ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE CIENCIAS

ALGORITMOS DE APROXIMACIÓN PARA UN PROBLEMA MULTI-PERIODO DE CALENDARIZACIÓN DE MAQUINAS PARALELAS

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGÍSTER EN OPTIMIZACIÓN MATEMÁTICA

PROYECTO DE INVESTIGACIÓN

FERNANDO GERMÁN JIMÉNEZ TORRES

fer-y-nando@hotmail.com

DIRECTOR: DR. LUIS MIGUEL TORRES CARVAJAL

luis.torres@epn.edu.ec

Quito, febrero 2023

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Fernando Germán Jiménez Torres, bajo mi supervisión.

Dr. Luis Miguel Torres Carvajal
Director del Proyecto

DECLARACIÓN

Yo, Fernando Germán Jiménez Torres, declaro bajo juramento que el trabajo aquí escrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual, correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normatividad institucional vigente.

Fernando Germán Jiménez Torres

DEDICATORIA

A mis padres y amigos.

AGRADECIMIENTOS

Primeramente, agradezco a mis padres por su apoyo, en especial a mi madre. Después, agradezco a Luis Miguel por ser uno de los mejores profesores en su área y apoyarme en el desarrollo de este trabajo. Además, agradezco a Diego Recalde, Ramiro Torres y demás profesores que me han ayudado en mi desarrollo profesional. Finalmente agradezco a todas esas personas que se han atrevido a formar parte de mi vida tanto personal como profesional.

Índice general

Índice de figuras

Índice de tablas

Resumen

El problema de calendarización de trabajos en máquinas paralelas, con tiempos de llegada y plazos de ejecución (SRDM, Scheduling with Release times and Deadlines on a minimum number of Machines) se lo puede definir de la siguiente forma: Dados n trabajos, cada uno asociado con un tiempo de llegada, un plazo máximo de ejecución y una duración; ¿cuál es el menor número de máquinas idénticas que se necesitan para ejecutar todos los trabajos antes de su plazo máximo de ejecución ? Este problema es difícil de aproximar en el caso general, sin embargo, para casos particulares se han probado diferentes factores de aproximabilidad.

El estudio de los problemas de calendarización de intervalos comenzó en los años cincuenta del siglo pasado y entre ellos se encuentra el problema JISP (Job Interval Selection Problem). A diferencia del problema SRDM, el problema JISP tiene como objetivo ejecutar la mayor cantidad de trabajos en un número fijo de máquinas.

En este trabajo se propone el estudio de una versión multi-periodo del problema SRDM, en la cual se requiere procesar un conjunto dado de trabajos, minimizando la cantidad total de periodos-máquina empleados para el efecto. Se propone adaptar un algoritmo de aproximación empleado para el problema JISP y dos algoritmos de aproximación conocidos para el problema SRDM a este nuevo problema, y analizar su comportamiento, tanto en lo que respecta al factor de aproximabilidad de los algoritmos, como a su desempeño en pruebas computacionales.

Palabras claves: JISP
SRDM
Programación lineal entera
Algoritmos de aproximación
Calendarización de intervalos

Abstract

The Scheduling with Release times and Deadlines on a minimum number of Machines (SRDM) problem can be defined as follows: Given n jobs, each one associated with a release time, a deadline, and a processing time; what is the minimum number of identical machines needed to run all the jobs before their deadline? This problem is difficult to approximate in the general case, however, different approximation factors have been established for particular cases.

The study of interval scheduling problems began in the 1950s and one of them is the Job Interval Selection Problem (JISP). Unlike the SRDM problem, the JISP aims to execute as many jobs as possible on a fixed number of machines.

The aim of this thesis is to study a multi-period version of SRDM, in which some given jobs with release times and deadlines have to be scheduled minimizing the number of required machine-periods. We investigate the application of an approximation algorithm used for the JISP and two approximation algorithms for the SRDM to the solution of this new problem. We establish bounds on the approximability factor of the proposed algorithms and test their performance on computational experiments.

Keywords: JISP
SRDM
Linear Integer Programming
Approximation algorithms
Interval Scheduling

Capítulo 1

Introducción

1.1. Problema de asignación de turnos a ventanillas

Proveer de una buena atención a los clientes en los diferentes puntos de servicio constituye un problema común en las instituciones, tanto públicas como privadas. Evidentemente, es importante evitar o minimizar el daño a la imagen institucional que se causa cuando los usuarios deben realizar largas filas o esperar una cantidad considerable de tiempo por sus trámites. Por otra parte, las soluciones al problema deben tomar en consideración también la limitada disponibilidad de recursos humanos y materiales. El problema abordado en esta tesis está motivado por un proyecto de cooperación previo con el Servicio de Rentas Internas del Ecuador (SRI), cuyo objetivo es establecer modelos que permitan simular y optimizar los tiempos de atención a los usuarios en las distintas sucursales de la entidad. En cada sucursal, debe decidirse la cantidad óptima de ventanillas a mantener abiertas en cada hora a lo largo del día, para poder atender a los usuarios que arriban a realizar sus trámites, garantizando un tiempo de espera no superior a 20 minutos. Cuando los empleados no están atendiendo ventanillas, se dedican a otras tareas internas asignadas, por lo que es importante minimizar la cantidad de horas-ventanilla requeridas en total en el día. Denotaremos a este problema como Problema de Asignación de Turnos a Ventanillas (PATV).

1.2. Problema de calendarización de trabajos con tiempos de inicio y plazos de ejecución minimizando el número de máquinas disponibles

El problema de calendarización de trabajos en máquinas paralelas, con tiempos de llegada y plazos de ejecución (SRDM, Scheduling with Release times and Deadlines on a minimum number of Machines) se lo describe en [6] de la siguiente forma: Se tienen dados n trabajos y m máquinas disponibles, cada trabajo $j \in J = \{1, 2, \cdots, n\}$ cuenta con un tiempo de llegada (r_j, release date), un tiempo de ejecución (p_j, processing time) y un plazo máximo para finalizar su ejecución (d_j, deadline). Por otra parte, cada una de las m máquinas puede estar encendida o apagada. Una máquina puede procesar trabajos solamente si está encendida, y cada máquina puede procesar un solo trabajo a la vez. El problema consiste en decidir cuáles máquinas deben estar encendidas, y cómo asignar los trabajos a estas máquinas, de tal forma que sea posible atender todos los trabajos dentro de su ventana de tiempo $[r_j, d_j)$, empleando la menor cantidad de máquinas.

Los problemas de calendarización han sido estudiados por más de 50 años. Una de las variantes que ha recibido bastante atención es el problema clásico de calendarización de intervalos (interval scheduling), el cual surge como una motivación del problema de gestión de operaciones. Su estudio comenzó en los años cincuenta con Dantzing y Fulkerson con un problema de planificación de buques para cumplir con un horario fijo (tanker scheduling problem). Este problema consiste en determinar el menor número de buques requeridos para cubrir un calendario de transporte de petróleo con varios puntos de carga y descarga preestablecidos [10, 19].

Dentro de los problemas de calendarización de intervalos se encuentra el problema de selección de intervalos de trabajo (JISP, Job Interval Selection Problem), cuya modelización resulta muy sencilla pero a su vez es muy interesante. En este problema se considera un conjunto de n trabajos y m máquinas disponibles, cada trabajo cuenta con un conjunto determinado de intervalos en los cuales pueden ser procesados. Estos intervalos pueden ser listados de forma explícita o implicados por otros parámetros que se definen junto con cada trabajo. Se quiere seleccionar a lo más uno de los intervalos

de cada trabajo de tal forma que para cada instante de tiempo dentro del horizonte de tiempo a lo más m intervalos de trabajos diferentes se encuentran procesando simultáneamente. El objetivo del problema consiste en seleccionar tantos trabajos como sean posibles [5, 25].

Algunas aplicaciones motivadas por el problema JISP son: fabricación de placas de circuitos impresos [13, 16], selección de proyectos a realizar durante una misión espacial [15], programación de comunicaciones con restricciones de tiempo en redes lineales [1], programación de la tripulación [22], y asignación de ancho de banda con preferencias [3]. Otras aplicaciones adicionales de los problemas de asignación de intervalos se encuentran descritas en [19].

1.3. Complejidad de los problemas de calendarización

La Optimización Combinatoria es un área de la Investigación Operativa que se ocupa de resolver problemas donde se requiere encontrar, en un tiempo limitado, la solución óptima para una función objetivo dada, de entre un conjunto finito, pero muy grande, de soluciones factibles [23]. Para resolver un problema necesitamos elaborar un algoritmo. En primer lugar, este algoritmo debe ser correcto, es decir, debe encontrar la respuesta esperada para cada instancia del problema. Además, este algoritmo debe ser eficiente. Fueron Cook y Levin quienes establecieron ciertos criterios que aún hoy son ampliamente utilizados dentro de la Teoría de Complejidad Computacional para calificar cuándo un algoritmo resuelve un problema de forma eficiente [8, 21]. Un algoritmo es eficiente cuando el tiempo requerido para resolver cada instancia del problema está acotado por un polinomio en la cantidad de memoria requerida para almacenar los datos de esa instancia en un computador, conocida como su longitud de codificación. A estos algoritmos se los llama también polinomiales, y decimos que un problema es polinomial cuando existe al menos un algoritmo polinomial para el mismo. Ejemplos de problemas polinomiales son: los problemas de flujo máximo, corte mínimo, camino más corto, árbol generador de peso mínimo (MST), entre otros [17, 23]. Sin embargo, así mismo hay problemas que resultan más difíciles de resolver desde el punto de vista

computacional. En particular, en [8, 17, 21] se establecen los criterios fundamentales para determinar la existencia de una clase particularmente difícil de problemas, conocidos como problemas NP-difíciles. La existencia de un algoritmo polinomial para cualquier problema de esta clase implicaría la existencia de algoritmos polinomiales para todos los problemas de una clase muy grande de problemas, conocida como NP, que contiene a aquellos problemas de decisión (problemas con respuesta "sí" o "no") cuyas instancias afirmativas tienen asociados siempre certificados que puede validarse con un algoritmo polinomial. Dado que casi cualquier problema de optimización combinatoria puede asociarse a un problema de decisión en NP, la existencia de un algoritmo polinomial para todos los problemas de esta clase es algo contra intuitivo, aunque este hecho no ha podido ser aún demostrado, ni tampoco descartado teóricamente. (La conjetura P vs NP es uno de los Problemas del Milenio [9]). Si se asume que las clases P y NP son de hecho distintas, entonces ningún problema de NP-difícil admite un algoritmo polinomial. Entre los problemas NP-difíciles se encuentran el problema de coloramiento de grafos, problema del agente viajero, problema del camino Hamiltoneano, clique máxima en un grafo, árboles de Steiner, etc [17, 23].

La complejidad computacional del problema SRDM ha sido abordada en trabajos previos. Por un lado, si la duración de cada trabajo es igual a la longitud de la ventana de tiempo asociada al trabajo, el problema SRDM se puede reducir a un problema clásico de asignación de intervalos, el cual a su vez se puede reescribir como un problema de coloramiento de nodos sobre un grafo de intervalos [19]. Adicionalmente, en [6], se prueba que el problema SRDM es polinomial para el caso en el que la diferencia entre la longitud de la ventana de tiempo y la duración de cada trabajo es a lo más uno. Finalmente, se demuestra que el problema es NP-difícil para el caso en el que dicha diferencia es mayor o igual a dos.

En la práctica, los problemas NP-difíciles se abordan mediante dos enfoques. Por un lado, se emplean métodos heurísticos y de aproximación, es decir, se resuelven mediante algoritmos que no garantizan encontrar la solución óptima, pero sí una solución aceptable en tiempos razonables. Por otra parte, se utilizan esquemas basados en la programación lineal entera y en métodos de ramificación-y-acotación (branch-and-bound) para encontrar soluciones factibles con una brecha de optimalidad garantizada. El problema SRDM ha sido abordado bajo los dos enfoques anteriores [6, 19, 20].

1.4. Una versión multi-periodo del problema SRDM

Sea $J = \{1, 2, \cdots, n\}$ un conjunto de n trabajos que se desea ejecutar y sea $M = \{1, 2, \cdots, m\}$ un conjunto de m máquinas disponibles. Cada trabajo $j \in J$ tiene asociado un tiempo de llegada (r_j, release date), un tiempo de ejecución (p_j, processing time) y un plazo máximo para finalizar su ejecución (d_j, deadline). Para cada trabajo $j \in J$, se considera a $\gamma_j = d_j - r_j - p_j$ como el máximo tiempo que j puede esperar antes de iniciar su procesamiento. El horizonte de planificación se encuentra dividido en T periodos de igual duración (L unidades de tiempo cada uno). Denotamos por $P = \{1, 2, \cdots, T\}$ al conjunto de periodos. Cada máquina puede ser encendida o apagada en cualquier periodo y solo puede procesar un trabajo si se encuentra encendida. Adicionalmente, una máquina encendida no puede procesar más de un trabajo simultáneamente, y no existen limitaciones sobre el número de trabajos que una máquina puede procesar a lo largo del horizonte de planificación.

El problema multi-periodo de calendarización de trabajos con tiempos de llegada y plazos de ejecución minimizando el número de máquinas disponibles (SRDM-T) consiste en decidir cuáles máquinas deben ser encendidas en cada periodo y cómo deben asignarse los trabajos a estas máquinas de tal forma que se procesen todos los trabajos dentro de su ventana de tiempo. El objetivo es minimizar la cantidad total de periodos-máquina, es decir, minimizar la suma del número de máquinas encendidas sobre todos los periodos.

En la presente tesis nos enfocaremos en el estudio del SRDM-T con intervalos discretos. En esta variante del problema, asumimos que los tiempos de llegada r_j de los trabajos, sus duraciones p_j y sus plazos máximos de ejecución d_j son cantidades enteras. Adicionalmente, requerimos que el tiempo de inicio del procesamiento de cada trabajo que denotaremos por τ_j sea una cantidad entera dentro del conjunto $\{r_j, r_j + 1, \cdots, d_j - p_j\}$.

1.4.1. Problema de asignación de turnos a ventanillas modelado como un problema SRDM-T

El PATV puede ser modelado como un problema SRDM-T. Para ello, las ventanillas de la sucursal pueden ser consideradas como máquinas, y los clientes que arriban a la sucursal como trabajos a ser procesados. Cada trabajo tiene asociados un tiempo de llegada (tiempo de arribo del cliente), un plazo máximo de ejecución (correspondiente al máximo tiempo de finalización del trámite) y una duración (dada por la duración del trámite). Todos los trabajos arriban y deben ser procesados dentro de un horizonte de tiempo, que se encuentra dividido en periodos (de una hora). Todas las máquinas son idénticas y cada una de ellas puede estar encendida en cualquier periodo. Un trabajo solo puede ser ejecutado en una máquina (es decir, un cliente puede ser atendido por una única ventanilla) que se encuentra encendida durante la ventana de tiempo de procesamiento de dicho trabajo. La ventana de tiempo de procesamiento de un trabajo es el intervalo (cerrado por izquierda y abierto por derecha) cuyo extremo izquierdo es el tiempo de llegada y cuyo extremo derecho es el plazo máximo de ejecución del trabajo. El objetivo del problema consiste en decidir cuáles máquinas deben ser encendidas en qué periodos de tal forma que cada trabajo sea procesado en alguna de las máquinas y el número de periodos-máquina, durante todo el horizonte de planificación, sea el menor posible.

1.4.2. Un modelo de programación lineal entera para el SRDM-T

Formulamos a continuación un modelo de programación lineal entera para la versión del SRDM-T con intervalos discretos que estudiaremos en la presente tesis. Con esta finalidad, para cada trabajo $j \in J$, se define el conjunto:

$$\mathfrak{I}(j) := \{[r_{j,\ell}, d_{j,\ell}) \colon \ell \in [\gamma_j]\},$$

donde $r_{j,\ell} = r_j + \ell$, $d_{j,\ell} = r_{j,\ell} + p_j$, $\gamma_j = d_j - r_j - p_j$ y $[\gamma_j] = \{0, 1, \cdots, \gamma_j\}$. Por tanto, cada intervalo en $\mathfrak{I}(j)$ representa una de las posibles calendarizaciones del trabajo j. Así, en toda solución factible se debe seleccionar, para cada trabajo, exactamente un solo intervalo de este conjunto de posibles intervalos de procesamiento.

Para cada periodo $q \in P$, se notará por t_q al tiempo de inicio del periodo, y

se define $H_q = \{t_q\} \cup \{r_{j,\ell} \colon j \in J, \ell \in [\gamma_j], t_q \leq r_{j,\ell} < t_q + L\}$. Adicionalmente, sea $H = \bigcup_{q \in P} H_q$ el conjunto de todos los posibles tiempos de inicio de ejecución de los trabajos. Dos intervalos se encuentran en conflicto en el instante t si y sólo si t pertenece a ambos intervalos. Se define $J_t = \{(j, \ell) \colon j \in J, \ell \in [\gamma_j], t \in [r_{j,\ell}, d_{j,\ell})\}$ como el conjunto de todos los posibles intervalos que se encuentran en conflicto en el instante de tiempo t.

A continuación se presenta una primera formulación para el SRDM-T como modelo de programación lineal entera que emplea las siguientes variables de decisión:

$$x^k_{j,\ell} = \begin{cases} 1, & \text{si el trabajo } j \in J \text{ es procesado en la máquina } k \in M \text{ en el} \\ & \text{intervalo } [r_{j,\ell}, d_{j,\ell}) \in \mathfrak{I}(j), \\ 0, & \text{caso contrario.} \end{cases}$$

$$y_{k,q} = \begin{cases} 1, & \text{si la máquina } k \in M \text{ se enciende en el periodo } q \in P, \\ 0, & \text{caso contrario.} \end{cases}$$

Podemos formular entonces el modelo M-SRDM-T:

$$\text{mín} \sum_{k \in M} \sum_{q \in P} y_{k,q} \tag{1.4.1}$$

s.a.r.

$$\sum_{\ell \in [\gamma_j]} \sum_{k \in M} x^k_{j,\ell} = 1, \quad \forall j \in J, \tag{1.4.2}$$

$$\sum_{(j,\ell) \in J_t} x^k_{j,\ell} \leq y_{k,q}, \quad \forall\, t \in H_q,\ \forall\, k \in M,\ \forall\, q \in P, \tag{1.4.3}$$

$$x^k_{j,\ell} \in \{0, 1\}, \quad \forall j \in J,\ \forall\, \ell \in [\gamma_j],\ \forall\, k \in M,$$

$$y_{k,q} \in \{0, 1\}, \quad \forall\, k \in M,\ \forall\, q \in P.$$

La función objetivo (1.4.1) calcula la cantidad total de periodos-máquina requeridos para procesar todos los trabajos. Las restricciones (1.4.2) aseguran que cada trabajo es ejecutado en exactamente una máquina en uno de sus periodos factibles. Las restricciones (1.4.3) garantizan que los trabajos puedan ser asignados a la máquina k en el periodo q si y sólo si la máquina k se encuentra encendida en el periodo q.

En [6] se propone un algoritmo de solución para el SRDM-T que consiste en descomponer el problema en dos fases. En la primera fase se determina el número de

máquinas que deben estar encendidas en cada periodo; mientras que en la segunda fase se realiza la asignación de los trabajos a estas máquinas.

Para resolver la primera etapa se propone una versión simplificada del modelo de programación lineal entera descrito arriba. Definamos las siguientes variables de decisión:

$$x_{j,\ell} = \begin{cases} 1, & \text{si el trabajo } j \in J \text{ es procesado en alguna máquina en su intervalo } [r_{j,\ell}, d_{j,\ell}) \in \mathfrak{I}(j), \\ 0, & \text{caso contrario.} \end{cases}$$

$z_q \in \mathbb{Z}_+$, es el número de máquinas encendidas en el periodo $q \in P$.

La versión simplificada del modelo M-SRDM-T (S-SRDM-T) puede entonces escribirse como:

$$\text{mín} \sum_{q \in P} z_q \tag{1.4.4}$$

s.a.r.

$$\sum_{\ell \in [\gamma_j]} x_{j,\ell} = 1, \quad \forall j \in J, \tag{1.4.5}$$

$$\sum_{(j,\ell) \in J_t} x_{j,\ell} \leq z_q, \quad \forall\, t \in H_q,\ \forall\, q \in P, \tag{1.4.6}$$

$$x_{j,\ell} \in \{0,1\}, \quad \forall j \in J,\ \forall\, \ell \in [\gamma_j],$$

$$z_q \in \{0,1,\cdots,m\}, \quad \forall\, q \in P.$$

La función objetivo (1.4.4) calcula la cantidad total de periodos-máquina requeridos para procesar todos los trabajos. Las restricciones (1.4.5) aseguran que cada trabajo sea ejecutado dentro de uno de sus intervalos factibles. Las restricciones (1.4.6) garantizan que el número de trabajos procesados en cualquier instante de tiempo dentro del periodo q no exceden el número de máquinas encendidas en dicho periodo.

Una vez calculado el número de máquinas que deben ser encendidas en cada periodo usando el modelo S-SRDM-T, en la segunda fase se calcula una asignación de trabajos a las máquinas encendidas. Este problema puede reducirse a un problema de coloreo de grafos de intervalos, el mismo que puede ser resuelto de manera eficiente, por ejemplo, empleando el coloreo glotón del grafo de intervalos descrito con mayor detalle en [11].

Debido a que el problema de asignación de intervalos a máquinas encendidas se lo puede resolver de manera óptima mediante un algoritmo de coloreo glotón, nos

enfocaremos en esta tesis en la solución del problema simplificado S-SRDM-T. De hecho, en la mayoría de trabajos relacionados con el problema SRDM se centran en resolver el problema de determinar cuál es la menor cantidad de máquinas que deben usarse para ejecutar los trabajos, y no en la asignación en sí de trabajos a máquinas.

Observemos, además, que toda solución factible del problema SRDM-T consiste en asignar a cada trabajo $j \in J$ una máquina $k \in M$ y un tiempo de inicio de ejecución $\tau_j \in [r_j, d_j - p_j)$ y decidir qué máquinas están encendidas en qué periodos, de tal forma que los trabajos solamente sean asignados a máquinas que están encendidas durante todo su periodo de ejecución $[\tau_j, \tau_j + p_j)$. Denotaremos por $\text{Sol}(\mathcal{I}) := (\mathcal{J}, \mathcal{Z})$ a una solución factible de una instancia $\mathcal{I}$ del problema SRDM-T, donde $\mathcal{J} = ([\tau_1, \tau_1 + p_1), [\tau_2, \tau_2 + p_2), \cdots, [\tau_n, \tau_n + p_n))$, $\mathcal{Z} = (z_1, z_2, \cdots, z_T)$ y z_q es la cantidad de ventanillas usadas en el periodo $q \in P$.

Definición 1.4.1 (Costo de una solución)**.** Dadas una instancia $\mathcal{I}$ del problema SRDM-T y una solución $\text{Sol}(\mathcal{I}) = (\mathcal{J}, \mathcal{Z})$, el costo de esta solución se define como la cantidad de periodos-máquina requeridos para procesar todos los trabajos en J, es decir, es la cantidad de máquinas encendidas en cada periodo sumada sobre todos los periodos:

$$\text{costo}(\mathcal{Z}) := \sum_{q \in P} \hat{z}_q,$$

donde $\hat{z}_q$ es la cantidad de máquinas encendidas en el periodo $q \in P$.

1.4.3. Organización de la tesis

Dado que el problema SRDM-T es NP-difícil, excepto para casos particulares, el objetivo general de la presente tesis es *estudiar distintos algoritmos de aproximación para el problema SRDM-T desde un punto de vista teórico y práctico*. En particular, estudiaremos tres algoritmos de aproximación para el SRDM-T basados en algoritmos similares propuestos para el problema SRDM y el problema JISP.

El resto de esta tesis está organizada de la siguiente manera:

En el Capítulo 2 se presentan algunas definiciones preliminares, notación y algoritmos empleados para resolver el problema SRDM y JISP. En el Capítulo 3 se formulan algoritmos y se analiza su factor de aproximación para resolver el problema SRDM-T. En el Capítulo 4 se presentan y discuten los resultados computacionales obtenidos en la aplicación de los algoritmos. El Capítulo 5 contiene conclusiones del trabajo.

Capítulo 2

Preliminares

En este capítulo se presentan varias definiciones preliminares y conceptos básicos, así como algunos algoritmos descritos para los problemas SRDM y JISP que serán utilizados en el desarrollo de este trabajo. En el Capítulo 3 analizaremos la posibilidad de extender los algoritmos y resultados presentados al SRDM-T.

2.1. Definiciones y notación

En esta sección se introducen definiciones y notación utilizadas en el resto de esta tesis. Emplearemos algunos conceptos definidos en [6] para el SRDM, adaptándolos a la notación empleada en el presente documento.

Dado un entero $n \in \mathbb{N}$, se nota por $[n]$ al conjunto $\{0, 1, \cdots, n\}$.

Denotaremos por $J = \{1, 2, \cdots, n\}$, a un conjunto de trabajos y al conjunto de máquinas lo denotaremos por $M = \{1, 2, \cdots, m\}$. Cada trabajo está asociado a un *tiempo de llegada* r_j, una *duración* p_j y un *plazo máximo de ejecución* d_j. En esta tesis nos enfocaremos en la versión discreta del problema de calendarización de intervalos, por lo que asumiremos que r_j, p_j y d_j son valores enteros y que el tiempo de inicio de procesamiento de cada trabajo tiene que ser un entero dentro del conjunto $\{r_j, r_j + 1, \cdots, d_j - p_j\}$. Adicionalmente, se define $\gamma_j = d_j - r_j - p_j$ como el máximo tiempo de espera permitido previo al inicio de la ejecución del trabajo $j \in J$. Denotaremos como el intervalo $[r_j, d_j)$ a la ventana de tiempo asociada al trabajo $j \in J$. Para cada periodo $q \in P$, se define como t_q al tiempo de inicio del periodo q.

Denotaremos por $\mathcal{I} := (J, P, L_P, M, M_P)$ a una instancia del problema SRDM-T,

donde J es el conjunto de trabajos, $P = \{1, 2, \cdots, T\}$ es el conjunto de periodos, $L_P = (L_1, \cdots, L_T)$ es un vector que indica la duración de cada periodo, M es el conjunto de máquinas disponibles, y $M_P = (m_1, \cdots, m_T)$ es un vector que indica el número de máquinas disponibles en cada periodo, es decir, m_q es la cantidad de máquinas disponibles en el periodo $q \in P$. Cada máquina $m \in M$ puede estar encendida o apagada en cada periodo $q \in P$. Una máquina m puede procesar un trabajo j solamente si se encuentra encendida durante todo el tiempo requerido para el procesamiento de j, y ninguna máquina puede procesar más de un trabajo simultáneamente. El problema consiste en seleccionar un tiempo válido de inicio de procesamiento para cada trabajo y definir el número z_q de máquinas a encender en cada periodo $q \in P$ (que debe ser menor o igual al número de máquinas disponibles en ese periodo), de tal forma que sea posible asignar cada trabajo a alguna máquina encendida, mientras se minimiza la suma de z_q sobre todos los periodos.

Dados n trabajos, con sus ventanas de tiempo $[r_j, d_j)$, para cada $j \in J$, se definen $r_{\text{mín}} = \min_{j \in J}\{r_j\}$, $r_{\text{máx}} = \max_{j \in J}\{r_j\}$ y $d_{\text{máx}} = \max_{j \in J}\{d_j\}$.

Dado un trabajo $j \in J$, supongamos que el trabajo j inicia su ejecución en el instante $\tau_j \in \{r_j, r_j+1, \cdots, r_j+\gamma_j\}$. Definimos como $\ell \in [\gamma_j]$ a la posición en la cual es procesado el trabajo j, es decir, $\tau_j = r_j + \ell$. Adicionalmente, se define $I_{j,\ell} = [r_{j,\ell}, d_{j,\ell})$ como el intervalo de procesamiento del trabajo j asociado a la posición ℓ, donde $r_{j,\ell} = r_j + \ell$ y $d_{j,\ell} = r_{j,\ell} + p_j$.

Dado un intervalo $I = [r, d)$, se dice que un trabajo $j \in J$ es compatible con el intervalo I si el trabajo puede ser procesado completamente dentro del intervalo de tiempo $[r, d)$, es decir, si existe una posición $\ell \in [\gamma_j]$ tal que $r \leq r_{j,\ell} < d_{j,\ell} \leq d$.

Se define el núcleo de un trabajo $j \in J$ como la intersección de los intervalos de procesamiento para todas las posiciones posibles del trabajo j. Es decir, si $\gamma_j \leq p_j$, entonces el núcleo del trabajo j es $[r_{j,\gamma_j}, d_{j,0})$, caso contrario el núcleo es vacío.

Denotaremos por $\mathcal{L} = (\ell_1, \cdots, \ell_n)$ al vector con las posiciones de cada uno de los trabajos, donde ℓ_j es la posición en la cual es procesado el trabajo j. $\mathfrak{I}^{\mathcal{L}} = (I_{1,\ell_1}, I_{2,\ell_2}, \cdots, I_{n,\ell_n})$ es un vector de intervalos descritos por los trabajos del conjunto J y el vector de posiciones $\mathcal{L}$.

Dada una instancia $\mathcal{I} = (J, P, L_P, M, M_P)$ y un vector de posiciones $\mathcal{L}$, se define la medida $h(\mathfrak{I}^{\mathcal{L}})$ como el máximo número de solapamientos de los intervalos de $\mathfrak{I}^{\mathcal{L}}$.

Adicionalmente, $h(\mathfrak{I}^{\mathcal{L}}, t)$ es el máximo número intervalos que contienen al instante de tiempo t. Por último, designaremos como $h(J)$ al menor número máximo de solapamientos, para todos los vectores de posiciones admisibles para el conjunto de trabajos, es decir:

$$h(J) := \min_{\mathcal{L}} h(\mathfrak{I}^{\mathcal{L}}).$$

Observar que una solución factible para $\mathcal{I}$ consiste de un vector de posiciones $\mathcal{L}$ y de un vector $z = (z_1, \cdots, z_T)$ que indica el número de máquinas encendidas entre cada periodo, de tal forma que:

$$z_q = \max_{t_q \leq t < t_q + L_q} h(\mathfrak{I}^{\mathcal{L}}, t), \qquad \forall q \in P.$$

El valor de la solución está dado por $\sum_{q \in P} z_q$.

El SRDM es un caso especial del SRDM-T donde el número de periodos es $T = 1$. De manera similar, se define al problema SRDM(k) como el problema SRDM con k máquinas disponibles. Es decir, para cualquier instancia $\mathcal{I} = (J, P, L_P, M, M_P)$ de este problema se tiene que $P = \{1\}$ y $M_P = (k)$. Observar que una instancia del problema SRDM(k) tiene solución afirmativa si y solamente si existe un vector de posiciones $\mathcal{L}$ tal que $h(\mathfrak{I}^{\mathcal{L}}) \leq k$.

Dada una instancia $\mathcal{I}$ del problema SRDM-T (o del problema SRDM), se define como Opt($\mathcal{I}$) al valor de una solución óptima para $\mathcal{I}$.

Sean $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^m$ y $c \in \mathbb{Q}^n$. Un programa lineal entero puro es un problema de la forma:

$$\text{(PE)} \begin{cases} \min c^T x \\ \text{s.a.r.} \\ Ax \leq b, \\ x \in \mathbb{Z}^n. \end{cases}$$

El conjunto $P_{\text{PE}} = \{x \in \mathbb{Z}^n \, : \, Ax \leq b\}$ es el conjunto de soluciones factibles para PE, y es llamado conjunto lineal entero puro.

La relajación lineal de PE es un problema de la forma:

$$\text{(PL)} \begin{cases} \min c^T x \\ \text{s.a.r.} \\ Ax \leq b, \\ x \in \mathbb{R}^n. \end{cases}$$

Notar que PL es un programa lineal y, si $P_{\text{PL}} = \{x \in \mathbb{R}^n : Ax \leq b\}$ es el poliedro formado por el conjunto de soluciones factibles para PL, entonces $P_{\text{PE}} = P_{\text{PL}} \cap \mathbb{Z}^n$ es el conjunto de soluciones factibles para PE [7].

2.2. Resultados importantes del problema SRDM

De las definiciones realizadas en la sección anterior se puede ver que el problema SRDM se reduce a encontrar un vector de posiciones $\mathcal{L}$ tal que $h(\mathfrak{I}^{\mathcal{L}})$ sea mínimo. En esta sección, revisaremos algunos resultados para este problema, los cuales han sido tomados de [6].

Un primer resultado indica que SRDM es un problema difícil de aproximar en tiempo polinomial.

Teorema 2.2.1. *No existe un algoritmo polinomial de aproximación para resolver el problema SRDM con un factor de aproximación menor que 2, a menos que $\mathcal{P} = \mathcal{NP}$.*

Demostración. En [12] se expone el problema de ordenamiento lógico con tiempos de llegada y plazos máximos de ejecución (Sequencing with release times and deadlines, SRD). Dicho problema de decisión es $\mathcal{NP}$-completo y pregunta si es posible ejecutar n trabajos en una máquina. Para cada trabajo se especifican una duración y una ventana de tiempo en la que debe ser ejecutado. El problema SRD se puede ver como la versión de decisión del SRDM: Si el SRD tiene solución afirmativa, entonces existe un vector de posiciones que permite ubicar los trabajos en una sola máquina. De manera recíproca, si toda solución óptima del SRDM requiere de al menos dos máquinas, significa que el SRD tiene solución negativa.

Ahora, supongamos que existe un algoritmo polinomial y con un factor de aproximación estrictamente menor que 2 para el SRDM. Este algoritmo permitiría decidir, en tiempo polinomial, si una instancia de SRD tiene respuesta afirmativa o no. En efecto, una instancia del SRD tiene solución afirmativa si y solamente si, el algoritmo retorna una solución para el problema SRDM cuyo valor es estrictamente menor que dos, pues en este caso la solución óptima del SRDM requiere de una sola máquina. □

Por otra parte, en el caso especial cuando $\gamma_j = 0$ para cada trabajo $j \in J$, es decir, si los trabajos tienen intervalos fijos para su procesamiento, entonces el SRDM puede

reducirse a un problema de coloración de intervalos y resolverse en tiempo polinomial [10, 11]. En [6] se extiende este resultado como se indica a continuación:

Teorema 2.2.2. *El problema SRDM con $\gamma_j \leq 1$ para cada $j \in J$ puede ser resuelto en tiempo polinomial.*

Demostración. Para esta demostración, consideremos $\mathcal{I} = (J, \{1\}, (L), M, (m))$ una instancia del problema SRDM, con $J = \{1, \cdots, n\}$. Supongamos que solo $\hat{n}$ trabajos pueden ser procesados en dos posiciones, es decir, podemos dividir el conjunto de trabajos en dos subconjuntos J_1 y J_2 de tal forma que $J = J_1 \sqcup J_2$, con $|J_1| = \hat{n}$ y $|J_2| = n - \hat{n}$, y además se satisface que $\gamma_j = 1$, para todo $j \in J_1$ y $\gamma_j = 0$, para todo $j \in J_2$.

Se conoce que si el problema de decisión SRDM(k), con $k \leq m$ admite un algoritmo polinomial, entonces es posible resolver el problema de optimización SRDM en tiempo polinomial. En efecto, basta resolver varias instancias de SRDM(k), para distintos valores de k, empezando con $k = m$ y aplicando búsqueda binaria. Adicionalmente, notar que si el número de núcleos de los trabajos que contienen a algún $t \in [r_{\text{mín}}, d_{\text{máx}})$ es mayor que k, entonces el problema SRDM(k) no tiene solución.

Sabemos que cada trabajo $j \in J_1$ puede ser ubicado en la posición 0 o 1, de donde el intervalo correspondiente contiene el tiempo r_j ($\ell = 0$) o $d_j - 1$ ($\ell = 1$). Este hecho, permite formular un problema de flujos para resolver el SRDM(k). Para ello, definimos una red capacitada de la siguiente manera: s y u representan el nodo fuente y el nodo sumidero. Para cada trabajo $j \in J_1$ se crea un nodo s_j. Luego, por cada $t \in \{r_{\text{mín}}, r_{\text{mín}} + 1, \cdots, d_{\text{máx}} - 1\}$ realizamos la diferencia entre el número de ventanas de tiempo y el número de núcleos que contienen a t. Si esta diferencia es positiva se crea un nodo u_t. Entre los nodos s y s_j se crea un arco (s, s_j) de capacidad uno, y de los nodos s_j hacia los nodos u_{r_j} y u_{d_j-1} (en caso de que existan) de igual manera. Entre los nodos u_t y u se coloca un arco de capacidad $e(t)$, donde $e(t)$ se define como la diferencia entre k y el número de núcleos que contienen a t. Notar que si $e(t) = 0$, para algún arco (u_t, u), entonces ninguno de los posibles caminos de s a u que pasan por u_t serán seleccionados para transportar flujo. Por lo tanto, en este caso se pueden eliminar el nodo u_t y sus arcos incidentes de la red capacitada.

Notar que en todo flujo entero factible, únicamente uno de los dos arcos (s_j, u_{r_j})

y (s_j, u_{d_j-1}) tiene flujo positivo, lo que determina una única posición para el trabajo j. La capacidad entre los nodos u_t y u nos asegura que esta selección de intervalos de procesamiento no exceda las k máquinas disponibles. Por tanto, un flujo de s a u retorna un vector de posiciones $\mathcal{L}$ para los trabajos de J_1 asociados a arcos (s, s_j) con flujo igual a 1. Por construcción, esta selección es tal que $h(\mathfrak{I}^{\mathcal{L}}) \leq k$. Notar además que toda asignación de los trabajos a k máquinas induce un flujo de valor $\hat{n}$ en esta red. Luego, si el flujo máximo en la red es menor a $\hat{n}$, entonces el problema SRDM(k) tiene solución negativa. □

Para comprender un poco mejor la demostración del teorema anterior se plantea el siguiente ejemplo.

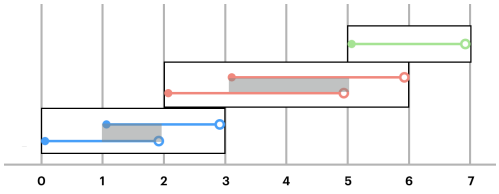
Ejemplo 2.2.1. *Supongamos que se tienen tres trabajos como se describen en la siguiente tabla:*

Trabajo (j)	r_j	d_j	p_j
1	0	3	2
2	2	6	3
3	5	7	2

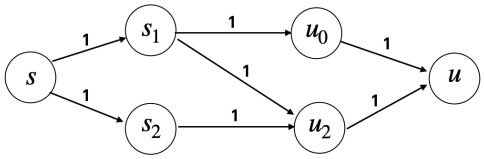
Tabla 2.1: Trabajos con $\gamma_j \leq 1$

Se quiere determinar si es posible ejecutar todos los trabajos en una sola máquina.

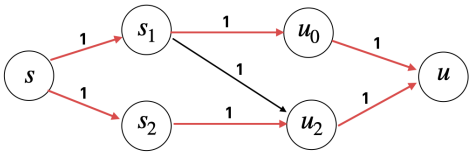
En la Figura 2.1a se muestra gráficamente como se representarían los trabajos en una línea temporal. En un rectángulo blanco se muestra la ventana de tiempo, dentro del rectángulo se encuentran segmentos de recta que representan los posibles intervalos de procesamiento de un determinado trabajo. La intersección de todos los posibles intervalos de procesamiento de un determinado trabajo es su núcleo, y el mismo se muestra en un rectángulo gris. Para el trabajo 3*, dado que solo admite un único posicionamiento (pues* $\gamma_3 = 0$*), su único posible intervalo de procesamiento coincide con su núcleo. En la Figura 2.1b se muestra la red capacitada que se describe en el Teorema 2.2.2.*



(a) Gráfico de los trabajos de la Tabla 2.1 en una linea de tiempo.



(b) Red capacitada en el Teorema 2.2.2.



(c) Flujo máximo sobre la red capacitada.

Figura 2.1: Gráficos de los trabajos descritos en la Tabla 2.1.

Para resolver el problema SRDM(1)*, se debe encontrar un flujo maximal sobre esta red, el cual se puede observar en color rojo en la Figura 2.1c.*

Como podemos observar, el flujo tiene un valor de dos, por tanto se puede concluir que es posible ejecutar todos los trabajos en una sola máquina. De hecho, la solución del flujo maximal nos dice que se ejecute cada uno de los dos primeros trabajos en su primera posición posible ($\ell_1 = \ell_2 = 0$)*. Notar que el trabajo* 3 *admite una sola posición, pues debe ser procesado obligatoriamente en el intervalo* $[5, 7)$*. Por tanto, dicho trabajo no se incluye en la construcción de la red capacitada.*

En [6] se demuestra además que SRDM es NP-difícil cuando el tiempo máximo de espera de algún trabajo supera las dos unidades. Resumimos este resultado a continuación y referimos a dicho artículo para su demostración.

Teorema 2.2.3. *[6] El problema SRDM con* $\max\{\gamma_j : j \in J\} \geq 2$ *es NP-difícil.*

Spieksma [25] propone un algoritmo glotón para el problema JISP (Job Interval Selection Problem) que consiste en seleccionar intervalos de procesamiento para un conjunto de trabajos, de tal forma que se maximice la cantidad de trabajos que pueden ser procesados en una sola máquina (ver Algoritmo 1). Este algoritmo se puede modificar para resolver el problema SRDM(k), con $k \in \mathbb{N}$, y se ha probado en [6] que dicho algoritmo tiene un orden de aproximación de $O(\log(n))$, donde n es el número de trabajos que se quiere procesar.

Algoritmo 1: Algoritmo glotón de Spieksma.

$J := \{1, 2, \cdots, n\}$ conjunto de trabajos ;

$S_j := \left\{r_{j,0}, \cdots, r_{j,\gamma_j}\right\}$, para todo $j \in J$;

$S := \bigcup_{j \in J} S_j$;

$s_{\text{máx}} := \max_{r \in S} r$;

$J' = \emptyset$;

$\text{ind} := \emptyset$;

$t := -\infty$;

mientras $s_{\text{máx}} \geq t$ **hacer**

  $(j, \ell) := \arg\min \{d_{j,\ell} \mid r_{j,\ell} \in S,\ j \in J \setminus J',\ r_{j,\ell} \geq t\}$;

  $\text{ind} := \text{ind} \cup \{(j, \ell)\}$;

  $J' = J' \cup \{j\}$;

  $S := \bigcup_{j' \in J \setminus J'} S_{j'}$;

  $t := d_{j,\ell}$;

  $s_{\text{máx}} := \max_{r \in S} r$;

fin

Uno de los resultados más relevantes que podemos mencionar sobre el Algoritmo de Spieksma es el hecho de que el problema JISP puede resolverse con un factor de aproximabilidad de $\frac{1}{2}$ al ejecutar el Algoritmo 1 en una sola máquina. Este resultado se resume a continuación y su demostración se encuentra detallada en [25].

Teorema 2.2.4. *[25] El problema JISP puede resolverse con un factor de aproximación de $\frac{1}{2}$ usando el Algoritmo 1.*

El siguiente teorema nos muestra el factor de aproximabilidad del Algoritmo de Spieksma al usarse repetidamente para resolver el problema SRDM.

Teorema 2.2.5. *El problema SRDM puede resolverse con un factor de aproximación $O(\log(n))$ al aplicarse repetidamente el Algoritmo 1.*

Demostración. Sean $\mathcal{I} = (J, P, L_P, M, M_P)$ una instancia del problema SRDM y $k^* = h(J)$ el número de máquinas requeridas por una solución óptima de $\mathcal{I}$. Para asignar los trabajos de J a las máquinas, vamos a aplicar reiteradamente el Algoritmo 1.

Al inicio de una iteración cualquiera, denotaremos por J' al conjunto de trabajos que ya han sido asignados a alguna máquina, mientras que el conjunto $J \setminus J'$ contiene los trabajos por asignar. Denotaremos por $\hat{n}$ a la cardinalidad de $J \setminus J'$. Notar que al menos $\frac{\hat{n}}{k^*}$ trabajos pueden ser procesados en una misma máquina, pues al menos $\hat{n}/h(J \setminus J')$ pueden ser procesados en una misma máquina, y $h(J \setminus J') < h(J) = k^*$. Del Teorema 2.2.4 se sigue entonces que al aplicar el Algoritmo 1 al menos $\frac{\hat{n}}{2k^*}$ trabajos serán asignados a la siguiente máquina.

En la primera iteración $\frac{n}{2k^*}$ trabajos son asignados a la primera máquina. Luego, el número de trabajos sin asignar al inicio de la segunda iteración es:

$$n - \frac{n}{2k^*} = n\left(1 - \frac{1}{2k^*}\right).$$

En la segunda aplicación del Algoritmo 1 se asignarán a la próxima máquina al menos la siguiente cantidad de trabajos:

$$\frac{n - \frac{n}{2k^*}}{2k^*} = n\left(1 - \frac{1}{2k^*}\right)\left(\frac{1}{2k^*}\right).$$

El número de trabajos sin asignar después de la segunda iteración es:

$$n\left(1 - \frac{1}{2k^*}\right) - n\left(1 - \frac{1}{2k^*}\right)\left(\frac{1}{2k^*}\right) = n\left(1 - \frac{1}{2k^*}\right)^2.$$

De esta manera, después de la iteración i-ésima tendremos que

$$n\left(1 - \frac{1}{2k^*}\right)^i$$

trabajos se encuentran sin asignar. Luego, después de $\lceil 2k^* \log(n) \rceil$ iteraciones, el número de trabajos sin asignar será igual a:

$$n\left(1 - \frac{1}{2k^*}\right)^{\lceil 2k^* \log(n) \rceil} \leq ne^{-\log(n)} = 1.$$

De aquí obtenemos que requieren a lo más $\lceil 2k^* \log(n) \rceil + 1$ máquinas para ejecutar todos los trabajos. Es decir, el algoritmo tiene un factor de aproximabilidad $O(\log(n))$. □

En [6] se propone una versión modificada del algoritmo glotón planteado por Spieksma y se analiza su factor de aproximabilidad. Los autores designan a este nuevo algoritmo como algoritmo glotón de mejor ajuste (Greedy Best Fit, GBF) (Reproducimos la descripción de su pseudocódigo en el Algoritmo 2).

Algoritmo 2: Algoritmo GBF.

$J := \{1, 2, \cdots, n\}$ conjunto de trabajos ;

$I_j := \emptyset$, para cada $j \in J$;

$I_{\text{Sol}} := (I_1, I_2, \cdots, I_n)$;

Ordenamos los trabajos de J en forma ascendente al tamaño de su ventana de tiempo ;

para $j \in J$ **hacer**

  para $\ell \in [d_j - r_j - p_j]$ **hacer**

    $h_{\text{máx}}(\ell) = \underset{t \in I_{j,\ell}}{\text{máx}} h(I_{\text{Sol}}, t)$;

  fin

  $h_{\text{mín}} = \underset{\ell \in [d_j - r_j - p_j]}{\text{mín}} \{h_{\text{máx}}(\ell)\}$;

  $\ell_j = \text{mín}\{\ell \colon \ell \in [d_j - r_j - p_j] \wedge h_{\text{mín}} = h_{\text{máx}}(\ell)\}$;

  Actualizar $I_j = [r_j + \ell_j, r_j + p_j + \ell_j)$ en I_{Sol};

fin

Como se puede observar, el Algoritmo 1 primero asigna todos los trabajos que sean posibles en una determinada máquina y repite este proceso las veces que sean necesarias hasta que todos los trabajos sean asignados a alguna máquina. Por otro lado, el Algoritmo 2 procesa un determinado trabajo en cuya posición se encuentra el menor número máximo de solapamientos de los trabajos que ya han sido procesados. De esta forma se ejecutan los trabajos mientras que al mismo tiempo se intenta asignarlos en una posición que no incremente el número de máquinas usadas, a menos que sea estrictamente necesario.

Teorema 2.2.6. *El Algoritmo 2 tiene un factor de aproximabilidad de* $\Omega(\log(n))$.

Demostración. Vamos a construir una familia de instancias $\mathcal{I}_{\hat{h}} = \left(J_{\hat{h}}, \{1\}, (L), M, (\hat{h})\right)$, con $M = \{1, 2, \cdots, \hat{h}\}$ y $\hat{h} \in \mathbb{N}^*$, para el SRDM tales que la solución óptima de cada instancia $\mathcal{I}_{\hat{h}}$ tenga el valor $\text{Opt}(\mathcal{I}_{\hat{h}}) = 1$, para todo $\hat{h} \in \mathbb{N}^*$, mientras que la solución encontrada por el Algoritmo 2 tenga el valor de $\hat{h}$.

Definimos estas instancias recursivamente. Cuando $\hat{h} = 1$, tenemos que $J_1 = \{1\}$, con $r_1 = 0$, $d_1 = 2$ y $p_1 = 1$.

Para $\hat{h} > 1$, definimos

$$J_{\hat{h}} = J_{\hat{h}-1} \cup \bar{J}_{\hat{h}-1} \cup \left\{2^{\hat{h}} - 1\right\},$$

donde $\bar{J}_{\hat{h}-1} = \left\{\left|J_{\hat{h}-1}\right| + 1, \left|J_{\hat{h}-1}\right| + 2, \cdots, 2\left|J_{\hat{h}-1}\right|\right\}$. Notar que $J_{\hat{h}} = \{1, 2, \cdots, 2^{\hat{h}} - 1\}$. Los tiempos de llegada, plazos de ejecución y duraciones de los nuevos trabajos se establecen como se indica a continuación. Denotamos por $\alpha_{\hat{h}-1}$ al mayor plazo de ejecución de los trabajos en $J_{\hat{h}-1}$, es decir, $\alpha_{\hat{h}-1} = \max\limits_{j \in J_{\hat{h}-1}} \{d_j\}$. De manera similar, $\rho_{\hat{h}-1} = \max\{d_j - r_j : j \in J_{\hat{h}-1}\}$ es el mayor ancho de ventana de tiempo de un trabajo en $J_{\hat{h}-1}$. En función de estos dos parámetros, definimos $\beta_{\hat{h}-1} = \left\lfloor \frac{1}{2}\alpha_{\hat{h}-1}(\rho_{\hat{h}-1} - 1) + \frac{3}{2} \right\rfloor$.

Notar que cada trabajo $j' \in \bar{J}_{\hat{h}-1}$ satisface que $j' = \left|J_{\hat{h}-1}\right| + j$ para un cierto $j \in J_{\hat{h}-1}$. Definimos los parámetros de j' en función de los parámetros de j:

$$r_{j'} = \alpha_{\hat{h}-1} r_j + \beta_{\hat{h}-1}$$
$$d_{j'} = \alpha_{\hat{h}-1} d_j + \beta_{\hat{h}-1}$$
$$p_{j'} = \alpha_{\hat{h}-1} p_j$$

Finalmente, para el trabajo $2^{\hat{h}} - 1$ definimos $r_{2^{\hat{h}}-1} = 0$, $d_{2^{\hat{h}}-1} = \alpha_{\hat{h}-1} + 2\beta_{\hat{h}-1} - 1$ y $p_{2^{\hat{h}}-1} = \beta_{\hat{h}-1}$.

Puede demostrarse por inducción sobre $\hat{h}$ que $\mathrm{Opt}(\mathcal{I}_{\hat{h}}) = 1$ se cumple para cada $\hat{h} \in \mathbb{N}^*$. En efecto, cuando $\hat{h} = 1$ es evidente que se requiere de una sola máquina para procesar el único trabajo en J_1. Vamos a seleccionar para esta instancia la solución óptima donde el inicio del procesamiento del trabajo 1 se da en el instante $\tau_1 = 1$. Notar que la finalización del procesamiento de este trabajo se da en el tiempo $\tau_1 + 1 = 2 = d_1 = \alpha_1$.

Supongamos ahora que para cierto $\hat{h} \in \mathbb{N}^*$, todos los trabajos de $J_{\hat{h}-1}$ pueden ser procesados por una sola máquina, y demostremos que lo mismo ocurre para todos los trabajos de $J_{\hat{h}}$. En efecto, para cada $j' \in \bar{J}_{\hat{h}-1}$ fijamos su tiempo de inicio de procesamiento en $\tau_{j'} = \alpha_{\hat{h}-1}\tau_j + \beta_{\hat{h}-1}$. Como todos los intervalos de $\bar{J}_{\hat{h}-1}$ se obtienen al escalar los intervalos respectivos de $J_{\hat{h}-1}$ por el factor $\alpha_{\hat{h}-1}$ y desplazarlos en $\beta_{\hat{h}-1}$, esta solución requiere por hipótesis de inducción de una sola máquina para procesar los trabajos de $\bar{J}_{\hat{h}-1}$. Adicionalmente, por hipótesis de inducción, tenemos que el último trabajo de $J_{\hat{h}-1}$ termina de ser procesado en el tiempo $\alpha_{\hat{h}-1} = \max\{d_j : j \in J_{\hat{h}-1}\}$. Por

otra parte, el primer trabajo de $\bar{J}_{\hat{h}-1}$ empieza a ser procesado en el tiempo

$$\tau_{|J_{\hat{h}-1}|+1} = \alpha_{\hat{h}-1}\tau_1 + \beta_{\hat{h}-1} = \alpha_{\hat{h}-1} + \beta_{\hat{h}-1}.$$

Como $p_{2^{\hat{h}}-1} = \beta_{\hat{h}-1}$, se sigue que es posible procesar en una misma máquina todos los trabajos de $J_{\hat{h}-1}$, luego el trabajo $2^{\hat{h}} - 1$ (empezando en $\tau_{2^{\hat{h}}-1} = \alpha_{\hat{h}-1}$ y terminando en $\tau_{2^{\hat{h}}-1} + p_{2^{\hat{h}}-1} = \alpha_{\hat{h}-1} + \beta_{\hat{h}-1}$), y finalmente todos los trabajos de $\bar{J}_{\hat{h}-1}$. De aquí se concluye que $\mathrm{Opt}(\mathcal{I}_{\hat{h}}) = 1$, para todo $\hat{h} \in \mathbb{N}^*$.

Por otra parte, puede verse que el Algoritmo 2 emplea al menos $\hat{h}$ máquinas para procesar los trabajos de $\mathcal{I}_{\hat{h}}$. En efecto, como el algoritmo ordena los trabajos por el ancho de sus ventanas de tiempo, asignará primero todos los trabajos de $J_{\hat{h}-1}$, luego todos los trabajos de $\bar{J}_{\hat{h}-1}$ y finalmente el trabajo $2^{\hat{h}} - 1$. Además, cada trabajo j es asignado a la primera posición $\ell \in [\gamma_j]$ que satisface $h_{\text{máx}}(\ell) = h_{\text{mín}}$ (ver Algoritmo 2). De aquí se sigue que los $\hat{h}-1$ trabajos de $J_{\hat{h}-1}$ que tienen tiempo de llegada igual a cero empezarán a ser procesados en ese instante. De igual forma, los $\hat{h}-1$ trabajos de $\bar{J}_{\hat{h}-1}$ que tienen tiempo de llegada $\beta_{\hat{h}-1}$ iniciarán su procesamiento en ese momento. Por lo tanto, el trabajo $2^{\hat{h}} - 1$ deberá ser procesado en una máquina adicional, y empezará su procesamiento en $\tau_{2^{\hat{h}}-1} = r_{2^{\hat{h}}-1} = 0$.

Si denotamos por $\mathrm{Alg2}(\mathcal{I}_{\hat{h}})$ al valor de la solución encontrada por el Algoritmo 2 para la instancia $\mathcal{I}_{\hat{h}}$, hemos demostrado que el factor de aproximabilidad de este algoritmo está acotado por debajo por

$$\frac{\mathrm{Alg2}(\mathcal{I}_{\hat{h}})}{\mathrm{Opt}(\mathcal{I}_{\hat{h}})} = \frac{\hat{h}}{1} = \hat{h}.$$

Finalmente, notar que el número de trabajos de la instancia $\mathcal{I}_{\hat{h}}$ es

$$n = 2^{\hat{h}} - 1 \qquad \Longleftrightarrow \qquad \hat{h} = \log_2(n+1),$$

de donde se sigue la cota de $\Omega(\log(n))$ para el factor de aproximabilidad. □

Para comprender un poco mejor la demostración del teorema anterior se plantea el siguiente ejemplo.

Ejemplo 2.2.2. *Dada la instancia* $\mathcal{I}_1 = (J_1, \{1\}, (L), M, (1))$, *donde* $J_1 = \{1\}$, $r_1 = 0$, $d_1 = 2$, $p_1 = 1$. *Es claro que* $Alg2(\mathcal{I}_1) = 1 = Opt(\mathcal{I}_1)$.

Para $\hat{h} = 2$*, tenemos la instancia* $\mathcal{I}_2 = (J_2, \{1\}, (L), M, (2))$ *donde* $J_2 = \{1, 2, 3\}$*,* $\rho_1 = \text{máx}\{d_1 - r_1\} = \text{máx}\{2 - 0\} = 2$ *y* $\alpha_1 = \text{máx}\{d_1\} = \text{máx}\{2\} = 2$*. De donde se tiene que*

$$\beta_1 = \left\lfloor \frac{1}{2}\alpha_1(\rho_1 - 1) + \frac{3}{2} \right\rfloor = \left\lfloor \frac{1}{2}(2)(2-1) + \frac{3}{2} \right\rfloor = \left\lfloor \frac{5}{2} \right\rfloor = 2.$$

Calculando los parámetros del nuevo trabajo $2^{\hat{h}} - 1 = 2^2 - 1 = 3$ *tenemos que* $r_3 = 0$*,* $d_3 = 2\beta_1 + \alpha_1 - 1 = 2(2) + 2 - 1 = 5$ *y* $p_3 = \beta_1 = 2$*. Los parámetros del nuevo trabajo en* $\bar{J}_1 = \{2\}$ *son:* $r_2 = \alpha_1 r_1 + \beta_1 = 2(0) + 2 = 2$*,* $d_2 = \alpha_1 d_1 + \beta_1 = 2(2) + 2 = 6$ *y* $p_2 = \alpha_1 p_1 = 2(1) = 2$*.*

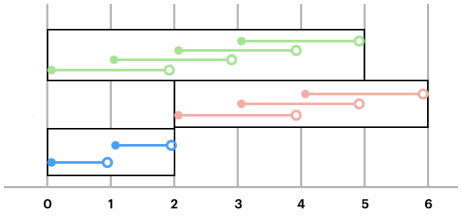
El Algoritmo 2 nos dice que ordenemos los trabajos de J_2 *de acuerdo al tamaño del intervalo. Sin embargo, por la forma en la que se crearon los trabajos de* J_2*, podemos observar que ya se encuentran ordenados. Continuando con el Algoritmo 2, para el trabajo* 1 *se selecciona el intervalo* $[0, 1)$*. Luego, para el trabajo* 2*, como no tiene conflicto con el intervalo* $[0, 1)$*, se selecciona el intervalo* $[2, 4)$*. Finalmente, se atiende al final el trabajo* 3*, y dado que los posibles intervalos de selección*

$$[0, 2), [1, 3), [2, 4), [3, 5)$$

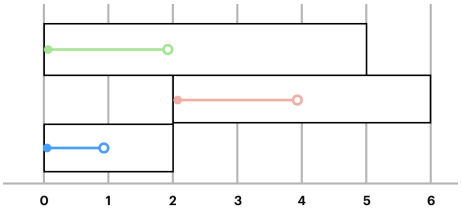
siempre están en conflicto con los dos primeros intervalos seleccionados, se selecciona el intervalo $[0, 2)$*. Con lo cuál, se tiene que el número de solapamientos en el instante* 0 *es* $2 = \hat{h}$*.*

Adicionalmente, se tiene que todos los trabajos pueden ser ejecutados en una sola máquina (Véase la Figura 2.2). En la Figura 2.2a se muestra una visualización gráfica de los trabajos de la instancia $\mathcal{I}_2$*. En un rectángulo blanco se muestra la ventana de tiempo asociada a cada trabajo. Dentro de cada ventana de tiempo se muestran las posibles posiciones de los intervalos de ejecución de cada trabajo.*

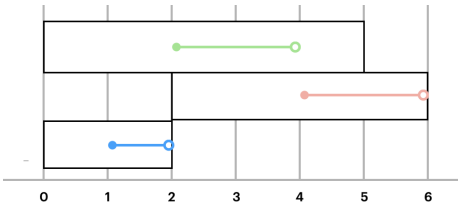
La solución obtenida por el Algoritmo 2 aplicado a la instancia $\mathcal{I}_2$*, se muestra en la Figura 2.2b. Como se puede notar en el instante cero se encuentra el mayor número de solapamientos. Por tanto, se necesitan dos máquinas (*$Alg2(\mathcal{I}_2) = 2$*), mientras que como se observa en la Figura 2.2c, todos los trabajos pueden ser procesados en una sola máquina (*$Opt(\mathcal{I}_2) = 1$*).*



(a) Gráfico de los trabajos de la instancia $\mathcal{I}_2$



(b) Gráfico de la solución encontrada por el Algoritmo 2 aplicado a la instancia $\mathcal{I}_2$



(c) Gráfico de una solución óptima de la instancia $\mathcal{I}_2$

Figura 2.2: Gráficos de la instancia $\mathcal{I}_2$.

Ahora, para $\hat{h} = 3$, *tenemos la instancia* $\mathcal{I}_3 = (J_3, \{1\}, (L), M, (3))$ *y se tiene los parámetros* $\rho_2 = \text{máx}\{d_1 - r_1, d_2 - r_2, d_3 - r_3\} = \text{máx}\{2 - 0, 5 - 0, 6 - 2\} = 5$, $\alpha_2 = \text{máx}\{d_1, d_2, d_3\} = \text{máx}\{2, 4, 6\} = 6$, *y*

$$\beta_2 = \left\lfloor \frac{1}{2}\alpha_2(\rho_2 - 1) + \frac{3}{2} \right\rfloor = \left\lfloor \frac{1}{2}(6)(5-1) + \frac{3}{2} \right\rfloor = \left\lfloor \frac{27}{2} \right\rfloor = 13.$$

Calculando los parámetros del trabajo $2^{\hat{h}} - 1 = 7$ *tenemos que* $r_7 = 0$, $d_7 = 2\beta_2 + \alpha_2 - 1 = 2(13) + 6 - 1 = 31$ *y* $p_7 = \beta_2 = 13$.

Los parámetros de los nuevos trabajos de $\bar{J}_3 = \{4, 5, 6\}$ *son:*

$$r_4 = \alpha_2 r_1 + \beta_2 = 6(0) + 13 = 13,$$
$$d_4 = \alpha_2 d_1 + \beta_2 = 6(2) + 13 = 25,$$
$$p_4 = \alpha_2 p_1 = 6(1) = 6,$$
$$r_5 = \alpha_2 r_2 + \beta_2 = 6(2) + 13 = 25,$$
$$d_5 = \alpha_2 d_2 + \beta_2 = 6(6) + 13 = 49,$$
$$p_5 = \alpha_2 p_2 = 6(2) = 12,$$
$$r_6 = \alpha_2 r_3 + \beta_2 = 6(0) + 13 = 13,$$
$$d_6 = \alpha_2 d_3 + \beta_2 = 6(5) + 13 = 43,$$

$$p_6 = \alpha_2 p_3 = 6(2) = 12.$$

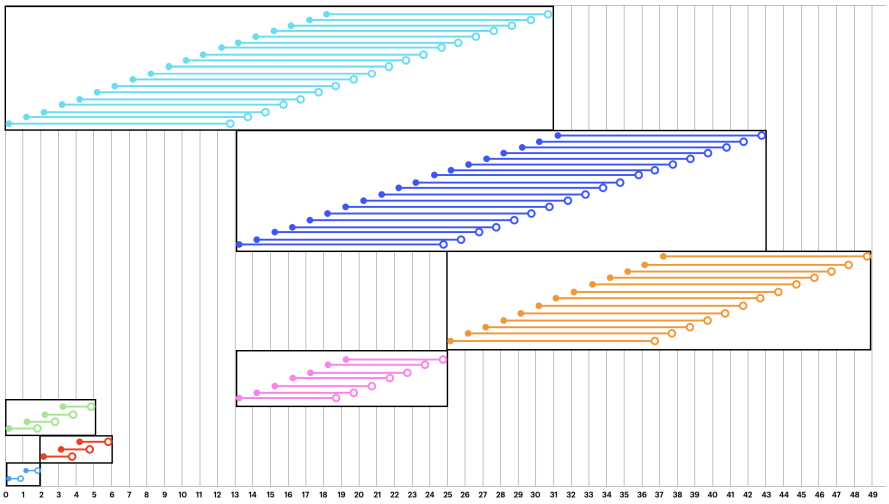
El Algoritmo 2 nos dice que ordenemos los trabajos de la instancia $\mathcal{I}_3$ de acuerdo al tamaño de la ventana de tiempo de los trabajos. Los trabajos ordenados se muestran en la Tabla 2.2. En la Figura 2.3 se muestra una visualización gráfica de los trabajos

Trabajo	r_j	d_j	p_j
1	0	2	1
2	2	6	2
3	0	5	2
4	13	25	6
5	25	49	12
6	13	43	12
7	0	31	13

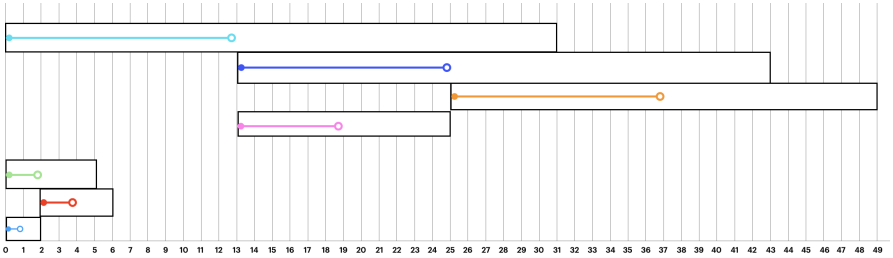
Tabla 2.2: Trabajos ordenados de la instancia $\mathcal{I}_3$.

de la instancia $\mathcal{I}_3$. En un rectángulo blanco se muestra la ventana de tiempo asociada a cada trabajo. Dentro de cada ventana de tiempo se muestran las posibles posiciones de los intervalos de ejecución de cada trabajo.

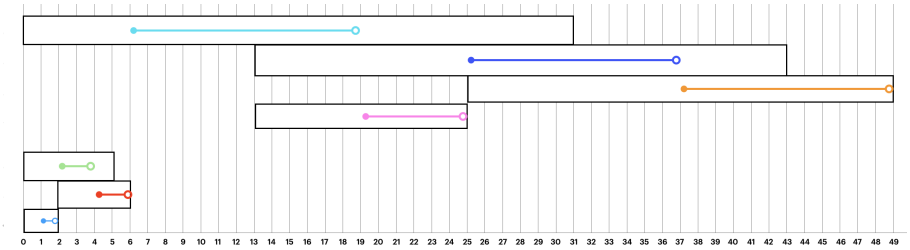
La solución obtenida por el Algoritmo 2 aplicado a la instancia $\mathcal{I}_3$, se muestra en la Figura 2.3b. Como se puede notar en el instante cero se encuentra el mayor número de solapamientos. Donde se puede notar que se requieren tres máquinas ($Alg2(\mathcal{I}_3) = 3$) mientras que como se observa en la Figura 2.3c, todos los trabajos pueden ser procesados en una sola máquina ($Opt(\mathcal{I}_3) = 1$).



(a) Gráfico de los trabajos de la instancia $\mathcal{I}_3$



(b) Gráfico de la solución encontrada por el Algoritmo 2 aplicado a la instancia $\mathcal{I}_3$



(c) Gráfico de una solución óptima de la instancia $\mathcal{I}_3$

Figura 2.3: Gráficos de la instancia $\mathcal{I}_3$.

Finalmente, si bien el resultado anterior nos muestra que el Algoritmo 2 no posee un factor de aproximación constante, notar que la familia de instancias empleada en la demostración tiene la particularidad de que los tiempos de procesamiento de los trabajos en las nuevas instancias crecen de manera no acotada. Si se imponen límites máximos y mínimos para la duración de los trabajos, es posible demostrar que dicho algoritmo es aproximable con un factor asintótico de $9\frac{p_{\text{máx}}}{p_{\text{mín}}}$ para el problema SRDM, donde $p_{\text{mín}} = \min_{j \in J}\{p_j\}$ y $p_{\text{máx}} = \max_{j \in J}\{p_j\}$. Este resultado fue presentado en [6]. A continuación proponemos una demostración alternativa del mismo, para el caso especial

cuando las ventanas de tiempo de los trabajos son suficientemente grandes.

Teorema 2.2.7. *[6] Si $\delta_j \geq 2p_j - 1$ para todo $j \in J$, entonces el Algoritmo 2 tiene un factor asintótico de aproximabilidad de $9\frac{p_{\text{máx}}}{p_{\text{mín}}}$.*

Demostración. Sean $\mathcal{I} = (J, \{1\}, (L), M, (m))$ una instancia del problema SRDM y $\text{Sol}_{\text{GBF}}(\mathcal{I}) = (\mathcal{J}, \mathcal{Z})$ la solución encontrada por el Algoritmo 2 para $\mathcal{I}$. Sabemos que $\mathcal{J} = I_{\text{Sol}}$ y $z = \text{Alg2}(\mathcal{I}) = h(I_{\text{Sol}})$, donde $\mathcal{Z} = \{z\}$.

Sea $j \in J$ el primer trabajo tal que al procesar j el valor de la solución del Algoritmo 2 se incrementa por primera vez a z. Adicionalmente, sea $\mathcal{J}' = (\hat{I}_1, \hat{I}_2, \cdots, \hat{I}_{\hat{n}})$ el estado del vector I_{Sol} antes del procesamiento de j. Notar que

- $h(\mathcal{J}') = z - 1$.
- $h\left((\hat{I}_1, \cdots, \hat{I}_{\hat{n}}, I_{j,\ell})\right) = z$, para todo $\ell \in [\gamma_j]$.

Definamos como $\delta_j = d_j - r_j$ el tamaño de la ventana de tiempo asociada al trabajo j y tomemos como $\mathcal{J}''$ al vector de intervalos de $\mathcal{J}'$ que se intersecan con el intervalo $[r_j, d_j)$. Definamos la siguiente función real:

$$\begin{aligned} h_{\text{GBF}} \colon \mathbb{R}_+ \cup \{0\} &\longrightarrow \mathbb{R}_+ \cup \{0\}. \\ t &\longmapsto h(\mathcal{J}'', t) \end{aligned}$$

Se denota como $\omega_j(\mathcal{J}'')$ al área bajo la curva $h_{\text{GBF}}(t)$ entre los límites r_j y d_j, es decir,

$$\omega_j(\mathcal{J}'') = \int_{r_j}^{d_j} h(\mathcal{J}'', t)dt.$$

Notar que esta área puede pensarse como si estuviera formada por áreas rectangulares de longitudes $p_{j'}$ que corresponden al tiempo de procesamiento de cada uno de los trabajos en J'', donde J'' son los trabajos asociados a los intervalos de $\mathcal{J}''$. Sin embargo, no necesariamente se cumple que $\omega_j(\mathcal{J}'') = \sum_{j' \in J''} p_{j'}$, ya que pueden haber trabajos asociados a intervalos de $\mathcal{J}''$ que empezaron a procesarse antes de r_j o que terminan de procesarse después de d_j. Estos trabajos pueden generar rectángulos con una longitud menor a su tiempo de procesamiento dentro del intervalo $[r_j, d_j)$.

Sabemos que el Algoritmo 2 atiende a los trabajos en orden ascendente del tamaño de su ventana de tiempo, es decir, todos los trabajos asociados a los intervalos de $\mathcal{J}''$ tienen ventanas de tiempo con longitudes menor o igual a δ_j unidades. De aquí

concluimos que, en cualquier solución factible, todos los trabajos de J'' deben ser procesados dentro del intervalo de tiempo

$$[r_j - \delta_j, d_j + \delta_j).$$

En particular, esto se cumple también para cualquier solución óptima. Definamos las funciones:

$$\begin{array}{rcl} h_{\text{Opt}} \colon \mathbb{R}_+ \cup \{0\} & \longrightarrow & \mathbb{R}_+ \cup \{0\} \\ t & \longmapsto & h(\mathcal{J}_{\text{Opt}}, t) \end{array} \qquad \text{y} \qquad \begin{array}{rcl} h''_{\text{Opt}} \colon \mathbb{R}_+ \cup \{0\} & \longrightarrow & \mathbb{R}_+ \cup \{0\} \\ t & \longmapsto & h(\mathcal{J}''_{\text{Opt}}, t) \end{array},$$

donde $\mathcal{J}_{\text{Opt}}$ es el vector de intervalos seleccionados para cada uno de los trabajos en la solución óptima, mientras que $\mathcal{J}''_{\text{Opt}}$ contiene únicamente los intervalos seleccionados para cada uno de los trabajos de J''. Por construcción, tenemos que

$$h_{\text{Opt}}(t) \geq h''_{\text{Opt}}(t), \qquad \forall t \geq 0.$$

Además, el número de máquinas requeridas por el óptimo es:

$$z^* = \max_{t \in [0,L]} h_{\text{Opt}}(t).$$

Por lo tanto, se cumple que $h''_{\text{Opt}}(t) \leq z^*$ para todo $t \geq 0$. Sea $\omega_j^*(\mathcal{J}'')$ el área bajo la curva $h_{\text{Opt}}(t)''$ entre los límites $r_j - \delta_j$ y $d_j + \delta_j$. Notemos que cada trabajo $j' \in J''$ aporta con un rectángulo de longitud igual a $p_{j'}$ a dicha área, por lo que su valor es $\sum_{j' \in J''} p_{j'}$. Luego,

$$\omega_j^*(\mathcal{J}'') = \sum_{j' \in J''} p_{j'} \geq \omega_j(\mathcal{J}''). \tag{2.2.1}$$

Por otro lado, como $h''_{\text{Opt}}(t) \leq z^*$, para todo $t \geq 0$, se tiene que el área $\omega_j^*(\mathcal{J}'')$ está completamente contenida en un rectángulo de base igual a la longitud del intervalo $[r_j - \delta_j, d_j + \delta_j]$ y de altura igual a z^*, es decir,

$$\omega_j^* \leq z^*(d_j + \delta_j - (r_j - \delta_j)) = 3z^*\delta_j.$$

De esta última desigualdad con (2.2.1), tenemos que

$$z^* \geq \frac{\omega_j(\mathcal{J}'')}{3\delta_j}.$$

Nuestro objetivo es acotar inferiormente el valor del área $\omega_j(\mathcal{J}'')$ y acotar superiormente el valor de δ_j.

Sea $\tilde{M} \subseteq M$ el conjunto de máquinas usadas por el Algoritmo 2 para procesar los trabajos de J''. Sabemos que ninguna de las $z-1$ máquinas de $\tilde{M}$ puede estar inactiva por más de $p_j - 1$ unidades consecutivas dentro del intervalo $[r_j, d_j)$, porque eso implicaría que existe un posicionamiento $\ell \in [\gamma_j]$ tal que

$$h\left((\hat{I}_1, \cdots, \hat{I}_{\hat{n}}, I_{j,\ell})\right) = z - 1.$$

Si $\delta_j \geq 2p_j - 1$, para cada $t \in \{r_j, r_j + 1, d_j - (2p_j - 1)\}$, las máquinas en $\tilde{M}$ se encuentran inactivas a lo más en $p_j - 1$ unidades de tiempo consecutivas dentro del intervalo $[t, t + 2p_j - 1)$ y se encuentran procesando un trabajo por al menos $p_{\text{mín}}$ unidades de tiempo (consecutivas o no) en ese intervalo. En efecto, si en el instante t la máquina está inactiva, deberá empezar a procesar un trabajo máximo en el instante $t + p_j - 1$ y este procesamiento terminará en $t + p_j + p_{\text{mín}} - 1$ o en un instante posterior. Como $t + p_j + p_{\text{mín}} - 1 \leq t + 2p_j - 1$, la máquina habrá estado activa durante al menos $p_{\text{mín}}$ unidades de tiempo dentro del intervalo. Por otra parte, supongamos que en el instante t la máquina está ocupada con el procesamiento de alguno de los trabajos de J'' que empezó antes de t, y supongamos que este procesamiento continúa durante $\bar{p}$ unidades de tiempo. Si $\bar{p} \geq p_{\text{mín}}$, se sigue que la máquina habrá estado ocupada al menos durante $p_{\text{mín}}$ unidades dentro del intervalo. Caso contrario, si $\bar{p} = p_{\text{mín}} - a$, con $a > 0$, la máquina entrará en inactividad en el instante $t + p_{\text{mín}} - a$ y deberá iniciar el procesamiento de un nuevo trabajo a más tardar en el instante $t + p_{\text{mín}} + p_j - 1 - a$. El procesamiento de este nuevo trabajo durará al menos $p_{\text{mín}}$ unidades de tiempo, de las cuales al menos a unidades estarán dentro del intervalo $[t, t + 2p_j - 1)$, pues $t + p_{\text{mín}} + p_j - 1 - a + a = t + p_{\text{mín}} + p_j - 1 \leq t + 2p_j - 1$. Luego, la máquina estará procesando trabajos al menos durante $p_{\text{mín}} - a + a = p_{\text{mín}}$ unidades de tiempos dentro del intervalo. De aquí se concluye que el área bajo la curva $h_{\text{GBF}}(t)$ entre los límites t y $t + 2p_j - 1$, con $t \in \{r_j, r_j + 1, d_j - (2p_j - 1)\}$, es de por lo menos $(z-1)p_{\text{mín}}$.

Empleando los mismos argumentos el resultado anterior se puede generalizar de la siguiente forma: Si $\delta_j \geq p_j + \hat{p}$, con $\hat{p} \in [p_j - 1]$, para cada $t \in \{r_j, r_j + 1, d_j - (p_j + \hat{p})\}$, el área bajo la curva $h_{\text{GBF}}(t)$ entre los límites t y $t + p_j + \hat{p}$ es de al menos:

$$\begin{cases} p_{\text{mín}}(z-1), & \text{si } \hat{p} + 1 \geq p_{\text{mín}}, \\ (\hat{p}+1)(z-1), & \text{si } \hat{p} + 1 < p_{\text{mín}}. \end{cases}$$

Definamos como N a la cantidad de intervalos de longitud $2p_j - 1$ contenidos dentro del intervalo $[r_j, d_j]$, es decir,

$$N = \left\lfloor \frac{\delta_j}{2p_j - 1} \right\rfloor.$$

Observar que de la condición $\delta_j \geq 2p_j - 1$, se sigue que $N \geq 1$. Luego,

$$\omega_j(\mathcal{J}'') \geq N p_{\text{mín}}(z-1).$$

Notemos que se puede escribir $\delta_j = N(2p_j - 1) + \tilde{p}$, para algún $\tilde{p} \in [2p_j - 2]$.

Si $\tilde{p} \leq p_j$, entonces:

$$N = \frac{\delta_j - \tilde{p}}{2p_j - 1} \geq \frac{\delta_j - p_j}{2p_j - 1} > \frac{\delta_j - p_j}{2p_j}$$

De donde se sigue que

$$\delta_j < 2Np_j + p_j \leq (2N+1)p_{\text{máx}}.$$

Por tanto, como $N \geq 1$, se obtiene que

$$z^* \geq \frac{\omega_j(\mathcal{J}'')}{3\delta_j} > \frac{N(z-1)p_{\text{mín}}}{3(2N+1)p_{\text{máx}}} = \frac{z-1}{3(2+1/N)} \cdot \frac{p_{\text{mín}}}{p_{\text{máx}}} \geq \frac{z-1}{9} \cdot \frac{p_{\text{mín}}}{p_{\text{máx}}}.$$

Es decir,

$$9\frac{p_{\text{máx}}}{p_{\text{mín}}} z^* + 1 > z.$$

Por otra parte, supongamos ahora que $\tilde{p} > p_j$, es decir, $\tilde{p} = p_j + \hat{p}$, con $\hat{p} \in \{1, 2, \cdots, p_j - 2\}$. Observemos que el área de $\omega_j(\mathcal{J}'')$ se puede descomponerse en el área bajo la curva $h_{\text{GBF}}(t)$ para los N intervalos de longitud $2p_j - 1$ más el área bajo la curva para un intervalo de longitud $p_j + \hat{p}$. Vamos a acotar cada una de estas áreas empleando las observaciones señaladas arriba. Consideraremos dos casos.

Primero, supongamos que $\hat{p} + 1 \geq p_{\text{mín}}$:

$$\begin{aligned}\omega_j(\mathcal{J}'') &\geq N(z-1)p_{\text{mín}} + (z-1)p_{\text{mín}} \\ &= (N+1)(z-1)p_{\text{mín}}.\end{aligned}$$

Además, sabemos que $\delta_j = N(2p_j - 1) + p_j + \hat{p}$. De donde se sigue que:

$$\frac{\omega_j(\mathcal{J}'')}{3\delta_j} \geq \frac{(N+1)(z-1)p_{\text{mín}}}{3[N(2p_j-1) + p_j + \hat{p}]} \tag{2.2.2}$$

Véase que:

$$\begin{aligned}\frac{(N+1)p_{\text{mín}}}{N(2p_j-1) + p_j + \hat{p}} > \frac{p_{\text{mín}}}{3p_{\text{máx}}} &\iff 3(N+1)p_{\text{mín}}p_{\text{máx}} > Np_{\text{mín}}(2p_j - 1) + p_{\text{mín}}p_j + p_{\text{mín}}\hat{p} \\ &\iff Np_{\text{mín}}[3p_{\text{máx}} - 2p_j + 1] + (3p_{\text{máx}} - p_j - \hat{p})p_{\text{mín}} > 0\end{aligned}$$

Sabemos que $p_j \leq p_{\text{máx}}$, de donde se sigue que:

$$3p_{\text{máx}} - 2p_j + 1 \geq p_{\text{máx}} + 1 > 0.$$

Además, sabemos que $\hat{p} \leq p_j - 2 < p_{\text{máx}}$, obteniendo que

$$3p_{\text{máx}} - p_j - \hat{p} \geq p_{\text{máx}} > 0.$$

Por lo tanto, como $p_{\text{mín}} > 0$, se obtiene que:

$$Np_{\text{mín}}[3p_{\text{máx}} - 2p_j + 1] + (3p_{\text{máx}} - p_j - \hat{p})p_{\text{mín}} > 0.$$

Equivalentemente,

$$\frac{(N+1)p_{\text{mín}}}{N(2p_j - 1) + p_j + \hat{p}} > \frac{p_{\text{mín}}}{3p_{\text{máx}}}.$$

De esta última desigualdad y (2.2.2) se obtiene que:

$$\frac{\omega_j(\mathcal{J}'')}{3\delta_j} > \frac{(z-1)p_{\text{mín}}}{9p_{\text{máx}}}.$$

Eb el segundo caso, si $\hat{p} + 1 < p_{\text{mín}}$, se tiene que:

$$\begin{aligned}\omega_j(\mathcal{J}'') &\geq N(z-1)p_{\text{mín}} + (z-1)(\hat{p}+1) \\ &= (z-1)(Np_{\text{mín}} + \hat{p} + 1).\end{aligned}$$

Además, sabemos que $\delta_j = N(2p_j - 1) + p_j + \hat{p}$. De donde se sigue que:

$$\frac{\omega_j(\mathcal{J}'')}{3\delta_j} \geq \frac{(Np_{\text{mín}} + \hat{p} + 1)}{3[N(2p_j - 1) + p_j + \hat{p}]} \tag{2.2.3}$$

Véase que:

$$\begin{aligned}\frac{Np_{\text{mín}} + \hat{p} + 1}{N(2p_j - 1) + p_j + \hat{p}} > \frac{p_{\text{mín}}}{3p_{\text{máx}}} \iff & 3Np_{\text{mín}}p_{\text{máx}} + 3(\hat{p}+1)p_{\text{máx}} - Np_{\text{mín}}(2p_j - 1) \\ & - p_{\text{mín}}p_j - p_{\text{mín}}\hat{p} > 0 \\ \iff & Np_{\text{mín}}[3p_{\text{máx}} - 2p_j + 1] + 3(\hat{p}+1)p_{\text{máx}} \\ & - (p_j + \hat{p})p_{\text{mín}} > 0\end{aligned}$$

Como $N \geq 1$, $3p_{\text{máx}} - 2p_j + 1$, $\hat{p} \geq 1$ y $0 < p_{\text{mín}} \leq p_j \leq p_{\text{máx}}$, se sigue que

$$\begin{aligned}Np_{\text{mín}}[3p_{\text{máx}} - 2p_j + 1] + 3(\hat{p}+1)p_{\text{máx}} - (p_j + \hat{p})p_{\text{mín}} &\geq p_{\text{mín}}(3p_{\text{máx}} - 2p_j + 1 + 3\hat{p} + 3 \\ &\quad - p_j - \hat{p}) \\ &= p_{\text{mín}}(3p_{\text{máx}} - 3p_j + 2\hat{p} + 4) \\ &\geq 6p_{\text{mín}} \\ &> 0\end{aligned}$$

Por lo tanto,

$$\frac{Np_{\text{mín}} + \hat{p} + 1}{N(2p_j - 1) + p_j + \hat{p}} > \frac{p_{\text{mín}}}{3p_{\text{máx}}}.$$

De esta última desigualdad y (2.2.3) se obtiene que:

$$\frac{\omega_j(\mathcal{J}'')}{3\delta_j} > \frac{(z-1)p_{\text{mín}}}{9p_{\text{máx}}}.$$

Por lo tanto, hemos probado que

$$z = \text{costo}(\mathcal{Z}) = \text{Alg2}(\mathcal{I}) < 9\frac{p_{\text{máx}}}{p_{\text{mín}}}\text{Opt}(\mathcal{I}) + 1.$$

□

Para comprender un poco mejor el resultado del Teorema anterior se plantea el siguiente ejemplo.

Ejemplo 2.2.3. *Consideremos la instancia $\mathcal{I}_2$ descrita en la familia de instancias del Teorema 2.2.6, es decir,*

$$\mathcal{I}_2 = (J_2, \{1\}, L_P, M, M_P),$$

donde $J_2 = \{1, 2, 3\}$, $L_P = (L)$, $M = 2$ y $M_P = (2)$. En la Tabla 2.3 se muestran los detalles de los trabajos de J_2. Si resolvemos la instancia $\mathcal{I}_2$ usando el Algoritmo 2

Trabajo	r_j	d_j	p_j
1	0	2	1
2	2	6	2
3	0	5	2

Tabla 2.3: Trabajos ordenados de la instancia $\mathcal{I}_2$.

sabemos que se usan dos máquinas. Por otro lado, se puede procesar todos los trabajos en una sola máquina como se vio en la Figura 2.2c. De donde tenemos que $\text{Sol}_{GBF}(\mathcal{I}_2) = (\mathcal{J}, \mathcal{Z})$*, con $\mathcal{J} = ([0, 1), [2, 4), [0, 2))$ y $\mathcal{Z} = \{z = 2\}$. El primer trabajo que hace que la solución del Algoritmo 2 pase de 1 a 2 es el trabajo 3, es decir,*

$$\mathcal{J}' = ([0, 1), [2, 4)).$$

Adicionalmente, sabemos que $h(\mathcal{J}') = 1$, $\delta_3 = 5$ y $\mathcal{J}'' = \mathcal{J}'$. Por tanto, se tiene que

$$\omega_3(\mathcal{J}'') = \int_0^5 h(\mathcal{J}'', t)dt = 3.$$

De donde se puede verificar que:

$$z^* = 1 \geq \frac{1}{5} = \frac{\omega_3(\mathcal{J}'')}{3\delta_3}.$$

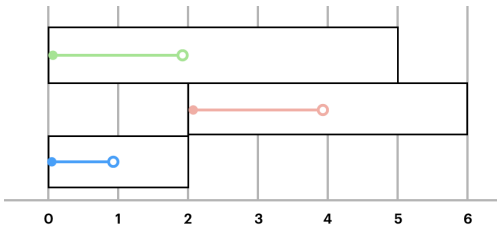
En la figura 2.4d se muestra en gris el área bajo la curva $h_{GBF}(t)$ *entre los límites 0 y 5. Adicionalmente, podemos observar que* $N = 1$*, es decir, el número de intervalos de longitud* $2p_3 - 1 = 3$ *que se pueden formar dentro del intervalo* $[0, 5]$ *es igual a uno. Y por tanto, se puede reescribir* $\delta_3 = N(2p_3 - 1) + p_3 = 5$*. Véase que en cualquier intervalo de longitud* 3 *dentro del intervalo* $[0, 5]$*, el área bajo la curva* $h_{GBF}(t)$ *es de al menos* $N(z - 1)p_{\text{mín}} = 1$*.*

Finalmente, podemos verificar que

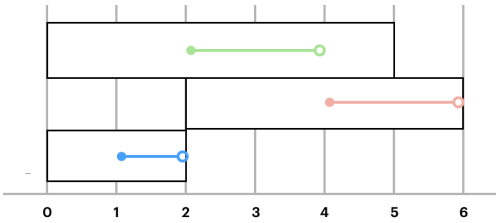
$$\frac{\omega_3(\mathcal{J}'')}{3\delta_3} = \frac{1}{5} > \frac{1}{18} = \frac{z-1}{9} \cdot \frac{p_{\text{mín}}}{p_{\text{máx}}},$$

donde $p_{\text{mín}} = 1$ *y* $p_{\text{máx}} = 2$*. Es decir, se verifica que*

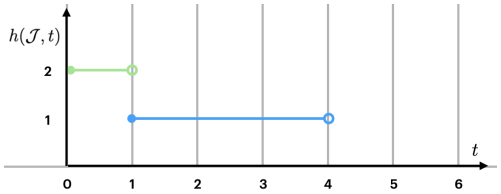
$$9\frac{p_{\text{máx}}}{p_{\text{mín}}} Opt(\mathcal{I}_2) + 1 = 19 > 2 = Alg2(\mathcal{I}_2).$$



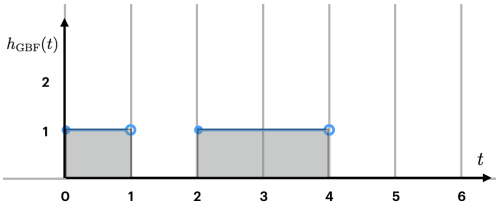
(a) Gráfico de la solución encontrada por el Algoritmo 2 aplicado a la instancia $\mathcal{I}_2$



(b) Gráfico de una solución óptima de la instancia $\mathcal{I}_2$



(c) Gráfico del número de solapamientos generados por el vector $\mathcal{J}$.



(d) Gráfico del número de solapamientos generados por el vector $\mathcal{J}''$.

Figura 2.4: Gráficos de solapamientos para el ejemplo de la instancia $\mathcal{I}_2$.

2.3. Otra versión multi-periodo del problema SRDM

En [20] se considera una versión multi-periodo del problema SRDM, que es ligeramente distinta al problema SRDM-T presentado en la Sección 1.4. Sea $J = \{1, 2, \cdots, n\}$

un conjunto de trabajos a ejecutar. Para cada trabajo $j \in J$ se especifican un tiempo de llegada r_j, un plazo máximo de ejecución d_j y un tiempo constante de procesamiento $p_j = p$. Adicionalmente, supongamos que el horizonte de tiempo se encuentra dividido en T periodos. Cada periodo $q \in P$ corresponde a un intervalo $[t_q, t_q + L_q)$ que inicia en el tiempo t_q y tiene una longitud de L_q. Además, para cada periodo se conoce el número disponible de máquinas m_q. El problema consiste en ejecutar todos los trabajos, de tal forma que en cada periodo se respete el número de máquinas disponibles, y que se minimice el número de máquinas utilizadas simultáneamente. Llamaremos a este problema *calendarización de trabajos con igual tiempo de procesamiento en máquinas paralelas con tiempos de disponibilidad, minimizando el número de máquinas usadas simultáneamente* (SRDM-D).

El problema SRDM-D se diferencia del problema SRDM-T estudiado en el presente trabajo en dos características fundamentales:

- En el SRDM-D se minimiza la (máxima) cantidad de máquinas utilizadas simultáneamente, es decir, en la notación de la Sección 1.4.2, la función objetivo a minimizar es $\max\limits_{q \in P} z_q$, mientras que la función objetivo del SRDM-T es $\sum\limits_{q \in P} z_q$.
- En el SRDM-D se asume que todos los trabajos tienen el mismo tiempo de procesamiento, es decir, que $p_j = p$ para todo $j \in J$.

En [20] los autores proponen un algoritmo polinomial para el SRDM-D basado en la programación lineal. Revisaremos su algoritmo en el resto de esta sección y en el Capítulo 3 estudiaremos la posibilidad de adaptarlo a un caso específico del SRDM-T donde todos los trabajos tienen igual duración.

Utilizando ideas de un artículo previo sobre la calendarización de trabajos de igual duración en máquinas idénticas y paralelas, con el objetivo de minimizar el tiempo de completación [2], los autores demuestran que es posible asumir ciertas condiciones sobre los tiempos óptimos de inicio de procesamiento de los trabajos. Resumimos este resultado en el siguiente lema.

Lema 2.3.1. *[20] Existe una solución óptima para el SRDM-D tal que el tiempo de inicio de cada uno de los trabajos pertenece al conjunto:*

$$\{r_j + \eta p \colon j \in J,\ \eta \in [n]\} \cup \{t_q + \eta p \colon q \in P,\ \eta \in [n]\}.$$

Es decir, el siguiente conjunto de intervalos

$$\{[r_j + \eta p, r_j + (\eta+1)p) : j \in J, \eta \in \mathbb{Z}_+\} \cup \{[t_q + \eta p, t_q + (\eta+1)p) : q \in P, \eta \in \mathbb{Z}_+\}$$

contiene una solución óptima del problema SRDM-D.

En [20] se demuestra además que existe una calendarización óptima de los trabajos de tal forma que todos se ejecutan en el horizonte de tiempo $[r_{\text{mín}}, r_{\text{máx}} + 2np)$. Por lo tanto, aplicando el Lema 2.3.1, se puede concluir que existe una solución óptima tal que los intervalos de procesamiento de los trabajos pertenecen al conjunto $A = A_1 \cup A_2$, donde

$$A_1 = \left\{ [r_j + \eta p, r_j + (\eta+1)p) : \begin{array}{l} j \in J, \eta \in \mathbb{Z}, r_j + \eta p \geq r_{\text{mín}}, \\ r_j + (\eta+1)p \leq \text{mín}\{r_{\text{máx}} + 2np, t_T + L_T\} \end{array} \right\}, \quad \text{y}$$

$$A_2 = \left\{ [t_q + \eta p, t_q + (\eta+1)p) : \begin{array}{l} q \in P, \eta \in \mathbb{Z}, t_q + \eta p \geq r_{\text{mín}}, \\ t_q + (\eta+1)p \leq \text{mín}\{r_{\text{máx}} + 2np, t_T + L_T\} \end{array} \right\}.$$

De aquí en adelante, supondremos que A está formado por los intervalos $I_1, \cdots, I_{|A|}$, los cuales se encuentran ordenados de forma ascendente por su extremo izquierdo. El conjunto A tiene una propiedad de regularidad que será de utilidad para la formulación del SRDM-D que estudiaremos, y que resumimos a continuación.

Lema 2.3.2. *Existe $\phi \in \mathbb{N}$ tal que:*

$$I_{i+1} \cap I_{i+2} \cap \cdots \cap I_{i+\phi} \neq \emptyset,$$

para todo $i \in [|A| - \phi]$, y

$$I_{i+1} \cap I_{i+2} \cap \cdots \cap I_{i+\phi+1} = \emptyset,$$

para todo $i \in [|A| - \phi - 1]$.

Para formular el SRDM-D como un programa lineal entero, se emplean las siguientes variables de decisión:

$$x_{j,i} := \begin{cases} p, & \text{si el trabajo } j \text{ es ejecutado en alguna máquina en el intervalo } I_i \in A, \\ 0, & \text{caso contrario.} \end{cases}, \text{ y}$$

z : cantidad de máquinas requeridas en la solución.

En [20] se propone la siguiente formulación de programación lineal entera para el SRDM-D (PE-SRDM-D):

$$\text{mín } z \tag{2.3.1}$$

s.a.r.

$$\sum_{i\in\{1,2,\cdots,|A|\}} x_{j,i} = p, \quad \forall j \in J, \tag{2.3.2}$$

$$\sum_{\eta\in\{1,2,\cdots,\kappa\}} \sum_{j\in J} x_{j,i+\eta} \leq \text{mín}\{z, m_q\}p, \quad \forall i \in [|A| - \kappa], \kappa \in \{1, 2, \cdots, \phi\}, q \in P, \bigcap_{\eta\in\{1,2,\cdots,\kappa\}} I_{i+\eta} \cap [t_q, t_q + L_q) \neq \emptyset, \tag{2.3.3}$$

$$x_{j,i} = 0, \quad \text{si } R(I_i) < r_j, \forall i \in \{1, 2, \cdots, |A|\}, j \in J, \tag{2.3.4}$$

$$x_{j,i} = 0, \quad \text{si } d_j < D(I_i), \forall i \in \{1, 2, \cdots, |A|\}, j \in J, \tag{2.3.5}$$

$$x_{j,i} \in \{0, p\}, \quad \forall i \in \{1, 2, \cdots, |A|\}, j \in J,$$

$$z \in \mathbb{Z}_+ \cup \{0\},$$

donde $R(I_i)$ es el extremo izquierdo del intervalo I_i, $D(I_i)$ es el extremo derecho del intervalo I_i, y ϕ es la cantidad máxima de intervalos consecutivos de A que tienen intersección no vacía, definida en el Lema 2.3.2.

La función objetivo (2.3.1) mide la cantidad total de máquinas requeridas para procesar todos los trabajos. Las restricciones (2.3.2), (2.3.4) y (2.3.5) establecen que cada trabajo $j \in J$ debe ser asignado a un intervalo de A, el cual debe estar enteramente contenido en $[r_j, d_j)$. Las restricciones (2.3.3) requieren que el número de trabajos procesados en cualquier instante de tiempo no exceda el número de máquinas disponibles m_q en el periodo $q \in P$ correspondiente. Además, estas restricciones fijan el valor de la variable z para que sea por lo menos igual al máximo número de intervalos asignados al procesamiento de trabajos que tienen intersección no nula, es decir, al máximo número de trabajos que se procesarán simultáneamente en algún momento.

El algoritmo propuesto en [20] para resolver el problema SRDM-D, en tiempo polinomial consiste en resolver la relajación lineal del modelo PE-SRDM-D y posteriormente usar la solución óptima fraccionaria para obtener una calendarización de los trabajos. Estos pasos se describen más detalladamente en el Algoritmo 3.

Denotaremos como PL-SRDM-D a la relajación lineal del modelo PE-SRDM-D, la

misma que se puede escribir como:

$$\text{mín } z$$

s.a.r.

$$\sum_{i\in\{1,2,\cdots,|A|\}} x_{j,i} = p, \quad \forall j \in J, \tag{2.3.6}$$

$$\sum_{\eta\in\{1,2,\cdots,\kappa\}} \sum_{j\in J} x_{j,i+\eta} \leq \text{mín}\{z, m_q\}p, \quad \forall i \in [|A| - \kappa], \kappa \in \{1, 2, \cdots, \phi\}, q \in P, \bigcap_{\eta\in\{1,2,\cdots,\kappa\}} I_{i+\eta} \cap [t_q, t_q + L_q) \neq \emptyset, \tag{2.3.7}$$

$$x_{j,i} = 0, \quad \text{si } R(I_i) < r_j, \forall i \in \{1, 2, \cdots, |A|\}, j \in J, \tag{2.3.8}$$

$$x_{j,i} = 0, \quad \text{si } d_j < D(I_i), \forall i \in \{1, 2, \cdots, |A|\}, j \in J, \tag{2.3.9}$$

$$x_{j,i} \geq 0, \quad \forall i \in \{1, 2, \cdots, |A|\}, j \in J,$$

$$z \geq 0.$$

Notar que la condición $x_{j,i} \leq p$ está implicada por las restricciones (2.3.6).

Dada una solución $x \in \mathbb{R}^{n\times|A|}$ del programa lineal PL-SRDM-D y sea I_i un intervalo de A, para algún $i \in \{1, 2, \cdots, |A|\}$, se define la medida

$$v(I_i) = \sum_{\kappa\in\{1,2,\cdots,i\}} \sum_{j\in J} x_{j,\kappa},$$

como la cantidad de tiempo procesada de los trabajos asignados por el modelo PL-SRDM-D en los intervalos $I_1, I_2, \cdots, I_i$.

Por otra parte, veamos que el número de variables de decisión en el programa lineal PL-SRDM-D es igual a $n|A| + 1$. Además, sabemos que

$$\begin{aligned} |A| &\leq (n+T)\frac{1}{p}\left[\text{mín}\{r_{\text{máx}} + 2np, t_T + L_T\} - \text{máx}\{r_{\text{mín}}, t_1\}\right] \\ &\leq (n+T)\left[2n + \frac{r_{\text{máx}} - r_{\text{mín}}}{p}\right] \\ &= (n+T)(2n+c), \end{aligned}$$

con $c = \frac{r_{\text{máx}} - r_{\text{mín}}}{p} \leq r_{\text{máx}} < t_T + L_T$. Por tanto,se sigue que:

$$\begin{aligned} n|A| + 1 &\leq n(n+T)(2n+c) + 1. \\ &= 2n^3 + 2n^2T + cn^2 + cnT + 1. \end{aligned}$$

Es decir, el número de variables de decisión del modelo PL-SRDM-D es de orden $O(n^3 + n^2T)$.

Por otra parte, dado que $1 \leq \phi \leq |A|$ y $T \geq 1$, podemos verificar que el número de restricciones del programa lineal PL-SRDM-D está acotado por:

$$
\begin{aligned}
& n + 2 \sum_{q \in P} \sum_{\kappa \in \{1,2,\cdots,\phi\}} (|A| - \kappa + 1) + 3n|A| + 1 \\
& = n + 2T \left[\phi(|A| + 1) - \frac{1}{2}\phi(\phi + 1) \right] + 3n|A| + 1 \\
& \leq n + 2T|A|^2 + 2T|A| + 3n|A| \\
& \leq n + 2T(n+T)^2(2n+c)^2 + 2T(n+T)(2n+c) + 3n(2n+c)(n+T) \\
& = n + (2c^2n^2T + 4c^2nT^2 + 2c^2T^3 + 8cn^3T + 16cn^2T^2 + 8cnT^3 + 8n^4T + 16n^3T^2 + \\
& \quad + 8n^2T^3) + (2cnT + 2cT^2 + 4n^2T + 4nT^2) + (3cn^2 + 3cnT + 6n^3 + 6n^2T) \\
& = 2c^2n^2T + 4c^2nT^2 + 2c^2T^3 + 8cn^3T + 16cn^2T^2 + 3cn^2 + 8cnT^3 + 5cnT + 2cT^2 + \\
& \quad + 8n^4T + 16n^3T^2 + 6n^3 + 8n^2T^3 + 10n^2T + 4nT^2 + n
\end{aligned}
$$

De donde, el número de restricciones es de orden $O(n^4T + n^3T^2 + n^2T^3) = O(n^2T(n + T)^2)$.

Por lo tanto, debido a la polinomialidad de la Programación Lineal, un algoritmo basado en la solución del modelo PL-SRDM-D es un algoritmo polinomial para el problema SRDM-D. A continuación se presenta un algoritmo con estas características.

Algoritmo 3: Algoritmo basado en programación lineal para el problema SRDM-D.

Entrada: Instancia $\mathcal{I} = (J, P, L_P, M, M_P)$.

1. Ordenar los trabajos de J en forma ascendente de acuerdo al valor del plazo de ejecución d_j, con $j \in J$.
2. Resolver el modelo PL-SRDM-D. Sea $(x^*, z^*) \in \mathbb{R}^{n \times |A|} \times \mathbb{R}$ una solución óptima del mismo.
3. Definir $v(I_i) = \sum_{\kappa \in \{1,2,\cdots,i\}} \sum_{j \in J} x^*_{j,\kappa}, \quad , \forall i \in \{1, 2, \cdots, |A|\}$.
4. Para cada $i \in \{1, 2, \cdots, |A|\}$ definir $\mu_i = \text{mín}\{m_q \colon q \in P, I_i \cap [t_q, t_q + L_q) \neq \emptyset\}$ y fijar $\mu_0 = 0$.
5. **para** $i \in \{1, 2, \cdots, |A|\}$ **hacer**
   - **para** $i' \in \{1, 2, \cdots, \mu_i\}$ **hacer**
     - Definir $\varphi = \mu_0 + \cdots + \mu_{i-1} + i'$.
     - Definir $v_\varphi = v(I_i)$.
   - **fin**

   fin
6. Definir $\theta(0) = 0$.
7. **para** $\lambda \in \{1, 2, \cdots, |J|\}$ **hacer**
   - Definir $\theta(\lambda) = \text{mín}\{\varphi \colon \theta(\lambda - 1) < \varphi \leq \bar{\mu}, v_\varphi > (\lambda - 1) \cdot p\}$, donde $\bar{\mu} = \mu_1 + \cdots + \mu_{|A|}$.

   fin
8. **para** $\lambda \in \{1, 2, \cdots, |J|\}$ **hacer**
   - Definir $I^*_\lambda = I_\eta$, donde $\eta \in \{1, 2, \cdots, |A|\}$ está dado por $v(I_\eta) = v_{\theta(\lambda)}$.

   fin
9. **para** $\lambda \in \{1, 2, \cdots, |J|\}$ **hacer**
   - De los trabajos compatibles con el intervalo I^*_λ, que aún no han sido procesados, asignar a I^*_λ el trabajo con el menor plazo de ejecución.

   fin

Para comprender como funciona el Algoritmo 3 se presenta el siguiente ejemplo:

Ejemplo 2.3.1. *Consideremos la instancia* $\mathcal{I}_4 = (J, P, L_P, M, M_P)$*, donde* $J = \{1, 2, 3, 4\}$*,* $P = \{1, 2, 3\}$*,* $L_P = (4, 2, 4)$*,* $M = \{1, 2\}$ *y* $M_P = (2, 1, 2)$*. Los parámetros de los trabajos de* J *se muestran en la Tabla 2.4 y supongamos que* $t_1 = 0$*,* $t_2 = 4$ *y* $t_3 = 6$*.*

Trabajo	r_j	d_j	p_j
1	0	9	4
2	1	9	4
3	1	9	4
4	0	10	4

Tabla 2.4: Trabajos de la instancia $\mathcal{I}_4$.

Sabemos que $n = 4$, $p = 4$, $r_{\text{mín}} = 0$ *y* $r_{\text{máx}} = 1$. *Como* $r_{\text{máx}} + 2np = 1 + 2(4)(4) = 33 \geq 10 = 6 + 4 = t_3 + L_3$, *se tiene que* $A = A_1 \cup A_2 = \{I_1, \cdots, I_{|A|}\}$, *con*

$$A_1 = \left\{ [r_j + \eta p, r_j + (\eta+1)p) : \begin{array}{l} j \in J, \eta \in \mathbb{Z}, r_j + \eta p \geq 0, \\ r_j + (\eta+1)p \leq 10 \end{array} \right\}, \text{ y}$$

$$A_2 = \left\{ [t_q + \eta p, t_q + (\eta+1)p) : \begin{array}{l} q \in P, \eta \in \mathbb{Z}, t_q + \eta p \geq 0, \\ t_q + (\eta+1)p \leq 10 \end{array} \right\}.$$

Es decir,

$$A_1 = \{[0,4), [4,8), [1,5), [5,9)\}, \text{ y}$$
$$A_2 = \{[0,4), [4,8), [2,6), [6,10)\}.$$

De donde se sigue que

$$A = \{[0,4), [4,8), [1,5), [5,9), [2,6), [6,10)\}.$$

Ordenando los intervalos en forma ascendente al extremo izquierdo se tiene que $I_1 = [0,4)$, $I_2 = [1,5)$, $I_3 = [2,6)$, $I_4 = [4,8)$, $I_5 = [5,9)$ *y* $I_6 = [6,10)$.

Calculando el valor de ϕ, *tenemos que a lo más tres intervalos consecutivos se pueden intersecar sin ser vacío, es decir,* $\phi = \text{máx}\{1,2,3\} = 3$. *Por tanto, podemos escribir el siguiente modelo lineal:*

$$\text{mín } z$$

s.a.r.

$$\sum_{i \in \{1,2,\cdots,6\}} x_{j,i} = 4, \quad \forall j \in J \tag{2.3.10}$$

$$\sum_{\eta\in\{1,2,\cdots,\kappa\}}\sum_{j\in J} x_{j,i+\eta} \leq 4\,\text{mín}\{z, m_q\}, \quad \forall i \in [6-\kappa], \kappa \in \{1,2,3\}, q \in P, \bigcap_{\eta\in\{1,2,\cdots,\kappa\}} I_{i+\eta} \cap [t_q, t_q + L_q) \neq \emptyset, \tag{2.3.11}$$

$$x_{j,i} = 0, \quad si\ R(I_i) < r_j, \forall i \in \{1,2,\cdots,6\}, j \in J, \tag{2.3.12}$$

$$x_{j,i} = 0, \quad si\ d_j < D(I_i), \forall i \in \{1,2,\cdots,6\}, j \in J, \tag{2.3.13}$$

$$x_{j,i} \geq 0, \quad \forall i \in \{1,2,\cdots,6\}, j \in J,$$

$$z \geq 0.$$

Las restricciones (2.3.10)-(2.3.13) *se muestran detalladamente en el Anexo A.*

Se puede probar que $z^* = 2$, $x^*_{1,1} = 4$, $x^*_{2,2} = 2$, $x^*_{2,5} = 2$, $x^*_{3,2} = 2$, $x^*_{3,5} = 2$, $x^*_{4,6} = 4$ *y* $x^*_{j,i} = 0$, *para todo* $(j,i) \notin \{(1,1),(2,2),(2,5),(3,2),(3,5),(4,6)\}$ *es una solución óptima para este problema lineal.*

Ahora, conforme se indica en el tercer y cuarto paso del Algoritmo 3, definamos:

$$\begin{aligned}
v(I_1) &= x^*_{1,1} = 4, & \mu_1 &= \text{mín}\{m_1\} = m_1 = 2,\\
v(I_2) &= x^*_{1,1} + x^*_{2,2} + x^*_{3,2} = 8, & \mu_2 &= \text{mín}\{m_1, m_2\} = m_2 = 1,\\
v(I_3) &= x^*_{1,1} + x^*_{2,2} + x^*_{3,2} = 8, & \mu_3 &= \text{mín}\{m_1, m_2\} = m_2 = 1,\\
v(I_4) &= x^*_{1,1} + x^*_{2,2} + x^*_{3,2} = 8, & \mu_4 &= \text{mín}\{m_2, m_3\} = m_2 = 1,\\
v(I_5) &= x^*_{1,1} + x^*_{2,2} + x^*_{3,2} + x^*_{2,5} + x^*_{3,5} = 12, & \mu_5 &= \text{mín}\{m_2, m_3\} = m_2 = 1,\\
v(I_6) &= x^*_{1,1} + x^*_{2,2} + x^*_{3,2} + x^*_{2,5} + x^*_{3,5} + x^*_{4,6} = 16, & \mu_6 &= \text{mín}\{m_3\} = m_3 = 2.
\end{aligned}$$

A continuación, definimos $\mu_0 = 0$ *y el quinto paso nos dice que definamos los siguientes valores:*

$$\begin{aligned}
v_1 &= v_{\mu_0+1} = v(I_1) = 4, && (i = 1, i' = 1)\\
v_2 &= v_{\mu_0+2} = v(I_1) = 4, && (i = 1, i' = 2)\\
v_3 &= v_{\mu_0+\mu_1+1} = v(I_2) = 8, && (i = 2, i' = 1)\\
v_4 &= v_{\mu_0+\mu_1+\mu_2+1} = v(I_3) = 8, && (i = 3, i' = 1)\\
v_5 &= v_{\mu_0+\mu_1+\mu_2+\mu_3+1} = v(I_4) = 8, && (i = 4, i' = 1)\\
v_6 &= v_{\mu_0+\mu_1+\mu_2+\mu_3+\mu_4+1} = v(I_5) = 12, && (i = 5, i' = 1)\\
v_7 &= v_{\mu_0+\mu_1+\mu_2+\mu_3+\mu_4+\mu_5+1} = v(I_6) = 16, && (i = 6, i' = 1)\\
v_8 &= v_{\mu_0+\mu_1+\mu_2+\mu_3+\mu_4+\mu_5+2} = v(I_6) = 16. && (i = 6, i' = 2)
\end{aligned}$$

Posteriormente, definimos $\theta(0) = 0$.

En el paso 7 del Algoritmo 3, definimos:

$$\theta(1) = \text{mín}\{i\colon \theta(0) < i \leq 8, v_i > 0\} = \text{mín}\{1, 2, \cdots, 8\} = 1,$$

$$\theta(2) = \text{mín}\{i\colon \theta(1) < i \leq 8, v_i > 4\} = \text{mín}\{3, 4, \cdots, 8\} = 3,$$

$$\theta(3) = \text{mín}\{i\colon \theta(2) < i \leq 8, v_i > 8\} = \text{mín}\{6, 7, 8\} = 6,$$

$$\theta(4) = \text{mín}\{i\colon \theta(3) < i \leq 8, v_i > 12\} = \text{mín}\{7, 8\} = 7.$$

Con estos valores, en el paso 8 definimos los intervalos $I_1^* = I_1$, $I_2^* = I_2$, $I_3^* = I_5$ *y* $I_4^* = I_6$.

Finalmente, el paso 9 nos dice que ejecutemos el trabajo 1 en el intervalo I_1^*, *el trabajo 2 en el intervalo* I_2^*, *el trabajo 3 en el intervalo* I_3^* *y el trabajo 4 en el intervalo* I_4^*.

El Algoritmo 3 emplea la solución de la relajación lineal PL-SRDM-D para definir, para cada intervalo I_i, con $i \in \{1, \cdots, |A|\}$, una cantidad $v(I_i)$ que corresponde al tiempo total invertido en el procesamiento de trabajos (por todas las máquinas) desde el instante de inicio del primer intervalo I_1 hasta el momento de finalización de I_i. Posteriormente, se determina el número de máquinas μ_i que están disponibles en cada intervalo I_i. Decimos que una máquina está disponible durante I_i, si el periodo de disponibilidad de la máquina contiene a todo el intervalo I_i.

En el paso 5 del algoritmo se definen varias *copias* de cada intervalo, según el número de máquinas disponibles, es decir, para el intervalo $I_i \in A$ se definen μ_i copias. Estas copias se denotan por v_φ, con $1 \leq \varphi \leq \bar{\mu}$, donde $\bar{\mu} := \sum_{i \in \{1,2,\cdots,|A|\}} \mu_i$. Notar que las copias están etiquetadas de tal forma que se respeta el orden de los intervalos: si $v_{\theta(\lambda_1)}$ es una copia del intervalo I_{i_1}, $v_{\theta(\lambda_2)}$ es una copia del intervalo I_{i_2}, y $i_1 < i_2$, entonces se cumple que $\theta(\lambda_1) < \theta(\lambda_2)$.

En el paso 7 se selecciona una copia $v_{\theta(\lambda)}$ de intervalo por cada índice $\lambda \in J$, mientras que en el paso 8 se determina cuál es el intervalo I_λ^* al que corresponde $v_{\theta(\lambda)}$. Finalmente, en el paso 9 cada trabajo es asignado a uno de estos intervalos según sus plazos de ejecución.

El algoritmo retorna como solución los n intervalos $(I_1^*, \cdots, I_n^*)$, a los que, en adelante, denominaremos intervalos marcados. Notar que cada intervalo marcado es un intervalo del conjunto A, es decir, para cada I_λ^*, con $\lambda \in J$, existe un $\sigma(\lambda) \in \{1, 2, \cdots, |A|\}$

tal que $I^*_\lambda = I_{\sigma(\lambda)}$. Adicionalmente, para cada $i \in \{1, 2, \cdots, |A|\}$, se puede definir μ^*_i como el número de copias seleccionadas del intervalo $I_i \in A$ por el Algoritmo 3, es decir, $\mu^*_i = |\{I^*_\lambda \colon \lambda \in J, \sigma(\lambda) = i\}|$. De esta manera, un intervalo I_i es un intervalo marcado, si al menos una de sus copias es seleccionada en el paso 7, es decir, si $\mu^*_i > 0$. Observar además que $\sum_{i \in \{1,2,\cdots,|A|\}} \mu^*_i = n$.

Uno de los resultados presentados en [20] es el siguiente lema.

Lema 2.3.3 ([20]). *Para cada $\lambda \in J$ tal que $I^*_\lambda = I_{\sigma(\lambda)}$ para algún $\sigma(\lambda) \in \{1, 2, \cdots, |A|\}$, se satisface la desigualdad:*

$$(\mu^*_1 + \cdots + \mu^*_{\sigma(\lambda)})p \geq v(I_{\sigma(\lambda)}) > (\mu^*_1 + \cdots + \mu^*_{\sigma(\lambda)} - 1)p. \tag{2.3.14}$$

Demostración. Sea $(x^*, z^*) \in \mathbb{R}^{n \times |A|} \times \mathbb{R}$ la solución descrita en el paso 2 del Algoritmo 3. Vamos a proceder por inducción sobre el valor del índice λ. Recordemos que del paso 7 del algoritmo, $\theta(1) \leq \theta(2) \leq \cdots \leq \theta(n)$, por tanto se sigue que $\sigma(1) \leq \sigma(2) \leq \cdots \leq \sigma(n)$. Además, no necesariamente $\sigma(1) = 1$, y si $\sigma(1) > 1$, entonces $\mu^*_1 = \cdots = \mu^*_{\sigma(1)-1} = 0$. De hecho, $\mu^*_i = 0$ para todo $i \in \{1, 2, \cdots, |A|\} \setminus \{\sigma(1), \cdots, \sigma(n)\}$.

Para $\lambda = 1$, del paso 7 en el Algoritmo 3 se sabe que la ν-ésima copia del intervalo $I_{\sigma(1)}$ fue seleccionada debido a que $v(I_{\sigma(1)}) > (\nu - 1) \cdot p$, para cada $\nu \in \{1, 2, \cdots, \mu^*_{\sigma(1)}\}$. De donde se tiene que

$$v(I_{\sigma(1)}) > (\mu^*_{\sigma(1)} - 1)p.$$

Si adicionalmente asumimos que $\mu^*_{\sigma(1)} < \mu_{\sigma(1)}$, y como sabemos que la copia $\mu^*_{\sigma(1)} + 1$ del intervalo $I_{\sigma(1)}$ no se seleccionó, entonces

$$v(I_{\sigma(1)}) \not> \mu^*_{\sigma(1)} \cdot p \quad \Longleftrightarrow \quad v(I_{\sigma(1)}) \leq \mu^*_{\sigma(1)} \cdot p.$$

Por otra parte, supongamos ahora que $\mu^*_{\sigma(1)} = \mu_{\sigma(1)}$. Como x^* es una solución factible del modelo lineal PL-SRDM-D, conocemos que se satisface las restricciones (2.3.7), de donde por definición de $v(I_{\sigma(1)})$ se tiene:

$$v(I_{\sigma(1)}) \leq m_q p,$$

para cada $q \in P$ tal que $I_{\sigma(1)} \cap [t_q, t_q + L_q) \neq \emptyset$. Luego, por definición de $\mu_{\sigma(1)}$, se obtiene que

$$v(I_{\sigma(1)}) \leq \mu_{\sigma(1)} p,$$

y como $\mu_{\sigma(1)} = \mu^*_{\sigma(1)}$, se sigue

$$v(I_{\sigma(1)}) \leq \mu^*_{\sigma(1)} p.$$

Así, en ambos casos se satisface:

$$\mu^*_{\sigma(1)} p \geq v(I_{\sigma(1)}) > (\mu^*_{\sigma(1)} - 1)p.$$

Si $\sigma(1) = 1$, se obtiene de forma trivial la desigualdad (2.3.14). Luego, si $\sigma(1) > 1$ y como $\mu^*_1 = \cdots = \mu^*_{\sigma(1)-1} = 0$ se obtiene la desigualdad (2.3.14):

$$(\mu^*_1 + \cdots + \mu^*_{\sigma(1)})p \geq v(I_{\sigma(1)}) > (\mu^*_1 + \cdots + \mu^*_{\sigma(1)} - 1)p.$$

Supongamos ahora que la desigualdad se satisface para $\lambda \leq \hat{n} < n$. Es decir,

$$(\mu^*_1 + \cdots + \mu^*_{\sigma(\lambda)})p \geq v(I_{\sigma(\lambda)}) > (\mu^*_1 + \cdots + \mu^*_{\sigma(\lambda)} - 1)p, \qquad \forall \lambda \leq \hat{n} < n.$$

Para $\sigma(\hat{n}+1) = \sigma(\hat{n})$, de la desigualdad anterior se obtiene (2.3.14).

Luego, para $\sigma(\hat{n}+1) > \sigma(\hat{n})$, del paso 7 en el Algoritmo 3 se tiene que para todo $\nu \in \{1, 2, \cdots, \mu^*_{\sigma(\hat{n}+1)}\}$, la ν-ésima copia del intervalo $I_{\sigma(\hat{n}+1)}$ fue seleccionada debido a que $v(I_{\sigma(\hat{n}+1)}) > (\mu^*_1 + \cdots + \mu^*_{\sigma(\hat{n})} + \nu - 1) \cdot p$. Es decir, se satisface la desigualdad:

$$v(I_{\sigma(\hat{n}+1)}) > (\mu^*_1 + \cdots + \mu^*_{\sigma(\hat{n})} + \mu^*_{\sigma(\hat{n})+1} + \cdots + \mu^*_{\sigma(\hat{n}+1)} - 1) \cdot p.$$

Ahora, si $\mu^*_{\sigma(\hat{n}+1)} < \mu_{\sigma(\hat{n}+1)}$, por definición de $\mu^*_{\sigma(\hat{n}+1)}$, sabemos que la copia $\mu^*_{\sigma(\hat{n}+1)} + 1$ del intervalo $I_{\sigma(\hat{n}+1)}$ no fue seleccionada y por tanto

$$v(I_{\sigma(\hat{n}+1)}) \not> (\mu^*_1 + \cdots + \mu^*_{\sigma(\hat{n}+1)}) \cdot p \quad \Longleftrightarrow \quad v(I_{\sigma(\hat{n}+1)}) \leq (\mu^*_1 + \cdots + \mu^*_{\sigma(\hat{n}+1)}) \cdot p.$$

Por otra parte, si $\mu^*_{\sigma(\hat{n}+1)} = \mu_{\sigma(\hat{n}+1)}$, del tercer paso del Algoritmo 3 tenemos que

$$v(I_{\sigma(\hat{n}+1)}) = v(I_{\sigma(\hat{n})}) + \sum_{j \in J} x^*_{j,\sigma(\hat{n})+1} + \cdots + \sum_{j \in J} x^*_{j,\sigma(\hat{n}+1)}.$$

Adicionalmente, de las restricciones (2.3.7) sabemos que

$$\sum_{j \in J} x^*_{j,\sigma(\hat{n}+1)-\phi+1} + \cdots + \sum_{j \in J} x^*_{j,\sigma(\hat{n}+1)-1} + \sum_{j \in J} x^*_{j,\sigma(\hat{n}+1)} \leq m_q p,$$

para cada $q \in P$ tal que $I_{\sigma(\hat{n}+1)-\phi+1} \cap \cdots \cap I_{\sigma(\hat{n}+1)} \cap [t_q, t_q + L_q) \neq \emptyset$. De donde, por definición de $\mu_{\sigma(\hat{n}+1)}$ tenemos que

$$\sum_{j \in J} x^*_{j,\sigma(\hat{n}+1)-\phi+1} + \cdots + \sum_{j \in J} x^*_{j,\sigma(\hat{n}+1)-1} + \sum_{j \in J} x^*_{j,\sigma(\hat{n}+1)} \leq \mu_{\sigma(\hat{n}+1)} p.$$

Por un lado, si $\phi \geq \sigma(\hat{n}+1) - \sigma(\hat{n})$ se tiene que

$$\begin{aligned} v(I_{\sigma(\hat{n}+1)}) &= v(I_{\sigma(\hat{n})}) + \sum_{j\in J} x^*_{j,\sigma(\hat{n})+1} + \cdots + \sum_{j\in J} x^*_{j,\sigma(\hat{n}+1)} \\ &\leq v(I_{\sigma(\hat{n})}) + \mu_{\sigma(\hat{n}+1)}p \\ &\leq (\mu^*_1 + \cdots + \mu^*_{\sigma(\hat{n})}) \cdot p + \mu_{\sigma(\hat{n}+1)}p \\ &= (\mu^*_1 + \cdots + \mu^*_{\sigma(\hat{n})} + \mu_{\sigma(\hat{n}+1)}) \cdot p \\ &= (\mu^*_1 + \cdots + \mu^*_{\sigma(\hat{n})} + \mu^*_{\sigma(\hat{n})+1} + \cdots + \mu^*_{\sigma(\hat{n}+1)-1} + \mu_{\sigma(\hat{n}+1)}) \cdot p. \end{aligned}$$

Por otra parte, si $\phi < \sigma(\hat{n}+1) - \sigma(\hat{n})$ se sigue que

$$\begin{aligned} v(I_{\sigma(\hat{n}+1)}) &= v(I_{\sigma(\hat{n})}) + \sum_{j\in J} x^*_{j,\sigma(\hat{n})+1} + \cdots + \sum_{j\in J} x^*_{j,\sigma(\hat{n}+1)-\phi} + \cdots + \sum_{j\in J} x^*_{j,\sigma(\hat{n}+1)} \\ &= v(I_{\sigma(\hat{n}+1)-\phi}) + \sum_{j\in J} x^*_{j,\sigma(\hat{n}+1)-\phi+1} + \cdots + \sum_{j\in J} x^*_{j,\sigma(\hat{n}+1)} \\ &\leq v(I_{\sigma(\hat{n}+1)-\phi}) + \mu_{\sigma(\hat{n}+1)}p. \end{aligned}$$

Como el intervalo $I_{\sigma(\hat{n}+1)-\phi}$ no ha sido seleccionado por el Algoritmo 3 sabemos que

$$v(I_{\sigma(\hat{n}+1)-\phi}) \leq (\mu^*_1 + \cdots + \mu^*_{\sigma(\hat{n})})p. \tag{2.3.15}$$

Dado que

$$v(I_{\sigma(\hat{n})}) > (\mu^*_1 + \cdots + \mu^*_{\sigma(\hat{n})} - 1)p,$$

podemos tomar

$$v(I_{\sigma(\hat{n})}) = (\mu^*_1 + \cdots + \mu^*_{\sigma(\hat{n})} - 1)p + \delta,$$

para algún $\delta \in (0, p]$. Así, tenemos que

$$\begin{aligned} v(I_{\sigma(\hat{n}+1)-\phi}) &= v(I_{\sigma(\hat{n})}) + \sum_{j\in J} x^*_{j,\sigma(\hat{n})+1} + \cdots + \sum_{j\in J} x^*_{j,\sigma(\hat{n}+1)-\phi} \\ &= (\mu^*_1 + \cdots + \mu^*_{\sigma(\hat{n})} - 1)p + \delta + \sum_{j\in J} x^*_{j,\sigma(\hat{n})+1} + \cdots + \sum_{j\in J} x^*_{j,\sigma(\hat{n}+1)-\phi}. \end{aligned}$$

Reemplazando esta última igualdad en (2.3.15) se obtiene que

$$\sum_{j\in J} x^*_{j,\sigma(\hat{n})+1} + \cdots + \sum_{j\in J} x^*_{j,\sigma(\hat{n}+1)-\phi} \leq p - \delta.$$

Sumando $v(I_{\sigma(\hat{n})})$ a ambos lados tenemos

$$v(I_{\sigma(\hat{n})}) + \sum_{j\in J} x^*_{j,\sigma(\hat{n})+1} + \cdots + \sum_{j\in J} x^*_{j,\sigma(\hat{n}+1)-\phi} \leq v(I_{\sigma(\hat{n})}) + p - \delta.$$

Como $v(I_{\sigma(\hat{n})}) = (\mu_1^* + \cdots + \mu_{\sigma(\hat{n})}^* - 1)p + \delta$, se tiene

$$v(I_{\sigma(\hat{n})}) + \sum_{j\in J} x_{j,\sigma(\hat{n})+1}^* + \cdots + \sum_{j\in J} x_{j,\sigma(\hat{n}+1)-\phi}^* \leq (\mu_1^* + \cdots + \mu_{\sigma(\hat{n})}^*)p,$$

o lo que es lo mismo

$$v(I_{\sigma(\hat{n}+1)-\phi}) \leq (\mu_1^* + \cdots + \mu_{\sigma(\hat{n})}^*)p.$$

Y por tanto se tiene

$$\begin{aligned} v(I_{\sigma(\hat{n}+1)}) &\leq v(I_{\sigma(\hat{n}+1)-\phi}) + \mu_{\sigma(\hat{n}+1)}p \\ &\leq (\mu_1^* + \cdots + \mu_{\sigma(\hat{n})}^*) \cdot p + \mu_{\sigma(\hat{n}+1)}p \\ &= (\mu_1^* + \cdots + \mu_{\sigma(\hat{n})}^* + \mu_{\sigma(\hat{n}+1)}) \cdot p \\ &= (\mu_1^* + \cdots + \mu_{\sigma(\hat{n})}^* + \mu_{\sigma(\hat{n})+1}^* + \cdots + \mu_{\sigma(\hat{n}+1)-1}^* + \mu_{\sigma(\hat{n}+1)}) \cdot p. \end{aligned}$$

Así, en ambos casos se concluye que

$$v(I_{\sigma(\hat{n}+1)}) \leq (\mu_1^* + \cdots + \mu_{\sigma(\hat{n}+1)-1}^* + \mu_{\sigma(\hat{n}+1)}) \cdot p,$$

y como $\mu_{\sigma(\hat{n}+1)}^* = \mu_{\sigma(\hat{n}+1)}$, se infiere que

$$v(I_{\sigma(\hat{n}+1)}) \leq (\mu_1^* + \cdots + \mu_{\sigma(\hat{n}+1)-1}^* + \mu_{\sigma(\hat{n}+1)}^*) \cdot p.$$

Por tanto, se concluye que para $\lambda \leq \hat{n} + 1$, se satisface la desigualdad (2.3.14):

$$(\mu_1^* + \cdots + \mu_{\sigma(\lambda)}^*)p \geq v(I_{\sigma(\lambda)}) > (\mu_1^* + \cdots + \mu_{\sigma(\lambda)}^* - 1)p,$$

con lo cual se obtiene el resultado. □

Teorema 2.3.1 ([20]). *El Algoritmo 3 permite construir una solución factible del modelo PE-SRDM-D.*

Demostración. Observar que el Algoritmo 3, en el paso 8, selecciona n intervalos

$$I_\lambda^* = I_{\sigma(\lambda)} \in A,$$

para cada $\lambda \in \{1, 2, \cdots, |J|\}$.

Un mismo intervalo puede seleccionarse varias veces, es decir, se puede tener $\sigma(\lambda_1)$ $= \sigma(\lambda_2)$ para $\lambda_1, \lambda_2 \in \{1, 2, \cdots |J|\}$ distintos entre sí. Posteriormente, en el paso 9 del algoritmo, se consideran estos intervalos en orden creciente del índice λ y se asigna

al intervalo I^*_λ aquel trabajo j^* que sea compatible con el intervalo (es decir, que satisfaga $I^*_\lambda \subseteq [r_{j^*}, d_{j^*})$), que tenga el menor plazo de ejecución d_{j^*}, y que no haya sido previamente asignado a otro intervalo. Notar que, por construcción, cada trabajo es asignado máximo a un intervalo y cada intervalo recibe máximo un trabajo asignado. De hecho, demostraremos a continuación que la asignación realizada por el algoritmo está asociada a una solución factible para el programa lineal entero PE-SRDM-D.

Denotaremos por $\pi(j) \in \{0, 1, \cdots, |A|\}$ al índice del intervalo al cual es asignado el trabajo $j \in J$ en el paso 9 del algoritmo, donde $\pi(j) = 0$ indica que el trabajo no fue asignado a ningún intervalo.

Sea $\mathcal{J} = (I^*_1, I^*_2, \cdots, I^*_n)$ el vector de intervalos seleccionados por el Algoritmo 3. Definimos $\hat{z} = h(\mathcal{J})$ como el número de máquinas necesarias para procesar los trabajos asignados a estos intervalos, donde h es la medida definida en la Sección 2.1. (por ejemplo, el mayor número de intervalos de $\mathcal{J}$ que tienen intersección no vacía.) Adicionalmente, definimos el vector $\hat{x} \in \mathbb{R}^{n\times|A|}$ por:

$$\hat{x}_{j,i} = \begin{cases} p, & \text{si el trabajo } j \in J \text{ es asignado al intervalo } I_i \text{ por el Algoritmo 3, es decir,} \\ & \text{si } \pi(j) = i, \\ 0, & \text{caso contrario.} \end{cases}$$

Demostraremos a continuación que $(\hat{x}, \hat{z}) \in \mathbb{R}^{n\times|A|} \times \mathbb{Z}$ es una solución factible del modelo lineal entero. De las definiciones de $\hat{x}$ y $\hat{z}$ sabemos que $\hat{x} \in \{0, p\}^{n\times|A|}$ y $\hat{z} \in \mathbb{Z}_+ \cup \{0\}$. Por lo tanto, para probar que $(\hat{x}, \hat{z})$ es una solución factible del problema PE-SRDM-D nos resta probar que $\hat{x}$ satisface las familias de restricciones (2.3.2) - (2.3.5).

Primero, para que $\hat{x}$ satisfaga la familia de restricciones (2.3.2), debemos probar que

$$\sum_{i\in\{1,2,\cdots,|A|\}} \hat{x}_{j,i} = p, \quad \forall j \in J.$$

Como cada trabajo es asignado a máximo una máquina, conocemos que

$$\sum_{i\in\{1,2,\cdots,|A|\}} \hat{x}_{j,i} \in \{0, p\}.$$

Por lo tanto, $\hat{x}$ satisface la familia de restricciones (2.3.2) si y sólo si

$$\pi(j) \neq 0, \qquad \forall j \in J.$$

Supongamos que existe $j' \in J$ tal que $\pi(j') = 0$. Sea $(x^*, z^*) \in \mathbb{R}^{n\times|A|} \times \mathbb{R}$ la solución obtenida al resolver el programa lineal en el paso 2 del Algoritmo 3. Para cada $j \in J$, se define el intervalo restringido de j, $[r_j^*, d_j^*) \subseteq [r_j, d_j)$, donde $r_j^* = \text{mín}\left\{R(I_i)\colon i \in \{1, 2, \cdots, |A|\}, x_{j,i}^* \neq 0\right\}$ y $d_j^* = \text{máx}\left\{D(I_i)\colon i \in \{1, 2, \cdots, |A|\}, x_{j,i}^* \neq 0\right\}$. Notar que debido a las restricciones (2.3.8) y (2.3.9) se tiene que $r_j \leq r_j^* < d_j^* \leq d_j$.

Sea Υ_1 el conjunto de intervalos de A que están contenidos en el intervalo restringido asociado a j', es decir:

$$\Upsilon_1 = \{I_i \in A\colon I_i \subseteq [r_{j'}^*, d_{j'}^*)\}.$$

Notar que $\Upsilon_1 \neq \emptyset$, pues de lo contrario x^* no satisface las restricciones (2.3.6).

Adicionalmente, si suponemos que

$$\Upsilon_1 = \{I_a, I_{a+1}, \cdots, I_b\}, \qquad \text{con } 1 \leq a < b \leq |A|,$$

entonces de la definición de intervalo restringido y de (2.3.6) se sigue que

$$\begin{aligned} v(I_b) - v(I_{a-1}) &= \sum_{i\in\{1,2,\cdots,b\}} \sum_{j\in J} x_{j,i}^* - \sum_{i\in\{1,2,\cdots,a-1\}} \sum_{j\in J} x_{j,i}^* \\ &= \sum_{i\in\{a,\cdots,b\}} \sum_{j\in J} x_{j,i}^* \\ &\geq \sum_{i\in\{a,\cdots,b\}} x_{j',i}^* \\ &= p. \end{aligned}$$

Por lo tanto, el conjunto Υ_1 contiene al menos uno de los intervalos seleccionados por el Algoritmo 3, en el paso 8. Por otra parte, notar que a cada intervalo $I_i \in \Upsilon_1$ se le deben haber asignado μ_i^* trabajos $j \in J$, con $d_j \leq d_{j'}$, pues de lo contrario el trabajo j' sería asignado a una de las μ_i^* copias de este intervalo.

Consideremos el conjunto de trabajos que han sido asignados a intervalos de Υ_1:

$$J_1 = \{j \in J\colon I_{\pi(j)} \in \Upsilon_1\}.$$

De las observaciones anteriores, se sigue que $J_1 \neq \emptyset$. Sea j_1 un elemento de J_1 que satisface

$$r_{j_1}^* \leq r_j^*, \qquad \forall j \in J_1.$$

Si $r_{j_1}^* \geq r_{j'}^*$, definimos $\hat{j} := j'$. Caso contrario, si $r_{j_1}^* < r_{j'}^*$, consideremos el conjunto

$$\Upsilon_2 = \{I_i \in A\colon I_i \subseteq [r_{j_1}^*, d_{j'}^*)\}.$$

Nuevamente, se cumple que cada intervalo $I_i \in A$, con $I_i \subseteq [r^*_{j_1}, r^*_{j'})$, debe tener asignados μ^*_i trabajos $j \in J$, con $d_j \leq d_{j_1}$, pues de lo contrario el trabajo j_1 habría sido asignado a uno de estos intervalos. Luego, cada intervalo $I_i \in \Upsilon_2$ tiene μ^*_i trabajos asignados, todos ellos con $d_j \leq d_{j'}$. Consideramos ahora el conjunto

$$J_2 = \{j \in J \colon I_{\pi(j)} \in \Upsilon_2\}$$

de todos los trabajos asignados a intervalos en Υ_2 y definamos j_2 como un elemento de J_2 que satisface:

$$r^*_{j_2} \leq r^*_j, \qquad \forall j \in J_2.$$

Si $r^*_{j_2} \geq r^*_{j_1}$, definimos $\hat{j} := j_1$. Caso contrario, si $r^*_{j_2} < r^*_{j_1}$, repetimos la construcción anterior. De esta manera, en un número finito de iteraciones (este número puede ser cero, si $r^*_{j_1} \geq r^*_{j'}$) construimos un conjunto de intervalos

$$\tilde{\Upsilon} = \{I_i \in A \colon I_i \subseteq [r^*_{\hat{j}}, d^*_{j'})\},$$

y un conjunto de trabajos

$$\tilde{J} = \{j \in J \colon I_{\pi(j)} \in \tilde{\Upsilon}\}$$

que satisfacen las siguientes propiedades:

- $r^*_{\hat{j}} = \text{mín}\{r^*_j \colon j \in \tilde{J} \cup \{j'\}\}$,
- $d^*_{j'} = \text{máx}\{d^*_j \colon j \in \tilde{J} \cup \{j'\}\}$,
- cada intervalo $I_i \in \tilde{\Upsilon}$ tiene asignados μ^*_i trabajos de $\tilde{J}$,
- para cada trabajo $j \in \tilde{J}$ se satisface $[r^*_j, d^*_j) \subseteq [r^*_{\hat{j}}, d^*_{j'})$ y por tanto $\sum\limits_{I_i \in \tilde{\Upsilon}} x^*_{j,i} = p$.

Sea $\hat{n} = |\tilde{J}|$. De la última propiedad listada arriba se sigue que

$$\begin{aligned} \sum_{I_i \in \tilde{\Upsilon}} \sum_{j \in J} x^*_{j,i} &\geq \sum_{I_i \in \tilde{\Upsilon}} \sum_{j \in \tilde{J}} x^*_{j,i} + \sum_{I_i \in \tilde{\Upsilon}} x^*_{j',i} \\ &= \hat{n}p + p \\ &= (\hat{n} + 1)p. \end{aligned} \tag{2.3.16}$$

Por otra parte, supongamos que

$$\hat{\Upsilon} = \{I_{\sigma(c_1)}, I_{\sigma(c_2)}, \cdots, I_{\sigma(c_g)}\} \subseteq \tilde{\Upsilon},$$

es el conjunto de intervalos seleccionados por el Algoritmo 3 y que pertenecen al conjunto $\tilde{\Upsilon}$, con $c_1, c_2, \cdots, c_g \in \{1, 2, \cdots, |J|\}$ y tales que

$$c_1 \leq c_2 \leq \cdots \leq c_g.$$

Si $c_1 \geq 2$, por el Lema 2.3.3 aplicado al intervalo $I_{\sigma(c_1-1)}$ tenemos que

$$(\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_1-1)}^*)p \geq v(I_{\sigma(c_1-1)}) > (\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_1-1)}^* - 1)p.$$

Luego, $v(I_{\sigma(c_1-1)}) = (\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_1-1)}^* - 1)p + \delta$, para algún $\delta \in (0, p]$. De manera similar, para el intervalo $I_{\sigma(c_g)}$ se satisface la desigualdad:

$$(\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_g)}^*)p \geq v(I_{\sigma(c_g)}) > (\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_g)}^* - 1)p,$$

de donde $v(I_{\sigma(c_g)}) = (\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_g)}^* - 1)p + \epsilon$, para algún $\epsilon \in (0, p]$.

Por tanto, se sigue que

$$v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) = (\mu_{\sigma(c_1)}^* + \mu_{\sigma(c_1+1)}^* + \cdots + \mu_{\sigma(c_g)}^*)p + \epsilon - \delta,$$

con $\delta, \epsilon \in (0, p]$. Para todo $i \in \{1, 2, \cdots, |A|\} \setminus \{\sigma(c_1), \sigma(c_2), \cdots, \sigma(c_g)\}$ tal que $\sigma(c_1 - 1) + 1 \leq i \leq \sigma(c_g)$, sabemos que $\mu_i^* = 0$. Por lo tanto, se sigue que

$$v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) = (\mu_{\sigma(c_1)}^* + \mu_{\sigma(c_2)}^* + \cdots + \mu_{\sigma(c_g)}^*)p + \epsilon - \delta = \hat{n}p + \epsilon - \delta,$$

con $\delta, \epsilon \in (0, p]$. La última igualdad se sigue del hecho de que cada intervalo $I_i \in \hat{\Upsilon}$ está asignado a un trabajo en $\tilde{J}$ y que ningún trabajo es asignado a más de un intervalo.

Además, de la definición de $v(I_i)$ en el paso 3 del algoritmo, se tiene que:

$$v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) = \sum_{j \in J} x_{j,\sigma(c_1-1)+1}^* + \cdots + \sum_{j \in J} x_{j,\sigma(c_g)-1}^* + \sum_{j \in J} x_{j,\sigma(c_g)}^* \geq \sum_{I_i \in \tilde{\Upsilon}} \sum_{j \in J} x_{j,i}^*.$$

Conjuntamente con (2.3.16), las dos últimas ecuaciones implican que:

$$(\hat{n} + 1)p \leq \sum_{I_i \in \tilde{\Upsilon}} \sum_{j \in J} x_{j,i}^* \leq \hat{n}p + \epsilon - \delta \quad \Longleftrightarrow \quad p \leq \epsilon - \delta,$$

lo que contradice que $\delta, \epsilon \in (0, p]$. Luego, el supuesto inicial de que el trabajo j' no fue procesado en algún intervalo dentro de su intervalo restringido nos lleva hacia una contradicción en este caso.

En el caso restante, si $c_1 = 1$, tenemos que $v(I_{\sigma(c_g)}) = (\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_g)}^* - 1)p + \epsilon$, para algún $\epsilon \in (0, p]$. Además,

$$v(I_{\sigma(c_g)}) = \sum_{j \in J} x_{j,1}^* + \cdots + \sum_{j \in J} x_{j,\sigma(c_g)-1}^* + \sum_{j \in J} x_{j,\sigma(c_g)}^* \geq (\hat{n} + 1)p,$$

y por otra parte $\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_g)}^* = \hat{n}$, de donde se obtiene que

$$(\hat{n} - 1)p + \epsilon \geq (\hat{n} + 1)p \qquad \Longleftrightarrow \qquad \epsilon \geq 2p,$$

lo cual nuevamente contradice que $\epsilon \in (0, p]$. Es decir, nuestro supuesto inicial de que el trabajo j' no fue procesado en algún intervalo dentro de su intervalo restringido no es cierto. En conclusión, todos los trabajos $j \in J$ son asignados a algún intervalo seleccionado por el Algoritmo 3.

Así, hemos probado que se satisfacen las restricciones (2.3.2). Pero como además, todos los trabajos son asignados a algún intervalo seleccionado por el Algoritmo 3 dentro de su intervalo restringido, concluimos también que se satisfacen las restricciones (2.3.4) y (2.3.5).

Por último, verifiquemos que la solución $(\hat{x}, \hat{z})$ satisface las restricciones (2.3.3). Para ello, debemos verificar que se satisface la desigualdad:

$$\sum_{\eta \in \{1,2,\cdots,\kappa\}} \sum_{j \in J} \hat{x}_{j,i+\eta} \leq \text{mín}\{\hat{z}, m_q\}p, \tag{2.3.17}$$

para todo $i \in [|A| - \kappa]$, $\kappa \in \{1, 2, \cdots, \phi\}$ y $q \in P$ tal que $\bigcap_{\eta \in \{1,2,\cdots,\kappa\}} I_{i+\eta} \cap [t_q, t_q + L_q) \neq \emptyset$. Para ello, sean $q \in P$, $\kappa \in \{1, 2, \cdots, \phi\}$, $i \in [|A| - \kappa]$ tales que

$$\bigcap_{\eta \in \{1,2,\cdots,\kappa\}} I_{i+\eta} \cap [t_q, t_q + L_q) \neq \emptyset.$$

Supongamos además que de los intervalos $I_{i+1}, \cdots, I_{i+\kappa}$ el Algoritmo 3 ha seleccionado los intervalos:

$$I_{\sigma(c_1)}, I_{\sigma(c_2)}, \cdots, I_{\sigma(c_g)},$$

donde $\sigma(c_1), \sigma(c_2), \cdots, \sigma(c_g) \in \{i + 1, i + 2, \cdots, i + \kappa\}$ y $c_1 \leq c_2 \leq \cdots \leq c_g$. Como sabemos que μ_i^* es el número de copias seleccionadas del intervalo I_i, para todo $i \in A$, verificar la desigualdad (2.3.17) es equivalente a verificar que

$$(\mu_{\sigma(c_1)}^* + \mu_{\sigma(c_2)}^* + \cdots + \mu_{\sigma(c_g)}^*)p \leq m_q p, \quad \text{y}$$
$$(\mu_{\sigma(c_1)}^* + \mu_{\sigma(c_2)}^* + \cdots + \mu_{\sigma(c_g)}^*)p \leq \hat{z}p.$$

Pero por definición de $\hat{z}$, se tiene que

$$\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)} \leq \hat{z}.$$

Por tanto, resta probar que

$$\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)} \leq m_q.$$

Si $c_1 > 1$, del Lema 2.3.3 sabemos que

$$(\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_1-1)})p \geq v(I_{\sigma(c_1-1)}) \quad \text{y} \quad (\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)} - 1)p < v(I_{\sigma(c_g)}).$$

Entonces,

$$\begin{aligned} v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) &> (\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)} - 1)p - (\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_1-1)})p \\ &= (\mu^*_{\sigma(c_1-1)+1} + \mu^*_{\sigma(c_1-1)+2} + \cdots + \mu^*_{\sigma(c_g)} - 1)p \\ &= (\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)} - 1)p. \end{aligned}$$

De la definición de $v(I_{\sigma(c_g)})$ y $v(I_{\sigma(c_1-1)})$ junto con la restricción (2.3.7), sabemos que

$$v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) = \sum_{j\in J} x^*_{j,\sigma(c_1-1)+1} + \sum_{j\in J} x^*_{j,\sigma(c_1-1)+2} + \cdots + \sum_{j\in J} x^*_{j,\sigma(c_g)} \leq m_q p.$$

Por tanto, se tiene que

$$(\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)} - 1)p < v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) \leq m_q p.$$

De donde

$$(\mu^*_{\sigma(c_1)}+\mu^*_{\sigma(c_2)}+\cdots+\mu^*_{\sigma(c_g)})p-p < m_q p \quad \Longleftrightarrow \quad (\mu^*_{\sigma(c_1)}+\mu^*_{\sigma(c_2)}+\cdots+\mu^*_{\sigma(c_g)})p < (m_q+1)p,$$

y como todas las cantidades son enteras, se concluye

$$\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)} \leq m_q.$$

Ahora, si $c_1 = 1$, del Lema 2.3.3 sabemos que

$$(\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)} - 1)p < v(I_{\sigma(c_g)}).$$

De la definición de $v(I_{\sigma(c_g)})$ junto con la restricción (2.3.7), sabemos que

$$v(I_{\sigma(c_g)}) = \sum_{j\in J} x^*_{j,1} + \sum_{j\in J} x^*_{j,2} + \cdots + \sum_{j\in J} x^*_{j,\sigma(c_g)} \leq m_q p.$$

Por tanto, se tiene que

$$(\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_g)}^* - 1)p < v(I_{\sigma(c_g)}) \leq m_q p.$$

De donde

$$(\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_g)}^*)p - p < m_q p \quad \Longleftrightarrow \quad (\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_g)}^*)p < (m_q + 1)p.$$

Es decir,

$$\mu_1^* + \mu_2^* + \cdots + \mu_{\sigma(c_g)}^* \leq m_q.$$

Como $\mu_i^* = 0$, para cada $i \in \{1, 2, \cdots, |A|\} \setminus \{\sigma(1), \sigma(2), \cdots, \sigma(c_g), \cdots, \sigma(n)\}$, se sigue que

$$\mu_{\sigma(c_1)}^* + \mu_{\sigma(c_2)}^* + \cdots + \mu_{\sigma(c_g)}^* \leq m_q.$$

Así, hemos probado que $\hat{x}$ satisface las familias de restricciones (2.3.2) - (2.3.5). Lo que prueba que la solución $(\hat{x}, \hat{z})$ es una solución factible del modelo PE-SRDM-D. □

Del teorema anterior tenemos que la solución $(\hat{x}, \hat{z})$ es una solución factible del modelo PE-SRDM-D. Si (z^*, x^*) es una solución óptima del modelo PL-SRDM-D se quiere conocer ahora cual es la relación entre z^* y $\hat{z}$. Definamos

$$\hat{v}(I_i) = \sum_{\kappa \in \{1,2,\cdots,i\}} \sum_{j \in J} \hat{x}_{j,\kappa}, \qquad \forall i \in \{1, 2, \cdots, |A|\}.$$

Las funciones $v(\cdot)$ y $\hat{v}(\cdot)$ tiene una relación que será de utilidad para analizar la relación entre z^* y $\hat{z}$, y que resumimos en el siguiente lema.

Lema 2.3.4. *Para cada $i \in \{1, 2, \cdots, |A|\}$ se satisface la siguiente desigualdad:*

$$0 \leq \hat{v}(I_i) - v(I_i) < p.$$

Demostración. Sean $i \in \{1, 2, \cdots, |A|\}$ y $\hat{\lambda} := \frac{1}{p}\hat{v}(I_i)$. Notar que $\hat{\lambda}$ es un entero que indica la cantidad total de trabajos asignados a intervalos en el conjunto $\{I_1, \cdots, I_i\}$. Por construcción, esta cantidad corresponde al número de copias de estos intervalos seleccionados en el paso 7 del Algoritmo 3. Supongamos primero que $\hat{\lambda} < |J|$, esto significa existen dos índices $\hat{\varphi}_1$ y $\hat{\varphi}_2$, con $1 \leq \hat{\varphi}_1 < \hat{\varphi}_2 \leq \bar{\mu}$, tales que:

- $\theta(\hat{\lambda}) = \hat{\varphi}_1$ y $v_{\hat{\varphi}_1} = v(I_i)$, es decir, $v_{\hat{\varphi}_1}$ está asociado a una copia del intervalo I_i.

- $\theta(\hat{\lambda}+1) = \hat{\varphi}_2$ y $v_{\hat{\varphi}_2} = v(I_{i'}) > v(I_i)$, con $i' > i$, es decir, $v_{\hat{\varphi}_2}$ está asociado a un intervalo $I_{i'}$ posterior a I_i.

Según la definición del paso 7 del Algoritmo 3, se sigue que

$$(\hat{\lambda}-1)\cdot p < v(I_i) \leq \hat{\lambda}\cdot p$$

Como $\hat{\lambda} = \frac{1}{p}\hat{v}(I_i)$, se obtiene que

$$\left(\frac{1}{p}\hat{v}(I_i) - 1\right)\cdot p < v(I_i) \leq \frac{1}{p}\hat{v}(I_i)\cdot p,$$

equivalentemente se tiene:

$$\hat{v}(I_i) - p < v(I_i) \leq \hat{v}(I_i).$$

Restando $\hat{v}(I_i)$ y multiplicando por menos uno obtenemos que:

$$0 \leq \hat{v}(I_i) - v(I_i) < p.$$

Por otra parte, si $\hat{\lambda} = |J| = n$, significa que existe un ínidce $\hat{\varphi}$ tal que $\theta(n) = \hat{\varphi}$. De aquí se sigue que

$$v_{\hat{\varphi}} = v(I_i) > (n-1)p.$$

Adicionalmente, de las restricciones (2.3.6) tenemos que

$$\begin{aligned} v(I_i) &= \sum_{\kappa\in\{1,2,\cdots,i\}} \sum_{j\in J} x^*_{j,\kappa} \\ &\leq \sum_{\kappa\in\{1,2,\cdots,|A|\}} \sum_{j\in J} x^*_{j,\kappa} \\ &= \sum_{j\in J} p \\ &= np. \end{aligned}$$

Luego, también en este caso se concluye:

$$(n-1)p < v(I_i) \leq np,$$

equivalentemente

$$\left(\frac{1}{p}\hat{v}(I_i) - 1\right)p < v(I_i) \leq \left(\frac{1}{p}\hat{v}(I_i)\right)p.$$

De donde, se sigue que:

$$0 \leq \hat{v}(I_i) - v(I_i) < p.$$

□

Ahora, sean $i \in [|A| - \kappa]$, $\kappa \in \{1, 2, \cdots, \phi\}$, $q \in P$ tales que

$$\bigcap_{\eta \in \{1,2,\cdots,\kappa\}} I_{i+\eta} \cap [t_q, t_q + L_q) \neq \emptyset.$$

De la definición de $\hat{v}(\cdot)$ se tiene que

$$\sum_{\eta \in \{1,2,\cdots,\kappa\}} \sum_{j \in J} \hat{x}_{j,i+\eta} = \hat{v}(I_{i+\kappa}) - \hat{v}(I_i).$$

Pero además, del Lema 2.3.4 conocemos que $\hat{v}(I_{i+\kappa}) < p + v(I_{i+\kappa})$ y $\hat{v}(I_i) \geq v(I_i)$. Luego, se sigue que

$$\sum_{\eta \in \{1,2,\cdots,\kappa\}} \sum_{j \in J} \hat{x}_{j,i+\eta} < v(I_{i+\kappa}) + p - v(I_i).$$

Por otra parte, de las restricciones (2.3.7) del modelo PL-SRDM-D, se obtiene que

$$v(I_{i+\kappa}) - v(I_i) = \sum_{\eta \in \{1,2,\cdots,\kappa\}} \sum_{j \in J} x^*_{j,i+\eta} \leq z^* p.$$

Combinando las dos desigualdades anteriores, se concluye que

$$\sum_{\eta \in \{1,2,\cdots,\kappa\}} \sum_{j \in J} \hat{x}_{j,i+\eta} < z^* p + p = (z^* + 1)p.$$

Por último, observar que todos los términos del doble sumatorio en el extremo izquierdo de la última desigualdad son múltiplos de p y por ende,

$$\sum_{\eta \in \{1,2,\cdots,\kappa\}} \sum_{j \in J} \hat{x}_{j,i+\eta} \leq \lceil z^* \rceil p.$$

Sea z_{opt} el valor de una solución óptima para PE-SRDM-D. Como z^* es una cota inferior para este valor se tiene que $\lceil z^* \rceil \leq z_{\text{opt}}$. Por otra parte, de la última desigualdad se sigue que si se elige $\hat{z} = \lceil z^* \rceil$, entonces $(\hat{x}, \hat{z})$ es una solución factible, y por tanto óptima, para el modelo PE-SRDM-D. Por tanto, se tiene que el Algoritmo 3 construye una solución óptima del problema PE-SRDM-D.

Capítulo 3

Análisis del problema SRDM-T

En este capítulo se estudian extensiones de los algoritmos descritos en el Capítulo 2 para aplicarlos al problema SRDM-T. También se analizan las cotas de aproximabilidad de dichos algoritmos.

3.1. Complejidad del problema SRDM-T

Como se mencionó en la Sección 2.2, el SRDM es un problema NP-difícil. En esta sección se analiza cómo algunos resultados conocidos acerca de la complejidad computacional del problema SRDM pueden ser trasladados al problema SRDM-T.

En primer lugar, recordemos que el objetivo del problema SRDM-T consiste en minimizar la cantidad de periodos-máquina, mientras que en el problema SRDM se minimiza la cantidad de máquinas requeridas para procesar todos los trabajos. La siguiente proposición es un resultado que se conserva de manera directa del problema SRDM.

Proposición 3.1.1. *No existe un algoritmo polinomial de aproximación para resolver el problema SRDM-T con un factor de aproximación menor que 2, a menos que* $\mathcal{P} = \mathcal{NP}$.

Demostración. Como se indicó en la demostración del Teorema 2.2.1, en [12] se expone el problema de ordenamiento lógico con tiempos de llegada y plazos máximos de ejecución (Scheduling with Release Times and Deadlines, SRD). Dicho problema es $\mathcal{NP}$-completo y pregunta si es posible ejecutar todos los trabajos en una máqui-

na. Para cada trabajo se especifican una duración y una ventana de tiempo en la que debe ser ejecutado. El problema SRD se puede ver como la versión de decisión de un caso especial del SRDM-T, donde se tiene un único periodo ($|P| = 1$): dada una instancia del SRD, esta instancia tiene respuesta afirmativa si y solamente si la solución óptima de la instancia correspondiente del SRDM-T requiere de un periodo-máquina para ejecutar todos los trabajos.(Notar que, al existir un solo periodo, la cantidad de periodos-máquina en cualquier solución coincide con la cantidad de máquinas, lo que nos permite replicar los mismos argumentos de la demostración del Teorema 2.2.1).

Ahora, supongamos que existe un algoritmo polinomial y con un factor de aproximación estrictamente menor que dos para el SRDM-T. Este algoritmo permitiría decidir, en tiempo polinomial, si una instancia del SRD tiene respuesta afirmativa o no. En efecto, una instancia del SRD tiene solución afirmativa si y solamente si, el algoritmo retorna una solución para el problema SRDM-T cuyo valor es estrictamente menor que dos, pues en este caso la solución óptima del SRDM-T requiere de un único periodo-máquina. □

No ha sido posible determinar si el resultado del Teorema 2.2.2 presentado en la Sección 2.2 se puede transferir al problema SRDM-T. En todo caso, es posible formular un problema de decisión asociado al SRDM-T, de manera análoga a cómo se lo realizó en el caso del SRDM. En efecto, si consideramos la tupla de enteros $(k_1, \cdots, k_T)$, con $0 \leq k_q \leq m_q$ para todo $q \in P$, entonces en el problema SRDM-T$(k_1, \cdots, k_T)$ se pregunta si es posible calendarizar un conjunto J de trabajos dentro de un horizonte de tiempo dividido en periodos indexados por P, de tal manera que para cada periodo $q \in P$ se requieran a lo sumo k_q máquinas. Este problema puede ser resuelto en tiempo polinomial si la diferencia entre el ancho de la ventana de tiempo y el tiempo de procesamiento es menor a 2 para cada trabajo, como se demuestra en la siguiente proposición.

Proposición 3.1.2. *El problema de decisión SRDM-T$(k_1, \cdots, k_T)$ con $\gamma_j \leq 1$ para cada trabajo $j \in J$ puede ser resuelto en un tiempo polinomial.*

Demostración. Vamos a construir un grafo similar a aquel descrito en la demostración de la Teorema 2.2.2.

Sea $\mathcal{I} = (J, P, L_P, M, M_P)$ una instancia del problema SRDM-T, con $J = \{1, \cdots, n\}$, y consideremos una instancia asociada del SRDM-T$(k_1, \cdots, k_T)$, con $0 \leq k_q \leq m_q$, para todo $q \in P$. Supongamos que solo $\hat{n}$ trabajos pueden ser procesados en dos posiciones, es decir, podemos dividir el conjunto de trabajos en dos subconjuntos J_1 y J_2 de tal forma que $J = J_1 \sqcup J_2$, con $|J_1| = \hat{n}$ y $|J_2| = n - \hat{n}$, y además se satisface que $\gamma_j = 1$, para todo $j \in J_1$ y $\gamma_j = 0$, para todo $j \in J_2$.

Observar que la instancia del problema SRDM-T$(k_1, k_2, \cdots, k_T)$ tiene solución afirmativa si y solamente si existe un vector de posiciones $\mathcal{L}$ tal que $h(\mathfrak{I}_q^{\mathcal{L}}) \leq k_q$, para cada $q \in P$, donde $\mathfrak{I}_q^{\mathcal{L}}$ es un vector formado por los intervalos del vector $\mathfrak{I}^{\mathcal{L}}$ tales que se intersecan con el intervalo $[t_q, t_q + L_q)$.

Notar que si el número de núcleos de los trabajos que contienen a algún $t \in [r_{\text{mín}}, d_{\text{máx}})$ es mayor que k_q, para algún $q \in P$, entonces la instancia del problema SRDM-T$(k_1, k_2, \cdots, k_T)$ no tiene solución.

Sabemos que cada trabajo $j \in J_1$ puede ser ubicado en una de dos posiciones posibles ($\ell \in \{0, 1\}$), de donde el intervalo $[\tau_j, \tau_j + p_j)$ correspondiente debe contener el instante de tiempo r_j (si $\ell_j = 0$) o $d_j - 1$ (si $\ell_j = 1$). Este hecho, permite formular un problema de flujos para resolver el SRDM-T$(k_1, k_2, \cdots, k_T)$. Para ello, definimos una red capacitada de la siguiente manera: s y u representan el nodo fuente y el nodo sumidero. Para cada trabajo $j \in J_1$ se crea un nodo s_j. Luego, por cada $t \in \{r_{\text{mín}}, r_{\text{mín}} + 1, \cdots, d_{\text{máx}} - 1\}$, si existe al menos un trabajo que contenga a t fuera de su núcleo, se crea un nodo u_t. Entre los nodos s y s_j se crea un arco (s, s_j) de capacidad uno, y de los nodos s_j hacia los nodos u_{r_j} y $u_{d_j - 1}$ (en caso de que existan) de igual manera. Entre los nodos u_t y u se coloca un arco de capacidad $e(t)$, donde, si q es el periodo que contiene a t, $e(t)$ se define como la diferencia entre k_q y el número de núcleos que contienen a t. Notar que si $e(t) = 0$, para algún arco (u_t, u), entonces ninguno de los posibles caminos de s a u que pasan por u_t serán seleccionados para transportar flujo. Por lo tanto, en este caso se pueden eliminar el nodo u_t y sus arcos incidentes de la red capacitada.

Notar que en todo flujo entero factible, únicamente uno de los dos arcos (s_j, u_{r_j}) y $(s_j, u_{d_j - 1})$ tiene flujo positivo, lo que determina una única posición para el trabajo j. La capacidad entre los nodos u_t y u nos asegura que esta selección de intervalos de procesamiento no exceda las k_q máquinas disponibles en cada periodo $q \in P$. Por tanto,

un flujo de s a u retorna un vector de posiciones $\mathcal{L}_1$ para los trabajos de J_1 asociados a arcos (s, s_j) con flujo igual a 1. Este vector puede extenderse de manera natural a un vector $\mathcal{L}$ de posiciones para todos los trabajos. Por construcción, se tiene que $h(\mathfrak{I}_q^{\mathcal{L}}) \leq k_q$, para todo $q \in P$. Notar además que toda asignación de los trabajos a k_q máquinas, para cada $q \in P$, induce un flujo de valor $\hat{n}$ en esta red. Luego, si el flujo máximo en la red es menor a $\hat{n}$, entonces la instancia del problema SRDM-T$(k_1, k_2, \cdots, k_T)$ tiene solución negativa. □

Para comprender un poco mejor la demostración de la proposición anterior, consideremos el siguiente ejemplo.

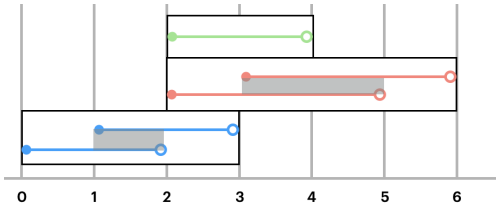
Ejemplo 3.1.1. *Supongamos que se tienen tres trabajos como se describen en la siguiente tabla:*

Trabajo (j)	r_j	d_j	p_j
1	0	3	2
2	2	6	3
3	2	4	2

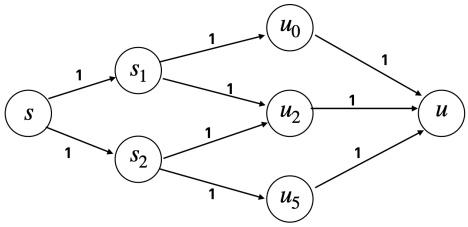
Tabla 3.1: Instancia del SRDM-T con $\gamma_j \leq 1$, $L_P = (2, 2, 2)$ y $M_P = (2, 2, 2)$.

Adicionalmente, sean que $T = 3$, $m = 2$, $L_P = (2, 2, 2)$, $M_P = (2, 2, 2)$, $t_1 = 0$, $t_2 = 2$ *y* $t_3 = 4$.

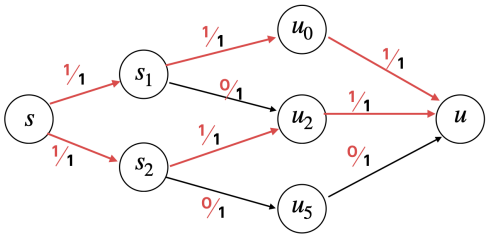
En la Figura 3.1a se muestra gráficamente como se representarían los trabajos en una línea temporal. Cada rectángulo blanco corresponde a la ventana de tiempo de un trabajo, dentro del rectángulo se encuentran dos segmentos de recta que representan los dos posibles intervalos de procesamiento del trabajo. La intersección de los dos posibles intervalos de procesamiento es el núcleo del trabajo y el mismo se muestra como un rectángulo gris. Como el trabajo 3 *admite un único posicionamiento (pues* $\gamma_3 = 0$*), su único posible intervalo de procesamiento* $[2, 4)$ *coincide con su núcleo. En la Figura 3.1b se muestra la red capacitada que se describe en la Proposición 3.1.2. Observar que el trabajo* 3 *no está asociado a ningún nodo de la red.*



(a) Gráfico de los trabajos de la Tabla 3.1 en una linea de tiempo.



(b) Red capacitada en el Teorema 3.1.2.



(c) Flujo máximo sobre la red capacitada.

Figura 3.1: Gráficos de los trabajos descritos en la Tabla 3.1.

Para resolver el problema SRDM-T$(1, 2, 1)$*, se debe encontrar un flujo maximal sobre esta red, el cual se puede observar en color rojo en la Figura 3.1c.*

Como podemos observar, el flujo tiene un valor de dos, por tanto se puede concluir que es posible ejecutar todos los trabajos en esta combinación de periodos-máquina. De hecho, la solución del flujo maximal nos dice que se ejecute cada uno de los dos primeros trabajos en su primera posición posible $(\ell_1 = \ell_2 = 0)$*.*

Por otro lado, podemos ver que el vector $(1, 2, 1)$ *es uno de los 4 posibles vectores con una solución factible. Los otros posibles vectores que proporcionan una solución factible son* $(2, 2, 1)$*,* $(1, 2, 2)$ *y* $(2, 2, 2)$*; y el cuadrado de su norma uno son los valores* 5*,* 5 *y* 6*, respectivamente. De donde tenemos que la solución óptima del problema tiene un valor de* 4 *periodos-máquina.*

Por otra parte, la construcción empleada para obtener un algoritmo polinomial para el SRDM a partir de la solución de múltiples instancias del problema de decisión SRDM(k) no se puede generalizar al SRDM-T. El problema principal radica en definir un orden adecuado sobre las tuplas $(k_1, \cdots, k_T)$ que permita la aplicación de un método de búsqueda eficiente para encontrar la tupla con el menor valor de $\sum_{q \in P} k_q$ que se encuentre asociada a una instancia de respuesta afirmativa. Notar que la inspección exhaustiva de todas las tuplas posibles tiene un orden de complejidad computacional

exponencial en T.

En la Sección 2.2 se comentó que el SRDM es NP-difícil cuando el tiempo máximo de espera de algún trabajo supera las dos unidades. Como es de esperar, este resultado se conserva para el problema SRDM-T y se lo presenta a continuación.

Proposición 3.1.3. *El problema SRDM-T con* $\max\{\gamma_j : j \in J\} \geq 2$ *es NP-difícil.*

Demostración. El resultado se sigue del hecho de que el problema SRDM es NP-difícil cuando $\max\{\gamma_j : j \in J\} \geq 2$ [6] y dicho problema es un caso particular del problema SRDM-T donde todo el horizonte de planificación representa un único periodo y cada máquina se encuentra disponible en cualquier instante de tiempo. □

3.2. Algoritmo Glotón

En esta sección se presentan la adaptación del algoritmo glotón descrito por Spieksma en [25] y el análisis de las cotas de aproximación para el mismo.

Para comprender cómo funciona la adaptación del algoritmo glotón, recordemos que el algoritmo propuesto por Spieksma intenta ejecutar la mayor cantidad de trabajos en una sola máquina disponible durante todo el horizonte de planificación. El Algoritmo 4 consiste en aplicar esta idea iterativamente sobre cada una de las máquinas disponibles. Si al final de estas iteraciones se procesan todos los trabajos, se ha encontrado una solución factible para el problema SRDM-T.

Dada una instancia $\mathcal{I} = (J, P, L_P, M, M_P)$,la solución encontrada por el Algoritmo 4 se guarda en las siguientes variables:

- El conjunto *ind* en el cuál se guardan los índices (j, ℓ) correspondientes al intervalo seleccionado para el trabajo j con ℓ instantes de tiempo de espera para ser procesado.

- La variable $G(\mathcal{I})$ (inicializada en cero) donde se guarda el número de periodos-máquina usados para procesar todos los trabajos.

Adicionalmente, se emplean otras variables que nos permiten conservar la estructura del algoritmo glotón propuesto por Spieksma. S_j representa el conjunto de todos los posibles tiempos de inicio de procesamiento del trabajo $j \in J$. S es la unión de los

conjuntos S_j correspondientes a los trabajos $j \in J$ que aún no han sido procesados, y $s_{\text{máx}} = \text{máx}\{r\colon r \in S\}$. Al inicio del algoritmo, S contiene todos los posibles tiempos de inicio de todos los trabajos.

El Algoritmo 4 retorna como resultado el número de periodos-máquina $G(\mathcal{I})$ usados para ejecutar todos los trabajos. El contador $\hat{n}$ nos garantiza que cada trabajo sea procesado en alguna máquina.

Algoritmo 4: Algoritmo glotón para la calendarización de trabajos.

$\mathcal{I} = (J, P, L_P, M, M_P)$;
ind $:= \emptyset$;
$S_j := \left\{r_{j,0}, \cdots, r_{j,\gamma_j}\right\}$, para todo $j \in J$;
$S := \bigcup_{j \in J} S_j$;
$s_{\text{máx}} := \max_{r \in S} r$;
$J' = \emptyset$;
$\hat{n} := |\text{ind}|$;
$G(\mathcal{I}) := 0$;
mientras $\hat{n} < n$ **hacer**
    $t := -\infty$;
    mientras $s_{\text{máx}} \geq t$ **hacer**
        Seleccionar los índices (j, ℓ) bajo un criterio establecido ;
        ind $:=$ ind $\cup \{(j, \ell)\}$;
        $J' = J' \cup \{j\}$;
        $S := \bigcup_{j' \in J \setminus J'} S_{j'}$;
        $t := d_{j,\ell}$;
        $s_{\text{máx}} := \max_{r \in S} r$;
    fin
    $\hat{n} = |\text{ind}|$;
    $G(\mathcal{I}) = G(\mathcal{I}) + 1$;
fin

La selección de los intervalos en los que se procesan los trabajos se encuentra en el interior del segundo lazo "**mientras**". Primero, determinamos un trabajo j y una posición ℓ para el mismo, bajo un criterio de selección preestablecido. Posteriormente,

se agrega el par (j, ℓ) a la lista de ind, se agrega el trabajo j al conjunto de trabajos ejecutados (J'), y se actualiza el conjunto S. Se registra en la variable t el tiempo de finalización del procesamiento del trabajo j. Se actualiza la variable $s_{\text{máx}}$ y se procede a verificar si existe algún trabajo con un posible tiempo de inicio de procesamiento mayor o igual al tiempo de finalización del último trabajo asignado a la primera máquina (es decir, se verifica si $s_{\text{máx}} \geq t$). De ser el caso, se procede a repetir el lazo interno, caso contrario, significa que la máquina ya no puede procesar más trabajos. En este momento, se verifica si todos los trabajos ya han sido ejecutados. Si no es el caso, se reinicia el proceso en una nueva máquina.

Es posible construir distintas versiones del Algoritmo 4, de acuerdo al criterio empleado para la selección del par (j, ℓ), es decir, del próximo trabajo a ser calendarizado y de su tiempo de inicio de procesamiento. Vamos a considerar aquí cuatro versiones diferentes.

En la primera versión, se utiliza el criterio de selección definido en el Algoritmo 1, que consiste en tomar el intervalo que termina lo más pronto posible, es decir, el par (j, ℓ) que satisface:

$$(j, \ell) := \arg\min \{d_{j,\ell} \mid r_{j,\ell} \in S,\ j \in J \setminus J',\ r_{j,\ell} \geq t\}\,. \tag{3.2.1}$$

En caso de existir empate entre varios candidatos, se selecciona el par que tenga el menor valor de j y el menor valor de ℓ, en ese orden. Para la segunda y tercera versión emplearemos criterios obtenidos de las restricciones débiles para el problema JISP descritas en [24]. Una primera idea consiste en minimizar el tiempo de ejecución, es decir, seleccionar (j, ℓ) de tal forma que la duración de procesamiento sea la menor:

$$(j, \ell) := \arg\min \{d_{j,\ell} - r_{j,\ell} \mid r_{j,\ell} \in S,\ j \in J \setminus J',\ r_{j,\ell} \geq t\}\,. \tag{3.2.2}$$

Al igual que en el primer criterio, en caso de existir empate entre varios candidatos, se selecciona el par que tenga el menor valor de j y el menor valor de ℓ, en ese orden. Otra alternativa es minimizar el tiempo de retraso, es decir, minimizar el tiempo libre entre cambios de trabajos. En la tercera versión, se escoge el par (j, ℓ) que satisface:

$$(j, \ell) := \arg\min \{r_{j,\ell} - t \mid r_{j,\ell} \in S,\ j \in J \setminus J',\ r_{j,\ell} \geq t\}\,. \tag{3.2.3}$$

Nuevamente, en caso de existir empate entre varios candidatos, se selecciona el par que tenga el menor valor de j y el menor valor de ℓ, en ese orden. Por otra parte, es

posible combinar los criterios establecidos en las ecuaciones (3.2.2) y (3.2.3) de manera jerárquica. En la cuarta y última versión del algoritmo, se seleccionan primero los índices (j, ℓ) para los cuales el retraso es el menor y posteriormente se selecciona (de entre estos) el índice tal que su duración sea la menor de todas:

$$(j,\ell) := \arg\min \left\{ d_{j,\ell} - r_{j,\ell} \;\middle|\; \begin{array}{l} r_{j,\ell} \in S, \\ j \in J \setminus J', \\ r_{j,\ell} - t = \min\{r_{j,\ell} - t \mid r_{j,\ell} \in S,\ j \in J \setminus J',\ r_{j,\ell} \geq t\} \end{array} \right\}. \tag{3.2.4}$$

3.2.1. Cotas de aproximación

En el Teorema 2.2.5 se demostró que la adaptación para el SRDM del Algoritmo Glotón de Spieksma tiene una cota superior para el factor de aproximabilidad de orden $O(\log(n))$. Dicho factor de aproximabilidad se conserva para el problema SRDM-T, como se demuestra en la siguiente proposición.

Proposición 3.2.1. *El Algoritmo 4 (versión 1) tiene un factor de aproximabilidad de* $O(T\log(n))$ *para el problema SRDM-T.*

Demostración. Considerar una instancia del SRDM-T y suponer que la solución óptima emplea k_q^* máquinas en cada periodo $q \in P$. Sea $k^* = \max_{q \in P} k_q^*$. De la demostración del Teorema 2.2.5 se puede concluir que el número máximo de iteraciones requeridas por el Algoritmo 4 hasta asignar todos los trabajos es de a lo más $\lceil 2k^* \log(n) \rceil + 1$. Como conocemos que cada máquina puede ser utilizada máximo durante T periodos dentro del horizonte de tiempo, se concluye que a lo más se usan $\lceil 2k^* \log(n) + 1 \rceil T$ periodos-máquina por el Algoritmo 4. Por otra parte, la solución óptima usa simultáneamente k^* máquinas al menos durante un periodo, por lo que su valor es de al menos k^* periodos-máquina. Luego, el factor de aproximabilidad del algoritmo se encuentra acotado superiormente por $O(T\log(n))$. □

Basados en la demostración del Teorema 2.2.6 se propone la siguiente proposición para determinar una cota inferior al factor de aproximabilidad del Algoritmo 4 al resolver el problema SRDM-T.

Proposición 3.2.2. *El Algoritmo 4 (versión 2) tiene un factor de aproximabilidad de* $\Omega(\log(n))$.

Demostración. Puede utilizarse la misma familia de instancias $\mathcal{I}_{\hat{h}} = \left(J_{\hat{h}}, P, L_P, M, (\hat{h})\right)$ descritas en el Teorema 2.2.6, donde $P = \{1\}$, $L_P = (L_{\hat{h}})$, $M = \{1, 2, \cdots, \hat{h}\}$ y $\hat{h} \in \mathbb{N}^*$. Con los argumentos expuestos en la demostración de ese teorema, si se elige $L_{\hat{h}}$ suficientemente grande, por ejemplo, $L_{\hat{h}} \geq \text{máx}\{d_j : j \in J_{\hat{h}}\}$, entonces puede demostrarse que la solución óptima de la instancia $\mathcal{I}_{\hat{h}}$ del SRDM-T requiere de 1 periodo-máquina, pues consiste en asignar todos los trabajos a una única máquina dentro del primer (y único) periodo de la instancia. Por otra parte, debido a que por construcción de la instancia se tiene que $p_{j_1} \leq p_{j_2}$ para todo $j_1 < j_2$, con $j_1, j_2 \in J_{\hat{h}}$, la solución del Algoritmo 4 (en su versión 2) asigna $\hat{h}$ trabajos a máquinas distintas en el instante cero, por lo que requiere de al menos $\hat{h} = \log_2(n+1)$ periodos-máquina. De aquí se concluye que el factor de aproximabilidad del Algoritmo 4 (versión 2) es de $\Omega(\log(n))$. □

Para comprender un poco mejor la demostración del teorema anterior se plantea el siguiente ejemplo.

Ejemplo 3.2.1. *Para* $\hat{h} = 1$*, se tiene la instancia* $\mathcal{I}_1 = (J_1, P, L_P, M, M_P)$*, donde* $J_1 = \{1\}$*,* $r_1 = 0$*,* $d_1 = 2$*,* $p_1 = 1$*,* $P = \{1\}$ $L_P = (2)$*,* $M = \{1\}$ *y* $M_P = (1)$*. El Algoritmo 4 nos dice que seleccionemos el intervalo* $[0, 1)$ *para procesar el trabajo 1, el cual ocupa una máquina. Es decir, tenemos que la solución del algoritmo requiere de* $\hat{h} = 1 = Opt(\mathcal{I}_1)$ *periodos-máquina.*

Ahora, para $\hat{h} = 2$*, del Ejemplo 2.2.2 tenemos la instancia* $\mathcal{I}_2 = (J_2, P, L_P, M, M_P)$*, donde* $J_2 = \{1, 2, 3\}$*,* $L_P = (6)$*,* $M = \{1, 2\}$ *y* $M_P = (2)$*. Los trabajos de* J_2 *se muestran en la Tabla 3.2*

Trabajo	r_j	d_j	p_j
1	0	2	1
2	2	6	2
3	0	5	2

Tabla 3.2: Trabajos de la instancia $\mathcal{I}_2$.

El Algoritmo 4 nos dice que definamos $S_1 = \{0, 1\}$*,* $S_2 = \{2, 3, 4\}$*,* $S_3 = \{0, 1, 2, 3\}$*,* $S = \{0, 1, 2, 3, 4\}$*,* $s_{\text{máx}} = 4$*,* $J' = \emptyset$*,* $ind = \emptyset$*,* $\hat{n} = 0$ *y* $G(\mathcal{I}_2) = 0$*. Luego, como*

$\hat{n} = 0 < 3 = n$, definimos $t = -\infty$. Como $s_{\text{máx}} = 4 \geq t$, se realizan los siguientes pasos:

- $(j, \ell) = \arg\min \{d_{j,\ell} - r_{j,\ell} \mid r_{j,\ell} \in S,\ j \in J \setminus J',\ r_{j,\ell} \geq t\} = \arg\min\{d_{1,0} - r_{1,0} = 1, d_{1,1} - r_{1,1} = 1, d_{2,0} - r_{2,0} = 2, \cdots, d_{3,3} - r_{3,3} = 2\} = (1, 0)$.
- $ind = \{(1, 0)\}$.
- $J' = \{1\}$.
- $S = \{0, 1, 2, 3, 4\}$.
- $t = d_{1,0} = 1$.
- $s_{\text{máx}} = 4$.

Como $s_{\text{máx}} = 4 \geq 1 = t$, se vuelve a ejecutar el mismo lazo interno, obteniendo:

- $(j, \ell) = \arg\min \{d_{j,\ell} - r_{j,\ell} \mid r_{j,\ell} \in S,\ j \in J \setminus J',\ r_{j,\ell} \geq t\} = \arg\min\{d_{2,0} - r_{2,0}, d_{2,1} - r_{2,1}, d_{2,2} - r_{2,2}, d_{3,1} - r_{3,1}, d_{3,2} - r_{3,2}, d_{3,3} - r_{3,3}\} = (2, 0)$.
- $ind = \{(1, 0), (2, 0)\}$.
- $J' = \{1, 2\}$.
- $S = \{0, 1, 2, 3\}$.
- $t = d_{2,0} = 4$.
- $s_{\text{máx}} = 3$.

Como $s_{\text{máx}} = 3 \ngeq 4 = t$, se actualiza los siguientes valores:

- $\hat{n} = 2$.
- $G(\mathcal{I}_2) = 1$.

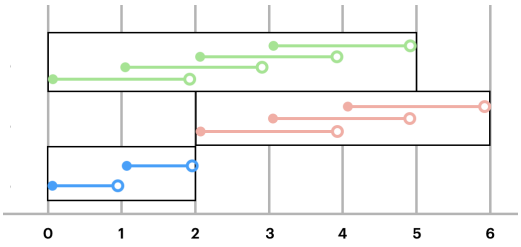
Como $\hat{n} = 2 < 3 = n$, se define $t = -\infty$ y se vuelve a ejecutar el lazo interno en una nueva máquina. Obteniendo:

- $(j, \ell) = \arg\min \{d_{j,\ell} - r_{j,\ell} \mid r_{j,\ell} \in S,\ j \in J \setminus J',\ r_{j,\ell} \geq t\} = \arg\min\{d_{3,0} - r_{3,0}, d_{3,1} - r_{3,1}, d_{3,2} - r_{3,2}, d_{3,3} - r_{3,3}\} = (3, 0)$.

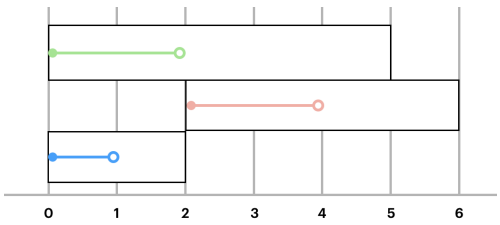
- $ind = \{(1,0),(2,0),(3,0)\}$.
- $J' = \{1,2,3\}$.
- $S = \emptyset$.
- $t = d_{3,0} = 2$.
- $s_{\text{máx}} = 0$.

Como $s_{\text{máx}} = 0 \not\geq 2 = t$, *se termina el lazo interno. Luego, actualizamos* $\hat{n} = 3$ *y* $G(\mathcal{I}_2) = 2$. *Verificamos que* $\hat{n} = 3 \not< 3 = n$, *y terminamos el lazo.*

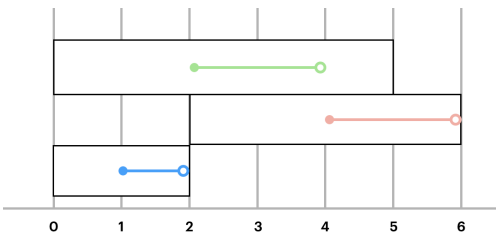
Con lo cual, se tiene que el número de solapamientos en el instante 0 *es* $\hat{h} = 2 = G(\mathcal{I}_2)$. *Adicionalmente, se tiene que todos los trabajos pueden ser ejecutados en una sola máquina* ($Opt(\mathcal{I}_2) = 1$) *como podemos observar en la Figura 3.2. En la Figura 3.2a se muestra una visualización gráfica de los trabajos de la instancia* $\mathcal{I}_2$. *En un rectángulo blanco se muestra la ventana de tiempo asociada a cada trabajo. Dentro de cada ventana de tiempo se muestran las posibles posiciones de los intervalos de ejecución de cada trabajo. La Figura 3.2b indica los intervalos seleccionados por el algoritmo, mientras que la Figura 3.2c indica los intervalos a seleccionar en la solución óptima.*



(a) Gráfico de los trabajos de la instancia $\mathcal{I}_2$



(b) Gráfico de la solución encontrada por el Algoritmo 4 aplicado a la instancia $\mathcal{I}_2$



(c) Gráfico de una solución óptima de la instancia $\mathcal{I}_2$

Figura 3.2: Gráficos de la instancia $\mathcal{I}_2$.

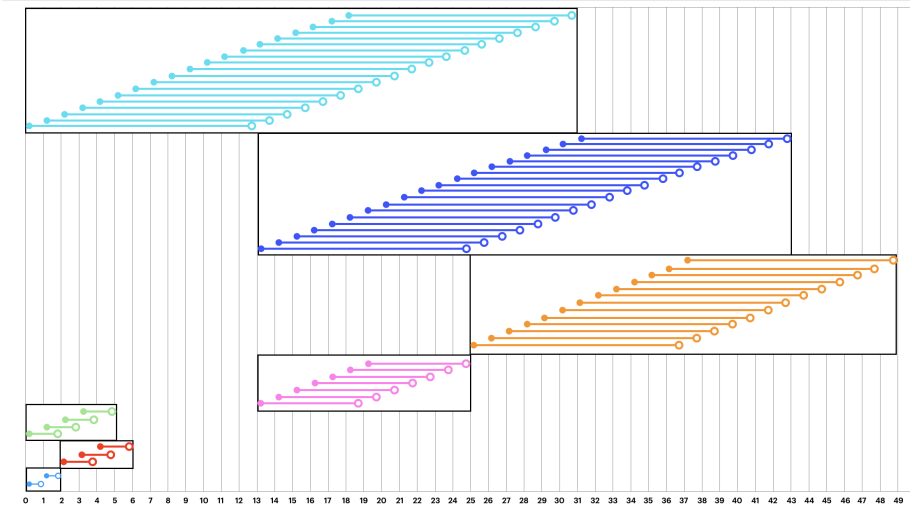
Ahora, para $\hat{h} = 3$, del Ejemplo 2.2.2 tenemos la instancia $\mathcal{I}_3 = (J_3, P, L_P, M, M_P)$, donde $J_3 = \{1, 2, \cdots, 7\}$, $L_P = (49)$, $M = \{1, 2, 3\}$ y $M_P = (3)$. Los trabajos de J_3 se muestran en la Tabla 3.3

Trabajo	r_j	d_j	p_j
1	0	2	1
2	2	6	2
3	0	5	2
4	13	25	6
5	25	49	12
6	13	43	12
7	0	31	13

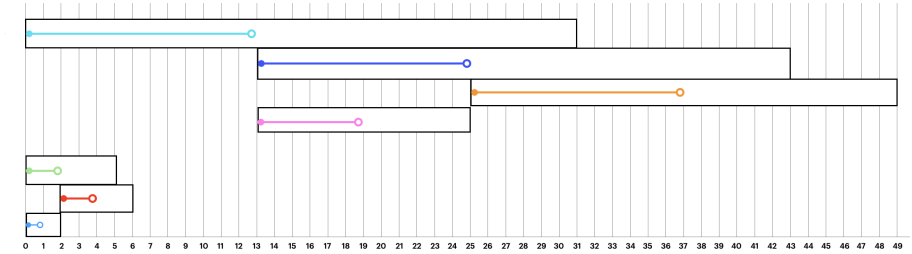
Tabla 3.3: Trabajos de la instancia $\mathcal{I}_3$.

En la Figura 3.3a se muestra una visualización gráfica de los trabajos de la instancia $\mathcal{I}_3$. En un rectángulo blanco se muestra la ventana de tiempo asociada a cada trabajo. Dentro de cada ventana de tiempo se muestran las posibles posiciones de los intervalos de ejecución de cada trabajo.

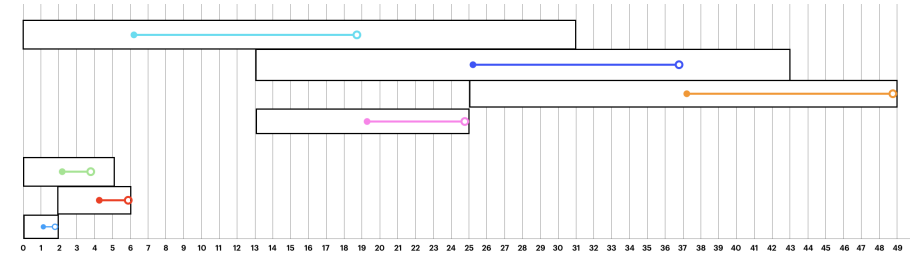
La solución obtenida por el Algoritmo 4 aplicado a la instancia $\mathcal{I}_3$, se muestra en la Figura 3.3b. Como se puede notar en el instante cero se encuentra el mayor número de solapamientos. Por tanto, se necesitan tres máquinas, mientras que como se observa en la Figura 3.3c, todos los trabajos pueden ser procesados en una sola máquina ($Opt(\mathcal{I}_3) = 1$).



(a) Gráfico de los trabajos de la instancia $\mathcal{I}_3$



(b) Gráfico de la solución encontrada por el Algoritmo 4 aplicado a la instancia $\mathcal{I}_3$



(c) Gráfico de una solución óptima de la instancia $\mathcal{I}_3$

Figura 3.3: Gráficos de la instancia $\mathcal{I}_3$.

3.3. Algoritmo Glotón de Mejor Ajuste

En esta sección se analiza el algoritmo GBF para el problema SRDM-T. Notemos que el valor de la solución encontrada por el algoritmo para la instancia $\mathcal{I}$ es:

$$\mathrm{GBF}(\mathcal{I}) := \sum_{q \in P} \max_{t \in [t_q, t_q + L_q)} h(I_{\mathrm{Sol}}, t),$$

I_{Sol} es el vector de intervalos obtenido por el Algoritmo 2. Para el problema SRDM la solución tiene el valor de $\mathrm{Alg2}(\mathcal{I}) = h(I_{\mathrm{Sol}})$.

Ahora, analizamos la cota inferior del Algoritmo 2 para el problema SRDM-T. La siguiente proposición nos muestra la conservación de la cota inferior respecto al problema SRDM.

Proposición 3.3.1. *El Algoritmo 2 tiene un factor de aproximabilidad de* $\Omega(\log(n))$ *para el problema SRDM-T.*

Demostración. Generando la misma familia de instancias del Teorema 2.2.6 se tiene que para atender todos los trabajos empleando el Algoritmo 2, al menos se requieren $\frac{\log(n+1)}{\log(2)}$ máquinas. De hecho, si solo se tiene un periodo ($T = 1$), se tiene que el número de periodos-máquina es igual a $\frac{\log(n+1)}{\log(2)}$. En general, para $T \geq 1$, se tiene que a lo más se requieren

$$T * \frac{\log(n+1)}{\log(2)}$$

periodos-máquina. Por tanto, se tiene una cota inferior de $\Omega(\log(n))$ para el Algoritmo 2. □

Por otra parte, en [6] se muestra que para el problema SRDM la solución obtenida por el Algoritmo 2 difiere de un factor de asintótico $9\frac{p_{\text{máx}}}{p_{\text{mín}}}$, donde $p_{\text{máx}} = \max_{j \in J} p_j$ y $p_{\text{mín}} = \min_{j \in J} p_j$. Basados en dicho análisis se plantea el siguiente teorema.

Teorema 3.3.1. *Si* $\delta_j \geq 2p_j - 1$ *para todo* $j \in J$*, entonces el Algoritmo 2 tiene un factor de aproximabilidad asintótico de* $9T\frac{p_{\text{máx}}}{p_{\text{mín}}}$ *para el problema SRDM-T.*

Demostración. Considerar una instancia $\mathcal{I} = (J, P, L_P, M, M_P)$ del problema SRDM-T. Sean $\text{Sol}_{\text{GBF}}(\mathcal{I}) = (\mathcal{J}_{\text{GBF}}, \mathcal{Z}_{\text{GBF}})$ la solución encontrada por el Algoritmo 2 para la instancia $\mathcal{I}$ y $\text{Sol}_{\text{Opt}}(\mathcal{I}) = (\mathcal{J}_{\text{Opt}}, \mathcal{Z}_{\text{Opt}})$ la solución óptima de $\mathcal{I}$. Sabemos que $\mathcal{J}_{\text{GBF}} = I_{\text{Sol}}$ y $z_q = \max_{t \in [t_q, t_q + L_q)} h(\mathcal{J}_{\text{GBF}}, t)$, para cada $q \in P$, donde $\mathcal{Z}_{\text{GBF}} = (z_1, z_2, \cdots, z_T)$. Además, $\mathcal{Z}_{\text{Opt}} = (z_1^*, z_2^*, \cdots, z_T^*)$.

Adicionalmente, sean $z_{\text{máx}} := \text{máx}\{z_q \colon q \in P\}$ y $z^*_{\text{máx}} := \text{máx}\{z_q^* \colon q \in P\}$.

En primer lugar, notar que para el costo de la solución óptima se cumple:

$$\text{costo}(\mathcal{Z}_{\text{Opt}}) = \sum_{q \in P} z_q^* \geq z^*_{\text{máx}}. \tag{3.3.1}$$

Por otra parte, sea $\hat{q} \in P$ el periodo para el cual se tiene que $z_{\hat{q}} = z_{\text{máx}}$. Sea además, $j \in J$ el trabajo que satisface la propiedad de que, tras asignar j, el valor de $z_{\hat{q}}$ alcanza por primer vez el valor de $z_{\text{máx}}$.

Definimos $J' \subseteq J$ como el conjunto de trabajos procesados antes que j para los cuales el Algoritmo 2 ha seleccionado intervalos que se intersecan con la ventana de tiempo $[r_j, d_j)$. Sea $\mathcal{J}'_{\text{Opt}} \subseteq \mathcal{J}_{\text{Opt}}$ el conjunto de intervalos asignados a los trabajos de J' en la solución óptima. Notar que, por construcción del Algoritmo 2, las ventanas de tiempo de todos estos trabajos están siempre contenidos en $[r_j - \delta_j, d_j + \delta_j)$, donde $\delta_j = d_j - r_j$ es el tamaño de la ventana de tiempo asociada al trabajo j. Luego, todos los intervalos de $\mathcal{J}'_{\text{Opt}}$ están contenidos en $[r_j - \delta_j, d_j + \delta_j)$. Consideremos el área definida por:

$$\omega_j^* = \int_{r_j - \delta_j}^{d_j + \delta_j} h(\mathcal{J}'_{\text{Opt}}, t)dt,$$

donde, conforme se definió en la Sección 2.1, $h(\mathcal{J}'_{\text{Opt}}, t)$ es la cantidad de intervalos de $\mathcal{J}'_{\text{Opt}}$ que contienen al instante t.

Empleando los mismos argumentos utilizados en la demostración del Teorema 2.2.7, concluimos que

$$z^*_{\text{máx}} \geq \frac{\omega_j^*}{3\delta_j} > \frac{z_{\text{máx}} - 1}{9} \cdot \frac{p_{\text{mín}}}{p_{\text{máx}}} \tag{3.3.2}$$

Por último, observar que

$$\text{costo}(\mathcal{Z}_{\text{GBF}}) = \sum_{q \in P} z_q \leq T z_{\text{máx}}. \tag{3.3.3}$$

De (3.3.1), (3.3.2) y (3.3.3) obtenemos:

$$\text{costo}(\mathcal{Z}_{\text{Opt}}) \geq z^*_{\text{máx}} > \frac{z_{\text{máx}} - 1}{9} \frac{p_{\text{mín}}}{p_{\text{máx}}} \geq \frac{p_{\text{mín}}}{9p_{\text{máx}}} \left(\frac{1}{T} \text{costo}(\mathcal{Z}_{\text{GBF}}) - 1 \right).$$

Equivalentemente,

$$\text{costo}(\mathcal{Z}_{\text{GBF}}) < \frac{9Tp_{\text{máx}}}{p_{\text{mín}}} \text{costo}(\mathcal{Z}_{\text{Opt}}) + T.$$

Luego, el Algoritmo 2 tiene un factor asintótico de aproximabilidad de $\frac{9Tp_{\text{máx}}}{p_{\text{mín}}}$. □

Ahora presentamos un ejemplo para comprender un poco mejor el porque no se puede demostrar la desigualdad $z_q < 9\frac{p_{\text{máx}}}{p_{\text{mín}}} z^*_q + 1$ para cada $q \in P$.

Ejemplo 3.3.1. *Consideremos la instancia $\mathcal{I}_5 = (J, P, L_P, M, M_P)$, donde $J = \{1, 2, 3\}$, $P = \{1, 2\}$, $L_P = (5, 5)$, $M = 2$ y $M_P = (2, 2)$. En la Tabla 3.4 se muestran los detalles de los trabajos de J. Además, definimos $t_1 = 0$ y $t_2 = 5$.*

Si resolvemos la instancia $\mathcal{I}_5$ usando el Algoritmo 2 sabemos que se usan 4 periodos-máquina, la solución la podemos observar en la Figura 3.4a. Por otro lado, se puede

Trabajo	r_j	d_j	p_j
1	4	8	3
2	4	9	4
3	3	10	2

Tabla 3.4: Trabajos ordenados de la instancia $\mathcal{I}_5$.

procesar todos los trabajos en 2 periodos-máquina como se puede observar en la Figura 3.4b.

Así, podemos escribir $\mathrm{Sol}_{GBF}(\mathcal{I}_5) = (\mathcal{J}, \mathcal{Z})$*, con* $\mathcal{J} = ([4,7),[4,8),[8,10))$ *y* $\mathcal{Z} = (z_1 = 2, z_2 = 2)$*. Adicionalmente tenemos que* $z_1^* = 0$ *y* $z_2^* = 2$ *son el número óptimo de máquinas encendidas en cada periodo, respectivamente. En la Figura 3.4c se muestra un gráfico de la función* $h(\mathcal{J}, t)$*, con* $t \in [0, 10)$*.*

Consideremos el primer periodo. El primer trabajo que hace que la solución del Algoritmo 2 pase de 1 a 2 es el trabajo 2*, es decir,*

$$\mathcal{J}' = ([4,7)).$$

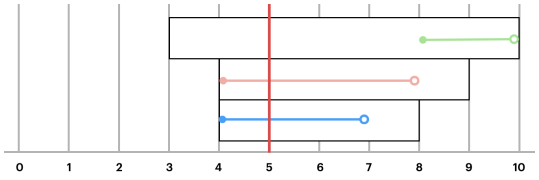
Adicionalmente, sabemos que $h(\mathcal{J}') = 1$*,* $\delta_2 = 5$ *y* $\mathcal{J}'' = \mathcal{J}'$*. Por tanto, se tiene que*

$$\omega_2(\mathcal{J}'') = \int_4^9 h(\mathcal{J}'', t)dt = 3.$$

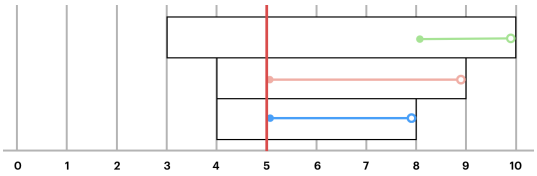
De donde se puede verificar que:

$$z_1^* = 0 \ngeq \frac{1}{5} = \frac{\omega_2(\mathcal{J}'')}{3\delta_2}.$$

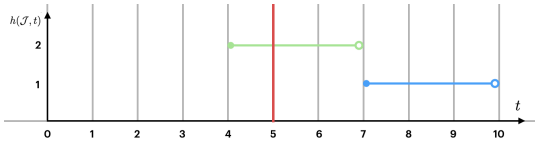
Es decir, el análisis utilizado en la demostración del Teorema 2.2.7 no puede aplicarse de manera independiente para cada periodo.



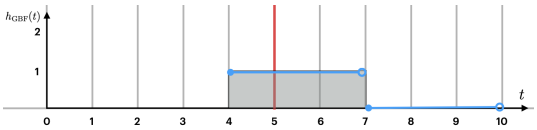
(a) Solución de la instancia $\mathcal{I}_5$ encontrada por el Algoritmo 2.



(b) Solución óptima de la instancia $\mathcal{I}_5$.



(c) Gráfico del número de solapamientos generados por el vector $\mathcal{J}$.



(d) Gráfico del número de solapamientos generados por el vector $\mathcal{J}''$.

Figura 3.4: Gráficos para la instancia $\mathcal{I}_5$ descrita en la Tabla 3.4.

Adicionalmente, en [6] se muestra que para el problema SRDM la solución obtenida por el Algoritmo 2 admite un factor de aproximabilidad asintótico de $18\log_2\left(\frac{p_{\text{máx}}}{p_{\text{mín}}}\right)$, particionando el conjunto de trabajos de tal forma que el cociente entre $p_{\text{máx}}$ y $p_{\text{mín}}$ para cada subconjunto de trabajos sea siempre menor que dos. Utilizando el mismo argumento, se plantea la siguiente proposición.

Proposición 3.3.2. *La solución obtenida por el Algoritmo 2 para el problema SRDM-T admite un factor de aproximabilidad asintótico de* $18T\log_2\left(\frac{p_{\text{máx}}}{p_{\text{mín}}}\right)$, *cuando* $p_{\text{máx}} > p_{\text{mín}}$ *y* $\delta_j \geq 2p_j - 1$ *para todo* $j \in J$.

Demostración. Dada una instancia $\mathcal{I} = (J, P, L_P, M, M_P)$ del problema SRDM-T. Consideremos los siguientes conjuntos de trabajos:

$$J_\eta = \left\{ j \in J \colon \eta - 1 \leq \log_2\left(\frac{p_j}{p_{\text{mín}}}\right) < \eta \right\}, \qquad \forall \eta \in \left\{1, 2, \cdots, \left\lceil \log_2\left(\frac{p_{\text{máx}}}{p_{\text{mín}}}\right)\right\rceil\right\}.$$

Podemos ver que $J = J_1 \cup J_2 \cup \cdots \cup J_{\left\lceil \log_2\left(\frac{p_{\text{máx}}}{p_{\text{mín}}}\right)\right\rceil}$ y $J_{\eta_1} \cap J_{\eta_2} = \emptyset$ para cada $\eta_1, \eta_2 \in \left\{1, 2, \cdots, \left\lceil \log_2\left(\frac{p_{\text{máx}}}{p_{\text{mín}}}\right)\right\rceil\right\}$, con $\eta_1 \neq \eta_2$.

Notemos que para un η cualquiera en el conjunto $\left\{1, 2, \cdots, \left\lceil \log_2\left(\frac{p_{\text{máx}}}{p_{\text{mín}}}\right)\right\rceil\right\}$, se sigue que

$$J_\eta = \left\{ j \in J \colon \eta - 1 \leq \log_2\left(\frac{p_j}{p_{\text{mín}}}\right) < \eta \right\} = \left\{ j \in J \colon 2^{\eta-1} p_{\text{mín}} \leq p_j < 2^\eta p_{\text{mín}} \right\}.$$

Del Teorema 3.3.1 se obtiene entonces que

$$\text{GBF}(\mathcal{I}_\eta) < 9T \cdot \frac{2^\eta p_{\text{mín}}}{2^{\eta-1} p_{\text{mín}}} \text{Opt}(\mathcal{I}_\eta) + T \leq 18T\text{Opt}(\mathcal{I}) + T,$$

donde $\mathcal{I}_\eta = (J_\eta, P, L_P, M, M_P)$.

Sumando sobre todos los η se tiene que

$$\mathrm{GBF}(\mathcal{I}) \leq \sum_{\eta} \mathrm{GBF}(\mathcal{I}_\eta) < 18T \left\lceil \log_2 \left(\frac{p_{\text{máx}}}{p_{\text{mín}}} \right) \right\rceil \mathrm{Opt}(\mathcal{I}) + T \left\lceil \log_2 \left(\frac{p_{\text{máx}}}{p_{\text{mín}}} \right) \right\rceil .$$

□

3.4. Problema SRDM-T con duraciones iguales

En esta sección se analizan los Algoritmos 2 y 3 para una versión particular del problema SRDM-T en la que todos los trabajos tienen igual tiempo de ejecución, es decir, que $p_j = p$ para todo $j \in J$. Al problema SRDM-T con duraciones iguales lo denotaremos como SRDM-TD.

3.4.1. Algoritmo Glotón de Mejor Ajuste

Recordemos que el Algoritmo 2 nos entrega como salida un vector I_{Sol} de intervalos asignados a cada uno de los trabajos. El valor de la solución encontrada por el Algoritmo 2 para una instancia $\mathcal{I}$ es:

$$\mathrm{GBF}(\mathcal{I}) := \sum_{q \in P} \max_{t \in [t_q, t_q + L_q)} h(I_{\text{Sol}}, t).$$

En [6] se muestra que para el caso de instancias con tiempos de procesamiento iguales (p unidades de tiempo) el Algoritmo 2 admite un factor de aproximabilidad asintótico constante igual a nueve para el problema SRDM. A partir de dicho resultado y del Teorema 3.3.1 se obtiene lo siguiente.

Proposición 3.4.1. *El Algoritmo 2 admite un factor de aproximación asintótico de $9T$ para el problema SRDM-TD.*

En [27] se presenta un nuevo análisis del Algoritmo 2, que permite disminuir su factor de aproximación de 9 a 6 para el problema SRDM. Usando las ideas allí propuestas, se demuestra en la siguiente proposición una cota superior de $7T$ para el factor de aproximabilidad del Algoritmo 2 en el problema SRDM-TD.

Proposición 3.4.2. *El Algoritmo 2 admite un factor de aproximación de $7T$ para el problema SRDM-TD, para todas las intancias del SRDM-TD donde la ventana de tiempo de cada trabajo es de al menos $2p - 1$.*

Demostración. Considerar una instancia $\mathcal{I} = (J, P, L_P, M, M_P)$ del problema SRDM-T. Sean $\text{Sol}_{\text{GBF}}(\mathcal{I}) = (\mathcal{J}_{\text{GBF}}, \mathcal{Z}_{\text{GBF}})$ la solución encontrada por el Algoritmo 2 para la instancia $\mathcal{I}$ y $\text{Sol}_{\text{Opt}}(\mathcal{I}) = (\mathcal{J}_{\text{Opt}}, \mathcal{Z}_{\text{Opt}})$ la solución óptima de $\mathcal{I}$. Sabemos que $\mathcal{J}_{\text{GBF}} = I_{\text{Sol}}$ y $z_q = \max\limits_{t \in [t_q, t_q + L_q)} h(\mathcal{J}_{\text{GBF}}, t)$, para cada $q \in P$, donde $\mathcal{Z}_{\text{GBF}} = (z_1, z_2, \cdots, z_T)$. Además, $\mathcal{Z}_{\text{Opt}} = (z_1^*, z_2^*, \cdots, z_T^*)$. Al igual que en la demostración del Teorema 3.3.1, designaremos por $z_{\text{máx}} := \text{máx}\{z_q \colon q \in P\}$ al máximo número de máquinas utilizadas simultáneamente por el Algoritmo 2, y por $z^*_{\text{máx}} := \text{máx}\{z_q^* \colon q \in P\}$ al máximo número de máquinas que se usan simultáneamente en una solución óptima. Sean $j \in J$ el trabajo para cuyo procesamiento el Algoritmo 2 requiere por primera vez de z_q máquinas en el periodo q, y J' el conjunto de trabajos procesados antes que el trabajo j y para los cuales el Algoritmo 2 ha seleccionado intervalos que se intersecan con el intervalo $[r_j, d_j)$. Asimismo, denotaremos por $\mathcal{J}'$ a los intervalos asignados a los trabajos de J' en la solución encontrada por el Algoritmo 2 y $\mathcal{J}'_{\text{Opt}}$ a los intervalos asignados a estos trabajos en la solución óptima. Finalmente, ω_j es la cantidad total de trabajos (o fracciones de trabajo) que son procesados dentro de la ventana de tiempo $[r_j, d_j)$ en la solución encontrada por el Algoritmo 2, es decir,

$$\omega_j := \int_{r_j}^{d_j} h(\mathcal{J}', t)dt.$$

En la demostración del Teorema 3.3.1 ω_j es acotado superiormente al notar que todo trabajo en J' debe ser asignado a un intervalo completamente contenido en $[r_j - \delta_j, d_j + \delta_j)$. Aquí, refinaremos esta observación conforme a las ideas en [27], y demostraremos que si el Algoritmo 2 procesa un trabajo $j' \in J'$ durante $\bar{p} \leq p$ unidades de tiempo en el intervalo $[r_j, d_j)$, entonces cualquier solución factible (y en particular en cualquier solución óptima), el trabajo j' se procesará también al menos durante $\bar{p}$ unidades de tiempo dentro del intervalo $[r_j, \delta_j + p, d_j + \delta_j - p)$.

Sabemos que en cada una de las $z_{\text{máx}} - 1$ máquinas encendidas en dentro del intervalo $[r_j, d_j)$, existe al menos un trabajo cuyo tiempo de inicio de ejecución se encuentra dentro del intervalo $[r_j, d_j)$ o su tiempo de finalización de procesamiento se encuentra dentro del intervalo $(r_j, d_j]$. Sin pérdida de generalidad supongamos que j' es un trabajo cuyo tiempo de inicio de ejecución se encuentra dentro del intervalo $[r_j, d_j)$. Sea $\bar{p} \in (0, p]$ la cantidad de tiempo procesada del trabajo j' en el intervalo $[r_j, d_j)$.

Así, tenemos que

$$r_j \leq \tau_{j'} < \tau_{j'} + \bar{p} \leq d_j \leq \tau_{j'} + p \leq d_{j'},$$

con $\tau_{j'}$ el tiempo de inicio de procesamiento del trabajo j' en la solución encontrada por el Algoritmo 2.

De donde se sigue que

$$r_j + p \leq \tau_{j'} + p \leq d_{j'} \qquad \text{y} \qquad r_{j'} \leq \tau_{j'} \leq d_j - \bar{p}.$$

Como, por construcción del Algoritmo 2, $\delta_j \geq d_{j'} - r_{j'}$, se sigue que

$$r_j - (\delta_j - p) \leq r_j - (d_{j'} - r_{j'} - p) = r_{j'} + r_j + p - d_{j'} \leq r_{j'},$$

y

$$d_{j'} - p + \bar{p} \leq \delta_j + r_{j'} - p + \bar{p} \leq d_j + \delta_j - p.$$

Estos resultados implican que en cualquier solución factible (y en particular en cualquier solución óptima), al menos $\bar{p}$ unidades del trabajo j' son procesadas dentro del intervalo $[r_j - \delta_j + p, d_j + \delta_j - p)$, es decir, para cualquier trabajo de J' que sea procesado parcial o completamente dentro del intervalo $[r_j, d_j)$ por el Algoritmo 2, toda solución óptima procesa al menos la misma fracción de ese trabajo dentro del intervalo $[r_j - \delta_j + p, d_j + \delta_j - p)$.

Por tanto, tenemos que

$$z^*_{\text{máx}} \geq \frac{\int_{r_j - \delta_j + p}^{d_j + \delta_j - p} h(\mathcal{J}'_{\text{Opt}}, t)dt}{3\delta_j - 2p} \geq \frac{\omega_j}{3\delta_j - 2p}$$

Recordemos que $\delta_j = N(2p-1) + \tilde{p}$, para $N \geq 1$ y algún $\tilde{p} \in [2p-2]$.

Si $\tilde{p} \leq p$, se sigue que $\delta_j < (2N+1)p$, obteniendo:

$$\frac{\omega_j}{3\delta_j - 2p} > \frac{N(z_{\text{máx}} - 1)}{3(2N+1) - 2} \geq \frac{z_{\text{máx}} - 1}{7}$$

Si $\tilde{p} = \hat{p} + p$, con $\hat{p} \in \{1, 2, \cdots, p-2\}$ y $\hat{p} + 1 \geq p$, se tiene:

$$\frac{\omega_j}{3\delta_j - 2p} \geq \frac{(N+1)(z_{\text{máx}} - 1)p}{3(N(2p-1) + p + \hat{p}) - 2p}$$

Notar que:

$$\begin{aligned}
\frac{(N+1)p}{3[N(2p-1) + p + \hat{p}] - 2p} > \frac{1}{7} &\iff 7(N+1)p > 3N(2p-1) + p + 3\hat{p} \\
&\iff N(7p - 3(2p-1)) + 7p - p - 3\hat{p} > 0 \\
&\iff N(p+3) + 6p - 3\hat{p} > 0
\end{aligned}$$

Como $\hat{p} \leq p-2$, se tiene que

$$6p - 3\hat{p} \geq 3p + 6 > 0.$$

Por tanto,

$$\frac{\omega_j}{3\delta_j - 2p} \geq \frac{(N+1)(z_{\text{máx}}-1)p}{3(N(2p-1)+p+\hat{p})-2p} > \frac{z_{\text{máx}}-1}{7}.$$

Finalmente, si $\tilde{p} = \hat{p} + p$ y $\hat{p} + 1 < p$, se tiene

$$\frac{\omega_j}{3\delta_j - 2p} \geq \frac{(z_{\text{máx}}-1)(Np+\hat{p}+1)}{3(N(2p-1)+p+\hat{p})-2p}$$

Observar que:

$$\begin{aligned}
\frac{Np+\hat{p}+1}{3(N(2p-1)+p+\hat{p})-2p} > \frac{1}{7} &\iff 7Np + 7\hat{p} + 7 > 6Np - 3N + p + 3\hat{p} \\
&\iff Np + 4\hat{p} + 7 + 3N - p > 0 \\
&\iff p(N-1) + 4\hat{p} + 7 + 3N > 0
\end{aligned}$$

Como $N \geq 1$, entonces $p(N-1) + 4\hat{p} + 7 + 3N > 0$, de donde concluimos que

$$\frac{\omega_j}{3\delta_j - 2p} > \frac{z_{\text{máx}}-1}{7}.$$

Por lo tanto, en cualquier caso, siempre se tiene que

$$z^*_{\text{máx}} > \frac{z_{\text{máx}}-1}{7} \quad \iff \quad 7z^*_{\text{máx}} > z_{\text{máx}} - 1.$$

Como $z^*_{\text{máx}}$ y $z_{\text{máx}}$ son valores enteros, se obtiene que:

$$7\text{costo}(\mathcal{Z}_{\text{Opt}}) \geq 7z^*_{\text{máx}} \geq z_{\text{máx}} \geq \frac{1}{T}\text{costo}(\mathcal{Z}_{\text{GBF}})$$

Equivalentemente,

$$\text{GBF}(\mathcal{I}) \leq 7T\text{Opt}(\mathcal{I}).$$

□

A continuación se presenta un ejemplo donde la solución encontrada por el Algoritmo 2 difiere del valor de la solución óptima por un factor de 5. Dicho resultado nos muestra que el factor de aproximabilidad esta acotado inferiormente por 5.

Ejemplo 3.4.1. *Consideremos la instancia* $\mathcal{I}_6 = (J, P, L_P, M, M_P)$*, donde* $J = \{1, 2, \cdots, 9\}$*,* $P = \{1, 2\}$*,* $L_P = (100, 100)$*,* $M = \{1, 2, 3\}$ *y* $M_P = (3, 3)$*. En la Tabla 3.5 se muestran los detalles de los trabajos de* J*. Además, definimos* $t_1 = 0$ *y* $t_2 = 100$*.*

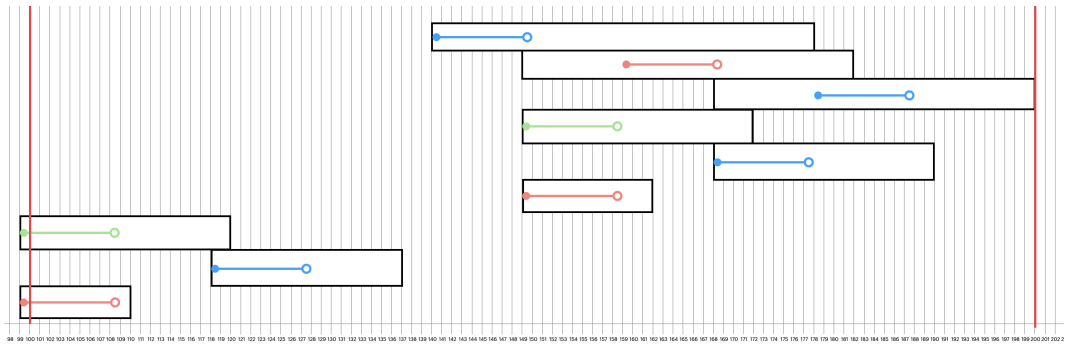
Trabajo	r_j	d_j	p_j
1	99	110	10
2	149	162	10
3	118	137	10
4	99	120	10
5	168	190	10
6	149	172	10
7	149	180	10
8	168	200	10
9	140	178	10

Tabla 3.5: Trabajos ordenados de la instancia $\mathcal{I}_6$.

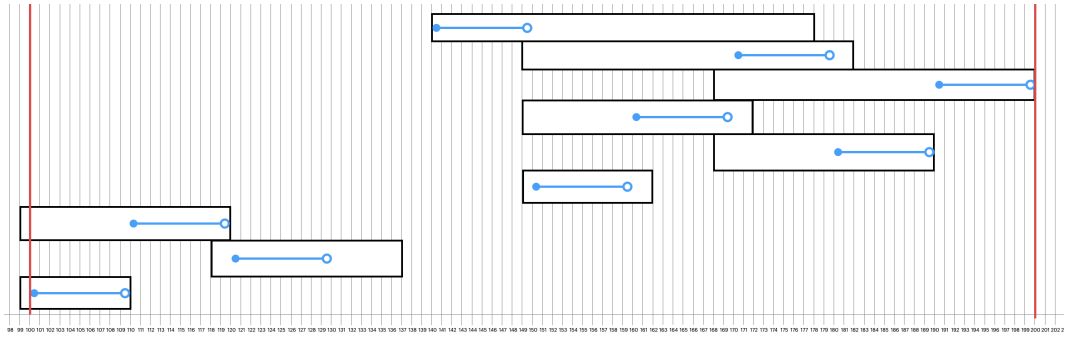
Si resolvemos la instancia $\mathcal{I}_6$ usando el Algoritmo 2 sabemos que se usan 5 periodos-máquina, la solución la podemos observar en la Figura 3.5a, el intervalo seleccionado para cada trabajo se encuentra coloreado con el color de la máquina que lo procesa. El color azul representa la máquina 1, el color verde representa la máquina 2 y el color rojo representa la máquina 3. Por otro lado, se pueden procesar todos los trabajos en 1 periodo-máquina como se puede observar en la Figura 3.5b.

Así, podemos escribir $\mathrm{Sol}_{GBF}(\mathcal{I}_6) = (\mathcal{J}, \mathcal{Z})$*, con* $\mathcal{J} = ([99, 109), [149, 159), [118, 128), [99, 109), [168, 178), [149, 159), [159, 169), [178, 188), [140, 150))$ *y* $\mathcal{Z} = (z_1 = 2, z_2 = 3)$*. Adicionalmente tenemos que* $z_1^* = 0$ *y* $z_2^* = 1$ *son el número óptimo de máquinas encendidas en cada periodo, respectivamente.*

Por lo tanto, $Opt(\mathcal{I}_6) = 1$ *y* $GBF(\mathcal{I}_6) = 5$*, es decir, se tiene que* 5 *es una cota inferior para el factor de aproximación del Algoritmo 2 para el problema SRDM-TD.*



(a) Solución de la instancia $\mathcal{I}_5$ encontrada por el Algoritmo 2.



(b) Solución óptima de la instancia $\mathcal{I}_5$.

Figura 3.5: Gráficos para la instancia $\mathcal{I}_6$ descrita en la Tabla 3.5.

3.4.2. Algoritmo basado en programación lineal

Recordemos que en [2] se demuestra que es posible asumir que los instantes de tiempo en los que inicia o termina el procesamiento de cualquier trabajo pertenecen a un conjunto finito de valores, para un problema similar al SRDM-D. Basados en las mismas ideas, en esta sección demostraremos un lema análago para el problema SRDM-TD.

Definición 3.4.1 (Solución eficiente)**.** Considerar una instancia $\mathcal{I} = (J, P, L_P, M, M_P)$ para el problema SRDM-TD. Notar que en este caso se tiene que $L_q = L$ para cada $q \in P$ y además: r_j, d_j y p corresponden al tiempo de llegada, plazo máximo de ejecución y duración (tiempo de procesamiento) de cada trabajo $j \in J$.

Una solución factible para la instancia $\mathcal{I}$ se dice eficiente si no es posible construir otra solución del mismo costo y cuyo vector de los tiempos de procesamiento de los trabajos sea lexicográficamente menor, es decir, la solución $\text{Sol}(\mathcal{I}) = (\mathcal{J}, \mathcal{Z})$ es eficiente

si, para toda solución factible $(\hat{\mathcal{J}}, \hat{\mathcal{Z}})$ tal que

$$\text{costo}(\mathcal{Z}) = \text{costo}(\hat{\mathcal{Z}}),$$

se cumple que:

$$\tau_j \leq \hat{\tau}_j, \qquad \forall j \in J.$$

Definición 3.4.2 (Solución óptima eficiente)**.** Una solución del problema SRDM-TD se dice óptima eficiente si es una solución óptima y además es una solución eficiente.

Lema 3.4.1. *En toda solución óptima eficiente del problema SRDM-TD, los tiempos de inicio y finalización de procesamiento de cada trabajo pertenecen al conjunto* $\{r_j + \eta p \colon j \in J, \eta \in [n]\} \cup \{t_q + \eta p \colon q \in P, \eta \in [n]\}$.

Demostración. Sea $\mathcal{I} = (J, P, L_P, M, M_P)$ una instancia del problema SRDM-TD y sea $j \in J$. Supongamos que en una solución óptima eficiente de $\mathcal{I}$, $\text{Sol}(\mathcal{I}) = (\mathcal{J}, \mathcal{Z})$, el trabajo j es asignado a la máquina $k \in M$ para empezar en el instante de tiempo τ. Adicionalmente, consideremos la sucesión $j_1, j_2, \cdots, j_\eta$ de todos los trabajos que son ejecutados en la máquina k de manera ininterrumpida antes de ejecutar el trabajo j. Notar que esta sucesión de trabajos puede ser vacía, en cuyo caso asumimos que $\eta = 0$. Si τ_{j_1} es el tiempo de inicio de procesamiento del trabajo j_1, tenemos que

$$\tau = \tau_{j_1} + \eta p,$$

es el tiempo de inicio de ejecución del trabajo j y $\tau + p = \tau_{j_1} + (\eta + 1)p$ es el tiempo de finalización del trabajo j. Además, sabemos que $\eta \leq n - 1$, de donde $(\eta + 1) \in [n]$.

Si el trabajo j_1 inicia su procesamiento al inicio de su ventana de tiempo, es decir, inicia su ejecución en el instante r_{j_1}. Entonces, se sigue que

$$\tau = r_{j_1} + \eta p,$$

con $j_1 \in J$ y $\eta \in [n]$. Por otra parte, si el trabajo j_1 inicia después del inicio de su ventana de tiempo, pero al inicio de algún periodo, digamos que al inicio del periodo $q \in P$, entonces

$$\tau = t_q + \eta p,$$

con $q \in P$ y $\eta \in [n]$. Por último, si el trabajo j_1 inicia después del inicio de su ventana de tiempo y en un tiempo distinto al inicio de cualquier periodo, es posible encontrar

un $\epsilon > 0$ suficientemente pequeño tal que al mover el inicio de ejecución del trabajo j_1 al instante de tiempo $\tau - \epsilon$ obtenemos una nueva solución factible de igual valor que la solución $\text{Sol}(\mathcal{I})$, lo que significa que $\text{Sol}(\mathcal{I})$ no es eficiente. Esto contradice con nuestro supuesto inicial de que la solución $\text{Sol}(\mathcal{I})$ es eficiente.

Por lo tanto, tenemos que en cualquier caso los instantes de tiempo τ y $\tau + p$ pertenecen al conjunto $\{r_j + \eta p \colon j \in J, \eta \in [n]\} \cup \{t_q + \eta p \colon q \in P, \eta \in [n]\}$. □

Del Lema 3.4.1, se puede observar que el conjunto de intervalos:

$$\{[r_j + \eta p, r_j + (\eta + 1)p) : j \in J, \eta \in \mathbb{Z}_+\} \cup \{[t_q + \eta p, t_q + (\eta + 1)p) : q \in P, \eta \in \mathbb{Z}_+\}$$

contiene los intervalos de procesamiento de todos los trabajos para cualquier solución óptima eficiente de una instancia del problema SRDM-TD.

En [20] se demuestra que existe una calendarización óptima de los trabajos para el problema SRDM-D de tal forma que todos los trabajos se ejecutan dentro el horizonte de tiempo $[r_{\text{mín}}, r_{\text{máx}} + 2np)$.

Dicho resultado se puede extender al problema SRDM-TD. Para ello, se plantea la siguiente proposición.

Proposición 3.4.3. *Existe una solución óptima del problema SRDM-TD, en la que todos los trabajos son ejecutados dentro del horizonte de tiempo:* $[r_{\text{mín}}, r_{\text{máx}} + 2np)$.

Demostración. Notar que en toda solución factible, los trabajos son procesados dentro del intervalo de tiempo $[r_{\text{mín}}, +\infty)$. Demostraremos a continuación que existe al menos una solución óptima en la que el procesamiento del último trabajo finaliza antes del tiempo $r_{\text{máx}} + 2np$.

De manera similar a la demostración del Lema 3.4.1, considerar una solución óptima eficiente y sea $j^* \in J$ un trabajo con el mayor tiempo de inicio de procesamiento dentro de esta solución, es decir,

$$\tau_{j^*} \geq \tau_j, \qquad \forall j \in J.$$

Suponer que este trabajo es procesado en una máquina $k \in M$ y considerar la sucesión

$$j_1, j_2, \cdots, j_\eta$$

de todos los trabajos que son procesados por k inmediatamente antes que j^*. Notar que esta sucesión puede ser vacía, en cuyo caso $\eta = 0$.

De la demostración del Lema 3.4.1, conocemos que $\tau_{j_1} = r_{\hat{j}}$, para algún $\hat{j} \in J$, o $\tau_{j_1} = t_{\hat{q}}$, para algún $\hat{q} \in P$.

En el primer caso, si $\tau_{j_1} = r_{\hat{j}}$, para algún $\hat{j} \in J$, se sigue que el tiempo de finalización del último trabajo es:

$$\begin{aligned}
\tau_{j^*} + p &= \tau_{j_1} + \eta p + p \\
&= r_{\hat{j}} + (\eta + 1)p \\
&\leq r_{\text{máx}} + (\eta + 1)p \\
&\leq r_{\text{máx}} + (n + 1)p \\
&\leq r_{\text{máx}} + 2np.
\end{aligned}$$

En el segundo caso, $\tau_{j_1} = t_{\hat{q}}$, para algún $\hat{q} \in P$, como la solución es eficiente, la máquina k debe estar sin uso en el periodo $\hat{q} - 1$, pues de lo contrario es posible adelantar el inicio del procesamiento del trabajo j_1 sin incrementar el costo de la solución. Notar además que debe tenerse que $r_{\text{máx}} > t_{\hat{q}-1}$, ya que de lo contrario es posible adelantar el procesamiento de todos los trabajos $j_1, j_2, \cdots, j_\eta, j^*$ en L unidades de tiempo sin incrementar el costo de la solución. De aquí se concluye que

$$t_{\hat{q}} = t_{\hat{q}-1} + L < r_{\text{máx}} + L$$

y por tanto el tiempo de finalización del último trabajo satisface

$$\begin{aligned}
\tau_{j^*} + p &= t_{\hat{q}} + (\eta + 1)p \\
&< r_{\text{máx}} + L + (\eta + 1)p \\
&\leq r_{\text{máx}} + L + np.
\end{aligned}$$

Si $L \leq np$, se sigue que

$$\tau_{j^*} + p \leq r_{\text{máx}} + 2np.$$

Caso contrario, si $L > np$, notar que debe tenerse que

$$t_{\hat{q}} - r_{\text{máx}} < (\eta + 1)p,$$

pues de lo contrario es posible adelantar el inicio de todos los trabajos $j_1, j_2, \cdots, j^*$ de tal forma que

$$\tau_{j_1} = r_{\text{máx}}$$

sin incrementar el costo de la solución. Por lo tanto, para el tiempo de finalización de j^* se cumple que

$$\begin{aligned}\tau_{j^*} + p &= t_{\hat{q}} + (\eta+1)p \\ &< r_{\text{máx}} + 2(\eta+1)p \\ &\leq r_{\text{máx}} + 2np\end{aligned}$$

lo que prueba que en cualquier caso, todos los trabajos son procesados antes de $r_{\text{máx}} + 2np$ unidades de tiempo. □

Corolario 3.4.3.1. *Existe una solución óptima del problema SRDM-TD, en la que todos los trabajos son ejecutados dentro del horizonte de tiempo:* $[r_{\text{mín}}, \text{mín}\{r_{\text{máx}} + 2np, t_T + L\})$.

Presentamos a continuación un modelo de programación lineal entera para el problema SRDM-TD, con el objetivo de proponer una modificación del Algoritmo 3. Notar que en nuestro caso la duración de cada periodo es constante, es decir, $L_q = L$, para todo $q \in P$. Adicionalmente, la función objetivo que se quiere minimizar es el número de periodos-máquina, es decir, se quiere minimizar $\sum_{q \in P} z_q$.

Del Corolario 3.4.3.1 junto con el Lema 3.4.1, se puede concluir que existe una solución óptima del problema SRDM-TD tal que los intervalos de procesamiento de los trabajos pertenecen al conjunto $A = A_1 \cup A_2$, donde

$$A_1 = \left\{ [r_j + \eta p, r_j + (\eta+1)p) : \begin{array}{l} j \in J, \eta \in \mathbb{Z}, r_j + \eta p \geq r_{\text{mín}}, \\ r_j + (\eta+1)p \leq \text{mín}\{r_{\text{máx}} + 2np, t_T + L\} \end{array} \right\}, \text{ y}$$

$$A_2 = \left\{ [t_q + \eta p, t_q + (\eta+1)p) : \begin{array}{l} q \in P, \eta \in \mathbb{Z}, t_q + \eta p \geq r_{\text{mín}}, \\ t_q + (\eta+1)p \leq \text{mín}\{r_{\text{máx}} + 2np, t_T + L\} \end{array} \right\}.$$

Notemos que el conjunto A es el mismo descrito en la Sección 2.3, tomando $L_q = L$ para cada $q \in P$. Al igual que en esa sección, asumimos que el conjunto A está formado por los intervalos $I_1, \cdots, I_{|A|}$, los cuales se encuentran ordenados de forma ascendente por su extremo izquierdo. Observar además que el conjunto A conserva la misma propiedad de regularidad expuesta en el Lema 2.3.2.

Para formular el SRDM-TD como un programa lineal entero, se emplean las si-

guientes variables de decisión:

$$x_{j,i} := \begin{cases} 1, & \text{si el trabajo } j \text{ es ejecutado en alguna máquina en el intervalo } I_i \in A, \\ 0, & \text{caso contrario,} \end{cases}, \text{ y}$$

z_q : cantidad de máquinas requeridas en la solución en el periodo $q \in P$.

De esta manera, podemos formular el siguiente modelo lineal entero para el SRDM-TD (MS-SRDM-TD):

$$\text{mín} \sum_{q \in P} z_q \tag{3.4.1}$$

s.a.r.

$$\sum_{i \in \tilde{H}} x_{j,i} = 1, \quad \forall j \in J, \tag{3.4.2}$$

$$\sum_{\eta \in [\kappa]_+} \sum_{j \in J} x_{j,i+\eta} \leq z_q, \quad \forall i \in \tilde{H}_\kappa, \kappa \in \Phi, q \in P, \bigcap_{\eta \in [\kappa]_+} I_{i+\eta} \cap [t_q, t_q + L) \neq \emptyset, \tag{3.4.3}$$

$$x_{j,i} = 0, \quad \text{si } R(I_i) < r_j, \forall i \in \tilde{H}, j \in J, \tag{3.4.4}$$

$$x_{j,i} = 0, \quad \text{si } d_j < D(I_i), \forall i \in \tilde{H}, j \in J, \tag{3.4.5}$$

$$z_q \leq m_q, \quad \forall q \in P, \tag{3.4.6}$$

$$x_{j,i} \in \{0, 1\}, \quad \forall i \in \tilde{H}, j \in J,$$

$$z_q \in \mathbb{Z}_+, \quad \forall q \in P,$$

donde

$$\tilde{H} = \{1, 2, \cdots, |A|\},$$

$$\tilde{H}_\kappa := \{0, 1, \cdots, |A| - \kappa\},$$

$$\phi = \text{máx}\{\eta \colon I_{i+1} \cap \cdots \cap I_{i+\eta} \neq \emptyset, \forall i \in \tilde{H}_\eta\},$$

$$\Phi = \{1, \cdots, \phi\},$$

$$[\kappa]_+ = \{1, \cdots, \kappa\}.$$

La función objetivo (3.4.1) mide la cantidad total de periodos-máquina requeridos para procesar todos los trabajos. Las restricciones (3.4.2), (3.4.4) y (3.4.5) establecen que cada trabajo $j \in J$ debe ser asignado a un intervalo de A, el cual debe estar enteramente contenido en $[r_j, d_j)$. Las restricciones (3.4.3) garantizan que el número de trabajos procesados en cualquier instante de tiempo dentro del periodo q no excedan el número

de máquinas encendidas en dicho periodo. Las restricciones (3.4.6) nos garantizan que el número de máquinas encendidas en cada periodo no exceden el número de máquinas disponibles.

Si multiplicamos por p a ambos lados de las desigualdades (3.4.2) hasta (3.4.5), y realizamos el cambio de variable:

$$y_{j,i} = px_{j,i} = \begin{cases} p, & \text{si el trabajo } j \text{ es ejecutado en alguna máquina en el intervalo } I_i \in A, \\ 0, & \text{caso contrario,} \end{cases}$$

el modelo MS-SRDM-TD se puede reescribir como (PES-SRDM-TD):

$$\text{mín} \sum_{q \in P} z_q$$

s.a.r.

$$\sum_{i \in \tilde{H}} y_{j,i} = p, \quad \forall j \in J, \tag{3.4.7}$$

$$\sum_{\eta \in [\kappa]_+} \sum_{j \in J} y_{j,i+\eta} \leq p z_q, \quad \forall i \in \tilde{H}_\kappa, \kappa \in \Phi, q \in P, \bigcap_{\eta \in [\kappa]_+} I_{i+\eta} \cap [t_q, t_q + L) \neq \emptyset, \tag{3.4.8}$$

$$y_{j,i} = 0, \quad \text{si } R(I_i) < r_j, \forall i \in \tilde{H}, j \in J, \tag{3.4.9}$$

$$y_{j,i} = 0, \quad \text{si } d_j < D(I_i), \forall i \in \tilde{H}, j \in J, \tag{3.4.10}$$

$$z_q \leq m_q, \quad \forall q \in P, \tag{3.4.11}$$

$$y_{j,i} \in \{0, p\}, \quad \forall i \in \tilde{H}, j \in J,$$

$$z_q \in \mathbb{Z}_+, \quad \forall q \in P.$$

La relajación lineal del problema (LPS-SRDM-TD) es:

$$\text{mín} \sum_{q \in P} z_q$$

s.a.r.

$$\sum_{i \in \tilde{H}} y_{j,i} = p, \quad \forall j \in J, \tag{3.4.12}$$

$$\sum_{\eta \in [\kappa]_+} \sum_{j \in J} y_{j,i+\eta} \leq p z_q, \quad \forall i \in \tilde{H}_\kappa, \kappa \in \Phi, q \in P, \bigcap_{\eta \in [\kappa]_+} I_{i+\eta} \cap [t_q, t_q + L) \neq \emptyset, \tag{3.4.13}$$

$$y_{j,i} = 0, \quad \text{si } R(I_i) < r_j, \forall i \in \tilde{H}, j \in J, \tag{3.4.14}$$

$$y_{j,i} = 0, \quad \text{si } d_j < D(I_i), \forall i \in \tilde{H}, j \in J, \tag{3.4.15}$$

$$z_q \leq m_q, \quad \forall q \in P, \tag{3.4.16}$$

$$y_{j,i} \geq 0, \quad \forall i \in \tilde{H}, j \in J,$$
$$z_q \geq 0, \quad \forall q \in P.$$

Por otra parte, veamos que el número de variables de decisión en el programa lineal LPS-SRDM-TD es igual a $n|A| + T$. Además, sabemos que

$$\begin{aligned} |A| &\leq (n+T)\frac{1}{p}\left[\text{mín}\{r_{\text{máx}} + 2np, t_T + L_T\} - \text{máx}\{r_{\text{mín}}, t_1\}\right] \\ &\leq (n+T)\left[2n + \frac{r_{\text{máx}} - r_{\text{mín}}}{p}\right] \\ &= (n+T)(2n+c), \end{aligned}$$

con $c = \frac{r_{\text{máx}} - r_{\text{mín}}}{p} \leq r_{\text{máx}} < t_T + L_T$. Por tanto,se sigue que:

$$\begin{aligned} n|A| + T &\leq n(n+T)(2n+c) + T. \\ &= 2n^3 + 2n^2T + cn^2 + cnT + T. \end{aligned}$$

Es decir, el número de variables de decisión del modelo LPS-SRDM-TD es de orden $O(n^3 + n^2T)$.

Por otra parte, dado que $1 \leq \phi \leq |A|$, podemos verificar que el número de restricciones del programa lineal LPS-SRDM-TD está acotado por:

$$\begin{aligned} & n + \sum_{q \in P}\sum_{\kappa \in \Phi}(|A| - \kappa + 1) + 3n|A| + 2T \\ &= n + T\left[\phi(|A|+1) - \frac{1}{2}\phi(\phi+1)\right] + 3n|A| + 2T \\ &\leq n + T|A|^2 + T|A| + 3n|A| + 2T \\ &\leq n + T(n+T)^2(2n+c)^2 + T(n+T)(2n+c) + 3n(2n+c)(n+T) + 2T \\ &= n + (c^2n^2T + 2c^2nT^2 + c^2T^3 + 4cn^3T + 8cn^2T^2 + 4cnT^3 + 4n^4T + 8n^3T^2 + 4n^2T^3) + \\ &\quad + (cnT + cT^2 + 2n^2T + 2nT^2) + (3cn^2 + 3cnT + 6n^3 + 6n^2T) + 2T \\ &= n + c^2n^2T + 2c^2nT^2 + c^2T^3 + 4cn^3T + 8cn^2T^2 + 3cn^2 + 4cnT^3 + 4cnT + \\ &\quad + cT^2 + 4n^4T + 8n^3T^2 + 6n^3 + 4n^2T^3 + 8n^2T + 2nT^2 + 2T. \end{aligned}$$

De donde, el número de restricciones es de orden $O(n^4T + n^3T^2 + n^2T^3) = O(n^2T(n+T)^2)$.

Por lo tanto, debido a la polinomialidad de la Programación Lineal, un algoritmo basado en la solución de LPS-SRDM-TD es un algoritmo polinomial para el SRDM-TD.

Formularemos a continuación un algoritmo con estas características, el mismo que puede considerarse como una variante del Algoritmo 3. La modificación consiste en cambiar el modelo de programación lineal empleado en el segundo paso. Así, tenemos el siguiente algoritmo:

Algoritmo 5: Algoritmo basado en programación lineal para el problema SRDM-TD.

Entrada: Instancia $\mathcal{I} = (J, P, L_P, M, M_P)$.

1. Ordenar los trabajos de J en forma ascendente de acuerdo al valor del plazo ejecución d_j, con $j \in J$.
2. Resolver el modelo LPS-SRDM-TD, sea $(y^*, z^*) \in \mathbb{R}^{n \times |A|} \times \mathbb{R}^T$ la solución del mismo.
3. Definir $v(I_i) = \sum_{\kappa \in \{1,2,\cdots,i\}} \sum_{j \in J} y^*_{j,\kappa}, \quad , \forall i \in \tilde{H}$.
4. Para cada $i \in \tilde{H}$ definir $\mu_i = \text{mín}\{m_q \colon q \in P, I_i \cap [t_q, t_q + L) \neq \emptyset\}$ y fijar $\mu_0 = 0$.
5. **para** $i \in \tilde{H}$ **hacer**
   - **para** $i' \in \{1, 2, \cdots, \mu_i\}$ **hacer**
     - Definir $\varphi = \mu_0 + \cdots + \mu_{i-1} + i'$.
     - Definir $v_\varphi = v(I_i)$.
   - **fin**

   fin
6. Definir $\theta(0) = 0$.
7. **para** $\lambda \in \{1, 2, \cdots, |J|\}$ **hacer**
   - Definir $\theta(\lambda) = \text{mín}\{\varphi \colon \theta(\lambda - 1) < \varphi \leq \bar{\mu}, v_\varphi > (\lambda - 1) \cdot p\}$, donde $\bar{\mu} = \mu_1 + \cdots + \mu_{|A|}$.

   fin
8. **para** $\lambda \in \{1, 2, \cdots, |J|\}$ **hacer**
   - Definir $I^*_\lambda = I_\eta$, donde $\eta \in \tilde{H}$ está dado por $v(I_\eta) = v_{\theta(\lambda)}$.

   fin
9. **para** $\lambda \in \{1, 2, \cdots, |J|\}$ **hacer**
   - De los trabajos compatibles con el intervalo I^*_λ, que aún no han sido procesados, asignar a I^*_λ el trabajo con el menor plazo de ejecución.

   fin

Teorema 3.4.1. *El Algoritmo 5 permite construir una solución factible del modelo PES-SRDM-TD.*

Demostración. Observar que el Algoritmo 5, en el paso 8, selecciona n intervalos

$$I^*_\lambda = I_{\sigma(\lambda)} \in A,$$

con $\lambda \in \{1, 2, \cdots, |J|\}$.

Un mismo intervalo puede seleccionarse varias veces, es decir, se puede tener $\sigma(\lambda_1) = \sigma(\lambda_2)$ para $\lambda_1, \lambda_2 \in \{1, 2, \cdots |J|\}$ distintos entre sí. Posteriormente, en el paso 9 del algoritmo, se consideran estos intervalos en orden creciente del índice λ y se asigna al intervalo I^*_λ aquel trabajo j^* que sea compatible con el intervalo (es decir, que satisfaga $I^*_\lambda \subseteq [r_{j^*}, d_{j^*})$), que tenga el menor plazo de ejecución d_{j^*}, y que no haya sido previamente asignado a otro intervalo. Notar que, por construcción, cada trabajo es asignado máximo a un intervalo y cada intervalo recibe máximo un trabajo asignado. De hecho, demostraremos a continuación que la asignación realizada por el algoritmo está asociada a una solución factible para el programa lineal entero PES-SRDM-TD. Denotaremos por $\pi(j) \in \{0, 1, \cdots, |A|\}$ al índice del intervalo al cual es asignado el trabajo $j \in J$ en el paso 9 del algoritmo, donde $\pi(j) = 0$ indica que el trabajo no fue asignado a ningún intervalo.

Sea $\mathcal{J} = (I^*_1, I^*_2, \cdots, I^*_n)$ el vector de intervalos seleccionados por el Algoritmo 3. Para cada $q \in P$, definimos $\hat{z}_q = \max\limits_{t \in [t_q, t_q + L)} h(\mathcal{J}, t)$ como el número de máquinas necesarias para procesar los trabajos asignados a estos intervalos en el periodo q, donde h es la medida definida en la Sección 2.1. Adicionalmente, definimos el vector $\hat{y} \in \mathbb{R}^{n \times |A|}$ por:

$$\hat{y}_{j,i} = \begin{cases} p, & \text{si el trabajo } j \in J \text{ es asignado al intervalo } I_i \text{ por el Algoritmo 5,} \\ & \text{es decir, si } \pi(j) = i, \\ 0, & \text{caso contrario.} \end{cases}$$

Demostraremos a continuación que $(\hat{y}, \hat{z}) \in \mathbb{R}^{n \times |A|} \times \mathbb{Z}^T$ es una solución factible del modelo lineal entero. De las definiciones de $\hat{y}$ y $\hat{z}$ sabemos que $\hat{y} \in \{0, p\}^{n \times |A|}$ y $\hat{z} \in \mathbb{Z}^T_{\geq 0}$. Por lo tanto, para probar que $(\hat{y}, \hat{z})$ es una solución factible del problema PES-SRDM-TD nos resta probar que $\hat{y}$ satisface las familias de restricciones (3.4.7) - (3.4.11).

Primero, para que $\hat{y}$ satisfaga la familia de restricciones (3.4.7), debemos probar que

$$\sum_{i \in \tilde{H}} \hat{y}_{j,i} = p, \quad \forall j \in J.$$

Como cada trabajo es asignado a máximo una máquina, conocemos que

$$\sum_{i \in \tilde{H}} \hat{y}_{j,i} \in \{0, p\}.$$

Por lo tanto, $\hat{y}$ satisface la familia de restricciones (3.4.7) si y sólo si

$$\pi(j) \neq 0, \qquad \forall j \in J.$$

Supongamos que existe $j' \in J$ tal que $\pi(j') = 0$. Sea $(y^*, z^*) \in \mathbb{R}^{n \times |A|} \times \mathbb{R}^T$ la solución obtenida al resolver el programa lineal en el paso 2 del Algoritmo 5. Para cada $j \in J$, se define el intervalo restringido de j, $[r_j^*, d_j^*) \subseteq [r_j, d_j)$, donde $r_j^* = \text{mín}\left\{R(I_i) \colon i \in \tilde{H}, y_{j,i}^* \neq 0\right\}$ y $d_j^* = \text{máx}\left\{D(I_i) \colon i \in \tilde{H}, y_{j,i}^* \neq 0\right\}$. Notar que debido a las restricciones (3.4.14) y (3.4.15) se tiene que $r_j \leq r_j^* < d_j^* \leq d_j$.

Sea Υ_1 el conjunto de intervalos de A que están contenidos en el intervalo restringido asociado a j', es decir:

$$\Upsilon_1 = \{I_i \in A \colon I_i \subseteq [r_{j'}^*, d_{j'}^*)\}.$$

Notar que $\Upsilon_1 \neq \emptyset$, pues de lo contrario y^* no satisface las restricciones (3.4.12).

Adicionalmente, si suponemos que

$$\Upsilon_1 = \{I_a, I_{a+1}, \cdots, I_b\}, \qquad \text{con } 1 \leq a \leq b \leq |A|,$$

entonces de la definición de intervalo restringido y de (3.4.12) se sigue que

$$\begin{aligned} v(I_b) - v(I_{a-1}) &= \sum_{i \in \{1,2,\cdots,b\}} \sum_{j \in J} y_{j,i}^* - \sum_{i \in \{1,2,\cdots,a-1\}} \sum_{j \in J} y_{j,i}^* \\ &= \sum_{i \in \{a,\cdots,b\}} \sum_{j \in J} y_{j,i}^* \\ &\geq \sum_{i \in \{a,\cdots,b\}} y_{j',i}^* \\ &= p. \end{aligned}$$

Por lo tanto, el conjunto Υ_1 contiene al menos uno de los intervalos seleccionados por el Algoritmo 5, en el paso 8. Por otra parte, recordemos que μ_i^* es el número de copias seleccionadas del intervalo $I_i \in A$ por el Algoritmo 5, es decir, $\mu_i^* =$

$|\{I^*_\lambda \colon \lambda \in J, \sigma(\lambda) = i\}|$. De donde se puede notar que a cada intervalo $I_i \in \Upsilon_1$ se le deben haber asignado μ^*_i trabajos $j \in J$, con $d_j \leq d_{j'}$, pues de lo contrario el trabajo j' sería asignado a una de las μ^*_i copias de este intervalo.

Consideremos el conjunto de trabajos que han sido asignados a intervalos de Υ_1:

$$J_1 = \{j \in J \colon I_{\pi(j)} \in \Upsilon_1\}.$$

De las observaciones anteriores, se sigue que $J_1 \neq \emptyset$. Sea j_1 un elemento de J_1 que satisface

$$r^*_{j_1} \leq r^*_j, \qquad \forall j \in J_1.$$

Si $r^*_{j_1} \geq r^*_{j'}$, definimos $\hat{j} := j'$. Caso contrario, si $r^*_{j_1} < r^*_{j'}$, consideremos el conjunto

$$\Upsilon_2 = \{I_i \in A \colon I_i \subseteq [r^*_{j_1}, d^*_{j'})\}.$$

Nuevamente, se cumple que cada intervalo $I_i \in A$, con $I_i \subseteq [r^*_{j_1}, r^*_{j'})$, debe tener asignados μ^*_i trabajos $j \in J$, con $d_j \leq d_{j_1}$, pues de lo contrario el trabajo j_1 habría sido asignado a uno de estos intervalos. Luego, cada intervalo $I_i \in \Upsilon_2$ tiene μ^*_i trabajos asignados, todos ellos con $d_j \leq d_{j'}$. Consideramos ahora el conjunto

$$J_2 = \{j \in J \colon I_{\pi(j)} \in \Upsilon_2\}$$

de todos los trabajos asignados a intervalos en Υ_2 y definamos j_2 como un elemento de J_2 que satisface:

$$r^*_{j_2} \leq r^*_j, \qquad \forall j \in J_2.$$

Si $r^*_{j_2} \geq r^*_{j_1}$, definimos $\hat{j} := j_1$. Caso contrario, si $r^*_{j_2} < r^*_{j_1}$, repetimos la construcción anterior. De esta manera, en un número finito de iteraciones (este número puede ser cero, si $r^*_{j_1} \geq r^*_{j'}$) construimos un conjunto de intervalos

$$\tilde{\Upsilon} = \{I_i \in A \colon I_i \subseteq [r^*_{\hat{j}}, d^*_{j'})\},$$

y un conjunto de trabajos

$$\tilde{J} = \{j \in J \colon I_{\pi(j)} \in \tilde{\Upsilon}\}$$

que satisfacen las siguientes propiedades:

- $r^*_{\hat{j}} = \text{mín}\{r^*_j \colon j \in \tilde{J} \cup \{j'\}\}$,

- $d^*_{j'} = \text{máx}\{d^*_j \colon j \in \tilde{J} \cup \{j'\}\}$,
- cada intervalo $I_i \in \tilde{\Upsilon}$ tiene asignados μ^*_i trabajos de $\tilde{J}$,
- para cada trabajo $j \in \tilde{J}$ se satisface $[r^*_j, d^*_j) \subseteq [r^*_j, d^*_{j'})$ y por tanto $\sum_{I_i \in \tilde{\Upsilon}} y^*_{j,i} = p$.

Sea $\hat{n} = |\tilde{J}|$. De la última propiedad listada arriba se sigue que

$$\begin{aligned}\sum_{I_i \in \tilde{\Upsilon}} \sum_{j \in J} y^*_{j,i} &\geq \sum_{I_i \in \tilde{\Upsilon}} \sum_{j \in \tilde{J}} y^*_{j,i} + \sum_{I_i \in \tilde{\Upsilon}} y^*_{j',i} \\ &= \hat{n}p + p \\ &= (\hat{n}+1)p. \end{aligned} \tag{3.4.17}$$

Por otra parte, supongamos que

$$\hat{\Upsilon} = \{I_{\sigma(c_1)}, I_{\sigma(c_2)}, \cdots, I_{\sigma(c_g)}\} \subseteq \tilde{\Upsilon},$$

es el conjunto de intervalos seleccionados por el Algoritmo 5 y que pertenecen al conjunto $\tilde{\Upsilon}$, con $c_1, c_2, \cdots, c_g \in \{1, 2, \cdots, |J|\}$ y tales que

$$c_1 \leq c_2 \leq \cdots \leq c_g.$$

Si $c_1 \geq 2$, por el Lema 2.3.3 aplicado al intervalo $I_{\sigma(c_1-1)}$ tenemos que

$$(\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_1-1)})p \geq v(I_{\sigma(c_1-1)}) > (\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_1-1)} - 1)p.$$

Luego, $v(I_{\sigma(c_1-1)}) = (\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_1-1)} - 1)p + \delta$, para algún $\delta \in (0, p]$. De manera similar, para el intervalo $I_{\sigma(c_g)}$ se satisface la desigualdad:

$$(\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)})p \geq v(I_{\sigma(c_g)}) > (\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)} - 1)p,$$

de donde $v(I_{\sigma(c_g)}) = (\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)} - 1)p + \epsilon$, para algún $\epsilon \in (0, p]$.

Por tanto, se sigue que

$$v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) = (\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_1+1)} + \cdots + \mu^*_{\sigma(c_g)})p + \epsilon - \delta,$$

con $\delta, \epsilon \in (0, p]$. Para todo $i \in \tilde{H} \setminus \{\sigma(c_1), \sigma(c_2), \cdots, \sigma(c_g)\}$ tal que $\sigma(c_1 - 1) + 1 \leq i \leq \sigma(c_g)$, sabemos que $\mu^*_i = 0$, de donde

$$v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) = (\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)})p + \epsilon - \delta = \hat{n}p + \epsilon - \delta,$$

con $\delta, \epsilon \in (0, p]$. La última igualdad se sigue del hecho de que cada intervalo $I_i \in \hat{\Upsilon}$ está asignado a un trabajo en $\tilde{J}$ y que ningún trabajo es asignado a más de un intervalo.

Además, de la definición de $v(I_i)$ en el paso 3 del algoritmo, se tiene que:

$$v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) = \sum_{j\in J} y^*_{j,\sigma(c_1-1)+1} + \cdots + \sum_{j\in J} y^*_{j,\sigma(c_g)-1} + \sum_{j\in J} y^*_{j,\sigma(c_g)} \geq \sum_{I_i\in\tilde{\Upsilon}} \sum_{j\in J} y^*_{j,i}.$$

Conjuntamente con (3.4.17), las dos últimas ecuaciones implican que:

$$(\hat{n}+1)p \leq \sum_{I_i\in\tilde{\Upsilon}} \sum_{j\in J} y^*_{j,i} \leq \hat{n}p + \epsilon - \delta \quad \Longleftrightarrow \quad p \leq \epsilon - \delta,$$

lo que contradice que $\delta, \epsilon \in (0, p]$. Luego, el supuesto inicial de que el trabajo j' no fue procesado en algún intervalo dentro de su intervalo restringido nos lleva hacia una contradicción en este caso.

En el caso restante, si $c_1 = 1$, tenemos que $v(I_{\sigma(c_g)}) = (\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)} - 1)p + \epsilon$, para algún $\epsilon \in (0, p]$. Además,

$$v(I_{\sigma(c_g)}) = \sum_{j\in J} y^*_{j,1} + \cdots + \sum_{j\in J} y^*_{j,\sigma(c_g)-1} + \sum_{j\in J} y^*_{j,\sigma(c_g)} \geq (\hat{n}+1)p,$$

y por otra parte $\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)} = \hat{n}$, de donde se obtiene que

$$(\hat{n}-1)p + \epsilon \geq (\hat{n}+1)p \qquad \Longleftrightarrow \qquad \epsilon \geq 2p,$$

lo cual nuevamente contradice que $\epsilon \in (0, p]$. Es decir, nuestro supuesto inicial de que el trabajo j' no fue procesado en algún intervalo dentro de su intervalo restringido no es cierto. En conclusión, todos los trabajos $j \in J$ son asignados a algún intervalo seleccionado por el Algoritmo 5.

Así, hemos probado que se satisfacen las restricciones (3.4.7). Pero como además, todos los trabajos son asignados a algún intervalo seleccionado por el Algoritmo 5 perteneciente al intervalo restringido de cada trabajo, sabemos que se satisfacen las restricciones (3.4.9) y (3.4.10).

Verifiquemos ahora que la solución $(\hat{y}, \hat{z})$ satisface las restricciones (3.4.8) y (3.4.11). Notar primero que la desigualdad:

$$\sum_{\eta\in\{1,2,\cdots,\kappa\}} \sum_{j\in J} \hat{y}_{j,i+\eta} \leq p\hat{z}_q,$$

se satisface claramente para todo $i \in [|A| - \kappa]$, $\kappa \in \{1, 2, \cdots, \phi\}$ y $q \in P$ tal que $\bigcap_{\eta\in\{1,2,\cdots,\kappa\}} I_{i+\eta} \cap [t_q, t_q + L) \neq \emptyset$, por la definición de $\hat{z}$.

Por tanto, nos resta verificar la familia de restricciones (3.4.11), es decir, debemos verificar que:

$$\hat{z}_q \leq m_q, \qquad \forall q \in P.$$

Para ello, sean $q \in P$, $\kappa \in \{1, 2, \cdots, \phi\}$, $i \in [|A| - \kappa]$ tales que

$$\bigcap_{\eta \in \{1,2,\cdots,\kappa\}} I_{i+\eta} \cap [t_q, t_q + L_q) \neq \emptyset.$$

Supongamos además que de los intervalos $I_{i+1}, \cdots, I_{i+\kappa}$ el Algoritmo 5 ha seleccionado los intervalos:

$$I_{\sigma(c_1)}, I_{\sigma(c_2)}, \cdots, I_{\sigma(c_g)},$$

donde $\sigma(c_1), \sigma(c_2), \cdots, \sigma(c_g) \in \{i+1, i+2, \cdots, i+\kappa\}$ y $c_1 \leq c_2 \leq \cdots c_g$. Como sabemos que μ_i^* es el número de copias seleccionadas del intervalo I_i, para todo $i \in A$, probar que $\hat{z}_q \leq m_q$ es equivalente a probar que

$$\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)} \leq m_q.$$

Si $c_1 > 1$, del Lema 2.3.3 sabemos que

$$(\mu_1^* + \mu_2^* + \cdots + \mu^*_{\sigma(c_1-1)})p \geq v(I_{\sigma(c_1-1)}) \quad \text{y} \quad (\mu_1^* + \mu_2^* + \cdots + \mu^*_{\sigma(c_g)} - 1)p < v(I_{\sigma(c_g)}).$$

Entonces,

$$\begin{aligned} v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) &> (\mu_1^* + \mu_2^* + \cdots + \mu^*_{\sigma(c_g)} - 1)p - (\mu_1^* + \mu_2^* + \cdots + \mu^*_{\sigma(c_1-1)})p \\ &= (\mu^*_{\sigma(c_1-1)+1} + \mu^*_{\sigma(c_1-1)+2} + \cdots + \mu^*_{\sigma(c_g)} - 1)p \\ &= (\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)} - 1)p. \end{aligned}$$

De la definición de $v(I_{\sigma(c_g)})$ y $v(I_{\sigma(c_1-1)})$ junto con la restricción (3.4.13) y (3.4.16), sabemos que

$$v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) = \sum_{j \in J} y^*_{j,\sigma(c_1-1)+1} + \sum_{j \in J} y^*_{j,\sigma(c_1-1)+2} + \cdots + \sum_{j \in J} y^*_{j,\sigma(c_g)} \leq m_q p.$$

Por tanto, se tiene que

$$(\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)} - 1)p < v(I_{\sigma(c_g)}) - v(I_{\sigma(c_1-1)}) \leq m_q p.$$

De donde

$$(\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)})p - p < m_q p \quad \Longleftrightarrow \quad (\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)})p < (m_q + 1)p,$$

y como todas las cantidades son enteras, se concluye que

$$\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)} \leq m_q.$$

Ahora, si $c_1 = 1$, del Lema 2.3.3 sabemos que

$$(\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)} - 1)p < v(I_{\sigma(c_g)}).$$

De la definición de $v(I_{\sigma(c_g)})$ junto con la restricción (3.4.13), sabemos que

$$v(I_{\sigma(c_g)}) = \sum_{j\in J} y^*_{j,1} + \sum_{j\in J} y^*_{j,2} + \cdots + \sum_{j\in J} y^*_{j,\sigma(c_g)} \leq m_q p.$$

Por tanto, se tiene que

$$(\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)} - 1)p < v(I_{\sigma(c_g)}) \leq m_q p.$$

De donde

$$(\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)})p - p < m_q p \quad \Longleftrightarrow \quad (\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)})p < (m_q + 1)p.$$

Es decir,

$$\mu^*_1 + \mu^*_2 + \cdots + \mu^*_{\sigma(c_g)} \leq m_q,$$

y en particular tenemos que

$$\mu^*_{\sigma(c_1)} + \mu^*_{\sigma(c_2)} + \cdots + \mu^*_{\sigma(c_g)} \leq m_q.$$

Así, hemos probado que $(\hat{y}, \hat{z})$ satisface las restricciones (3.4.11).

Por lo tanto, se sigue que $(\hat{y}, \hat{z})$ es una solución factible para el modelo PES-SRDM-TD. Consecuentemente, tomando $\hat{x}_{j,i} = \hat{y}_{j,i}/p$ para todo $j \in J$, $i \in \tilde{H}$, se tiene que $(\hat{x}, \hat{z})$ también es una solución factible del modelo MS-SRDM-TD. □

Proposición 3.4.4. *Para cualquier instancia $\mathcal{I}$ del SRDM-TD, la solución obtenida por el Algoritmo 5 difiere en un valor máximo de T de la solución óptima, es decir,*

$$Alg_{LP}(\mathcal{I}) \leq Opt(\mathcal{I}) + T.$$

Demostración. Dada una instancia $\mathcal{I}$ para el problema SRDM-TD. Sea costo$(\mathcal{Z}) = \sum_{q\in P} \hat{z}_q$ el valor de la función objetivo obtenido con la solución $\hat{y}$ construida en el Teorema 3.4.1 al resolver la instancia $\mathcal{I}$. Se quiere conocer cual es la diferencia entre costo$(\mathcal{Z})$

y $\text{Opt}(\mathcal{I})$. Sea además, (y^*, z^*) la solución óptima encontrada al resolver el modelo LPS-SRDM-TD.

Sea $i \in \tilde{H}_\kappa$, $\kappa \in \Phi$, $q \in P$ tales que

$$\bigcap_{\eta \in [\kappa]_+} I_{i+\eta} \cap [t_q, t_q + L) \neq \emptyset.$$

Definamos

$$\hat{v}(I_i) = \sum_{\kappa \in \{1,2,\cdots,i\}} \sum_{j \in J} \hat{y}_{j,\kappa}, \qquad \forall i \in \{1, 2, \cdots, |A|\}.$$

De donde se sigue que

$$\sum_{\eta \in [\kappa]_+} \sum_{j \in J} \hat{y}_{j,i+\eta} = \hat{v}(I_{i+\kappa}) - \hat{v}(I_i)$$

Pero además, del Lema 2.3.4 conocemos que $\hat{v}(I_{i+\kappa}) < p + v(I_{i+\kappa})$ y $\hat{v}(I_i) \geq v(I_i)$. Luego, se sigue que

$$\sum_{\eta \in [\kappa]_+} \sum_{j \in J} \hat{y}_{j,i+\eta} < v(I_{i+\kappa}) + p - v(I_i)$$

Por otra parte, de las restricciones (3.4.13) del modelo LPS-SRDM-TD, se obtiene que

$$v(I_{i+\kappa}) - v(I_i) = \sum_{\eta \in [\kappa]_+} \sum_{j \in J} y^*_{j,i+\eta} \leq z^*_q p$$

Combinando las dos desigualdades anteriores, se concluye que

$$\sum_{\eta \in [\kappa]_+} \sum_{j \in J} \hat{y}_{j,i+\eta} < z^*_q p + p = (z^*_q + 1)p$$

Por último, observar que todos los términos del doble sumatorio en el extremo izquierdo de la última desigualdad son múltiplos de p y por ende,

$$\sum_{\eta \in [\kappa]_+} \sum_{j \in J} \hat{y}_{j,i+\eta} \leq \lceil z^*_q \rceil p$$

Como $\hat{z}_q$ es el menor valor entero suficiente para garantizar el cumplimiento de la restricción (3.4.8), se sigue que $\hat{z}_q = \left\lceil z^*_q \right\rceil$ y por lo tanto se tiene que el valor de la función objetivo correspondiente a la solución del Algoritmo 5 es:

$$\text{Alg}_{\text{LP}}(\mathcal{I}) := \text{costo}(\mathcal{Z}) = \sum_{q \in P} \lceil z^*_q \rceil \leq \sum_{q \in P} (z^*_q + 1) \leq \text{Opt}(\mathcal{I}) + T,$$

donde la última desigualdad se sigue del hecho de que LPS-SRDM-TD es una relajación lineal del modelo MS-SRDM-TD. □

Corolario 3.4.4.1. *El Algoritmo 5 tiene un factor de aproximación de 2 para el SRDM-TD.*

Demostración. De la Proposición 3.4.4 se sigue que

$$\mathrm{Alg}_{\mathrm{LP}}(\mathcal{I}) \leq \mathrm{Opt}(\mathcal{I}) + T \qquad \Longleftrightarrow \qquad \frac{\mathrm{Alg}_{\mathrm{LP}}(\mathcal{I})}{\mathrm{Opt}(\mathcal{I})} \leq 1 + \frac{T}{\mathrm{Opt}(\mathcal{I})}$$

Si $\mathrm{Opt}(\mathcal{I}) \geq T$, se sigue que

$$\frac{\mathrm{Alg}_{\mathrm{LP}}(\mathcal{I})}{\mathrm{Opt}(\mathcal{I})} \leq 1 + \frac{T}{\mathrm{Opt}(\mathcal{I})} \leq 2.$$

Para el caso en que $\mathrm{Opt}(\mathcal{I}) < T$, supongamos que $\mathrm{Opt}(\mathcal{I}) = \hat{T} < T$. Además, sean $P_1 = \{q \in P\colon \left\lceil z_q^* \right\rceil = z_q^*\}$ y $P_2 = P \setminus P_1$. Entonces

$$\begin{aligned}
\mathrm{Alg}_{\mathrm{LP}}(\mathcal{I}) &= \sum_{q \in P} \left\lceil z_q^* \right\rceil \\
&= \sum_{q \in P_1} \left\lceil z_q^* \right\rceil + \sum_{q \in P_2} \left\lceil z_q^* \right\rceil \\
&\leq \sum_{q \in P_1} z_q^* + \sum_{q \in P_2} (z_q^* + 1) \\
&= \sum_{q \in P} z_q^* + |P_2| \\
&\leq \hat{T} + |P_2|\,.
\end{aligned}$$

Sea $\mathrm{Sol}_{\mathrm{Opt}}(\mathcal{I}) = (\mathcal{J}, \mathcal{Z})$ la solución óptima de $\mathcal{I}$, donde $\mathcal{Z} = (z_1^{\mathrm{Opt}}, z_2^{\mathrm{Opt}}, \cdots, z_T^{\mathrm{Opt}})$. Como $\mathrm{Opt}(\mathcal{I}) = \hat{T} < T$, podemos concluir que a lo más en $\hat{T}$ periodos se tiene que $z_q^{\mathrm{Opt}} > 0$. En los periodos restantes, debe cumplirse que $z_q^* \leq z_q^{\mathrm{Opt}} = 0$, es decir, $z_q^* = 0$. Luego, a lo más en $\hat{T}$ periodos se tiene que $\left\lceil z_q^* \right\rceil \neq z_q^*$, es decir, $|P_2| \leq \hat{T}$, de donde

$$\mathrm{Alg}_{\mathrm{LP}}(\mathcal{I}) \leq \hat{T} + |P_2| \leq 2\hat{T} = 2\mathrm{Opt}(\mathcal{I}).$$

Equivalentemente,

$$\frac{\mathrm{Alg}_{\mathrm{LP}}(\mathcal{I})}{\mathrm{Opt}(\mathcal{I})} \leq 2.$$

□

3.5. Otras cotas generales

En esta sección se presentan dos resultados generales para acotar los valores de cualquier solución factible para el SRDM-T encontrada por los Algoritmos 2, 4 y 5.

Recordemos que en la formulación del modelo S-SRDM-T presentada en la Sección 1.4.2 se definió el conjunto $J_t = \{(j,\ell)\colon j \in J, \ell \in [\gamma_j], t \in [r_{j,\ell}, d_{j,\ell})\}$ como el conjunto de todos los posibles intervalos que se encuentran en conflicto en el instante de tiempo t y $H_q = \{t_q\} \cup \{r_{j,\ell}\colon j \in J, \ell \in [\gamma_j], t_q \leq r_{j,\ell} < t_q + L\}$ es el conjunto de todos los instantes de tiempo en los cuales se puede iniciar el procesamiento de un trabajo en el periodo $q \in P$ (incluimos t_q en este conjunto para contabilizar aquellos trabajos que venían procesándose desde el periodo anterior). Empleando estos conjuntos podemos formular la siguiente proposición.

Proposición 3.5.1. *El valor de cualquier solución óptima del problema SRDM-T está acotada superiormente por*

$$\sum_{q \in P} \max_{t \in H_q} |J_t| .$$

Demostración. Sean $\mathcal{I}$ una instancia del problema SRDM-T y $\text{Sol}(\mathcal{I}) = (\mathcal{J}, \mathcal{Z})$ una solución óptima para $\mathcal{I}$. Para esta solución sabemos que el costo está dado $\text{costo}(\mathcal{Z}) = \sum_{q \in P} z_q$, donde z_q es el número de máquinas usadas en el periodo $q \in p$.

Sea $q \in P$ un periodo cualquiera tal que

$$z_q > \max_{t \in H_q} |J_t| .$$

Es decir, para todo $t \in H_q$, se tiene que

$$z_q > |J_t| .$$

Pero si esto fuera cierto y dado que $z_q \in \mathbb{N}$, significa que

$$z_q - 1 \geq |J_t| , \qquad \forall t \in H_q,$$

lo cual es una contradicción con el supuesto de que la solución es óptima, es decir que $\sum_{q \in p} z_q$ es la mínima cantidad de periodos-máquina requeridos para procesar todos los trabajos. Por lo tanto, para todo $q \in P$ se satisface que

$$z_q \leq \max_{t \in H_q} |J_t| .$$

Sumando sobre todos los periodos se tiene que

$$\text{costo}(\mathcal{Z}) \leq \sum_{q \in P} \max_{t \in H_q} |J_t| .$$

□

Del resultado anterior se satisface trivialmente que, para toda solución óptima,

$$\text{costo}(\mathcal{Z}) \leq T \max_{t \in H} |J_t| \leq mT,$$

donde $H = \bigcup_{q \in P} H_q$.

Proposición 3.5.2. *El valor de cualquier solución factible para el SRDM-T está acotado inferiormente por*

$$\sum_{q \in P} \max_{t \in H_q} |N_t| ,$$

donde $N_t := \left\{ j \colon j \in J, t \in [r_{j,\gamma_j}, d_{j,0}) \right\}$.

Demostración. Sean $\mathcal{I}$ una instancia del problema SRDM-T y $\text{Sol}(\mathcal{I}) = (\mathcal{J}, \mathcal{Z})$ una solución factible para $\mathcal{I}$. Para esta solución se tiene que $\text{costo}(\mathcal{Z}) = \sum_{q \in P} z_q$, donde z_q es el número de máquinas usadas en el periodo $q \in P$.

Supongamos que existe un periodo $q \in P$ tal que

$$z_q < \max_{t \in H_q} |N_t| .$$

Sea $t^* \in H_q$ el tiempo para el cual se alcanza este máximo, es decir, $|N_{t^*}| = \max_{t \in H_q} |N_t|$. Notar que cada trabajo $j \in N_t$ ocupará una máquina en el instante t^*, indistintamente del intervalo seleccionado para el procesamiento de j. De aquí se concluye que existen al menos $|N_{t^*}|$ solapamientos en el instante de tiempo $t^* \in H_q$. Por lo tanto, se requieren al menos $|N_{t^*}|$ máquinas encendidas en el periodo q, lo que contradice el supuesto de que $z_q < |N_{t^*}|$. Luego,

$$z_q \geq \max_{t \in H_q} |N_t| , \qquad \forall q \in P.$$

Sumando sobre todos los periodos se tiene que

$$\text{costo}(\mathcal{Z}) \geq \sum_{q \in P} \max_{t \in H_q} |N_t| .$$

□

Capítulo 4

Resultados Computacionales

En este capítulo se reportan los resultados de experimentos computacionales realizados con los algoritmos descritos en el Capítulo 3. Se analizan el tiempo de ejecución de los modelos lineales y los algoritmos de aproximación.

Los experimentos se realizaron en un computador MacBook Pro con CPU Apple M1 de 8 núcleos y memoria RAM de 16 GB; corriendo el sistema operativo macOS Big Sur y empleando Gurobi 9.1 [14] como herramienta para resolver los programas lineales enteros. Los modelos y algoritmos están implementados en el lenguaje Python [26] utilizando la interfaz de Jupyter notebooks [18].

4.1. Instancias

Las pruebas computacionales se realizaron sobre instancias con datos reales obtenidos de la base de datos del SRI. La base cuenta con datos de diferentes sucursales, de ellas se seleccionó una de las sucursales más grandes (Agencia Plataforma Gubernamental Amazonas). Los datos seleccionados corresponden a los clientes de 15 días seleccionados aleatoriamente; 5 días correspondientes al mes de octubre del 2017, 5 días del mes de enero del 2018 y 5 días pertenecientes al mes de junio del 2018. De cada cliente se extrajo el tiempo de llegada (r_j) y el tiempo de atención (p_j).

Para analizar la versión del Algoritmo 5 fijaremos el tiempo de atención para todos los clientes igual a 10 minutos, este valor corresponde a una aproximación del tiempo promedio de atención de los clientes.

La Tabla 4.1 contiene los detalles de las instancias de prueba. De izquierda a derecha,

las columnas muestran la siguiente información: nombre de la instancia (ddmmyy), número de clientes (trabajos) y número de periodos.

Instancia	n	T
03Oct17	995	13
19Oct17	920	14
25Oct17	904	13
27Oct17	733	13
30Oct17	947	13
04Ene18	811	10
05Ene18	800	10
12Ene18	844	10
15Ene18	892	10
31Ene18	884	10
01Jun18	744	11
07Jun18	811	11
08Jun18	779	10
21Jun18	784	10
25Jun18	760	10

Tabla 4.1: Descripción de las instancias de prueba.

4.2. Comparación entre el modelo M-SRDM-T y la heurística en dos fases (modelo S-SRDM-T + Alg. de coloreo glotón)

En esta sección se presenta una comparación entre el modelo M-SRDM-T y la heurística en dos fases descrita en la Sección 1.4.1. Para cada formulación se corre el algoritmo Branch-and-Cut implementado en Gurobi [14] sobre las instancias de prueba. Se emplean los valores por defecto de los parámetros de configuración del solver. En la Tabla 4.2 se muestran los resultados para el modelo y la heurística. De izquierda

a derecha, se presentan el nombre de la instancia, el mejor valor para la función objetivo (Obj.), el mejor valor para la cota inferior (LB), el valor relativo de la brecha de optimalidad (GAP) en porcentaje, el tiempo de ejecución (Time) en segundos, la cantidad de nodos explorados (Nodes), el número de variables (Vars.) y el número de restricciones (Constr.) del programa lineal entero. Para todas las instancias se utilizó un tiempo de parada de $3,600$ segundos, 25 ventanillas disponibles ($m_q = 25$ para cada $q \in P$) y el valor de $\gamma_j = 20$ para cada $j \in J$. Además, el tiempo reportado en la heurística de dos fases incluye el tiempo de ejecución de ambas fases.

Como se puede notar, en cuanto al número de variables y restricciones; el modelo M-SRDM-T es mucho más grande que el modelo S-SRDM-T. Las variables son en promedio 25 veces menos en el modelo S-SRDM-T y las restricciones son en promedio casi 13 veces menos en el modelo S-SRDM-T, respecto al modelo M-SRDM-T. Adicionalmente, el tiempo disminuye considerablemente, el modelo S-SRDM-T resuelve las instancias hasta la optimalidad en no más de 10 minutos. Mientras que el modelo M-SRDM-T no logra encontrar soluciones optimas en una hora de ejecución. De este hecho, podemos concluir que la heurística en dos fases nos permite encontrar soluciones al problema PATV en un menor tiempo de ejecución. Lo que nos permite una vez más corroborar por qué en la literatura el análisis se enfoca en tratar de resolver problemas del estilo S-SRDM-T en lugar de resolver el problema con un modelo del estilo M-SRDM-T.

En la siguiente sección procedemos a verificar los algoritmos de aproximación estudiados en las Secciones 2.2 y 3.2.

			Modelo M-SRDM-T							Heurística en dos fases					
Instancia	n	T	Obj	LB.	Gap(%)	Time	Nodos	Vars	Constr	Obj	Gap(%)	Time	Nodos	Vars	Constr
03Oc17	995	13	289	170	41.18	3639.11	0	522700	18795	170	0.00	569.59	22384	20908	1707
19Oc17	920	14	162	155	4.32	3637.28	1	483350	19095	156	0.00	127.88	3082	19334	1647
25Oc17	904	13	281	145	48.40	3636.39	0	474925	19254	146	0.00	543.79	25716	18997	1638
27Oc17	733	13	121	117	3.31	3629.08	1	385150	18708	117	0.00	291.47	9525	15406	1452
30Oc17	947	13	154	149	3.25	3638.05	1	497500	19147	150	0.00	360.60	11167	19900	1675
04Ja18	811	10	129	127	1.55	3627.44	1	426025	15186	127	0.00	70.15	2422	17041	1386
05Ja18	800	10	123	120	2.44	3625.25	1	420250	14750	121	0.00	225.73	9704	16810	1358
12Ja18	844	10	138	135	2.17	3627.82	1	443350	14819	136	0.00	19.81	1	17734	1403
15Ja18	892	10	146	141	3.42	3628.33	1	468550	14742	142	0.00	194.43	9070	18742	1446
31Ja18	884	10	149	146	2.01	3628.78	1	464350	14934	146	0.00	53.11	285	18574	1446
01Ju18	744	11	120	118	1.67	3623.49	1	390875	14644	118	0.00	283.95	10633	15635	1300
07Ju18	811	11	137	135	1.46	3626.39	1	426050	14786	135	0.00	57.04	883	17042	1370
08Ju18	779	10	125	122	2.40	3625.26	1	409225	14779	123	0.00	288.40	9117	16369	1339
21Ju18	784	10	130	126	3.08	3624.69	1	411850	14759	127	0.00	249.76	7555	16474	1343
25Ju18	760	10	126	124	1.59	3624.16	1	399250	14810	124	0.00	57.90	1320	15970	1322

Tabla 4.2: Resultados del modelo M-SRDM-T y la heurística en dos fases.

4.3. Evaluación de algoritmos de aproximación para el problema SRDM-T

En esta sección se reportan los resultados de los experimentos computacionales sobre los Algoritmos 4 y 2 propuestos en las Secciones 2.2 y 3.2. Las mejores cotas obtenidas para cada instancia, entre todos los escenarios de cada tabla, se indican siempre en negrilla.

Recordemos que los algoritmos de aproximación en general tienen tiempos de ejecución muy cortos, por tanto el tiempo que se presenta considera desde la lectura de los datos hasta que finaliza el algoritmo.

En la Tabla 4.3 se muestran los resultados obtenidos para las cuatro versiones del Algoritmo 4 descritas al final de la Sección 3.2.

De las cuatro versiones del Algoritmo 4, podemos observar que la Versión 4 obtiene las mejores cotas en 10 de las 15 instancias de prueba. En segundo lugar se encuentra la Versión 1, que a pesar de nuestros resultados teóricos para el factor de aproximación, en las instancias de prueba se comporta bastante bien. En la Tabla 4.4 se muestra los valores óptimos obtenidos por el modelo S-SRDM-T y se comparan con las cotas obtenidas por el Algoritmo 4 (Versión 1). En la última columna se muestran los cocientes entre la solución encontrada por la versión 1 del Algoritmo 4 y la solución óptima, los cuales están por debajo de 1.3 para todas las instancias de prueba.

En la Tabla 4.5 se reportan los resultados obtenidos por el Algoritmo 2. De izquierda a derecha, se presentan el nombre de la instancia, el número de clientes (n), el número de periodos (T), la cantidad óptima de periodos-ventanilla ($\text{Opt}(\mathcal{I})$), la cantidad de periodos-ventanilla obtenida por la versión 1 del Algoritmo 4 ($G(\mathcal{I})$), la cantidad de periodos-ventanilla obtenida por el Algoritmo 2 ($\text{GBF}(\mathcal{I})$) y el tiempo de ejecución del Algoritmo 2 (Time) en segundos.

De los resultados mostrados en la Tabla 4.5 podemos concluir que la versión 1 del Algoritmo 4 obtiene mejores soluciones en todas las instancias de prueba.

			Versión 1		Versión 2		Versión 3		Versión 4	
Instancia	n	T	$G(\mathcal{I})$	Time	$G(\mathcal{I})$	Time	$G(\mathcal{I})$	Time	$G(\mathcal{I})$	Time
03Oct17	995	13	199	1.81	341	2.23	202	1.94	**198**	1.99
19Oct17	920	14	192	1.53	331	1.90	201	1.67	**188**	1.67
25Oct17	904	13	176	1.48	302	1.83	183	1.67	**173**	1.63
27Oct17	733	13	143	0.98	276	1.18	150	1.08	**138**	1.08
30Oct17	947	13	175	1.61	311	1.97	186	1.77	**172**	1.83
04Ene18	811	10	151	1.16	248	1.46	150	1.27	**147**	1.32
05Ene18	800	10	**143**	1.13	248	1.46	144	1.25	144	1.31
12Ene18	844	10	160	1.25	255	1.53	157	1.40	**154**	1.44
15Ene18	892	10	**167**	1.43	272	1.77	**167**	1.58	**167**	1.57
31Ene18	884	10	169	1.38	270	1.68	**165**	1.51	169	1.51
01Jun18	744	11	**140**	0.98	251	1.22	146	1.08	141	1.12
07Jun18	811	11	**158**	1.16	252	1.43	166	1.27	159	1.33
08Jun18	779	10	**150**	1.08	247	1.34	**150**	1.21	153	1.24
21Jun18	784	10	151	1.09	255	1.36	158	1.19	**150**	1.22
25Jun18	760	10	**146**	1.15	253	1.26	158	1.14	**146**	1.14

Tabla 4.3: Comparando criterios de selección del Algoritmo 4.

Instancia	n	T	$\text{Opt}(\mathcal{I})$	$G(\mathcal{I})$	$\frac{G(\mathcal{I})}{\text{Opt}(\mathcal{I})}$
03Oct17	995	13	170	199	1.17
19Oct17	920	14	156	192	1.23
25Oct17	904	13	146	176	1.21
27Oct17	733	13	117	143	1.22
30Oct17	947	13	150	175	1.17
04Ene18	811	10	127	151	1.19
05Ene18	800	10	121	143	1.18
12Ene18	844	10	136	160	1.18
15Ene18	892	10	142	167	1.18
31Ene18	884	10	146	169	1.16
01Jun18	744	11	118	140	1.19
07Jun18	811	11	135	158	1.17
08Jun18	779	10	123	150	1.22
21Jun18	784	10	127	151	1.19
25Jun18	760	10	124	146	1.18

Tabla 4.4: Análisis del factor de aproximabilidad para la versión 1 del Algoritmo 4.

4.4. Evaluación de algoritmos de aproximación para el problema SRDM-TD

El Algoritmo 5 solo se puede emplear en instancias con tiempos de duración de trámite igual para cada cliente. Para las pruebas computacionales, se emplearon las mismas instancias descritas en la Tabla 4.1, pero fijando los tiempos de duración de todos los trámites a 10 minutos. Para garantizar la factibilidad, fue necesario también incrementar el número de ventanillas disponibles de 25 a 35. Adicionalmente se corrieron el Algoritmo 2 y la versión 1 del Algoritmo 4 para comparar los resultados.

En la Tabla 4.6 se muestran los resultados de los tres algoritmos antes mencionados para las instancias modificadas. De cada algoritmo se muestra el número de periodos-ventanilla y el tiempo de procesamiento. Además, se muestra el valor óptimo para el número de periodos-ventanilla por cada instancia. Como se puede notar, los Algoritmos 2 y 4 tienen los tiempos de procesamiento más bajos, mientras que para el Algoritmo 5

Instancia	n	T	Opt($\mathcal{I}$)	$G(\mathcal{I})$	GBF($\mathcal{I}$)	Time
03Oct17	995	13	170	199	208	22.61
19Oct17	920	14	156	192	197	19.37
25Oct17	904	13	146	176	181	17.17
27Oct17	733	13	117	143	148	11.34
30Oct17	947	13	150	175	183	19.02
04Ene18	811	10	127	151	156	13.49
05Ene18	800	10	121	143	150	12.21
12Ene18	844	10	136	160	168	14.57
15Ene18	892	10	142	167	173	16.22
31Ene18	884	10	146	169	177	16.59
01Jun18	744	11	118	140	146	11.47
07Jun18	811	11	135	158	164	14.13
08Jun18	779	10	123	150	156	12.35
21Jun18	784	10	127	151	161	12.94
25Jun18	760	10	124	146	157	12.09

Tabla 4.5: Comparación de resultados con el Algoritmo 2.

el tiempo de procesamiento varía entre dos y cinco minutos. Por otra parte, los valores de las soluciones encontradas por el Algoritmo 5, son más cercanos al óptimo. En las instancias de prueba, el cociente entre el valor de la solución del Algoritmo 5 y el valor de la solución óptima estuvo entre 1.02 y 1.04. En la Tabla 4.7 se comparan estos valores con las cotas teóricas demostradas en el Capítulo 3. Puede notarse que el valor de la cota teórica se ajusta bastante al valor real sobre las instancias de prueba.

				Algoritmo 2		Algoritmo 4		Algoritmo 5	
Instancia	n	T	Opt($\mathcal{I}$)	GBF($\mathcal{I}$)	Time	$G(\mathcal{I})$	Time	$\text{Alg}_{\text{LP}}(\mathcal{I})$	Time
03Oct17	995	13	167	198	19.45	198	2.43	172	286.34
19Oct17	920	14	156	207	16.46	205	2.13	163	296.08
25Oct17	904	13	153	188	13.96	187	2.07	157	299.43
27Oct17	733	13	123	149	9.30	152	1.46	128	277.67
30Oct17	947	13	160	191	16.91	196	2.25	164	300.89
04Ene18	811	10	137	162	11.79	162	1.73	141	160.29
05Ene18	800	10	137	158	11.51	160	1.69	141	150.24
12Ene18	844	10	143	163	12.54	163	1.87	147	154.69
15Ene18	892	10	151	174	14.24	175	2.00	154	156.28
31Ene18	884	10	150	173	13.85	175	1.96	154	158.99
01Jun18	744	11	126	150	10.78	151	1.50	130	147.37
07Jun18	811	11	138	170	12.92	170	1.74	141	152.87
08Jun18	779	10	134	163	11.87	164	1.63	138	152.54
21Jun18	784	10	134	168	11.79	167	1.66	138	150.06
25Jun18	760	10	130	167	10.89	167	1.57	134	151.93

Tabla 4.6: Comparación de resultados de los Algoritmos 2, 4 y 5 en instancias con tiempos de procesamiento constantes.

Instancia	n	T	$\mathrm{Opt}(\mathcal{I})$	$\mathrm{Alg}_{\mathrm{LP}}(\mathcal{I})$	$\frac{\mathrm{Alg}_{\mathrm{LP}}(\mathcal{I})}{\mathrm{Opt}(\mathcal{I})}$	$1+\frac{T}{\mathrm{Opt}(\mathcal{I})}$
03Oct17	995	13	167	172	1.03	1.08
19Oct17	920	14	156	163	1.04	1.09
25Oct17	904	13	153	157	1.03	1.08
27Oct17	733	13	123	128	1.04	1.11
30Oct17	947	13	160	164	1.03	1.08
04Ene18	811	10	137	141	1.03	1.07
05Ene18	800	10	137	141	1.03	1.07
12Ene18	844	10	143	147	1.03	1.07
15Ene18	892	10	151	154	1.02	1.07
31Ene18	884	10	150	154	1.03	1.07
01Jun18	744	11	126	130	1.03	1.09
07Jun18	811	11	138	141	1.02	1.08
08Jun18	779	10	134	138	1.03	1.07
21Jun18	784	10	134	138	1.03	1.07
25Jun18	760	10	130	134	1.03	1.08

Tabla 4.7: Evaluación del factor de aproximabilidad del Algoritmo 5.

Capítulo 5

Conclusiones

El problema SRDM es un problema que busca asignar un conjunto de trabajos, dentro de sus ventanas de tiempo, a un conjunto de máquinas idénticas de manera que se minimice la cantidad total de máquinas usadas. Este problema es conocido por ser NP-difícil en general, por lo que se han estudiado varias variantes y algoritmos de aproximación.

Una de estas variantes es el problema SRDM-T, el cual se define como una extensión del problema SRDM en el que el horizonte de planificación se encuentra dividido en periodos, y cada máquina disponible puede ser encendida o apagada en cada periodo. En este problema, el objetivo es encontrar una asignación de trabajos a máquinas que minimice la cantidad total de periodos-máquina, es decir, la suma del número de periodos en que cada maquina está encendida.

Dado que el problema SRDM-T es NP-difícil, resulta importante estudiar algoritmos de aproximación. En este sentido, se han propuesto varios algoritmos para el problema SRDM-T que son extensiones de algoritmos previamente desarrollados para el problema SRDM. En particular, se ha encontrado que los algoritmos expuestos en las Secciones 2.2 y 2.3 para el problema SRDM se pueden extender de manera sencilla al problema SRDM-T. Sin embargo, se ha demostrado que el Algoritmo 2 y el Algoritmo 4 no poseen un factor de aproximación constante para el problema SRDM-T.

Adicionalmente, se han estudiado cuatro versiones del Algoritmo 4, obtenidas al cambiar el criterio de selección de trabajos. De los cuatro criterios, se analizaron dos, y se determinó que para ninguno de ellos el algoritmo posee un factor de aproximación

constante.

Posteriormente, se estudió el Algoritmo 2 para el problema SRDM-T, y se construyó una familia de instancias sobre las cuales el algoritmo alcanza un factor de aproximación de $\Omega(\log(n))$, es decir, el Algoritmo 2 no posee un factor de aproximación constante. Sin embargo, si el máximo tiempo de espera permitido previo al inicio de la ejecución del trabajo $j \in J$ es mayor o igual a su duración menos una unidad, es decir, si $\gamma_j \geq p_j - 1$, entonces el factor de aproximación es igual a $9T\frac{p_{\text{máx}}}{p_{\text{mín}}}$. Si además se tiene que al menos dos trabajos tienen duraciones diferentes, es decir, $p_{\text{máx}} > p_{\text{mín}}$, entonces el factor de aproximación se reduce a $18T\log_2\left(\frac{p_{\text{máx}}}{p_{\text{mín}}}\right)$. Por otra parte, si suponemos que las duraciones son iguales para todos los trabajos, entonces podemos reducir el factor de aproximación de $9T$ a $7T$. Además, el ejemplo propuesto nos permite obtener una cota inferior de 5 para el factor de aproximabilidad.

Por otra parte, mientras que el Algoritmo 2 tiene un factor de aproximabilidad de $7T$ para el caso en el que las duraciones del tiempo de procesamiento son iguales para todos los trabajos, el Algoritmo 5 nos proporciona un factor de dos. Más aún, se demostró que el factor de aproximabilidad del Algoritmo 5 es igual a $1 + \frac{T}{\text{Opt}(\mathcal{I})}$. Esto significa que cuanto menor sea el valor de $\frac{T}{\text{Opt}(\mathcal{I})}$, mejor será la solución encontrada por el algoritmo en comparación con la solución óptima. En otras palabras, cuanto menor sea la relación entre el número de periodos y el costo de la solución óptima, más cercana será la solución encontrada por el algoritmo a la solución óptima.

El hecho de que el Algoritmo 5 resuelva el problema SRDM-T con $T = 1$ y duraciones de los trabajos constantes en tiempo polinomial es importante, ya que nos muestra que para este caso particular existe un algoritmo exacto que puede resolver el problema de manera eficiente. Sin embargo, este resultado no se extiende al caso general con $T \geq 2$.

La complejidad computacional del problema SRDM-T con duraciones constantes de los trabajos y $T \geq 2$ sigue siendo un problema abierto. Por lo tanto, el estudio de algoritmos de aproximación para el problema SRDM-T sigue siendo importante, ya que los mismos se pueden emplear para encontrar soluciones cercanas a la óptima en un tiempo razonable. Además, este problema tiene aplicaciones prácticas en la asignación de tareas en la industria, por lo que seguir investigando su resolución es importante para mejorar la eficiencia de los procesos productivos.

De los experimentos computacionales, podemos decir que los Algoritmos 2 y 4 obtienen buenas cotas para las instancias con datos reales. De hecho, se puede ver que el criterio de selección es muy importante en el Algoritmo 4, obteniendo los mejores resultados cuando se minimiza el tiempo de retraso. Sin embargo, aunque el Algoritmo 2 intenta usar la menor cantidad de máquinas disponibles, no nos lleva a obtener los mejores resultados para el problema SRDM-T. Aunque en los resultados teóricos no se ha encontrado un factor asintótico para el Algoritmo 4, en la práctica podemos ver que el algoritmo se comporta muy bien. De hecho, el factor de aproximabilidad para las instancias de prueba no es mayor a 1.3.

Por otra parte, cuando el caso se reduce a problemas con duraciones iguales, el Algoritmo 5 resulta ser el mejor de los tres. Es decir, a pesar de la mayor cantidad de tiempo necesaria para construir y resolver el modelo lineal, se obtienen soluciones factibles cuyo valor de periodos-máquina es el más cercano al valor óptimo. Notemos además que los Algoritmos 2 y 4 encuentran soluciones muy parecidas, por lo que si se quiere encontrar soluciones rápidas, la mejor alternativa es el Algoritmo 4.

Algunas direcciones de investigación que pueden ser abordadas en trabajos futuros son las siguientes:

- Estudiar más a fondo la complejidad del problema para el caso del problema SRDM-T con duraciones de procesamiento iguales.
- Estudiar nuevos algoritmos de aproximación para el problema SRDM-T.
- Estudiar otras formulaciones lineales enteras para el problema SRDM-T.
- Analizar los problemas duales de los modelos propuestos.
- Estudiar el factor de aproximabilidad para las demás versiones del Algoritmo 4.
- Estudiar nuevas versiones del Algoritmo 2 modificando el criterio de selección de intervalos.

Bibliografía

[1] M. Adler, A. L. Rosenberg, R. K. Sitaraman, and W. Unger. Scheduling time-constrained communication in linear networks. *Theory of Computing Systems*, 35(6):599–623, 2002.

[2] P. Baptiste. Scheduling equal-length jobs on identical parallel machines. *Discrete Applied Mathematics*, 103(1-3):21–32, 2000.

[3] A. Bar-Noy, R. Canetti, S. Kutten, Y. Mansour, and B. Schieber. Bandwidth allocation with preemption. *SIAM journal on computing*, 28(5):1806–1828, 1999.

[4] J. Chuzhoy, S. Guha, S. Khanna, and J. Naor. Machine minimization for scheduling jobs with interval constraints. In *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, volume 2004, pages 81–90, 2004.

[5] J. Chuzhoy, R. Ostrovsky, and Y. Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research*, 31(4):730–738, 2006.

[6] M. Cieliebak, T. Erlebach, F. Hennecke, B. Weber, and P. Widmayer. Scheduling with release times and deadlines on a minimum number of machines. In *Exploring new frontiers of theoretical informatics*, pages 209–222. Springer, 2004.

[7] M. Conforti, G. Cornuéjols, and G. Zambelli. *Integer programming*, volume 271. Springer, 2014.

[8] S. Cook. The complexity of theorem-proving procedures. *Proceedings of the 3rd Symposium on the Theory of Computing, ACM*, page 151–158, 1971.

[9] S. Cook. The p versus np problem. *Clay Mathematics Institute*, 2, 2000.

[10] G. B. Dantzig and D. Fulkerson. Minimizing the number of carriers to meet a fixed schedule. Technical report, RAND CORP SANTA MONICA CA, 1954.

[11] L. Ford and D. Fulkerson. Flows in networks. princeton university press, princeton, new jersey. 1962.

[12] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., USA, 1990.

[13] U. I. Gupta, D.-T. Lee, and J.-T. Leung. An optimal solution for the channel-assignment problem. *IEEE Transactions on Computers*, 28(11):807–810, 1979.

[14] L. Gurobi Optimization. Gurobi optimizer reference manual, 2019.

[15] N. G. Hall and M. J. Magazine. Maximizing the value of a space mission. *European journal of operational research*, 78(2):224–241, 1994.

[16] A. Hashimoto and J. Stevens. Wire routing by optimizing channel assignment within large apertures. In *Proceedings of the 8th Design Automation Workshop*, pages 155–169, 1971.

[17] R. Karp. *Reducibility among combinatorial problems*. In: Miller R.E., Thatcher J.W., Bohlinger J.D. (eds) Complexity of Computer Computations. The IBM Research Symposia Series, Springer, Boston, MA, 1972.

[18] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing. Jupyter notebooks – a publishing format for reproducible computational workflows. In F. Loizides and B. Schmidt, editors, *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87 – 90. IOS Press, 2016.

[19] A. W. Kolen, J. K. Lenstra, C. H. Papadimitriou, and F. C. Spieksma. Interval scheduling: A survey. *Naval Research Logistics (NRL)*, 54(5):530–543, 2007.

[20] S. A. Kravchenko and F. Werner. Minimizing the number of machines for scheduling jobs with equal processing times. *European Journal of Operational Research*, 199(2):595–600, 2009.

[21] L. A. Levin. Universal sequential search problems. *Problemy peredachi informatsii*, 9(3):115–116, 1973.

[22] A. Mingozzi, M. A. Boschetti, S. Ricciardelli, and L. Bianco. A set partitioning approach to the crew scheduling problem. *Operations Research*, 47(6):873–888, 1999.

[23] I. Méndez-Díaz. *Problema de Coloreo de Grafos, Un Estudio Poliedral y un Algoritmo Branch-and-Cut*. Tesis doctoral, Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires, 2003.

[24] E. Sanlaville and G. Schmidt. Machine scheduling with availability constraints. *Acta Informatica*, 35(9):795–811, 1998.

[25] F. C. Spieksma. On the approximability of an interval scheduling problem. *Journal of Scheduling*, 2(5):215–227, 1999.

[26] G. Van Rossum and F. L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

[27] G. Yu and G. Zhang. Scheduling with a minimum number of machines. *Operations Research Letters*, 37(2):97–101, 2009.

Anexos

Anexo A

Modelo lineal del Ejercicio 2.3.1

Las restricciones (2.3.10) de forma desglosada son:

$$x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} + x_{1,6} = 4,$$
$$x_{2,1} + x_{2,2} + x_{2,3} + x_{2,4} + x_{2,5} + x_{2,6} = 4,$$
$$x_{3,1} + x_{3,2} + x_{3,3} + x_{3,4} + x_{3,5} + x_{3,6} = 4,$$
$$x_{4,1} + x_{4,2} + x_{4,3} + x_{4,4} + x_{4,5} + x_{4,6} = 4.$$

Las restricciones (2.3.11) se detallan a continuación (en paréntesis se muestran los valores de q, κ y i para los cuales se satisface la condición de $\bigcap_{\eta\in\{1,2,\cdots,\kappa\}} I_{i+\eta} \cap [t_q, t_q + L_q) \neq \emptyset$):

$$x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} \leq 4z, \qquad (q = 1, \kappa = 1, i = 0)$$
$$x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} \leq 8, \qquad (q = 1, \kappa = 1, i = 0)$$
$$x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} \leq 4z, \qquad (q = 1, \kappa = 1, i = 1)$$
$$x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} \leq 8, \qquad (q = 1, \kappa = 1, i = 1)$$
$$x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} \leq 4z, \qquad (q = 1, \kappa = 1, i = 2)$$
$$x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} \leq 8, \qquad (q = 1, \kappa = 1, i = 2)$$
$$x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} + x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} \leq 4z, \qquad (q = 1, \kappa = 2, i = 0)$$
$$x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} + x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} \leq 8, \qquad (q = 1, \kappa = 2, i = 0)$$
$$x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} + x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} \leq 4z, \qquad (q = 1, \kappa = 2, i = 1)$$
$$x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} + x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} \leq 8, \qquad (q = 1, \kappa = 2, i = 1)$$
$$x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} + x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} + x_{1,3} + \qquad (q = 1, \kappa = 3, i = 0)$$
$$x_{2,3} + x_{3,3} + x_{4,3} \leq 4z,$$

$$x_{1,1}+x_{2,1}+x_{3,1}+x_{4,1}+x_{1,2}+x_{2,2}+x_{3,2}+x_{4,2}+x_{1,3}+ \qquad (q=1,\kappa=3,i=0)$$
$$x_{2,3}+x_{3,3}+x_{4,3}\leq 8,$$
$$x_{1,2}+x_{2,2}+x_{3,2}+x_{4,2}\leq 4z, \qquad (q=2,\kappa=1,i=1)$$
$$x_{1,2}+x_{2,2}+x_{3,2}+x_{4,2}\leq 4, \qquad (q=2,\kappa=1,i=1)$$
$$x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}\leq 4z, \qquad (q=2,\kappa=1,i=2)$$
$$x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}\leq 4, \qquad (q=2,\kappa=1,i=2)$$
$$x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}\leq 4z, \qquad (q=2,\kappa=1,i=3)$$
$$x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}\leq 4, \qquad (q=2,\kappa=1,i=3)$$
$$x_{1,5}+x_{2,5}+x_{3,5}+x_{4,5}\leq 4z, \qquad (q=2,\kappa=1,i=4)$$
$$x_{1,5}+x_{2,5}+x_{3,5}+x_{4,5}\leq 4, \qquad (q=2,\kappa=1,i=4)$$
$$x_{1,2}+x_{2,2}+x_{3,2}+x_{4,2}+x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}\leq 4z, \qquad (q=2,\kappa=2,i=1)$$
$$x_{1,2}+x_{2,2}+x_{3,2}+x_{4,2}+x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}\leq 4, \qquad (q=2,\kappa=2,i=1)$$
$$x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}+x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}\leq 4z, \qquad (q=2,\kappa=2,i=2)$$
$$x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}+x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}\leq 4, \qquad (q=2,\kappa=2,i=2)$$
$$x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}+x_{1,5}+x_{2,5}+x_{3,5}+x_{4,5}\leq 4z, \qquad (q=2,\kappa=2,i=3)$$
$$x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}+x_{1,5}+x_{2,5}+x_{3,5}+x_{4,5}\leq 4, \qquad (q=2,\kappa=2,i=3)$$
$$x_{1,2}+x_{2,2}+x_{3,2}+x_{4,2}+x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}+x_{1,4}+ \qquad (q=2,\kappa=3,i=1)$$
$$x_{2,4}+x_{3,4}+x_{4,4}\leq 4z,$$
$$x_{1,2}+x_{2,2}+x_{3,2}+x_{4,2}+x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}+x_{1,4}+ \qquad (q=2,\kappa=3,i=1)$$
$$x_{2,4}+x_{3,4}+x_{4,4}\leq 4,$$
$$x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}+x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}+x_{1,5}+ \qquad (q=2,\kappa=3,i=2)$$
$$x_{2,5}+x_{3,5}+x_{4,5}\leq 4z,$$
$$x_{1,3}+x_{2,3}+x_{3,3}+x_{4,3}+x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}+x_{1,5}+ \qquad (q=2,\kappa=3,i=2)$$
$$x_{2,5}+x_{3,5}+x_{4,5}\leq 4,$$
$$x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}\leq 4z, \qquad (q=3,\kappa=1,i=3)$$
$$x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}\leq 8, \qquad (q=3,\kappa=1,i=3)$$
$$x_{1,5}+x_{2,5}+x_{3,5}+x_{4,5}\leq 4z, \qquad (q=3,\kappa=1,i=4)$$
$$x_{1,5}+x_{2,5}+x_{3,5}+x_{4,5}\leq 8, \qquad (q=3,\kappa=1,i=4)$$
$$x_{1,6}+x_{2,6}+x_{3,6}+x_{4,6}\leq 4z, \qquad (q=3,\kappa=1,i=5)$$
$$x_{1,6}+x_{2,6}+x_{3,6}+x_{4,6}\leq 8, \qquad (q=3,\kappa=1,i=5)$$
$$x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}+x_{1,5}+x_{2,5}+x_{3,5}+x_{4,5}\leq 4z, \qquad (q=3,\kappa=2,i=3)$$
$$x_{1,4}+x_{2,4}+x_{3,4}+x_{4,4}+x_{1,5}+x_{2,5}+x_{3,5}+x_{4,5}\leq 8, \qquad (q=3,\kappa=2,i=3)$$

$$x_{1,5} + x_{2,5} + x_{3,5} + x_{4,5} + x_{1,6} + x_{2,6} + x_{3,6} + x_{4,6} \leq 4z, \qquad (q = 3, \kappa = 2, i = 4)$$

$$x_{1,5} + x_{2,5} + x_{3,5} + x_{4,5} + x_{1,6} + x_{2,6} + x_{3,6} + x_{4,6} \leq 8, \qquad (q = 3, \kappa = 2, i = 4)$$

$$x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} + x_{1,5} + x_{2,5} + x_{3,5} + x_{4,5} + x_{1,6} + \qquad (q = 3, \kappa = 3, i = 3)$$
$$x_{2,6} + x_{3,6} + x_{4,6} \leq 4z,$$

$$x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} + x_{1,5} + x_{2,5} + x_{3,5} + x_{4,5} + x_{1,6} + \qquad (q = 3, \kappa = 3, i = 3)$$
$$x_{2,6} + x_{3,6} + x_{4,6} \leq 8.$$

Finalmente, las restricciones (2.3.12) y (2.3.13) son:

$$x_{1,6} = 0,$$
$$x_{2,1} = 0,$$
$$x_{2,6} = 0,$$
$$x_{3,1} = 0,$$
$$x_{3,6} = 0.$$

Por otra parte, observemos que de las restricciones (2.3.11), si se satisface la restricción:

$$x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} + x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} + x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} \leq 4z,$$

entonces las restricciones:

$$x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} \leq 4z$$
$$x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} \leq 4z,$$
$$x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} \leq 4z,$$
$$x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} + x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} \leq 4z,$$
$$x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} + x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} \leq 4z,$$

se vuelven redundantes para el modelo. Así, eliminando las variables con el valor de cero y quitando las restricciones redundantes obtenemos el siguiente modelo:

mín z

s.a.r.

$$x_{1,1} + x_{1,2} + x_{1,3} + x_{1,4} + x_{1,5} = 4,$$
$$x_{2,2} + x_{2,3} + x_{2,4} + x_{2,5} = 4,$$
$$x_{3,2} + x_{3,3} + x_{3,4} + x_{3,5} = 4,$$

$$x_{4,1} + x_{4,2} + x_{4,3} + x_{4,4} + x_{4,5} + x_{4,6} = 4,$$

$$x_{1,1} + x_{4,1} + x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} + x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} \leq 4z,$$

$$x_{1,1} + x_{4,1} + x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} + x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} \leq 8,$$

$$x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} + x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} + x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} \leq 4z,$$

$$x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} + x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} + x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} \leq 4,$$

$$x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} + x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} + x_{1,5} + x_{2,5} + x_{3,5} + x_{4,5} \leq 4z,$$

$$x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} + x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} + x_{1,5} + x_{2,5} + x_{3,5} + x_{4,5} \leq 4,$$

$$x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} + x_{1,5} + x_{2,5} + x_{3,5} + x_{4,5} + x_{4,6} \leq 4z,$$

$$x_{1,4} + x_{2,4} + x_{3,4} + x_{4,4} + x_{1,5} + x_{2,5} + x_{3,5} + x_{4,5} + x_{4,6} \leq 8,$$

$$x_{j,i} \geq 0, \quad \forall i \in \{1, 2, \cdots, 6\}, j \in J,$$

$$z \geq 0.$$