otro lado, es importante destacar las siguientes líneas de trabajo futuro:

- Evaluar extender el resultado al caso en que las funciones de ganancia no son necesariamente cóncavas.
- Estudiar el caso en que las funciones de ganancia $\{f_j\}_j$ tienen una dependencia en la máquina i donde son procesadas, es decir que $f_j = f_{ij}$.

Bibliografía

[CK05] Chandra Chekuri y Sanjeev Khanna. "A Polynomial Time Approximation Scheme for the Multiple Knapsack Problem". En: *SIAM Journal on Computing* 35.3 (2005), págs. 713-728.

[CM11] R. Chi Lau Lap Ravi y Singh Mohit. *Iterative Methods in Combinatorial Optimization*. Cambridge, 2011, pág. 17.

[EW19] Friedrich Eisenbrand y Robert Weismantel. *Priximity results and faster algorithms for Integer Programming using the Steinitz Lemma*. 2019.

[GJ79a] M. R. Garey y D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GJ79b] Michael R. Garey y David S. Johnson. *Computers and Intractability; A guide to the Theory of NP-Completeness.* 1979, págs. 96-105, 224.

[L Z86] S. Geng L. Zhang. "The complexity of the 0/1 multi-knapsack problem". En: *J. of Compt. Sci. & Technol.* 1 (1986), págs. 46-50.

[WS11] David P. Williamson y David B. Shmoys. *The Design of Approximation Algorithms.* Cambridge University Press, 2011, págs. I-XI, 1-504.