UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

INSTITUTO DE INVESTIGACIÓN EN CIENCIAS BÁSICAS Y APLICADAS

CENTRO DE INVESTIGACIÓN EN CIENCIAS

ASPECTOS TEÓRICOS Y PRÁCTICOS DEL PROBLEMA

$1|ri \in \{ri, ..., rk\}, qi \in \{q1, q2\}|Cmax$

Tesis profesional

para obtener el grado de

LICENCIADO EN CIENCIAS

ÁREA TERMINAL EN MATEMÁTICAS

PRESENTA

DIANA ESCALONA CONTRERAS

DIRECTOR DE TESIS

DR. NODARI VAKHANIA MAISURADZE

Cuernavaca, Morelos

2024

Esta tesis fue evaluada por el siguiente Comité Tutelar:

Presidente	Dr. Antonio Daniel Rivera López
Secretario	Dra. Lorena Díaz González
Vocal	Dr. Nodari Vakhania Maisuradze
Suplente	Dra. María Elisa Chinos Olivan
Suplente	Dr. Dan Sidney Díaz Guerrero

Resumen

La optimización combinatoria (o discreta) es uno de los campos más activos en el área de las matemáticas discretas. Los problemas de optimización combinatoria se presentan en diversas áreas, tales como programación lineal, programación lineal entera, teoría de grafos, inteligencia artificial y teoría de números. Todos estos problemas, cuando se formulan matemáticamente como la minimización o maximización de una determinada función definida en un dominio, tienen en común un carácter discreto. Algunos de los problemas de optimización combinatoria más conocidos son el problema de suma de subconjuntos, el problema del agente viajero, el problema de bin packing y el problema de calendarización.

Los modelos de calendarización en una sola máquina constituyen el núcleo de la investigación del problema de calendarización debido a que pueden ser utilizados para resolver modelos más complejos. Por esta razón, la investigación del problema de calendarización en una sola máquina ha atraído un nivel de atención significativo en la literatura y, como consecuencia de ello, la publicación de varios artículos enfocados no solo en el método de solución sino también en las características particulares del problema. El término calendarización se refiere a la asignación de trabajos sobre un conjunto de recursos dado a lo largo del tiempo cumpliendo con un criterio de optimalidad dado.

En este proyecto de investigación se estudia un problema de calendarización en una sola máquina con un número constante de tiempos de liberación y dos tiempos de entrega permitidos. Este es un caso particular del problema general de calendarización NP-duro en el sentido estricto en una sola máquina con un número arbitrario de tiempos de liberación y un número arbitrario de tiempos de entrega cuyo objetivo es minimizar el máximo tiempo de completés total; por lo tanto, nuestro problema particular sigue siendo NP-duro. El estudio de este problema es de utilidad para la solución del problema general; no obstante, el caso restringido del problema de calendarización es de utilidad por su cuenta debido a que se adapta a algunas aplicaciones de la vida real.

Se llevó a cabo el estudio de las propiedades estructurales del problema de calendarización en una sola máquina con un número constante de tiempos de liberación y dos tiempos de entrega permitidos con la finalidad de establecer condiciones de optimalidad bajo las cuales el problema se puede resolver de manera óptima en tiempo $O(n \log n)$.

Ya que la existencia de un algoritmo exacto que solucione el problema general de calendarización es poco probable, el desarrollo de algoritmos heurísticos y de aproximación es de gran importancia. Asimismo, consideramos la explosión combinatoria causada por el aumento del número de posibles combinaciones de entrada. Esta explosión en complejidad hace que algunos problemas matemáticos sean intratables por fuerza bruta. Se sabe que la fuerza bruta garantiza la obtención de la solución óptima pero en la práctica ésto no va a funcionar porque el número de soluciones factibles que se deben considerar para resolver un problema NP-duro crece exponencialmente con el tamaño de la entrada. Por lo tanto, el proceso de optimización requiere del uso de herramientas matemáticas avanzadas implementadas en algoritmos.

Abstract

Combinatorial (or discrete) optimization is one of the most active fields in the area of discrete mathematics. Combinatorial optimization problems occur in many diverse areas such as linear and integer programming, graph theory, artificial intelligence, and number theory. All these problems, when formulated mathematically as the minimization or maximization of a certain function defined on some domain, have a commonality of discreteness. Some of the most known combinatorial optimization problems are the subset sum problem, the travelling salesman problem, the bin packing problem, and the scheduling problem.

Single machine scheduling models constitute the core of scheduling problem research because they can be utilized to solve more complex scheduling models. For this reason, single machine scheduling research has attracted significant attention in the literature and, as a result, the publication of several papers focused not only on the solution method but also on the characteristics of the problem. The scheduling term refers to the assignment of jobs on a given set of resources over time fulfilling the optimality criteria.

This project studies a single-machine scheduling problem with a constant number of job release and two allowable delivery times. This is a particular case of the strongly NP-hard single machine scheduling problem with an arbitrary number of job release and delivery times whose aim is to minimize the maximum job completion time, therefore our particular problem still remains NP-hard. The study of this problem is useful for the solution of a more general setting, however, the restricted setting of the scheduling problem is useful on its own because it fits some real-life applications.

The study of the structural properties of the single-machine scheduling problem with a constant number of job release and two allowable delivery times was carried out in order to stablish optimality conditions under which the problem can be solved optimally in time $O(n \log n)$.

Since the existence of an exact solution algorithm for the general scheduling problem is highly unlikely, the development of heuristic and approximation algorithms is of great importance. We also consider the combinatorial explosion caused by the increasing the number of possible combinations of inputs. This explosion in complexity make some mathematical problems intractable to brute force solutions. It is known that brute force guarantees obtaining the optimal solution but in practice this will not work because the number of feasible solutions that must be considered to solve an NP-hard problem grows exponentially with the size of the input. Thus the optimization process requires the use of advanced mathematical tools implemented in algorithms.

A mis padres,
quienes lo saben todo y a la vez no tienen ni idea.

Y a mi hermana,
mi cómplice de travesuras.

Agradecimientos

A mi mamá. Eres la persona más fuerte que conozco. Gracias por estar conmigo y por todo lo que has hecho para que yo esté aquí.

A mi papá. Gracias por todo lo que me has enseñado, por confiar en mí y aún así estar cerca para ayudarme cuando lo necesito. Gracias por darme alas para volar.

A mi hermana. Seremos las primeras matemáticas en la familia, ojalá no seamos las últimas.

A Karla y Regina. Hemos crecido juntas desde pequeñas; tal vez no pudimos compartir esta etapa de nuestras vidas pero siempre estuvieron presentes. Estoy orgullosa de ustedes, y de lo que han podido lograr hasta ahora.

A Yehtli, Yatziry, Kevin, Vicente, Jesus, Alexandra, Coche y Arturo. Nuestros caminos serán diferentes a partir de ahora; no obstante, estoy segura de que harán de sus vidas una gran historia. Les deseo éxito en lo que se propongan.

A Samuel. Por su cariño y apoyo incondicional.

A la Dra. Elisa, el Dr. Daniel y a el Dr. Dan quienes dieron ese paso extra.

Al comité tutoral por su tiempo y dedicación.

Al Dr. Nodari. Mi maestro y asesor en la residencia de investigación.

A la Dr. Gabriela. Mi tutora durante toda la Licenciatura.

A cada uno de mis profesores. Gracias por todo su esfuerzo y todos los conocimientos que me han transmitido.

A la Dra. Larissa. Usted tenía razón, los matemáticos no dependen de laboratorios ni de salones de clase...

Al profesor LEF. Carlos Escobar. Usted me enseñó que el talento no sirve de nada sin compromiso ni trabajo duro.

A la Dra. Mesuma, que en paz descanse. Me hubiera gustado compartir este momento con usted.

A Clau. Un increíble ser humano y una mujer que inspira.

A Teo. Siempre estás dispuesto a escuchar y brindar consejo.

A Ricardo. Gracias por compartir tu pasión por las matemáticas y por detenerme cuando iba a elegir un camino que sabías que no me haría feliz y que no me permitiría ser la persona que soy ahora.

A mí... porque retomaste el camino y eres quien me motiva a ser una mejor persona.

Índice general

Capítulo 1

Introducción

1.1. Optimización combinatoria

Los problemas de optimización combinatoria están presentes en numerosos ámbitos de la vida cotidiana, desde la secuenciación de actividades en procesos de fabricación, redes de telecomunicaciones hasta en la secuenciación del ADN. El crecimiento y desarrollo acelerado del área de las metaheurísticas se debe a que una gran cantidad de problemas del mundo actual pueden formularse como problemas abstractos de optimización combinatoria. Estos problemas han sido de gran importancia desde los años cuarenta debido al crecimiento industrial y, con esto, la necesidad de establecer métodos eficientes que generen soluciones óptimas para los problemas de manejo de recursos y de logística.

Históricamente, la optimización combinatoria inicia con la programación lineal, la cual a su vez cuenta con una extensa lista de aplicaciones de gran importancia, incluyendo la planificación y distribución de producción, asignación de personal, finanzas, asignación de recursos económicos, simulación de circuitos y sistemas de control.

Un problema de optimización combinatoria cuenta con un conjunto finito de soluciones factibles, el cual está definido por un conjunto de restricciones. Una solución factible es aquella solución que satisface todas las restricciones de un problema. Normalmente hay una función objetivo cuyo dominio es el conjunto de soluciones factibles. Nuestro propósito es determinar cuál de las soluciones factibles maximiza (o minimiza, según sea el caso) la función objetivo, esta solución es conocida como la solución óptima. Existen dos clases de problemas de optimización combinatoria, los de clase P y los de clase NP-duro. Los problemas de clase P son aquellos para los cuales existe un algoritmo que encuentra la solución óptima en tiempo polinomial. Por otro lado, los problemas de clase NP-duro se definen de la siguiente manera:

Un problema p es NP-duro si

- Cualquier problema en NP puede reducirse a p en tiempo polinomial. Esto significa que los problemas en NP pueden transformarse en instancias de p, pero no necesariamente al revés.
- El número de soluciones factibles de p es exponencial y no existe un algoritmo polinomial que lo resuelva.

Si existiera un algoritmo polinomial que resuelva a p, entonces para todos los problemas NP-duro existe un algoritmo polinomial que los resuelve.

Es importante destacar que los problemas NP-duro no necesariamente forman parte de la clase NP. Así que un problema NP-duro no necesariamente tiene una solución factible verificable en tiempo polinomial. Los problemas NP-duro son de gran importancia en la teoría de la computación y la ciencia de la computación, ya que proporcionan una idea de la dificultad de los problemas dentro de la clase NP y su relación con la clase de problemas P.

La explosión combinatoria del número de soluciones de un problema de optimización combinatoria de clase NP-duro hace impracticable su enumeración, es decir, generar todas las soluciones y elegir la mejor. Es por esta razón que el proceso de optimización requiere del uso de herramientas matemáticas avanzadas implementadas en algoritmos.

1.2. Problemas de calendarización

Los problemas de calendarización son importantes problemas de optimización combinatoria y pertenecen a la clase NP-duro (Garey and Johnson [1]). Un problema de calendarización se compone de cuatro partes principales:

- Los trabajos que se deben realizar. Cada trabajo puede tener diferentes parámetros como tiempo de procesamiento, tiempo de liberación, tiempo de entrega, fecha límite, etc.
- Los recursos disponibles para su implementación, los cuales llamamos *máquinas*.
- El conjunto de restricciones que definen el conjunto de soluciones factibles.
- Los criterios de optimalidad, los cuales identifican la solución óptima.

El objetivo de este problema es elegir el orden de procesamiento de los trabajos sobre las máquinas disponibles tal que se cumple el criterio de optimalidad dado. Los trabajos pueden tener distintos parámetros, el número de máquinas disponibles y los criterios de optimalidad pueden ser diferentes para cada problema; por lo tanto, existe una enorme cantidad de problemas de calendarización.

El problema de calendarización de esta investigación está definido de la siguiente manera. Dado un conjunto $\mathbf{J} = \{j_1, j_2, ..., j_n\}$ de n trabajos con **tiempos de liberación**, **tiempos de procesamiento** y **tiempos de entrega**, los trabajos de **J** deben ser calendarizados en una sola máquina. Además, se cumplen las siguientes restricciones:

- El trabajo j no se puede calendarizar antes de su tiempo de liberación r_j
- El trabajo j necesita un tiempo de procesamiento continuo p_j en la máquina
- Una vez completado el trabajo j en la máquina, necesita un tiempo de entrega adicional q_j para su completés total (la entrega del trabajo j es independiente de la máquina y no requiere más recursos, ya que lo realiza un agente independiente)
- La máquina solamente procesa un trabajo a la vez

Un calendario factible σ asigna cada trabajo j a la máquina en el intervalo de tiempo $[t_j(\sigma), t_j(\sigma) + p_j)$ en el intervalo $[0, \infty)$, de tal forma que $t_j(\sigma) \geq r_j$ y este intervalo no tiene intersección con el intervalo de ningún otro trabajo (por esa razón consideramos intervalos de tiempo medio abiertos), donde $t_j(\sigma)$ es el **tiempo de inicio del trabajo j** en el calendario σ y $c_j(\sigma) = t_j(\sigma) + p_j$ es el **tiempo de completés del trabajo j**. El **tiempo de completés total del trabajo j** en el calendario σ está dado por $C_j(\sigma) = c_j(\sigma) + q_j$.

Para un calendario factible σ, el ***makespan*** $C_{max}(\sigma) = max\{C_j(\sigma)\}$ es el máximo tiempo de completés total de todos los trabajos en el conjunto **J**, es decir, la longitud total del calendario [2]. El objetivo es encontrar un calendario óptimo σ_{opt}, el cual está dado por un calendario factible con el mínimo tiempo de completés total $C_{max}(\sigma_{opt}) = min_\sigma \{C_{max}(\sigma)\}$.

Una solución que no es factible para este problema sería una permutación en la cual se asignan trabajos en el calendario que aún no han sido liberados.

De acuerdo con la notación introducida por Graham [3], el problema bajo investigación es abreviado como

$$1|r_i \in \left\{r^1, ..., r^k\right\}, q_i \in \{q^1, q^2\} |C_{max}$$

El primer campo indica el número de máquinas disponibles. El segundo campo especifica los parámetros de los trabajos, en este caso consideramos un número constante de tiempos de liberación $r^1, ..., r^k$ $(r^1 < ... < r^k)$ y solo dos posibles tiempos de entrega q^1 y q^2 $(q^1 < q^2)$. Y, por último, el tercer campo indica el criterio de optimalidad.

Cada asignación específica de valores a los parámetros se denomina ***instancia*** del problema; por tanto, se define a través del conjunto de parámetros. A continuación se da un ejemplo de una instancia del problema

$$1|r_i \in \{r^1, r^2, r^3, r^4\}, q_i \in \{q^1, q^2\} |C_{max}$$

$r_1 = 0$	$p_1 = 2$	$q_1 = 3$
$r_2 = 0$	$p_2 = 3$	$q_2 = 7$
$r_3 = 0$	$p_3 = 1$	$q_3 = 3$
$r_4 = 6$	$p_4 = 3$	$q_4 = 7$
$r_5 = 8$	$p_5 = 5$	$q_5 = 7$
$r_6 = 11$	$p_6 = 1$	$q_6 = 3$
$r_7 = 11$	$p_7 = 2$	$q_7 = 3$

1.3. Posibles aplicaciones

El problema $1|r_i \in \left\{r^1, ..., r^k\right\}, q_i \in \{q^1, q^2\} |C_{max}$ puede tener diferentes aplicaciones en la vida cotidiana. Consideremos el siguiente ejemplo. Tres proveedores diferentes entregan piezas a una pequeña fábrica, la cual cuenta con una sola máquina de ensamblaje. Las piezas del proveedor 1, proveedor 2 y proveedor 3 son entregadas al tiempo r^1, r^2 y r^3, respectivamente.

Hay tres clientes a los que se les debe entregar las piezas ensambladas y terminadas. El tiempo de entrega de los clientes 1 y 3 es q^1, mientras que el tiempo de entrega del cliente 2 es q^2. El objetivo es minimizar el tiempo en el que todas las piezas terminadas son entregadas a los tres clientes o, equivalentemente, encontrar un calendario factible σ que minimice el tiempo máximo de completés total. Este problema no se restringe a tres proveedores y a tres clientes, es posible tener i_1 proveedores con $r^1, ..., r^k$ tiempos de liberación diferentes y también i_2 clientes con dos tiempos diferentes de entrega.

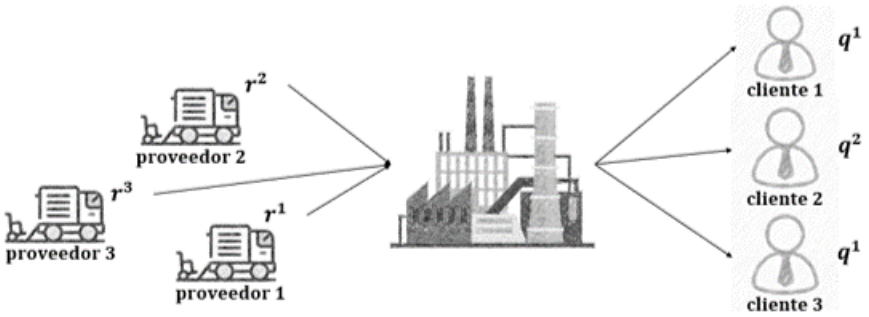


Figura 1.1: Posible aplicación

Ahora abordaremos un problema de calendarización de la vida real descrito en [14], el cual dividiremos con la finalidad de establecer una posible aplicación del problema $1|r_i \in \left\{r^1, ..., r^k\right\}, q_i \in \{q^1, q^2\}|C_{max}$.

Una empresa escocesa debe producir todos los días horarios para recolectar y transportar un gran número de pollos vivos. El problema no solo es complejo sino que también cuenta con numerosas restricciones. La empresa recolecta hasta 1.3 millones de aves vivas por semana en granjas instaladas por toda Escocia y las procesa en dos fábricas. El objetivo es programar la recolección y la entrega de aves utilizando un recurso fijo de brigadas y camiones para mantener las fábricas suministradas con aves a un ritmo constante. Cada día, un conjunto de órdenes debe ser procesado en cada una de las dos fábricas. Cada fábrica opera su propia flota de camiones, los cuales están disponibles en dos tamaños diferentes y poseen una capacidad de 22 o 24 módulos para aves. Ciertos tipos de aves necesitan un camión más grande para su transporte, mientras que otros pueden caber en cualquier tamaño. Aunque un camión puede entregar su carga a cualquiera de las fábricas, éste debe entregar su última carga por la noche a la fábrica donde está su base.

Una *órden* consiste en m módulos para aves de tipo t y peso w que deben ser recolectados de una granja f y entregados a una fábrica F. El trabajo definido por las órdenes debe ser asignado a los equipos de las brigadas de recolección. Después los camiones deben ser programados para recibir la carga de cada granja y entregarlas a la fábrica. Las fábricas deben ser abastecidas continuamente durante el día; no obstante, hay regulaciones estrictas que rigen el periodo de tiempo en el cual las aves pueden permanecer fuera de la fábrica antes de ser procesadas. Por lo tanto, las aves deben llegar dentro del lapso de tiempo establecido en que la fábrica puede procesarlas.

En la figura 1.2 se ilustra un ejemplo de este proceso para una fábrica. La *brigada* deja su base en la fábrica al comienzo de su turno, viaja entre un conjunto de granjas, recolecta un número variable de aves de cada granja y regresa a su base una vez que terminó su turno.

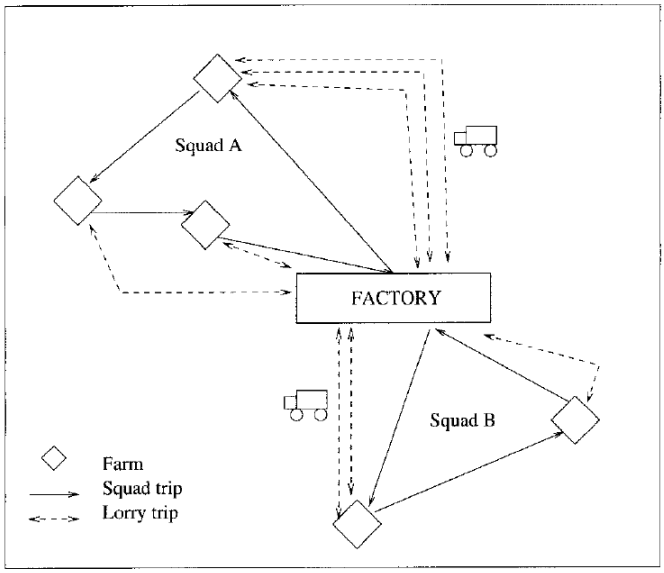


Figura 1.2: Ilustración del programa de trabajo de dos brigadas de recolección para una fábrica. Imagen extraída de [14]

Una *brigada* tiene su base en una de las fábricas y trabaja ya sea medio tiempo o tiempo completo. La *brigada* rota entre tres turnos, clasificados como *temprano*, *flotante* o *tarde*, dependiendo de la hora del día cuando el turno comienza. Existen condiciones contractuales que especifican el mínimo y el máximo número de módulos que una *brigada* puede recolectar en un día y en una semana, además de la duración máxima de los turnos que pueden trabajar. Algunas granjas tienen sus propias brigadas de recolección y sus propias restricciones en el orden en el cual las brigadas pueden visitar ciertas granjas debido a la prevención de propagación de enfermedades existentes en algunas de ellas.

Los conductores también deben ser asignados a cada viaje, un conductor de camión puede trabajar un turno de 13 horas máximo de las cuales solo pueden conducir un total de 9 horas. Si es necesario, se les puede pagar para realizar trabajo extra aunque, por supuesto, es preferible evitar esta situación.

Es posible generar horarios prácticos y de calidad para los conductores que satisfacen las restricciones operativas de modo que se minimiza el tiempo en el que todas las órdenes son procesadas.

Suponiendo que a un conductor se le asigna un conjunto de granjas (como a las brigadas) se tiene lo siguiente:

- Los tiempos de liberación r_i representan el momento en que el conductor puede recoger la carga de una granja i pues la brigada terminó su trabajo de recolección. Dos o más granjas pueden tener el mismo tiempo de liberación debido a que algunas granjas tienen sus propias brigadas.
- Sean los tiempos de procesamiento p_i el tiempo de viaje desde la granja i hasta la fábrica. Además p_i cumple con el periodo de tiempo en el cual las aves pueden permanecer fuera de la fábrica.
- Los tiempos de entrega q^1 y q^2 representan el tiempo que tarda cada fábrica en procesar las aves.

1.4. Algoritmos heurísticos

Anteriormente se mencionó que los problemas de clase P son aquellos para los cuales existe un algoritmo que los resuelve en tiempo polinomial y, además, que es poco probable que existan tales algoritmos para resolver un problema NP-duro; es por esta razón que se empezaron a desarrollar algoritmos que resuelvan estos problemas de forma aproximada. Los algoritmos heurísticos son de los algoritmos aproximados más simples, estos algoritmos generan una o más soluciones factibles en tiempo polinomial y son utilizados usualmente para los problemas que no se pueden resolver fácilmente; sin embargo, no garantizan generar la solución óptima.

1.5. Estado del arte

El problema $1|r_i, q_i|C_{max}$ es uno de los problemas más estudiados en la literatura de calendarización, por lo cual se ha recurrido a diferentes algoritmos heurísticos para lograr un mejor enfoque. La heurística LDT (Largest Delivery Time) genera una solución de dos aproximaciones (una solución que a lo sumo es dos veces peor que una solución óptima). Esta heurística fue descrita por primera vez por Schrage [4] en 1971 y es una extensión del algoritmo voraz antes propuesto por Jackson [5] para el caso sin tiempos de liberación.

Varios autores se han enfocado en identificar los casos para los cuales el problema $1|r_i, q_i|C_{max}$ se soluciona en tiempo polinomial. Jackson y Smith en 1955 y 1956 respectivamente, fueron los primeros en enfocarse en el estudio de este problema.

Jackson en [6] demostró que el problema se soluciona en forma óptima en tiempo $O(n \log n)$ cuando los tiempos de liberación de los trabajos son iguales y los trabajos son calendarizados en orden decreciente con respecto a sus tiempos de entrega. Similarmente, se obtiene un calendario óptimo si todos los tiempos de entrega son iguales y los trabajos son calendarizados en orden decreciente con respecto a sus tiempos de liberación. Además se tiene que si el tiempo de procesamiento de todos los trabajos es igual a 1, entonces la heurística de [4] encuentra la solución óptima.

En 1978, Dessouky y Larson en [5] mostraron que cuando se cumple alguna de las restricciones, dadas a continuación, para todos los pares de trabajos i y j entonces el problema se resuelve en tiempo polinomial:

- $r_i \geq r_j + p_j$ o $r_j \geq r_i + p_i$
- $q_i \geq q_j + p_j$ o $q_j \geq q_i + p_i$
- $max_j \{r_j\} < min_j \{r_j + p_j\}$ y $max_j \{q_j\} < min_j \{q_j + p_j\}$

En 2016, Vakhania en [7] desarrolló un algoritmo polinomial que genera una solución óptima cuando los tiempos de procesamiento son mutuamente divisibles.

Más tarde en 2021, Reynoso y Vakhania en [8] llevaron a cabo el estudio del problema de calendarización para una sola máquina con dos tiempos de liberación y dos tiempos de entrega permitidos. Se realizó un análisis profundo y se obtuvo una serie de condiciones de optimalidad para el problema $1|r_i \in \{r^1, r^2\}, q_i \in \{q^1, q^2\}|C_{max}$, las cuales se propuso que se pueden generalizar a un número constante de tiempos de liberación y tiempos de entrega. Los criterios de optimalidad que se establecieron son condiciones explícitas bajo las cuales, el problema restringido se puede resolver de manera óptima en tiempo $O(n \log n)$. Normalmente se puede esperar que, para un problema NP-duro, tales condiciones deberían ser muy restrictivas; sin embargo, el extenso estudio computacional que se llevó a cabo demostró que estas condiciones, utilizadas de manera combinada, cubrieron prácticamente todos los casos problemáticos que pudieron surgir en la práctica.

Capítulo 2

Las heurísticas

Las heurísticas propuestas en este proyecto son una variación de la heurística LDT estándar, la cual consiste en calendarizar en cada iteración el trabajo con el mayor tiempo de entrega de entre los trabajos que aún están disponibles. En este capítulo se describen las heurísticas; no obstante, primero introducimos algunas notaciones que nos serán útiles.

2.1. Notación

Denotamos por $J(r^i, q^l)$ el conjunto de trabajos con tiempos de liberación y entrega r^i y q^l, respectivamente, con $i = 1, ..., k$ y $l = 1, 2$. Recordemos que $r^1 < ... < r^k$ y que $q^2 > q^1$. Sea $J(r^i)$ el conjunto de trabajos con tiempo de liberación r^i y $J(q^l)$ el conjunto de trabajos con tiempo de entrega q^l.

Dado un conjunto de trabajos A, definimos $P(A)$ como la suma de los tiempos de procesamiento de todos los trabajos del conjunto A.

$$P(A) = \sum_{j \in A} p_j$$

Sin pérdida de generalidad, asumimos que $J(r^i, q^1) \neq \emptyset$ con $1 \leq i \leq k-1$ y $J(r^i, q^2) \neq \emptyset$ con $2 \leq i \leq k$ ya que, en caso contrario, el calendario generado por la heurística LDT es óptimo (ver Observación 1 en el siguiente capítulo).

Por otro lado, asumiremos que dada una instancia los trabajos de ésta ya se encuentran en orden creciente con respecto a sus tiempos de liberación y, a la vez, en orden decreciente con respecto a sus tiempos de entrega antes de aplicar cualquiera de las heuríticas.

2.2. Heurística LDT

Comenzamos con la descripción detallada de la heurística LDT estándar, la cual procede con n asignaciones. El tiempo inicial de calendarización t es el mínimo tiempo de liberación r^1. El siguiente tiempo de calendarización es actualizado de manera iterativa de la siguiente manera. A partir del tiempo r^1 los trabajos del conjunto $J(r^1, q^2)$ son asignados de forma continua en el siguiente tiempo de calendarización, el cual es determinado por la suma del tiempo de

calendarización actual y el tiempo de procesamiento del trabajo asignado. Una vez asignados todos los trabajos del conjunto $J(r^1, q^2)$ se asignará iterativamente el trabajo del conjunto $J(r^1, q^1)$ con el mayor tiempo de procesamiento de tal manera que el último trabajo asignado de este conjunto comience su procesamiento antes del tiempo r^2 y sea completado después o al tiempo r^2. En caso de haber trabajos del conjunto $J(r^1, q^1)$ que no fueron asigandos, éstos serán incluidos en el conjunto $J(r^2, q^1)$. Una vez que el último trabajo asignado del conjunto $J(r^1, q^1)$ concluyó su procesamiento, se asignan inmediatamente los trabajos del conjunto $J(r^2, q^2)$. Nótese que si todos los trabajos del conjunto $J(r^1)$ terminan su procesamiento antes del tiempo r^2, entonces el primer trabajo asignado del conjunto $J(r^2, q^2)$ inicia su procesamiento al tiempo r^2. Después se asignan los trabajos del conjunto $J(r^2, q^1)$ siguiendo las mismas condiciones de antes, en esta parte el calendario puede contener trabajos tanto del conjunto $J(r^2, q^1)$ como trabajos del conjunto $J(r^1, q^1)$. Decimos que el trabajo asignado $j \in J(r^i, q^2) \cup J(r^i, q^1)$ **"empuja"** al trabajo asignado $j+1 \in J(r^{i+1}, q^2) \cup J(r^{i+1}, q^1)$ cuando $t_j(\sigma) < r^{i+1} < c_j(\sigma)$. Este procedimiento se repite de la misma manera para los siguientes tiempos de liberación $r^3, ..., r^k$, respectivamente. Esta heurística se ejecuta en tiempo $O(n \log n)$ pues en cada uno de los n tiempos de calendarización la búsqueda de un elemento máximo de una lista ordenada es llevada a cabo.

Dado que la heurística LDT siempre asigna un trabajo liberado lo más pronto posible cada vez que la máquina queda inactiva, ésta no crea ningún *hueco* que se pueda evitar.

El pseudocódigo a nivel de implementación es el siguiente:

ALGORITHM LDT { returns LDT-schedule σ^S }
    $t := r^1$;
    $i := 1$;
    $l := 2$;
    WHILE ($J(q^1) \neq \emptyset$ and $J(q^2) \neq \emptyset$) DO { schedule jobs }
        IF $l = 2$ THEN:
            WHILE ($J(r^i, q^l) \neq \emptyset$) DO { schedule jobs of set $J(r^i, q^2)$ }
                schedule any $j \in J(r^i, q^l)$ at time $t_j(\sigma^S) = t$;
                $J(r^i, q^l) := J(r^i, q^l) \setminus \{j\}$;
                $t = t + p_j$
            END
            $l := 1$
        ELSE:
            WHILE ($J(r^i, q^l) \neq \emptyset$) DO { schedule jobs of set $J(r^i, q^1)$ }
                IF $r^i < r^k$ THEN:
                    Find job $j \in J(r^i, q^l)$ with the largest processing time p_j
                    IF $t < r^{i+1}$ THEN:
                        schedule j at time $t_j(\sigma^S) = t$;
                        $J(r^i, q^l) := J(r^i, q^l) \setminus \{j\}$;
                        $t = t + p_j$
                    ELSE:
                        $J(r^i, q^l) := J(r^i, q^l) \setminus \{j\}$;

$J(r^{i+1}, q^l) := J(r^{i+1}, q^l) \cup \{j\}$;

ELSE: { schedule the remaining jobs of set $J(r^k, q^1)$ }

schedule any $j \in J(r^i, q^l)$ at time $t_j(\sigma^S) = t$

$J(r^i, q^l) := J(r^i, q^l) \setminus \{j\}$

$t = t + p_j$

END

IF $r^{i+1} < r^k$ THEN:

$t := max\left\{t, r^{i+1}\right\}$ { $t < r^{i+1}$ only if all jobs from $J(r^i)$ are scheduled }

$i := i + 1$

$l := 2$

ELSE:

$t := max\left\{t, r^k\right\}$ { $t < r^k$ only if all jobs from $J(r^{k-1})$ are scheduled }

$i := k$

$l := 2$

END

RETURN σ^S.

2.2.1. Ejemplos

Consideremos las siguientes instancias del problema $1|r_i \in \left\{r^1, ..., r^k\right\}, q_i \in \left\{q^1, q^2\right\}|C_{max}$.

Comencemos primero con una instancia para el problema $1|r_i \in \left\{r^1, r^2, r^3, r^4\right\}, q_i \in \left\{q^1, q^2\right\}|C_{max}$, donde $r^1 = 0,\ r^2 = 4,\ r^3 = 13,\ r^4 = 19,\ q^1 = 3$ y $q^2 = 5$.

$r_1 = 0$	$p_1 = 2$	$q_1 = 5$
$r_2 = 0$	$p_2 = 1$	$q_2 = 5$
$r_3 = 0$	$p_3 = 3$	$q_3 = 3$
$r_4 = 4$	$p_4 = 3$	$q_4 = 5$
$r_5 = 4$	$p_5 = 2$	$q_5 = 5$
$r_6 = 4$	$p_6 = 4$	$q_6 = 3$
$r_7 = 13$	$p_7 = 3$	$q_7 = 3$
$r_8 = 13$	$p_8 = 2$	$q_8 = 3$
$r_9 = 19$	$p_9 = 6$	$q_9 = 5$
$r_{10} = 19$	$p_{10} = 3$	$q_{10} = 3$

Entonces se forman los siguientes conjuntos:

$J(r^1, q^2) = \{j_1, j_2\}$	$J(r^2, q^2) = \{j_4, j_5\}$	$J(r^3, q^2) = \emptyset$	$J(r^4, q^2) = \{j_9\}$
$J(r^1, q^1) = \{j_3\}$	$J(r^2, q^1) = \{j_6\}$	$J(r^3, q^1) = \{j_7, j_8\}$	$J(r^4, q^1) = \{j_{10}\}$

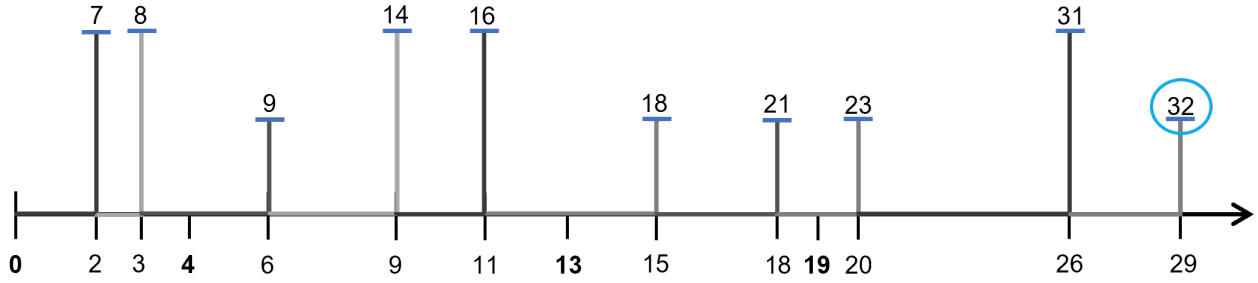


Figura 2.1: Calendario σ^S

En la figura 2.1 se muestra una instancia con 10 trabajos y el calendario σ^S generado por la heurística LDT. Las líneas verticales representan los tiempos de entrega de cada trabajo. Se observa que el tiempo de liberación mínimo de los trabajos es 0, entonces la heurítica LDT asigna los trabajos con mayor tiempo de entrega liberados al tiempo $r^1 = 0$ de forma continua y en cualquier orden. En este ejemplo la heurística asigna el trabajo 1 primero. Ahora $t = 3$ una vez asignados los trabajos 1 y 2, $t < r^2$ entonces el trabajo 3 es asignado en el calendario. Los trabajos del conjunto $J(4, 5)$ son asignados inmediatamente al igual que el trabajo 6 que pertenece al conjunto $J(4, 3)$ ya que $t = 11$ y $t < r^3$. Observemos que los trabajos 3 y 6 empujan a los trabajos 4 y 7, respectivamente. Ya que $J(13, 5) = \emptyset$, los trabajos 7 y 8 son asignados en este orden pues $p_7 > p_8$. Finalmente se asignan los trabajos 9 y 10, siendo el *makespan* igual a 32.

Veamos otro ejemplo con 8 trabajos.

$r_1 = 1$	$p_1 = 3$	$q_1 = 3$
$r_2 = 1$	$p_2 = 1$	$q_2 = 3$
$r_3 = 1$	$p_3 = 2$	$q_3 = 3$
$r_4 = 1$	$p_5 = 4$	$q_5 = 1$
$r_5 = 6$	$p_5 = 7$	$q_5 = 3$
$r_6 = 6$	$p_6 = 2$	$q_6 = 1$
$r_7 = 20$	$p_7 = 1$	$q_7 = 3$
$r_8 = 20$	$p_8 = 5$	$q_8 = 1$

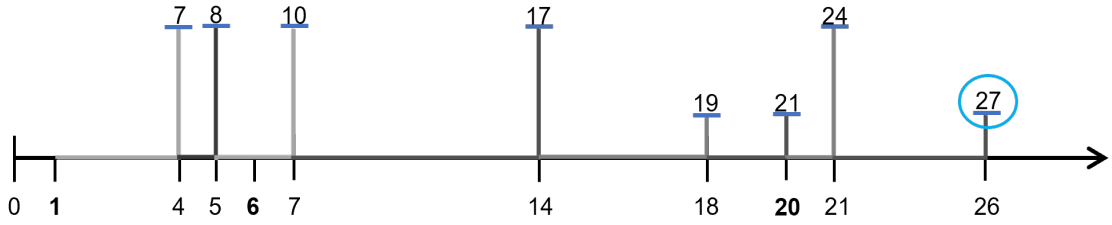


Figura 2.2: Calendario σ^S

En la figura 2.2 se muestra una instancia con 8 trabajos y el calendario generado σ^S para esta instancia. Ya que $t_3(\sigma^S) + p_3 > r^2$, la heurística LDT incluye el trabajo 4 en el conjunto $J(6, 1)$. En este ejemplo también es posible observar que el trabajo 7 inicia su procesamiento en su tiempo de liberación, es decir, $t_7(\sigma^S) = r_7$.

Ahora veamos un ejemplo donde la heurística LDT genera un hueco que no se puede evitar.

$r_1 = 0$	$p_1 = 2$	$q_1 = 4$
$r_2 = 0$	$p_2 = 5$	$q_2 = 1$
$r_3 = 5$	$p_3 = 3$	$q_3 = 4$
$r_4 = 5$	$p_4 = 2$	$q_4 = 1$
$r_5 = 15$	$p_5 = 3$	$q_5 = 4$

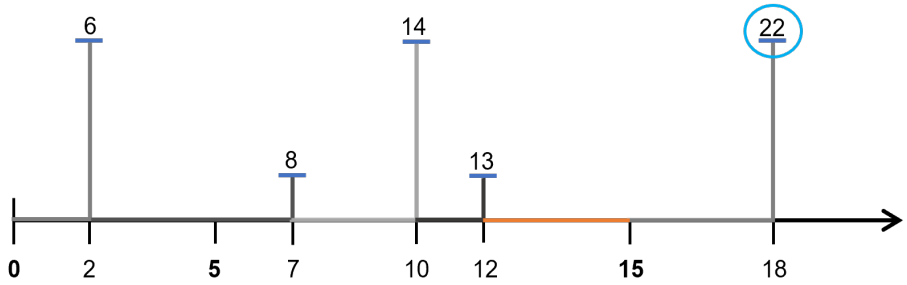


Figura 2.3: Calendario σ^S

En la figura 2.3 se muestra una instancia con 5 trabajos y el calendario σ^S para esta instancia. Observamos que el calendario σ^S contiene un hueco dado por el intervalo $[12, 15]$.

La siguiente variación de la heurística LDT presenta partes en común; las diferencias entre ellas surgen básicamente en la forma en cómo se asignan los trabajos alrededor de cada tiempo de liberación $r^2, ..., r^k$. Primero especificamos las partes en las que son similares y dónde se producen las diferencias.

Las dos heurísticas consisten en k fases, siendo k el número de tiempos de liberación de la instancia dada. Cada fase tiene dos partes, la primera parte asigna los trabajos del conjunto $J(q^2)$, y la segunda parte asigna los trabajos del conjunto $J(q^1)$ y algunos de los trabajos que no fueron asignados en la fase anterior con tiempo de entrega q^1. La primera parte de cada fase es la misma para ambas heurísticas. Inician con la primera parte de la fase uno asignando continuamente los trabajos del conjunto $J(r^1, q^2)$ apartir del tiempo r^1 en cualquier orden pero sin crear ningún hueco entre ellos. En seguida, algunos trabajos del conjunto $J(r^1, q^1)$ son asignados consecutivamente hasta dejar un pequeño hueco antes del tiempo r^2; de esta forma un solo trabajo del conjunto $J(r^1, q^1)$ termina su procesamiento antes o después del tiempo r^2, o al tiempo r^2. Lo mismo se repite en cada fase; por tanto, en cada fase i se pueden presentar los siguientes casos (dependiendo de la heurística aplicada):

- $r^i + P(J(r^i, q^2)) + P(J(r^i, q^1)) < r^{i+1}$
- $r^i + P(J(r^i, q^2)) + P(J(r^i, q^1)) = r^{i+1}$
- $r^i + P(J(r^i, q^2)) + P(J(r^i, q^1)) > r^{i+1}$

Cada uno corresponde a un hueco, un hueco de longitud cero y un empuje, respectivamente.

2.3. Heurística LDT-G

La primera variante es la heurística LDT-G, la cual crea huecos (posiblemente de longitud cero) antes de cada tiempo de liberación r^i. Sea $L_i = (j_1, ..., j_l)$ con $1 \leq i \leq k$ una lista con los trabajos del conjunto $J(r^i, q^1)$ ordenados de forma decreciente con respecto a sus tiempos de procesamiento. En cada iteración, si el siguiente trabajo $j \in L_i$ cumple que $t + p_j \leq r^{i+1}$, entonces j es asignado al tiempo t y se actualiza $t = t + p_j$. En caso contrario, el trabajo j es reubicado al conjunto $J(r^{i+1}, q^1)$ y el siguiente trabajo (el siguiente con mayor tiempo de procesamiento) en L_i se verifica de manera similar hasta que se consideren todos los trabajos de la lista L_i.

ALGORITHM LDT-G {returns LDTG-schedule σ^G}
$t := r^1$;
$i := 1$;
$l := 2$;
WHILE ($J(q^1) \neq \emptyset$ and $J(q^2) \neq \emptyset$) DO { schedule jobs }
IF $l = 2$ THEN:
WHILE ($J(r^i, q^l) \neq \emptyset$) DO { schedule jobs of set $J(r^i, q^2)$ }
schedule any $j \in J(r^i, q^l)$ at time $t_j(\sigma^G) = t$;
$J(r^i, q^l) := J(r^i, q^l) \setminus \{j\}$;
$t = t + p_j$
END
$l := 1$
ELSE:
WHILE ($J(r^i, q^l) \neq \emptyset$) DO { schedule jobs of set $J(r^i, q^1)$ }
IF $r^i < r^k$ THEN:
Find job $j \in J(r^i, q^l)$ with the largest processing time p_j
IF $t + p_j \leq r^{i+1}$ THEN:
schedule j at time $t_j(\sigma^G) = t$;
$J(r^i, q^l) := J(r^i, q^l) \setminus \{j\}$;
$t = t + p_j$
ELSE:
$J(r^i, q^l) := J(r^i, q^l) \setminus \{j\}$;
$J(r^{i+1}, q^l) := J(r^{i+1}, q^l) \cup \{j\}$;
ELSE: { schedule the remaining jobs of set $J(r^k, q^1)$ }
schedule any $j \in J(r^i, q^l)$ at time $t_j(\sigma^G) = t$
$J(r^i, q^l) := J(r^i, q^l) \setminus \{j\}$
$t = t + p_j$
END
IF $r^{i+1} < r^k$ THEN:
$t := max\{t, r^{i+1}\}$ { $t < r^{i+1}$ only if all jobs from $J(r^i)$ are scheduled }
$i := i + 1$

$l := 2$

ELSE:

$t := max\{t, r^k\}$ { $t < r^k$ only if all jobs from $J(r^{k-1})$ are scheduled }

$i := k$

$l := 2$

END

RETURN σ^G.

2.3.1. Ejemplos

Consideramos las instancias del ejemplo anterior, entonces los calendarios generados por la heurística LDT-G son los siguientes.

$r_1 = 0$	$p_1 = 2$	$q_1 = 5$
$r_2 = 0$	$p_2 = 1$	$q_2 = 5$
$r_3 = 0$	$p_3 = 3$	$q_3 = 3$
$r_4 = 4$	$p_4 = 3$	$q_4 = 5$
$r_5 = 4$	$p_5 = 2$	$q_5 = 5$
$r_6 = 4$	$p_6 = 4$	$q_6 = 3$
$r_7 = 13$	$p_7 = 3$	$q_7 = 3$
$r_8 = 13$	$p_8 = 2$	$q_8 = 3$
$r_9 = 19$	$p_9 = 6$	$q_9 = 5$
$r_{10} = 19$	$p_{10} = 3$	$q_{10} = 3$

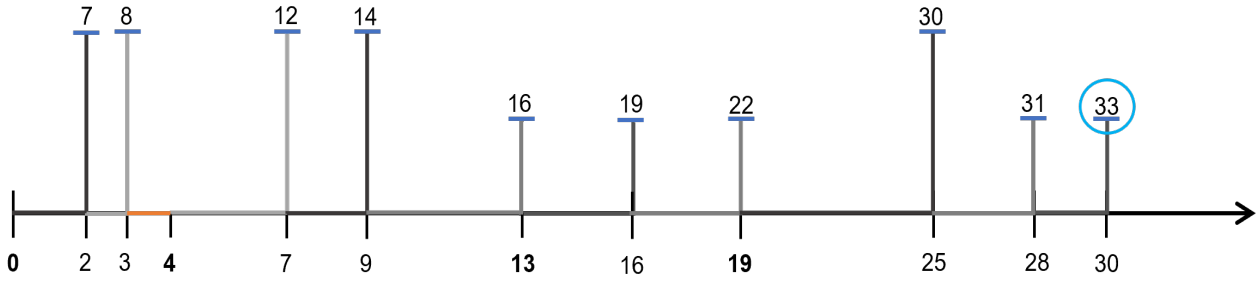


Figura 2.4: Calendario σ^G

En la figura 2.4 se observa que la heurítica LDT-G asigna los trabajos con mayor tiempo de entrega liberados al tiempo $r^1 = 0$ de forma continua y en cualquier orden. Para el trabajo 3 se verifica que $t + p_3 > r^2$ entonces el trabajo 3 se incluye en el conjunto $J(4, 3)$ y al no haber más trabajos en el conjunto $J(0, 3)$ la heurística crea un hueco dado por el intervalo $[3, 4]$; ahora $t = r^2$. Inmediatamente se asignan los trabajos del conjunto $J(4, 5)$ y el trabajo 6 ya que $p_6 > p_3$ y $t + p_6 = r^3$. El trabajo 3 pasa a ser incluido en el conjunto $J(13, 3)$. Se asignan los trabajos 7 y 3; el trabajo 8 es incluido al conjunto $J(19, 3)$ ya que $t + p_8 > r^4$. Finalmente el resto de los trabajos son asignados. Nótese que los trabajos 4, 7 y 9 inician su procesamiento en su tiempo de liberación; sin embargo, el *makespan* es igual a 33.

$r_1 = 1$	$p_1 = 3$	$q_1 = 3$
$r_2 = 1$	$p_2 = 1$	$q_2 = 3$
$r_3 = 1$	$p_3 = 2$	$q_3 = 3$
$r_4 = 1$	$p_5 = 4$	$q_5 = 1$
$r_5 = 6$	$p_5 = 7$	$q_5 = 3$
$r_6 = 6$	$p_6 = 2$	$q_6 = 1$
$r_7 = 20$	$p_7 = 1$	$q_7 = 3$
$r_8 = 20$	$p_8 = 5$	$q_8 = 1$

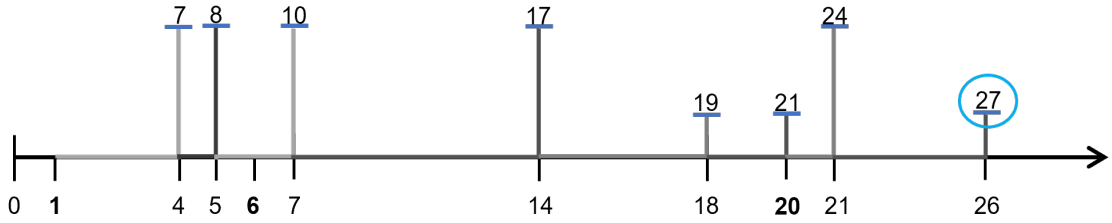


Figura 2.5: Calendario σ^G

En la figura 2.5 se observa que las heurísticas LDT y LDT-G generaron el mismo calendario. El trabajo 6 cumple $t_6(\sigma^G) + p_6 = r^3$; por lo tanto, la heurística LDT-G creó un hueco de longitud cero.

$r_1 = 0$	$p_1 = 2$	$q_1 = 4$
$r_2 = 0$	$p_2 = 5$	$q_2 = 1$
$r_3 = 5$	$p_3 = 3$	$q_3 = 4$
$r_4 = 5$	$p_4 = 2$	$q_4 = 1$
$r_5 = 15$	$p_5 = 3$	$q_5 = 4$

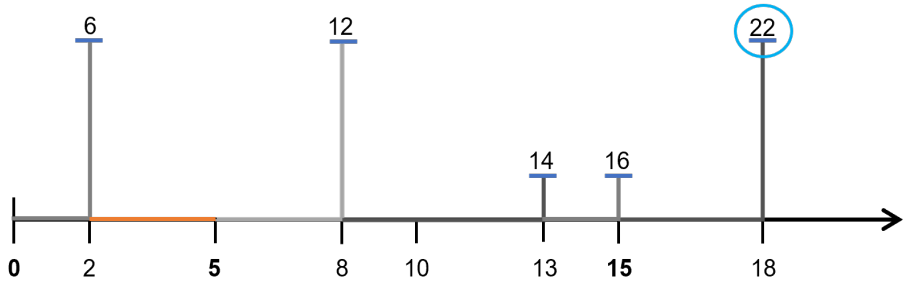


Figura 2.6: Calendario σ^G

En la figura 2.5 se muestra que el calendario σ^G contiene un hueco dado por el intervalo $[2, 5]$. Este calendario y el calendario generado por la heurística LDT son semejantes ya que ambos contienen un hueco de longitud 3 y el *makespan* es igual a 22.

Denotamos por σ^S y σ^G los calendarios generados por la heurística LDT y la heurística LDT-G, respectivamente. La complejidad temporal de ambas heurísticas descritas anteriormente es $O(n \log n)$.

Capítulo 3

Conceptos y propiedades básicas

3.1. Conceptos básicos y sus definiciones

A continuación definimos algunos conceptos auxiliares que son útiles para comprender la estructura básica de los calendarios generados por las heurísticas. Estos conceptos se introdujeron en [9] y [10], aunque algunos antes se conocían por otro nombre.

Como se mencionó anteriormente, un **hueco** es un intervalo de tiempo en el cual ningún trabajo es procesado por la máquina. Un hueco de longitud cero ocurre siempre que el tiempo de procesamiento del trabajo i empieza inmediatamente después del tiempo de completés del trabajo j, es decir, $t_i(\sigma) = c_j(\sigma)$.

Un **bloque** es la secuencia más larga de trabajos calendarizados consecutivamente sin ningún hueco entre ellos. También se marca un bloque si el trabajo j inicia su procesamiento al tiempo r_j, es decir, $t_j(\sigma) = r_j$.

El ***overflow job*** es el trabajo que tiene el tiempo de completés total máximo en σ y se denota por $o(\sigma)$. El bloque crítico de σ, $B(\sigma)$, es aquel que contiene a $o(\sigma)$.

Un trabajo e es llamado **trabajo emergente** en el calendario σ si $e \in B(\sigma)$ y $q_e < q_{o(\sigma)}$. El último trabajo emergente calendarizado antes del *overflow job* $o(\sigma)$ es llamado **activo** y es denotado por e'.

El ***kernel*** o **núcleo** de σ, $K(\sigma)$, es la secuencia de trabajos calendarizados en σ que están entre el trabajo emergente activo e' y el *overflow job* $o(\sigma)$, sin incluir a e' pero que incluye a $o(\sigma)$. Un *kernel* se conoce como una secuencia crítica. El tiempo de entrega de cualquier trabajo en $K(\sigma)$ no es menor que el tiempo de entrega del trabajo $o(\sigma)$. Si el calendario σ no tiene trabajos emergentes, entonces tampoco tendrá *kernel*; entonces el *overflow job* en el calendario σ no será parte de ningún *kernel*.

A continuación se ilustra los conceptos introducidos anteriormente.

$r_1 = 0$	$p_1 = 2$	$q_1 = 10$
$r_2 = 0$	$p_2 = 1$	$q_2 = 10$
$r_3 = 9$	$p_3 = 4$	$q_3 = 3$
$r_4 = 9$	$p_4 = 3$	$q_4 = 3$
$r_5 = 9$	$p_5 = 3$	$q_5 = 3$
$r_6 = 17$	$p_6 = 6$	$q_6 = 10$
$r_7 = 17$	$p_7 = 3$	$q_7 = 10$
$r_8 = 17$	$p_8 = 2$	$q_8 = 10$

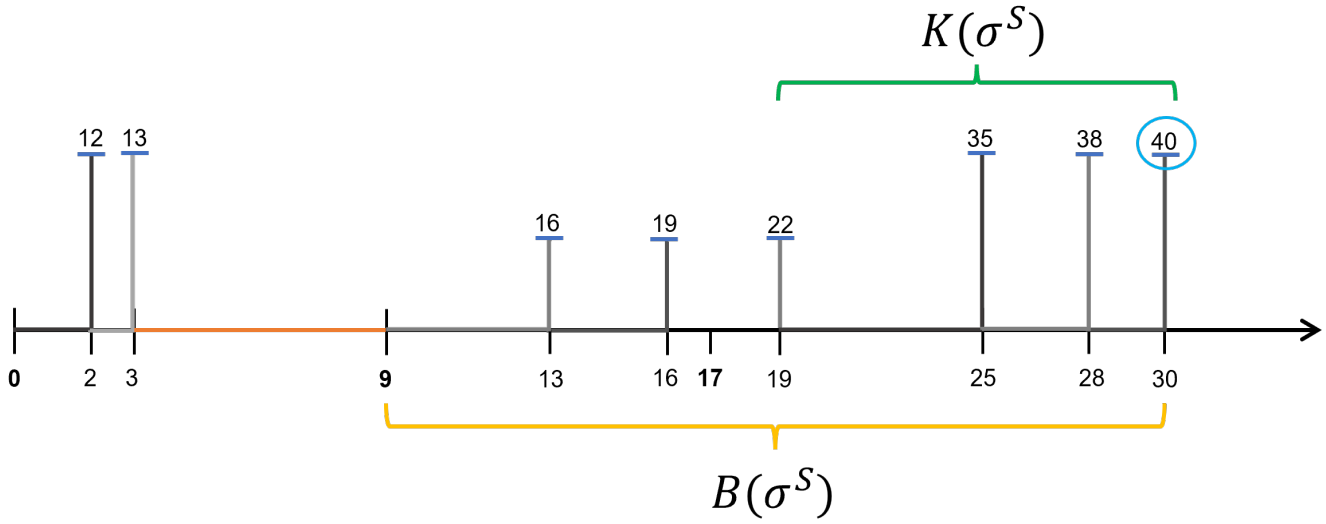


Figura 3.1: Ilustración de los conceptos

En la figura 3.1 se muestra una instancia con 8 trabajos y el calendario generado σ^S. Este calendario contiene dos bloques ya que la heurítica LDT crea un hueco que no se puede evitar dado por el intervalo $[3, 9]$. El segundo bloque es el bloque crítico $B(\sigma^S)$ pues contiene el *overflow job*, el trabajo 8. También es posible identificar tres trabajos emergentes en $B(\sigma^S)$, los trabajos 3, 4 y 5. El trabajo 5 es el trabajo emergente activo; por lo tanto, el *kernel* está conformado por los trabajos 6, 7 y 8.

3.2. Propiedades básicas del problema $1|r_i, q_i|C_{max}$

Consideremos el problema $1|r_i \in \{r^1, ..., r^k\}, q_i \in \{q^1, ..., q^l\}|C_{max}$. Recuerde que k y l son el número de tiempos de liberación y el número de tiempos de entrega distintos, respectivamente.

Lema 1. *Un calendario σ tiene a lo más l posibles overflow jobs dados por los trabajos j_i tales que $t_{j_i}(\sigma) = max\{t_j(\sigma) \mid j \in J(q^i)\}$ para cada $1 \leq i \leq l$.*

Demostración. Por la definición de j_i es claro que $c_{j_i}(\sigma) > c_j(\sigma)\ \forall\ j \in J(q^i)$, $1 \leq i \leq l$; es decir, los trabajos $j_1, ..., j_l$ dominan a todos los trabajos asignados que tienen su mismo tiempo de entrega. Por lo anterior, se tiene que $C_{j_i}(\sigma) > C_j(\sigma)\ \forall\ j \in J(q^i)$. Entonces $C_{max}(\sigma) = max\{C_{j_1}(\sigma), ..., C_{j_l}(\sigma)\}$. Por lo tanto, $j_1, ..., j_l$ son los posibles *overflow jobs*. □

Entonces, por el lema anterior, un calendario σ del problema $1|r_i \in \{r^1, ..., r^k\}, q_i \in \{q^1, q^2\}|C_{max}$ tiene a lo más dos posibles *overflow jobs*, en particular $j_1 \in J(q^1)$ y $j_2 \in J(q^2)$ tales que $t_{j_i}(\sigma) = max\{t_j(\sigma) \mid j \in J(q^i),\ i = 1, 2\}$.

Lema 2. *Si el overflow job $o(\sigma)$ está asignado en su tiempo de liberación, entonces σ es óptimo.*

Demostración. Si el *overflow job* $o(\sigma)$ está asignado en su tiempo de liberación, entonces $t_{o(\sigma)}(\sigma) = r_{o(\sigma)}(\sigma)$. De esta manera el $o(\sigma)$ inicia su procesamiento en su tiempo de inicio más temprano y finaliza en su tiempo de completés mínimo. Así que $o(\sigma)$ no puede ser reasignado más temprano en σ y entonces no existe σ' tal que $C_{max}(\sigma') < C_{max}(\sigma)$; por lo tanto, σ es óptimo. □

El siguiente lema es una extensión de un resultado anterior conocido para el caso general $1|r_i, q_i|C_{max}$ para la heurística LDT (ver [11]).

Lema 3. *Si* $q_j \geq q_{o(\sigma^S)}$ *para todo* $j \in B(\sigma^S)$*, es decir,* σ^S *no contiene ningún trabajo emergente, entonces* σ^S *es óptimo.*

Demostración. Si $t_{o(\sigma^S)}(\sigma^S) = r_{o(\sigma^S)}(\sigma^S)$ el lema resulta trivial. Supongamos que el *overflow job* inicia su procesamiento después de $r_{o(\sigma^S)}(\sigma^S)$. La única forma de que $o(\sigma^S)$ comience antes del tiempo $t_{o(\sigma^S)}(\sigma^S)$ es reasignando después del $o(\sigma^S)$ un trabajo $j \in B(\sigma^S)$ que haya sido asignado antes del $o(\sigma^S)$. Basta con mostrar que cualquier reasignación de un trabajo de $B(\sigma^S)$ después del $o(\sigma^S)$ no reduce el *makespan*. En efecto, si después de esta reasignación el $o(\sigma^S)$ no inicia antes de $t_{o(\sigma^S)}(\sigma^S)$ (inicia igual o después) entonces el lema es claro. Consideremos el caso cuando $o(\sigma^S)$ inicia antes de $t_{o(\sigma^S)}(\sigma^S)$ después de la reasignación. Ahora un trabajo j debe completarse después del $o(\sigma^S)$ y ya que $q_j \geq q_{o(\sigma^S)}\ \forall\ j \in B(\sigma^S)$ entonces el *makespan* no puede ser menor que $C_{o(\sigma^S)}(\sigma^S)$. Por lo tanto, σ^S es óptimo. □

Podemos observar que el calendario σ^G no está incluido en el lema anterior ya que la heurística LDT-G crea huecos de manera intencional. De esta forma podemos deducir que los bloques del calendario σ^G siempre cumplen con la condición del Lema 3; es decir, no contienen trabajos emergentes y; por lo tanto, el calendario σ^G no tiene *kernel*.

3.3. Observaciones de la heurística LDT

Observación 1. *El calendario generado por la heurística LDT es óptimo si* $J(r^i, q^1) = \emptyset$ *para cada* $1 \leq i \leq k-1$ *o* $J(r^i, q^2) = \emptyset$ *para cada* $2 \leq i \leq k$*.*

Demostración. Es fácil ver que si $J(r^i, q^1) = \emptyset$ para cada $1 \leq i \leq k-1$ o $J(r^i, q^2) = \emptyset$ para cada $2 \leq i \leq k$ entonces puede no existir un trabajo emergente en el calendario σ^S y; por lo tanto, la heurística LDT genera un calendario óptimo por el Lema 3. □

Del resultado anterior se deduce que si $J(r^i, q^1) = \emptyset$ para cada $1 \leq i \leq k-1$ entonces el calendario generado por la heurística LDT-G es óptimo pues los posibles huecos que se crean son inevitables. Sin embargo, si $J(r^i, q^2) = \emptyset$ para cada $2 \leq i \leq k$ entonces el calendario σ^G no necesariamente es óptimo, en particular si el calendario σ^G contiene un hueco evitable antes del tiempo de liberación r^k.

Observación 2. *El trabajo asignado* j_i *tal que* $t_{j_i}(\sigma^S) < r^{i+1} \leq c_{j_i}(\sigma^S)$ *tiene el máximo tiempo de entrega de entre todos los trabajos liberados que no han sido asignados al tiempo* r^i*.*

Demostración. Por la naturaleza de la heurística LDT, de entre los trabajos disponibles, un trabajo no puede ser asignado antes que otro con mayor tiempo de entrega. □

La siguiente observación es una extensión de un resultado anterior para el problema $1|r_i \in \{r^1, r^2\}, q_i \in \{q^1, q^2\}|C_{max}$ (ver Reynoso y Nodari [7]).

Observación 3. *Dada una instancia para el problema* $1|r_i \in \{r^1, ..., r^k\}, q_i \in \{q^1, q^2\}|C_{max}$*, el calendario generado* σ^S *tiene a lo más un kernel. Además, cada trabajo emergente para ese núcleo pertenece al conjunto* $J(r^{k-1}, q^1)$*.*

Demostración. Sin pérdida de generalidad, supongamos que $J(r^i, q^1) \neq \emptyset,\ 1 \leq i \leq k-1$ y $J(r^i, q^2) \neq \emptyset,\ 2 \leq i \leq k$ pues, de lo contrario, la heurística LDT genera un calendario óptimo (por la observación anterior). Primero mostramos que no puede existir un *kernel* en el calendario σ^S cuyo *overflow job* termina su procesamiento antes del tiempo r^k. Por contradicción, supongamos que σ^S tiene un *kernel* tal que el *overflow job* termina su procesamiento antes del tiempo r^k. Por la suposición anterior, $o(\sigma^S) \notin J(r^{k-1}, q^1)$ y $o(\sigma^S) \notin J(r^{k-1}, q^2)$. Nótese que el *overflow job* tampoco podría pertenecer a los conjuntos $J(r^i, q^1),\ 1 \leq i \leq k-2$ y $J(r^i, q^2),\ 2 \leq i \leq k-2$ porque los domina $J(r^{k-1}, q^1)$ y $J(r^{k-1}, q^2)$, respectivamente. De hecho, no existen trabajos emergentes para la secuencia de trabajos de $K(\sigma^S)$ ya que la heurística LDT no puede asignar trabajos del conjunto $J(r^{k-1}, q^1)$ antes de un trabajo del conjunto $J(r^{k-1}, q^2)$. Por tanto, $K(\sigma^S)$ no puede existir. Ahora, ya que hay trabajos liberados al tiempo r^k, el calendario $o(\sigma^S)$ puede tener como máximo un *kernel* que contenga un trabajo que termina su procesamiento después de r^k. Además, un trabajo emergente para ese núcleo solo puede iniciar su procesamiento antes del tiempo r^k y no puede pertenecer al conjunto $J(r^{k-1}, q^2)$; por lo tanto, $e \in J(r^{k-1}, q^1)$ □

Capítulo 4

Condiciones de optimalidad

En este capítulo se presentan los resultados obtenidos en el estudio del problema $1|r_i \in \{r^1, ..., r^k\}, q_i \in \{q^1, q^2\}|C_{max}$. Más específicamente, se demuestra que si se cumplen ciertas condiciones en un calendario generado por alguna de las heurísticas entonces ese calendario es óptimo. Los criterios de optimalidad son condiciones explícitas bajo las cuales el problema se puede resolver de manera óptima en tiempo $O(n \log n)$ cuando el calendario fue generado por las heurísticas LDT o LDT-G. En el capítulo anterior se establecieron las primeras condiciones cuando los calendarios σ^S y σ^G son óptimos.

Lema 4. *Cualquiera de las heurísticas genera una solución óptima si para cada* $1 \leq i \leq k-1$ *se tiene que* $r^i + P(J(r^i, q^2)) \geq r^{i+1}$.

Demostración. Es claro que los calendarios generados por las heurísticas LDT y LDT-G tienen el mismo *makespan* por la condición del lema; por lo tanto, si uno de ellos es óptimo entonces el otro también. Dada la condición del lema, en el calendario generado por la heurística LDT (o la heurística LDT-G) todos los trabajos del conjunto $J(q^2)$ se asignan antes que los trabajos del conjunto $J(q^1)$. De lo anterior se sigue que, el calendario no tiene *kernel* y entonces la condición del Lema 3 se mantiene; por lo tanto, el calendario es óptimo. □

Del resultado anterior se deduce que si $J(r^i, q^2) = \emptyset$ para alguna $1 \leq i \leq k-1$ entonces el calendario generado σ no necesariamente es óptimo, en particular si existe un trabajo de empuje crítico en el calendario. Del Lema 4 se tiene el siguiente corolario.

Corolario 1. *Sea* $m \in \mathbb{N}$ *tal que* $r^1 < r^m \leq r^k$, *si* $J(r^p, q^2) = \emptyset$ *para cada* $m \leq p \leq k$ *y se tiene que* $r^i + P(J(r^i, q^2)) \geq r^{i+1}$ *para cada* $1 \leq i \leq m-1$ *entonces la heurística LDT genera una solución óptima.*

Podemos observar que el calendario σ^G no está incluido en el corolario anterior ya que la heurística LDT-G crea huecos (posiblemente evitables) antes de cada r^i, $m+1 \leq i \leq k$, entre todos los trabajos asignados del conjunto $J(q^1)$.

Lema 5. *Cualquiera de las heurísticas genera una solución óptima si para cada* $1 \leq i \leq k-1$ *se tiene que* $r^i + P(J(r^i)) \leq r^{i+1}$.

Demostración. Es claro que por la condición del lema los calendarios generados por las heurísticas tienen el mismo *makespan* entonces si uno de ellos es óptimo, el otro también lo es. Sin pérdida de generalidad, supongamos que $J(r^i, q^1) \neq \emptyset$, $1 \leq i \leq k-1$ y $J(r^i, q^2) \neq \emptyset$, $2 \leq i \leq k$. Dada la condición del lema, el calendario generado por la heurística LDT (o la heurística LDT-G) contiene $k-1$ huecos, $o(\sigma) \in J(r^k)$ y $B(\sigma)$ comienza al tiempo r^k. Además, todos los trabajos del conjunto $J(r^k, q^2)$ se asignan antes que los trabajos del conjunto $J(r^k, q^1)$ por la naturaleza de las heurísticas. Por lo tanto, el calendario es óptimo por el Lema 3. □

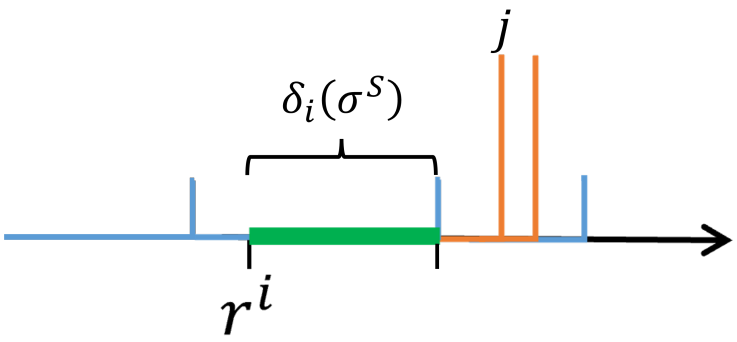
De los Lemas 4 y 5 se deduce que el calendario generado por cualquiera de las heurísticas puede no ser óptimo si y solo si $r^i + P(J(r^i, q^2)) < r^{i+1} < r^i + P(J(r^i))$ para cada $1 \leq i \leq k-1$.

Lema 6. *El calendario generado por la heurística LDT es óptimo si* $o(\sigma^S) \in J(q^1)$.

Demostración. Por la deducción anterior, supongamos que $r^i + P(J(r^i)) > r^{i+1}$ para cada $1 \leq i \leq k-1$. Por la naturaleza de la heurística LDT, sabemos que el último trabajo asignado en σ^S pertenece al conjunto $J(q^1)$. Dado que $o(\sigma^S) \in J(q^1)$, es claro que $o(\sigma^S)$ es el último trabajo asignado en el calendario por el Lema 1. Además, σ^S no contiene ningún hueco por la suposición, entonces σ^S consiste de un solo bloque. Ya que $q_j \geq q_{o(\sigma^S)}$ $\forall\, j \in B(\sigma^S)$ entonces σ^S es óptimo por el Lema 3. □

Es claro que el calendario σ^G no está incluido en el lema anterior por la naturaleza de la heurística LDT-G.

Sea j el primer trabajo asignado del conjunto $J(r^i)$ para cada $2 \leq i \leq k$ en el calendario σ^S, definimos por $\delta_i(\sigma^S) = t_j(\sigma^S) - r^i$ la longitud del desplazamiento del trabajo j. Se tiene en cuenta que $\delta_i(\sigma^G) = 0$, $2 \leq i \leq k$.



Por otro lado, sea C_{j_i} el tiempo de completés total del trabajo j_i tal que $t_{j_i} = max\left\{t_j(\sigma) \mid j \in J(q^i),\ i = 1, 2\right\}$.

Corolario 2. *La heurística LDT genera un calendario óptimo si*

$$0 \leq \delta_k(\sigma^S) \leq C_{j_1}(\sigma^S) - r^k - P(J(r^k, q^2)) - q^2$$

Demostración. Supongamos que $r^i + P(J(r^i, q^2)) < r^{i+1}$ para cada $1 \leq i \leq k-1$, entonces

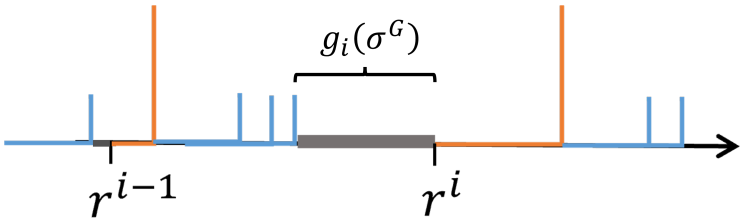
$$C_{j_2}(\sigma^S) = r^k + \delta_k(\sigma^S) + P(J(r^k, q^2)) + q^2.$$

Por la condición del corolario se deduce que $C_{j_2}(\sigma^S) \leq C_{j_1}(\sigma^S)$. Por lo tanto, $o(\sigma^S) \in J(q^1)$ y σ^S es óptimo por el Lema 6. □

Lema 7. *El calendario generado por la heurística LDT-G es óptimo si* $o(\sigma^G) \in J(r^k, q^2)$.

Demostración. Es fácil ver que $B(\sigma^G)$ siempre comienza al tiempo r^k y, además, se tiene que $q_j \geq q_{o(\sigma^G)}$ $\forall$ $j \in B(\sigma^G)$ asignado antes del *overflow job*. Ahora, aplicando un razonamiento similar a la demostración del Lema 3 a los trabajos de $B(\sigma^G)$ se deduce que cualquier reasignación de un trabajo de $B(\sigma^G)$ no reduce el *makespan*. Por lo tanto, σ^G es óptimo. □

Sea $g_i(\sigma^G)$ la longitud del hueco creado antes del tiempo r^i para cada $2 \leq i \leq k$ en el calendario generado por la heurística LDT-G. Nótese que un calendario σ^G tiene $k-1$ huecos y que σ^S puede contener huecos (inevitables).



Recordemos que J es el conjunto de los n trabajos.

Corolario 3. *La heurística LDT-G genera un calendario óptimo si*

$$0 \leq \sum_{i=2}^{k} g_i \leq C_{j_2}(\sigma^G) - r^1 - P(J) - q^1$$

Demostración. Notemos que

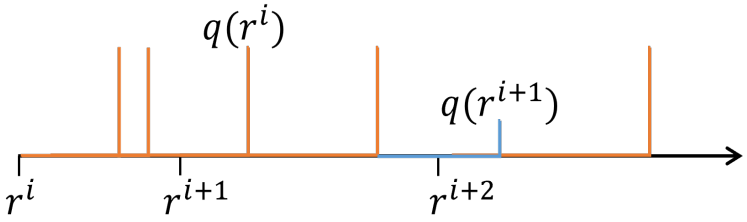
$$C_{j_1}(\sigma^G) = r^1 + P(J) + \sum_{i=2}^{k} g_i + q^1$$

ya que el último trabajo asignado en el calendario σ^G pertenece al conjunto $J(q^1)$. Por la condición del corolario se deduce que $C_{j_1}(\sigma^S) \leq C_{j_2}(\sigma^S)$; por lo tanto, $o(\sigma^G) \in J(r^k, q^2)$ y σ^G es óptimo por el Lema 7. □

Lema 8. *Sea* σ *el calendario generado por cualquiera de las heurísticas,* σ *es óptimo si* $\delta_i(\sigma) = 0$ *y* $g_i(\sigma) = 0$ *para cada* $2 \leq i \leq k$.

Demostración. Primero, se observa que si la condición $g_i(\sigma) = 0$ para cada $2 \leq i \leq k$ no se cumple, entonces todos los trabajos del conjunto $J(r^i)$ terminan su procesamiento antes del tiempo r^i para cada $1 \leq i \leq k-1$ y; por lo tanto, el calendario generado es óptimo por el Lema 5. Dadas las condiciones del lema, es claro que σ^S consta de k bloques asignados en su mínimo tiempo de inicio y que algún trabajo del conjunto $J(r^{k-1}, q^1)$ termina su procesamiento al tiempo r^k. Además, por la naturaleza de la heurística LDT, a partir del tiempo r^k todos los trabajos aún disponibles del conjunto $J(q^2)$ son asignados antes que los trabajos restantes del conjunto $J(q^1)$. Luego σ^S no contiene trabajos emergentes, entonces la demostración procede de manera similar a la del Lema 3. En el caso de σ^G, la condición $\delta_i(\sigma) = 0$ siempre se cumple y la condición $g_i(\sigma) = 0$ asegura que el calendario σ^G consta de k bloques asignados en su mínimo tiempo de inicio; por tanto, la demostración es similar al caso anterior. □

Sea $q(r^i)$ para cada $1 \leq i \leq k-1$ el tiempo de entrega del trabajo j cuando $t_j(\sigma^S) < r^{i+1} < c_j(\sigma^S)$ y recordemos que $q_{max}(r^{i+1})$ es el máximo tiempo de entrega de los trabajos liberados al tiempo r^{i+1}.



Lema 9. *Si para cada* $1 \leq i \leq k-1$ *se verifica que* $q(r^i) \geq q_{max}(r^{i+1})$, *entonces* σ^S *es óptimo.*

Demostración. Si $q(r^i) \geq q_{max}(r^{i+1})$ para cada $1 \leq i \leq k-1$, entonces $q_j \geq q_{o(\sigma^S)}$ para todo $j \in B(\sigma^S)$; por lo tanto, el calendario es óptimo por el Lema 3. □

Consideramos la posibilidad teórica de que ninguna de las condiciones establecidas anteriormente se cumple; por lo tanto, se aplica un algoritmo de programación dinámica que resuelve el problema SUBSET SUM y que puede utilizarse para resolver el problema de calendarización en tiempo pseudo-polinomial.

Capítulo 5

Heurística LDT-R

5.1. Descripción

Anteriormente se estableció que las heurísticas LDT y LDT-G consisten en k fases, siendo k el número de tiempos de liberación de la instancia dada, donde cada fase tiene dos partes. La primera parte asigna los trabajos del conjunto $J(q^2)$, y la segunda parte asigna los trabajos del conjunto $J(q^1)$ y algunos de los trabajos que no fueron asignados en la fase anterior con tiempo de entrega q^1. Esta segunda variante de la heurística LDT estándar consiste en agregar una tercera parte a cada fase k pero solo cuando es necesario. Con la intención de simplificar la descripción de esta heurística se define lo siguiente.

Se denota el **trabajo de empuje crítico** por j_{r^i} para cada $1 \leq i \leq k-1$ cuando $t_{j_{r^i}} < r^{i+1} < c_{j_{r^i}}(\sigma)$ tal que $q_{j_{r^i}} < q_{max}(r^{i+1})$ donde $q_{max}(r^{i+1})$ es el máximo tiempo de entrega de los trabajos liberados al tiempo r^{i+1}. Decimos que es un empuje crítico debido a que el trabajo j_{r^i} retrasa el tiempo de inicio de un trabajo con mayor tiempo de entrega.

Sea $A^i \subseteq J(r^i, q^1)$ el subconjunto de trabajos calendarizados en el intervalo $[r^i + P(J(r^i, q^2)), c_{j_{r^i}}(\sigma)]$, el objetivo es recalendarizar los trabajos del subconjunto A^i resolviendo el problema SUBSET SUM.

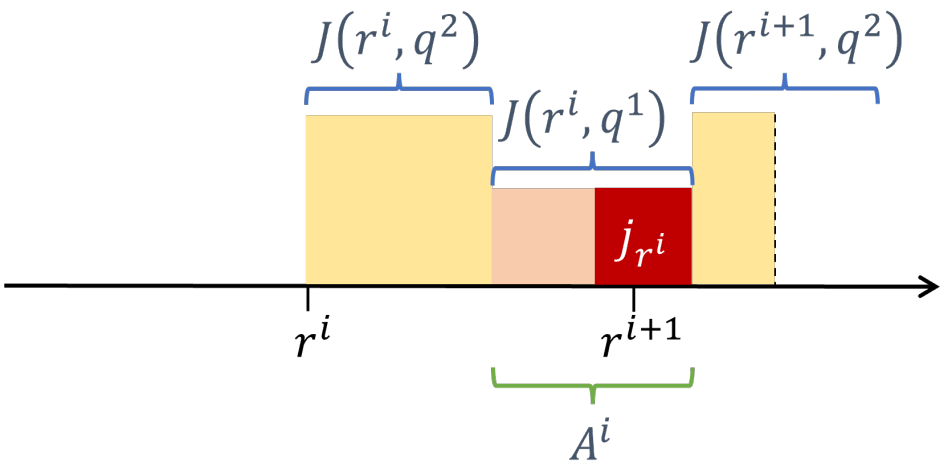


Figura 5.1: Representación del conjunto A^i

5.2. SUBSET SUM

El problema SUBSET SUM [12] consiste en determinar si un subconjunto de un conjunto finito de números enteros $\{a_1, a_2, ..., a_n\}$ puede sumar un valor objetivo $s = \sum_{i=1}^{n} a_i$; por ende, se trata de un problema de decisión cuya respuesta será "sí" en caso de que exista tal subconjunto cuyos elementos a_j suman s o "no" en caso contrario. Este problema es NP-completo [13]; no obstante, en su versión computacional es NP-duro.

5.3. SUBSET SUM y el problema de calendarización

La heurística LDT-R invoca un algoritmo de programación dinámica con tiempo $O(ns)$ que resuelve el problema SUBSET SUM [1]. La subrutina empleada genera una solución al problema de calendarización gracias a las propiedades estructurales derivadas de éste y su estrecha relación establecida con el problema.

Dada una instancia para el problema $1|r_i \in \{r^1, ..., r^k\}, q_i \in \{q^1, q^2\}|C_{max}$, las instancias correspondientes para el problema SUBSET SUM se definen de la siguiente manera. Los tiempos de procesamiento de los trabajos del conjunto $J(r^i, q^1)$, $1 \leq i \leq k-1$, son los p números enteros para el problema SUBSET SUM ($p < n$, $p = |J(r^i, q^1)|$). Sea s definido por la longitud del intervalo $[r^i + P(J(r^i, q^2)), c_{j_{r^i}}(\sigma)]$, es decir, $s = c_{j_{r^i}}(\sigma) - r^i - P(J(r^i, q^2))$. Esta subrutina solo se aplica cuando existe j_{r^i} ya que, de lo contrario, el calendario generado es óptimo según el Lema 3 (ver en el capítulo 3).

Para la heurística LDT-R consideraremos dos versiones, las cuales generarán dos posibles calendarios. Sean $\sigma^R(A)$ y $\sigma^R(\widehat{A})$ los calendarios generados por las heurísticas LDT-R^1 y LDT-R^2, respectivamente. Para resolver el problema SUBSET SUM, en ambas versiones, se aplican diferentes posibles subconjuntos A^i. En cada versión solo se utilizará un posible subconjunto por cada tiempo de liberación que potencialmente puede conducir a generar una solución óptima. Estos subconjuntos, los cuales nombraremos A^i_{opt} y $\widehat{A}^i_{opt}$, se generan mediante la subrutina antes mencionada.

Sea $A^i_{opt} \subseteq J(r^i, q^1)$ la solución obtenida del problema SUBSET SUM tal que

$$P(A^i_{opt}) \geq r^{i+1} - r^i - P(J(r^i, q^2))$$

$$P(A^i_{opt}) = \min_{A^i} P(A^i)$$

Lo cual significa que A^i_{opt} es el subconjunto más pequeño de tal forma que produce el menor desplazamiento y con esto el calendario $\sigma^R(A)$ no contiene ningún hueco (que se pueda evitar).

Análogamente, sea $\widehat{A}^i_{opt} \subseteq J(r^i, q^1)$ la solución obtenida del problema SUBSET SUM tal que

$$P(\widehat{A}^i_{opt}) \leq r^{i+1} - r^i - P(J(r^i, q^2))$$

$$P(\widehat{A}^i_{opt}) = \max_{\widehat{A}^i} P(\widehat{A}^i)$$

donde $\widehat{A}^i \subseteq J(r^i, q^1)$ es un subconjunto similar a A^i que cumple con la siguiente restricción

$$r^i + P(J(r^i, q^2)) + P(\widehat{A}^i) \leq r^{i+1}$$

Entonces $\widehat{A}^i_{opt}$ es el subconjunto más grande de modo que produce el hueco más pequeño posible antes de cada tiempo de liberación r^i, $2 \leq i \leq k$. En el calendario $\sigma^R(\widehat{A})$ los trabajos del conjunto $J(r^i, q^2)$ ocupan el intervalo $[r^i, r^i + P(J(r^i, q^2))]$, mientras que el intervalo $[r^i + P(J(r^i, q^2)), r^{i+1}]$ queda reservado únicamente para los trabajos del subconjunto $\widehat{A}^i_{opt}$.

5.4. Primera versión de la heurística LDT-R: LDT-R^1

Similar a la heurística LDT, la heurística LDT-R^1 asigna los trabajos con mayor tiempo de entrega liberados al tiempo r^1 de forma continua y en cualquier orden. Una vez asignados todos los trabajos del conjunto $J(r^1, q^2)$, se asigna de forma iterativa el trabajo del conjunto $J(r^1, q^1)$ con el mayor tiempo de procesamiento de tal manera que el último trabajo asignado de este conjunto comience su procesamiento antes del tiempo r^2 y sea completado después o al tiempo r^2. Si el último trabajo asignado de $J(r^1, q^1)$ se completa después de r^2, entonces se verifica la existencia del trabajo de empuje crítico j_{r^1} y se aplica la subrutina SUBSET SUM para obtener el subconjunto A^1_{opt}. Los trabajos del conjunto $J(r^1, q^1)$ que no pertenecen al subconjunto A^1_{opt} son incluidos en el conjunto $J(r^2, q^1)$. Una vez asignados los trabajos de A^1_{opt}, se asignan inmediatamente los trabajos del conjunto $J(r^2, q^2)$. Después se asignan los trabajos de $J(r^2, q^1)$ siguiendo las mismas condiciones de antes, se verifica la existencia de j_{r^2} para invocar la subrutina y se asignan los trabajos de A^2_{opt} en el calendario $\sigma^R(A)$. Este procedimiento se repite de la misma manera para los siguientes tiempos de liberación.

5.4.1. Ejemplo 1

Consideramos la siguiente instancia para comparar los calendarios generados por las heurísticas LDT y LDT-R^1. Cabe mencionar que los calendarios generados por la heurística LDT de las dos instancias consideradas en este capítulo no cumplen con ninguna de las condiciones de optimalidad antes establecidas.

$r_1 = 0$	$p_1 = 5$	$q_1 = 7$
$r_2 = 0$	$p_2 = 9$	$q_2 = 2$
$r_3 = 0$	$p_3 = 7$	$q_3 = 2$
$r_4 = 0$	$p_4 = 4$	$q_4 = 2$
$r_5 = 17$	$p_5 = 2$	$q_5 = 7$

LDT

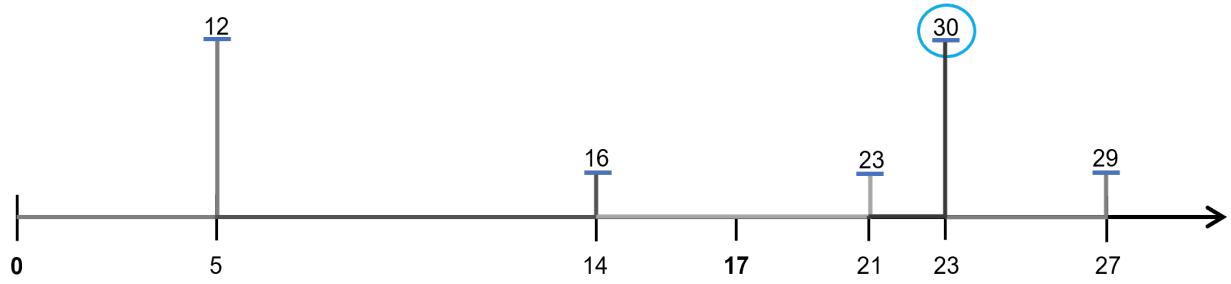


Figura 5.2: Calendario σ^S

En la figura 5.2 se muestra el calendario σ^S generado por la heurística LDT.

LDT-R^1

En la primera fase, la heurística LDT-R^1 asigna los trabajos con mayor tiempo de entrega liberados al tiempo $r^1 = 0$, en este caso solo asigna el trabajo j_1. Ahora se asigna de forma iterativa el trabajo del conjunto $J(r^1, q^1)$ con el mayor tiempo de procesamiento de tal manera que el último trabajo asignado de este conjunto comience su procesamiento antes del tiempo $r^2 = 17$ y sea completado después o al tiempo r^2. Ya que $t_3(\sigma^R(A)) + p_3 > 17$ se identifica el trabajo 3 como el trabajo de empuje crítico j_{r^1}. Entonces los trabajos del subconjunto $A^1 = \{j_2, j_3\}$ serán reasignados.

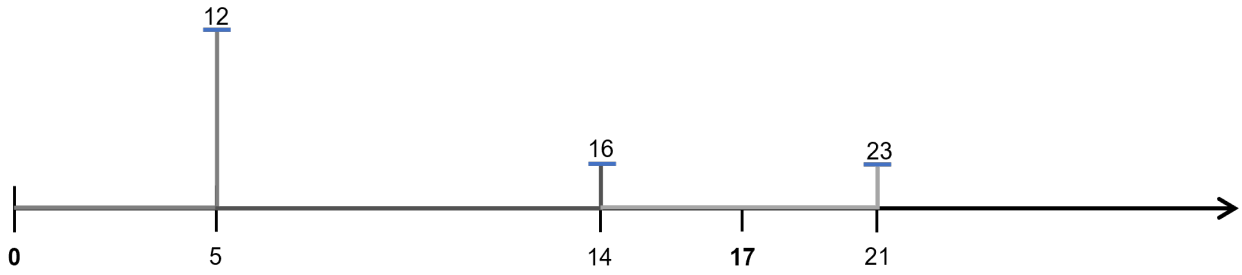


Figura 5.3: Calendario $\sigma^R(A)$ generado hasta la fase 1

Se aplica la subrutina SUBSET SUM, obteniendo como resultado el subconjunto $A^1_{opt} = \{j_2, j_4\}$. El trabajo j_3 es incluido en el conjunto $J(r^2, q^1)$. Los trabajos del conjunto $J(r^2, q^2)$ son calendarizados inmediatamente después de los trabajos de A^1_{opt}. Finalmente se asignan los trabajos del conjunto $J(r^2, q^1)$.

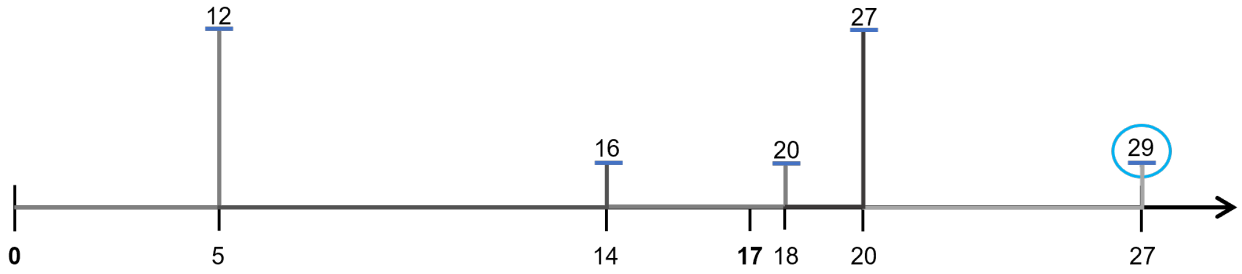


Figura 5.4: Calendario $\sigma^R(A)$

A diferencia del calendario generado σ^S, el *makespan* del calendario $\sigma^R(A)$ es igual a 29; en otras palabras, se redujo el máximo tiempo de competés total.

5.4.2. Ejemplo 2

Veamos otro ejemplo con una instancia de 11 trabajos.

$r_1 = 1$	$p_1 = 1$	$q_1 = 3$
$r_2 = 1$	$p_2 = 1$	$q_2 = 3$
$r_3 = 1$	$p_3 = 3$	$q_3 = 1$
$r_4 = 1$	$p_4 = 2$	$q_4 = 1$
$r_5 = 9$	$p_5 = 2$	$q_5 = 3$
$r_6 = 9$	$p_6 = 7$	$q_6 = 1$
$r_7 = 9$	$p_7 = 5$	$q_7 = 1$
$r_8 = 9$	$p_8 = 2$	$q_8 = 1$
$r_9 = 9$	$p_9 = 1$	$q_9 = 1$
$r_{10} = 24$	$p_{10} = 2$	$q_{10} = 3$
$r_{11} = 24$	$p_{11} = 1$	$q_{11} = 3$

LDT

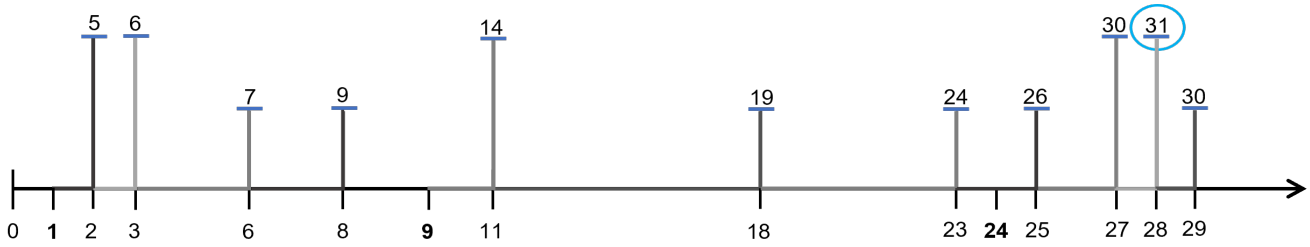


Figura 5.5: Calendario σ^S

En la figura 5.5 se muestra el calendario σ^S generado por la heurística LDT.

LDT-R[1]

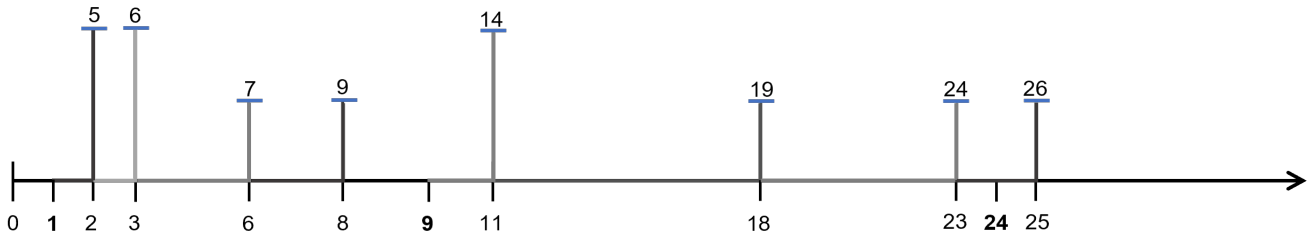


Figura 5.6: Calendario $\sigma^R(A)$ generado hasta la fase 2

Dado que no existe el trabajo j_{r^1}, la subrutina no se aplica en esta primera fase. Por esta razón la heurística crea un hueco de longitud 1 que no se puede evitar en el intervalo $[8, 9]$. A continuación se identifica el trabajo 8 como el trabajo j_{r^2} pues $t_8(\sigma^R(A)) + p_8 > 24$; los trabajos de $A^2 = \{j_6, j_7, j_8\}$ son reasignados.

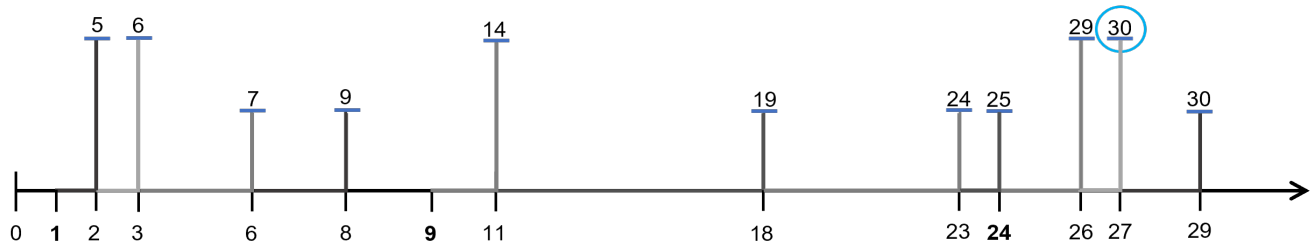


Figura 5.7: Calendario $\sigma^R(A)$

Se obtiene el subconjunto $A^2_{opt} = \{j_6, j_7, j_9\}$ tras aplicar la subrutina. Nótese que el *makespan* del calendario $\sigma^R(A)$ es menor que el de σ^S.

5.5. Segunda versión de la heurística LDT-R: LDT-R^2

La heurística LDT-R^2 es semejante a la heurística LDT-R^1 puesto que en cada fase i se verifica la existencia del trabajo j_{r^i} pero al aplicar la subrutina SUBSET SUM se obtiene el subconjunto $\widehat{A}^1_{opt}$.

5.5.1. Ejemplo 1

Retomamos la instancia del Ejemplo 1 de la heurística LDT-R^1 con el fin de comparar los calendarios generados.

$r_1 = 0$	$p_1 = 5$	$q_1 = 7$
$r_2 = 0$	$p_2 = 9$	$q_2 = 2$
$r_3 = 0$	$p_3 = 7$	$q_3 = 2$
$r_4 = 0$	$p_4 = 4$	$q_4 = 2$
$r_5 = 17$	$p_5 = 2$	$q_5 = 7$

LDT

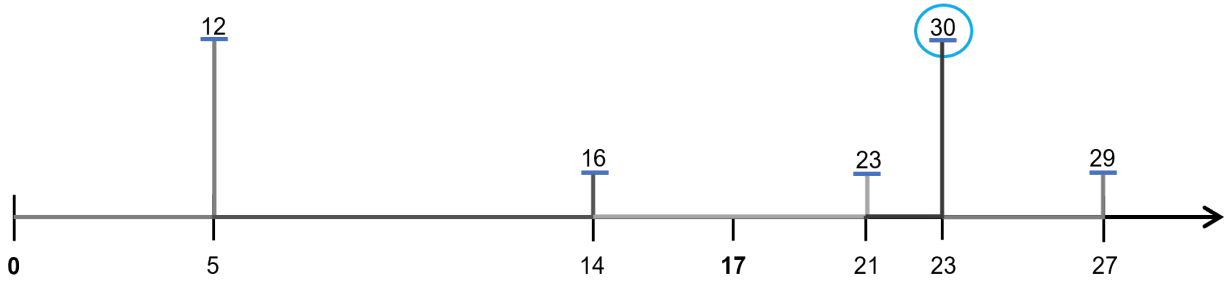


Figura 5.8: Calendario σ^S

LDT-R^2

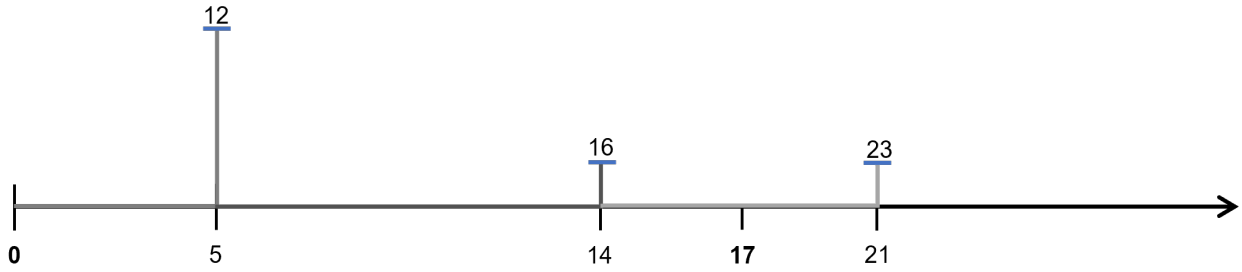


Figura 5.9: Calendario $\sigma^R(A)$ generado hasta la fase 1

Una vez que se identifica el trabajo 3 como el trabajo de empuje crítico j_{r^1}, se aplica la subrutina SUBSET SUM a partir de la cual se obtiene el subconjunto $\widehat{A}^1_{opt} = \{j_3, j_4\}$.

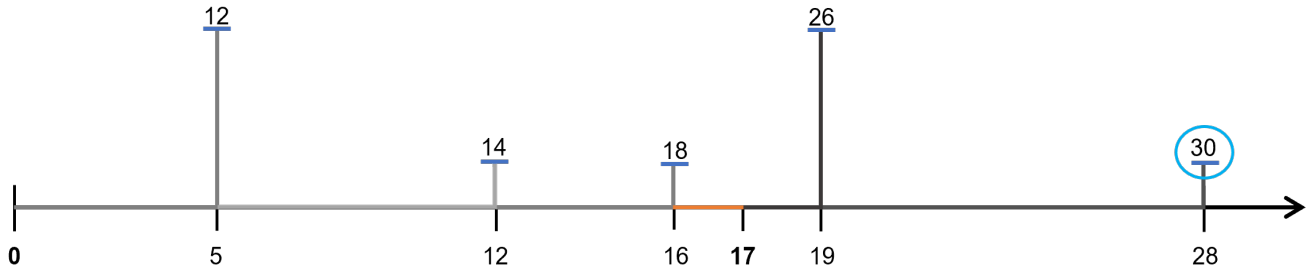


Figura 5.10: Calendario $\sigma^R(\widehat{A})$

En este caso, se produjo en el calendario $\sigma^R(\widehat{A})$ un hueco dado por el intervalo $[16, 17]$ y el *makespan* se mantuvo igual con respecto al calendario generado por la heurística LDT.

5.5.2. Ejemplo 2

Retomamos la instancia del Ejemplo 2 de la heurística LDT-R[1].

$r_1 = 1$	$p_1 = 1$	$q_1 = 3$
$r_2 = 1$	$p_2 = 1$	$q_2 = 3$
$r_3 = 1$	$p_3 = 3$	$q_3 = 1$
$r_4 = 1$	$p_4 = 2$	$q_4 = 1$
$r_5 = 9$	$p_5 = 2$	$q_5 = 3$
$r_6 = 9$	$p_6 = 7$	$q_6 = 1$
$r_7 = 9$	$p_7 = 5$	$q_7 = 1$
$r_8 = 9$	$p_8 = 2$	$q_8 = 1$
$r_9 = 9$	$p_9 = 1$	$q_9 = 1$
$r_{10} = 24$	$p_{10} = 2$	$q_{10} = 3$
$r_{11} = 24$	$p_{11} = 1$	$q_{11} = 3$

LDT

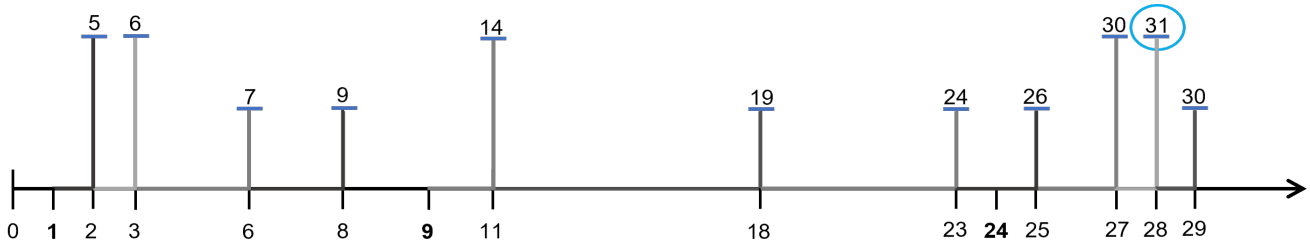


Figura 5.11: Calendario σ^S

LDT-R[2]

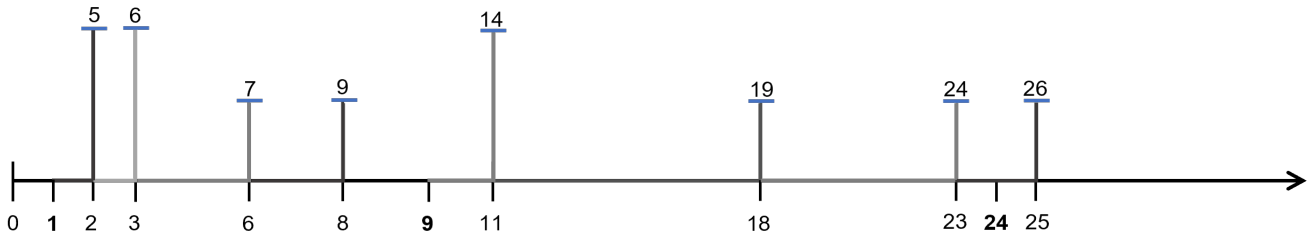


Figura 5.12: Calendario $\sigma^R(\widehat{A})$ generado hasta la fase 2

Tras identificar el trabajo 8 como el trabajo j_{r^2} pues $t_8(\sigma^R(A)) + p_8 > 24$ se aplica la subrutina, entonces los trabajos de $A^2 = \{j_6, j_7, j_8\}$ son reasignados.

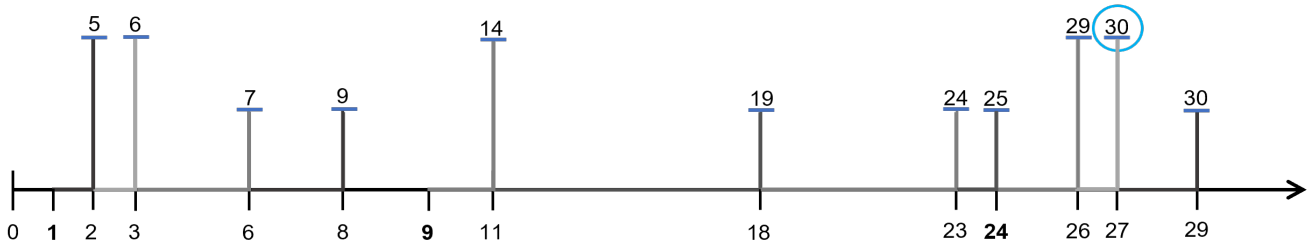


Figura 5.13: Calendario $\sigma^R(\widehat{A})$

Al aplicar la subrutina en la segunda fase se obtiene que $\widehat{A}^2_{opt} = A^2_{opt} = \{j_6, j_7, j_9\}$ debido a que

$$P(A^2_{opt}) = r^3 - r^2 - P(J(r^2, q^2))$$

Es decir, A^2_{opt} produce el menor desplazamiento ($\delta_3(\sigma^R(A)) = 0$) y, a la vez, el hueco más pequeño ($g_3(\sigma^R(\widehat{A})) = 0$). Entonces ambas versiones de la heurística generan el mismo calendario, un calendario que no contiene ningún trabajo de empuje crítico.

5.6. Condiciones de optimalidad

Lema 10. *El calendario σ^R generado por cualquiera de las dos versiones de la heurística LDT-R es óptimo si $o(\sigma^R) \in J(q^2)$ y σ^R no contiene un trabajo j_{r^i}.*

Demostración. Dado que σ^R no contiene un trabajo j_{r^i}, entonces $r^i + P(J(r^i)) \leq r^{i+1}$ para cada $1 \leq i \leq k-1$. Por la última condición, $B(\sigma^R)$ comienza al tiempo r^k. Es en este punto donde es claro que los calendarios generados por ambas versiones de la heurística LDT-R tienen el mismo *makespan*; de esta forma si uno de ellos es óptimo, el otro también lo es. Además $o(\sigma^R) \in J(q^2)$ por lo cual se tiene que $q_j \geq q_{o(\sigma^R)}$ $\forall$ $j \in B(\sigma^R)$ asignado antes del *overflow job*. Aplicando un razonamiento similar a la demostración del Lema 3 a los trabajos de $B(\sigma^R)$ se deduce que cualquier reasignación de un trabajo de éste conjunto no reduce el *makespan*. Por tanto, σ^R es óptimo. □

Lema 11. *El calendario $\sigma^R(A)$ es óptimo si $o(\sigma^R(A)) \in J(q^1)$ y $\sigma^R(A)$ contiene los trabajos j_{r^i}, $1 \leq i \leq k-1$.*

Demostración. Ya que $\sigma^R(A)$ contiene los trabajos j_{r^i}, $1 \leq i \leq k-1$, entonces $\sigma^R(A)$ no contiene ningún hueco pues $r^i + P(J(r^i)) > r^{i+1}$ para cada $1 \leq i \leq k-1$. Esto implica que el calendario $\sigma^R(A)$ consta de un solo bloque. También $o(\sigma^R(A)) \in J(q^1)$, siendo así, $o(\sigma^R(A))$ es el último trabajo asignado en $\sigma^R(A)$ por el Lema 1. Puesto que $q_j \geq q_{o(\sigma^R(A))}$ $\forall$ $j \in B(\sigma^R)$ entonces $\sigma^R(A)$ es óptimo por el Lema 3. □

El calendario $\sigma^R(\widehat{A})$ no está incluido en el lema anterior por la naturaleza de la segunda versión de la heurística LDT-R.

Capítulo 6

Conclusiones

El estudio del problema con un número arbitrario de tiempos de liberación, tiempos de procesamiento y dos tiempos de entrega nos permitió generalizar las condiciones de optimalidad establecidas anteriormente en el estudio del problema con dos tiempos de liberación, tiempos de procesamiento y dos tiempos de entrega.

Llevar a cabo el estudio considerando solo dos tiempos de entrega nos permitió identificar los casos cuando hay conflicto para garantizar optimalidad ya que el problema surge principalmente cuando un trabajo de menor tiempo de entrega (menos urgente) retrasa el tiempo de inicio de un trabajo urgente (que pertenece a $J(q^2)$).

Conforme se agregan más tiempos de entrega en las restricciones del problema, más consideraciones se deberán tomar pues para encontrar dónde ocurre el potencial conflicto se deben identificar los posibles overflow jobs. Por ejemplo, para el problema $1|r_i \in \{r^1, ..., r^k\}, q_i \in \{q^1, q^2, q^3\}|C_{max}$ se deberán considerar los retrasos generados por los trabajos con tiempos de entrega q1 y q2.

Será necesario realizar experimentos computacionales con la finalidad de probar la eficiencia práctica de las condiciones de optimalidad establecidas en este estudio para las diferentes heurísticas.

Este enfoque propuesto, tanto las heurísticas como las condiciones de optimalidad, puede extenderse para las configuraciones con más de dos tiempos de entrega. No obstante, para llevarlo a la práctica será necesario profundizar en el estudio de los subcasos con tres o más tiempos de entrega.

Capítulo 7

Notaciones

J	Conjunto de n trabajos con tiempos de liberación, tiempos de procesamiento y tiempos de entrega.
j	Un trabajo del conjunto **J**.
r_j	Tiempo de liberación del trabajo j.
p_j	Tiempo de procesamiento del trabajo j.
q_j	Tiempo de entrega del trabajo j.
σ	Calendario.
$t_j(\sigma)$	Tiempo de inicio del trabajo j en el calendario σ.
$c_j(\sigma)$	Tiempo de completés del trabajo j en el calendario σ.
$C_j(\sigma)$	Tiempo de completés total del trabajo j en el calendario σ.
$C_{max}(\sigma)$	***makespan***.
$J(r^i, q^l)$	El conjunto de trabajos con tiempos de liberación y entrega r^i y q^l, respectivamente, con $i = 1, ..., k$ y $l = 1, 2$
$J(r^i)$	El conjunto de trabajos con tiempo de liberación r^i.
$J(q^l)$	El conjunto de trabajos con tiempo de entrega q^l.
$P(A)$	La suma de los tiempos de procesamiento de todos los trabajos del conjunto $A \subseteq J$.
σ^S	El calendario generado por la heurística LDT.
σ^G	El calendario generado por la heurística LDT-G.
$o(\sigma)$	***overflow job***.
$B(\sigma)$	El bloque crítico del calendario σ.

e	Trabajo emergente.
e'	Activo.
$K(\sigma)$	***kernel***.
$\delta_i(\sigma)$	Longitud del desplazamiento del primer trabajo asignado del conjunto $J(r^i)$ para cada $2 \leq i \leq k$ en σ.
$g_i(\sigma)$	Longitud del hueco creado antes del tiempo r^i para cada $2 \leq i \leq k$ en σ.
j_{r^i}	El trabajo de empuje crítico.
A^i	El subconjunto de trabajos calendarizados en el intervalo $[r^i + P(J(r^i, q^2)), c_{j_{r^i}}(\sigma)]$.
$\widehat{A}^i$	El subconjunto de trabajos tal que $r^i + P(J(r^i, q^2)) + P(\widehat{A}^i) \leq r^{i+1}$.
$\sigma^R(A)$	El calendario generado por la heurística LDT-R^1.
$\sigma^R(\widehat{A})$	El calendario generado por la heurística LDT-R^2.
A^i_{opt}	Subconjunto obtenido tras resolver el problema SUBSET SUM de la heurística LDT-R^1.
$\widehat{A}^i_{opt}$	Subconjunto obtenido tras resolver el problema SUBSET SUM de la heurística LDT-R^2.

Bibliografía

[1] Garey, M.R. and Johnson, D.S. (1979). Computers and Intractability: A Guide to the Theory of NP completeness. Freeman, San Francisco.

[2] Behnamian, J. (2015). Combined Electromagnetism-Like Algorithm with Tabu Search to Scheduling. In *Handbook of Research on Artificial Intelligence Techniques and Algorithms.* (pp. 478-508)

[3] R.L. Graham., E.L. Lawler , J.L. Lenstra , and A.H.G. Rinnooy Kan. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. *Ann. Discrete Math.* 5 287 326 (1979). Information Science Reference.

[4] L. Schrage. Obtaining optimal solutions to resource constrained network scheduling problems. Unpublished manuscript, (1971).

[5] J.R. Jackson. Scheduling a production line to minimize the maximum tardiness. *Management Science Research Project*, University of California, Los Angeles, CA (1955).

[6] M. I. Dessouky and R. E. Larson. Symmetry and optimally properties of the single machine problem. *AIIE Transactions*, 10(2):170-175 (1978).

[7] N. Vakhania. Dynamic restructuring algorithm for minimizing maximum lateness. Unpublished manuscript, (2016).

[8] A. Reynoso and N. Vakhania. Theoretical and practical issues in single-machine scheduling with two job release and delivery times. *Journal of Scheduling* 24, 615-647 (2021). https://doi.org/10.1007/s10951-021-00708-4

[9] N. Vakhania. A better algorithm for sequencing with release and delivery times on identical processors. *Journal of Algorithms* 48, p.273-293 (2003).

[10] N. Vakhania. Single-Machine Scheduling with Release Times and Tails. *Annals of Operations Research*, 129, p.253-271 (2004).

[11] E. Chinos and N. Vakhania. Adjusting scheduling model with release and due dates in production planning. *Cogent Engineering* 4(1), p. 1-23 (2017). DOI: 10.1080/23311916.2017.1321175

[12] Menezes, A., van Oorschot, P., and Vanstone, S. (1996). Number-Theoretic Reference Problems. *In Handbook of Applied Cryptography.* (pp. 117-118)

[13] Nemhauser, G. and Wolsey, L. (2014). Computational Complexity. *In Integer and Combinatorial Optimization* (pp. 137)

[14] E. Hart, P. Ross, and J. Nelson. Solving a real-world problem using an evolving heuristically driven schedule builder. *Evolutionary Computation*, 6(1):61-80 (1998).

[15] Horowitz, E. and Sahni, S. (1998). Fundamentals of computer algorithms. Computer Science Press, Maryland.

[16] Du, D.-Z. and Pardalos, P. M., eds. (1998). Handbook of Combinatorial Optimization. Volume 3. Kluwer Academic Publishers, The Netherlands.

[17] C. Koulamas and G. J. Kyparisis. A classification of dynamic programming formulations for offline deterministic single-machine scheduling problems. *European Journal of Operational Research*, Volume 305, Issue 3 (2023). https://doi.org/10.1016/j.ejor.2022.03.043

[18] M. A. Cruz, P. Moreno y M. Martínez. Optimización combinatoria. *Inventio*, 9(18), 45-49. http://inventio.uaem.mx/index.php/inventio/article/view/354

[19] Papadimitriou, C. and Steiglitz, K. (1998). Combinatorial Optimization: Algorithms and Complexity. Dover Publications, Inc., New York.

[20] Korte, B. and Vygen, J. (2006). Combinatorial Optimization. Theory and Algorithms. Third Edition. Springer, Germany. https://doi.org/10.1007/3-540-29297-7

[21] Brassard, G. y Bratley, P. (2000). Fundamentos de algoritmia. Prentice Hall, España.

[22] Neapolitan, R. E. (2014) Foundations of algorithms. Fifth Edition. Jones & Bartlett Learning, United States of America.

[23] Rosen, K. H. (2005) Matemática discreta y sus aplicaciones. Quinta edición. McGraw-Hill Companies, España.

VOTOS DE APROBATORIOS

SECRETARIA EJECUTIVA
INSTITUTO DE INVESTIGACIÓN EN CIENCIAS BÁSICAS Y APLICADAS
UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

P R E S E N T E

Por medio del presente le informamos que después de revisar la versión escrita de la tesis que realizó la **C. ESCALONA CONTRERAS DIANA** con número de matrícula **10018772** cuyo título es:

ASPECTOS TEÓRICOS Y PRÁCTICOS DEL PROBLEMA $1|ri \in \{ri, \dots, rk\}, qi \in \{q1, q2\}|C\max$

Consideramos que **SI** reúne los méritos que son necesarios para continuar los trámites para obtener el título de **LICENCIADO EN CIENCIAS ÁREA TERMINAL EN MATEMATICAS**

Cuernavaca, Mor a 28 de mayo de 2024

Atentamente
Por una humanidad culta

Se adiciona página con la e-firma UAEM de los siguientes:

DR. ANTONIO DANIEL RIVERA LÓPEZ	**(PRESIDENTE)**
DRA. LORENA DÍAZ GONZÁLEZ	**(SECRETARIO)**
DR. NODARI VAKHANIA MAISURADZE	**(VOCAL)**
DRA. MARÍA ELISA CHINOS OLIVAN	**(SUPLENTE)**
DR. DAN SIDNEY DÍAZ GUERRERO	**(SUPLENTE)**

MIE/VRRC/eae

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

LORENA DIAZ GONZALEZ | Fecha:2024-05-28 11:50:27 | FIRMANTE

R8B8t5DSsMnjJ7H7eaEQlZ+iyqR7XYYHykHqzCuTDH5C1s42/ZFR1PmLX+t0X9qrGOrQV2X9O8F8H6zEoebHOgwsmKNyPBoHoBDNEYwIMOeXI64uSA/UiFX2WzBfb/1M7e3p13ostrZb8sSQu7RQDL3sib3NhsYOl7ShjV8zYFFRB9zS/4CvP9+vkmWlhdusejTlC8Pm6JSMjembwemshvKTdVmEfwQ5/2OLv8ENWQVzaAvBYgRhbfmPUg659ulEfVFhrDQnOHf3fpSBtID6N2mxtzx+dh7h9X8Nxk7ITclv+bzlrTP1TqWaM0yTNeFZLL7Xf4PYIL90iS63snpchg==

NODARI VAKHANIA MAISURADZE | Fecha:2024-05-28 14:38:20 | FIRMANTE

Gk0EM0aP4+NhStsQr6+/w/LP1w3C3/eywmFF3rIT2ivki5BA10JEMSKDPF/f8bk9IgmEULKvupeW9EtDkqUILCdOivhpMjBrRC+f/PWEluRENH3s12YnbPjZBm6DczhVgiEW4saamCROIfjRNn6TCl+G0tVf7cxdQFyefzjmJOVnBFvTzG9AJ2qJvgho8KJvPqLPpv1gLBTVFXUYgJKwg93pQjumhLolySTlguYfXMpkFP1jA5ox8H7j7sjRQtmETENNrrrMLqv9SUqWhKE9Z0CwqGslC2SJcrtyrem3/u+tnz5QY0eU9+bauYvY3Oe6xlw3ql5ZaX5I6pEnvUZ+GQ==

DAN SIDNEY DIAZ GUERRERO | Fecha:2024-05-28 20:23:24 | FIRMANTE

hpmrZXHVvAVWbHcVzwVc4EHuBo9HvvV6KhG2//VftClxHwx0WEQfmydwiBVVAi+wMhw8RcvWXKRc6Tl8zJPrW8hqdw6Viis/FeOo0QukPEagwyJVa7Pa+XSrrLGMnspwNEp0X1xVyMJTWqONT1BzAGfTDuizPNNM3pbFICfMVzxVOTCx/ZKPTzhZroSNBrSShv2Uj3+MiUakY2v0UOMARMfeBjwiGkgg7KFhDHu9mbaZn7uyYIjTPEytGwU5sqgnL8Rq+w7HwiwT0stiEm+8dlc3F5lNa5id3iSL0Z3kAh4kLlflZf0Cv+94p0fROvAWb6owZtMfeLCqXSnfEBQ/JA==

ANTONIO DANIEL RIVERA LOPEZ | Fecha:2024-05-31 14:15:41 | FIRMANTE

GhElGZLHaOtyZLBKKGTkO5eon+n0pYB1mj5raQVyIr/X8zv6TUU68wzVecIAVajNFXd8x5BK1RgxnVmx+UOd/EHYYtn7QskhvDQg30CgfL3O9/2xnV/R0QehpyxnFlgPQ5Dgtx087qlDAlP+jq3qOrxOHON9T1yg9Nvfsf0QXtnJRWWlLl1rkyzr4SrcUO6ZYpOm/kUm6daegiSkkzJdEm/EFnQnf0AN/BN35dqTnC+ED3SVF/DvwzORbPgUmWxP+58dFjLhIhcFuahFlhddOa69xdtmDAerTyGa8+lUXPda3kDAlv0JEvcFlWOkK79Qydbk+0iLgk2A13ZBe9e5CA==

MARIA ELISA CHINOS OLIVAN | Fecha:2024-06-01 21:38:42 | FIRMANTE

Hi0NDxNIocm96BRX3JnW5FwHDc6+MtnT0uuB5cv0ryAQXAZuF5gv0svzC7MxtGIW9mtHofpaFSzCWFfHmTsSi0a5hER1mUKJvMGJD5DUoGoMYiMkFR+n5fz5WSltCI65OhwpVaIyjhm8TUzY8rRKhJBrJmpVpmiPaVOD13rrOwBkssOL5NA7tmYJy7IxpCBC6VqJ/j9XL9J7emPxAKf8tEzjo20ffb8iuyO5LiZ7N+qLP1s55Dt95F1VtCrQw9K4/48lavVMnqYfkQ57/Y2i51dq/yGgtyD9Yp5AXhcel5nJopa+ErSwU4FkrExJiJ6sOxvQ4YiqGE9QcOml10w2TA==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

xJ0LBvfrO

https://efirma.uaem.mx/noRepudio/M2h1O0dH9tJSXq8mKennZzxgC2aQIceK