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

ALGORITMOS DE BALANCEO DE CARGA EN PROBLEMAS DE AGENDAMIENTO BAJO UNA FUNCIÓN CÓNCAVA DE LA CARGA

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN MATEMÁTICAS APLICADAS

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL MATEMÁTICO

CRISTIAN JAVIER PALMA FOSTER

PROFESOR GUÍA:
JOSÉ SOTO SAN MARTÍN

MIEMBROS DE LA COMISIÓN:
MARCOS KIWI KRAUSKOPF
JOSÉ CORREA HAEUSSLER
VÍCTOR VERDUGO SILVA
WALDO GÁLVEZ VERDUGO

Este trabajo ha sido parcialmente financiado por CMM ANID BASAL FB210005 y FONDECYT REGULAR 1231669

SANTIAGO DE CHILE
2024

RESUMEN DE TESIS PARA OPTAR AL GRADO
DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA,
MENCIÓN MATEMÁTICAS APLICADAS
Y MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO
POR: CRISTIAN JAVIER PALMA FOSTER
FECHA: 2024
PROF. GUÍA: JOSÉ SOTO SAN MARTÍN

ALGORITMOS DE BALANCEO DE CARGA EN PROBLEMAS DE AGENDAMIENTO BAJO UNA FUNCIÓN CÓNCAVA DE LA CARGA

En el área de algoritmos combinatoriales de optimización, los problemas de agendamiento buscan asignaciones de trabajos a máquinas. La carga de una máquina es la suma de los tiempos de proceso de sus trabajos asignados. En este estudio se busca analizar la complejidad y desarrollar algoritmos para un problema de balanceo de carga, el cual consiste en encontrar una asignación de trabajos a máquinas maximizando la suma de una función cóncava no negativa evaluada en la carga.

La principal motivación detrás de este problema está en el área de Investigación de Operaciones, donde la función cóncava corresponde a la productividad de una máquina y los trabajos a recursos. Otra aplicación en Economía proviene de ver a los trabajos como objetos de valor y a las máquinas como personas con funciones de utilidad.

El problema fue considerado por primera vez por Alon et al. [2], donde se obtiene un **EPTAS** (esquema de aproximación a tiempo polinomial eficiente) para la versión *offline*. Esto es esencialmente lo mejor posible dado que el problema es fuertemente **NP**-difícil [12]. En este trabajo se prueba que en este contexto regla *Longest Processing Time* [13] alcanza una $2(\sqrt{2}-1) \approx 0{,}828$-aproximación. Se conjetura que este análisis no es ajustado, teniendo solo una cota superior de $11/12 \approx 0{,}916$. Para el caso de dos máquinas, un análisis ajustado muestra que esta regla es exactamente una $11/12$-aproximación.

En la versión *online* de este problema los trabajos se van revelando uno a uno y deben ser asignados sin arrepentimientos. En orden adversarial, se prueba que el algoritmo glotón *List Scheduling* [13] es exactamente 3/4-competitivo. Se prueba además que esto es óptimo para algoritmos online deterministas que no evalúen la función cóncava. Se exhibe una cota superior constante sobre la competitividad de cualquier algoritmo determinista de $\phi/2 \approx 0{,}809$, con ϕ el número de oro y se propone un algoritmo que la alcanza en dos máquinas. Como siguiente paso en búsqueda de un algoritmo óptimo se encuentran cotas que aplican para más de dos máquinas. Se obtienen también cotas para algoritmos aleatorios y en orden aleatorio.

RESUMEN DE TESIS PARA OPTAR AL GRADO
DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA,
MENCIÓN MATEMÁTICAS APLICADAS
Y MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL MATEMÁTICO
POR: CRISTIAN JAVIER PALMA FOSTER
FECHA: 2024
PROF. GUÍA: JOSÉ SOTO SAN MARTÍN

LOAD BALANCING ALGORITHMS ON SCHEDULING PROBLEMS WITH A CONCAVE FUNCTION OF THE LOAD

In the field of combinatorial optimization algorithms, scheduling problems consider the assignment of jobs to machines. The load on a machine is the sum of the processing times of the assigned jobs. This study aims to analyze the complexity and develop algorithms for a load balancing problem, which involves finding an assignment of jobs to machines that maximizes the sum of a non-negative concave function valuation of the load.

The main motivation behind this problem lies in the field of Operations Research, where the concave function corresponds to the productivity of a machine, and the jobs to resources. Another application in Economics arises from viewing jobs as valuable objects and machines as individuals with utility functions.

The problem was first considered by Alon et al. [2], where an **EPTAS** (Efficient Polynomial-Time Approximation Scheme) is obtained for the offline version. This is essentially the best possible since the problem is strongly **NP**-hard [12]. In this work, we prove that in this context, the *Longest Processing Time* rule [13] achieves a $2(\sqrt{2}-1) \approx 0{,}828$ approximation. It is conjectured that this analysis is not tight, having only an upper bound of $11/12 \approx 0{,}916$. For the case of two machines, a tight analysis shows that this rule is exactly an $11/12$-approximation.

In the online version of this problem jobs are revealed one by one and must be assigned without regrets. In adversarial order, it is proven that the greedy algorithm *List Scheduling* [13] is exactly 3/4-competitive. It is also proven that it is optimal for deterministic online algorithms that do not evaluate the concave function. A constant upper bound on the competitiveness of any deterministic algorithm of $\phi/2 \approx 0{,}809$ is exhibited, with ϕ being the golden ratio, and an algorithm that achieves it on two machines is given. As a next step towards an optimal algorithm, bounds are obtained that apply to more than two machines. Bounds for random and arbitrary order algorithms are also derived.

"'Supera tus límites, aquí y ahora."
- Yami Sukehiro

Agradecimientos

Comienzo agradeciendo a mi familia, por su constante apoyo e incentivo en mi interés por la matemática y mi desarrollo moral. A mi mamá Isabel por su gran cariño, todas sus enseñanzas y por su apoyo incondicional, gran parte de esta tesis te pertenece. A mi papá Miguel por ser un ejemplo de trabajo duro y por proporcionarme siempre las herramientas para mi crecimiento. Gracias a ambos por siempre haberme dado el mejor ejemplo posible. A mi hermana Francisca por su apoyo y preocupación constante. Y especialmente a mi hermanita Isabel, quien me alegra los días desde que llegó a este mundo, gracias por cada sonrisa compartida.

Quiero también agradecer a todos mis amigos por impulsarme a tomar los desafíos que me forjaron como persona. Mis amigos del colegio, Pancho, Vicho y Pingu. Mis amigos de olimpíadas Bruno y Pato. A mis amigos de plan común Vicky, Nico, Max y David. Mis amigos del DIM; Arie, Felipe, Azócar y Benja. Mis amigos del doble título Marga, Elisa, Laura, Nico, Nacho, Afnan y Théo. A mi polola Anto, gracias por siempre estar ahí para mí, para subirme el ánimo y apoyarme. Y a muchos otros que no alcanzo a mencionar. Son personas increíbles y estoy muy agradecido de haberlos conocido.

Luego, agradecer a quienes conocí en la universidad y complementaron mi formación. Agradezco a todos mis profesores de la educación media y superior. En particular a Joaquín, quien amplió mi comprensión de la matemática. A Matías, quien fue el mejor profesor que podría haber tenido en plan común, reavivando mi pasión por la matemática, y quién me dio mi primera oportunidad como profesor auxiliar. A mis profesores del área discreta, Maya, Marcos e Iván, quienes me presentaron herramientas avanzadas en esta hermosa área y me aceptaron reiteradas veces en sus equipos docentes. También agradecer a todos los que fueron mis alumnos, llegando con preguntas interesantes y motivándome a crear nuevos problemas y respuestas.

Agradecer notablemente a mi profesor guía José Soto, siempre apuntándome en la dirección correcta para avanzar. Gracias a usted tengo una mejor apreciación del desarrollo y análisis de algoritmos. También le agradezco su gran preocupación y apoyo en mis siguientes pasos. Sin importar que tan ocupado estuviera, siempre se daba el tiempo para una charla. Gracias por su compromiso conmigo y con esta tesis.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Motivación

Considere la siguiente situación: usted dispone de n unidades de distintos tamaños de un mismo recurso (combustible), y desea asignarlos a m máquinas iguales que trabajan de forma paralela (independientes entre ellas), de manera de que sean lo más productivas posibles. La productividad de cada máquina está dada por una función f que depende solamente de la cantidad del recurso que se le asigna. Los recursos se consideran indivisibles (galones), por lo que cada unidad del recurso debe ser asignado completamente a una máquina. Se define la productividad total como la suma de la productividad de cada máquina. Se puede suponer además que la función de productividad f cumple:

- (Normalizada) Las máquinas no producen nada si no reciben recursos.
- (Creciente) Entre más recursos recibe una máquina, mayor productividad.
- (Cóncava) Asignarle un recurso a una máquina aumenta menos la productividad mientras más recursos tenga la máquina.

Este problema, al cual nos referiremos como *Machine Productivity*, es importante en el área de Investigación de Operaciones (*Operations Research*). *Machine Productivity* cae dentro de la familia de problemas de *Agendamiento* (*Scheduling* en inglés). Más particularmente, al solo importar la carga en cada máquina, esto corresponde a un problema de *Balanceo de Carga* (*Load Balancing* en inglés). Otros problemas de este tipo son *Makespan Minimization* (Minimización del Tiempo de Completación) y *Machine Covering* (Cubrimiento de Máquinas). Estos se pueden definir en el mismo contexto: se tienen n trabajos de tiempos de proceso distintos a ser agendadas en m máquinas paralelas idénticas. Por un lado, en *Makespan Minimization*, se busca minimizar la mayor carga entre las máquinas. Por el otro, en *Machine Covering*, se busca maximizar la menor carga entre las máquinas. Notar que si en *Machine Productivity* se cambia la suma por un mínimo de productividades, se vuelve a *Machine Covering*, pues al ser f creciente, la productividad mínima la tendrá la máquina de menor carga.

Desde un punto de vista económico, *Machine Productivity* es un caso particular de otro problema general de asignación de recursos, *Submodular Welfare*. En este problema, se tienen n objetos de valor distintos a ser repartidos en m personas con funciones de utilidad distintas, las cuales dependen de los objetos asignados. Se supone que estás funciones son submodulares, monótonas y normalizadas, extensiones de las propiedades descritas para una función de productividad. Más allá de la hipótesis de funciones de utilidad distintas (la cual se puede aplicar en *Machine Productivity* considerando distintas funciones de productividad), como los objetos pueden ser distintos, no se puede decir que haya uno mejor que otro pues dependerá de cada persona. Yendo más allá en cuanto a generalidad se encuentra *Submodular Matroid Maximization*, donde ya no se habla de recursos ni personas, sino que uno busca en general un conjunto dentro de una matroide que maximice una función submodular. Se puede probar que esto generaliza a *Submodular Welfare* notando que el problema de asignación define una matroide de partición. Resulta interesante notar que a pesar de esto, los resultados en ambos problemas tienden a coincidir, es decir, las mejores aproximaciones y resultados negativos son los mismos, lo cual se puede interpretar como que sus dificultades no son tan distintas.

Al tener relativamente malos resultados en *Submodular Welfare* tanto en sus versiones *online* como *offline*, es de interés buscar modelos con más específicos, sacrificando generalidad pero resolviendo mucho mejor problemas concretos. Un ejemplo es *Budgeted Allocation*, el cual ha captado la atención en los últimos años. Este problema nace de la asignación de anuncios en sistemas de búsqueda, donde los postores tiene un presupuesto, por lo que la función de utilidad es exactamente el costo del anuncio hasta que el costo supere el presupuesto, y en tal caso se cobra solo el presupuesto. Notar que estas funciones representan funciones de productividad, pero aun así este problema es distinto pues permite que cada postor pueda o no valorar un objeto, concepto que para *Machine Productivity* no existe pues los objetos son idénticos y un objeto más grande será más valorado por todas las máquinas.

1.2. Trabajo Relacionado

La versión *offline* del problema consiste en tener toda la información de la instancia al inicio de la ejecución de un algoritmo. Mientras tanto, en la versión *online* de *Machine Productivity* los trabajos llegan uno a uno y deben ser asignados inmediatamente sin arrepentimientos. Sea $0 \leq \alpha \leq 1$. Un algoritmo para un problema online se dice α-competitivo si en cada instancia, el valor de la solución encontrada por el algoritmo es al menos un factor α del óptimo offline. En general, se pide además que el algoritmo decida en tiempo polinomial a qué máquina agregar cada trabajo. Análogamente, un algoritmo para un problema offline se dice α-aproximación si en cada instancia, el valor de la solución encontrada por el algoritmo es al menos un factor α del óptimo y es a tiempo polinomial en la entrada. Un esquema de aproximación a tiempo polinomial, *Polynomial-time Approximation Scheme* o **PTAS**, es un algoritmo tal que dado un $\varepsilon > 0$, asegura una $(1 - \varepsilon)$-aproximación. Dicho esquema se dice eficiente, **EPTAS**, si en su tiempo de ejecución la dependencia en ε multiplica a un polinomio en el tamaño de la entrada. Finalmente, el esquema se dice que es completamente a tiempo polinomial, **FPTAS**, si también es a tiempo polinomial en $1/\varepsilon$ además de en la entrada.

1.2.1. Versión Offline

Alon et al. [2] conjeturaron que la versión offline de *Machine Productivity* es fuertemente **NP**-difícil y exhibieron un **EPTAS**. El esquema se enuncia no solo para la minimización de la suma de una función cóncava, sino también para la maximización de la suma de una función convexa. Esto se hace bajo una hipótesis de continuidad fuerte, la cual se puede quitar en el caso cóncavo, por lo que el esquema es válido para *Machine Productivity*. El tiempo de ejecución del **EPTAS** fue posteriormente mejorado por Jansen et al. [16].

Suponiendo que *Machine Productivity* es fuertemente **NP**-difícil, salvo si $\mathbf{P} = \mathbf{NP}$, no existen algoritmos polinomiales ni **FPTAS** para este problema. Entonces, el **EPTAS** de Alon et al. es esencialmente lo mejor posible para aproximar *Machine Productivity* eficientemente. Esto deja cerrada la versión offline del problema, quedando abierta la online, la cual se aborda en esta tesis. Aún está abierta la pregunta si es que la versión con máquinas distintas admite o no un **PTAS**.

Una extensión natural de *Machine Productivity* es permitir que la carga de cada trabajo sea distinta según en qué máquina se asigne. Kell y Sun [19] definen el problema de *Asignación Indivisible con Evaluaciones Cóncavas-Aditivas* (ICA por sus siglas en inglés). La entrada consiste en tiempos de proceso $p_{i,j} \geq 0$ distintas para cada máquina i y trabajo j, y funciones de productividad f_i para cada máquina i, buscando una asignación de trabajos a máquinas que maximice la productividad. Usando la curvatura de las funciones como parámetro, Kell y Sun obtuvienen aproximaciones aditivas y multiplicativas. Esta extensión incluye el problema de *Emparejamiento de Cardinalidad Máxima* en grafos bipartitos (restringiendo a $p_{i,j} \in \{0, 1\}$, y $f_i(x) = \min(x, 1)$) y el de *Maximum Budgeted Allocation* (Asignación Máxima con Presupuestos) usando funciones lineales con presupuesto (o truncadas) $f_i(x) = \min(x, B_i)$ para algún presupuesto $B_i \geq 0$ en cada máquina. Mientras que el primer problema se puede resolver en tiempo polinomial, el segundo no puede ser aproximado con un factor mejor que 15/16 [8], incluso en el caso de cargas uniformes, esto es, tal que cada trabajo j, cuyo tiempo de proceso $p_{i,j}$ para una máquina i es o bien un valor fijo $\alpha_j \geq 0$ o bien 0. Por el lado positivo, hay una 3/4-aproximación para *Maximum Budgeted Allocation* [8, 25].

Generalizando más allá, tenemos el problema *Submodular Welfare*, en el que cada máquina tiene una función de utilidad (submodular, normalizada y monótona) $u_i : \mathcal{P}([n]) \to \mathbb{R}_+$ sobre el conjunto de todos los trabajos, y el problema consiste en asignar los trabajos maximizando la suma de las utilidades. Esto es un caso especial de maximización de una función submodular sobre un matroide de partición, el algoritmo glotón offline logra una aproximación de 1/2 [11]. Este algoritmo encuentra iterativamente un par (i, j) donde j es un trabajo no asignado e i es una máquina, de modo que asignar j a i maximice el aumento marginal en la función objetivo. Más tarde, Vondrák [26] desarrolló un algoritmo glotón continuo basado en *pipage rounding* que logra una aproximación ajustada de $1 - 1/e$ para este problema. Es interesante destacar que Vondrák muestra que en el caso simétrico donde todas las funciones u_i son iguales, el algoritmo de asignación aleatoria que elige para cada trabajo una máquina de manera uniforme al azar e independiente del resto, tiene la misma garantía que el producido por el algoritmo glotón continuo.

1.2.2. Versión Online

En un contexto online, también podemos permitir cargas dependientes de la máquina para los trabajos: al llegar, el trabajo j revela todos los valores $p_{i,j} \geq 0$ para cada máquina i. En el trabajo seminal de Karp, Vazirani y Vazirani [18], se exhibe un algoritmo $(1 - 1/e)$-competitivo para el problema de *Emparejamiento Online* (con $p_{i,j} \in \{0, 1\}, f_i = \text{mín}(x, 1)$) y se demuestra que esta garantía es ajustada. Para *Online Budgeted Allocation* ($p_{i,j} \geq 0$ y funciones lineales con presupuesto), existen algoritmos $(1-1/e)$-competitivos bajo el supuesto de cargas pequeñas [23]. Para cargas generales, el mejor algoritmo actual [15] logra un factor competitivo de 0,5016, y la mejor cota superior sigue siendo $1 - 1/e$, heredado del problema de Emparejamiento Online.

En *Submodular Welfare* online, suponemos que un algoritmo online puede consultar el valor de u_i sobre cualquier subconjunto de trabajos que ya hayan llegado. Quizás sorprendentemente, el algoritmo glotón online que asigna cada elemento que llega a la máquina que maximiza el aumento marginal en el objetivo es 1/2-competitivo [21]. Kapralov, Post y Vondrak [17] demuestran que ningún algoritmo online puede superar 1/2 salvo si **NP** = **RP**, y esta cota se mantiene incluso si las funciones submodulares son funciones de cobertura. Como el algoritmo de asignación aleatoria de Vondrák [26] es naturalmente online, también es $(1 - 1/e)$-competitivo para *Submodular Welfare* online con funciones de utilidad idénticas.

1.3. Nuestros Resultados

La mejor garantía actual la versión online de *Machine Productivity* es $(1 - 1/e) \approx 0{,}6321$. Se logra mediante el algoritmo de asignación aleatoria de Vondrák y no existen límites superiores no triviales. En el Capítulo 3 mostramos que el algoritmo *List Scheduling* (*LS*) (que coincide con el algoritmo glotón online que es solo 1/2 competitivo para *Submodular Welfare* online) para *Machine Productivity* alcanza una factor competitivo de exactamente $3/4 = 0{,}75$ cuando m es par y exactamente $3/4 + 1/(4m^2)$ si m es impar.

Denotando por $\phi = (1 + \sqrt{5})/2$ al número de oro, en el Capítulo 4 mostramos que ningún algoritmo online determinista tiene una competitividad mejor que $\phi/2 \approx 0{,}809$ incluso en dos máquinas, y que ningún algoritmo online aleatorio tiene una competitividad mejor que $9/10 = 0{,}9$. También se demuestra que si nos restringimos a algoritmos *return-oblivious*, es decir, que no pueden acceder a la función de productividad, por lo que solo pueden ver las cargas (tal es el caso de *LS*), entonces la cota es de $3/4 = 0{,}75$ para algoritmos deterministas y $5/6 \approx 0{,}833$ para algoritmos aleatorios. Luego se exhibe un algoritmo óptimo para el contexto online determinista en dos máquinas, denominado *Balanceo Áureo*, pues su competitividad es exactamente $\phi/2 \approx 0{,}809$, ajustada a la cota superior. Se dan además cotas en orden aleatorio y cotas para el orden adversarial que se cumplen para más de dos máquinas.

En el Capítulo 5, analizamos la regla *Longest Processing Time First* (*LPT*) para la versión offline de *Machine Productivity* y mostramos que su factor de aproximación está entre $2(\sqrt{2} - 1) \approx 0{,}828$ y $11/12 \approx 0{,}9166$. Curiosamente, la única garantía disponible para *LPT* era 1/2, ya que corresponde al algoritmo glotón offline para *Submodular Welfare*. En

segundo lugar, LPT es un algoritmo *return-oblivious* y, antes de nuestro análisis, la mejor garantía de este tipo era el factor $(1 - 1/e)$ alcanzado por una asignación aleatoria [26]. Refinando el análisis de LPT en dos máquinas, mostramos que esta regla es exactamente una 11/12-aproximación.

En el Anexo A se discute el estado del arte con respecto a esquemas de aproximación polinomial en problemas de balanceo de carga y se enuncian ligeros avances. Por medio de simples observaciones se logran simplificaciones tanto para *Machine Productivity* como para la generalización de *Machine Covering* cuando las cargas se evalúan en una función cóncava no negativa. Se generaliza además un componente del esquema para *Machine Productivity* al caso de máquinas distintas.

Capítulo 2

Productividad en Máquinas Cóncavas Paralelas

2.1. Problema de Decisión

Denotaremos por $\mathbb{N} = \{0, 1, 2, \ldots\}$ y $\mathbb{R}_+ = [0, +\infty)$ al conjunto de números enteros y reales no negativos, respectivamente. Dado $n \in \mathbb{N}$, denotaremos $[n] = \{1, \ldots, n\}$ al conjunto de enteros entre 1 y n incluidos. Finalmente, dado un conjunto X, denotamos por $\mathcal{P}(X) = \{A \subseteq X\}$ el conjunto de sus partes.

Para modelar la productividad de las máquinas consideraremos el siguiente tipo de funciones:

Definición 2.1 (Función de Productividad)**.** $f : \mathbb{R}_+ \to \mathbb{R}_+$ se dice función de productividad si es cóncava y satisface $f(0) = 0$.

Observación 2.2. La hipótesis de normalización $f(0) = 0$ se puede relajar siempre y cuando se mantenga la no negatividad de la función. Para una función productividad no normalizada g, se puede considerar la función $f = g - g(0)$, la cual es de productividad normalizada. Si se tiene una garantía de aproximación sobre f, está se traduce a una para g pues para toda asignación se suma el mismo valor $mg(0)$, con m el número de máquinas.

Dentro de las funciones de productividad se encuentran $\log(x + 1)$, x^α para $\alpha \in [0, 1]$. Además de la siguiente familia.

Ejemplo 2.3 (Funciones Lineales Truncadas)**.** Dado $L \in \mathbb{R}_+$, definimos la función $f_L : \mathbb{R}_+ \to \mathbb{R}_+$, tal que $\forall x \in \mathbb{R}_+, f_L(x) = \text{mín}(x, L)$. Se tiene que f_L es función de productividad.

Nos será útil considerar esta familia de funciones de productividad pues son a la vez simples de visualizar, permitiendo definir algoritmos y probar resultados fácilmente, además de ser razonablemente representativas, permitiendo trasladar dichos algoritmos al caso general y siendo muchas veces suficientes para probar cotas superiores.

También se pueden obtener más funciones de las existentes: si f es función de productividad, también lo es $f(\lambda\cdot)$ para $\lambda \geq 0$. Más aún, las funciones de productividad son cerradas bajo combinaciones cónicas.

Planteemos el problema de decisión que buscamos resolver. Dada $p : [n] \to \mathbb{R}_+$ una función de peso sobre $[n]$, consideraremos su extensión como función de conjuntos, es decir, $p : \mathcal{P}([n]) \to \mathbb{R}_+$ definida por: $\forall J \subseteq [n], p(J) = \sum_{j \in J} p_j$.

Definición 2.4 (Machine Productivity)**.** El problema de decisión de *Machine Productivity* consiste en decidir si dados $m, n \in \mathbb{N}$ cantidad de máquinas y trabajos, respectivamente, $p : [n] \to \mathbb{Q}_+$ tiempos de proceso, $k \in \mathbb{Q}_+$ productividad objetivo, $f : \mathbb{Q}_+ \to \mathbb{Q}_+$ función de productividad de las máquinas en función de su carga, tales que existe una asignación en m máquinas tal que la suma de las productividades es al menos k. Formalmente:

$$\textsc{MachineProd} = \{\langle m, n, p, f, k\rangle \mid \exists \{A_i\}_{i\in[m]} \text{ asignación de } [n], \sum_{i\in[m]} f(p(A_i)) \geq k\}.$$

Para que el problema esté bien definido es necesario contar con una manera de describir las funciones de productividad. Para atender a esta pregunta en la siguiente sección se presentan dos alternativas usadas en la literatura, uso de oráculos de valor y restricción a representaciones explícitas.

2.2. Supuestos

Es usual considerar para *Submodular Welfare* y *Submodular Matroid Maximization* [26, 24, 17] oráculos de valor, oráculos de demanda y representaciones explícitas. Un oráculo de valor para una función $g : \mathcal{P}(X) \to \mathbb{R}$ es tal que dado un conjunto de trabajos $S \subseteq X$, entrega el valor de $g(S)$. Similarmente, un oráculo de valor para una función $f : \mathbb{R} \to \mathbb{R}$ recibe $x \in \mathbb{R}$ y entrega el valor de $f(x)$. Así, este oráculo solo permite el acceso a la función bajo un mecanismo de *caja negra*. Por otro lado, un oráculo de demanda para g puede responder preguntas más fuertes: dados precios sobre los objetos $\pi : X \to \mathbb{R}$, retorna el conjunto $S \subseteq X$ que maximiza $g(S) - \pi(S)$. Para f esto corresponde a: dada una ponderación $\alpha \in \mathbb{R}$, el oráculo retorna el $x \in \mathbb{R}$ que maximiza $f(x) - \alpha x$. Cabe destacar que en el contexto de funciones sobre conjuntos, el oráculo de demanda es estrictamente más poderoso que el de valor [3], en el sentido en que un oráculo de demanda puede simular uno de valor en tiempo polinomial, pero uno de valor puede necesitar un tiempo exponencial para simular uno de demanda. Finalmente, una representación explícita de g corresponde a una descripción en binario que pueda ser interpretada y así ser evaluada. Por ejemplo, descripciones de máquinas de Turing que calculen la función g. Esto limita la cantidad y las posibles funciones a utilizar. De la misma manera podemos considerar representaciones explícitas para f.

Notemos que en caso de elegir usar representaciones explícitas, se evita la verificación de que sea efectivamente una función de productividad, asumiendo que lo es. Este tipo de problemas se les llama problema *promesa*. Determinar qué funciones de productividad tienen representación explícita está fuera del marco de esta tesis. Aun así, es importante notar que las funciones lineales truncadas sí la tienen, pues basta conocer $L \in \mathbb{Q}_+$. Con esto, *Machine*

Productivity restringido a funciones calculables en tiempo polinomial cuenta, al menos, con esta familia de funciones para ejemplos y cotas superiores.

2.3. Dificultad del Problema

Primero estudiamos la dificultad del problema de decisión asociado a *Machine Productivity*. Es fácil ver que este problema está en **NP**.

Proposición 2.5. MachineProd $\in$ **NP**

Demostración. Sea $\langle m, n, p, f, k\rangle$ una instancia de MachineProd, consideremos el certificado $\{J_i\}_{i\in[m]}$ partición de $[n]$ que representa un agendamiento de m procesadores. Una partición de $[n]$ se puede codificar como un arreglo de tamaño m de listas de números en $[n]$, cada uno se codifica en $\log n$ bits, por lo que el certificado es de tamaño $O(n \log n)$. Verificar que $\sum_{i\in[m]} f(p(J_i)) \geq k$ se puede hacer en tiempo polinomial, pues las sumas tienen una cantidad a lo más lineal de términos, f se evalúa mediante un oráculo de valor y la comparación se puede realizar en tiempo lineal. □

Nuestro primer resultado consiste en demostrar que MachineProd es fuertemente **NP**-difícil. Para ello, consideremos el problema *Machine Covering*. El problema de decisión asociado a *Machine Covering* consiste en decidir si dados $m, n \in \mathbb{N}$ cantidad de máquinas y trabajos, respectivamente, $p : [n] \to \mathbb{Q}_+$ tiempos de proceso, $k \in \mathbb{Q}_+$ tiempo de procesamiento objetivo, existe un agendamiento de m procesadores para p tal que cada máquina procese al menos un tiempo de k. Formalmente:

$$\text{MachineCover} = \{\langle m, n, p, k\rangle \mid \exists\{J_i\}_{i\in[m]} \text{ partición de } [n], \forall i \in [m], p(J_i) \geq k\}.$$

Para la reducción, representaremos explícitamente una función lineal truncada f_L por el valor $L \in \mathbb{Q}$.

Teorema 2.6. MachineCover *se reduce a* MachineProd *en tiempo polinomial usando representación unaria.*

Demostración. Sea $\langle m, n, p, k\rangle$ una instancia de MachineCover, consideramos la instancia $\langle m, n, p, k, mk\rangle$ de MachineProd, es decir con m máquinas, n trabajos, una función $f_k(x) = \text{mín}(x, k)$ lineal truncada al valor k y una productividad objetivo de mk.

Esta es efectivamente una reducción, ya que cualquier asignación A con carga de al menos k en cada máquina tiene productividad de mk y viceversa. □

Como MachineCover es fuertemente **NP**-difícil [12], juntando los dos resultados previos hemos demostrado el siguiente corolario.

Corolario 2.7. MachineProd *es fuertemente* **NP***-completo.*

Observación 2.8. Notar que con la misma demostración, MachineProd sigue siendo fuertemente **NP**-completo incluso restringido a solo usar funciones lineales truncadas f_L con $L \in \mathbb{Q}_+$.

Esto responde a la conjetura de Alon et al. [2] sobre la fuerte **NP**-completitud de este problema.

2.4. Preliminaries

2.4.1. Concavidad

Antes de estudiar el problema demostraremos lemas particularmente útiles cuando se trabaja con funciones de productividad. Una función f se dice *superhomogénea* si $\forall \alpha \in [0,1], \forall x \geq 0, f(\alpha x) \geq \alpha f(x)$, y se dice *subaditiva* si $\forall x, y \geq 0, f(x+y) \leq f(x) + f(y)$.

Lema 2.9. *Si f es cóncava y no negativa entonces f es superhomogénea.*

Demostración. Basta tomar la combinación convexa $\alpha x = \alpha x + (1-\alpha)0$. □

Corolario 2.10. *Si f es cóncava y no negativa entonces f es subaditiva.*

Demostración. Tomando $\alpha = \frac{x}{x+y}$, tenemos que $f(x) = f(\alpha(x+y)) \geq \alpha f(x+y)$ y $f(y) = f((1-\alpha)(x+y)) \geq (1-\alpha) f(x+y)$. Sumando ambas desigualdades se concluye. □

A pesar de haber mencionado la hipótesis de crecimiento en la motivación en la Sección 1.1, la siguiente propiedad nos dice que es redundante pedírselo a una función de productividad.

Proposición 2.11. *Si f cóncava y no negativa entonces es no decreciente.*

Demostración. Por contradicción, supongamos que existen $x > y \geq 0$ tales que $f(x) < f(y)$. Sea $z > x$, por concavidad, como $x = \frac{z-x}{z-y} y + \frac{x-y}{z-y} z$ es combinación convexa de y y de z,

$$f(x) \geq \frac{z-x}{z-y} f(y) + \frac{x-y}{z-y} f(z).$$

Despejando $f(z)$ se obtiene:

$$\frac{f(x) - f(y)}{x-y} z + \frac{x f(y) - y f(x)}{x-y} \geq f(z).$$

Como $f(x) - f(y) < 0$, entonces si $z \to \infty$, entonces $f(z) \to -\infty$. Una contradicción pues f es no negativa. □

En adelante consideraremos $p : [n] \to \mathbb{R}_+$ extendida a su función de peso y $f : \mathbb{R}_+ \to \mathbb{R}_+$ una función de productividad.

Lema 2.12 (Intercambio Convexo)**.** *Sean $a, b, c, d \in \mathbb{R}_+$ tales que $a \leq b \leq d$ y $b + c = a + d$ (luego $a \leq c \leq d$), entonces $f(b) + f(c) \geq f(a) + f(d)$.*

Demostración. Notemos que se verifican $0 \leq d - b \leq d - a$ y $0 \leq b - a \leq d - a$. Además se tiene $d - b + b - a = d - a$ y $b = \frac{d-b}{d-a}a + \frac{b-a}{d-a}d$, la cual es una combinación convexa de a y d. Por concavidad: $\frac{d-b}{d-a}f(a) + \frac{b-a}{d-a}f(d) \leq f\left(\frac{d-b}{d-a}a + \frac{b-a}{d-a}d\right) = f(b)$. Análogamente, se nota que $c = \frac{d-c}{d-a}a + \frac{c-a}{d-a}d$ y se obtiene: $\frac{d-c}{d-a}f(a) + \frac{c-a}{d-a}f(d) \leq f(c)$. Sumando ambas desigualdades y usando que $b + c = a + d$ se concluye que $f(a) + f(d) \leq f(b) + f(c)$. □

Los siguientes corolarios son usos particulares que se le dan a este lema a lo largo de esta tesis. Sea $J \subseteq [n]$ un conjunto de trabajos. Por simplicidad, para $j \in J$ y $k \notin J$, denotamos $J - j = J \setminus \{j\}$ y $J + k = J \cup \{k\}$. Además, para todo $j \in [n]$, definimos el aporte marginal de un trabajo j al conjunto de trabajos J, por: $f \circ p(j \mid J) = f \circ p(J + j) - f \circ p(J)$.

Corolario 2.13 (Marginales Decrecientes)**.** *Sean $A, B \subseteq [n] - j$. Si $p(A) \leq p(B)$, entonces $f \circ p(j \mid A) \geq f \circ p(j \mid B)$.*

Demostración. Se verifica $p(A) \leq p(B) \leq p(B+j)$ y se tiene $p(A+j)+p(B) = p(A)+p(B+j)$. Aplicando el Lema de Intercambio Convexo con $a = p(A)$, $b = p(B)$, $c = p(A + j)$ y $d = p(B + j)$, se concluye que $f \circ p(A) + f \circ p(B + j) \leq f \circ p(A + j) + f \circ p(B)$. Acomodando los términos se obtiene la relación buscada entre los marginales. □

Corolario 2.14 (Reasignación Convexa)**.** *Sea $\{J_i\}_{i\in[m]}$ una asignación de trabajos, sean $\bar{i}, \underline{i} \in [m]$ y $J \subseteq J_{\bar{i}}$ tales que $p(J_{\underline{i}}) \leq p(J_{\bar{i}} \setminus J)$. Entonces la asignación obtenida moviendo los trabajos J de $\bar{i}$ a $\underline{i}$ tiene al menos la misma productividad, es decir, la asignación definida por:*

$$\forall i \notin \{\bar{i}, \underline{i}\}, J'_i = J_i, \qquad J'_{\bar{i}} = J_{\bar{i}} \setminus J, \qquad J'_{\underline{i}} = J_{\underline{i}} \cup J,$$

cumple $\sum_{i\in[m]} f(p(J_i)) \leq \sum_{i\in[m]} f(p(J'_i))$.

Demostración. Notar que las máquinas $i \neq \bar{i}, \underline{i}$ mantienen su productividad, y para comparar las otras dos máquinas basta usar el Lema de Intercambio Convexo con $a = p(J_{\underline{i}})$, $b = p(J'_{\bar{i}})$, $c = p(J'_{\underline{i}})$ y $d = p(J_{\bar{i}})$. Donde se verifica que $p(J'_{\bar{i}})+p(J'_{\underline{i}}) = p(J'_{\bar{i}})+p(J)+p(J_{\underline{i}}) = p(J_{\bar{i}})+p(J_{\underline{i}})$, y por monotonía de p: $p(J_{\underline{i}}) \leq p(J'_{\underline{i}})$ y $p(J'_{\bar{i}}) \leq p(J_{\bar{i}})$. Además, por hipótesis $p(J_{\underline{i}}) \leq p(J'_{\bar{i}})$ y:

$$p(J'_{\underline{i}}) = p(J_{\underline{i}} \cup J) = p(J_{\underline{i}}) + p(J) \leq p(J_{\bar{i}} \setminus J) + p(J) = p(J_{\bar{i}}).$$ □

Lema 2.15. *$f \circ p$ es submodular monótona normalizada.*

Demostración. Es normalizada pues $f \circ p(\emptyset) = f(p(\emptyset)) = f(0) = 0$ y monótona pues p y f lo son. Además, del Corolario de Marginales Decrecientes, como $A \subseteq B$ implica $p(A) \leq p(B)$, $f \circ p$ tiene marginales decrecientes como función de conjuntos, por lo que es submodular. □

Observación 2.16. Sea A una partición de los trabajos y $I \subseteq [m]$ un conjunto de máquinas, cada una con carga no superior a $\overline{p}$, entonces

$$\sum_{i\in I} f(p(A_i)) \geq \frac{1}{\overline{p}}\Big(\sum_{i\in I} p(A_i)\Big) f(\overline{p}).$$

Demostración. Para cada $i \in I$, $p(A_i)/\overline{p} \leq 1$, entonces por superhomogeneidad $f(p(A_i)) \geq \frac{1}{\overline{p}} p(A_i) f(\overline{p})$. Sumando sobre $i \in I$ obtenemos el resultado deseado. □

2.4.2. Trabajos Gigantes y Cotas para el Óptimo

Se define la carga promedio de la instancia como $\overline{p} = \frac{1}{m} p([n])$. Sea opt el valor óptimo de productividad de la instancia.

Proposición 2.17 (Cota de la Carga Promedio)**.** *Se tiene que* $\text{opt} \leq m f(\overline{p})$.

Demostración. En efecto, sabemos que por la desigualdad de Jensen con coeficientes iguales a $\frac{1}{m}$ para los valores $\forall i \in [m], p(\text{OPT}_i) \geq 0$, tenemos que $\sum_{i \in [m]} p(\text{OPT}_i) = p([n])$ por lo que:

$$f\left(\frac{1}{m} p([n])\right) \geq \frac{1}{m} \sum_{i \in [m]} f(p(\text{OPT}_i)) = \frac{1}{m} \text{opt}.$$ □

Como los trabajos de largo nulo no afectan la productividad de una asignación, supondremos que $\forall j \in [n], p_j > 0$. Consideremos el siguiente proceso, fijar $L = \frac{1}{m} p([n])$ la carga promedio de la instancia completa. Si un trabajo j cumple $p_j \geq L$, entonces removemos j y una máquina de la instancia y recalculamos L. Se repite hasta que todos los tarbajos restantes cumplan $p_j < L$. Diremos que un trabajo es *gigante* si fue removido en el proceso descrito. Notar que, por ejemplo, en una instancia con trabajos geometricamente relacionados, pueden haber más gigantes que los que fueron marcados en la primera etapa.

El siguiente resultado fu probado por Alon et al. [2] para el problema de minimizar la suma de una función convexa de la carga, pero análogamente se prueba para el caso de maximización con una función cóncava.

Proposición 2.18 (Alon et al. [2])**.** *Existe una asignación óptima que asigna todos sus trabajos gigantes solos en una máquina cada uno.*

Denotemos ℓ la cantidad de trabajos gigantes (y máquinas) removidos en el proceso descrito. Si se llegan a remover $m-1$ trabajos y máquinas sin detenerse, entonces la instancia restante tiene una sola máquina, y la carga promedio es la suma de las cargas. Por esto, si queda más de un trabajo, el proceso se detiene. Esto dice que $n > m$ implica $\ell < m$. Si por el contrario, $n \leq m$, se puede probar inductivamente que en cada etapa al menos se remueve el trabajo más largo, terminando por remover todos los trabajos. Así, en tal caso $\ell = n$.

En lo que sigue, denotaremos $\tilde{p}$ la carga promedio en la instancia restante al final del proceso, en otras palabras, la carga promedio de los $n - \ell$ trabajos no gigantes en $m - \ell$ máquinas, o 0 si todos los trabajos son gigantes. Como en cada etapa del proceso se borran trabajos de largo mayor al promedio, se puede probar que los trabajos gigantes son exactamente aquellos cuyo largo es al menos $\tilde{p}$.

Si bien la noción de trabajo gigante y del valor $\tilde{p}$ podría ser considerada en un contexto online, siendo redefinidos en cada paso del algoritmo, en esta tesis solo lo consideraremos estático con respecto a la instancia completa.

El siguiente corolario mejora la cota del valor óptimo con respecto a la Cota de la Carga Promedio.

Corolario 2.19. *Se tiene que* $\text{opt} \leq \sum_{j \text{ gigante}} f(p_j) + (m-\ell)f(\tilde{p})$.

Demostración. Por la Proposición 2.18, existe un óptimo en el que cada gigante esta solo en una máquina. Por simplicidad, ordenaremos las máquinas de tal manera que las primeras ℓ máquinas sean las con trabajos gigantes. Luego, en las $m-\ell$ máquinas restantes la asignación de $\{j \in [n] \mid j \text{ no gigante}\}$ también debe ser óptima, y su carga promedio es precisamente $\tilde{p}$, luego por la Cota de la Carga Promedio: $\sum_{i>\ell} f(p(\text{OPT}_i)) \leq (m-\ell)f(\tilde{p})$. Por lo tanto:

$$\text{opt} = \sum_{j \text{ gigante}} f(p_j) + \sum_{i>\ell} f(p(\text{OPT}_i)) \leq \sum_{j \text{ gigante}} f(p_j) + (m-\ell)f(\tilde{p}). \qquad \square$$

2.5. Relación con *Submodular Welfare*

Machine Productivity, además de pertenecer a la familia de problemas de agendamiento, es un caso particular de un problema ampliamente estudiado, *Submodular Welfare*, el cual es a su vez un caso particular de *Submodular Matroid Maximization* [7].

Submodular Matroid Maximization considera una matroide $(X, \mathcal{I})$ y una función $f : \mathcal{P}(X) \to \mathbb{R}_+$ submodular monótona normalizada y busca un conjunto independiente que maximice dicha función.

Dado un conjunto de n objetos, m postores tales que el postor i tiene función de utilidad $u_i : \mathcal{P}([n]) \to \mathbb{R}_+$ monótonas, submodulares y normalizadas, el problema *Submodular Welfare* consiste en buscar una asignación de los objetos a los postores de tal manera que la suma de las utilidades sea máxima. Donde las funciones de utilidad se suponen representables en tamaño y calculables en tiempo polinomial en n y m, o funciones cualesquiera accesadas mediante oráculos de valor [20]. Consideremos su problema de decisión:

$$\textsc{SubWelfare} = \{\langle m, n, \{u_i\}_{i\in[m]}, k\rangle \mid \exists \{J_i\}_{i\in[m]} \text{ partición de } [n], \sum_{i\in[m]} u_i(J_i) \geq k\}$$

Proposición 2.20. MachineProd *se reduce a* SubWelfare *en tiempo polinomial.*

Demostración. Sea $\langle m, n, p, f, k\rangle$ una instancia de MachineProd, definimos $\forall i \in [m], u_i = f \circ p$, la cual, por el Lema 2.15, es submodular, monótona y normalizada, con lo que $\langle m, n, \{u_i\}_{i\in[m]}, k\rangle$ es una instancia de SubWelfare. Esto es efectivamente una reducción pues toda asignación con productividad al menos k tiene utilidad al menos k y viceversa. Además un oráculo de valor para f puede simular en tiempo lineal un oráculo para u_i, pues basta calcular previamente $p(J)$. $\square$

Observación 2.21. Notar que la misma reducción nos asegura que toda α-aproximación de *Submodular Welfare* es una α-aproximación de *Machine Productivity.*

Con esto heredaremos rápidamente los resultados positivos de dicho problema, que fueron demostrados para el problema general *Submodular Matroid Maximization.*

Capítulo 3

La Regla *List Scheduling*

3.1. Resultados Relacionados

Algoritmos glotones aparecen en diversos problemas de optimización. La característica que tienen en común es que van por etapas, tomando la mejor decisión posible en cada una solo dependiendo de sus decisiones pasadas. Esto es, no prevén que sus decisiones puedan llegar arruinar la calidad de la aproximación. De ahí su nombre en inglés *greedy*, que corresponde a *avaro*, pero en esta tesis se usa el término *glotón*.

A continuación haremos referencia a los resultados conocidos para *Submodular Matroid Maximization* y *Submodular Welfare* con respecto a sus algoritmos glotones. Por la Observación 2.21, podemos heredar los resultados positivos a *Machine Productivity*.

Teorema 3.1 (Fisher-Nemhauser-Wolsey [11])**.** Offline Greedy *es una* 1/2*-aproximación para* Submodular Matroid Maximization*. Este analisis es ajustado.*

Si bien el algoritmo de Fisher, Nemhauser y Wolsey se puede aplicar a la versión offline de *Submodular Welfare*, no se puede aplicar a la versión online donde los trabajos llegan uno a uno. Lehman-Lehman-Nisan [21] demuestran que una variante online de este algoritmo es 1/2-competitivo. Esta variante revisa los objetos en un orden predeterminado y va asignándolos iterativamente al postor que tiene una mayor ganancia marginal de utilidad. Su implementación se describe en Algoritmo 1.

Denotamos ALG_i^j los objetos asignados al postor i inmediatamente después de asignar el objeto j.

Teorema 3.2 (Lehmann-Lehmann-Nisan [21])**.** Online Greedy *es una* 1/2*-competitivo para online* Submodular Welfare*. Este analisis es ajustado.*

Demostración. La siguiente demostración no es la presentada originalmente, la cual usa inducción. Aquí lo probamos directamente. Como u_i es submodular monótona:

$$\mathrm{opt} - \mathrm{alg} = \sum_{i\in[m]} u_i(\mathrm{OPT}_i) - u_i(\mathrm{ALG}_i) \leq \sum_{i\in[m]} \sum_{j\in \mathrm{OPT}_i \setminus \mathrm{ALG}_i} u_i(j \mid \mathrm{ALG}_i).$$

Algoritmo 1: *Online Greedy* para *Submodular Welfare*.

Entrada: $\langle m, n, \{u_i\}_{i\in[m]}\rangle$ donde $m, n \in \mathbb{N}$ cantidad de postores (máquinas) y objetos (trabajos), $\forall i \in [m], u_i : [n] \to \mathbb{Q}_+$ función de utilidad (productividad) del postor i.

Salida: $\text{ALG} = \{\text{ALG}_i\}_{i\in[m]}$ asignaciones de objetos a postores.

```
ALG ← {∅}_{i∈[m]}
para j ∈ [n] hacer
    i ← arg max(u_i(j | ALG_i))
    ALG_i ← ALG_i ∪ {j}
fin
devolver {ALG_i}_{i∈[m]}
```

Como u_i es submodular, es a marginales decrecientes y $\forall j \in [n], \forall i \in [m], \text{ALG}_i^{j-1} \subseteq \text{ALG}_i$. Así, la última suma, para $i \in [m]$ fijo, queda:

$$\sum_{j\in\text{OPT}_i\setminus\text{ALG}_i} u_i(j \mid \text{ALG}_i) \leq \sum_{j\in\text{OPT}_i} u_i(j \mid \text{ALG}_i) \leq \sum_{j\in\text{OPT}_i} u_i(j \mid \text{ALG}_i^{j-1}).$$

Para $j \in [n]$, sea $i_j \in [m]$ el postor tal que $j \in \text{ALG}_{i_j}$. Usando la propiedad del algoritmo glotón, sabemos que el postor i_j es el que maximiza la ganancia marginal con respecto a ALG^{j-1}, así $u_i(j \mid \text{ALG}_i^{j-1}) \leq u_{i_j}(j \mid \text{ALG}_{i_j}^{j-1})$. Retomando la expresión inicial, y notando que ahora la segunda suma no depende de i y recordando que OPT es una partición de $[n]$:

$$\text{opt} - \text{alg} \leq \sum_{i\in[m]} \sum_{j\in\text{OPT}_i} u_{i_j}(j \mid \text{ALG}_{i_j}^{j-1}) = \sum_{j\in[n]} u_{i_j}(j \mid \text{ALG}_{i_j}^{j-1}).$$

Llamemos alg^j al valor de la asignación inmediatamente depués de asignar el trabajo j. Notamos que $u_{i_j}(j \mid \text{ALG}_{i_j}^{j-1}) = \text{alg}^j - \text{alg}^{j-1}$, obtenemos una suma telescópica de extremos $\text{alg}^n = \text{alg}$ y $\text{alg}^0 = 0$ pues u_i está normalizada. Así $\text{opt} - \text{alg} = \text{alg}$, entonces $\text{opt} \leq 2\text{alg}$, o equivalentemente $\frac{1}{2}\text{opt} \leq \text{alg}$. □

Por otro lado, en problemas de agendamiento y sobre todo de balanceo de carga, el algoritmo glotón más simple es *List Scheduling* (*LS*), el cual asigna iterativamente trabajos uno a uno a la máquina menos cargada. Esto se presenta en más detalle en el Algoritmo 2.

Algoritmo 2: *List Scheduling*.

Entrada: $\langle m, n, p, f\rangle$ donde $m, n \in \mathbb{N}$ cantidad de máquinas y trabajos, $p : [n] \to \mathbb{Q}_+$ tiempos de proceso y $f : \mathbb{Q}_+ \to \mathbb{Q}_+$ función de productividad de las máquinas.

Salida: $\text{ALG} = \{\text{ALG}_i\}_{i\in[m]}$ asignaciones de trabajos a máquinas.

```
ALG ← {∅}_{i∈[m]}
para j ∈ [n] hacer
    i ← arg min(p(ALG_i))
    ALG_i ← ALG_i ∪ {j}
fin
devolver {ALG_i}_{i∈[m]}
```

La siguiente proposición nos dice que para *Machine Productivity*, como las máquinas son idénticas, *Online Greedy* equivale a usar la regla *LS*. Esta observación es importante pues la implementación de este último es mucho más simple. Requiere de menos cálculos

para actualizar los valores considerados en el máximo y no necesita invocar al oráculo de la función de productividad.

Proposición 3.3. *Sea A una asignación de $[j-1]$, sea $\underline{i}$ la máquina menos cargada. Entonces $\underline{i}$ es la máquina de máximo aumento de productividad al asignar j.*

Demostración. Sea $i \in [m]$, por definición $p(A_{\underline{i}}) \leq p(A_i)$. Usando el Corolario de Marginales Decrecientes, como $A_i, A_{\underline{i}} \subseteq [j-1]$, se concluye que: $f \circ p(j \mid A_{\underline{i}}) \geq f \circ p(j \mid A_i)$. □

Corolario 3.4. LS *es una* $1/2$*-competitivo para online* Machine Productivity.

Notar que para obtener este resultado solo se usa que $f \circ p$ es submodular monótona normalizada, ignorando las hipótesis más fuertes que tenemos sobre f de concavidad, y sobre todo, que las máquinas sean idénticas. En la siguiente sección haremos un uso intensivo de estas hipótesis para mejorar esta garantía.

Cabe mencionar otro algoritmo glotón comúnmente utilizado en problemas de balanceo de carga es *Longest Processing Time First* (LPT), el cual es introducido y analizado en el Capítulo 5. Este algoritmo fue enunciado por primera vez junto a LS por Graham en [13].

Volviendo a *Submodular Matroid Maximization* y *Submodular Welfare*, en sus versiones *offline*, por el lado negativo, el siguiente resultado fue enunciado de manera condicional a $\mathbf{P} \neq \mathbf{NP}$ por primera vez en [20], luego probado de manera incondicional en [24].

Teorema 3.5 (Mirrokni-Schapira-Vondrák [24])**.** *Ningún algoritmo (determinista o aleatorio) es mejor que una* $(1-1/e)$*-aproximación para* Submodular Welfare.

Esta cota se basa en probar que un tal algoritmo requiere una cantidad exponencial de consultas de valor, fallando si no, con alta probabilidad.

Si bien el análisis de la cota de $1/2$ para los algoritmos glotones online y offline es ajustado, se ha logrado obtener mejores garantías usando variantes que son también, en esencia, algoritmos glotones. En concreto, en [7] se presenta un nuevo algoritmo aleatorio óptimo, que considera la extensión multilineal de la función original, aplicando un proceso denominado *Continuous Greedy*, para obtener una solución fraccionaria, para luego aplicarle la técnica de *Pipage Rounding* para obtener una solución discreta.

Teorema 3.6 (Calinescu et al. [7])**.** *Existe una* $1-1/e$*-aproximación aleatoria para* Submodular Matroid Maximization.

Corolario 3.7. *Existe una* $1-1/e$*-aproximación aleatoria para* Machine Productivity.

El proceso *Continuous Greedy* fue definido primero para *Submodular Welfare* en [26], resultado que motiva esta generalización a *Submodular Matroid Maximization*. Una manera de ver este proceso es como una partícula que se mueve en la dirección de mayor crecimiento. Por otro lado, la técnica de *Pipage Rounding* introducida en [1], consiste en un redondeo que preserva la factibilidad de la solución, que en este caso corresponde a ser un conjunto independiente de la matroide.

Por mucho tiempo no se ha sabido si se puede alcanzar la misma garantía de manera determinista. Ni siquiera se sabía si en este contexto se podía superar el factor 1/2 de *Greedy*, hasta que en [5] se logra superar ligeramente este resultado.

Teorema 3.8 (Buchbinder-Feldman-Garg [5])**.** *Existe una* 0,5008*-aproximación determinista para* Submodular Matroid Maximization*.*

Corolario 3.9. *Existe una* 0,5008*-aproximación determinista para* Machine Productivity*.*

El algoritmo de Buchbinder, Feldman y Garg consiste en una técnica de separación de bases junto a un algoritmo denominado *Random Residual Greedy* (*RRGreedy*). Este algoritmo originalmente introducido en [6], consiste en buscar múltiples elementos que maximizan el aporte marginal, para luego tomar uno de forma aleatoria. Para *Submodular Welfare*, *RRGreedy* es equivalente a aplicar *Greedy* en la versión *online* en orden aleatorio, y en [4] se analiza obteniendo un algoritmo 0,5096-competitivo en dicho contexto. El algoritmo puede ser desaleatorizado, convirtiéndose así en uno completamente determinista.

3.2. *List Scheduling* es 3/4-Competitivo

Ahora procederemos a exprimir todo el potencial de las funciones de productividad, logrando mejorar la cota para *LS* a 3/4. Más aún, probaremos que esta cota es ajustada, esto es, existen instancias donde el algoritmo da exactamente dicha fracción del óptimo. Denotaremos ALG_i^j los trabajos asignados a la máquina i inmediatamente después de asignar el trabajo j. Además, recordamos que los trabajos están enumerados en el orden en el cual aparecen de manera online de 1 a n.

Primero veamos que *LS*, si bien puede asignar un trabajo gigante sobre otros trabajos, no asignará nada más sobre este.

Proposición 3.10. *Si usando* LS *algún trabajo gigante es asignado a una máquina, entonces ningún otro trabajo se asigna a la misma.*

Demostración. En efecto, notamos que si $n \leq m$ es directo. Si $n > m$, sabemos que el número ℓ de trabajos gigantes satisface $\ell < m$. Sea j dicho trabajo gigante e i la máquina a la que fue asignado. Sea $j' > j$ un trabajo a ser asignado luego de j. Probemos que en ese momento alguna máquina tiene una carga estrictamente menor a la de la máquina i. Por contradicción, supongamos que todas las máquinas tienen, antes de asignar el trabajo j', una carga mayor o igual a $p(\text{ALG}_i) \geq p_j \geq \tilde{p}$, pues j es gigante. Por casos:

Si j' no es gigante, como no ha sido asignado, la masa de no gigantes asignada es menor a la total: $p(\{j \in [n] \mid j \text{ no gigante}\}) > p(\{j < j' \mid j \text{ no gigante}\})$. Notar que no puede haber más de ℓ máquinas con trabajos gigantes, por lo que al menos $m - \ell$ tienen carga de al menos $p(\text{ALG}_i) \geq \tilde{p}$ sin tener trabajos gigantes. Así: $p(\{j < j' \mid j \text{ no gigante}\}) \geq (m - \ell)\tilde{p}$. Pero por definición de $\tilde{p}$, esto último es $p(\{j \in [n] \mid j \text{ no gigante}\})$. Una contradicción.

Si j' es gigante, entonces a lo más hay $\ell - 1$ máquinas con trabajos gigantes, pues j' no ha sido asignado. Por lo que al menos $m - \ell + 1$ tienen carga de al menos $p(\text{ALG}_i) \geq \tilde{p}$ sin

tener trabajos gigantes. Así, la masa de no gigantes es de al menos $(m-\ell+1)\tilde{p} > (m-\ell)\tilde{p}$. Donde, igual que antes, este último por definición es la masa total de no gigantes.

Una contradicción en ambos casos, por lo que existe una máquina con carga menor estricta a la de la máquina i, y así LS no asignará $j' > j$ a i. Cabe destacar que entonces la asignación de la máquina i no cambia luego de asignar j. Así, $\mathrm{ALG}_i = \mathrm{ALG}_i^j$. □

Esto nos dice entonces que hay exactamente ℓ máquinas con trabajos un trabajo gigante cada una. Para facilitar la notación, reordenaremos las máquinas de manera que estas sean las ℓ primeras.

Sea k la cantidad de máquinas sin trabajos gigantes que tienen carga sobre $\tilde{p}$ al aplicar LS, formalmente:

$$k = |\{i \in [m] \mid p(\mathrm{ALG}_i) \geq \tilde{p}, \mathrm{ALG}_i \subseteq \{j \in [n] \mid j \text{ no gigante}\}\}|.$$

Por el reordenamiento, podemos decir también $k = |\{i > \ell \mid p(\mathrm{ALG}_i) \geq \tilde{p}\}|$. Nuevamente, por simplicidad, supondremos que estas máquinas son las siguientes k máquinas, es decir son exactamente las máquinas i tales que $\ell < i \leq \ell + k$. Con esto, las últimas $m - \ell - k$ máquinas no tienen gigantes y tienen una carga menor a $\tilde{p}$. Esto nos da una vista particular sobre el resultado de aplicar LS en una instancia, viéndose como ilustra la Figura 3.1.

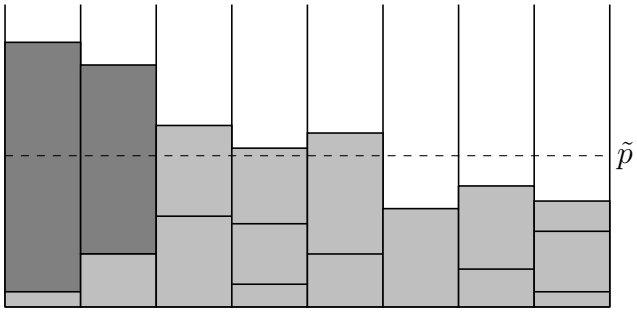

Figura 3.1: Resultado de aplicar LS a una instancia en $m = 8$ máquinas. Las máquinas ordenadas de izquierda a derecha según: $\ell = 2$ máquinas con un trabajo gigante (oscuros), $k = 3$ máquinas con carga mayor a $\tilde{p}$, y luego las 3 restantes con carga menor.

El siguiente lema nos da una cota inferior sobre la carga que se reparte entre las máquinas con carga menor a $\tilde{p}$.

Lema 3.11. *Se tiene que:*

$$\sum_{i>\ell+k} p(\mathrm{ALG}_i) \geq \frac{(m-\ell-k)^2}{m}\tilde{p}.$$

Demostración. Por contradicción, si no, entonces hay una máquina $\underline{i} > \ell + k$ con carga menor al promedio en estas máquinas $p(\mathrm{ALG}_{\underline{i}}) < \frac{m-\ell-k}{m}\tilde{p}$. Veremos que la masa total de no gigantes no calza.

Sea $i \leq \ell$, una máquina con un trabajo gigante j. Como j se asignó a i y no a $\underline{i}$, usando *LS*:

$$p(\mathrm{ALG}_i^{j-1}) \leq p(\mathrm{ALG}_{\underline{i}}^{j-1}) \leq p(\mathrm{ALG}_{\underline{i}}) < \frac{m-\ell-k}{m}\tilde{p}. \tag{3.1}$$

Donde recordemos que en ALG_i^{j-1} no hay gigantes pues j era el único gigante de i.

Sea i tal que $\ell < i \leq \ell + k$, ALG_i no es vacía, llamemos ahora j a su último trabajo, el cual no es gigante. Como j se agregó a i y no a $\underline{i}$ usando *LS*:

$$p(\mathrm{ALG}_i - j) = p(\mathrm{ALG}_i^{j-1}) \leq p(\mathrm{ALG}_{\underline{i}}^{j-1}) \leq p(\mathrm{ALG}_{\underline{i}}) < \frac{m-\ell-k}{m}\tilde{p}.$$

Como $p(\mathrm{ALG}_i - j) = p(\mathrm{ALG}_i) - p_j$, tenemos:

$$p(\mathrm{ALG}_i) < \frac{m-\ell-k}{m}\tilde{p} + p_j < \frac{m-\ell-k}{m}\tilde{p} + \tilde{p}. \tag{3.2}$$

Separemos la masa de no gigantes en las tres categorías según la máquina en la que se encuentran: máquina con gigante, con carga mayor y menor a la promedio. Juntando la hipótesis, (3.1) y (3.2), acotamos dicha masa:

$$p(\{j \in [n] \mid j \text{ no gigante}\}) < \ell\frac{m-\ell-k}{m}\tilde{p} + \left(k\frac{m-\ell-k}{m}\tilde{p} + k\tilde{p}\right) + \frac{(m-\ell-k)^2}{m}\tilde{p}.$$

Lo cual es exactamente $(m-\ell)\tilde{p}$, la masa total de no gigantes, una contradicción. □

Finalmente estamos en condiciones de demostrar la ansiada garantía de *LS*.

Teorema 3.12. LS *es* 3/4*-competitivo para online* Machine Productivity.

Demostración. Claramente si $n \leq m$, entonces *LS* asigna cada trabajo a una máquina distinta y así alg = opt. Si $n > m$, sabemos que $\ell < n$, y entonces no todos los trabajos son gigantes, por lo que $\tilde{p} \neq 0$.

Para las máquinas $i > \ell + k$ con carga menor a la promedio, por el Lema 3.11, y aplicando la Observación 2.16, obtenemos que:

$$\sum_{i>\ell+k} f(p(\mathrm{ALG}_i)) \geq \frac{1}{\tilde{p}}\left(\sum_{i>\ell+k} p(\mathrm{ALG}_i)\right) f(\tilde{p}) \geq \frac{(m-\ell-k)^2}{m} f(\tilde{p}).$$

Por otro lado, para una máquina i, tal que $\ell < i \leq \ell + k$, sobrecargada sin gigantes, como f es monótona $f(p(\mathrm{ALG}_i)) \geq f(\tilde{p})$. Sumándolas se obtiene:

$$\sum_{\ell<i\leq\ell+k} f(p(\mathrm{ALG}_i)) \geq k f(\tilde{p}).$$

Juntando las dos partes anteriores:

$$\sum_{i>\ell} f(p(\mathrm{ALG}_i)) \geq k f(\tilde{p}) + \frac{(m-\ell-k)^2}{m} f(\tilde{p}).$$

Factorizando, el término que acompaña a $f(\tilde{p})$ es $\frac{1}{m}(k^2-(m-2\ell)k+(m-\ell)^2)$, una función cuadrática convexa en la variable k. Su mínimo se encuentra en $k=\frac{m}{2}-\ell$ y vale $\frac{3}{4}m-\ell$. Así se concluye que:

$$\sum_{i>\ell} f(p(\mathrm{ALG}_i)) \geq \left(\frac{3}{4}m-\ell\right) f(\tilde{p}). \tag{3.3}$$

Por último, para $i \leq \ell$, como su carga es al menos la de su trabajo gigante, por monotonía de f y sumándolas se obtiene:

$$\sum_{i\leq\ell} f(p(\mathrm{ALG}_i)) \geq \sum_{j \text{ gigante}} f(p_j). \tag{3.4}$$

Además, todo gigante j cumple que $p_j \geq \tilde{p}$, entonces $f(p_j) \geq f(\tilde{p})$ y luego:

$$\sum_{j \text{ gigante}} f(p_j) \geq \ell f(\tilde{p}). \tag{3.5}$$

Juntando (3.3) y (3.4) podemos estimar el resultado de nuestro algoritmo.

$$\mathrm{alg} \geq \sum_{j \text{ gigante}} f(p_j) + \left(\frac{3}{4}m-\ell\right) f(\tilde{p}) = \sum_{j \text{ gigante}} f(p_j) - \ell f(\tilde{p}) + \frac{3}{4}m f(\tilde{p}).$$

Usando la cota sobre el óptimo del Corolario 2.19.

$$\mathrm{opt} \leq \sum_{j \text{ gigante}} f(p_j) + (m-\ell) f(\tilde{p}) = \sum_{j \text{ gigante}} f(p_j) - \ell f(\tilde{p}) + m f(\tilde{p}).$$

Así, concluimos que $\mathrm{alg} \geq \frac{3}{4}\mathrm{opt}$. □

Notar que la cota inferior sobre alg toma $k=\frac{m}{2}-\ell$, pero si m es impar, o si $\ell > \frac{m}{2}$ no se puede tomar tal k, y la garantía es mayor. Con ℓ como parámetro se pueden obtener más cotas, pero aquí solo nos interesamos a ajustar el análisis a los parámetros de entrada como m o n.

Teorema 3.13. *Si m es impar, entonces* LS *es $(3/4 + 1/(4m^2))$-competitivo para online* Machine Productivity*.*

Demostración. En efecto, en tal caso el mínimo de la función cuadrática $\frac{1}{m}(k^2-(m-2\ell)k+(m-\ell)^2)$ sobre los valores k enteros se alcanza en $k=\frac{m\pm1}{2}-\ell$, con un valor de $\frac{1}{4m}+\frac{3}{4}m-\ell$. Entonces el algoritmo nos asegura:

$$\mathrm{alg} \geq \sum_{j \text{ gigante}} f(p_j) - \ell f(\tilde{p}) + \left(\frac{3}{4}m+\frac{1}{4m}\right) f(\tilde{p}).$$

Procediendo de la misma manera que en el teorema se concluye que $\mathrm{alg} \geq \left(\frac{3}{4}+\frac{1}{4m^2}\right)\mathrm{opt}$. Lo cual es mayor, pero asintóticamente igual a 3/4. □

Teorema 3.14. *Los análisis anteriores son ajustados* $\forall m \in \mathbb{N}$*, es decir, si* m *es par* LS *es exactamente* $3/4$*-competitivo y si* m *es impar,* LS *es exactamente* $(3/4+1/(4m^2))$*-competitivo.*

Demostración. Consideremos la función de productividad $f(x) = \min(x, 2)$. Veamoslo según la paridad de m. Primero para m par: Consideremos $\frac{3}{2}m$ trabajos de largos: $p_j = 1$ para $j \leq m$, y $p_j = 2$ para $m < j \leq \frac{3}{2}m$. Entonces LS asigna primero cada trabajo $j \leq m$ solo a una máquina y luego los trabajos $j > m$ cada uno a una máquina distinta, sin pérdida de generalidad lo hace a las primeras $m/2$ máquinas. Obteniendo un valor de:

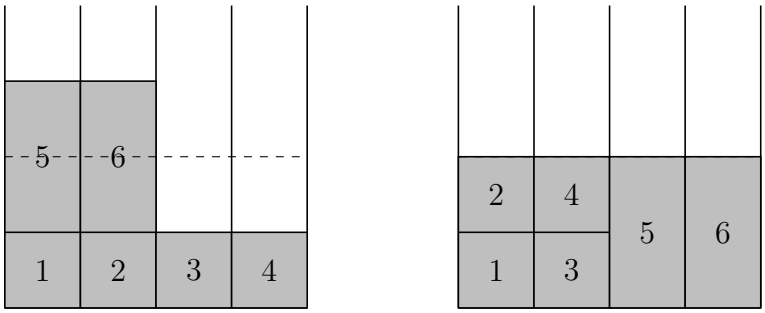

Figura 3.2: Resultado de aplicar LS (izquierda) y una asignación óptima (derecha) para $m = 4$ máquinas.

$$\text{alg} = \frac{m}{2}f(3) + \frac{m}{2}f(1) = \frac{3}{2}m.$$

Y una asignación óptima es asignar los primeros $j \leq m$ trabajos de a dos en $m/2$ máquinas y luego los trabajos $j > m$ solos en una máquina cada uno, obteniendo $\text{opt} = mf(2) = 2m$. Por lo tanto $\text{alg} = \frac{3}{4}\text{opt}$.

Segundo, si m impar: Consideremos $m(m-1) + \frac{m+1}{2}$ trabajos de largos: $p_j = 1/m$ para $j \leq m(m-1)$, y $p_j = 2$ para $m(m-1) < j \leq m(m-1) + \frac{m+1}{2}$. Entonces LS asigna primero los trabajos $j \leq m(m-1)$ uniformemente, $m-1$ a cada máquina, y luego los trabajos $j > m(m-1)$ cada uno a una máquina distinta, sin pérdida de generalidad lo hace a las primeras $\frac{m+1}{2}$ máquinas. Obteniendo un valor de:

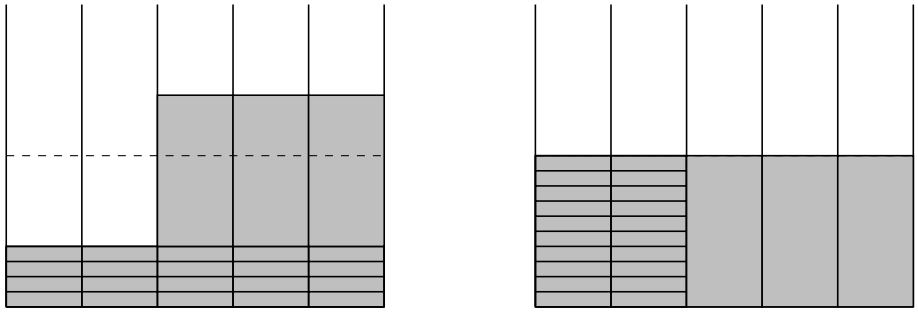

Figura 3.3: Asignación resultante de aplicar LS (izquierda) y una asignación óptima (derecha) para $m = 5$ máquinas.

$$\text{alg} = \frac{m+1}{2} f\left(2 + \frac{m-1}{m}\right) + \frac{m-1}{2} f\left(\frac{m-1}{m}\right) = \frac{3}{2}m + \frac{1}{2m}.$$

Y una asignación óptima, es asignar $j > m(m-1)$ solos en una máquina cada uno y luego como $m-1$ es par, es asignar los trabajos $j \leq m(m-1)$ de a $2m$ en las $\frac{m-1}{2}$ máquinas restantes, obteniendo $\text{opt} = mf(2) = 2m$. Por lo tanto $\text{alg} = \left(\frac{3}{4} + \frac{1}{4m^2}\right)\text{opt}$. □

Capítulo 4

Cotas Superiores para la Versión Online

4.1. Resultados Relacionados

En un contexto online adversarial, el mejor resultado heredable sigue siendo el factor 1/2 de la competitividad de los algoritmos glotones para *Submodular Welfare* y *Submodular Matroid Maximization*. Más aún, a través de una reducción a *Max k-Cover*, Kapralov-Post-Vondrák [17] prueban que esto es óptimo para *Submodular Welfare*, lo cual también implica la optimalidad para *Submodular Matroid Maximization*.

Teorema 4.1 (Kapralov-Post-Vondrák [17])**.** *Ningún algoritmo online (determinista o aleatorio) es mejor que* 1/2*-competitivo para* Submodular Welfare *en orden adversarial, salvo que* **NP** = **RP**.

Por lo visto en el capítulo anterior, sabemos que es posible superar este factor 1/2 de competitividad en online *Machine Productivity*, al menos hasta 3/4 gracias a *LS*. Nuestro interés se fija entonces en saber cuanto más puede garantizar algún algoritmo. En otras palabras, para cada contexto, encontrar una cota que ningún algoritmo pueda sobrepasar, y definir un algoritmo que la alcance.

4.2. Cotas Superiores Simples

Denotaremos por $\phi = (1 + \sqrt{5})/2$ al número de oro. A continuación damos una cota superior de $\phi/2 = (1+\sqrt{5})/4$, aproximadamente 0,809, para la competitividad de algoritmos deterministas para *Machine Productivity*. Si bien esto no cierra el problema, conseguimos al menos un gap constante para la respuesta final para la competitividad de un algoritmo óptimo para el problema. Este debe estar entre 0,75 y 0,809.

Teorema 4.2. *Ningún algoritmo online determinista es mejor que* $\phi/2$*-competitivo para* Machine Productivity *en orden adversarial. Donde* $\phi = (1 + \sqrt{5})/2$ *es el número de oro.*

Demostración. Consideremos las siguientes instancias en dos máquinas, con tres trabajos y función de productividad $f(x) = \text{mín}(x, \phi)$. Sea I_1 una instancia trabajos de largos $(1, 1, 0)$ e I_2 una instancia trabajos de largos $(1, 1, \phi)$. Por construcción, estas instancias son indistinguibles hasta que llega el tercer trabajo. Consideremos un algoritmo que asigna los trabajos 1 y 2 juntos a la misma máquina. En I_1, obtiene una productividad de ϕ. Mientras que el óptimo es de 2, y se obtiene al asignar los dos primeros trabajos a máquinas distintas. Así, el algoritmo puede tener una competitividad de a lo más $\phi/2$. La Figura 4.1 ilustra este caso.

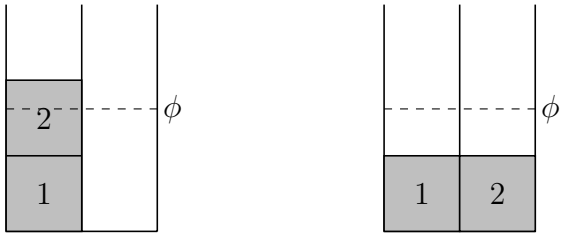

Figura 4.1: Resultado en I_1 de asignar ambos trabajos a la misma máquina (izquierda) y su asignación óptima (derecha).

Consideremos ahora un algoritmo que asigna los trabajos 1 y 2 a máquinas distintas. En I_2, debe asignar el trabajo 3 a alguna máquina, obteniendo una productividad de $1 + \phi = \phi^2$, donde se usó que ϕ es el número de oro. Pero la asignación óptima tiene a los trabajos 1 y 2 en la misma máquina y el trabajo 3 en la otra, obteniendo 2ϕ. Entonces, el algoritmo puede tener una competitividad de a lo más $\phi/2$. La Figura 4.2 ilustra este caso.

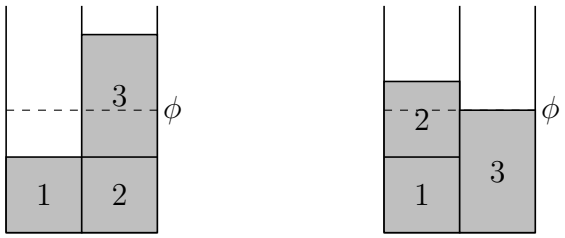

Figura 4.2: Resultado en I_2 de asignar 1 y 2 a máquinas distintas (izquierda) y su asignación óptima (derecha).

Como todo algoritmo determinista debe asignar o bien juntos o bien separados los dos primeros trabajos, se concluye. □

Si se permite aleatoriedad, la cota obtenida es un tanto mayor.

Teorema 4.3. *Ningún algoritmo online (ni siquiera aleatorio) es mejor que* 9/10*-competitivo para* Machine Productivity *en orden adversarial.*

Demostración. Consideremos las siguientes instancias en dos máquinas con 3 trabajos cada una, y función de productividad $f(x) = \text{mín}(x, 3)$. I_1 recibe los trabajos dados por la secuencia $(2, 2, 0)$, de valor óptimo $\text{opt}_1 = 4$ obtenido asignando los dos primeros trabajos por separado. I_2 recibe $(2, 2, 3)$, de óptimo $\text{opt}_2 = 6$ obtenido asignando los dos primeros trabajos juntos

y el tercero en la otra máquina. Por construcción, estas instancias son indistinguibles hasta que el tercer trabajo llega. Sea p la probabilidad con la que un algoritmo A asigna los dos primeros trabajos juntos. La esperanza de la competitividad del algoritmo A en I_1 es de a lo más $(p \cdot 3 + (1-p) \cdot 4)/\text{opt}_1 = p \cdot 3/4 + (1-p)$. Mientras que en I_2 es de a lo más $(p \cdot 6 + (1-p) \cdot 5)/\text{opt}_2 = p + (1-p) \cdot 5/6$.

Así, la competitividad esperada de cualquier algoritmo es de a lo más $\max_{p \in [0,1]} \min(p \cdot 3/4 + (1-p), p + (1-p) \cdot 5/6)$. Esta expresión se maximiza en $p = 2/5$, con valor de $9/10$. □

4.3. Cotas Superiores *Return-Oblivious*

En esta sección presentamos dos cotas de interés en el contexto de algoritmos en un modelo online adversarial. Ambas aplican sobre todo algoritmo que sea *return-oblivious*, es decir, que no evalúe ni conozca la función de productividad. Si bien esta categoría parece muy restrictiva, *LS* cae dentro de la categoría de algoritmos deterministas *return-oblivious*. La primera cota aplica solo sobre algoritmos deterministas, mientras que la segunda cota aplica al caso general, incluyendo algoritmos aleatorios.

Teorema 4.4. *Ningún algoritmo online determinista y return-oblivious es mejor que* $3/4$*-competitivo para* Machine Productivity *en orden adversarial. En particular,* LS *es un algoritmo óptimo en esta clase.*

Demostración. Consideremos las siguientes instancias en dos máquinas. Sea I_1 una instancia con función de productividad $f_1(x) = \min(1, x)$ y trabajos de largos $(1, 1, 0)$. La productividad óptima es de $\text{opt}_1 = 2$, obtenida al colocar los dos primeros trabajos por separado. Sea I_2 una instancia con función de productividad $f_2(x) = \min(2, x)$ y trabajos de largos $(1, 1, 2)$. La productividad óptima es de $\text{opt}_1 = 4$, obtenida al colocar los dos primeros trabajos juntos en una máquina y el tercero en la otra.

Por construcción, estas instancias son indistinguibles para cualquier algoritmo return-oblivious hasta que llega el tercer trabajo. Sea A un algoritmo return-oblivious. Si coloca los dos primeros trabajos juntos, logra un valor de $\text{alg} = 1$ en la instancia I_1, por lo que $\text{alg}/\text{opt} = 1/2$. Si coloca los dos primeros trabajos por separado, logra un valor de $\text{alg} = 3$ en la instancia I_2, por lo que $\text{alg}/\text{opt} = 3/4$. Dado que cualquier algoritmo determinista debe asignar los dos primeros trabajos juntos o por separado, concluimos que su razón competitiva es a lo sumo $3/4$. □

Teorema 4.5. *Ningún algoritmo online return-oblivious (ni siquiera aleatorio) es mejor que* $5/6$*-competitivo para* Machine Productivity *en orden adversarial.*

Demostración. Retomemos las instancias I_1 e I_2 del Teorema 4.4. Igual que antes, estas instancias son indistinguibles para cualquier algoritmo return-oblivious hasta que llega el tercer trabajo. Sea A un algoritmo return-oblivious y p la probabilidad de que ponga los dos primeros trabajos juntos. La competitividad de A en I_1 es $(p \cdot 1 + (1-p) \cdot 2)/\text{opt}_1 =$

$p\cdot(1/2)+(1-p)$. Y en I_2 es $(p\cdot 4+(1-p)\cdot 3)/\text{opt}_2 = p+(1-p)\cdot 3/4$. Luego, la competitividad de cualquier algoritmo es a lo más:

$$\max_{p\in[0,1]} \min(p\cdot 1/2 + (1-p), p + (1-p)\cdot 3/4).$$

La cual se maximiza en $p = 1/3$, con un valor de $5/6$. □

4.4. Algoritmo Óptimo en Dos Máquinas

Recordemos que denotamos por $\phi = (1+\sqrt{5})/2$ al número de oro. Como primer paso en búsqueda del mejor algoritmo online, presentamos el Algoritmo *Balanceo Áureo*. En esencia, el algoritmo busca estar lo más desbalanceado que pueda siempre y cuando esté seguro de ser una $\phi/2$-aproximación. Formalmente, se define como sigue: cuando llega el trabajo j, sean $\bar{i}$ e $\underline{i}$ la máquina más y menos cargada respectivamente. Sea $\overline{p}_j$ la carga promedio de una asignación que consista en los trabajos revelados, es decir, hasta el trabajo j incluyéndolo. Si $f(p(\text{ALG}_{\bar{i}} + j)) + f(p(\text{ALG}_{\underline{i}})) \geq \phi f(\overline{p}_j)$, entonces j se asigna a $\bar{i}$, y si no, a $\underline{i}$. Una implementación en detalle se encuentra en el Algoritmo 3. Este algoritmo alcanza la cota del Teorema 4.2 en el caso de dos máquinas, por lo que es óptimo en el contexto determinista en dos máquinas y en orden adversarial.

Algoritmo 3: *Balanceo Áureo.*

Entrada: $\langle m, n, p\rangle$ donde $m, n \in \mathbb{N}$ cantidad de máquinas y trabajos, $p : [n] \to \mathbb{Q}_+$ tiempos de proceso y $f : \mathbb{Q}_+ \to \mathbb{Q}_+$ función de productividad de las máquinas.
Salida: $\text{ALG} = \{\text{ALG}_i\}_{i\in[2]}$ asignaciones de trabajos a máquinas.

```
ALG ← {∅}_{i∈[2]}
para j ∈ [n] hacer
    Sea ī la máquina más cargada y i̲ la menos cargada.
    Sea p̄ la carga promedio de los trabajos revelados.
    si f(p(ALG_ī + j)) + f(p(ALG_i̲)) ≥ φf(p̄) entonces
        ALG_ī ← ALG_ī + j
    en otro caso
        ALG_i̲ ← ALG_i̲ + j
    fin
fin
devolver {ALG_i}_{i∈[m]}
```

Primero veamos dos lemas que nos ayudarán a justificar que sin importar el largo del trabajo que llegue, el algoritmo sigue siendo una buena aproximación.

Lema 4.6. *Sean $x, y, z \geq 0$ tales que $x \geq y$ y $f(x+y) \leq \phi f(x)$, entonces:*

$$z \geq x+y \implies \frac{f(x)+f(y+z)}{f(x+y)+f(z)} \geq \frac{\phi}{2}, \qquad x+y \geq z \geq x-y \implies \frac{f(x)+f(y+z)}{2f(\frac{x+y+z}{2})} \geq \frac{\phi}{2}.$$

Demostración. Consideremos primero el caso $z \geq x+y$. Como $y+z \geq z$, por monotonía de f y por la hipótesis sobre $f(x)$:

$$f(x)+f(y+z) \geq \frac{1}{\phi}f(x+y)+f(z) = \frac{1}{\phi}(f(x+y)+f(z)) + \left(1-\frac{1}{\phi}\right)f(z).$$

Como $z \geq x + y$, por monotonía $2f(z) \geq f(x + y) + f(z)$. Por lo tanto:

$$f(x) + f(y + z) \geq \left(\frac{1}{\phi} + \left(1 - \frac{1}{\phi}\right)\frac{1}{2}\right)(f(x + y) + f(z)).$$

Recordando que $\phi = 1 + 1/\phi$, se obtiene que el factor es exactamente $\phi/2$.

Veamos ahora el caso $x + y \geq z \geq x - y$. En particular notemos que a la vez:

$$\frac{x + y + z}{2} \leq \frac{x + y + x + y}{2} = x + y, \qquad \frac{x + y + z}{2} \leq \frac{z + y + y + z}{2} = y + z.$$

Y así, por la hipótesis y monotonía:

$$f(x) + f(y + z) \geq \frac{1}{\phi}f(x + y) + f(y + z) \geq \frac{1}{\phi}f\left(\frac{x + y + z}{2}\right) + f\left(\frac{x + y + z}{2}\right).$$

Factorizando esto último, como $\phi = 1 + 1/\phi$, se obtiene el factor de $\phi/2$ buscado. □

En el lema precedente se usa un análisis similar al que hemos usado en otros casos: Si llega una trabajo suficientemente grande, este será asignado solo en una máquina en algún óptimo. Si el trabajo no es tan grande basta compararnos con la cota de la carga promedio.

El único rango de tamaños que queda fuera del lema precedente es el caso $z \leq x - y$, el cual corresponde a la Figura 4.3. Informalmente, al ser una asignación más balanceada que la anterior, es esperable que tenga mayor valor incluso con respecto al nuevo óptimo. El siguiente lema formaliza esta intuición.

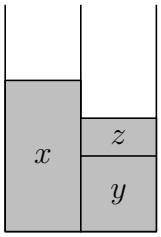

Figura 4.3: Situación en la que teniendo dos máquinas de cargas x e y, con $x \geq y$, se agrega la carga z sobre y tal que $z \leq x - y$.

Lema 4.7 (Mejora de Aproximación)**.** *Sea* ALG *una asignación de los trabajos* $[n]$ *en dos máquinas. Sean* $\bar{i}$ *y* $\underline{i}$ *sus máquinas más y menos cargada, respectivamente. Sea un trabajo* $j \in \text{ALG}_{\underline{i}}$*. Consideremos* ALG′ *la asignación* ALG *pero quitando el trabajo* j*. Si* ALG′ *es una* α*-aproximación, entonces* ALG *también es una* α*-aproximación.*

Demostración. Sea OPT una asignación óptima de $[n]$, de valor opt. Sean alg′ y alg el valor de ALG′ y de ALG, respectivamente. Afirmamos que existe una asignación óptima tal que $p(\text{ALG}_{\underline{i}}) \leq p(\text{OPT}_1)$ y $p(\text{ALG}_{\underline{i}}) \leq p(\text{OPT}_2)$. En efecto, si esta desigualdad no se tuviera para alguna máquina, sin pérdida de generalidad, para la máquina 1, entonces: $p(\text{OPT}_1) < p(\text{ALG}_{\underline{i}}) \leq p(\text{ALG}_{\bar{i}}) < p(\text{OPT}_2)$. Donde la última desigualdad se tiene pues la carga total

en OPT y en ALG es la misma, pues son asignaciones del mismo conjunto, $[n]$. Por el Lema de Intercambio Convexo, opt $\leq$ alg. En tal caso es óptima y cumple la propiedad buscada trivialmente.

Sin pérdida de generalidad supongamos que $j \in \mathrm{OPT}_1$. Por la justificación anterior, $p(\mathrm{ALG}_{\underline{i}}) \leq p(\mathrm{OPT}_1)$. Se tiene también que $p(\mathrm{ALG}'_{\underline{i}}) = p(\mathrm{ALG}_{\underline{i}}) - p_j \leq p(\mathrm{ALG}_{\underline{i}})$. Como además $p(\mathrm{OPT}_1 - j) + p(\mathrm{ALG}_{\underline{i}}) = p(\mathrm{OPT}_1 - j) + p_j + p(\mathrm{ALG}'_{\underline{i}}) = p(\mathrm{OPT}_1) + p(\mathrm{ALG}'_{\underline{i}})$, usando el Lema de Intercambio Convexo obtenemos que:

$$f(p(\mathrm{ALG}'_{\underline{i}})) + f(p(\mathrm{OPT}_1)) \leq f(p(\mathrm{ALG}_{\underline{i}})) + f(p(\mathrm{OPT}_1 - j)).$$

Equivalentemente:

$$f(p(\mathrm{OPT}_1)) - f(p(\mathrm{OPT}_1 - j)) \leq f(p(\mathrm{ALG}_{\underline{i}})) - f(p(\mathrm{ALG}'_{\underline{i}})). \tag{4.1}$$

Sea OPT' una asignación óptima de $[n] \setminus \{j\}$ y opt' su valor. Notemos que $\{\mathrm{OPT}_1 - j, \mathrm{OPT}_2\}$ es una asignación de $[n] \setminus \{j\}$. Entonces $f(p(\mathrm{OPT}_1 - j)) + f(p(\mathrm{OPT}_2)) \leq \mathrm{opt}'$, así:

$$\mathrm{opt} \leq \mathrm{opt}' + f(p(\mathrm{OPT}_1)) - f(p(\mathrm{OPT}_1 - j)).$$

Por otro lado, como la carga de la máquina $\overline{i}$ no cambia, $\mathrm{ALG}_{\overline{i}} = \mathrm{ALG}'_{\overline{i}}$, aplicando (4.1):

$$\mathrm{alg} = \mathrm{alg}' + f(p(\mathrm{ALG}_{\underline{i}})) - f(p(\mathrm{ALG}'_{\underline{i}})) \geq \mathrm{alg}' + f(p(\mathrm{OPT}_1)) - f(p(\mathrm{OPT}_1 - j)).$$

Soltando el término común no negativo: $\mathrm{alg}/\mathrm{opt} \geq \mathrm{alg}'/\mathrm{opt}' \geq \alpha$. □

Es fácil generalizar el lema precedente a agregar más de un trabajo por inducción.

Corolario 4.8. *Sea* ALG *una asignación de los trabajos* $[n]$ *en dos máquinas. Sean* $\overline{i}$ *y* $\underline{i}$ *sus máquinas más y menos cargada, respectivamente. Sean los trabajos* $K \subseteq \mathrm{ALG}_{\underline{i}}$*. Consideremos* ALG' *la asignación* ALG *pero quitando los trabajos* K*. Si* ALG' *es una* α*-aproximación, entonces* ALG *también es una* α*-aproximación.*

Demostración. Por inducción en $k = |K|$. En el caso base $|K| = 0$, y entonces $K = \emptyset$ y la propiedad se cumple por hipótesis. Suponemos que si $|K| = k - 1$, y ALG' es una α-aproximación también lo será ALG. Si $|K| = k$, sea $j \in K$ cualquiera, tenemos que $|K - j| = k - 1$ y $K - j \subseteq \mathrm{ALG}_{\underline{i}}$. Suponemos que ALG' es una α-aproximación. Sea ALG'' la asignación ALG pero quitando los trabajos $K - j$ (de la máquina menos cargada). Notar que ALG' se obtiene de ALG'' quitando j de la máquina menos cargada. Por el Lema de Mejora de Aproximación, ALG'' también es una α-aproximación. Luego, como por construcción ALG'' se obtiene de ALG quitando $k - 1$ trabajos de la máquina menos cargada, ALG también es una α-aproximación. □

Con esto, ya estamos en condiciones de probar la garantía sobre el Algoritmo 3.

Teorema 4.9. *Balanceo Áureo es* $\phi/2$*-competitivo para* Machine Productivity *en orden adversarial restringido a dos máquinas. Donde* $\phi = (1 + \sqrt{5})/2$ *es el número de oro.*

Demostración. Diremos que un trabajo es *bajo* si fue asignado a la máquina que en ese momento era la menos cargada. Sean $\overline{i}$ y $\underline{i}$ las máquinas más y menos cargada, respectivamente, justo antes de asignar el trabajo n. Notar que la carga promedio hasta la iteración n es la carga promedio final, $\overline{p}_n = \overline{p}$. Si n no es bajo, por la descripción del algoritmo: alg $\geq \phi f(\overline{p})$. Por la Cota de la Carga Promedio se tiene que: opt $\leq 2f(\overline{p})$, luego alg $\geq \frac{\phi}{2}$opt.

Si n es bajo, lo veremos por inducción en la cantidad k de trabajos bajos. Recordemos que denotamos ALG_i^j los trabajos asignados a la máquina i inmediatamente después de asignar el trabajo j. Cuando $k = 1$: Como el trabajo n es bajo, es el único bajo. Entonces, la máquina $\underline{i}$ estaba vacía antes de la última iteración y $\text{ALG}_{\underline{i}} = \{n\}$. Si $p_n \geq p(\text{ALG}_{\overline{i}})$, entonces n es un trabajo gigante y por la Observación 2.18 existe una asignación óptima que lo deja solo, la cual es precisamente ALG, luego alg = opt. Si $p_n < p(\text{ALG}_{\overline{i}})$, notamos que ALG^{n-1} es una asignación sin trabajos bajos. Por la justificación inicial, esta es una $\phi/2$-aproximación. Por el Lema de Mejora de Aproximación, ALG también lo es.

Para el paso inductivo, suponemos ahora que si hay $k-1$ trabajos bajos, el resultado es una $\phi/2$-aproximación. Probamos que esto se mantiene cuando un nuevo trabajo bajo n llega. Probémoslo para $k \geq 2$. Consideremos el caso $p_n \leq p(\text{ALG}_{\overline{i}}) - p(\text{ALG}_{\underline{i}}^{n-1})$. Por hipótesis inductiva ALG^{n-1} es una $\phi/2$-aproximación. Por el Lema de Mejora de Aproximación, ALG también.

Ahora vemos el caso $p_n \geq p(\text{ALG}_{\overline{i}}) - p(\text{ALG}_{\underline{i}}^{n-1})$. Sea $j \leq n-1$ el penúltimo (n es el último) trabajo bajo, y sea $\underline{r}$ la máquina a la que fue asignado. Sea $\overline{r}$ la otra máquina. Buscando aplicar el Lema 4.6, tomamos $x = p(\text{ALG}_{\overline{i}}^j) \geq p(\text{ALG}_{\underline{i}}^j) = y$ y $z = p_n$. Como entre las iteraciones j y n no se han asignado trabajos a $\underline{i}$, $p(\text{ALG}_{\underline{i}}^{n-1}) = p(\text{ALG}_{\underline{i}}^j) = y$. Como además $p(\text{ALG}_{\overline{i}}) \geq x$, también se tiene $z \geq x - y$. Notemos que $\{\text{ALG}_{\overline{r}}^{j-1} + j, \text{ALG}_{\underline{r}}^{j-1}\}$ es una asignación de $[j]$, luego su carga total es $p([j]) = x + y$. Por subaditividad de f, y como j es bajo:

$$f(x+y) \leq f(p(\text{ALG}_{\overline{r}}^{j-1} + j)) + f(p(\text{ALG}_{\underline{r}}^{j-1})) < \phi f\left(\frac{x+y}{2}\right).$$

Como $x \geq y$, entonces $x \geq \frac{x+y}{2}$. Por monotonía de f, concluimos que $f(x+y) \leq \phi f(x)$. Con esto, podemos aplicar el Lema 4.6, obteniendo que $A = \{\text{ALG}_{\overline{i}}^j, \text{ALG}_{\underline{i}}^j + n\} = \{\text{ALG}_{\overline{i}}^j, \text{ALG}_{\underline{i}}\}$, de productividad alg $= f(x) + f(y+z)$, es una $\phi/2$-aproximación con respecto a la instancia definida por los trabajos $[j] \cup \{n\}$. En efecto, veamos esta garantía por casos:

Si $z \geq x + y$, n es gigante, por la Observación 2.18, existe un óptimo que lo asigna solo, el cual tendría productividad opt $= f(x+y) + f(z)$. Así, el lema nos dice que alg/opt $\geq \phi/2$. Si $x + y \geq z$, como $x + y + z$ es la carga total de la instancia con $[j] \cup \{n\}$, por la Cota de la Carga Promedio opt $\leq 2f(\frac{x+y+z}{2})$. De nuevo el lema nos permite concluir alg/opt $\geq \phi/2$.

Resta ver que al completar A para obtener ALG esta garantía se mantiene. Considerando la parte faltante $K = \text{ALG}_{\overline{i}} - \text{ALG}_{\overline{i}}^j$, con lo cual $p(K) = p(\text{ALG}_{\overline{i}}) - p(\text{ALG}_{\overline{i}}^j)$. Como $p(\text{ALG}_{\underline{i}}) = p(\text{ALG}_{\underline{i}}^{n-1}) + p_n$, por la hipótesis sobre p_n, se tiene que $p(K) \leq p(\text{ALG}_{\underline{i}}) - p(\text{ALG}_{\overline{i}}^j)$. Por lo tanto, por el Corolario 4.8, ALG también es una $\phi/2$-aproximación, lo cual termina la inducción. □

4.5. Más Cotas Superiores

4.5.1. Cotas Superiores en Orden Aleatorio

En esta subsección cambiamos el modelo de llegada de los trabajos. En vez de suponer que los trabajos llegan en orden adversarial, se considera que los trabajos llegan en un orden elegido uniformemente al azar dentro de todas las permutaciones posibles. La competitividad esperada de un algoritmo ahora se mide como un promedio de la competitividad sobre todas las permutaciones posibles. Para los siguientes resultados, basta considerar instancias de solo 3 trabajos, donde dos de ellos son iguales, por lo que esencialmente solo importa la posición del trabajo distinto. Si el trabajo distinto llega primero o segundo, sin mucho que decir, el algoritmo podría ser óptimo y se acota la competitividad por 1. Si llega al final, podemos aprovechar resultados previos para obtener cotas sobre la competitividad.

De la misma instancia del Teorema 4.2 podemos sacar una cota para todo algoritmo determinista en orden aleatorio de aproximadamente 0,936.

Corolario 4.10. *Ningún algoritmo online determinista es mejor que* $(\frac{2}{3} + \frac{\phi}{6})$*-competitivo para* Machine Productivity *en orden aleatorio. Donde* $\phi = \frac{1+\sqrt{5}}{2}$ *es el número de oro.*

Demostración. Si el trabajo distinto llega último, según como haya asignado los dos trabajos iguales: Si los asignó separados en I_1 se obtiene una asignación óptima, pero en I_2 no, con competitividad de $\phi/2$. Si los asignó juntos en I_2 se puede obtener una asignación óptima, pero en I_1 no, con competitividad de $\phi/2$. Por otro lado, si el trabajo distinto llega primero o segundo, acotamos la competitividad por 1. Por lo tanto, la competitividad de cualquier algoritmo es de a lo más $1 \cdot 2/3 + \phi/2 \cdot 1/3 = 2/3 + \phi/6$. □

Como una garantía en orden adversarial es una cota inferior a la competitividad en cualquier orden, en particular *LS* tiene una garantía de al menos 3/4 en orden aleatorio. Notar que el análisis de que esta garantía era ajustada ya no corre, y conjeturamos que es mayor.

De manera análoga, a partir de la instancia del Teorema 4.3 podemos obtener otra cota cuando se permite aleatoriedad en el algoritmo, la cual es de aproximadamente 0,967

Corolario 4.11. *Ningún algoritmo online (ni siquiera aleatorio) es mejor que* 29/30*-competitivo para* Machine Productivity *en orden aleatorio.*

Demostración. Si el trabajo distinto llega último, sea p la probabilidad de que un algoritmo asigne los dos primeros trabajos juntos, como se vio en el Teorema 4.3, un algoritmo aleatorio puede asegurar a lo más 9/10 del óptimo en este caso. Por otro lado, si el trabajo distinto llega primero o segundo, acotamos la competitividad por 1. Por lo tanto, la competitividad de cualquier algoritmo es de a lo más $1 \cdot 2/3 + 9/10 \cdot 1/3 = 29/30$. □

4.5.2. Cotas Superiores para Más de Dos Máquinas

En busca de una generalización del Algoritmo 3, lo primero es saber siquiera si es posible obtener la misma garantía. Para ello se buscan cotas superiores para $m > 2$.

Lema 4.12. *Sea $m \geq 2$ entonces el polinomio $4x^m - 2x^{m-1} - 1$ tiene una única solución real entre $1/\sqrt[m-1]{2}$ y 1.*

Demostración. Llamemos $p(x)$ al polinomio, entonces evaluamos: $p(1) = 1 > 0$. Por el otro lado $p(1/\sqrt[m-1]{2}) = 2/\sqrt[m-1]{2} - 2 < 0$. Así, como un polinomio es una función continua, por el Teorema de los Valores Intermedios se concluye. □

Proposición 4.13. *Sea $m \geq 2$ y α la raíz del polinomio $4x^m - 2x^{m-1} - 1$ que cumple $1/\sqrt[m-1]{2} < \alpha < 1$. Ningún algoritmo online determinista es mejor que α-competitivo para* Machine Productivity *en orden adversarial en m máquinas.*

Demostración. Consideremos las instancias en $m \geq 2$ máquinas, con función de productividad $f(x) = \text{mín}(x, 1)$ y con $m + 1$ trabajos definidas por:

$$\forall j \in [m-1], I_j : p_1 = \frac{1}{2\alpha}, p_2 = \frac{1}{2\alpha}, \ldots, p_{j+1} = \frac{1}{2\alpha^j}, p_{j+2} = 0, \ldots, p_{m+1} = 0.$$

Y también:

$$I_m : p_1 = \frac{1}{2\alpha}, p_2 = \frac{1}{2\alpha}, \ldots, p_m = \frac{1}{2\alpha^{m-1}}, p_{m+1} = 1.$$

Notar que todos los trabajos $j \leq m$ tienen un tamaño menor a 1 y mayor a 1/2. En efecto, como $\alpha > 1/\sqrt[m-1]{2}$, entonces $\alpha^{m-1} > 1/2$, y así $p_m = 1/2\alpha^{m-1} < 1$. Como $\alpha < 1$, para $j \leq m$, $1/2 < p_j \leq p_m < 1$. Esto implica que un trabajo solo no llena la capacidad de una máquina, pero dos sí.

Probemos que si un algoritmo asigna dos trabajos de los m primeros en I_m a la misma máquina entonces es a lo más una α-aproximación. Procedemos por inducción en j el trabajo que se asigna en la misma máquina que otro, con $2 \leq j \leq m$. Para $j = 2$, es decir si el trabajo 2 se asigna a la misma máquina que el trabajo 1. Consideremos I_2, la cual tiene los dos primeros trabajos iguales a I_m, luego el algoritmo procede de la misma forma obteniendo un valor de alg = 1. Como $j \leq m$, el óptimo asigna ambos trabajos a máquinas distintas, obteniendo opt $= 1/2\alpha + 1/2\alpha = 1/\alpha$. Así alg $= \alpha$opt. Supongamos que si alguno de los trabajos hasta el $j - 1$ se asigna sobre otra máquina, el algoritmo es a lo más una α-aproximación. Para j trabajos: Si el algoritmo asigna algún trabajo hasta el $j - 1$ sobre otro, por hipótesis inductiva se concluye. Si los asigna todos a máquinas distintas, solo queda ver que pasa si asigna j sobre otro trabajo. Consideremos la instancia: I_j, que tiene exactamente los mismos j primeros trabajos y debe hacer la misma asignación. Sea $k < j$ el trabajo sobre el cual el algoritmo asigna el trabajo j. Como $j \leq m$, el óptimo asigna cada trabajo a una máquina distinta. Veamos la aproximación:

$$\text{alg} = \sum_{i=1}^{j-1} p_i - p_k + 1 \leq \sum_{i=1}^{j-1} p_i - p_1 + 1 = \sum_{i=2}^{j-1} p_i + 1.$$

Y se concluye que:

$$\text{opt} = \sum_{i=1}^{j} p_i = \frac{1}{2\alpha} + \frac{1}{2\alpha} + \sum_{i=3}^{j} \frac{1}{2\alpha^{i-1}} = \frac{1}{\alpha} + \sum_{i=2}^{j-1} \frac{1}{2\alpha^i} = \frac{1}{\alpha} + \frac{1}{\alpha} \sum_{i=2}^{j-1} \frac{1}{2\alpha^{i-1}} \geq \frac{1}{\alpha}\text{alg}.$$

Si se asignaran dos trabajos $j \leq m$ juntos.

Si cada trabajo $j \leq m$ se coloca en una máquina distinta, llega el $m+1$ y debe ser asignado sobre otro, digamos k. Notemos que una asignación óptima deja dos trabajos juntos, pues son $m+1$, y estos deben ser los más cortos. Así, una asignación óptima es tener a los trabajos 1 y 2 en la misma máquina, y los demás solos cada uno en una máquina.

$$\text{opt} = 1 + \sum_{i=3}^{m} p_i + 1 = 2 + \sum_{i=3}^{m} p_i.$$

Por otro lado, como α es raíz del polinomio $4x^m - 2x^{m-1} - 1$, se puede verificar que $2\alpha = 1 + 1/2\alpha^{m-1}$. Entonces:

$$\text{alg} = \sum_{i=1}^{m} p_i - p_k + 1 \leq \sum_{i=1}^{m} p_i - p_1 + 1 = \sum_{i=2}^{m} \frac{1}{2\alpha^{i-1}} + 1 = \sum_{i=2}^{m-1} \frac{1}{2\alpha^{i-1}} + 2\alpha.$$

Factorizando por α y acomodando la suma, se concluye que $\text{alg} \leq \alpha\text{opt}$. Como todo algoritmo determinista debe asignar dos trabajos en la misma máquina, se concluye la cota. □

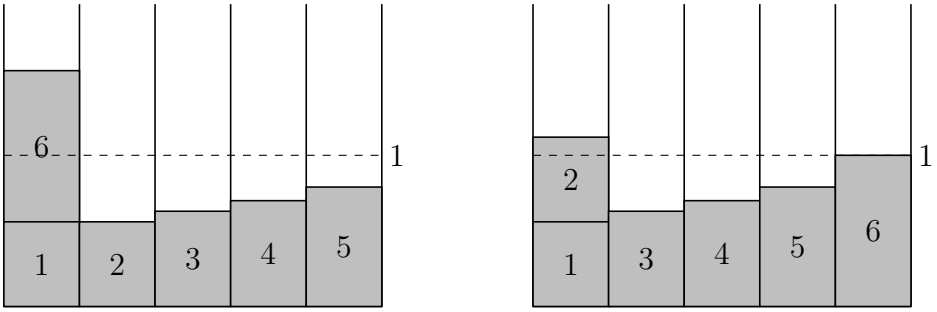

Figura 4.4: Asignación de un algoritmo habiendo asignado todos los primeros trabajos a máquinas distintas, y el último sobre el primer trabajo (izquierda) y una asignación óptima (derecha) con $m = 5$, para I_m.

En la Tabla 4.1 se exhiben valores explícitos para $m \leq 8$ de las cotas obtenidas.

Observación 4.14. Por un lado, para $m = 2$, se tiene que $\alpha = \phi/2$, recuperando la cota obtenida en el Teorema 4.2. Por el otro, si m tiende a infinito, como $1/\sqrt[m-1]{2}$ tiende a 1, también lo hace la cota α.

Esto nos hace pensar que la cota solo es interesante en el caso de pocas máquinas. La siguiente cota justifica esta intuición, pues otorga una cota constante de aproximadamente 0,9045 cuando la cantidad de máquinas m es par.

Proposición 4.15. *Ningún algoritmo online determinista es mejor que* $(\frac{1}{2} + \frac{\phi}{4})$*-competitivo para* Machine Productivity *en orden adversarial en una cantidad par de máquinas. Donde* $\phi = (1 + \sqrt{5})/2$ *es el número de oro.*

Demostración. Consideremos las instancias en m máquinas, con m par, con función de productividad $f(x) = \text{mín}(x, 1)$ y $\frac{3}{2}m$ trabajos definidas por:

$$I_1 : \forall j \leq m, p_j = \frac{1}{\phi}, \quad \forall j > m, p_j = 0, \qquad I_2 : \forall j \leq m, p_j = \frac{1}{\phi}, \quad \forall j > m, p_j = 1.$$

Sea $k \leq m/2$ la cantidad de trabajos dentro de los m primeros que un algoritmo asigna sobre otros trabajos.

En I_1, el óptimo puede dejar un trabajo en cada máquina, obteniendo $\text{opt} = \frac{1}{\phi}m$ y el algoritmo en el mejor caso deja de a pares los k trabajos, obteniendo $\text{alg} \leq k + (m - 2k)\frac{1}{\phi} = \frac{1}{\phi}(m - (2 - \phi)k)$. Así $\text{alg/opt} \leq 1 - (2 - \phi)\frac{k}{m}$, lo cual es decreciente en k.

En I_2, el óptimo puede dejar los primeros m trabajos de a pares, y el resto uno en cada máquina restante, así: $\text{opt} = m$ y el algoritmo en el mejor caso asigna de a pares los k trabajos y luego asigna trabajos de largo 1 en las máquinas vacías, para luego colocar los restantes en máquinas con un solo trabajo de largo $1/\phi$, obteniendo $\text{alg} = (2 - \phi)k + \frac{\phi}{2}m$. Así $\text{alg/opt} \leq (2 - \phi)\frac{k}{m} + \frac{\phi}{2}$, lo cual es creciente en k.

La garantía del algoritmo es entonces de a lo más:

$$c_k = \text{mín}\left(1 - (2 - \phi)\frac{k}{m}, (2 - \phi)\frac{k}{m} + \frac{\phi}{2}\right).$$

Cuyo máximo en k se encuentra cuando ambos términos son iguales, y esto pasa cuando $k = m/4$. Por lo tanto, una cota superior a la competitividad de cualquier algoritmo es $1/2 + \phi/4$. □

Observación 4.16. La demostración deja a lugar una mejor cota cuando m es par, pero no múltiplo de 4, pues en tal caso k no puede ser $m/4$. En tal caso, se acota por el valor con $k = \frac{m \pm 2}{4}$, obteniendo una cota de $1/2 + \phi/4 - (2 - \phi)/2m$. Lo cual es aún mejor, pero asintóticamente igual a $1/2 + \phi/4$.

En la Tabla 4.1 se resumen los dos tipos de cota obtenidos para más de dos máquinas. Se observa que ambas cotas coinciden con la inicial de $\phi/2$ para $m = 2$. Además, la segunda cota superior es mejor desde $m = 6$, para cantidades de máquinas m par. En particular, la segunda cota nos dice que asíntoticamente cuando m tiende a infinito, se tiene una cota superior de aproximadamente 0,905.

m	Proposición 4.13	Proposición 4.15
2	0,809	0,809
3	0,848	-
4	0,874	0,905
5	0,893	-
6	0,907	0,872
7	0,918	-
8	0,927	0,905
Múltiplo de 4	-	0,905
Limite a ∞	1	0,905

Tabla 4.1: Valores de las cotas superiores obtenidas para los casos entre 2 y 8 máquinas.

Capítulo 5

La Regla *Longest Processing Time First*

La regla *Longest Processing Time First* (LPT), al igual que LS, fue considerada por primera vez por Graham [13] para el problema *Makespan Minimization*. LPT es esencialmente el mismo algoritmo, pero ordena de mayor a menor según tiempo de proceso los trabajos antes de asignarlos. Para efectos del análisis del algoritmo, podemos suponer sin pérdida de generalidad que los trabajos están ya ordenados de dicha forma, de modo que LPT los asigna en orden del 1 al n. Este ordenamiento requiere conocer todos los trabajos al iniciar la ejecución, por lo que es un algoritmo que solo funciona en el contexto offline. En algunos problemas LPT logra estrictamente mejores garantías que LS. Por ejemplo, para *Makespan Minimization*, LS es una $(2-1/m)$-aproximación, mientras que LPT es una 4/3-aproximación [13].

5.1. Análisis de *LPT*

En esta sección demostramos que LPT logra al menos un factor de aproximación de $2(\sqrt{2}-1) \approx 0{,}828$ para *Machine Productivity*, lo cual es estrictamente mejor que el factor 3/4 de LS.

Sea ALG el resultado de aplicar LPT a una instancia dada. Supondremos que todos los trabajos tienen largos positivos.

Lema 5.1. LPT *deja solo a cada gigante en una máquina distinta.*

Demostración. Dado que $\ell \leq m$ y los gigantes son los ℓ trabajos más largos, estos son los primeros ℓ trabajos asignados. Como LPT es un caso particular de LS, por el Lema 3.10, ningún otro trabajo se asigna en estas máquinas, por lo que los gigantes quedan solos. □

Por el Corolario 2.18, existe un óptimo que hace lo mismo. Si obtenemos una cota para la instancia sin gigantes y con ℓ máquinas menos, por el Lema 5.1 el factor de la aproximación

será al menos el mismo.

Teorema 5.2. LPT *es una* $2(\sqrt{2}-1)$*-aproximación para* Machine Productivity.

Demostración. Suponemos que la instancia no tiene gigantes. Como en la demostración del Teorema 3.12, sea k la cantidad de máquinas (sin gigantes) que tienen una carga de al menos $\tilde{p} = \overline{p}$ la promedio, pues no hay gigantes. Asumimos además que estas son las primeras $i \leq k$ máquinas.

Si $n \leq m$, entonces se puede ver que todos los trabajos son gigantes, una contradicción. Así, consideramos $n \geq m+1$. Notemos que los primeros m trabajos se asignan uno a cada máquina. Así, la carga de cualquier máquina es al menos p_m, ya que cada máquina contiene al menos un trabajo con al menos ese largo.

Sea $\underline{i}$ la máquina menos cargada y llamemos $x = p(\mathrm{ALG}_{\underline{i}})/\overline{p} \in (0,1]$. Tenemos que:

$$\sum_{i=k+1}^{m} p(\mathrm{ALG}_i) \geq (m-k)p(\mathrm{ALG}_{\underline{i}}) = (m-k)x\overline{p}. \tag{5.1}$$

Sea $i \leq k$ una máquina con carga al menos $\overline{p}$. Dado que no hay gigantes, i debe tener al menos dos trabajos. Sea j el último trabajo asignado, entonces $j \geq m+1$, por lo que $p_j \leq p_m$. Dado que j se asignó a i y no a $\underline{i}$, tenemos $p(\mathrm{ALG}_i - j) \leq p(\mathrm{ALG}_{\underline{i}})$. Y como $p_j \leq p_m \leq p(\mathrm{ALG}_{\underline{i}})$, al sumar las desigualdades se deduce que $p(\mathrm{ALG}_i) \leq 2p(\mathrm{ALG}_{\underline{i}})$. Sumando todas estas máquinas, obtenemos $\sum_{i=1}^{k} p(\mathrm{ALG}_i) \leq 2kx\overline{p}$. Por lo tanto:

$$\sum_{i=k+1}^{m} p(\mathrm{ALG}_i) = m\overline{p} - \sum_{i=1}^{k} p(\mathrm{ALG}_i) \geq (m-2kx)\overline{p}. \tag{5.2}$$

Usando la Observación 2.16 y las cotas inferiores (5.1) y (5.2):

$$\sum_{i=k+1}^{m} f(p(\mathrm{ALG}_i)) \geq \mathrm{máx}((m-k)x, m-2kx)f(\overline{p}).$$

Dado que $f(p(\mathrm{ALG}_i)) \geq f(\overline{p})$ para todos los $i \leq k$, por monotonicidad, obtenemos una cota inferior en el valor del algoritmo alg:

$$\mathrm{alg} \geq kf(\overline{p}) + \mathrm{máx}((m-k)x, m-2kx)f(\overline{p}).$$

Factorizando el término común $f(\overline{p})$ en el lado derecho, obtenemos un factor que es mínimo en $k = \frac{1-x}{x}m$, que ocurre cuando ambos términos en el máximo son iguales. Por lo tanto, el valor del algoritmo es al menos un factor de $1/x - 2 + 2x$ de $mf(\overline{p})$, cuyo mínimo para $x \in [0,1]$ es en $x = 1/\sqrt{2}$, con un valor de $2(\sqrt{2}-1)$. Usando la Cota de la Carga Promedio, concluimos que $\mathrm{alg} \geq 2(\sqrt{2}-1)\mathrm{opt}$. □

Se conjetura que el análisis no es ajustado, esto pues la cota de la carga promedio tiende a no ser ajustada.

Ejemplo 5.3. Consideremos la instancia que consiste en dos máquinas, la función $f(x) = \text{mín}(x, 3)$ y tres trabajos de largo 2. LPT asigna dos a una máquina y una la otra obteniendo un valor de alg $= 5$. Esto es de hecho óptimo pues la única otra asignación diferente es tener una máquina vacía, la cual tiene a lo más la misma productividad. Aun así, la cota de la carga promedio, como $\overline{p} = 3$, nos dice solo que opt ≤ 6. Y el análisis del Teorema 5.2 solo nos indicaría que LPT es una $5/6$ aproximación cuando es en realidad óptimo. En la Figura 5.1 se ilustra esta asignación. Más aún, se puede aumentar esta instancia teniendo $\approx \sqrt{2}m$

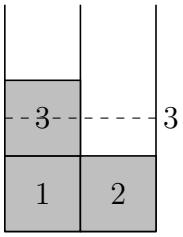

Figura 5.1: Resultado de aplicar LPT y asignación óptima para la instancia.

trabajos de largo 2, para los cuales LPT sigue siendo óptimo, pero un análisis como el del Teorema 5.2 contra la cota de la carga promedio solo da una garantía de $\approx 2(\sqrt{2} - 1)$.

Así, se cree que un análisis más exhaustivo y combinatorial, como los de [13, 10] del mismo algoritmo para *Makespan Minimization*, podría resultar en una mayor garantía para el algoritmo. Se tiene al menos una cota superior de $11/12$, que funciona para cualquier número par de máquinas m. Para dos máquinas, considere la función $f(x) = \text{mín}(x, 6)$, junto con la instancia que consiste en dos trabajos de largo 3 y tres de largo 2. LPT asigna primero cada trabajo de largo 3 a una máquina diferente, luego dos de largo 2 a una máquina y uno a la otra. Esto logra un valor de alg $= 11$. Mientras que el óptimo asigna ambos trabajos de largo 3 a una máquina y todos los de largo 2 a la otra, logrando opt $= 12$. Por lo tanto alg $= \frac{11}{12}$opt. En la Figura 5.2 se ilustran ambas asignaciones.

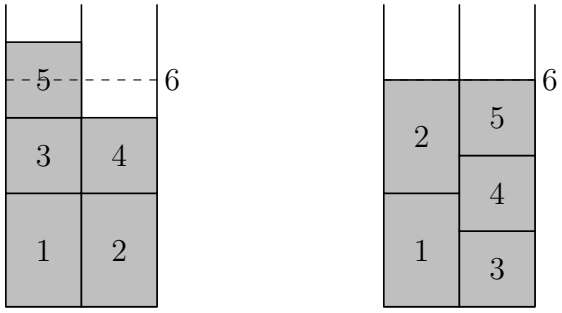

Figura 5.2: Resultado de aplicar LPT (izquierda) y una asignación óptima (derecha).

Para cualquier m par, basta con duplicar esta instancia $m/2$ veces para obtener la misma cota. Además, para cualquier $m \geq 3$ impar se puede duplicar esta instancia $(m-1)/2$ veces y agregar un trabajo de largo 6 para obtener una cota de $11/12 + 1/(6m)$, la cual es mayor pero asintóticamente igual a $11/12$.

5.2. Análisis Ajustado de *LPT* para $m = 2$

Como se discutió en la Sección 5.1, nuestro análisis general no está completo, teniendo solo una cota superior de 11/12 para dos máquinas. En esta sección mostraremos que la garantía de LPT en dos máquinas es exactamente $11/12 \approx 0{,}916$. Como antes, suponemos que los n trabajos están ordenados en orden descendente con respecto a sus largos, de modo que LPT los asigna en orden de 1 a n.

Dado que estamos trabajando con dos máquinas, si el último trabajo asignado a la máquina más cargada al final de la ejecución j no es el último trabajo n, entonces por el Corolario 4.8 la aproximación será al menos tan buena como la de la misma asignación quitando los trabajos $j' > j$. Así, basta probar la garantía cuando luego de asignar n, la máquina a la cual se asigna termina siendo la más cargada.

El análisis se puede dividir en tres partes: demostrar que LPT es una 11/12-aproximación para $n \geq 6$, probar que es óptimo para $n \leq 4$, y explorar exhaustivamente el caso $n = 5$ para mostrar que sigue siendo una 11/12-aproximación en ese caso. En el caso $n = 5$ se utilizan además programas lineales para encontrar cotas inferiores. Esta es una estrategia de demostración ya usada por Della Croce, Scatamacchia y T'kindt [10] y Della Croce y Scatamacchia [9], quienes también lo utilizan para analizar LPT y sus variantes para el problema *Makespan Minimization*.

Sea ALG el resultado de aplicar LPT a una instancia dada. Como antes, ALG_i^j son los trabajos asignados a la máquina i inmediatamente después de asignar el trabajo j.

Lema 5.4. LPT *es una* $1-1/(2n)$*-aproximación para* Machine Productivity *en dos máquinas cuando* n *está en la máquina más cargada.*

Demostración. El trabajo n es el más corto, por lo que es más corto que una fracción $1/n$ de la carga total $p([n]) = 2\overline{p}$, entonces $p_n \leq \frac{2}{n}\overline{p}$. Sin pérdida de generalidad, supongamos que la máquina 1 es la más cargada. Dado que n se asignó a 1, la carga de la máquina 1 era como máximo la de la máquina 2 antes de que se asignara n, $p(\mathrm{ALG}_2) \geq p(\mathrm{ALG}_1^{n-1})$. Tenemos:

$$2\overline{p} = p(\mathrm{ALG}_2) + p(\mathrm{ALG}_1^{n-1}) + p_n \leq 2p(\mathrm{ALG}_2) + \frac{2}{n}\overline{p}.$$

De donde obtenemos que $p(\mathrm{ALG}_2) \geq (1 - \frac{1}{n})\overline{p}$. Por monotonía y superhomogeneidad, esto implica que $f(p(\mathrm{ALG}_2)) \geq (1 - \frac{1}{n})f(\overline{p})$.

Por otro lado, para la máquina 1, como es la más cargada, su carga es al menos el promedio $\overline{p}$. Por monotonía, $f(p(\mathrm{ALG}_1)) \geq f(\overline{p})$. Sumando ambas partes, obtenemos: $\mathrm{alg} \geq (2-\frac{1}{n})f(\overline{p})$. Por la Cota de la Carga Promedio: $\mathrm{opt} \leq 2f(\overline{p})$, y así: $\mathrm{alg} \geq (1 - \frac{1}{2n})\mathrm{opt}$. □

Entonces, si $n \geq 6$, obtenemos el resultado deseado:

Corolario 5.5. LPT *es una* 11/12*-aproximación para* Machine Productivity *en dos máquinas cuando* $n \geq 6$ *está en la máquina más cargada.*

Por lo tanto, solo necesitamos verificar las posibilidades restantes para n. Usaremos la siguiente estrategia para probar que ALG es óptimo. Dada cualquier otra asignación A, probaremos que ALG tiene al menos el mismo valor. Para ello usaremos el Lema de Intercambio Convexo entre las cargas de ambas asignaciones $p(A_1)$, $p(A_2)$, $p(\mathrm{ALG}_1)$ y $p(\mathrm{ALG}_2)$. Como ambas serán asignaciones de los mismos trabajos, la hipótesis $p(A_1) + p(A_2) = p(\mathrm{ALG}_1) + p(\mathrm{ALG}_2)$ se cumplirá. Entonces, bastará verificar que $p(A_1) \leq p(\mathrm{ALG}_1) \leq p(A_2)$ o $p(A_1) \leq p(\mathrm{ALG}_2) \leq p(A_2)$. Obteniendo que $f(p(\mathrm{ALG}_1)) + f(p(\mathrm{ALG}_2)) \geq f(p(A_1)) + f(p(A_2))$, es decir, el valor de ALG es al menos el mismo de A.

Lema 5.6. LPT *es óptimo para* Machine Productivity *en dos máquinas cuando* $n \leq 4$.

Demostración. Para $n \leq 2$, LPT asigna cada trabajo a una máquina distinta, lo cual es óptimo.

Para toda asignación A, de ahora en adelante supondremos sin pérdida de generalidad que el trabajo 1 está en la máquina 1, es decir, $1 \in A_1$.

Para $n = 3$, se tiene que LPT genera la asignación $\{1\}$ y $\{2, 3\}$. Sea $A \neq \mathrm{ALG}$ otra asignación. Para que sean distintas, el trabajo 1 debe no estar solo en A. Entonces la máquina 1 tiene al menos dos elementos, y la máquina 2 tiene a lo más un trabajo, que puede ser el trabajo 2 o el 3. Luego $p(A_2) \leq p_1$ y $p_1 \leq p(A_1)$, donde $p_1 = p(\mathrm{ALG}_1)$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A. Por lo tanto, ninguna asignación es mejor que ALG, por lo que es óptima cuando $n = 3$.

Para $n = 4$, dividimos en dos casos. Si $p_1 \geq p_2 + p_3$, entonces LPT genera la asignación $\{1\}$ y $\{2, 3, 4\}$. Sea $A \neq \mathrm{ALG}$ otra asignación. Para que sean distintas, el trabajo 1 debe no estar solo en A, entonces la otra máquina tiene a lo más dos trabajos, que puede ser el trabajo 2, 3 o 4. Luego $p(A_2) \leq p_2 + p_3 \leq p_1$ por hipótesis, y $p_1 \leq p(A_1)$, donde $p_1 = p(\mathrm{ALG}_1)$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A. Por lo tanto, ninguna asignación es mejor que ALG, por lo que es óptima en este caso.

Si $p_1 < p_2 + p_3$, entonces LPT genera la asignación $\{1, 4\}$ y $\{2, 3\}$. Sea $A \neq \mathrm{ALG}$ otra asignación. Si A tiene una máquina vacía, es fácil ver que tiene a lo más el mismo valor que ALG.

Ahora suponemos que A no tiene máquinas vacías. Si A es la asignación $\{1\}$ y $\{2, 3, 4\}$, entonces por hipótesis $p(A_1) = p_1 \leq p_1 + p_4 < p_2 + p_3 + p_4 = p(A_2)$, donde $p_1 + p_4 = p(\mathrm{ALG}_1)$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A.

Queda ver el caso en que A no tiene máquinas vacías y el trabajo 1 no está solo. Para que sean distintas, A_1 no puede ser $\{1, 4\}$, por lo que la máquina 1 debe tener al trabajo 2 o al 3. Entonces $p(A_1) \geq p_1 + p_3 \geq p_1 + p_4 = p(\mathrm{ALG}_1)$. Por otro lado, la otra máquina debe tener a lo más dos trabajos, entre los que no está el trabajo 1 y tiene a lo más uno entre los trabajos 2 y 3, por lo que $p(A_2) \leq p_2 + p_4 \leq p_1 + p_4 = p(\mathrm{ALG}_1)$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A. Por lo tanto, ninguna asignación es mejor que ALG, por lo que es óptima también en este caso.

Habiendo visto que ALG siempre tiene al menos el mismo valor que cualquier otra asignación, concluimos que es óptimo para $n = 4$. □

El siguiente lema permite traducir la relación entre las máquinas menos cargadas y la relación entre los valores de productividad.

Lema 5.7. *Supongamos que A es una asignación de valor* alg' *y que la máquina menos cargada de* ALG *y A es la máquina* 1*. Supongamos que* $p(\text{ALG}_1) \leq p(A_1) \leq p(A_2) \leq p(\text{ALG}_2)$*.*

$$\text{alg} \geq \frac{1}{2}\left(1 + \frac{p(\text{ALG}_1)}{p(A_1)}\right)\text{alg}'.$$

Demostración. Por monotonía, $f(p(A_2)) \leq f(p(\text{ALG}_2))$. Sea $\alpha = p(\text{ALG}_1)/p(A_1)$. Por superhomogeneidad, $f(p(\text{ALG}_1)) \geq \alpha f(p(A_1))$. Entonces:

$$\text{alg} \geq \alpha f(p(A_1)) + f(p(A_2)) = -(1-\alpha)f(p(A_1)) + \text{alg}'.$$

Por monotonía, $f(p(A_1)) \leq f(p(A_2))$, por lo que $\text{alg}' \geq 2f(p(A_1))$. Por lo tanto, $\text{alg} \geq (1 - \frac{1}{2}(1-\alpha))\text{alg}'$. Donde el factor es igual a $\frac{1}{2}(1+\alpha)$. □

Esto nos dice que para probar que el factor es al menos 11/12, es suficiente mostrar una relación de 5/6 entre las máquinas menos cargadas de LPT y de la asignación óptima.

Para probar el siguiente lema, además de usar la misma estrategia que en la demostración del Lema 5.6 para probar optimalidad en algunos casos, probaremos que LPT logra al menos una fracción de 11/12 en comparación con cualquier otra asignación $A \neq \text{ALG}$ posible. Para ello, buscaremos usar el Lema 5.7 comparando la máquina menos cargada para ALG y de A. Llamando $\underline{i}$ a la máquina menos cargada de ALG y A, para encontrar la relación mínima entre sus cargas, buscaremos minimizar $p(\text{ALG}_{\underline{i}})$ sujeto a la carga normalizada $p(A_{\underline{i}}) = 1$. Añadiendo algún conocimiento de $\text{ALG}_{\underline{i}}$ y $A_{\underline{i}}$, veremos que esto se puede plantear esto como un problema de programación lineal en 5 variables $\{p_j\}_{j\in[5]}$.

Lema 5.8. LPT *es una* 11/12 *aproximación* Machine Productivity *en dos máquinas cuando* $n = 5$*.*

Demostración. Para toda asignación A, supondremos sin pérdida de generalidad que el trabajo 1 está en la máquina 1, es decir, $1 \in A_1$.

Si $p_1 \geq p_2 + p_3 + p_4$, entonces LPT genera la asignación $\{1\}$ y $\{2,3,4,5\}$. Sea otra asignación $A \neq \text{ALG}$, 1 no puede estar solo, por lo que $p(\text{ALG}_1) = p_1 \leq p(A_1)$. Además, la máquina 2 debe tener a lo más tres trabajos, con una carga de a lo más $p(A_2) \leq p_2+p_3+p_4 \leq p_1 = p(\text{ALG}_1)$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A. Por lo tanto, ninguna asignación es mejor que ALG, por lo que es óptima en este caso. Un ejemplo de este caso se ilustra en la Figura 5.3.

Ahora supongamos que $p_1 \leq p_2 + p_3 + p_4$. Asumiremos que $n = 5$ está en la máquina más cargada, pues si no por el Corolario 4.8 el resultado sigue de considerar la asignación hasta el último trabajo asignado a la máquina más cargada $j \leq 4$ y el Lema 5.6. Es directo que la asignación con una máquina vacía tiene a lo más el mismo valor que ALG.

Consideremos una asignación $B \neq \text{ALG}$, que tiene una máquina $\underline{i}$ con exactamente un trabajo. Sea $\overline{i}$ la otra máquina. Se tiene que $p(B_{\underline{i}}) \leq p_1$. Consideremos A la asignación

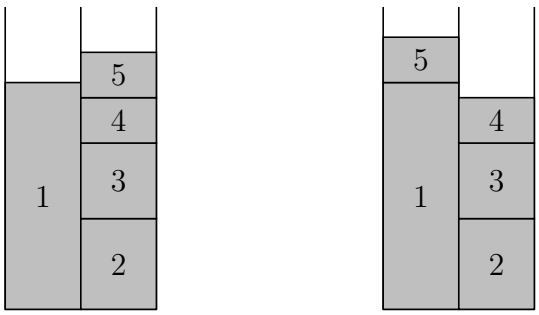

Figura 5.3: Resultado de aplicar LPT (izquierda) y una asignación donde 1 no queda solo (derecha) en el caso $p_1 \geq p_2 + p_3 + p_4$.

obtenida a partir de B moviendo el trabajo más corto de la máquina $\bar{i}$. Entonces $A_{\bar{i}}$ queda con tres trabajos, por lo que $p(A_{\bar{i}}) \geq p_2 + p_3 + p_4 \geq p_1$. Aplicando el Lema de Intercambio Convexo, el valor de A es al menos el mismo de B. Donde A es una asignación con al menos dos trabajos en cada máquina.

De ahora en adelante, sea $A \neq \text{ALG}$ una asignación que asigna al menos dos trabajos a cada máquina. Veremos que ALG tiene al menos 11/12 del valor de A. Juntándolo con lo anterior, se concluiría también para una asignación B con una máquina con exactamente un trabajo.

Si $p_1 \geq p_2 + p_3$, LPT genera la asignación $\{1, 5\}$ y $\{2, 3, 4\}$. La asignación A tiene en su máquina 2 a lo más tres trabajos, pero no tiene al trabajo 1, entonces $p(A_2) \leq p_2 + p_3 + p_4 = p(\text{ALG}_2)$. Como es distinta de ALG, en la máquina 1 no puede ser $\{1, 5\}$, y debe tener al menos dos trabajos, así que tiene al menos otro trabajo $j \in \{2, 3, 4\}$, por lo que $p_j \geq p_4$. Añadiendo la hipótesis, se obtiene que $p(\text{ALG}_2) = p_2 + p_3 + p_4 \leq p_1 + p_j = p(A_1)$. Usando el Lema de Intercambio Convexo, se concluye que el valor de ALG es al menos el mismo de A.

Cuando $p_1 \leq p_2 + p_3$, dividimos los casos nuevamente. En lo que sigue, referenciaremos frecuentemente a las siguientes desigualdades lineales. La primera condición (5.3) corresponde a que la instancia viene ordenada decrecientemente, mientras que la segunda (5.4) corresponde a la hipótesis que acabamos de agregar.

$$0 \leq p_5 \leq p_4 \leq p_3 \leq p_2 \leq p_1, \tag{5.3}$$
$$p_1 \leq p_2 + p_3, \tag{5.4}$$
$$p_1 + p_4 \leq p_2 + p_3, \tag{5.5}$$
$$p_2 + p_3 \leq p_1 + p_4 + p_5, \tag{5.6}$$
$$p_2 \leq p_4 + p_5, \tag{5.7}$$
$$p_1 + p_2 \leq p_3 + p_4 + p_5, \tag{5.8}$$
$$p_3 + p_4 + p_5 \leq p_1 + p_2, \tag{5.9}$$
$$p_3 \leq p_4 + p_5, \tag{5.10}$$
$$p_1 + p_3 \leq p_2 + p_4 + p_5, \tag{5.11}$$
$$p_2 + p_4 + p_5 \leq p_1 + p_3, \tag{5.12}$$
$$p_2 + p_3 \leq p_1 + p_4, \tag{5.13}$$

$$p_1 + p_4 \leq p_2 + p_3 + p_5, \tag{5.14}$$
$$p_1 \leq p_3 + p_5, \tag{5.15}$$
$$p_1 \leq p_2 + p_5. \tag{5.16}$$

Si (5.5) se tiene, es decir, si $p_1 + p_4 \leq p_2 + p_3$, LPT genera la asignación $\{1, 4, 5\}$ y $\{2, 3\}$. Notemos que como $n = 5$ está en la máquina más cargada, se tiene (5.6), $p_2+p_3 \leq p_1+p_4+p_5$. Con lo que la máquina menos cargada de ALG es la 2 con $p(\text{ALG}_2) = p_2 + p_3$. Separamos los casos en los que, en A, 1 está con dos trabajos, con 4 o 5, con 2 y con 3.

Con dos trabajos: Si el trabajo 1 está con dos trabajos en A. La suma de largos de estos dos trabajos es al menos p_4+p_5. Entonces, $p(A_1) \geq p_1+p_4+p_5 = p(\text{ALG}_1)$. En A, la máquina 2 tiene dos trabajos, entre los cuales no está el el trabajo 1. Usando además (5.6), se obtiene que $p(A_2) \leq p_2+p_3 \leq p_1+p_4+p_5 = p(\text{ALG}_1)$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A.

Con 4 o 5: Si el trabajo 1 está solo con el trabajo 4 o el 5 en la máquina 1, entonces $p(A_1) \leq p_1 + p_4 \leq p_1 + p_4 + p_5 = p(\text{ALG}_1)$. Además, la máquina 2 contiene al menos a los trabajos 2 y 3, además de otro trabajo (cuyo largo es al menos p_5), por lo que $p(A_2) \geq p_2 + p_3 + p_5 \geq p_1 + p_4 + p_5 = p(\text{ALG}_1)$. Donde se usó la hipótesis (5.5). Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A.

Con 2: Si A es la asignación $\{1, 2\}$ y $\{3, 4, 5\}$, distinguimos dos casos. Si $p_2 > p_4 + p_5$, entonces tenemos: $p(A_2) = p_3 + p_4 + p_5 \leq p_1 + p_4 + p_5 < p_1 + p_2 = p(A_1)$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A.

De lo contrario, tenemos que $p_2 \leq p_4 + p_5$, lo cual corresponde a la condición (5.7). Según qué máquina sea la menos cargada, consideramos un programa lineal distinto. Si (5.8) se tiene, es decir, si $p(A_1) = p_1 + p_2 \leq p_3 + p_4 + p_5 \leq p(A_2)$, entonces consideremos el programa lineal:

$$(PL_1) \qquad \text{mín}\{p_2 + p_3 \mid p_1 + p_2 = 1, (5.3), (5.4), (5.5), (5.6), (5.7), (5.8)\}.$$

Si por otro lado (5.8) no se tiene, entonces se deberá tener (5.9), es decir, si $p_3 + p_4 + p_5 \leq p_1 + p_2$, consideramos el programa lineal:

$$(PL_2) \qquad \text{mín}\{p_2 + p_3 \mid p_3 + p_4 + p_5 = 1, (5.3), (5.4), (5.5), (5.6), (5.7)\ (5.9)\}.$$

Ambos programas tienen un valor de 5/6. Esto quiere decir que la peor razón posible entre las máquinas menos cargadas de A y de ALG es 5/6. Por el Lema 5.7, el valor de ALG está en al menos una razón de al menos 11/12 con respecto al valor de A.

Con 3: Si A es la asignación $\{1, 3\}$ y $\{2, 4, 5\}$, distinguimos dos casos. Si $p_3 > p_4 + p_5$, entonces tenemos: $p(A_2) \leq p_2 + p_4 + p_5 \leq p_1 + p_4 + p_5 < p_1 + p_3 = p(A_1)$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A.

De lo contrario, tenemos que $p_3 \leq p_4 + p_5$, lo cual corresponde a la condición (5.10). Según que máquina sea la menos cargada, consideramos un programa lineal distinto. Si (5.11) se tiene, es decir, si $p(A_1) = p_1 + p_3 \leq p_2 + p_4 + p_5 = p(A_2)$, entonces consideremos el programa lineal:

$$(PL_3) \qquad \text{mín}\{p_2 + p_3 \mid p_1 + p_3 = 1, (5.3), (5.4), (5.5), (5.6), (5.10), (5.11)\}.$$

Si por otro lado (5.11) no se tiene, entonces se deberá tener (5.12), es decir, si $p_2 + p_4 + p_5 \leq p_1 + p_3$, consideramos el programa lineal:

$$(PL_4) \qquad \text{mín}\{p_2 + p_3 \mid p_2 + p_4 + p_5 = 1, (5.3), (5.4), (5.5), (5.6), (5.10), (5.12)\}.$$

Ambos programas tienen un valor de 6/7. Esto quiere decir que la peor razón posible entre las máquinas menos cargadas de A y de ALG es 6/7. Por el Lema 5.7, el valor de ALG está en al menos una razón de al menos 13/14 con respecto al valor de A.

Esto concluye el caso $p_1 + p_4 \leq p_2 + p_3$, teniendo siempre que ALG es al menos un factor 11/12 de cualquier otra asignación A.

Si por otro lado (5.5) no se tiene, se cumple (5.13), es decir, se tiene que $p_1 + p_4 \geq p_2 + p_3$. En este caso, LPT genera la asignación $\{1, 4\}$ y $\{2, 3, 5\}$. Notemos que como $n = 5$ está en la máquina más cargada, se tiene (5.14), $p_1 + p_4 \leq p_2 + p_3 + p_5$. Con lo que la máquina menos cargada de ALG es la 2 con $p(\text{ALG}_2) = p_2 + p_3 + p_5$. Separamos los casos en los que, en A, 1 está con dos trabajos, con 5, con 2 y con 3.

Con dos trabajos: Si el trabajo 1 está con dos trabajos en A. Alguno de estos trabajos debe ser al menos tan largo como p_4, por lo que $p(A_1) \geq p_1 + p_4 = p(\text{ALG}_1)$. En la máquina 2 de A hay dos trabajos, pero no está el trabajo 1, y como por hipótesis se cumple (5.13), se obtiene que $p(A_2) \leq p_2 + p_3 \leq p_1 + p_4 = p(\text{ALG}_1)$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A.

Con 5: Si A es la asignación $\{1, 5\}$ y $\{2, 3, 4\}$, tenemos que $p(A_1) = p_1 + p_5 \leq p_1 + p_4 = p(\text{ALG}_1)$ y como se cumple (5.14), entonces $p_1 + p_4 \leq p_2 + p_3 + p_5 = p(A_2)$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A.

Con 2: Si A es la asignación $\{1, 2\}$ y $\{3, 4, 5\}$, distinguimos dos casos. Si $p_1 > p_3 + p_5$, entonces tenemos: $p(A_2) = p_3 + p_4 + p_5 \leq p_2 + p_3 + p_5 < p_1 + p_2 = p(A_1)$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A.

De lo contrario, tenemos que $p_1 \leq p_3 + p_5$, lo cual corresponde a la condición (5.15). Según que máquina sea la menos cargada, consideramos un programa lineal distinto. Si (5.8) se tiene, es decir, si $p(A_1) = p_1 + p_2 \leq p_3 + p_4 + p_5 = p(A_2)$, entonces consideremos el programa lineal:

$$(PL_5) \qquad \text{mín}\{p_1 + p_4 \mid p_1 + p_2 = 1, (5.3), (5.4), (5.13), (5.14), (5.15), (5.8)\}.$$

Si por otro lado (5.8) no se tiene, entonces se deberá tener (5.9), es decir, si $p_3 + p_4 + p_5 \leq p_1 + p_2$, consideramos el programa lineal:

$$(PL_6) \qquad \text{mín}\{p_1 + p_4 \mid p_3 + p_4 + p_5 = 1, (5.3), (5.4), (5.13), (5.14), (5.15)\ (5.9)\}.$$

Ambos programas tienen un valor de 5/6. Esto quiere decir que la peor razón posible entre las máquinas menos cargadas de A y de ALG es 5/6. Por el Lema 5.7, el valor de ALG está en al menos una razón de al menos 11/12 con respecto al valor de A.

Con 3: Si A es la asignación $\{1, 3\}$ y $\{2, 4, 5\}$, distinguimos dos casos. Si $p_1 > p_2 + p_5$, entonces tenemos: $p(A_2) = p_2 + p_4 + p_5 \leq p_2 + p_3 + p_5 < p_1 + p_3$. Por el Lema de Intercambio Convexo, el valor de ALG es al menos el mismo de A.

De lo contrario, tenemos que $p_1 \leq p_2 + p_5$, lo cual corresponde a la condición (5.16). Según que máquina sea la menos cargada, consideramos un programa lineal distinto. Si (5.11) se tiene, es decir, si $p(A_1) = p_1 + p_3 \leq p_2 + p_4 + p_5 = p(A_2)$, entonces consideremos el programa lineal:

$$(PL_7) \qquad \text{mín}\{p_1 + p_4 \mid p_1 + p_3 = 1, (5.3), (5.4), (5.13), (5.14), (5.16), (5.11)\}.$$

Si por otro lado (5.11) no se tiene, entonces se deberá tener (5.12), es decir, si $p_2 + p_4 + p_5 \leq p_1 + p_3$, consideramos el programa lineal:

$$(PL_8) \qquad \text{mín}\{p_1 + p_4 \mid p_2 + p_4 + p_5 = 1, (5.3), (5.4), (5.14), (5.13), (5.16), (5.12)\}.$$

Ambos programas tienen un valor de 6/7. Esto quiere decir que la peor razón posible entre las máquinas menos cargadas de A y de ALG es 6/7. Por el Lema 5.7, el valor de ALG está en al menos una razón de al menos 13/14 con respecto al valor de A.

Esto concluye el caso $p_1 + p_4 \leq p_2 + p_3$, teniendo siempre que ALG es al menos un factor 11/12 de cualquier otra asignación A.

Hemos probado que el valor de ALG es al menos un factor 11/12 del valor de cualquier otra asignación de los 5 trabajos en dos máquinas. Por lo tanto, LPT es una 11/12-aproximación en dos máquinas con cinco trabajos. □

Habiendo cubierto todos los casos, se prueba finalmente la garantía.

Teorema 5.9. LPT *es una* 11/12*-aproximación para* Machine Productivity *en dos máquinas.*

Demostración. Sea j el último trabajo de la máquina más cargada. Por el Corolario 4.8, la aproximación es mejor que la de la restricción de la instancia a los trabajos 1 hasta j.

Si $j \leq 4$, por el Lema 5.6 se concluye que ALG es óptimo. Si $j = 5$, por el Lema 5.8 se concluye que ALG es al menos una 11/12-aproximación. Y si $j \geq 6$, el Corolario 5.5 concluye que ALG es al menos una 11/12-aproximación.

Así, en cualquier caso, ALG es al menos una 11/12-aproximación. □

Capítulo 6

Conclusión y Problemas Abiertos

En esta tesis se desarrollan técnicas y argumentos que pueden ser de interés individual en balanceo de carga bajo funciones cóncavas. Con esto se hicieron avances para el problema *Machine Productivity*, bajo la forma de algoritmos y análisis respectivos, y cotas superiores para la mejor garantía posible.

Se probó que regla *Longest Processing Time First* alcanza una $2(\sqrt{2}-1) \approx 0{,}828$-aproximación, y se conjetura que un mejor análisis podría mejorar el valor hasta $11/12 \approx 0{,}916$. Mediante un análisis exhaustivo se demuestra que $11/12$ es la garantía exacta en dos máquinas, quedando abierta la pregunta en el caso general.

En la versión online adversarial, se prueba que *List Scheduling* es exactamente $3/4$-competitivo, lo cual es óptimo dentro de los algoritmos online deterministas que sean return-oblivious. Se exhibe una cota superior constante sobre la competitividad de cualquier algoritmo determinista de 0,809, y se propone un algoritmo que la alcanza en dos máquinas. Queda abierto si esto es posible de asegurar en más de dos máquinas. Obtener cotas ajustadas a cada número de máquinas puede dar una idea para desarrollar un tal algoritmo. Para algoritmos aleatorios se obtiene una cota superior de $9/10$, y tampoco se sabe si esta cota es ajustada. Como trabajo futuro se propone estudiar el algoritmo de asignación aleatoria (*Random Assign*) que tiende a tener buenas garantías, como lo hace para *Submodular Welfare* [26].

En orden aleatorio se obtiene una cota superior de aproximadamente 0,936 para algoritmos deterministas y de 0,966 para aleatorios. Se cree que *LS* puede superar el $3/4$ en este contexto. Más allá, se puede explorar el problema en un contexto estocástico, esto es cuando los trabajos llegan con largos aleatorios. Un primer contexto frecuentemente estudiado es el de largos i.i.d. (independientes idénticamente distribuidos).

La versión de *Machine Productivity* con máquinas distinta tiene gran interés al estar entre la versión idéntica y *Submodular Welfare* en términos de dificultad. Posiblemente se pueda extender el **PTAS** de Alon et al. [2] para la versión offline. Otra variante interesante es el caso semionline, el cual es una versión online pero con información extra. Un dato útil podría ser la carga promedio de la instancia completa, pues en el caso offline y en los análisis demuestra ser extremadamente relevante, por lo que puede permitir a un algoritmo mejorar en garantía.

Bibliografía

[1] Alexander A Ageev and Maxim I Sviridenko. Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8:307–328, 2004.

[2] Noga Alon, Yossi Azar, Gerhard J Woeginger, and Tal Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

[3] Liad Blumrosen and Noam Nisan. On the computational power of demand queries. *SIAM Journal on Computing*, 39(4):1372–1391, 2010.

[4] Niv Buchbinder, Moran Feldman, Yuval Filmus, and Mohit Garg. Online submodular maximization: Beating 1/2 made simple. *Mathematical Programming*, 183(1-2):149–169, 2020.

[5] Niv Buchbinder, Moran Feldman, and Mohit Garg. Deterministic $(1/2 + \varepsilon)$-approximation for submodular maximization over a matroid. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 241–254. SIAM, 2019.

[6] Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 1433–1452. SIAM, 2014.

[7] Gruia Calinescu, Chandra Chekuri, Martin Pal, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[8] Deeparnab Chakrabarty and Gagan Goel. On the approximability of budgeted allocations and improved lower bounds for submodular welfare maximization and gap. *SIAM Journal on Computing*, 39(6):2189–2211, 2010.

[9] Federico Della Croce and Rosario Scatamacchia. The longest processing time rule for identical parallel machines revisited. *Journal of Scheduling*, 23(2):163–176, 2020.

[10] Federico Della Croce, Rosario Scatamacchia, and Vincent T'kindt. A tight linear time 13/12-approximation algorithm for the $P2||C_{\text{máx}}$ problem. *Journal of Combinatorial Optimization*, 38:608–617, 2019.

[11] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. *An analysis of approximations for maximizing submodular set functions - II.* Springer, 1978.

[12] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[13] Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics*, 17(2):416–429, 1969.

[14] Dorit S Hochbaum and David B Shmoys. Using dual approximation algorithms for scheduling problems theoretical and practical results. *Journal of the ACM (JACM)*, 34(1):144–162, 1987.

[15] Zhiyi Huang, Qiankun Zhang, and Yuhao Zhang. Adwords in a panorama. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020*, 2020.

[16] Klaus Jansen, Kim-Manuel Klein, and José Verschae. Closing the gap for makespan scheduling via sparsification techniques. *Mathematics of Operations Research*, 45(4):1371–1392, 2020.

[17] Michael Kapralov, Ian Post, and Jan Vondrák. Online submodular welfare maximization: Greedy is optimal. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1216–1225. SIAM, 2013.

[18] Richard M. Karp, Umesh V. Vazirani, and Vijay V. Vazirani. An optimal algorithm for on-line bipartite matching. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, STOC 1990*, 1990.

[19] Nathaniel Kell and Kevin Sun. Approximations for indivisible concave allocations with applications to nash welfare maximization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37(5), pages 5705–5713, 2023.

[20] Subhash Khot, Richard J Lipton, Evangelos Markakis, and Aranyak Mehta. Inapproximability results for combinatorial auctions with submodular utility functions. In *Internet and Network Economics: First International Workshop, WINE 2005, Hong Kong, China, December 15-17, 2005. Proceedings 1*, pages 92–101. Springer, 2005.

[21] Benny Lehmann, Daniel Lehmann, and Noam Nisan. Combinatorial auctions with decreasing marginal utilities. In *Proceedings of the 3rd ACM conference on Electronic Commerce*, pages 18–28, 2001.

[22] Hendrik W Lenstra Jr. Integer programming with a fixed number of variables. *Mathematics of operations research*, 8(4):538–548, 1983.

[23] Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *J. ACM*, 54(5):22, 2007.

[24] Vahab Mirrokni, Michael Schapira, and Jan Vondrák. Tight information-theoretic lower bounds for welfare maximization in combinatorial auctions. In *Proceedings of the 9th ACM conference on Electronic commerce*, pages 70–77, 2008.

[25] Aravind Srinivasan. Budgeted allocations in the full-information setting. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop*, 2008.

[26] Jan Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74, 2008.

[27] Gerhard J Woeginger. A polynomial-time approximation scheme for maximizing the minimum machine completion time. *Operations Research Letters*, 20(4):149–154, 1997.

Anexo A

Esquemas de Aproximación a Tiempo Polinomial

Los siguientes resultados son pequeñas mejoras a lo ya conocido con respecto a la existencia de esquemas de aproximación a tiempo polinomial para el problemas *Machine Productivity*. Por esto, no se presentan junto a los resultados principales en el cuerpo de esta tesis. Aún así, tienen cierto valor agregado, como para merecer ser mencionados.

Los aportes son de dos tipos. Los primeros son simplificaciones de las hipótesis e implementación con respecto al **EPTAS** existente de Alon et al. [2]. El segundo es una extensión de uno de los componentes del esquema al caso de máquinas distintas, lo cual no es suficiente para tener la existencia en este caso más general.

A.1. Estado del Arte

Dada una asignación $A = \{A_i\}_{i\in[m]}$, recordemos que $p(A_i)$ denota la carga de la máquina i en la asignación A. Consideremos los siguientes problemas con $f : \mathbb{R}_+ \to \mathbb{R}_+$:

(I) $\min \sum_{i=1}^{m} f(p(A_i))$ para f convexa,

(II) $\min \max_{i=1}^{m} f(p(A_i))$ para f convexa,

(I') $\max \sum_{i=1}^{m} f(p(A_i))$ para f cóncava,

(II') $\max \min_{i=1}^{m} f(p(A_i))$ para f cóncava.

Notar que el problema (I') corresponde *Machine Productivity* relajando la hipótesis de normalización, es decir, que $f(0) = 0$. Más aún, como se menciona en la Observación 2.2, se puede quitar dicha hipótesis en *Machine Productivity* manteniendo los resultados de aproximación.

Definamos la siguiente condición sobre la función f:

$$(F*) \qquad \forall \varepsilon > 0, \exists \delta > 0, \forall x, y \geq 0, |y - x| \leq \delta x \implies |f(y) - f(x)| \leq \epsilon f(x).$$

Alon et al. extienden los **EPTAS** para otros problemas de balanceo de carga como *Makespan Minimization* [14] y *Machine Covering* [27], logrando que se apliquen a los problemas (I), (II), (I') y (II') cuando f cumple la condición $(F*)$.

El esquema para el problema (I') se describe a continuación. Primero los trabajos gigantes son asignados en una máquina distinta cada uno. Luego se considera la instancia I restante sin los trabajos gigantes ni sus máquinas. Sea $\varepsilon > 0$, sea $\delta > 0$ dado por la propiedad $(F*)$, se deine $\lambda = \lceil 5/\delta \rceil$, un trabajo j se dice largo si $p_j > \overline{p}/\lambda$, y corto si no. Se define la instancia redondeada $I^{\#}$ como la obtenida a partir de I redondeando los tiempos de proceso de los trabajos largos al multiplo entero más cercano de $\overline{p}/\lambda^2$. Sea S la suma de los tiempos de proceso de los trabajos cortos, y redondeemoslo a $S^{\#}$ el multiplo entero más cercano de $\overline{p}/\lambda$. En $I^{\#}$ se reemplazan los trabajos cortos por $S^{\#}\lambda/\overline{p}$ trabajos nuevos de largo $\overline{p}/\lambda$ cada uno. Además, $I^{\#}$ tiene la misma cantidad de máquinas que I.

Se puede encontrar el óptimo para la instancia redondeada $I^{\#}$ de dos formas. La primera es usando programación dinámica, avanzando de una en una la cantidad de máquinas, se exploran todos los subconjuntos de trabajos de la entrada. Notar que en $I\#$ todos los trabajos tienen tiempos de proceso que son multiplos de $\overline{p}/\lambda^2$ menores o iguales a $\overline{p}$. Para un conjunto de trabajos J, definimos una configuración s como un vector cuyas coordenada $h \in \{0, \ldots, \lambda^2\}$ representa la cantidad de trabajos de largo $h\overline{p}/\lambda^2$ en J. Naturalmente se define $p(s) = p(J)$ la carga de asignar la configuración s a una máquina. Consideremos la matriz P, cuya coordenada (i, s) representa el valor de productividad óptima en las primeras i máquinas y con los trabajos indicados por la configuración s. Se inicia el programa con: $P(1, s) = f(p(s))$. Luego, se resuelve iterativamente el problema de optimización para toda cantidad de máquinas $i \in [m]$ y configuración s:

$$\begin{aligned} (\mathcal{V}_{i,s}) \quad & \text{máx} \quad V(i-1, s-r) + f(p(r)), \\ & \text{s.a} \quad r \text{ configuración tal que } r \leq s \text{ coordenada a coordenada.} \end{aligned}$$

Definiendo $V(i, s)$ como el valor del máximo. Este método toma tiempo $O(mn^{\lambda^2})$. En la sección A.3 se retoma y detalla este método extendiéndolo al caso de máquinas distintas.

La segunda forma de resolver la instancia $I^{\#}$ es usando programación lineal entera a dimensión constante. Dada una configuración s, la variable $x_s \in \mathbb{N}$ indica la cantidad de máquinas que son asignadas la configuración s. La suma de todas las variables x_s debe ser m, indicando que las m máquinas son usadas. Además, la suma de todas las configuraciones usadas debe coincidir con la configuración total de la instancia. Esto define un programa lineal entero de dimensión y cantidad de ecuaciones constante que no depende de la entrada, solo de λ asociado a la precisión buscada. Usando el algoritmo de Lenstra [22] esto se puede resolver en tiempo exponencial en la dimensión del programa pero polilogarítmico en el valor de los coeficientes (que están acotados por m y n). El tiempo de construcción del programa, al ser lineal en n, domina el de resolución, por lo que este método requiere tiempo $O(n)$.

Finalmente, a partir de la solución óptima $\text{ALG}^{\#}$ para $I^{\#}$ se construye una asignación ALG para la instancia original I. En ALG están todos los trabajos largos asignados de la

misma manera que sus versiones redondeadas en ALG$^{\#}$. Sean $s_i^{\#}$ la cantidad de trabajos de tiempo de proceso $\overline{p}/\lambda$ asignados a la máquina i en ALG$^{\#}$. Se asignan en cualquier orden trabajos cortos a cada máquina i, hasta que superan una carga de $(s_i^{\#} - 2)\overline{p}/\lambda$. Los trabajos cortos que sobren, se asignan de nuevo en cualquier orden sin que la carga de trabajos cortos supere $(s_i^{\#} + 1)\overline{p}/\lambda$. Esto termina el algoritmo.

Alon et al. prueban que el algoritmo recién descrito produce una asignación de la instancia I cuyo valor es de a lo más $(1 + \varepsilon)$ veces el valor óptimo para el problema (I). Probando así la existencia de un **EPTAS** para (I), y enunciando las adaptaciones necesarias para hacerlo funcionar en los demás problemas.

Teorema A.1 (Alon et al. [2])**.** *Para toda función f convexa y no negativa que satisface $(F*)$, los problemas de agendamiento de minimizar $\sum_{i=1}^{m} f(p(A_i))$ y $\max_{i=1}^{m} f(p(A_i))$ poseen un* **EPTAS** *a tiempo lineal en la entrada pero a tiempo exponencial en el recíproco de la precisión deseada.*

Teorema A.2 (Alon et al. [2])**.** *Para toda función f cóncava y no negativa que satisface $(F*)$, los problemas de agendamiento de minimizar $\sum_{i=1}^{m} f(p(A_i))$ y $\max_{i=1}^{m} f(p(A_i))$ poseen un* **EPTAS** *a tiempo lineal en la entrada pero a tiempo exponencial en el recíproco de la precisión deseada.*

A.2. Simplificacion en el Caso Cóncavo

Por la demostración implícita del Teorema A.2, mejoras potenciales propias del caso cóncavo son pasadas por alto. En este capítulo veremos en detalle los problemas de maximización bajo función cóncava (I') y (II').

En la propiedad $(F*)$, la dependencia de δ en función de ε podría llegar a imposibilitar la implementación para algunas funciones de productividad. El siguiente resultado nos dice que la hipótesis de que f cumpla $(F*)$ es redudante, y simplifica la implementación del esquema, justificando que basta tomar $\delta = \varepsilon$.

Proposición A.3. *Si f cóncava y no negativa, entonces cumple la propiedad $(F*)$.*

Demostración. Por el Lema 2.9, f es superhomogénea. Sea $\varepsilon > 0$, tomamos $\delta = \varepsilon$. Sean $x, y \geq 0$, tales que $(1 + \varepsilon)x \geq y \geq (1 - \varepsilon)x$. Por monotonía y superhomogeneidad para $y \geq (1 - \varepsilon)x$ y $\frac{1}{1+\varepsilon} y \leq x$, se concluye: $(1 + \varepsilon)f(x) \geq f(y) \geq (1 - \varepsilon)f(x)$. □

Con esto se quita la condición extra del Teorema A.2.

Corolario A.4. *Para toda función f cóncava y no negativa, los problemas de agendamiento de maximizar $\sum_{i=1}^{m} f(p(A_i))$ y $\min_{i=1}^{m} f(p(A_i))$ poseen un* **EPTAS** *a tiempo lineal en la entrada pero a tiempo exponencial en el recíproco de la precisión deseada.*

Cabe destacar que la implicancia de la Proposición A.3 no se tiene en el caso convexo.

Ejemplo A.5. La función definida para $x \in \mathbb{R}_+$ por $f(x) = 2^x$ es convexa y no negativa, pero no satisface la propiedad $(F*)$.

Más aún, Alon et al. prueban que el problema (I) con la función objetivo 2^x no admite un **PTAS** ni aproximaciones a factor constante. Justificando así la necesidad de la hipótesis $(F*)$ en el caso convexo.

También se observa que *Machine Covering* es un caso particular de (II') tomando f como la identidad. Más aún, se puede probar que estos problemas son en cierto sentido equivalentes.

Proposición A.6. *Sea A una asignación que sea una α-aproximación de* Machine Covering*, entonces A es una α-aproximación del problema de agendamiento de maximizar* $\min_{i=1}^{m} f(p(A_i))$.

Demostración. Sea opt el valor óptimo de *Machine Covering* y opt_f el de (II'). Por la Proposición 2.11, f es creciente, por lo que una asignación óptima para *Machine Covering*, también lo es para (II'). Así $\text{opt}_f = f(\text{opt})$. Sea A una asignación de valor alg para *Machine Covering* tal que $\text{alg} \geq \alpha\text{opt}$. Por monotonía y superhomogeneidad, su valor para (II') es: $\text{alg}_f = f(\text{alg}) \geq \alpha f(\text{opt}) = \alpha\text{opt}_f$. □

Esto además de implicar que el **EPTAS** de *Machine Covering* sirve para el problema más general, también nos dice que (II') admite un **EPTAS** return-oblivious, pues no se necesita acceso a la función f.

A.3. Programa Dinámico con Cantidad Constante de Largos Distintos

Si bien el esquema con mejor tiempo de ejecución de Alon et al. [2] usa programas lineales a dimensión constante, su solución con programación dinámica puede ser generalizada fácilmente al caso de máquinas distintas. Es decir, cuando cada máquina i tiene su propia función de productividad $f_i : \mathbb{R} \to \mathbb{R}$, y se busca maximizar la productividad total $\sum_{i\in[m]} f_i(p(\text{ALG}_i))$. En esta sección profundizamos en dicho programa dinámico, el cual es a tiempo polinomial si la cantidad de largos distintos es constante. Este podría ser el primer paso para obtener un **PTAS** para *Machine Productivity* con máquinas distintas.

A.3.1. Preliminares

Para aprovechar que trabajos del mismo largo son indistinguibles, usaremos configuraciones de trabajos. Sea T los largos de los trabajos, es decir, $T = \{p_j \mid j \in [n]\}$. Definimos una configuración de trabajos como un vector $t = (t_h)_{h\in T} \in \mathbb{N}^T$ cuya coordenada h representa la cantidad de trabajos de ese largo en la configuración. Extendemos naturalmente la función de pesos a configuraciones como $p(t) = \sum_{h\in T} t_h h$. En este conjunto usaremos el orden parcial $\leq$, el cual equivale al orden usual en $\mathbb{N}$ coordenada a coordenada, es decir: Dos configuraciones

de trabajos s, t son tales que $s \leq t$ si y solo si $\forall h \in T, s_h \leq t_h$. Definimos el conjunto de subconfiguraciones de t por $\mathcal{T}_t = \{s \in \mathbb{N}^T \mid s \leq t\}$. Notemos que $\mathcal{T}_t = \prod_{h \in T} \{0, \ldots, t_i\}$, así, si n es la cantidad total de trabajos en la configuración y $K = |T|$ es la cantidad de largos de trabajos distintos, entonces $|\mathcal{T}_t| = \prod_{h \in T}(t_h + 1) \leq (n+1)^K = O(n^K)$. Más aún, esta cota se alcanza en orden en el caso en que los trabajos se distribuyan uniformemente entre los K tamaños distintos, ganando solo en un factor $1/K^K$, el cual es constante para todos nuestros propósitos posteriores.

Definimos además una asignación de configuraciones en m máquinas $(t^i)_{i \in [m]}$, como un vector donde la coordenada $i \in [m]$ es la configuración de trabajos t^i para la máquina i. Extendemos la noción de productividad de asignaciones de trabajos a configuraciones definiéndola como $\sum_{i \in [m]} f_i(p(t^i))$.

Sea T_h el conjunto de trabajos de largo $h \in T$, definimos la configuración de la instancia t como $\forall h \in T, t_h = |T_h|$. Notemos que toda asignación de trabajos A se puede transformar en una asignación de configuraciones $(s^i)_{i \in [m]}$ contando los trabajos de cada largo en cada máquina. En tal caso, como A define una partición, se cumple que:

$$\forall h \in T, \sum_{i \in [m]} s^i_h = \sum_{i \in [m]} |A_i \cap T_h| = |T_h|.$$

Luego esta condición es necesaria para que una asignación de configuraciones pueda relacionarse con una asignación de trabajos.

Ejemplo A.7. Consideremos una instancia en $m = 2$ máquinas con $n = 5$ trabajos de largos: $p_1 = p_2 = 8$, $p_3 = p_4 = p_5 = 11$. Entonces tenemos que $T = \{8, 11\}$ son los largos distintos y la configuración de la instancia t está dada por: $t_8 = 2$ y $t_{11} = 3$, pues hay dos trabajos de largo 8 y tres de largo 11 en la instancia. Sea la asignación A definida por $A_1 = \{1, 4, 5\}$ y $A_2 = \{2, 3\}$. Entonces, se obtiene la siguiente asignación de configuraciones: $s^1_8 = 1$, $s^1_{11} = 2$, $s^2_8 = 1$ y $s^2_{11} = 1$. Pues hay un trabajo de largo 8 y dos de largo 11 en la máquina 1, y uno de cada en la máquina 2.

Definición A.8 (Asignación de Configuraciones Válida)**.** Una asignación de configuraciones $(s^i)_{i \in [m]}$ se dice válida para una configuración de trabajos t si $\forall h \in T, \sum_{i \in [m]} s^i_h = t_h$.

Observación A.9. Trivialmente, $(s^i)_{i \in [m]}$ es válida para la configuración de trabajos $\sum_{i \in [m]} s^i$.

Salvo que de ser necesario, omitiremos la configuración con respecto a la cual la asignación es válida, entendiendo que se puede deducir del contexto.

Lema A.10. *Sea A una asignación de trabajos, entonces se puede encontrar una asignación de configuraciones válida de igual productividad.*

Demostración. Sea $(s^i)_{i \in [m]}$ obtenida contando los trabajos de cada largo en cada máquina, por lo anterior, esta asignación es válida. Veamos que tiene igual productividad. Nos fijamos en cada máquina $i \in [m]$, como $\{T_h\}_{h \in T}$ es una partición:

$$p(A_i) = \sum_{j \in A_i} p_j = \sum_{h \in T} \sum_{j \in A_i, p_j = h} h = \sum_{h \in T} |A_i \cap T_h| h = \sum_{h \in T} s^i_h h = p(s^i). \qquad \square$$

El siguiente lema nos dice que esta condición es suficiente para recuperar alguna asignación de trabajos equivalente.

Lema A.11. *Sea $(s^i)_{i\in[m]}$ una asignación de configuraciones válida, entonces se puede encontrar una asignación de trabajos de igual productividad en $O(n)$.*

Demostración. Veamos primero que esto es posible. Para cada clase $h \in T$, como las cantidades $(s_h^i)_{i\in[m]}$ suman $t_h = |T_h|$, se puede particionar T_h arbitrariamente en estas cantidades, asignando s_h^i trabajos de largo h a la máquina i.

De manera algorítmica: Se comienza con una asignación A vacía. Para cada máquina $i \in [m]$ y clase $h \in T$ se agregan s_h^i trabajos cualesquiera de T_h a A_i, cada uno en $O(1)$. Al final, se habrán asignado $\sum_{i\in[m]} s_h^i = |T_h|$ trabajos, asignando exactamente todos los trabajos de cada clase a alguna máquina. Necesitando una cantidad de iteraciones de:

$$\sum_{i\in[m]} \sum_{h\in T} s_h^i = \sum_{h\in T} |T_h| = n.$$

Además, en cada máquina $i \in [m]$: $p(A_i) = \sum_{h\in T} s_h^i h = p(s^i)$. Evaluando en la función de productividad de cada máquina y sumando se obtiene la equivalencia de productividad. □

Ejemplo A.12. Consideremos la misma instancia anterior. Sea la siguiente asignación de configuraciones s: $s_8^1 = 1$, $s_{11}^1 = 2$, $s_8^2 = 1$ y $s_{11}^2 = 1$. Notemos que s es válida, pues $s_8^1 + s_8^2 = 2 = t_8$ y $s_{11}^1 + s_{11}^2 = 3 = t_{11}$. Apliquemos el algoritmo del lema para recuperar una asignación de trabajos B. A la máquina 1 debemos asignar $s_8^1 = 1$ trabajo de largo 8, arbitrariamente elegimos el trabajo 1 y $s_{11}^1 = 2$ de largo 11, arbitrariamente 3 y 4. Por lo tanto $B_1 = \{1, 3, 4\}$. Asignando a la máquina 2 se obtiene $B_2 = \{2, 5\}$.

Notemos que esta asignación es distinta a la vista anteriormente A, de la cual se obtuvo la asignación de configuraciones s. Pero aun así estas tres asignaciones, A, B y s tienen la misma productividad, más aún tienen las mismas cargas asociadas a cada máquina. En efecto, para la máquina 1: $p(A_1) = p_1+p_4+p_5 = 8+11+11 = 30$, $p(A_1) = p_1+p_3+p_4 = 8+11+11 = 30$, $p(s_1) = s_8^1 \cdot 8 + s_{11}^1 \cdot 11 = 8 + 2 \cdot 11 = 30$. Análogamente para la máquina 2 se obtiene que en los tres casos hay una carga de 19.

Así vemos la utilidad de definir el problema auxiliar de encontrar la asignación de configuraciones válida $(s^i)_{i\in[m]}$ que maximice la productividad $\sum_{i\in[m]} f_i(p(s^i))$.

$$(\mathcal{P}_{m,s}) \quad \text{máx} \quad \sum_{i\in[m]} f_i(p(s^i)),$$
$$\text{s.a} \quad (s^i)_{i\in[m]} \text{ asignación de configuraciones válida para } s.$$

Corolario A.13. *Una asignación de trabajos en m máquinas con configuración de trabajos s es óptima si y solo si su asignación de configuraciones asociada es óptima para $(\mathcal{P}_{m,s})$.*

Demostración. En efecto, sea OPT una asignación de trabajos en m máquinas óptima y su asignación de configuraciones $(t^i)_{i\in[m]}$ asociada. Por el Lema A.10 es válida y tiene la misma

productividad. Sea $(s^i)_{i\in[m]}$ una asignación de configuraciones válida en m máquinas. Por el Lema A.11, existe A asignación de igual productividad. Entonces se tiene que la productividad de $(t^i)_{i\in[m]}$ es igual a la de OPT, esta es mayor o igual a la de A, la cual es igual a la de $(s^i)_{i\in[m]}$. Por lo tanto $(t^i)_{i\in[m]}$ es óptima.

Recíprocamente, sea $(t^i)_{i\in[m]}$ una asignación de configuraciones óptima. Por el Lema A.11, existe J asignación de igual productividad. Sea A una asignación de trabajos en m máquinas y su asignación de configuraciones $(s^i)_{i\in[m]}$ asociada. Por el Lema A.10 es válida y tiene la misma productividad. Entonces se tiene que la productividad de J es igual a la de $(t^i)_{i\in[m]}$, esta es mayor o igual a la de $(s^i)_{i\in[m]}$, la cual es igual a la de A. Por lo tanto J es óptima. □

A.3.2. El Programa Dinámico

El programa dinámico presentado en el Algoritmo 4, que aplica al caso de máquinas distintas, llena las siguientes matrices dinámicas con índices en $[m] \times \mathcal{T}_t$: P, cuya coordenada (i, s) representa el valor de productividad óptima en las primeras i máquinas y con los trabajos indicados por la configuración s. C, cuya coordenada (i, s) corresponde a una configuración de trabajos tal que existe una asignación de configuraciones óptima en las primeras i máquinas y con los trabajos indicados por la configuración s que la asigna a la máquina i. Ambas matrices se van llenando respetando las fórmulas recursivas $\forall i \in [m], \forall s \in \mathcal{T}_t$: Primero $P(1, s) = f_1(p(s))$ y $C(1, s) = s$. Luego, sea el problema de optimización:

$$\begin{aligned}(\mathcal{R}_{i,s}) \quad \text{máx} \quad & P(i-1, s-r) + f_i(p(r)),\\ \text{s.a} \quad & r \leq s.\end{aligned}$$

Se definen $P(i, s)$ y $C(i, s)$ como el valor y el argumento del máximo, respectivamente.

Finalmente se recupera $(t^i)_{i\in[m]}$, una asignación de configuraciones óptima en m máquinas para la configuración t recursivamente desde $i = m$ mediante la fórmula $t^i = C(i, t^{i+1} - r)$. Esta luego es convertida en una asignación de trabajos como se indica en el Lema A.11.

Teorema A.14. *El Algoritmo 4 resuelve* Machine Productivity *con máquinas distintas en tiempo $O(mn^{2K})$, con K la cantidad de largos de trabajos distintos.*

Demostración. Veamos la correctitud del algoritmo. Por inducción en $i \in [m]$, veamos que $\forall s \in \mathcal{T}_t$, $P(i, s)$ es una cota superior de $(\mathcal{P}_{i,s})$. Para $i = 1$, la única asignación válida es $s^1 = s$, pues se debe tener que $\forall h \in T, s^1_h = s_h$. Así, el valor óptimo de productividad es $f_1(p(s)) = P(1, s)$. Para $i > 1$ suponemos que $\forall s \in \mathcal{T}_t$, $P(i-1, s)$ es una cota superior de $(\mathcal{P}_{i-1,s})$. Sea $P(i, s)$ el valor del máximo de $(\mathcal{R}_{i,s})$. Sea $(s^k)_{k\in[i]}$ factible para $(\mathcal{P}_{i,s})$, entonces es válida, lo que implica que $s^i \leq s$, y que entre las primeras $i-1$ máquinas se asigna la configuración $s - s^i$. Juntando la hipótesis inductiva con la definición recursiva de $P(i, s)$, se concluye que:

$$\sum_{k\in[i]} f_k(p(s^k)) \leq P(i-1, s-s^i) + f_i(p(s^i)) \leq P(i, s).$$

Notemos que como $0 \leq C(i, s) \leq s$, entonces $0 \leq s - C(i, s) \leq s$. Así, $(t^i)_{i\in[m]}$ está bien definida. Además, como $C(1, s) = s$, entonces $t^1 = t - \sum_{i>1} t^i$, así $t = \sum_{i\in[m]} t^i$, por lo

Algoritmo 4: Programa Dinámico para *Machine Productivity* con máquinas distintas.

Entrada: $\langle m, n, p, \{f_i\}_{i\in[m]}\rangle$ donde $m, n \in \mathbb{N}$ cantidad de máquinas y trabajos, $p : [n] \to \mathbb{Q}_+$ tiempos de proceso y $\forall i \in [m], f_i : \mathbb{Q}_+ \to \mathbb{Q}_+$ función de productividad de la máquina i.
Salida: $\text{ALG} = \{\text{ALG}_i\}_{i\in[m]}$ asignaciones de trabajos a máquinas.

```
Particionar [n] en conjuntos T_h con los trabajos de largo h ∈ T
Sean t ← (|T_h|)_{h∈T}, T_t ← {s ∈ ℕ^T | s ≤ t} y P y C matrices de índices en [m] × T_t
para s ∈ T_t hacer
    P(1, s) ← f_1(p(s)) y C(1, s) ← s
fin
para i desde 2 a m hacer
    para s ∈ T_t hacer
        Sea (R_{i,s}) el problema de optimización máx{P(i − 1, s − r) + f_i(p(r)) | r ≤ s}
        Sean P(i, s) y C(i, s) el valor y el argumento del máximo de (R_{i,s}), respectivamente
    fin
fin
t^m ← C(m, t)
para i desde m − 1 a 1 hacer
    t^i ← C(i, t − Σ_{k>i} t^k)
fin
para i ∈ [m], h ∈ T hacer
    ALG_i ← t^i_h trabajos cualesquiera de T_h
fin
devolver ALG
```

que es una asignación válida para la configuración t. Es directo que $(t^k)_{k\in[i]}$ es válida para la configuración $\sum_{k\in[i]} t^k$. Por inducción en $i \in [m]$, veamos que $(t^k)_{k\in[i]}$ es una asignación de configuraciones valor $P(i, \sum_{k\in[i]} t^k)$. Para $i = 1$, la asignación t^1 tiene valor $f_1(p(t^1)) = P(1, t^1)$. Para $i > 1$ suponemos que $\forall s \in \mathcal{T}_t$, $(t^k)_{k\in[i-1]}$ es una asignación de valor $P(i - 1, \sum_{k\in[i-1]} t^k)$. Como $t - \sum_{k>i} t^k = \sum_{k\in[i]} t^k$, entonces por construcción $t^i = C(i, \sum_{k\in[i]} t^k)$. Aplicando la hipótesis inductiva, se concluye:

$$P\left(i, \sum_{k\in[i]} t^k\right) = P\left(i-1, \sum_{k\in[i-1]} t^k\right) + f_i(p(t^i)) = \sum_{k\in[i]} f_k(p(t^k)).$$

Por lo tanto, $(t^i)_{i\in[m]}$ es óptima para $(\mathcal{P}_{i-1,s})$. Por el Lema A.11, tomar t^i_h trabajos cualesquiera de T_h y asignarlos a la máquina i da una asignación de trabajos de igual productividad, y por el Corolario A.13 esta asignación de trabajos es óptima.

Veamos ahora la complejidad en tiempo del programa. Primero, para particionar los trabajos basta una pasada e irlos asignando a su clase $\mathcal{T}_h$ según su tamaño h, por lo que es $O(n)$. Segundo, para llenar las matrices P y C se ejecutan $m|\mathcal{T}_t| = O(mn^K)$ ciclos, en cada uno calculando un máximo entre $|\mathcal{T}_t| = O(n^K)$ maneras de separar la configuración s. Por lo que estas construcciones se realizan en $O(mn^{2K})$. Tercero y finalmente, deducir la asignación de configuraciones óptima toma $O(m)$ y recuperar una asignación de trabajos requiere $O(n)$, como indica el Lema A.11. Por lo tanto, el algoritmo es a tiempo $O(mn^{2K})$, el cual sería polinomial si K fuera una constante.

Por completitud veamos la complejidad en espacio y oracular del algoritmo. En espacio el algoritmo necesita recordar las matrices matriz P y C, de tamaños $m|\mathcal{T}_t|$ y $m|\mathcal{T}_t|K$, por

lo que es a espacio $O(mn^K)$. Por otro lado, observar que la cantidad de llamados oraculares a $\{f_i\}_{i\in[m]}$ es $O(mn^K)$, pues para evitar llamados innecesarios se pueden registrar las evaluaciones $f_i(p(s))$ para cada $i \in [m]$ y $s \in \mathcal{T}_t$ en tiempo y espacio $O(mn^K)$. □