CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA, BAJA CALIFORNIA

PROGRAMA DE POSGRADO EN CIENCIAS EN CIENCIAS DE LA COMPUTACIÓN

Diseño de algoritmos para el problema de coordinación de movimiento de múltiples robots a lo largo de trayectorias pre-especificadas

Tesis

para cubrir parcialmente los requisitos necesarios para obtener el grado de Maestro en Ciencias

Presenta:

Julio Antonio Juárez Jiménez

Ensenada, Baja California, México

2015

Tesis defendida por

Julio Antonio Juárez Jiménez

y aprobada por el siguiente comité

Dr. Carlos Alberto Brizuela Rodríguez
Director del Comité

Dr. José Alberto Fernández Zepeda
Miembro del Comité

Dr. Israel Marck Martínez Pérez
Miembro del Comité

Dr. Joaquín Álvarez Gallegos
Miembro del Comité

Dra. Ana Isabel Martínez García
Coordinador del Programa de Posgrado en Ciencias de la Computación

Dr. Jesús Favela Vara
Director de Estudios de Posgrado

Febrero, 2015

Resumen de la tesis que presenta Julio Antonio Juárez Jiménez como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación.

Diseño de algoritmos para el problema de coordinación de movimiento de múltiples robots a lo largo de trayectorias pre-especificadas

Resumen elaborado por:

Julio Antonio Juárez Jiménez

El problema de la coordinación de rutas de robots (RPC) y el problema conocido como *Job Shop Scheduling* (JSP) se han estudiado ampliamente de manera independiente a lo largo de los años. A pesar de que la similitud entre estos dos problemas se ha señalado (O'Donnell y Lozano-Pérez, 1989; Peng y Akella, 2005), hasta el momento se desconoce de algún trabajo que enuncie de manera formal la estrecha similitud entre éstos. La importancia de la similitud entre estos problemas radica en que el problema JSP pertenece a la clase de problemas conocidos como *NP-difícil*, por lo que hasta hoy en día se considera intratable bajo el modelo de cómputo convencional.

En el presente trabajo, se explora formalmente la semejanza entre ambos problemas y se propone una técnica de agrupamiento de robots basada en diagramas de coordinación *n*-dimensionales (originalmente propuestos para el RPC (Siméon *et al.*, 2002)) para encontrar soluciones para casos del JSP. Se proponen tres algoritmos determinísticos de calendarización implementando la técnica de agrupamiento de robots: el GANTT, el GANTT-T y el GANTT-T+. Los tres algoritmos realizan un procedimiento iterativo de agrupamiento y calendarización de un número constante (k) de robots, con diferencias particulares entre cada algoritmo. El GANTT busca maximizar el recorrido paralelo de segmentos de ruta que no pertenezcan a una misma región, en un mismo intervalo de tiempo. El GANTT-T implementa un adelgazador que permite calcular el tiempo de inicio de recorrido de un segmento de ruta; que inicie (el recorrido) tan pronto como el segmento que lo precede, en su misma ruta, haya sido recorrido y la región a recorrer se haya liberado. El GANTT-T+ implementa un adelgazador mejorado y calcula el mínimo tiempo de inicio de un segmento de ruta; permite iniciar el recorrido tan pronto como sea posible, respetando la restricción de precedencia de los segmentos de ruta.

Los resultados experimentales muestran que: $i)$ El algoritmo GANTT-T+ es superior (calidad de solución) a los algoritmos GANTT y GANTT-T. $ii)$ El algoritmo GANTT-T+ muestra un error relativo (ER) promedio del $5.48\,\%$ con $k = 3$ y del $6.84\,\%$ con $k = 2$, sobre 82 casos de prueba conocidos del JSP. $iii)$ La evidencia teórica y experimental muestran que el algoritmo GANTT-T+ se comporta como un algoritmo exacto de tiempo exponencial, cuando $k = n$.

Palabras Clave: **Coordinación de rutas de robots, Job Shop Scheduling, Diagramas de coordinación, NP-difícil.**

Abstract of the thesis presented by Julio Antonio Juárez Jiménez as a partial requirement to obtain the Master of Science degree in Master in Sciences in Computer Science.

Design of algorithms for the coordination of multiple moving robots along pre-specified trajectories

Abstract by:

Julio Antonio Juárez Jiménez

The Robot Path Coordination problem (RPC) and the Job Shop Scheduling problem (JSP) have been widely studied over the years. Despite the similarity between these two problems has been pointed out (O'Donnell y Lozano-Pérez, 1989; Peng y Akella, 2005), we do not know of any work that formally states the close relation between them. The importance of this similarity lies on the computational complexity of the JSP. The JSP belongs to a set of problems known as $NP-hard$, which are considered to be intractable under the conventional computing model.

In the present work, the formal similitude between the RPC and the JSP is explored. Also, a robot grouping technique, based on *n*-dimensional coordination diagrams (originally proposed for the RPC (Siméon *et al.*, 2002)) to find solutions for JSP instances, is proposed. Three deterministic scheduling algorithms implementing the grouping technique are proposed: GANTT, GANTT-T and GANTT-T+. The three algorithms perform an iterative procedure of grouping and scheduling of a constant number (k) of robots, with specific differences between each pair of algorithms. GANTT seeks to maximize the parallel execution of path segments that do not belong to the same region, in the same time interval. GANTT-T implements a trimming procedure which computes the starting execution time of a path segment; to start as soon as the segment which precedes it, in its same path, has been executed and the region to be executed has been freed. GANTT-T+ implements an improved trimming procedure which computes the minimum starting execution time of a path segment; to start as soon as possible, as long as the segment precedence constraints are met.

The experimental results, show that: $i)$ The GANTT-T+ algorithm is superior (in quality of solution) with respect to the GANTT and GANTT-T algorithms. $ii)$ The algorithm GANTT-T+ shows an average relative error of $5.48\,\%$ with $k = 3$ and of $6.84\,\%$ with $k = 2$, over 82 known instances of the JSP. $iii)$ Theoretical and experimental evidence suggests that, when $k = n$, the GANTT-T+ algorithm behaves as an exponential time exact algorithm.

Keywords: **Robot Path Coordination, Job Shop Scheduling, Coordination diagrams, NP-hard.**

Dedicatoria

A lo indecidible

Agradecimientos

A mis padres, por haberme dado la vida. En especial a mi madre, María Teresa, por haber estado para mí siempre que lo necesité.

A mi director de tesis, el Dr. Carlos Alberto Brizuela Rodríguez, quien me guío pacientemente a lo largo del desarrollo de este trabajo. Le agradezco sus enseñanzas y el haber despertado en mí el interés en el área.

A los miembros de mi comité de tesis, el Dr. José Alberto Fernández Zepeda, el Dr. Israel Marck Martínez Pérez y el Dr. Joaquín Álvarez Gallegos a quienes les agradezco su tiempo, sus observaciones y sugerencias para mejorar mi investigación.

A mis amigos que, por sus palabras, nunca me dejaron sentir que estaba lejos.

A Crsitina, quien estuvo a mi lado en los buenos y malos momentos durante la tesis.

A mis compañeros del posgrado, con quienes compartí muchas risas y momentos gratos a lo largo de la maestría. Siempre fueron una parte positiva en tiempos difíciles.

Al Centro de Investigación Científica y de Educación Superior de Ensenada.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de maestría.

Tabla de contenido

Página

Tabla de contenido (continuación)

Lista de figuras

Figura Página

Lista de figuras (continuación)

Lista de tablas

Capítulo 1. Introducción

Una meta fundamental en la robótica es el desarrollo de robots completamente autónomos. El lograr la autonomía de los robots es de gran interés no sólo para la ciencia sino también para la industria, pudiendo (el robot) realizar tareas de manufactura, ensamblado, exploración, limpieza, manejo de residuos tóxicos, entre otros (Siegwart *et al.*, 2011). La planificación de movimiento surge a raíz de la necesidad de resolver este problema (autonomía del robot) (Latombe, 1991).

Contar con un algoritmo eficiente (tiempo, espacio) que realice una planificación de movimiento es medular para la autonomía del robot. Sin embargo, existen un gran número de aspectos que se deben considerar al realizar una planificación de movimientos, como generar una ruta de desplazamiento, evitar colisiones con otros objetos, respetar las restricciones de movimiento del robot, entre otros. Es por estos aspectos que aún para los planificadores de hoy en día, la autonomía del robot resulta una tarea difícil de lograr. Por lo tanto, la problemática general no debe verse como un solo problema sino como una colección de problemas (Latombe, 1991).

Los primeros trabajos se remontan a los años 60, desde modelos de cinemática de objetos (Hartenberg y Denavit, 1964) hasta un problema de movimiento y acomodo de muebles en espacios reducidos (Howden, 1968). Desde entonces, se optó por particionar el problema definiendo problemas más simples a partir de la problemática general. Se definió entonces el problema de la **planificación de movimientos básica** (Latombe, 1991), el cual pretende encontrar una secuencia continua de movimientos de un solo robot que describan su desplazamiento desde un punto α a un punto β en un espacio conocido y con obstáculos fijos. Se han propuesto tres distintos enfoques para afrontar el problema: basado en hoja de rutas (*roadmap* en inglés), basado en descomposición de celdas, y basado en campos de potencial (Latombe, 1991). Estos enfoques se estudiaron ampliamente en los años 80 (Sharir, 1989), cuando se propusieron algunos algoritmos que encuentran solución en tiempo polinomial (Latombe, 1991; Lavalle y Hutchinson, 1998). Sin embargo, en la práctica, numerosos sistemas requieren de la planificación de movimientos de múltiples objetos (robots).

Posteriormente, se definió la **planificación de movimiento multi-robot** (Parker, 2009). En esta extensión de la problemática básica se requiere de la planificación de movimientos de un conjunto de robots (no necesariemente iguales) cuyo desplazamiento deseado se encuentra dentro de los límites de un mismo espacio conocido. Como consecuencia de planificar en espacios con una mayor cantidad de robots, la complejidad del problema también es mayor. Esta vez son dos enfoques los que se propusieron para resolverlo: el enfoque **acoplado** y el enfoque **desacoplado** (Parker, 2009).

El enfoque acoplado considera al conjunto de robots como si fueran un solo robot compuesto para después utilizar alguno de los enfoques de la planificación de movimientos básica. Se considera robot compuesto a la colección de robots donde la posición y orientación de cada uno de los elementos (robots) que lo componen se especifican en todo momento de forma tal que no colisionan entre sí. En general, el enfoque acoplado entrega soluciones de “buena calidad”; más aún, garantiza una solución, si es que ésta existe; sin embargo, el tiempo requerido para encontrar una solución suele ser muy grande.

Como alternativa, el enfoque desacoplado sacrifica calidad de solución por eficiencia en tiempo. Los enfoques desacoplados buscan reducir el tiempo de cómputo de una solución lidiando con algunos aspectos del problema de manera independiente; sin embargo, no necesariamente encuentran una solución, incluso aunque ésta exista. Dos principales técnicas forman parte de este enfoque: la **planificación prioritaria** y la **coordinación de rutas**.

La coordinación de rutas la propusieron O’Donnell y Lozano-Pérez (1989) por primera vez, inspirados en un trabajo de control de acceso a bases de datos. En la coordinación de rutas se supone que las rutas de cada robot se planifican de manera independiente en una primera etapa, de modo que el recorrido simultáneo de dichas rutas no garantiza que los robots no colisionen entre sí. Se supone además que una ruta se encuentra particionada en segmentos de ruta; los segmentos pueden variar en longitud y tiempo de recorrido. El problema entonces se encuentra en determinar ese esquema de velocidades de cada robot a lo largo del recorrido de sus segmentos de ruta, de modo que no exista colisión entre robots.

La coordinación de rutas es el tema central de este trabajo de tesis. Una taxonomía detallada de la planificación de movimientos se describe en el Capítulo 2. En la sección subsecuente se define de manera general el problema de la coordinación de rutas, se discute el trabajo previo relevante, y se mencionan las variantes del problema que se estudian en este trabajo.

1.1. Coordinación de rutas

El problema de la coordinación de rutas se puede enunciar como: Considere un conjunto $A = \{A_1, A_2, \cdots, A_n\}$ de n robots (no necesariamente iguales), cuya forma geométrica es bien conocida, en un espacio R^N con $N = 2$ o 3. Primero, genérese las rutas de cada robot de manera independiente usando planificadores básicos (un sólo robot). Segundo, genérese un perfil de velocidades para cada robot a lo largo de su ruta de modo que que no exista colisión entre robots durante el recorrido simultáneo de sus respectivas rutas.

1.1.1. Trabajo previo relevante

Suponiendo que se relaja el problema de la planificación de movimientos especificando previamente la ruta que cada robot debe recorrer, entonces se tiene un problema de coordinación.

En (O'Donnell y Lozano-Pérez, 1989) se propone un método para la coordinación de dos robots construyendo un diagrama de coordinación que resalta aquellos segmentos de ruta tales que al ser recorridos de manera simultánea existe un riesgo potencial de colisión. Además, proponen un algoritmo voraz que maximiza el recorrido simultáneo de los segmentos de ambos robots.

En (Lavalle y Hutchinson, 1998) se estudia un problema similar, generalizando la coordinación para más de dos robots. Modelan el problema de la coordinación de rutas como un problema de teoría de juegos, donde se pretende encontrar una estrategia en la que cada robot minimice el tiempo perdido asociado con la coordinación. Proponen un algoritmo para encontrar un conjunto de estategias mínimas; sin embargo, la complejidad de dicho algoritmo es exponencial en el número de robots.

Posteriormente, en (Siméon *et al.*, 2002), se extiende la definición de diagrama de coordinación para modelar los recorridos de *n* robots a lo largo de sus respectivas rutas fijas. Dado el crecimiento exponencial en tamaño del diagrama de coordinación *n*-dimensional, proponen representarlo como $n(n-1)/2$ diagramas de coordinación simples, uno por cada pareja de robots. El principal objetivo de este trabajo es encontrar una coordinación libre de colisión, para los robots, si es que ésta existe.

En (Akella y Hutchinson, 2002) se enuncia un problema de optimización para la coordinación de rutas con trayectorias (ruta y perfil de velocidad) pre-especificadas, en donde únicamente el tiempo de inicio de cada ruta se puede modificar. Dicho problema de optimización se enuncia a continuación:

> Dado un conjunto de robots especificados con sus rutas y perfiles de velocidad sobre las rutas, encontrar el tiempo de inicio de recorrido para cada robot tal que el tiempo máximo de finalización del conjunto de robots sea mínimo, las restricciones de velocidad sobre las rutas se satisfagan y no ocurran colisiones.

Para resolver el problema de optimización, ellos elaboran una formulación de programación lineal entera mixta (PLEM) (Schrijver, 1998), con la cual garantizan una coordinación libre de colisión de tiempo mínimo para múltiples robots especificando únicamente el tiempo de inicio de cada ruta. Sin embargo, en su trabajo se menciona también que el problema es *NP-difícil*. Incluso se menciona su similitud con el problema conocido como *No-wait Job Shop Scheduling* (Sahni y Cho, 1979).

Posteriormente, en (Peng y Akella, 2005), se propone un modelo para generar perfiles de velocidad libres de colisión, para cada robot, a lo largo de rutas fijas con restricciones *cinemáticas* (velocidad, aceleración), minimizando el tiempo de recorrido. Ellos se apoyan de dos formulaciones de PLEM para calcular la cota superior e inferior del valor de una solución óptima factible.

Otros trabajos más recientes estudian el problema de la coordinación de rutas con restricciones adicionales, como restricciones de rango de distancia entre robots (Abichanda-

ni *et al.*, 2008), o la coordinación de rutas tomando en cuenta eventos inesperados (Olmi *et al.*, 2008).

1.1.2. Variantes del problema

La coordinación de rutas consta de dos etapas. La primera etapa consiste en calcular la ruta de cada robot de manera independiente a los demás robots. La segunda etapa consiste en determinar un perfil de velocidades de modo que cada robot complete su ruta sin que ocurra colisión entre ellos. Existen variantes del problema de la coordinación de rutas, algunas de éstas se explican a continuación.

Una variante del problema se conoce como la coordinación de rutas fijas. Una vez que las rutas construidas en la primera etapa se han determinado, éstas no se pueden alteradar durante la segunda etapa.

Existen otras variantes del problema cuando se añaden restricciones de algún tipo, por ejemplo restricciones de velocidad, restricciones de aceleración o restricciones de movimiento. Las restricciones de velocidad acotan el movimiento de uno o varios robots a un cierto límite de velocidad v_{max}. Las restricciones de aceleración acotan el movimiento de uno o varios robots a un cierto límite de aceleración a_{max}. Las restricciones de movimiento limitan el movimiento de uno o varios robots, por ejemplo que los robots no puedan realizar movimientos laterales. Cada una de estas variantes supone un mayor o menor grado de complejidad al problema.

En este trabajo de tesis se supone que la trayectoria (ruta y perfil de velocidad) de cada robot se ha pre-especificado. Se supone que el perfil de velocidad es constante y está acotado por v_{max}. Se supone también que la aceleración es infinita, es decir, un robot puede alcanzar v_{max} partiendo del reposo de manera instantánea y viceversa. Además, se supone que las rutas se encuentran previamente particionadas en segmentos de ruta donde los puntos de inicio y final de cada segmento son libres de colisión. Se supone que una vez que un robot inicia a recorrer un segmento de ruta, el robot no se debe detener hasta completar dicho segmento. Finalmente, se supone que los movimientos del robot son monotónicos, es decir, que el robot no puede retroceder en ningún momento a lo largo de su ruta. Se considera una versión de optimización del problema, donde se desea

minimizar el máximo tiempo de finalización entre todos los robots.

1.2. *Job Shop Scheduling*

El *Job Shop Scheduling* (JSP) es un problema de calendarización conocido en investigación de operaciones y ciencias de la computación. Se ha estudiado ampliamente a lo largo de las últimas décadas. El problema consiste en procesar un conjunto de tareas abstractas en máquinas compartidas de modo que el máximo tiempo de finalización entre todas las tareas sea mínimo. Las tareas se componen de una secuencia fija de operaciones, secuencia que, no necesariamente es la misma para cada tarea. Dos restricciones se deben cumplir, la primera, una relación de precedencia entre las operaciones en cada tarea se debe respetar, la segunda, una máquina sólo puede procesar una operación a la vez.

El JSP recibe como entrada un conjunto de n tareas $\{j_1, j_2, \cdots, j_n\}$ y un conjunto $M = \{\mu_1, \mu_2, \cdots, \mu_m\}$ de m máquinas. Cada tarea j_i, $i \in \{1, 2, \cdots, n\}$, se compone por una secuencia fija de m operaciones $j_i = \{o_{i1}, o_{i2}, \cdots, o_{im}\}$. Una operación cualquiera o_{ij} requiere que se procese por una máquina en específico del conjunto $\{\mu_1, \mu_2, \cdots, \mu_m\}$. El tiempo de procesamiento de la operación o_{ij} se denota como p_{ij} y es dependiente de la tarea y de la máquina que lo procesa. Por ejemplo, los tiempos de procesamiento de dos operaciones distintas que se procesan en una misma máquina pueden ser, arbitrariamente, iguales o distintos. Una máquina μ_x cualquiera, de M, sólo puede procesar una operación a la vez, de modo que si μ_x se encuentra procesando alguna operación o_{ij} y se requiere procesar otra operación distinta o_{pq} en μ_x, ésta deberá aguardar hasta que μ_x haya concluído el procesamiento de o_{ij}. Además, para toda tarea j_i, ninguna operación $o_{i(j+1)}$ deberá procesarse antes de la finalización la operación o_{ij}. Finalmente, se debe considerar que una máquina no puede interrumpir el procesamiento una operación hasta haber concluido. Sea C_i el tiempo de finalización de la tarea j_i, el tiempo de finalización máximo se define como $C_{max} = \max_{i \in \{1,2,\cdots,n\}}(C_i)$. El objetivo es minimizar el máximo tiempo de finalización C_{max}, también conocido como *makespan*.

En la literatura se pueden encontrar numerosos trabajos que intentan resolver el problema. En (Adams *et al.*, 1988) se propone una heurística conocida como *shifting bottle-*

neck. En este método, una a una, se describe la secuencia de operaciones a procesar por cada máquina, describiendo siempre la secuencia de aquella máquina identificada como el cuello de botella entre aquellas máquinas que aún no se han secuenciado. En (Binato *et al.*, 2002), se describe una heurística que incorpora un método que se conoce como *GRASP* que consta de dos fases: una fase de construcción y una de búsqueda local. En la primera fase, se construye una solución factible un elemento a la vez. En cada iteración de la primera fase, se determina el siguiente elemento a añadir a la solución. Los elementos se encuentran ordenados con respecto a una función de elección voraz que mide el beneficio de un elemento al ser agregado a la solución. Posteriormente, mediante un procedimiento aleatorio se elige, con cierta probabilidad de acuerdo a su posición en el ordenamiento, el siguiente elemento que se agrega a la solución. En la segunda fase, se realiza una búsqueda local sobre la solución construida en al primera fase. En (Gonçalves *et al.*, 2005), se propone un algoritmo genético que realiza un procedimiento de búsqueda local en cada nuevo cromosoma producido dentro del proceso evolutivo. En (Pardalos y Shylo, 2006), se propone utilizar una técnica que se conoce como *GES* en la cual un subconjunto de soluciones conocidas se usan para generar nuevas soluciones en etapas subsecuentes.

1.3. Objetivos

1.3.1. Objetivo general

Diseñar algoritmos determinísticos para el problema de la coordinación de rutas de robots en su versión de optimización y formalizar la relación de este problema con el problema conocido como *Job Shop Scheduling*.

1.3.2. Objetivos específicos

1. Diseñar algoritmos determinísticos para el problema de la coordinación de rutas de robots en su versión de optimización.

2. Definir un conjunto de casos de prueba con características específicas que sirva como *benchmark* para el problema de la coordinación de rutas en su versión de optimización.

3. Determinar la relación que existe entre las características de los casos del problema de coordinación de rutas y el problema conocido como *Job Shop Scheduling*.

1.4. Propuesta de solución

Para un modelo donde la aceleración de los robots no está acotada, la velocidad es constante y donde las áreas de inicio y final de cada segmento de ruta son libres de colisión, en este documento se propone realizar un procedimiento iterativo de agrupamiento y coordinación de movimientos de un número constante (k) de robots. En cada iteración se propone realizar n/k agrupaciones de k robots y construir n/k diagramas de coordinación k-dimensionales, utilizando una versión generalizada de los diagramas de coordinación de (O'Donnell y Lozano-Pérez, 1989; Siméon *et al.*, 2002), donde n es el número de robots y k es la constante de agrupamiento. Una vez encontrado un calendario de coordinación que armonice los movimientos y minimice el máximo tiempo de terminación de recorrido de los k robots en una agrupación (por cada agrupación), éstos, junto con el calendario de coordinación, se consideran como un robot compuesto cuyos movimientos no provocan una colisión consigo mismo. Posteriormente, se repite el procedimiento a menos que el número de robots compuestos sea uno; en cuyo caso, éste está coordinado.

De esta manera se evita el costo exponencial de realizar un calendario de coordinación *n*-dimensional particionándolo en n/k diagramas de coordinación *k*-dimensionales. Además, se evita resolver un problema de PLEM como en (Akella y Hutchinson, 2002; Peng y Akella, 2005), para encontrar una solución al problema.

Finalmente, se propone utilizar esta técnica de agrupamiento de robots para resolver casos de prueba conocidos para el JSP.

1.5. Organización de la tesis

El resto del presente trabajo de tesis se organiza como sigue. En el Capítulo 2 se describe una taxonomía de la planificación de movimientos de robot, así como los enfoques clásicos existentes. En el Capítulo 3 se hace una descripción breve de las definiciones básicas que se utilizan y una definición formal del problema de la coordinación de rutas.

También se hace un análisis de la complejidad del problema y se definen dos cotas triviales para el óptimo del mismo. En el Capítulo 4 se propone una técnica de agrupamiento de robots, se describen tres algoritmos determinísticos usando esta técnica de agrupamiento, y se realiza un análisis de éstos. En el Capítulo 5 se describen los experimentos realizados y se presentan los resultados obtenidos. Por último, el Capítulo 6 contiene un sumario del trabajo de investigación realizado, se presentan las conclusiones de esta investigación, y se sugiere hacia dónde se puede encaminar el trabajo futuro.

Capítulo 2. Planificación de movimientos

Para lograr la autonomía de los robots es necesario resolver diferentes problemas fundamentales. La planificación de movimientos es uno de ellos. En la planificación de movimientos se debe generar una trayectoria libre de colisión. Una trayectoria de un objeto (robot) requiere de la descripción de dos elementos, una ruta y los movimientos en función del tiempo a lo largo de la ruta. Las colisiones ocurren cuando existe una superposición de objetos (robot-obstáculo, robot-robot) en un mismo punto en el espacio en un mismo instante de tiempo.

Esto sugiere que la planificación de movimientos no debe verse como un solo problema sino como un conjunto de problemas que incluye la planificación de rutas y la planificación de velocidades. Si bien el problema de la planificación de movimientos con un solo robot y con obstáculos estáticos admite algoritmos de tiempo polinomial, en (Reif y Sharir, 1994) se demuestra que con múltiples objetos (obstáculos o robots) en movimiento el problema se considera intratable.

El resto de este capítulo plantea la definición del problema de la planificación de movimientos básica así como la definición del problema con múltiples objetos en movimiento. Este capítulo contiene, además, una taxonomía del problema y los enfoques existentes para resolverlo.

2.1. Planificación de movimientos básica

En la planificación de movimientos básica se simplifican aspectos del problema inherentes en la práctica, para así definir el problema en un ambiente controlado. En esta simplificación del problema se supone un espacio euclidiano de dos o tres dimensiones, denominado espacio de trabajo W, donde residen un solo objeto rígido (robot) A en movimiento y un conjunto de objetos rígidos (obstáculos) B estáticos. Una configuración q es un vector en $\mathbf{R}^m$ que determina la posición y orientación de A con respecto a un marco de referencia fijo en W. La región que ocupa A en W, dada un configuración q, se denota como $A(q)$. El objetivo es encontrar una interpolación de configuraciones entre una configuración inicial q_{init} y una configuración final q_{goal}, tal que no exista colisión entre el robot

y los obstáculos. En esta sección se enuncia la definición del problema de la planificación de movimientos básica, al igual que los enfoques utilizados para estudiar el problema.

Haciendo una adaptación de (Latombe, 1991) se define la planificación de movimientos básica como:

> Sea $W \subseteq \mathbf{R}^N$ un espacio de trabajo, con $N = 2$ o 3. Sea $A \subset W$ un robot que se desplaza en W. Sea $B \subset W$ un conjunto de obstáculos fijos distribuidos en W. Suponga que tanto A como cada elemento en B son objetos rígidos cuyo tamaño se conoce de manera exacta.
>
> Dado un par de configuraciones inicial q_{init} y final q_{goal} de A en W, encontrar una ruta τ que describa una secuencia continua de configuraciones q de A que eviten el contacto con cualquier elemento en B. Indicar el fallo de no existir dicha ruta.

Al conjunto de configuraciones q de A se le denota como *espacio de configuración* $C \subseteq \mathbf{R}^m$, con $m = 3$ (si $N = 2$) o $m = 6$ (si $N = 3$). En este espacio de configuraciones, residen tanto el conjunto de configuraciones libres de colisión C_{free} como el conjunto de configuraciones donde ocurre colisión C_{obs}. Para cualquier configuración q en un espacio C_{free}, el espacio en W ocupado por $A(q)$ se encuentra libre de colisión; mientras que, para cualquier configuración q en un espacio C_{obs}, $A(q)$ se encuentra en colisión con algún obstáculo en B. Formalmente

$$C_{obs} = \{q \in C \mid A(q) \cap \bigcup_{B_i \in B} (B_i) \neq \emptyset\} \quad \text{y} \tag{1}$$

$$C_{free} = C \setminus C_{obs}. \tag{2}$$

En las siguientes subsecciones se describen brevemente los tres enfoques principales existentes en la literatura (Latombe, 1991) que estudian la planificación de movimientos básica: los basados en hojas de ruta (*roadmaps*), los basados en descomposición de celdas y los basados en campos de potenciales.

2.1.1. Hoja de ruta (*Roadmap*)

En este enfoque se construye un grafo topológico $G(V, E)$ de C_{free}, llamado *hoja de ruta*, donde V representa un conjunto de vértices en C_{free} y E representa un conjunto de aristas que los unen. Cada arista $e \in E$ une un par de vértices $v, v' \in V$ siempre y cuando exista una ruta descrita por una curva de configuraciones q en C_{free}. Entonces, de ser posible conectar la configuración de inicio q_{init} y final q_{goal} de un robot A al grafo G (*i.e.* $\exists v, v' \in V \mid (q_{init}, v), (v', q_{goal}) \in E$), basta con utilizar algún algoritmo de búsqueda en grafos para encontrar una solución factible si es que ésta existe en G.

Una vez conectados q_{init} y q_{goal} al grafo G, se calcula una *caminata* $q_{init} - q_{goal}$ que las una. Dado que cada arista $e \in E$ describe una curva de configuraciones entre un par de vértices $v, v' \in V$ cuyos puntos se asocian a configuraciones en C_{free}, entonces al concatenar las aristas de una *caminata*, éstas se pueden transformar en una secuencia de configuraciones en C_{free}.

El objetivo principal de este enfoque es construir la hoja de ruta, para ello existen distintos métodos como el **grafo de visibilidad** (Siméon *et al.*, 2000) o el **diagrama de Voronoi** (Leven y Sharir, 1987) entre otros. Las Figuras 1 y 2 ilustran un ejemplo de un grafo de visibilidad y unejemplo de un diagrama de Voronoi, respectivamente. El área sombreada representa el espacio de configuraciones C_{obs} y el área vacía representa el espacio de configuraciones C_{free}, los grafos son la *hoja de ruta* cuyos elementos representan una muestra de las configuraciones en C_{free}.

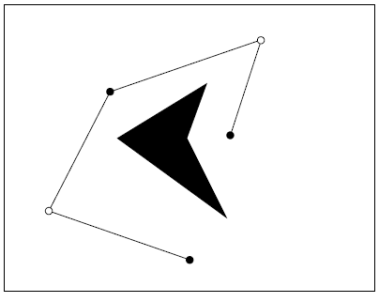


Figura 1: Grafo de visibilidad de un espacio de configuración.

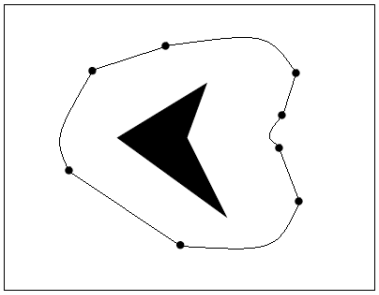


Figura 2: Diagrama de Voronoi de un espacio de configuración.

2.1.2. Descomposición de celdas

En este enfoque se realiza una segmentación del espacio de configuración C en polígonos de igual o distinta geometría, llamados celdas, cuyos bordes están en contacto pero no se traslapan. Posteriormente, se construye un grafo $G(V, E)$ donde V representa un conjunto de vértices y E un conjunto de aristas. Un vértice $v \in V$ representa una celda c siempre y cuando no exista obstáculo alguno (o parte de uno) en su interior, es decir, $\forall q \in c$, $A(q) \cap \bigcup_{B_i \in B}(B_i) = \emptyset$. Una arista une un par de vértices v y v' siempre y cuando sus celdas correspondientes c y c' sean adyacentes entre sí. De este modo, el grafo G contiene únicamente elementos que se traducen a elementos en C_{free}, excluyendo a C_{obs}.

De manera similar al enfoque de hoja de ruta, una vez construido el grafo G se puede utilizar un algoritmo de búsqueda en grafos para obtener una solución factible si es que ésta existe en G. Una solución factible sería una secuencia de celdas (vértices) que conecte la celda que contenga a q_{init} con la celda que contenga a q_{goal}. A dicha secuencia se le llama *canal*.

En la *descomposición de celdas*, una vez obtenido el canal, se debe poder recuperar una ruta τ del canal. Para ello, las celdas deben cumplir con dos características:

1. La geometría de cada celda debe permitir calcular una ruta entre cualesquiera dos configuraciones dentro de la celda.

2. Debe existir al menos una ruta que describa las configuraciones del robot a través de los límites entre dos celdas adyacentes cualesquiera.

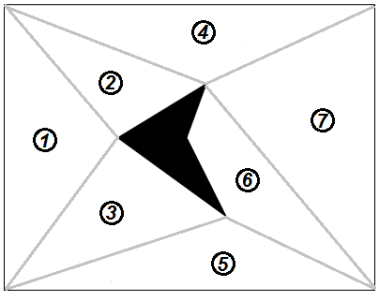


Figura 3: Espacio de configuración C_{free} descompuesto en celdas exactas.

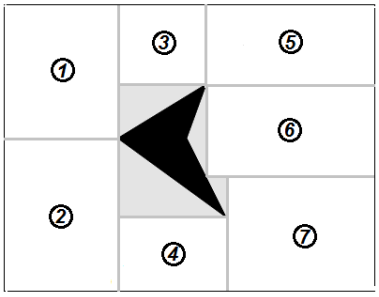


Figura 4: Espacio de configuración C_{free} descompuesto en celdas aproximadas.

Al descomponer el espacio de configuración C en celdas cuyo espacio de configuracion es libre de colisión (C_{free}), se particiona la dificultad de calcular una sola ruta a través de C en calcular $O(|V|)$ rutas de $O(|V|)$ fragmentos de C. Observe que se puede descomponer C en una sola celda y esto equivale a resolver el problema entero. La descomposición de celdas puede ser exacta, como se muestra en la Figura 3, o aproximada, como se muestra en la Figura 4.

2.1.3. Campos de potenciales

En este enfoque se modela el espacio de configuraciones C como un *campo de potencial artificial* U, mientras que las configuraciones q de un robot A se modelan como una partícula afectada por U. En U se define una función de potencial como la suma de potencial de atracción y potencial de repulsión. El potencial de atracción arrastra la partícula a una configuración final q_{goal} mientras que el potencial de repulsión empuja la partícula fuera de las configuraciones donde ocurre colisión $q \notin C_{obs}$.

Dada una configuración q de A, la función de potencial de U se define como

$$U(q) = U_{att}(q) + U_{rep}(q),$$

donde $U_{att}(q)$ y $U_{rep}(q)$ denotan el potencial de atracción y el potencial de repulsión, respectivamente. La definición de las funciones de potencial $U_{att}(q)$ y $U_{rep}(q)$ pueden variar según el autor y por lo general $U_{att}(q), U_{rep}(q) \in \mathbf{R}^+ \cup \{0\}$. El valor de $U_{att}(q)$ cambia en función de la distancia euclidiana entre q y q_{goal}, a menor distancia, menor atracción y, a mayor distancia, mayor atracción, siendo $U_{att}(q) = 0$ cuando $q = q_{goal}$. El valor de $U_{rep}(q)$ cambia en función de la distancia euclidiana entre q y las configuraciones $q' \in C_{obs}$, a menor distancia, mayor valor de repulsión y, a mayor distancia, menor valor, siendo $U_{rep}(q) = \infty$ cuando $q \in C_{obs}$.

En este enfoque la construcción de una secuencia de configuraciones es iterativa. En una iteración, dada una configuración q, se calcula el valor de potencial $U(q)$ correspondiente. Posteriormente, se elige una diferencia de configuración Δq tal que $q' = q - \Delta q$ minimice $U(q')$. La Δq debe ser lo suficientemente pequeña para que las configuraciones obtenidas por la interpolación entre q y q' pertenezcan también a C_{free}. En la siguiente iteración se reemplaza la configuración q por q' y se repite el procedimiento a menos que $q' = q_{goal}$.

2.2. Planificación de movimientos con múltiples objetos en movimiento

La planificación de movimiento con múltiples objetos en movimiento es una extensión del problema básico, la cual considera varios objetos con movimiento en un mismo ambiente además del robot A. A diferencia del problema básico (donde los obstáculos permanecen fijos en W), la planificación de movimientos, donde existen otros objetos en W, además de A, cuya posición es dependiente del tiempo supone un reto mayor. Los objetos en movimiento en W pueden ser obstáculos comunes, de los cuales no se tiene control de sus movimientos, o robots adicionales que también requieren de una planificación de movimientos para alcanzar su objetivo. En esta sección se enuncia el problema de planificación de movimientos con múltiples objetos en movimiento, así como los distintos enfoques y las técnicas propuestas para enfrentarlo.

2.2.1. Con múltiples obstáculos en movimiento

En esta extensión de la planificación de movimientos básica, en un espacio euclidiano denominado espacio de trabajo W donde existe un solo robot A cuyos movimientos se deben planificar. Existe también un conjunto de m obstáculos en movimiento B en W, donde la trayectoria de cada $B_i \in B$ se encuentra bien definida. Los movimientos de B en W son independientes de los movimientos de A en W.

Para modelar el problema, se agrega una dimensión más al espacio de configuraciones C, el tiempo t. Por ejemplo, sea A un robot en movimiento, con orientación fija, en un espacio de trabajo $W \subseteq \mathbf{R}^2$, entonces una configuración q de A se define como la tripleta (x, y, t). Sea B un conjunto $\{B_1, B_2, \cdots, B_m\}$ de m obstáculos en W cuya trayectoria es conocida. Dado que no se tiene control sobre el movimiento de los obstáculos en B, la posición de cada obstáculo $B_i \in B$ en un tiempo t está dada por $B_i(t)$.

Sea C el espacio de configuración de A en W. Para modelar el espacio de configuración donde ocurre colisión C_{obs} con obstáculos dinámicos, se hace una adaptación de la Ecuación 1, quedando ésta como muestra la Ecuación 3

$$C_{obs}(t) = \{q \in C \mid A(q) \cap \bigcup_{B_i \in B} (B_i(t)) \neq \emptyset\}. \tag{3}$$

Similarmente, para modelar el espacio de configuración libre de colisión C_{free} se hace una adaptación de la Ecuación 2, quedando como indica la Ecuación 4

$$C_{free}(t) = C \setminus C_{obs}(t). \tag{4}$$

Observe entonces que el espacio de configuraciones C no cambia, lo que cambia es el espacio C_{obs} en C con respecto al tiempo producido por los movimientos de los obstáculos en W. En la Figura 5 se muestra una gráfica del cambio del espacio que ocupa C_{obs} en C producido por un movimiento arbitrario de un obstáculo en W a lo largo del tiempo.

Una vez modelado el problema de esta forma, cualquier técnica usada para el problema básico (Sección 2.1) se puede usar de manera directa para encontrar una trayectoria para A si es que ésta existe. En el problema básico, la solución estaba limitada a encon-

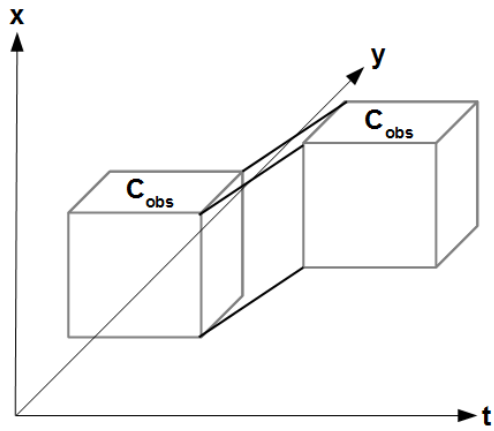


Figura 5: Espacio de configuración C a lo largo del tiempo.

trar una ruta τ que conecte un par de configuraciones q_{init} y q_{goal}. La diferencia entre una ruta y una trayectoria es el tiempo. Una trayectoria es una ruta más un perfil de velocidad que se debe cumplir a lo largo de una ruta. Por ejemplo, en $W \subseteq \mathbf{R}^2$, una configuración q en una ruta contiene elementos que denotan posición x, y u orientación ϕ, mientras que una configuración q en una trayectoria debe contener además al menos un elemento que indique el tiempo t en el que los otros (posición y orientación) se deben cumplir.

La planificación de movimientos con múltiples obstáculos en movimiento queda fuera del estudio de esta tesis, en este trabajo se considera que no existen otros objetos en movimiento en el espacio de trabajo además de los robots.

2.2.2. Con múltiples robots en movimiento

En la Sección 2.1 se define el espacio de configuración para un robot en un ambiente donde ningún otro objeto se encuentra en movimiento además de él. En este problema se encuentran n robots en movimiento cuya trayectoria se requiere planificar. En esta extensión del problema básico, existe un conjunto $A = \{A_1, A_2, \cdots, A_n\}$ de n robots con movimiento en un espacio de trabajo W. Además del conjunto A, existe un conjunto de m objetos estacionarios $B = \{B_1, B_2, \cdots, B_m\}$ en W denominados obstáculos. El objetivo es encontrar una trayectoria libre de colisión desde una configuración inicial q_{init} hasta una configuración final q_{goal} para cada robot A_i en A. Para esto, se necesita redefinir el espacio de configuración, uno para cada robot, que tome en cuenta la posición de los otros robots a lo largo del tiempo.

Sea C_i el espacio de configuración del robot A_i, donde el tiempo t forma parte de los elementos (posición y orientación) que conforman q_i en C_i, *i.e.* $q_i = (x, y, \cdots, \phi, \cdots, t)$. Se define el espacio de configuración donde ocurre colisión $C_{i_{obs}}$ como

$$C_{i_{obs}} = \left\{ q_i \in C_i \mid A_i(q_i) \cap \left(\bigcup_{k \in \{1,2,\cdots,m\}} B_k \ \cup \bigcup_{j \in \{1,2,\cdots,n\}, i \neq j} A_j(q_j) \right) \neq \emptyset \right\}. \tag{5}$$

El espacio de configuración libre de colisión $C_{i_{free}}$ se define como

$$C_{i_{free}} = C_i \setminus C_{i_{obs}}. \tag{6}$$

A la tupla de configuraciones $q = (q_1, q_2, \cdots, q_n)$ de cada robot en A se le llama configuración compuesta. De modo que $A(q) = A_1(q_1) \cup A_2(q_2) \cup \cdots \cup A_n(q_n)$. Se define entonces el espacio de configuración compuesto C como el producto cartesiano de los espacios de configuración $C_1, C_2, \cdots, C_n$ correspondientes a cada robot en A, formalmente $C = C_1 \times C_2 \times \cdots \times C_n$. Para C el espacio C_{obs} se define como

$$C_{obs} = \{q = (q_1, q_2, \cdots, q_n) \in C : q_i \in C_{i_{obs}}\}, \tag{7}$$

mientras que el espacio C_{free} se define como

$$C_{free} = C \setminus \bigcup_{i \in \{1,2,\cdots,n\}} (C_{i_{obs}}) = \bigcup_{i \in \{1,2,\cdots,n\}} (C_{i_{free}}). \tag{8}$$

Observe que este modelo permite que la planificación de movimientos de múltiples robots sobre un mismo espacio de trabajo W sea equivalente a la planificación de movimientos de un solo robot compuesto en W.

2.2.2.1. Enfoque acoplado

El enfoque acoplado busca resolver el problema de la planificación de movimientos de múltiples robots considerando un conjunto de n robots $\{A_1, A_2, \cdots, A_n\}$ como un solo robot compuesto A cuyas configuraciones se encuentran definidas en un espacio de configuración compuesto C. De este modo, es posible utilizar de manera directa cualquie-

ra de las técnicas descritas en la Sección 2.1. La ventaja de este enfoque es que tiene la capacidad de encontrar una solución al problema si es que ésta existe. Sin embargo, la complejidad del problema crece exponencialmente con el número de dimensiones del espacio de configuración C_i de cada robot A_i. Sea $|q_i|$ el número de elementos en una configuración arbitraria, la dimensión del espacio de configuración compuesto C está definida por $dim(C) = \sum_{q_i \in q} |q_i|$. Incluso para un caso que involucre dos robots ($n = 2$) con orientación fija en un espacio de trabajo W de dos dimensiones, la dimensión del espacio de configuración compuesto C de ambos robots es $dim(C) = \sum_{q_i \in q} |q_i| = 4$. Es por ello que, aunque diversas propuestas han tratado de resolver el problema bajo este enfoque, la problemática general se considera intratable (Parker, 2009).

2.2.2.2. Enfoque desacoplado

El enfoque desacoplado surge como alternativa para lidiar con la complejidad de la problemática general. En este enfoque se sacrifica calidad de solución por eficiencia en tiempo. Su cualidad es descomponer el problema general en dos, la planificación de las rutas y el manejo de colisiones, y resolverlos de manera independiente.

El enfoque desacoplado suele dividirse en dos categorías: la planificación prioritaria y la coordinación de rutas. En la **planificación prioritaria**, los movimientos de cada robot se planifican secuencialmente de acuerdo a un orden prioritario. El robot con mayor prioridad planifica sus movimientos antes que aquellos con menor prioridad; mientras que aquellos con menor prioridad planifican sus movimientos evitando colisionar con los robots que planificaron sus rutas antes que ellos. En la **coordinación de rutas**, se calculan las rutas de cada robot de manera independiente. Posteriormente, para cada robot se planifica un perfil de velocidades a lo largo de su ruta a modo de evitar alguna colisión inherente al recorrido en paralelo de las rutas de los robots en un mismo ambiente. Las siguientes subsecciones describen cada enfoque con mayor detalle.

Planificación prioritaria. Erdmann y Lozano-Pérez (1987) propusieron la planificación prioritaria. En su trabajo describen una técnica para realizar una planificación de movimiento para múltiples robots, asignando una prioridad a cada uno y planificando en orden de acuerdo a su prioridad.

Sea A un conjunto $\{A_1, A_2, \cdots, A_n\}$ de n robots. Se asigna una prioridad $\phi(A_i)$ a cada robot en A. Sea A_π un conjunto ordenado que contiene cada robot en A, donde $A_{\pi(i)} \leq A_{\pi(j)}$ si y sólo si $\phi(A_i) \leq \phi(A_j)$, con $A_{\pi(i)}, A_{\pi(j)} \in A_\pi$. Se planifican los movimientos de cada robot en A_π, uno a uno, de acuerdo a su orden de prioridad. El robot $A_{\pi(1)}$ planifica su trayectoria $\tau_{\pi(1)}$ como si no hubieran otros obstáculos (robots) en movimiento en W. El robot $A_{\pi(2)}$ planifica su trayectoria $\tau_{\pi(2)}$ tomando en cuenta la trayectoria recién calculada del robot $A_{\pi(1)}$ como un obstáculo en movimiento. Así sucesivamente hasta que el robot $A_{\pi(n)}$ planifique su trayectoria $\tau_{\pi(n)}$ considerando $n-1$ obstáculos en movimiento en W. Observe que este enfoque descompone el problema de la planificación de movimientos de múltiples robots, en n problemas de planificación de movimientos de un robot con múltiples obstáculos en movimiento, como se describe en la Sección 2.2.1.

Coordinación de rutas. O'Donnell y Lozano-Pérez (1989) propusieron la coordinación de rutas de robots. En su trabajo se describe una técnica para encontrar una planificación de movimientos para un par de robots, suponiendo que se hizo previamente una planificación de sus rutas por separado, para después calendarizar los movimientos de los robots sobre sus rutas para que no exista colisión.

Sea A un conjunto $\{A_1, A_2, \cdots, A_n\}$ de n robots. La coordinación de rutas planifica la ruta τ_i correspondiente a cada robot A_i de manera independiente. Esto se puede hacer con cualquier técnica usada en la planificación de movimientos básica (Sección 2.1). Posteriormente, se hace una calendarización de las configuraciones de cada ruta τ_i de A_i sujeto a que $A_i(q_i) \cap A_j(q_j) = \emptyset$, con $i, j \in \{1, 2, \cdots, n\}$, $i \neq j$.

En este capítulo se hizo una breve descripción de la planificación de rutas y los distintos enfoques existentes, en la Figura 6 se muestra de manera gráfica la taxonomía del problema. La coordinación de rutas es el tema central de esta tesis y en el siguiente capítulo se define a detalle el problema a tratar y sus variantes.

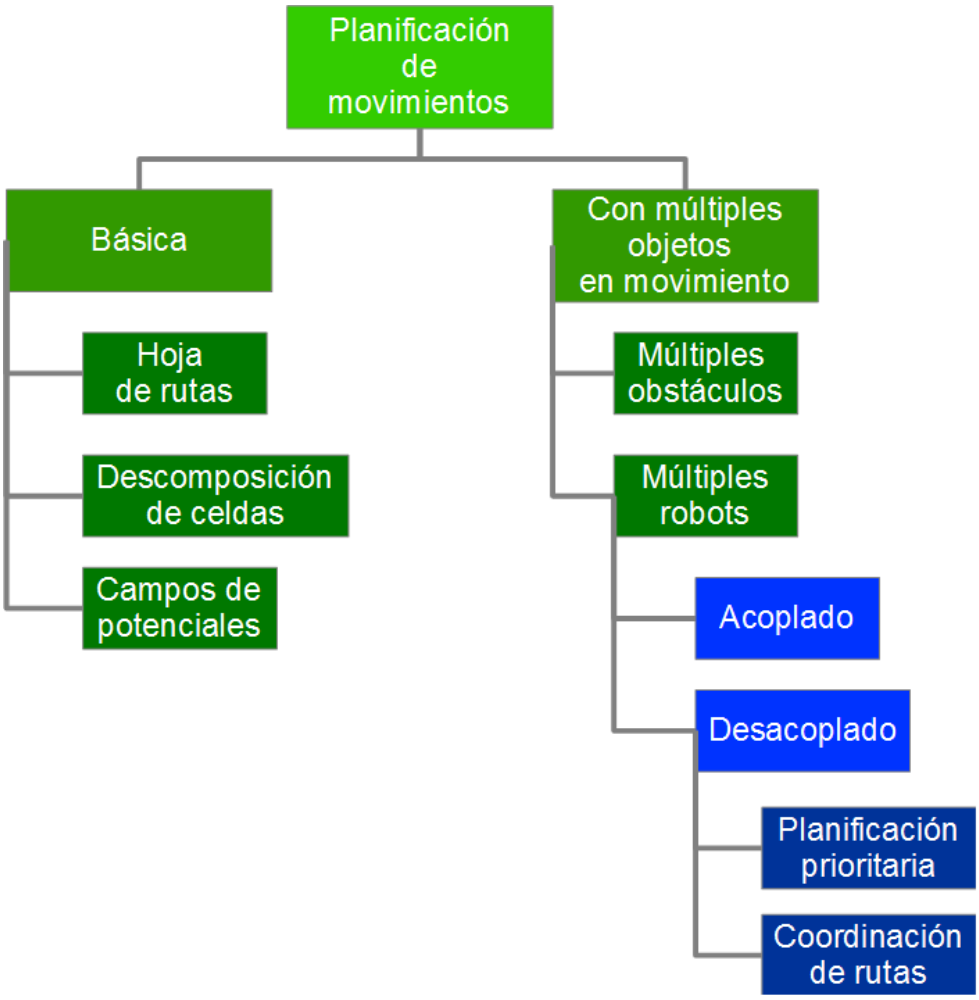


Figura 6: Taxonomía de la planificación de movimientos.

Capítulo 3. Coordinación de rutas de robots con trayectorias pre-especificadas

En el capítulo anterior se describió brevemente lo que es la planificación de movimientos, los problemas que se derivan de ésta y algunos enfoques propuestos. En este capítulo se hace una enunciación breve de la notación y definiciones básicas que se usan en la coordinación de rutas y se describe formalmente el problema. También se analiza la complejidad del problema y se definen dos cotas triviales para el óptimo del problema.

3.1. Notación y definiciones básicas

En la coordinación de rutas de robots, a cada robot le corresponde una ruta pre-especificada, que a su vez se encuentra previamente segmentada. El número de robots se denota por n, mientras que el número de segmentos en los que se divide cada ruta i se denota por m_i, con $i \in \{1, 2, \cdots, n\}$. Si bien el número de segmentos en cada ruta puede ser distinto, sin pérdida de generalidad, salvo que se indique lo contrario, se supone que cada ruta está dividida en m segmentos. El resto de la sección contiene una descripción de la notación usada en este trabajo.

Robots (A). Se denota $A = \{A_1, A_2, \cdots, A_n\}$ como un conjunto de n robots puntuales, es decir, los robots carecen de tamaño y orientación.

Rutas (τ). Se denota $\tau = \{\tau_1, \tau_2, \cdots, \tau_n\}$ como un conjunto de rutas pre-especificadas; siendo τ_i la ruta correspondiente al robot A_i. En el Capítulo 2 se habla de una ruta como una secuencia de configuraciones en un espacio de coordinación. En la coordinación de rutas se supone que cada ruta se ha generado previamente. Por lo tanto, independientemente de la geometría de la ruta, es posible definir una ruta τ_i como una secuencia de regiones por donde el robot A_i debe atravesar para completar su movimiento. A cada elemento de la secuencia se le denomina segmento de ruta.

Regiones (R). Se denota R como una clase de regiones $\{r_1, r_2, \cdots, r_m\}$. Una región r_i, identifica el área por donde atraviesan uno o más segmentos de ruta.

Segmentos de ruta (σ_{ij}). Cada ruta τ_i se divide en segmentos σ, de modo que $\tau_i = \{\sigma_{i1}, \sigma_{i2}, \cdots, \sigma_{im_i}\}$. Cada segmento σ_{ij} determina la región a la que pertenece, es decir, $\sigma_{ij} \in r_p$, $r_p \in R$. Ninguna combinación de robots A_i, A_j ($i \neq j$) podrá recorrer segmentos correspondientes σ_{ix}, σ_{jy} de manera simultánea que residan en la misma región, i.e., $\sigma_{ix}, \sigma_{jy} \in r_p$, en caso contrario podría ocurrir una colisión. Además existe una restricción de precedencia, para todo i, ningún segmento $\sigma_{i(j+1)}$ o posterior se puede recorrer antes que el segmento σ_{ij}.

Tiempo de recorrido (tr_{ij}). Se define tiempo de recorrdio tr_{ij} como el tiempo que le toma a un robot A_i recorrer su segmento de ruta σ_{ij}.

Diagrama de coordinación (CD). Considere el diagrama de coordinación CD (O'Donnell y Lozano-Pérez, 1989; Latombe, 1991) de A_i y A_j, mostrado en la Figura 7(a), como el producto cartesiano del conjunto de índices más uno de las rutas τ_i y τ_j, es decir, $CD_{ij} = \{(x_1, x_2) : x_1 \in \{0, 1, \cdots, m_i\}$ y $x_2 \in \{0, 1, \cdots, m_j\}\}$. A cada par (x_1, x_2) en un CD_{ij} se le denomina vértice de coordinación, de modo que CD_{ij} está conformado por $(m_i + 1) \cdot (m_j + 1)$ vértices de coordinación. Por simplicidad, ahora en adelante, entiéndase el término vértice como vértice de coordinación. Un vértice $v = (x_1, x_2)$ se encuentra en un espacio $\{0, 1, \cdots, m\}^2$, donde cada coordenada x_i denota que el robot A_i ha concluído el recorrido del segmento σ_{ix_i} de su ruta. Al vértice origen se le denomina como $v_0 = (0, 0)$, y representa a los robots A_i y A_j que se encuentran en espera de recorrer su primer segmento de ruta. Por otro lado, el vértice destino se denomina $v_f = (m_i, m_j)$, e indica que ambos robots A_i y A_j han finalizado el recorrido de su último segmento y por lo tanto su ruta. Observe que los segmentos recorridos σ_{i0} representados por los vértices con alguna coordenada $x_i = 0$, son segmentos ficticios y por lo tanto tienen un tiempo de recorrido $tr_{i0} = 0$, dichos segmentos se usan para indicar que el robot A_i está listo para recorrer su siguiente segmento, o sea el segmento σ_{i1}. El vecindario de un vértice v se define como $N(v) = \{v' \in CD_{ij} : |x'_i - x_i| \leq 1, i \in \{1, \cdots, n\}, v' \neq v\}$. Los vértices $v' \in N(v)$ vecinos de v denotan los segmentos de ruta que se pueden recorrer a continuación, si $x'_i - x_i \geq 0, i \in \{1, \cdots, n\}$; o denotan los segmentos de ruta que fueron recorridos previo a los segmentos recorridos en v, si $x_i - x'_i \geq 0, i \in \{1, \cdots, n\}$. La Figura 7 repre-

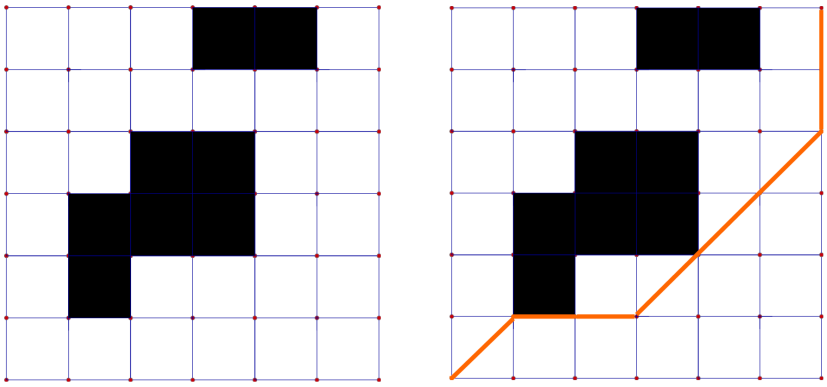


Figura 7: Ejemplo de un diagrama de coordinación y de un calendario de coordinación.

senta un diagrama de coordinación, los cuadros sombreados representan una región de posible colisión entre los robots al recorrer segmentos de ruta que se encuentran en una misma región $\sigma_{ij}, \sigma_{i'j'} \in r_p$, $i \neq i'$, $r_p \in R$. Por simplicidad, en las Figuras 7(a) y (b), la longitud de los lados de los cuadros se muestran iguales, sin embargo, los lados horizontales de un cuadro formado por los vértices diagonales vecinos v y v', representan el tiempo de recorrido $tr_{ix'_i}$ que le toma al robot A_i atravesar $\sigma_{ix'_i}$; del mismo modo, los lados verticales representan el tiempo de recorrido $tr_{jx'_j}$ que le toma al robot A_j atravesar $\sigma_{jx'_j}$. De modo que el tiempo transcurrido al recorrer los segmentos señalados por los vértices en diagonal v y v' se describe como $\max(tr_{ix'_i}, tr_{jx'_j})$.

Diagrama de coordinación *n*-dimensional (CD). Considere ahora la generalización de un diagrama de coordinación, llamado diagrama de coordinación *n*-dimensional. Dado un conjunto de n robots $A = \{A_1, A_2, \cdots, A_n\}$, el diagrama de coordinación *n*-dimensional CD_A se define como el producto cartesiano del conjunto de índices más uno de cada ruta τ_i, es decir, $CD_A = \{(x_1, x_2, \cdots, x_n) \mid x_i \in \{0, 1, \cdots, m_i\}, i \in \{1, 2, \cdots, n\}\}$. La Figura 8 muestra un ejemplo de dos diagramas de coordinación de 2 y 3 dimensiones. El diagrama de coordinación *n*-dimensional está conformado por $\prod_{i=1}^{n}(m_i + 1)$ vértices. Un vértice cualquiera v está representado por las coordenadas $v = (x_1, x_2, \cdots, x_n)$, mientras que el vértice origen por $v_0 = (0, 0, \cdots, 0)$ y el vértice destino por $v_f = (m_1, m_2, \cdots, m_n)$. La definición de vecindario $N(v)$ de un vértice v permanece sin cambios.

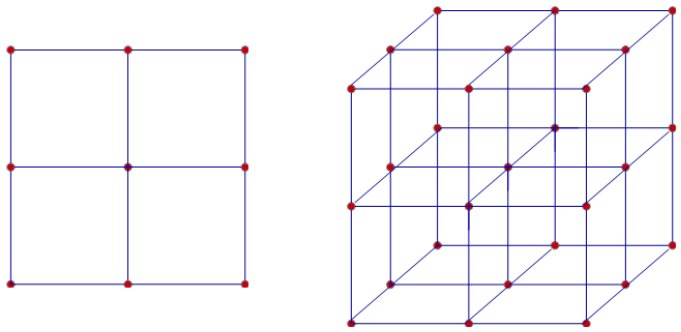


Figura 8: Del lado izquierdo se muestra un diagrama de coordinación bidimensional, mientras que del lado derecho se muestra un diagrama de coordinación tridimensional.

Calendario de coordinación (s). Dado un diagrama de coordinación, una solución al problema es cualquier secuencia no decreciente de vértices que una a v_0 y v_f; a dicha secuencia se la llama calendario de coordinación (Figura 7(b)). El objetivo es encontrar aquel calendario de coordinación que describa una secuencia de recorridos que minimice T_{max}. Para simplificar el problema, se restringe al calendario de coordinación para que pase únicamente por vértices vecinos en un diagrama, a este calendario restringido se le denota por s. De este modo, se puede representar un calendario de coordinación s cualquiera como una secuencia no decreciente de k vértices vecinos entre sí, es decir, $s = \{v_1, v_2, \cdots, v_k \mid v_{(t+1)} \in N(v_t), x_i^{(t+1)} - x_i^{(t)} \geq 0, i \in \{1, 2, \cdots, n\}, t \in \{1, 2, \cdots, k-1\}$, donde $v_1 = v_0\}$. De forma particular, se denota s_v como un calendario de coordinación de k vértices vecinos entre si tal que $v_1 = v_0$ y $v_k = v$. Al conjunto de todos los calendarios de coordinación restringidos que se pueden formar en un diagrama de coordinación se le llama universo de calendarios y se denota por S. Mientras que el universo de calendarios de coordinación tal que $v_k = v$, para un vértice v, se denota por S_v. Observe que, para todo vértice v en un diagrama de coordinación CD, $S_v \subseteq S$. Más aún, para todo vértice v en un diagrama de coordinación, $\bigcup_{v \in CD} S_v = S$.

Calendario de tiempos de inicio (γ). Se define γ_i como el calendario de tiempos de inicio correspondiente a la ruta τ_i. Un calendario γ_i, representa la secuencia de tiempos de inicio para cada segmento de ruta (τ_i) correspondiente, de modo que $\gamma_i = \{\iota_{i1}, \iota_{i2}, \cdots, \iota_{im_i}\}$. Para un robot A_i, ι_{i1} indica el tiempo de inicio del segmento σ_{i1}, ι_{i2} indica el tiempo de inicio del segmento σ_{i2}, y así sucesivamente. Entonces, el calendario γ se puede repre-

sentar como una matriz de tiempos de inicio ι_{ij}, de modo que

$$\gamma = \left\{ \begin{array}{ccc} \iota_{11} & \cdots & \iota_{1m} \\ \vdots & \ddots & \vdots \\ \iota_{n1} & \cdots & \iota_{nm} \end{array} \right\}.$$

Por la restricción de precedencia de segmentos, que impide que un segmento $\sigma_{i(j+1)}$ se recorra antes que el segmento σ_{ij}, se define la siguiente desigualdad

$$\iota_{i(j+1)} \geq \iota_{ij} + tr_{ij}, \quad \text{con } i \in \{1, \cdots, n\} \text{ y } j \in \{1, \cdots, m_i\}. \tag{9}$$

Calendarizador ($f(s)$). Para calcular el tiempo de finalización de cada robot, se debe transformar un calendario de coordinación s en un calendario de tiempos de inicio γ. Para ello, es necesario tomar ordenadamente cada vértice en s y verificar qué segmentos fueron recorridos entre un vértice v_t y un vértice $v_{(t+1)}$. Dado un calendario de coordinación s, se define la función $f : S \to \Gamma$ como

$$f(s) = \gamma = \left\{ \begin{array}{ccc} \iota_{11} & \cdots & \iota_{1m} \\ \vdots & \ddots & \vdots \\ \iota_{n1} & \cdots & \iota_{nm} \end{array} \right\}, \tag{10}$$

donde por cada vértice v_t en s, por cada coordenada $x_i^{(t)}$ en v_t,

$$\iota_{ix_i^{(t+1)}} = \text{máx}(\iota_{ix_i^{(t)}}, \underset{j \in \{1,2,\cdots,n\}}{\text{máx}} (\iota_{jx_j^{(t)}} + tr_{jx_j^{(t)}}) \cdot \mid x_i^{(t)} - x_i^{(t+1)} \mid), \tag{11}$$

donde cada término de la ecuación anterior se describe a continuación

v_t: *t*-ésimo vértice en un calendario s,
$x_i^{(t)}$: *i*-ésima coordenada del *t*-ésimo vértice (v_t), que indica el último segmento recorrido del robot A_i,
$\iota_{ix_i^{(t)}}$: tiempo de inicio del $x_i^{(t)}$-ésimo segmento de ruta τ_i del robot A_i.

Recuerde que, si $x_i^{(t)} = 0$ entonces $\iota_{ix_i^{(t)}} = 0, tr_{ix_i^{(t)}} = 0$.

Tiempo de finalización (T_i). Se define el tiempo de finalización T_i como el tiempo que le toma a un robot A_i completar su ruta τ_i dado un calendario γ_i, esto es,

$$T_i = \iota_{im_i} + tr_{im_i}. \tag{12}$$

Tiempo máximo de finalización (T_{max}). Se define T_{max} como el tiempo de finalización mayor de entre todos los T_is dado un calendario de tiempos de inicio γ, es decir,

$$T_{max}(f(s)) = T_{max}(\gamma) = \max_{i \in \{1,\cdots,n\}} T_i. \tag{13}$$

En la Figura 9 se puede ver un ejemplo de lo que representa la notación.

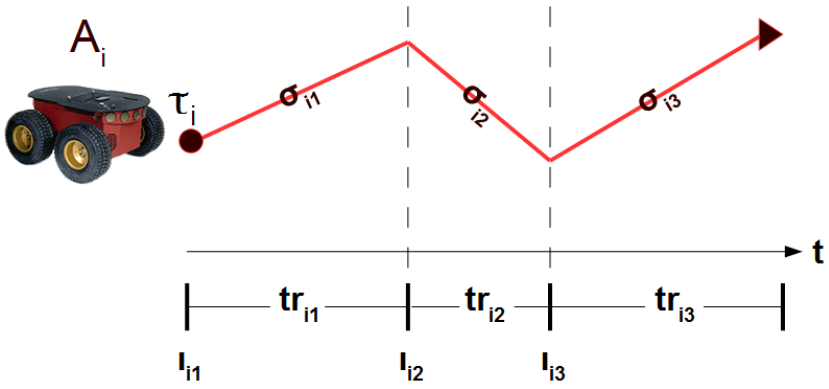


Figura 9: Ejemplificación gráfica de algunos elementos de la notación. A_i es el *i*-ésimo robot, τ_i es la ruta del robot A_i, σ_{ij} es el *j*-ésimo segmento de la ruta τ_i, tr_{ij} es el tiempo de recorrido del segmento σ_{ij}, ι_{ij} es el tiempo de inicio de recorrido del segmento σ_{ij}.

3.2. Definición formal del problema

En esta sección se define el problema general estudiado en este trabajo de tesis.

Sea $A = \{A_1, A_2, \cdots, A_n\}$ un conjunto de n robots con rutas pre-especificadas y segmentadas, donde τ_i denota la ruta correspondiente al robot A_i. Una ruta τ_i se compone

de una secuencia fija ordenado de segmentos $\{\sigma_{i1}, \sigma_{i2}, \cdots, \sigma_{im_i}\}$. Sea σ_{ij} un segmento cualquiera, los subíndices i y j indican la ruta a la que pertenece y su posición en la ruta. Los segmentos en una ruta se encuentran ordenados, dado que existe una restricción de precedencia; formalmente, un robot A_i no puede iniciar el recorrido de su segmento $\sigma_{i(j+1)}$ sin antes haber finalizado el recorrido del segmento σ_{ij}.

Se denomina región de coordinación R, al conjunto de regiones $\{r_1, r_2, \cdots, r_m\}$ por donde los robots en A recorren sus rutas. Una región r_p, es un identificador del área por donde atraviesan uno o más segmentos de ruta. Cuando dos o más segmentos de distintas rutas se intersectan en algún punto de una región cualquiera r_p en R, se dice que éstas pertenecen a la misma región, es decir, $\sigma_{ir}, \sigma_{js} \in r_p$, donde $i \neq j$. Observe que el número de regiones está acotado por el número total de segmentos en cada ruta, $\mid R \mid \leq \sum_{i=1}^{n} m_i$. En la Figura 10 se muestran dos ejemplos de cómo se deviden las regiones por donde pasan dos rutas segmentadas τ_1 y τ_2.

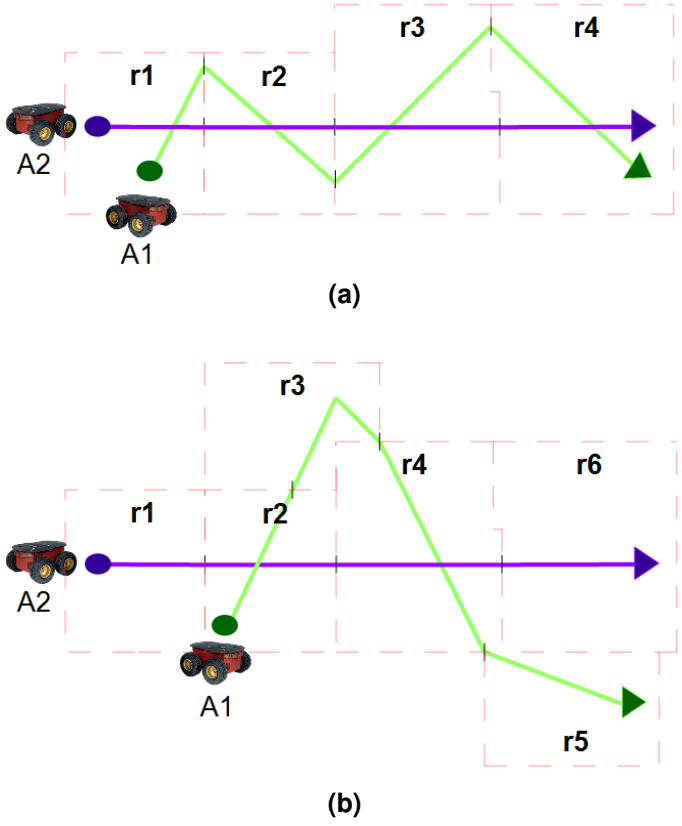


Figura 10: Dos ejemplos gráficos de la división de un espacio en regiones por donde atraviesan segmentos de ruta de dos robots A_1 y A_2.

Por simplicidad, en este trabajo, se supone que sólo existen m regiones y cada ruta pasa a lo sumo una vez por cada región como se observa en la Figura 10(b). Se considera que existe colisión siempre que dos o más robots se encuentren recorriendo, de manera simultánea, segmentos de ruta que se encuentren en una misma región. Además, se supone que, estarán libres de colisión las áreas de inicio y final de cada ruta así como donde cada robot finaliza de recorrer algún segmento. Una vez que un robot inicie el recorrido de algún segmento de su ruta, éste no podra detenerse hasta haber completado ese segmento. Por último, se debe dar por hecho que existe una solución trivial al problema, es decir, en el peor de los casos cada robot debe poder completar su ruta si el resto de los robots permanecen en reposo al inicio de sus respectivas rutas. Dicha solución tendrá un tiempo máximo de finalización $T_{max} = \sum_{i=1}^{n} \sum_{j=1}^{m} (tr_{ij})$.

Sea $\gamma \in \mathbf{R}_{n \times m}$ un calendario que señale los tiempos de inicio de cada segmento de cada ruta. El elemento ι_{ij} en γ indica el tiempo en el que el robot A_i debe iniciar el recorrido de su segmento σ_{ij}. Por la restricción de precedencia, se debe satisfacer la siguiente desigualdad $\forall \iota_{ij} \in \gamma$, $\iota_{i(j+1)} \geq \iota_{ij} + tr_{ij}$.

La definición formal del problema se enuncia a continuación.

Coordinación de rutas de robots (RPC).
Dado un conjunto A de n robots con rutas pre-especificadas τ.

$$\text{Minimizar}_{\gamma \in \Gamma} \;\; T_{max}(\gamma), \tag{14}$$

sujeto a

$$[\iota_{ij}, \iota_{ij} + tr_{ij}) \cap [\iota_{i'j'}, \iota_{i'j'} + tr_{i'j'}) = \emptyset \text{ cuando } \sigma_{ij}, \sigma_{i'j'} \in r_p, \tag{15}$$
$$\forall i, i' \in \{1, 2, \cdots, n\}, i \neq i', \forall j, j' \in \{1, 2, \cdots, m\}, r_p \in R.$$

3.3. Complejidad: equivalencia de Job Shop Scheduling - Coordinación de rutas de robots

En la coordinación de rutas de robots (RPC), las rutas de cada robot se encuentran pre-especificadas, al igual que la velocidad a la que cada robot recorre su ruta (v_{max}). Si se supone que existe aceleración infinita, es decir, que cada robot pueda alcanzar v_{max} o detenerse de forma inmediata, el problema RPC puede verse como un problema de calendarización. Si se considera cada ruta como una tarea y cada región como una máquina, la coordinación de rutas de robots es equivalente al problema conocido como *Job Shop Scheduling* (JSP) (O'Donnell y Lozano-Pérez, 1989; Peng y Akella, 2005).

Teorema 3.1 *Resolver el problema de la Coordinación de Rutas de Robots (RPC) es al menos tan difícil como el problema conocido como el Job Shop Scheduling (JSP).*

Demostración

Primero se define la versión de decisión del RPC. Considere un caso cualquiera de RPC, un conjunto $A = \{A_1, A_2, \cdots, A_n\}$ de n robots. Cada robot A_i en A, con una ruta pre-especificada τ_i dividida en m segmentos, de modo que $\tau_i = \{\sigma_{i1}, \sigma_{i2}, \cdots, \sigma_{im}\}$. A cada segmento σ_{ij}, en cada ruta, le corresponde un tiempo de recorrido tr_{ij} que indica el tiempo que le toma a un robot A_i recorrer dicho segmento. Se desea encontrar un calendario γ que indique el tiempo de inicio de recorrido de cada segmento de ruta para cada robot en A, tal que las restricciones descritas por las Ecuaciones 9 y 15 sean satisfechas. Entonces el problema RPC de decisión (RPC_D) se puede enunciar como, ¿tiene A un calendario de tiempos de inicio γ tal que el tiempo máximo de finalización (T_{max}) es a lo más T?.

1. $RPC_D \in NP$. En dado caso, es necesario verificar si $T_{max}(\gamma) \leq T$. Dado que

$$T_{max}(\gamma) = \max_{i \in \{1,2,\cdots,n\}} T_i, \tag{16}$$

donde

$$T_i = \iota_{im} + tr_{im}, \tag{17}$$

se puede verificar en tiempo polinomial con respecto al número de robots n que dicho certificado es correcto ya que lo único que se requiere hacer es verificar que se cuenta con un calendario válido (restricciones satisfechas; $O(nm)$) y calcular T_i para los n robots $(O(n))$, tomar el máximo de éstos y compararlo con T $(O(n))$. Por lo tanto se puede decir que $RPC_D \in NP$.

2. $JSP_D \in NP$-*completo*. El problema conocido como *Job Shop Scheduling* en su versión de decisión (JSP_D) es *NP-completo* (Garey *et al.*, 1976).

3. $JSP_D \leq_p RPC_D$. Considere un caso cualquiera de JSP_D, un conjunto de n tareas $\{j_1, j_2, \cdots, j_n\}$ y un conjunto $M = \{\mu_1, \mu_2, \cdots, \mu_m\}$ de m máquinas. Cada tarea j_i, $i \in \{1, 2, \cdots, n\}$, se compone por una secuencia fija de m operaciones $j_i = \{o_{i1}, o_{i2}, \cdots, o_{im}\}$. Una operación cualquiera o_{ij} requiere que se procese por una máquina en específico del conjunto $\{\mu_1, \mu_2, \cdots, \mu_m\}$. El tiempo de procesamiento de la operación o_{ij} se denota como p_{ij} y es dependiente de la tarea y de la máquina que lo procesa. Por ejemplo, los tiempos de procesamiento de dos operaciones distintas que se procesan en una misma máquina pueden ser, arbitrariamente, iguales o distintos. Una máquina μ_x cualquiera, de M, sólo puede procesar una operación a la vez, de modo que si μ_x se encuentra procesando alguna operación o_{ij} y se requiere procesar otra operación distinta o_{pq} en μ_x, ésta deberá aguardar hasta que μ_x haya concluído el procesamiento de o_{ij}. Además, para toda tarea j_i, ninguna operación $o_{i(j+1)}$ deberá procesarse antes de la finalización la operación o_{ij}. Finalmente, se debe considerar que una máquina no puede interrumpir el procesamiento una operación hasta haber concluido. Sea C_i el tiempo de finalización de la tarea j_i, el tiempo de finalización máximo se define como $C_{max} = \max_{i \in \{1,2,\cdots,n\}}(C_i)$. El objetivo es minimizar el máximo tiempo de finalización C_{max}, también conocido como *makespan*. Entonces, se puede mostrar que un caso de RPC_D se puede construir en

tiempo polinomial a partir de un caso de JSP_D con

$$\tau_i = j_i,$$
$$\sigma_{ij} = o_{ij},$$
$$tr_{ij} = p_{ij},$$
$$R = M,$$
$$T_{max} = C_{max}.$$

Se requiere verificar entonces que un calendario γ existe para JSP_D tal que $C_{max}(\gamma) \leq C$, si y sólo si $T_{max}(\gamma) \leq T$, donde $T = C$. A continuación se presentan los dos casos:

Caso sí de JSP_D. Suponga un caso cualquiera de JSP_D donde se conoce un calendario γ tal que $C_{max}(\gamma) \leq C$. Asociando cada tarea, cada operación, cada tiempo de procesamiento y cada máquina del caso del JSP_D a cada ruta, cada segmento, cada tiempo de recorrido y cada región, uno a uno respectivamente, como se muestra en el párrafo anterior. Además, al igual que su término asociado en JSP_D, ningún par de segmentos de ruta que pertenecen a una misma región se pueden recorrer de manera simultánea, y ningún $(j+1)$-ésimo segmento de ruta se puede recorrer antes que cualquier j-ésimo segmento en esa misma ruta. Entonces, es posible utilizar este mismo calendario γ para determinar directamente el tiempo de inicio de los recorridos de cada segmento en RPC_D, de modo que $T_{max}(\gamma) \leq C$.

Caso sí de RPC_D. De manera similar, al caso anterior, si se supone un caso cualquiera de RPC_D donde se conoce un calendario γ tal que $T_{max}(\gamma) \leq T$, entonces haciendo dicha misma asociación de términos, $C_{max}(\gamma) \leq T$. En caso contrario, $T_{max}(\gamma) > T$.

Por lo tanto

$$JSP_D \leq_p RPC_D.$$

■

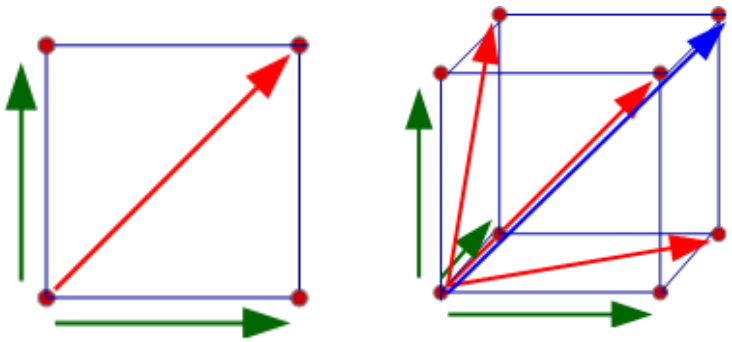
Por lo tanto encontrar aquella coordinación de rutas de robots con T_{max} menor o igual a un dado T, aún cuando las trayectorias se hayan especificado, es *NP-completo* y su versión de optimización es *NP-difícil*.

3.4. Espacio de soluciones

De manera general, sea A un conjunto de n robots $A = \{A_1, A_2, \cdots, A_n\}$, con rutas de m segmentos cada una, y su diagrama de coordinación *n*-dimensional correspondiente.

Proposición 3.1 *El universo de calendarios de coordinación S_{v_f} que se pueden generar está acotado por*

$$| S_{v_f} | \leq 2^{nm^n}.$$



(a) **(b)**

Figura 11: Ejemplo de un diagrama de coordinación. (a) 2-cubo. (b) 3-cubo.

Demostración

Considere un vértice cualquiera v de un diagrama de coordinación *n*-dimensional. Observe que v tiene un número máximo de vértices vecinos: tiene $\binom{n}{1}$ que indican el avance de un robot, $\binom{n}{2}$ que indican el avance de dos robots de manera simultánea, $\binom{n}{i}$ que indican el avance de i robots simultáneamente, y así hasta $\binom{n}{n}$ vértices que indican el avance de los n robots en paralelo (ver Figuras 11(a) y (b)). De modo que el número

de vecinos de v en un diagrama de coordinación *n*-dimensional se encuentra acotado por $|\overleftarrow{N}(v)| \leq \sum_{i=1}^{n} \binom{n}{i}$. Dicha expresión es parecida a la identidad binomial, explicitamente,

$$(1+x)^n = \sum_{i=0}^{n} \binom{n}{i} x^i.$$

De aquí que, con $x = 1$,

$$\sum_{i=0}^{n} \binom{n}{i} x^i = \left(\sum_{i=1}^{n} \binom{n}{i} \right) + \binom{n}{0} = 2^n,$$

entonces

$$\sum_{i=1}^{n} \binom{n}{i} = 2^n - 1.$$

Dado que en un diagrama de coordinación *n*-dimensional existen a lo más m^n segmentos de ruta, entonces,

$$\left(\sum_{i=1}^{n} \binom{n}{i} \right)^{m^n} = (2^n - 1)^{m^n}.$$

Finalmente, observe que

$$\left(\sum_{i=1}^{n} \binom{n}{i} \right)^{m^n} = (2^n - 1)^{m^n} \leq 2^{nm^n},$$

por lo tanto, el universo de calendarios de coordinación está acotado por,

$$\mid S_{v_f} \mid \leq 2^{nm^n}.$$

■

3.5. Cotas para el óptimo

En esta sección se describen dos cotas triviales de T_{opt}, una superior y otra inferior. La cota inferior de T_{opt} en RPC se define a continuación.

$$\max_{i\in\{1,\cdots,n\}}\left(\sum_{j=1}^{m_i} tr_{ij}\right) \leq T_{opt}. \tag{18}$$

Demostración

Por definición $T_i = \iota_{im_i} + tr_{im_i}$, y dado que existe una restricción de precedencia

$$\iota_{i(j+1)} \geq \iota_{ij} + tr_{ij}, \quad \text{con } i \in \{1,\cdots,n\} \text{ y } j \in \{1,\cdots,m_i\}.$$

Entonces,

$$\iota_{im_i} \geq \left(\sum_{j=1}^{m_i-1} tr_{ij}\right).$$

Por lo tanto, dada una calendarización óptima, el tiempo de finalización óptimo debe cumplir con la siguiente desigualdad,

$$T_{opt} = \max_{i\in\{1,\cdots,n\}}(T_i) = \max_{i\in\{1,\cdots,n\}}(\iota_{im_i} + tr_{im_i}) \geq \max_{i\in\{1,\cdots,n\}}\left(\sum_{j=1}^{m_i-1} tr_{ij} + tr_{im_i}\right).$$

Por lo tanto,

$$\max_{i\in\{1,\cdots,n\}}\left(\sum_{j=1}^{m_i} tr_{ij}\right) \leq T_{opt}.$$

■

La cota superior de T_{opt} en RPC se define a continuación

$$T_{opt} \leq \sum_{i=1}^{n}\sum_{j=1}^{m_i} tr_{ij} \leq n \cdot \max_{i\in\{1,\cdots,n\}}\left(\sum_{j=1}^{m_i} tr_{ij}\right). \tag{19}$$

Demostración

Dado que existe una solución trivial, es decir, cada ruta se puede recorrer de manera secuencial y sin riesgo de colisión. Entonces, en el peor de los casos, cuando ninguna

ruta, ni segmento de rutas, se pueda recorrer en paralelo con otra, entonces,

$$T_{opt} = \sum_{i=1}^{n} \sum_{j=1}^{m_i} tr_{ij}.$$

Obseve que T_{opt} no debe ser mayor, en caso contrario éste no sería el óptimo. Por lo tanto, en todo caso,

$$T_{opt} \leq \sum_{i=1}^{n} \sum_{j=1}^{m_i} tr_{ij}.$$

Por otro lado, teniendo n rutas, una cota más holgada es la siguiente,

$$\sum_{i=1}^{n} \sum_{j=1}^{m_i} tr_{ij} \leq n \cdot \max_{\forall i \in \{1,\cdots,n\}} \left(\sum_{j=1}^{m_i} tr_{ij} \right).$$

Por lo tanto,

$$T_{opt} \leq \sum_{i=1}^{n} \sum_{j=1}^{m_i} tr_{ij} \leq n \cdot \max_{\forall i \in \{1,\cdots,n\}} \left(\sum_{j=1}^{m_i} tr_{ij} \right).$$

■

En este capítulo se hizo una descripción de la notación y las definiciones básicas de la coordinación de rutas y una definición formal del problema RPC. También se hizo un análisis de complejidad del RPC y se definieron dos cotas triviales para el óptimo del mismo. En el siguiente capítulo se describe una técnica propuesta para hacer frente al problema de la coordinación de rutas con trayectorias pre-especificadas.

Capítulo 4. Técnica de agrupamiento de robots

En el capítulo anterior, en el contexto de la coordinación de rutas de robots, se enuncian algunas definiciones básicas y se define formalmente el problema. En este capítulo se propone una técnica de agrupamiento de robots para afrontar el problema.

En las secciones subsecuentes de este capítulo se propone una técnica de agrupamiento de robots, se decriben tres algoritmos usando esta técnica de agrupamiento y se analiza su aplicación para el RPC. El análisis preliminar de los algoritmos incluye tanto el tiempo de ejecución como las limitaciones de los mismos.

4.1. Propuesta

Considerando el espacio y el tiempo que implica construir y encontrar una solución en un diagrama de coordinación *n*-dimensional, en este documento se propone una técnica de agrupamiento de robots. Esta técnica, iterativamente, divide un conjunto de robots en subconjuntos de tamaño constante y coordina los subconjuntos de robots de manera independiente unos de otros. A continuación se describe la técnica de agrupamiento a grandes rasgos.

Técnica de agrupamiento de robots. Sea A un conjunto de n robots y k un número entero positivo. Se divide el conjunto A en n/k subconjuntos disjuntos de k robots, donde cada uno de los robots en A está contenido en alguno de los n/k subconjuntos. Note que la unión de todos los subconjuntos debe ser igual al conjunto original de robots A. Por cada subconjunto de robots, se construye su diagrama de coordinación k-dimensional y se calcula el calendario de coordinación s que minimiza el tiempo máximo de finalización entre los k robots del subconjunto. Posteriormente, se considera cada subconjunto de robots junto con su respectivo calendario de coordinación como un robot compuesto coordinado. De este modo se cuenta con un conjunto A' de n/k robots compuestos coordinados. Finalmente, se reemplaza A por A' y se repite el procedimiento a menos que $|A'| = 1$.

4.1.1. Robot compuesto

Un robot compuesto C se contruye a partir del par (A, s), donde A es un conjunto de n robots y s el calendario de coordinación de los segmentos que componen las rutas de cada robot en A. Por definición, un calendario de coordinación s describe una secuencia de vértices en un diagrama de coordinación, los cuales a su vez describen el desplazamiento de los robots sobre sus segmentos de ruta correspondientes, sin que ocurra colisión entre ellos. De este modo, un robot compuesto C se considera coordinado, y por lo tanto, éste no colisiona consigo mismo. Dado que un solo robot simple A_i no puede colisionar consigo mismo, éste se puede considerar como un robot compuesto si $s = \{v_0, v_2, \cdots, v_m\}$, con $C = (\{A_i\}, s)$, donde cada vértice v_j en s representa el recorrido secuencial de cada segmento de ruta τ_i. Observe además que el calendario de coordinación s de un diagrama de coordinación unidimensional CD_A, con $|A| = 1$, implica forzosamente que $\{v_j \mid v_j \in s\} = CD_A$.

La ruta compuesta $C\tau$ de un robot compuesto se compone de una secuencia de tuplas de longitud variable. Cada *n*-tupla recibe el nombre de segmento compuesto $C\sigma$. Cada elemento en $C\sigma$ regresa el segmento de ruta σ_{ix} que un robot simple A_i en A debe recorrer simultáneamente con cada segmento indicado por los demás elementos en $C\sigma$. Además, cada segmento en $C\sigma$ corresponde a un robot distinto, es decir, un $C\sigma$ no contiene más de un segmento a recorrer del mismo robot. Se puede representar gráficamente una ruta compuesta $C\tau$ como un árbol de segmentos ($C\sigma$ y σ) enraizados en $C\tau$ (ver Figura 12).

El tiempo de recorrido compuesto Ctr de un segmento compuesto $C\sigma$ (*n*-tupla) es igual al máximo tiempo de recorrido entre los segmentos en $C\sigma$, es decir, $Ctr = \max\{tr_{ij} \mid \sigma_{ij} \in C\sigma\}$.

Ejemplo 1. Sea C un robot compuesto asociado a un conjunto A de dos robots A_1 y A_2 con un calendario de coordinación s. Las rutas de cada robot en A son de longitud $m = 4$ y son: $\tau_1 = \{\sigma_{11} \in r_1, \sigma_{12} \in r_3, \sigma_{13} \in r_2, \sigma_{14} \in r_4\}$ y $\tau_2 = \{\sigma_{21} \in r_2, \sigma_{22} \in r_3, \sigma_{23} \in r_4, \sigma_{24} \in r_1\}$. Dado un calendario de coordinación $s = \{(0,0), (1,1), (2,1), (3,2), (3,3), (4,4)\}$. En-

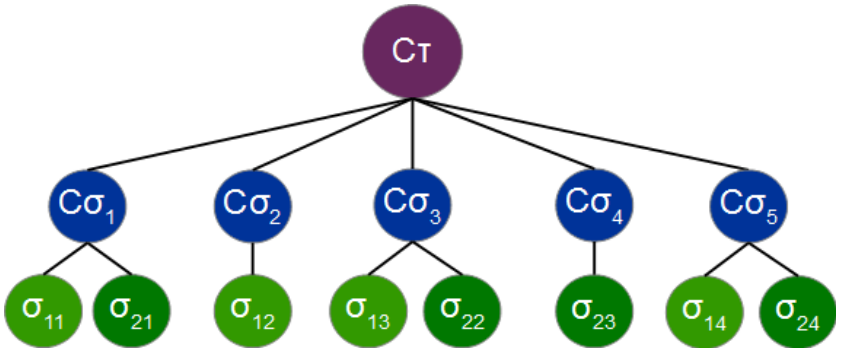


Figura 12: Representación gráfica de una ruta compuesta $C\tau$ que se forma a partir de dos rutas $\tau_1 = \{\sigma_{11}, \sigma_{12}, \sigma_{13}, \sigma_{14}\}$ y $\tau_2 = \{\sigma_{21}, \sigma_{22}, \sigma_{23}, \sigma_{24}\}$ y un calendario de coordinación $s = \{(0,0),(1,1),(2,1),(3,2),(3,3),(4,4)\}$ (Ejemplo 1).

tonces, una ruta compuesta de C sería $C\tau = \{(\sigma_{11}, \sigma_{21}), (\sigma_{12}), (\sigma_{13}, \sigma_{22}), (\sigma_{23}), (\sigma_{14}, \sigma_{24})\}$, como se muestra en la Figura 12, donde el segmento compuesto $C\sigma_1 = (\sigma_{11}, \sigma_{21})$, $C\sigma_2 = (\sigma_{12})$ y así sucesivamente, un segmento compuesto por cada dupla en s.

En las secciones subsecuentes de este capítulo se describen tres algoritmos que implementan la técnica de agrupamiento.

4.2. Algoritmo GANTT (*Grouping Algorithm for Numerous Traversal Tasks*)

En esta sección el autor de este documento describe un algoritmo para el problema del RPC basado en la técnica de agrupamiento de robots. El algoritmo recibe como entrada un conjunto A de n robots y un entero k. Cada robot A_i en A tiene asignada una ruta τ_i, y ésta a su vez está previamente segmentada en m segmentos de ruta. Cada segmento σ_{ij} de cada ruta τ_i tiene asignado un tiempo de recorrido tr_{ij}, el cual denota el tiempo que le toma al robot A_i recorrer la región especificada por el segmento de ruta σ_{ij}. El algoritmo regresa como salida un calendario de coordinación s. Éste especifica el orden (secuencial o paralelo) en el que los segmentos de cada ruta se deben recorrer de modo que minimicen el tiempo máximo de finalización T_{max}, sin que exista colisión entre cualesquier par de robots (Ecuación 9) y respetando la restricción de precedencia de cada ruta (Ecuación 15).

El Algoritmo 1 muestra el pseudocódigo de la propuesta. A continuación se describen los detalles de los procedimientos principales del mismo.

Algoritmo 1 GANTT(A,k)

si $|A| = 1$ **entonces**
  regresar s asociado al único robot (compuesto) en A
si no
  $A' \leftarrow \emptyset$
  $G \leftarrow$ AGRUPAR(A,k)
  para $g_i \in G$ **hacer**
    $CD_{g_i} \leftarrow$ CONSTRUIR-DIAGRAMA-COORDINACION(g_i,k)
    $s(g_i) \leftarrow$ CALCULAR-CALENDARIO-COORDINACION(CD_{g_i})
    $A'_i \leftarrow$ COMPONER(g_i,$s(g_i)$)
    $A' \leftarrow A' \cup \{A'_i\}$
  fin para
  regresar $GANTT(A', k)$
fin si

4.2.1. Procedimiento AGRUPAR()

El procedimiento AGRUPAR() recibe como entrada el conjunto A de robots (compuestos) y el entero k. La salida es un conjunto G de $|A|/k$ grupos de k robots (compuestos) cada uno. La agrupación se hace de la siguiente manera: se agrupan los primeros k robots en A, posteriormente los siguientes k, y así sucesivamente hasta agrupar los últimos r robots, donde r es el residuo de la división $|A|/k$.

Ejemplo 2. Si se tiene un conjunto A de 10 robots y se requiere agruparlos en grupos de tamaño 2. El procedimiento genera un conjunto G de $\lceil 10/2 \rceil = 5$ grupos de 2 robots cada uno. De modo que el conjunto de agrupaciones G en la primer iteración del algoritmo es $G = \{(A_1, A_2), (A_3, A_4), (A_5, A_6), (A_7, A_8), (A_9, A_{10})\}$. En la Figura 13 se puede ver un ejemplo de las agrupaciones en cada iteración del algoritmo con $n = 10$ y $k = 2$. En esta figura, en la primer iteración (columna), $g_1 = (A_1, A_2), g_2 = (A_3, A_4), \cdots, g_5 = (A_9, A_{10})$.

Ejemplo 3. Si se tiene un conjunto A de 10 robots y se requiere agruparlos en grupos de tamaño 3. El procedimiento genera un conjunto G de $\lceil 10/3 \rceil = 4$ grupos de robots, tres grupos de 3 robots y uno de 1. De modo que el conjunto de agrupaciones G en la primer iteración del algoritmo es $G = \{(A_1, A_2, A_3), (A_4, A_5, A_6), (A_7, A_8, A_9), (A_{10})\}$. En la Figura 14 se puede ver un ejemplo de las agrupaciones en cada iteración del algoritmo con $n = 10$ y $k = 3$.

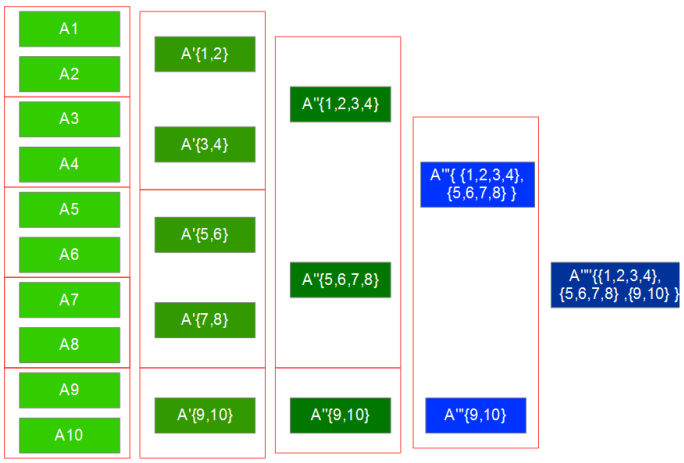


Figura 13: Ejemplo de las agrupaciones en cada iteración del algoritmo con $n = 10$ y $k = 2$ (Ejemplo 2).

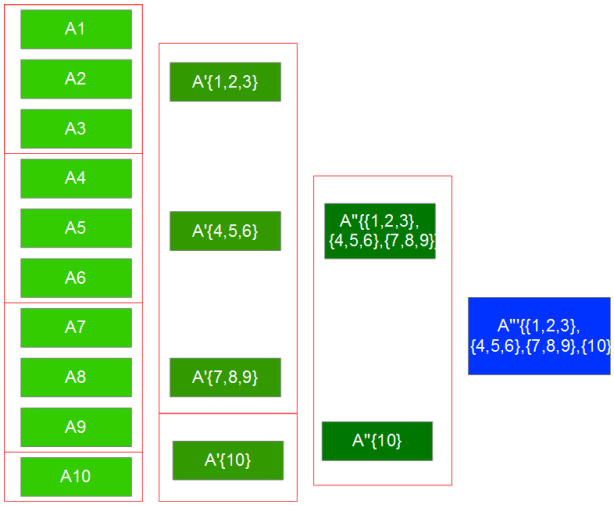


Figura 14: Ejemplo de las agrupaciones en cada iteración del algoritmo con $n = 10$ y $k = 3$ (Ejemplo 3).

4.2.2. Procedimiento CONSTRUIR-DIAGRAMA-COORDINACION()

El procedimiento CONSTRUIR-DIAGRAMA-COORDINACION() recibe como entrada un grupo de robots g y un entero k. La salida es un diagrama de coordinación *k*-dimensional de los k robots en g. Para construir un diagrama de coordinación existen dos casos posibles:

- **Caso 1.** El grupo g de G está conformado por robots simples. En este caso, el diagrama de coordinación se construye como se muestra en la Sección 3.1 (Diagrama

de coordinación y Diagrama de coordinación *n*-dimensional).

Ejemplo 4, Caso 1. Sea g un agrupamiento de dos robots A_1 y A_2, con rutas $\tau_1 = \{\sigma_{11} \in r_1, \sigma_{12} \in r_3, \sigma_{13} \in r_2, \sigma_{14} \in r_4\}$ y $\tau_2 = \{\sigma_{21} \in r_2, \sigma_{22} \in r_3, \sigma_{23} \in r_4, \sigma_{24} \in r_1\}$, respectivamente. El diagrama de coordinación correspondiente se muestra en la Figura 15.

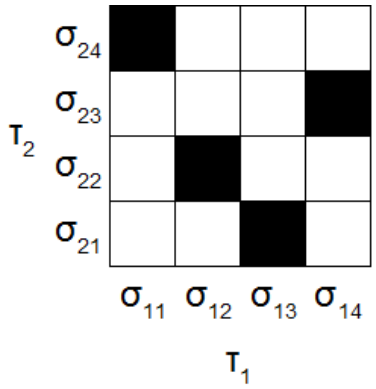


Figura 15: Diagrama de coordinación de dos robots A_1 y A_2 (Ejemplo 4, Caso 1).

- **Caso 2.** El grupo g de G está conformado por robots compuestos. En este caso, el diagrama de coordinación se construye de manera similar al Caso 1. La diferencia consiste en que cada coordenada x_i de cada vértice v en CD, denota que el robot compuesto C_i ha terminado de recorrer su segmento compuesto $C\sigma_{ix_i}$. Observe que $C\sigma_{ix_i}$ a su vez está compuesto de uno o más segmentos de una composición previa, como se observa en la Figura 12, con la diferencia de que los segmentos compuestos pueden indicar el recorrido de más de un segmento de distintos robots simples. Potencialmente más de una región se encuentra ocupada por un robot compuesto durante el recorrido de sus segmentos compuestos. Dicho de otro modo, un segmento compuesto puede pertenecer a múltiples regiones. Esto supone que para un par de robots compuestos puede existir riesgo de colisión en más de una región a la vez durante el recorrido de sus segmentos correspondientes.

Ejemplo 5, Caso 2. Sea g un agrupamiento de dos robots compuestos C_1 construido a partir de $(\{A_1, A_2\}, s_1)$ y C_2 construido a partir de $(\{A_3, A_4\}, s_2)$, con rutas $C\tau_1 =$

$\{(\sigma_{11}, \sigma_{21}), (\sigma_{12}), (\sigma_{13}, \sigma_{22}), (\sigma_{23}), (\sigma_{14}, \sigma_{24})\}$ y $C\tau_2 = \{(\sigma_{31}, \sigma_{41}), (\sigma_{32}, \sigma_{42}), (\sigma_{33}, \sigma_{43}), (\sigma_{34}, \sigma_{44})\}$, respectivamente, donde

$$\sigma_{11}, \sigma_{24}, \sigma_{34}, \sigma_{41} \in r_1,$$
$$\sigma_{13}, \sigma_{21}, \sigma_{32}, \sigma_{43} \in r_2,$$
$$\sigma_{12}, \sigma_{22}, \sigma_{31}, \sigma_{42} \in r_3,$$
$$\sigma_{14}, \sigma_{23}, \sigma_{33}, \sigma_{44} \in r_4.$$

El diagrama de coordinación correspondiente se muestra en la Figura 16.

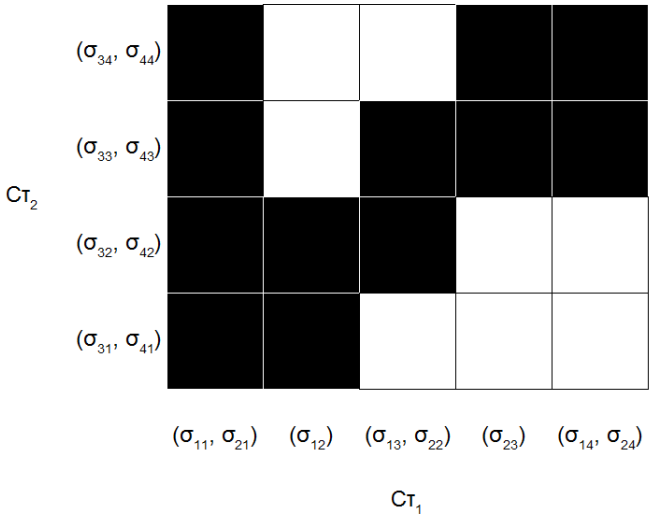


Figura 16: Diagrama de coordinación de dos robots compuestos C_1 y C_2 (Ejemplo 5, Caso 2).

4.2.3. Procedimiento CALCULAR-CALENDARIO-COORDINACION()

El procedimiento CALCULAR-CALENDARIO-COORDINACION() recibe como entrada un conjunto de calendarios de coordinación CD_g correspondiente a un grupo g de robots (compuestos). La salida es un calendario de coordinación s que describe una secuencia de vértices en CD_g. El calendario de coordinación s se calcula de manera recursiva bajo un enfoque conocido como *top-down*. Sea s_v un calendario de coordinación parcial cualquiera, la función de recurrencia se describe de la siguiente manera:

$$s_v = \begin{cases} \arg\min_{s_u \cup \{v\} | u \in \overleftarrow{N}(v)} T_{max}(f(s_u \cup \{v\})), \\ \{v_0\}, \text{ if } v = v_0. \end{cases} \quad (20)$$

Donde cada componente de la Ecuación 20 se describe a continuación:

s_v: calendario de coordinación parcial cuyo último elemento es el vértice v,
$\overleftarrow{N}(v)$: conjunto de vértices vecinos previos del vértice v, donde toda coordenada x_i de $u \in \overleftarrow{N}(v)$ es menor o igual que la coordenada x'_i de v,
T_{max}: tiempo máximo de finalización de recorrido (total o parcial) entre todos los robots,
$f(s)$: función (calendarizador) que transforma un calendario de coordinación s en un calendario de tiempos de inicio γ,
v_0: vértice origen del calendario de coordinación, cada una de las componentes se describe a detalle en el Capítulo 3.

El enfoque *top-down* se caracteriza por el uso de una técnica conocida como *memoization* (Cormen *et al.*, 2001), la cual guarda en un memo una copia de una solución resultante de una recursión previamente calculada.

De modo que, dado un diagrama de coordinación cualquiera CD_A, el calendario que describe el recorrido de cada robot en A hasta concluir su ruta s_{v_f} se calcula mediante la Ecuación 20.

Un calendario de coordinación s_v cualquiera se calcula a lo más una vez, realizando $O(2^k)$ comparaciones, y se deben calcular $O(m^k)$ calendarios. Por lo tanto, se requiere de $O((2m)^k)$ pasos computacionales para calcular s_{v_f}.

4.2.4. Procedimiento COMPONER()

El procedimiento COMPONER() recibe como entrada un grupo de robots g y un calendario de coordinación $s(g)$ correspondiente que describe los movimientos de cada robot en g sin que ocurra colisión entre ellos. La salida es un robot compuesto C con su ruta C_τ y su tiempo de recorrido compuesto C_{tr}, respectivos (ver Sección 4.1.1). La ruta compuesta se construye de la siguiente manera:

1: $C_\tau \leftarrow \emptyset$ *Inicializa la ruta compuesta*
2: **para** $t \in \{1, 2, \cdots, |s| - 1\}$ **hacer** *Por cada elemento en el calendario de coordinación (total o parcial)* s

3: $C_\sigma \leftarrow \{\sigma_{ix_i^{(t+1)}} \in \tau_i \mid x_i^{(t+1)} - x_i^{(t)} = 1, \forall i \in \{1, 2, \cdots, |v_t|\}\}$ *Crea un nuevo segmento compuesto que contenga cada segmento simple que haya sido recorrido entre los vértices v_t y $v_{(t+1)}$ de s, indicado por la diferencia entre sus coordenadas*

4: $C_\tau \leftarrow C_\tau \cup C_\sigma$ *Concatena el segmento compuesto a la ruta compuesta*

5: **fin para**

El tiempo de recorrido compuesto se construye de la siguiente manera:

1: $C_{tr} \leftarrow \emptyset$ *Inicializa la lista de tiempo compuesto*

2: **para** $t \in \{1, 2, \cdots, |C_\tau|\}$ **hacer** *Por cada segmento compuesto de la ruta compuesta generada*

3: $C_{tr_t} \leftarrow \max(tr_{ij} : \sigma_{ij} \in C_{\sigma_t})$ *Guarda el máximo tiempo de recorrido de los segmentos simples que conforman el t-ésimo segmento compuesto de la ruta compuesta generada*

4: $C_{tr} \leftarrow C_{tr} \cup C_{tr_t}$ *Añade el valor de tiempo compuesto a la lista de tiempos compuestos*

5: **fin para**

4.2.5. Análisis del tiempo de ejecución del Algoritmo 1

El costo computacional por línea del algoritmo propuesto se muestra en la Tabla 1. Dado que el algoritmo se manda a llamar a si mismo, se obtiene la recurrencia:

$$T(n) = \begin{cases} \theta(1), \text{ si } n = 1; \\ T(n/k) + O(\frac{n}{k}(2m')^k). \text{ en otro caso} \end{cases} \tag{21}$$

Donde

$T(n)$: tiempo de ejecución para una entrada de tamaño n,
n: número de robots,
m': número máximo de segmentos,
k: constante de agrupamiento ($k = O(1)$).

Observe que en el costo computacional de la Ecuación 21 está incluido un término m', este término denota el número máximo de segmentos de ruta con los que el algoritmo

Tabla 1: Costo computacional por línea del algoritmo GANTT.

Línea	Costo
1 - 4	$\theta(1)$
5	$O(\frac{n}{k})$
6	$\theta(1)$ $\frac{n}{k}$ veces
7	$O((m')^k)$ $\frac{n}{k}$ veces
8	$O((2m')^k)$ $\frac{n}{k}$ veces
9	$O(m')$ $\frac{n}{k}$ veces
10 - 11	$\theta(1)$
12	$T(\frac{n}{k})$

propuesto trabaja. Si bien, se dijo que m es el número de segmentos en los que se divide una ruta y que se supone que cada ruta tiene el mismo número de segmentos. Al hacer un robot compuesto C de k robots, la longitud de su ruta compuesta $C\tau$ será mayor o igual a m y menor o igual a km. En la Figura 16 se muestra como la ruta compuesta $C\tau_1$ tiene una longitud mayor a $m = 4$ segmentos de las rutas originales. Observe entonces que en cada etapa recursiva $i \in \{1, 2, \cdots, \log_k(n)\}$ del algoritmo, el número de segmentos está acotado superiormente por $m'_i = km'_{i-1}$, $m'_0 = m$. Por lo tanto

$$m' = O(k^{(\log_k(n))}m) = O(nm). \tag{22}$$

Proposición 4.1 *El algoritmo GANTT tiene un costo computacional de* $O(\frac{1}{km}(2nm)^{(k+1)}\log_k(n))$ *pasos.*

Demostración

Sea k la constante de agrupamiento, suponga la siguiente cota para la Ecuación 21,

$$T(n) = O(\frac{1}{km}(2nm)^{(k+1)}\log_k(n)).$$

Por lo tanto, para alguna constante $c > 0$,

$$T(n) < \frac{c}{km}(2nm)^{(k+1)}\log_k(n).$$

Probando la cota propuesta para un valor de $k = n$, $m' = nm$ (Ecuación 22) y dos constantes c y d, donde $c \geq d \geq 1$. Sustituyendo esto en la Ecuación 21, se tiene que

$$\begin{aligned} T(k) &= T(k/k) + O(\frac{k}{k}(2km)^k) \\ &= \theta(1) + d\frac{k}{k}(2km)^k \\ &= \theta(1) + d(2km)^k \\ &= \theta(1) + d\frac{1}{2km}(2km)^{(k+1)} \end{aligned}$$

$$\Rightarrow \frac{d}{2}\frac{1}{km}(2km)^{(k+1)}\log_k(k) \leq c\frac{1}{km}(2km)^{(k+1)}\log_k(k).$$

Suponiendo que $T(n/k) \leq \frac{c}{km}(2\frac{n}{k}m)^{(k+1)}\log_k(n/k)$ es correcto. Sustituyendo esta última desigualdad y m' por nm (Ecuación 22) en la recurrencia (Ecuación 21) se tiene que

$$\begin{aligned} T(n) &\leq \frac{c}{km}(2\frac{n}{k}m)^{(k+1)}\log_k(n/k) + d\frac{n}{k}(2nm)^k \\ &= \frac{c}{km}(2\frac{n}{k}m)^{(k+1)}\log_k(n) - \frac{c}{km}(2\frac{n}{k}m)^{(k+1)}\log_k(k) + d\frac{n}{k}(2nm)^k \\ &= \frac{c}{km}(2\frac{n}{k}m)^{(k+1)}\log_k(n) - \frac{c}{km}(2\frac{n}{k}m)^{(k+1)} + d\frac{n}{k}(2nm)^k \\ &= \frac{1}{k^{(k+1)}}\frac{c}{km}(2nm)^{(k+1)}\log_k(n) + \frac{d}{2nm}\frac{n}{k}(2nm)^{(k+1)} - \frac{c}{km}(2\frac{n}{k}m)^{(k+1)} \\ &= \frac{1}{k^{(k+1)}}\frac{c}{km}(2nm)^{(k+1)}\log_k(n) + \frac{d}{2c}\frac{c}{km}(2nm)^{(k+1)} - \frac{1}{k^{(k+1)}}\frac{c}{km}(2nm)^{(k+1)} \\ &= \frac{c}{km}(2nm)^{(k+1)}\log_k(n)\left(\frac{1}{k^{(k+1)}} + \frac{d}{2c\log_k(n)} - \frac{1}{k^{(k+1)}\log_k(n)}\right). \end{aligned}$$

Observe que, en la última igualdad de la ecuación anterior, los términos del polinomio $\left(\frac{1}{k^{(k+1)}} + \frac{d}{2c\log_k(n)} - \frac{1}{k^{(k+1)}\log_k(n)}\right)$ toman los valores $\frac{1}{k^{(k+1)}} = 1$, $\frac{d}{2c\log_k(n)} = 0$ y $\frac{1}{k^{(k+1)}\log_k(n)} = 0$, cuando $k = 1$, y están acotados por $\frac{1}{k^{(k+1)}} < 0.5$, $\frac{d}{2c\log_k(n)} < 0.5$ y $\frac{1}{k^{(k+1)}\log_k(n)} < 0.5$, cuando $k > 1$ y $c \geq d$. Dado que el tercer término $\left(\frac{1}{k^{(k+1)}\log_k(n)}\right)$ de dicho polinomio no toma valores negativos, el polinomio queda acotado por $\left(\frac{1}{k^{(k+1)}} + \frac{d}{2c\log_k(n)} - \frac{1}{k^{(k+1)}\log_k(n)}\right) \leq 1$. Entonces, se tiene que:

$$\begin{aligned} T(n) &\leq \frac{c}{km}(2nm)^{(k+1)}\log_k(n)\left(\frac{1}{k^{(k+1)}} + \frac{d}{2c\log_k(n)} - \frac{1}{k^{(k+1)}\log_k(n)}\right) \\ &\leq \frac{c}{km}(2nm)^{(k+1)}\log_k(n). \end{aligned}$$

Por lo tanto, se tiene que $T(n) = O(\frac{1}{km}(2nm)^{(k+1)} \log_k(n))$.

■

Considerando que $k = O(1)$, entonces el costo computacional del algoritmo GANTT es polinomial en el número de robots (n). Por ejemplo, para $k = 2$, $T(n) = O(\frac{1}{2m}(2nm)^3 \log_2(n))$ $= O(n(2nm)^2 \log_2(n))$.

4.2.6. Discusión

El algoritmo tiene dos limitantes principales, la primera tiene que ver con la holgura de las soluciones generadas; la segunda con el orden de una agrupación y el calendario que genera.

Debido a que se restringe al calendario de coordinación a generar soluciones que conectan únicamente vértices vecinos del diagrama de coordinación, se están perdiendo soluciones válidas que no cumplen con dicha restricción impuesta y entre ellas se pudiera encontrar T_{opt}. En la Figura 17 se muestra un ejemplo de un diagrama de coordinación y dos calendarios de coordinación con tiempo de finalización distintos. El diagrama corresponde a dos robots A_1 y A_2 donde $\tau_1 = \{\sigma_{11}, \sigma_{12}, \sigma_{13}\}$, $\tau_2 = \{\sigma_{21}, \sigma_{22}, \sigma_{23}\}$ y $tr_1 = \{1, 3, 2\}$, $tr_2 = \{3, 2, 2\}$. El calendario de coordinación que es obligado a pasar por vértices vecinos tiene un tiempo de finalización $T_{max} = 8$ (línea continua azul), mientras que el otro calendario que uniera de manera directa v_0 con v_f, tiene un tiempo de terminación óptimo $T_{max} = 6$ (línea discontinua verde). En la Figura 18 se muestra el diagrama de Gantt correspondiente a ambos calendarios mostrados en la Figura 17.

Por otro lado, el calendario de coordinación resultante es dependiente del orden de los robots en la agrupación. Dicho de otro modo, sea $A = \{A_1, A_2, A_3\}$ y $A' = \{A_2, A_1, A_3\}$ un par de conjuntos ordenados de 3 robots, los calendarios de coordinación $s_{v_f \in CD_A}$ y $s_{v'_f \in CD_{A'}}$, pudieran ser distintos incluso aunque $T_{max}(f(s_{v_f})) = T_{max}(f(s_{v'_f}))$. Esto se debe a que en cada iteración de la Ecuación 20 puede existir más de un vértice vecino previo $u \in \overleftarrow{N}(v)$ tal que $T_{max}(f(s_u \cup \{v\})) = T_{max}(f(s_v))$, para cualquier calendario s_v. Suponiendo que se elige el primer vértice u que satisfaga la igualdad bajo algún criterio arbitrario, entonces el calendario final s_{v_f} estará determinado por el orden de A.

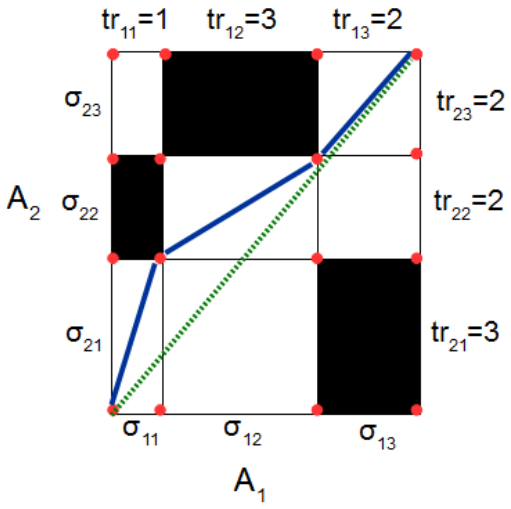


Figura 17: Ejemplo de un diagrama de coordinación de A_1 y A_2 con $\tau_1 = \{\sigma_{11}, \sigma_{12}, \sigma_{13}\}$, $\tau_2 = \{\sigma_{21}, \sigma_{22}, \sigma_{23}\}$ y $tr_1 = \{1, 3, 2\}$, $tr_2 = \{3, 2, 2\}$; y dos calendarios de coordinación distintos.

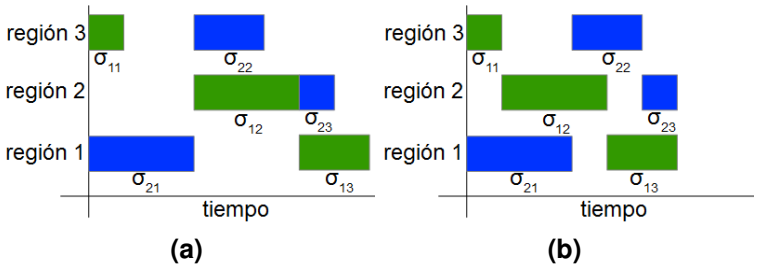


Figura 18: Diagramas de Gantt de los calendario de coordinación de la Figura 17.

4.3. Algoritmo GANTT-T

Para lidiar con la holgura de las soluciones generadas por el algoritmo GANTT, como se menciona en la subsección anterior, se implementa un *adelgazador*. La idea del algoritmo GANTT-T sigue siendo la misma que en el algoritmo GANTT, iterativamente dividir la coordinación de los *n* robots en grupos de tamaño *k*.

La modificación propuesta se describe en el pseudocódigo del Algoritmo 2

4.3.1. Adelgazador

Como se menciona en la Sección 4.2.6, uno de los inconvenientes del algoritmo GANTT, a modo de mantener la restricción de precedencia, los robots no deberán iniciar el recorrido de los segmentos indicados en el vértice $v_{(t+1)}$ hasta que haya finalizado

Algoritmo 2 GANTT-T(A,*k*)

si $|A| = 1$ **entonces**
  regresar s asociado al único robot (compuesto) en A
si no
  $A' \leftarrow \emptyset$
  $G \leftarrow$ AGRUPAR(A,k)
  para $g_i \in G$ **hacer**
    $CD_{g_i} \leftarrow$ CONSTRUIR-DIAGRAMA-COORDINACION(g_i,k)
    $s(g_i) \leftarrow$ CALCULAR-CALENDARIO-COORDINACION-ADELGAZADO(CD_{g_i})
    $A'_i \leftarrow$ COMPONER(g_i,$s(g_i)$)
    $A' \leftarrow A' \cup \{A'_i\}$
  fin para
  regresar $GANTT - T(A', k)$
fin si

cada recorrido de los segmentos indicados en el vértice v_t. De este modo, si se tiene un calendario s cuyos vértices v_t y $v_{(t+1)}$ indican el recorrido simultáneo de dos segmentos $\sigma_{ix_i^{(t)}}$ y $\sigma_{jx_j^{(t)}}$, donde $tr_{ix_i^{(t)}} >> tr_{jx_j^{(t)}}$, seguido por el recorrido del segmento $\sigma_{jx_j^{(t+1)}}$, respectivamente. Entonces el recorrido del segmento $\sigma_{jx_j^{(t+1)}}$ deberá aguardar al menos hasta que $\sigma_{ix_i^{(t)}}$ sea recorrido por completo para poder iniciar con su recorrido correspondiente. El adelgazador permite que el robot A_j pueda iniciar el recorrido del segmento $\sigma_{jx_j^{(t+1)}}$ sin que exista colisión, tan pronto como la región a recorrer esté libre y se haya completado el recorrido del segmento previo. Un ejemplo de esto se puede ver con los segmentos $\sigma_{11}, \sigma_{12}, \sigma_{21}$ y σ_{22} de las Figuras 17 y 18; se puede observar que en la Figura 17 el calendario de coordinación marcado con la línea continua de color azul indica el recorrido simultáneo de los segmentos σ_{11} y σ_{21} seguido por el recorrido el recorrido simultáneo del segmento σ_{12} y σ_{22}, éstos dos últimos segmentos deben esperar la finalización del recorrido de los dos segmentos anteriores para poder iniciar, como se ve en la Figura 18(a). Sin embargo, en la Figura 18(b) se observa que a diferencia del segmento σ_{22}, el segmento σ_{12} no requiere esperar a que ambos segmentos anteriores finalicen para poder iniciar su recorrido y al mismo tiempo garantizar que no exista una colisión.

Para ello es necesario redefinir la función f (calendarizador) con la que se calcula el calendario γ (Sección 3.1). Sea $\pi_j(\sigma) = (x, y)$ el par de índices (x, y) del *j*-ésimo segmento σ_{xy} que atravesó la región r tal que $\sigma \in r$. La función adelgazada $f^T : S \rightarrow \Gamma$ se define como

$$f^T(s) = \gamma = \left\{ \begin{array}{ccc} \iota_{11} & \cdots & \iota_{1m} \\ \vdots & \ddots & \vdots \\ \iota_{n1} & \cdots & \iota_{nm} \end{array} \right\}, \tag{23}$$

donde por cada vértice v_t en s, por cada coordenada $x_i^{(t)}$ en v_t,

$$\iota_{ix_i^{(t+1)}} = \text{máx}(\iota_{ix_i^{(t)}} + (tr_{ix_i^{(t)}} \cdot \mid x_i^{(t)} - x_i^{(t+1)} \mid), \underset{j \in \{1,2,\cdots,t\}}{\text{máx}} (\iota_{\pi_j(\sigma_{ix_i^{(t+1)}})} + tr_{\pi_j(\sigma_{ix_i^{(t+1)}})}) \cdot \mid x_i^{(t)} - x_i^{(t+1)} \mid), \tag{24}$$

Ejemplo 6. Sea $A = \{A_1, A_2, A_3\}$ un conjunto de 3 robots, con rutas $\tau_1 = \{\sigma_{11} \in r_3, \sigma_{12} \in r_2\}$, $\tau_2 = \{\sigma_{21} \in r_1, \sigma_{22} \in r_3\}$, $\tau_3 = \{\sigma_{31} \in r_1, \sigma_{32} \in r_2\}$, respectivamente. Los tiempos de recorrido de cada segmento son $tr_{11} = 2, tr_{12} = 1, tr_{21} = 2, tr_{22} = 4, tr_{31} = 2, tr_{32} = 1$. En la Figura 19(a) se muestra el diagrama de coordinación correspondiente, por simplicidad la longitud de cada lado que conforma un cuadro se muestra igual. Sea $s_{v_f} = \{(0,0,0), (0,1,0), (0,1,1), (1,1,1), (2,1,1), (2,2,1), (2,2,2)\}$ un calendario de coordinación sobre el diagrama de coordinación del conjunto A mostrado en la Figura 19(b).

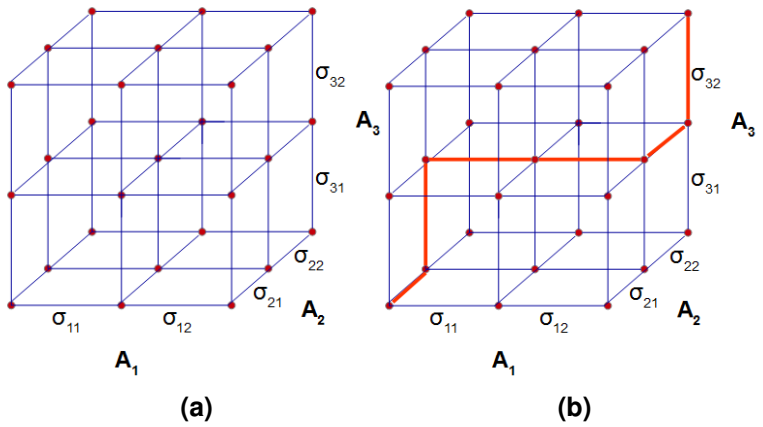


Figura 19: (a) Representación gráfica del diagrama y (b) calendario de coordinación del Ejemplo 6.

El calendario de tiempos de inicio γ, para este ejemplo, se construye de la siguiente manera:

- $t = 0$, $v_0 = (0,0,0)$, $v_{(0+1)} = (0,1,0)$

- $i = 1$, $x_1^{(1)} = 0$
  - $\iota_{10} = 0$, no se incluye en γ
- $i = 2$, $x_2^{(1)} = 1$, $\text{máx}_j(\iota_{\pi_j(\sigma_{21})} + tr_{\pi_j(\sigma_{21})}) = 0$
  - $\iota_{21} = \text{máx}(0 + (0 \cdot |0 - 1|), 0) = 0$
- $i = 3$, $x_3^{(1)} = 0$
  - $\iota_{30} = 0$, no se incluye en γ

$\gamma = \left\{ \begin{array}{cc} \cdot & \cdot \\ 0 & \cdot \\ \cdot & \cdot \end{array} \right\}$, ver Figura 20(a).

- $t = 1$, $v_1 = (0, 1, 0)$, $v_{(1+1)} = (0, 1, 1)$
  - $i = 1$, $x_1^{(2)} = 0$
    - $\iota_{10} = 0$, no se incluye en γ
  - $i = 2$, $x_2^{(2)} = 1$, $\text{máx}_j(\iota_{\pi_j(\sigma_{21})} + tr_{\pi_j(\sigma_{21})}) = 2$
    - $\iota_{21} = \text{máx}(0 + (0 \cdot |0 - 0|), 2 \cdot |0 - 0|) = 0$
  - $i = 3$, $x_3^{(2)} = 1$, $\text{máx}_j(\iota_{\pi_j(\sigma_{31})} + tr_{\pi_j(\sigma_{31})}) = 2$
    - $\iota_{31} = \text{máx}(0 + (0 \cdot |0 - 1|), 2 \cdot |0 - 1|) = 2$

$\gamma = \left\{ \begin{array}{cc} \cdot & \cdot \\ 0 & \cdot \\ 2 & \cdot \end{array} \right\}$, ver Figura 20(b).

- $t = 2$, $v_2 = (0, 1, 1)$, $v_{(2+1)} = (1, 1, 1)$
  - $i = 1$, $x_1^{(3)} = 1$, $\text{máx}_j(\iota_{\pi_j(\sigma_{11})} + tr_{\pi_j(\sigma_{11})}) = 2$
    - $\iota_{11} = \text{máx}(0 + (0 \cdot |0 - 1|), 0 \cdot |0 - 1|) = 0$
  - $i = 2$, $x_2^{(3)} = 1$, $\text{máx}_j(\iota_{\pi_j(\sigma_{21})} + tr_{\pi_j(\sigma_{21})}) = 4$
    - $\iota_{21} = \text{máx}(0 + (2 \cdot |0 - 0|), 4 \cdot |0 - 0|) = 0$

- $i = 3$, $x_3^{(3)} = 1$, $\max_j(\iota_{\pi_j(\sigma_{31})} + tr_{\pi_j(\sigma_{31})}) = 4$
  - $\iota_{31} = \max(2 + (2 \cdot |0 - 0|), 4 \cdot |0 - 0|) = 2$

$\gamma = \left\{ \begin{array}{cc} 0 & \cdot \\ 0 & \cdot \\ 2 & \cdot \end{array} \right\}$, ver Figura 20(c).

- $t = 3$, $v_3 = (1, 1, 1)$, $v_{(3+1)} = (2, 1, 1)$
  - $i = 1$, $x_1^{(4)} = 2$, $\max_j(\iota_{\pi_j(\sigma_{12})} + tr_{\pi_j(\sigma_{12})}) = 0$
    - $\iota_{12} = \max(0 + (2 \cdot |0 - 1|), 0 \cdot |0 - 1|) = 2$
  - $i = 2$, $x_2^{(4)} = 1$, $\max_j(\iota_{\pi_j(\sigma_{21})} + tr_{\pi_j(\sigma_{21})}) = 4$
    - $\iota_{21} = \max(0 + (2 \cdot |0 - 0|), 4 \cdot |0 - 0|) = 0$
  - $i = 3$, $x_3^{(4)} = 1$, $\max_j(\iota_{\pi_j(\sigma_{31})} + tr_{\pi_j(\sigma_{31})}) = 4$
    - $\iota_{31} = \max(2 + (2 \cdot |0 - 0|), 4 \cdot |0 - 0|) = 2$

$\gamma = \left\{ \begin{array}{cc} 0 & 2 \\ 0 & \cdot \\ 2 & \cdot \end{array} \right\}$, ver Figura 20(d).

- $t = 4$, $v_4 = (2, 1, 1)$, $v_{(4+1)} = (2, 2, 1)$
  - $i = 1$, $x_1^{(5)} = 2$, $\max_j(\iota_{\pi_j(\sigma_{12})} + tr_{\pi_j(\sigma_{12})}) = 0$
    - $\iota_{12} = \max(0 + (2 \cdot |0 - 0|), 0 \cdot |0 - 0|) = 2$
  - $i = 2$, $x_2^{(5)} = 2$, $\max_j(\iota_{\pi_j(\sigma_{22})} + tr_{\pi_j(\sigma_{22})}) = 2$
    - $\iota_{22} = \max(0 + (2 \cdot |0 - 1|), 2 \cdot |0 - 1|) = 2$
  - $i = 3$, $x_3^{(5)} = 1$, $\max_j(\iota_{\pi_j(\sigma_{31})} + tr_{\pi_j(\sigma_{31})}) = 4$
    - $\iota_{31} = \max(2 + (2 \cdot |0 - 0|), 4 \cdot |0 - 0|) = 2$

$\gamma = \left\{ \begin{array}{cc} 0 & 2 \\ 0 & 2 \\ 2 & \cdot \end{array} \right\}$, ver Figura 20(e).

- $t = 5$, $v_5 = (2, 2, 1)$, $v_{(5+1)} = (2, 2, 2)$
  - $i = 1$, $x_1^{(6)} = 2$, $\text{máx}_j(\iota_{\pi_j(\sigma_{12})} + tr_{\pi_j(\sigma_{12})}) = 3$
    - $\iota_{12} = \text{máx}(2 + (1 \cdot |0 - 0|), 3 \cdot |0 - 0|) = 2$
  - $i = 2$, $x_2^{(6)} = 2$, $\text{máx}_j(\iota_{\pi_j(\sigma_{22})} + tr_{\pi_j(\sigma_{22})}) = 6$
    - $\iota_{22} = \text{máx}(2 + (4 \cdot |0 - 0|), 6 \cdot |0 - 0|) = 2$
  - $i = 3$, $x_3^{(6)} = 1$, $\text{máx}_j(\iota_{\pi_j(\sigma_{32})} + tr_{\pi_j(\sigma_{32})}) = 3$
    - $\iota_{31} = \text{máx}(2 + (2 \cdot |0 - 1|), 3 \cdot |0 - 1|) = 4$

$\gamma = \left\{ \begin{array}{cc} 0 & 2 \\ 0 & 2 \\ 2 & 4 \end{array} \right\}$, ver Figura 20(f).

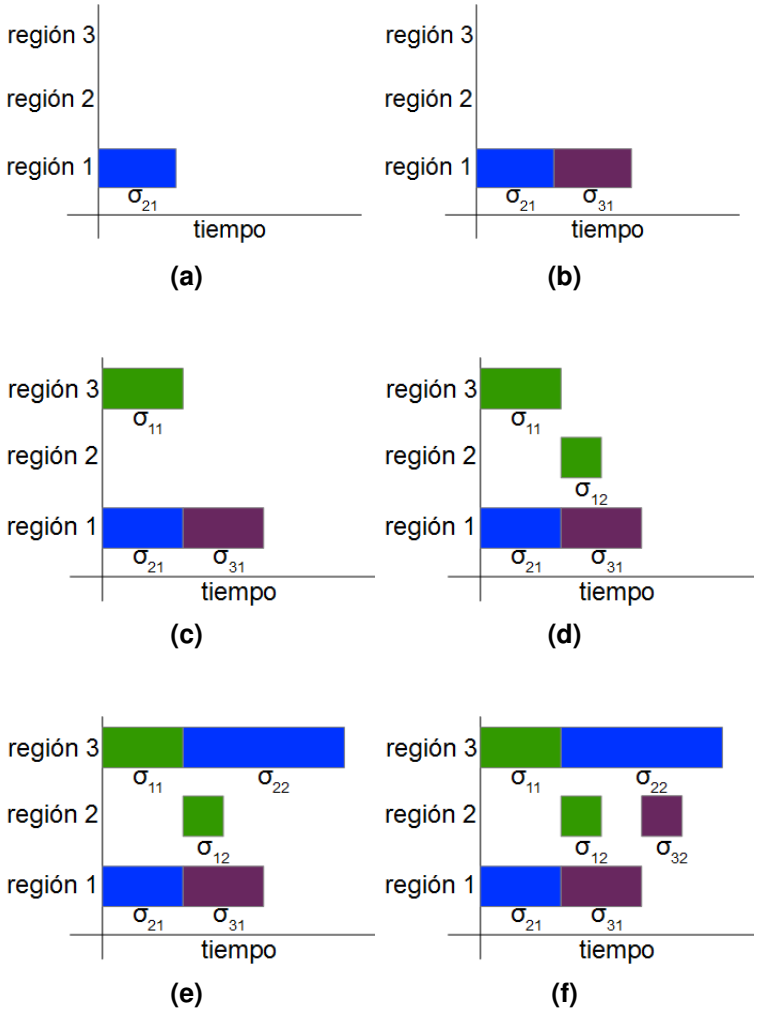


Figura 20: Construcción del diagrama de Gantt conforme al calendario de tiempos de inicio γ del Ejemplo 6.

4.3.2. Procedimiento CALCULAR-CALENDARIO-COORDINACION-ADELGAZADO()

El procedimiento CALCULAR-CALENDARIO-COORDINACION-ADELGAZADO() es similar al procedimiento CALCULAR-CALENDARIO-COORDINACION() descrito en la Sección 4.2.3, la diferencia principal entre ambos está en la definición de la función recursiva. En este procedimiento, se reemplaza la función recursiva descrita por la Ecuación 20 por la función recursiva descrita a continuación

$$s_v = \begin{cases} \arg\min_{s_u \cup \{v\} | u \in \overleftarrow{N}(v)} T_{max}(f^T(s_u \cup \{v\})), \\ \{v_0\}, \text{ if } v = v_0. \end{cases} \tag{25}$$

Donde cada componente de la Ecuación 20 tiene el siguiente significado:

s_v: calendario de coordinación parcial cuyo último elemento es el vértice v,
$\overleftarrow{N}(v)$: conjunto de vértices vecinos previos del vértice v, donde toda coordenada x_i de $u \in \overleftarrow{N}(v)$ es menor o igual que la coordenada x'_i de v,
T_{max}: tiempo máximo de finalización de recorrido (total o parcial) entre todos los robots,
$f^T(s)$: función (calendarizador) *adelgazada* que transforma un calendario de coordinación s en un calendario de tiempos de inicio γ,
v_0: vértice origen del calendario de coordinación, cada una de las componentes se describe a detalle en el Capítulo 3.

Observe que esta versión del procedimiento implementa la función f^T para construir un calendario de tiempos de inicio adelgazado descrito en la Sección 4.3.1. En este procedimiento se calculan calendarios de coordinación que no contengan vértices diagonales; diagonales en la rejilla del diagrama de coordinación. Estos calendarios de coordinación s se pueden ver como una secuencia de segmentos de las rutas τ. Dejando de lado los vértices vecinos diagonales, un calendario de coordinación requiere de $O(km^k)$ comparaciones para ser calculado, en comparación de las $O((2m)^k)$ requeridas en el procedimiento CALCULAR-CALENDARIO-COORDINACION().

4.3.3. Análisis del tiempo de ejecución del Algoritmo 2

En la Tabla 2 se muestra el costo computacional por línea de código del algoritmo GANTT-T. Un cálculo eficiente de $f^T(s)$ para un calendario de coordinación s no requiere más de $O(nm)$ número de pasos computacionales. Por el adelgazador $(f^T(s))$, la línea 8 del Algoritmo 2 es la única que se ve afectada en tiempo de ejecución con respecto al resto, quedando de la siguiente manera $O(nm \cdot k(m')^k)$. Sin embargo, $m' = O(nm)$ (Ecuación 22, Sección 4.2.5), entonces tenemos que la línea 8 tiene un costo de $O(k(m')^{(k+1)})$. Dado que el algoritmo se manda a llamar a si mismo, se obtiene la recurrencia:

$$T(n) = \begin{cases} \theta(1), & \text{si } n = 1; \\ T(n/k) + O(n(m')^{(k+1)}), & \text{en otro caso.} \end{cases} \tag{26}$$

Donde

$T(n)$: tiempo de ejecución para una entrada de tamaño n,

n: número de robots,

m': número máximo de segmentos,

k: constante de agrupamiento ($k = O(1)$).

Tabla 2: Costo computacional por línea del algoritmo GANTT-T.

Línea	Costo
1 - 4	$\theta(1)$
5	$O(\frac{n}{k})$
6	$\theta(1)$ $\frac{n}{k}$ veces
7	$O((m')^k)$ $\frac{n}{k}$ veces
8	$O(k(m')^{(k+1)})$ $\frac{n}{k}$ veces
9	$O(m')$ $\frac{n}{k}$ veces
10 - 11	$\theta(1)$
12	$T(\frac{n}{k})$

Proposición 4.2 *El algoritmo GANTT-T tiene un costo computacional de* $O(n(nm)^{(k+1)} \log_k(n))$ *pasos.*

Demostración

La demostración es similar a la de la Proposición 4.1 con $T(n) = O(n(nm)^{(k+1)} \log_k(n))$ y reemplazando la Ecuación 21 por la Ecuación 26. ■

4.3.4. Discusión

Por un costo adicional de nm por calendario, el algoritmo puede encontrar soluciones T_{max} menos holgadas e incluso pudiera encontrar soluciones $T_{max} = T_{opt}$ cuando $k = n$. Esto implica ($k = n$) que se requiere tiempo exponencial para que el Algortimo 2 pueda calcular un calendario óptimo (si es que puede encontrarlo), no obstante es congruente con la complejidad del problema. Adicionalmente, debido a la restricción impuesta de construir calendarios de coordinación con vértices vecinos no diagonales entre si, los

calendarios de coordinación secuenciales reducen el vecindario de los vértices en un diagrama de coordinación *k*-dimensional, de $|\overleftarrow{N}(v)| = 2^k - 1$ a $|\overleftarrow{N}(v)| = k$ para el vecindario previo y de $|\overrightarrow{N}(v)| = 2^k - 1$ a $|\overrightarrow{N}(v)| = k$ para el vecindario posterior, a un vértice v. Por lo tanto el número combinatorio de calendarios posibles se ve reducido. Sin embargo, esto implica que una ruta compuesta cualquiera C_τ, de un robot compuesto C, tenga una longitud de tantos segmentos de ruta compuestos como la suma de los segmentos de ruta de los robots que componen a C.

Por otra parte, el algoritmo continúa con la limitante de generar calendarios dependientes del orden de los robots en la agrupación. De modo que, un calendario s_{v_f} sigue estando determinado por el orden de la agrupación de robots con la que se genera el diagrama de coordinación.

4.4. Algoritmo GANTT-T+

El algoritmo GANTT-T+ aborda el problema de generar calendarios distintos dependientes del orden de los robots en una agrupación. Por ello, los robots simples, una vez agrupados, se ordenan. El criterio de ordenamiento es la suma de los tiempos de recorrido de una ruta, es decir $\sum_{j=1}^{m}(tr_{ij})$ para un robot A_i. De modo que para una agrupación ordenada de k robots $\{A_{\pi(1)}, A_{\pi(2)}, \cdots, A_{\pi(k)}\}$, $A_{\pi(i)}$ precede a $A_{\pi(i')}$ en el conjunto si $\sum_{j=1}^{m}(tr_{ij}) > \sum_{j=1}^{m}(tr_{i'j})$, $\forall i, i' \in \{1, 2, \cdots, k\}$, $i \neq i'$. El proceso de ordenamiento sólo se realiza con robots simples, por lo tanto únicamente se realiza en la primer iteración y una sola vez. De este modo existe una única permutación del orden para alguna agrupación de k robots $\{A_1, A_2, \cdots, A_k\}$, y por lo tanto existe un único calendario de coordinación independiente del orden de la agrupación inicial. Note que esto no implica que el resultado final sea independiente del ordenamiento del conjunto original A. Este algoritmo conserva la idea principal del algoritmo GANTT y una mejora del *adelgazador* del algoritmo GANTT-T. La variante propuesta se describe en el Algoritmo 3.

4.4.1. Adelgazador+

El *adelgazador* que se propone en la Sección 4.3.1, calcula el tiempo de inicio de un segmento de ruta σ_{ij} tan pronto como se cumplan las siguientes dos condiciones: se libere la región a ser recorrida y el segmento $\sigma_{i(j-1)}$ que lo precede haya concluido.

Algoritmo 3 GANTT-T+(A,k)

si $|A| = 1$ **entonces**
  regresar s asociado al único robot (compuesto) en A
si no
  $A' \leftarrow \emptyset$
  $G \leftarrow$ AGRUPAR-ORDENAR(A,k)
  para $g_i \in G$ **hacer**
    $CD_{g_i} \leftarrow$ CONSTRUIR-DIAGRAMA-COORDINACION(g_i,k)
    $s(g_i) \leftarrow$ CALCULAR-CALENDARIO-COORDINACION-ADELGAZADO+(CD_{g_i})
    $A'_i \leftarrow$ COMPONER(g_i,$s(g_i)$)
    $A' \leftarrow A' \cup \{A'_i\}$
  fin para
  regresar $GANTT - T + (A', k)$
fin si

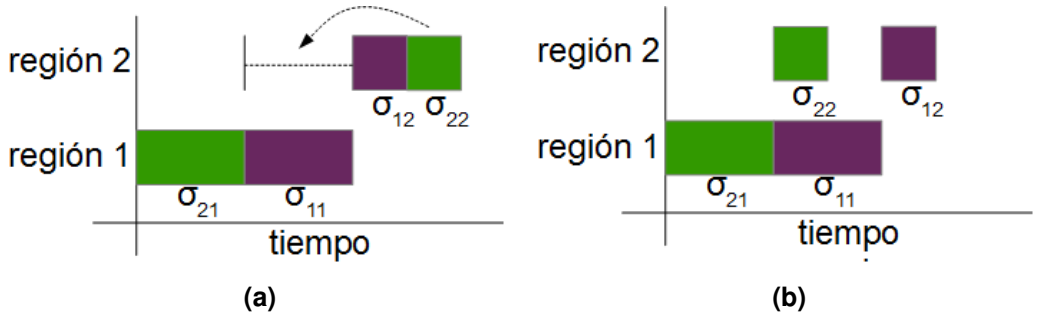


Figura 21: Ejemplo de diagrama de Gantt generado (a) por la función f^T y (b) por la función f^{T+}.

Sin embargo, pueden existir casos (sobretodo coordinando robots compuestos) donde el recorrido del segmento de ruta σ_{ij} se pudiera iniciar y concluir antes que inicie el recorrido de otro segmento de ruta σ_{xy} que ya se haya calendarizado, donde ambos pertenecen a la misma región. En la Figura 21(a) se muestra un ejemplo de un diagrama de Gantt donde ocurre dicho caso, la línea punteada muestra la ranura de tiempo que el segmento σ_{22} puede aprovechar sin violar las restricciones del problema.

Para ello es necesario redefinir la función f con la que se calcula el calendario γ (Sección 3.1). Sea $\pi_j(\sigma) = (x, y)$ el par de índices (x, y) del *j*-ésimo segmento σ_{xy} que atravesó la región r tal que $\sigma \in r$. La función adelagazadora $f^{T+} : S \rightarrow \Gamma$ se define como

$$f^{T+}(s) = \gamma = \left\{ \begin{array}{ccc} \iota_{11} & \cdots & \iota_{1m} \\ \vdots & \ddots & \vdots \\ \iota_{n1} & \cdots & \iota_{nm} \end{array} \right\}, \tag{27}$$

donde por cada vértice v_t en s, por cada coordenada $x_i^{(t)}$ en v_t,

$$\begin{aligned} &\iota_{ix_i^{(t+1)}} = \\ &\quad \text{máx}(\iota_{ix_i^{(t)}} + (tr_{ix_i^{(t)}} \cdot \mid x_i^{(t)} - x_i^{(t+1)} \mid), \min_{j \in \{1,2,\cdots,t\}} (\iota_{\pi_j(\sigma_{ix_i^{(t+1)}})} + tr_{\pi_j(\sigma_{ix_i^{(t+1)}})} \text{ tal que} \\ &\quad \iota_{\pi_j(\sigma_{ix_i^{(t+1)}})} - (\iota_{\pi_{(j-1)}(\sigma_{ix_i^{(t+1)}})} + tr_{\pi_{(j-1)}(\sigma_{ix_i^{(t+1)}})}) \geq tr_{ix_i^{(t+1)}} \ \wedge \\ &\quad \iota_{i(x_i^{(t)}-1)} + tr_{i(x_i^{(t)}-1)} \leq \iota_{\pi_j(\sigma_{ix_i^{(t+1)}})}) \cdot \mid x_i^{(t)} - x_i^{(t+1)} \mid) \end{aligned} \tag{28}$$

Observe que el primer argumento de la función máx del *adelgazador+* es similar a su homólogo del *adelgazador* (ver Sección 4.3.1), mientras que el segundo argumento es distinto. En el *adelgazador*, el segundo argumento de la función máx busca el máximo tiempo de finalización de un segmento recorrido en una región en específico ($\text{máx}_{j \in \{1,2,\cdots,t\}}(\iota_{\pi_j(\sigma_{ix_i^{(t+1)}})} + tr_{\pi_j(\sigma_{ix_i^{(t+1)}})})$), mientras que el argumento homólogo en el *adelgazador+* busca el mínimo tiempo de finalización de un segmento en una región en específico que cumpla con dos desigualdades ($\min_{j \in \{1,2,\cdots,t\}}(\iota_{\pi_j(\sigma_{ix_i^{(t+1)}})} + tr_{\pi_j(\sigma_{ix_i^{(t+1)}})} \mid \iota_{\pi_j(\sigma_{ix_i^{(t+1)}})} - (\iota_{\pi_{(j-1)}(\sigma_{ix_i^{(t+1)}})} + tr_{\pi_{(j-1)}(\sigma_{ix_i^{(t+1)}})}) \geq tr_{ix_i^{(t+1)}} \ \wedge \ \iota_{i(x_i^{(t)}-1)} + tr_{i(x_i^{(t)}-1)} < \iota_{\pi_j(\sigma_{ix_i^{(t+1)}})})$). La primer desigualdad requiere que la diferencia de tiempo entre el tiempo de finalización de recorrido y el tiempo de inicio del *j*-ésimo y *(j-1)*-ésimo segmentos de ruta recorridos en una región en específico, sea mayor o igual al tiempo de recorrido del segmento de ruta en cuestión ($\iota_{\pi_j(\sigma_{ix_i^{(t+1)}})} - (\iota_{\pi_{(j-1)}(\sigma_{ix_i^{(t+1)}})} + tr_{\pi_{(j-1)}(\sigma_{ix_i^{(t+1)}})}) \geq tr_{ix_i^{(t+1)}}$). La segunda desigualdad requiere que el tiempo de finalización de recorrido del segmento de ruta previo al segmento en cuestión, sea menor o igual al tiempo de inicio del *j*-ésimo segmento de ruta que pertenece a la misma región que el segmento en cuestión ($\iota_{i(x_i^{(t)}-1)} + tr_{i(x_i^{(t)}-1)} \leq \iota_{\pi_j(\sigma_{ix_i^{(t+1)}})}$). Existen casos en donde ambas funciones máx y mín del *adelgazador* y del *adelgazador+* coinciden, y serán aquellos en donde las desigualdades del *adelgazador+* no se puedan cumplir con excepción del último segmento de ruta que fue recorrido en la región del

segmento en cuestión.

Ejemplo 7. Sea $A = \{A_1, A_2\}$ un conjunto de 2 robots, con rutas $\tau_1 = \{\sigma_{11} \in r_1, \sigma_{12} \in r_2\}$, $\tau_2 = \{\sigma_{21} \in r_1, \sigma_{22} \in r_2\}$, respectivamente. Los tiempos de recorrido de cada segmento son $tr_{11} = 2, tr_{12} = 1, tr_{21} = 2, tr_{22} = 1$. Sea $s = (0,0), (0,1), (1,1), (2,1), (2,2)$ un calendario de coordinación para el conjunto de robots A. A partir del calendario s, el *adelgazador* genera un calendario γ cuyo diagrama de Gantt se muestra en la Figura 21(a), mientras que el *adelgazador+* genera un diagrama de Gantt como se muestra en la Figura 21(b). Observe además que los tiempos máximos de finalización $T_{max}(f^T(s)) = 6$ y $T_{max}(f^{T+}(s)) = 5$, respectivamente, son distintos. El calendario de tiempos de inicio γ construido por el *adelgazador+* ($f^{T+}(s)$) se realiza de manera similar al Ejemplo 6 de la Sección 4.3.1, tomando en cuenta la definición del argumento distinto y las desigualdades que conlleva.

4.4.2. Procedimiento AGRUPAR-ORDENAR()

El procedimiento AGRUPAR() recibe como entrada el conjunto A de robots (compuestos) y el entero k. La salida es un conjunto G de $|A|/k$ grupos de k robots (compuestos) cada uno. La agrupación se hace de la siguiente manera: se agrupan los primeros k robots en A; si los robots en A son simples, se ordenan de manera no creciente según la suma total de los tiempos de recorrido de cada robot $\sum_{j=1}^{m}(tr_{ij})$, posteriormente los siguientes k, y así sucesivamente hasta agrupar los últimos r robots, donde r es el residuo de la división $|A|/k$.

Ejemplo 8. Si se tiene un conjunto A de 10 robots y se requiere agruparlos en grupos de tamaño 4. El procedimiento genera un conjunto G de $\lceil 10/4 \rceil = 3$ grupos de robots, dos grupos de 4 robots y uno de 2. Una vez formados los grupos, los robots se ordenan de manera no creciente. De modo que, si se supone que $\sum_{j=1}^{m}(tr_{1j}) > \sum_{j=1}^{m}(tr_{4j}) > \sum_{j=1}^{m}(tr_{2j}) > \sum_{j=1}^{m}(tr_{3j})$; $\sum_{j=1}^{m}(tr_{6j}) > \sum_{j=1}^{m}(tr_{7j}) > \sum_{j=1}^{m}(tr_{5j}) > \sum_{j=1}^{m}(tr_{8j})$; $\sum_{j=1}^{m}(tr_{9j}) > \sum_{j=1}^{m}(tr_{10j})$, el conjunto de agrupaciones G en la primer iteración del algoritmo es $G = \{(A_1, A_4, A_2, A_3), (A_6, A_7, A_5, A_8), (A_9, A_{10})\}$. En la Figura 22 se puede ver un ejemplo de las agrupaciones en cada iteración del algoritmo con $n = 10$ y $k = 4$.

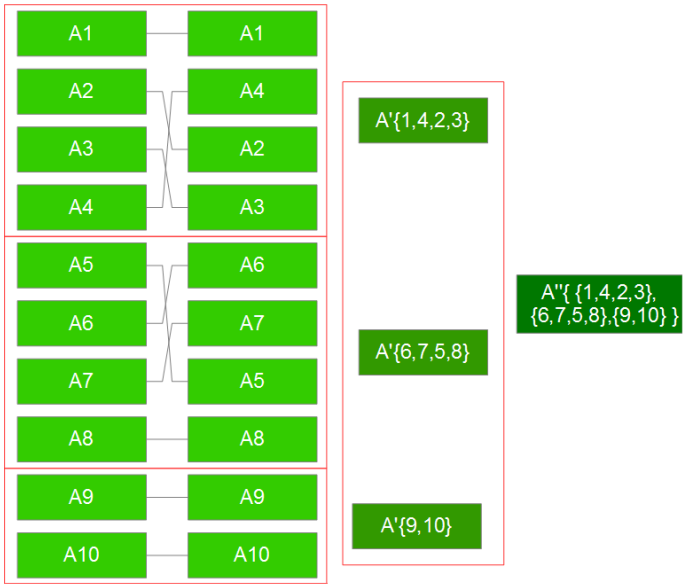


Figura 22: Ejemplo de las agrupaciones en cada iteración del algoritmo GANTT-T+ con $n = 10$ y $k = 4$ (Ejemplo 8).

4.4.3. Procedimiento CALCULAR-CALENDARIO-COORDINACION-ADELGAZADO+()

El procedimiento CALCULAR-CALENDARIO-COORDINACION-ADELGAZADO+() implementa diferencias significativas con respecto al procedimiento CALCULAR-CALENDARIO-COORDINACION-ADELGAZADO() descrito en la Sección 4.3.2. Esta vez, se requiere calcular un calendario de coordinación s que cumpla con ciertas características, entre ellas que s sea una secuencia no decreciente de vértices no diagonales, que esté ordenado y que ofrezca una cierta garantía del grado máximo de "deterioro" para T_{max}. Dichas características se describen a continuación.

Primero, se requiere que s sea una secuencia no decreciente de vértices no diagonales. Recordando, por definición s siempre ha sido una secuencia no decreciente de vértices (ver Sección 3.1). Además, a partir del procedimiento CALCULAR-CALENDARIO-COORDINACION-ADELGAZADO() del algoritmo GANTT-T, se restringe al calendario s a ser conformado por vértices vecinos que indiquen el recorrido de no más de un segmento de ruta a la vez (vértices vecinos no diagonales).

Segundo, se requiere que s esté ordenado. Para esto, es necesario incluir una cierta semántica en s para definir un orden. Sea $\lambda(\sigma_{ij}, s) = t$, la función que regresa el índice t

del primer vértice que indica el fin del recorrido del segmento σ_{ij}. Entonces, se dice que s está ordenado si para cualesquier par de segmentos σ_{ij} y σ_{xy}, se satisface la siguiente desigualdad

$$\lambda(\sigma_{ij}, s) > \lambda(\sigma_{xy}, s) \text{ tal que,}$$
$$\iota_{ij} + tr_{ij} > \iota_{xy} + tr_{xy} \text{ o,}$$
$$\iota_{ij} + tr_{ij} = \iota_{xy} + tr_{xy} \text{ si el índice de } r \text{ es mayor al índice de } r', \text{ donde } \sigma_{ij} \in r \text{ y } \sigma_{xy} \in r'.$$

Esta desigualdad indica que para todo segmento cuyo recorrido fue indicado, en s, después que otro, el tiempo de finalización de recorrido de éste debe ser mayor que el de los segmentos anteriores. En caso que dos segmentos coincidan en tiempo de finalización de recorrido, aquel que se recorra en una región cuyo índice es mayor deberá indicarse después que aquel con un menor índice.

Tercero, se requiere que s ofrezca una garantía de "deterioro" de T_{max}. Sea v un vértice cualquiera, se denota S_v^{min} como el conjunto de calendarios parciales tal que

$$S_v^{min} = \arg\min_{s_v \in S_v}(\max_{w \in \overrightarrow{N}(v)} (T_{max}(f^{T+}(s_v \cup \{w\})))),$$

donde $s_v \in S_v^{min}$ está ordenado y $S_v^{min} \subseteq S_v$. Esto puede leerse como el conjunto de calendarios parciales ordenados cuyo último elemento es el vértice v, donde para todo $s_v \in S_v^{min}$ se garantiza el menor "deterioro" de T_{max} cuando, para cualquier vértice vecino w siguiente de v ($w \in \overrightarrow{N}(v)$), w se añada al calendario s_v, *i.e.* $s_v \cup \{w\}$.

Antes de definir la ecuación para calcular el calendario de coordinación, se define una relación de dominancia $\prec$ entre cualesquier par de calendarios (parciales) ordenados s_v y $s'_v \in S_v$. Sea I el conjunto de índices de las rutas τ ordenadas de manera no creciente tal que aquella de menor índice tenga una suma de tiempo de recorridos mayor, es decir

$$\sum_{j=0}^{m} tr_{ij} \geq \sum_{j=0}^{m} tr_{(i+1)j}, \quad \forall i \in I \setminus \{n\}.$$

Se dice que $s_v \prec s'_v$ para $f^{T+}(s_v)$ y $f^{T+}(s'_v)$, si

$$min_{i \in I}(i \mid T_i < T'_i) < min_{i \in I}(i \mid T'_i < T_i).$$

Finalmente, en este procedimiento se reemplaza la función recursiva descrita en la Ecuación 20 por la siguiente función

$$s^*_v = \begin{cases} s_v \in S^{min}_v \mid s'_v \not\prec s_v, s'_v \in S^{min}_v \\ \{v_0\}, \text{ if } v = v_0. \end{cases} \tag{29}$$

donde,

$$S^{min}_v = \underset{s^*_u \cup \{v\} | u \in \overleftarrow{N}(v)}{\arg\min} \; (\underset{w \in \overrightarrow{N}(v)}{\max} (T_{max}(f^{T+}(\{s^*_u \cup \{v\}\} \cup \{w\})))). \tag{30}$$

Observe que, en la Ecuación 30, la recursividad está implícita al tener que calcular el calendario óptimo parcial s^*_u con la Ecuación 29. Además esta versión del procedimiento implementa la función f^{T+} para construir un calendario de tiempos de inicio adelgazado descrito en la Sección 4.4.1.

4.4.3.1. Optimalidad

Lo que se enuncia en esta subsección es válido únicamente cuando se calcula el calendario de coordinación de diagramas de coordinación de robots simples, o cuando $k = n$. En esta subsección, entiéndase por s^*_v como el calendario de coordinación óptimo parcial ordenado.

Se define $S^*_v \subseteq S^{min}_v$ como aquel conjunto de calendarios óptimos parciales ordenados tal que $\forall s^*_v \in S^*_v$, $\forall s_v \in S^{min}_v$, $s_v \not\prec s^*_v$. Es necesario recalcar que el tiempo máximo de finalización $T_{max}(f^{T+}(s^*_v))$ de un $s^*_v \in S^*_v$, al que se le llama calendario óptimo parcial, no es necesariamente menor que cualquier otro en S_v. Los calendarios s^*_v en S^*_v garantizan el menor del máximo T_{max} (entre cualquier otro calendario s_v en S_v) cuando se añade cualquier otro vértice vecino posterior w de v, es decir, $\forall s^*_v \in S^*_v, \forall s_v \in S_v$ se tiene que

$$\underset{w \in \overrightarrow{N}(v)}{\max} (T_{max}(f^{T+}(s^*_v \cup \{w\}))) \leq \underset{w \in \overrightarrow{N}(v)}{\max} (T_{max}(f^{T+}(s_v \cup \{w\}))).$$

Observe que los calendarios s_v^*, s_v donde la ecuación anterior corresponda a una igualdad, serán aquellos que por definición pertenezcan a S_v^{min}. Observe además que por definición, los calendarios s_v en S_v^{min} no dominan a los calendarios s_v^* en S_v^* ($s_v \not\prec s_v^*$), dado que $S_v^* \subseteq S_v^{min}$, esto implica que para todo s_v^*, $s_v'^*$ en S_v^*, $s_v'^* \not\prec s_v^*$ y por lo tanto

$$T_i = T_i', i \in I,$$

para $f^{T+}(s_v^*)$ y $f^{T+}(s_v'^*)$, respectivamente. De modo práctico, salvo por el orden de la secuencia de vértices que los componen (sin que esto se refiera a que estén desordenados según la definición de calendario ordenado), los calendarios en S_v^* son idénticos; basta con elegir alguno de ellos bajo un criterio arbitrario dado, para que el calendario óptimo parcial ordenado s_v^* de un vértice v sea único ($|S_v^*| = 1$).

De acuerdo con la definición de calendario óptimo del párrafo anterior, el conjunto $S_{v_f}^{min}$ debe ser aquel que $\forall s_{v_f} \in S_{v_f}^{min}$,

$$\max_{w \in \overrightarrow{N}(v_f)} (T_{max}(f^{T+}(s_{v_f} \cup \{w\}))) \leq \max_{w \in \overrightarrow{N}(v_f)} (T_{max}(f^{T+}(s'_{v_f} \cup \{w\}))), s'_{v_f} \in S_{v_f}.$$

Observe que para el vértice final v_f, no existe ningún vértice vecino siguiente w, es decir, $\overrightarrow{N}(v_f) = \emptyset$, y por lo tanto $s_{v_f} \in S_{v_f}^{min}$ debe ser aquel que $T_{max}(f^{T+}(s_{v_f})) = T_{opt}$. Entonces, por definición, un calendario $s_{v_f} \in S_{v_f}^*$ debe ser aquel que $s_{v_f} \in S_{v_f}^{min}$ y $\forall s'_{v_f} \in S_{v_f}^{min}$, $s'_{v_f} \not\prec s_{v_f}$. De modo que el calendario $s_{v_f} \in S_{v_f}^*$ es aquel calendario cuyo T_{max} es óptimo y además el T_i de las primeras k rutas críticas es menor que el T_i de las primeras k rutas críticas de cualquier otro calendario en $S_{v_f}^{min}$, donde $i \in I \setminus \{k+1, \cdots, n\}$.

La ventaja de definir un conjunto S_v^* para un vértice cualquiera v, es que exhibe la propiedad de la subestructura óptima, es decir, un calendario óptimo (parcial o final) s_v^* incorpora calendarios óptimos parciales que se pueden calcular de manera independiente. Para un vértice v, suponga que se conoce un calendario óptimo parcial s_u^* de S_u^* por cada uno de los vértices vecinos previos u de v, es decir, $u \in \overleftarrow{N}(v)$. Observe entonces que al menos uno de ellos debe cumplir con lo siguiente

$$\max_{w \in \overrightarrow{N}(v)} (T_{max}(f^{T+}(s_u \cup \{v\} \cup \{w\}))) \leq \max_{w \in \overrightarrow{N}(v)} (T_{max}(f^{T+}(s'_u \cup \{v\} \cup \{w\}))),$$

ya que cada uno de esos calendarios óptimos parciales s_u^*, por pertenecer a S_u^{min}, garantiza ser aquel de menor $\max_{v \in \overrightarrow{N}(u)} T_{max}((f^{T+}(s_u \cup \{v\}))$ entre los otros calendarios en S_u. Por lo tanto, un calendario s_u^*, garantiza ser también aquel de menor $\max_{w \in \overrightarrow{N}(v)}$ $T_{max}((f^{T+}(s_u \cup \{v\} \cup \{w\}))$ entre los otros calendarios en S_u. De aquellos calendarios $s_u^* \cup \{v\}$ que pertenecen a S_v^{min}, se elige aquel $s_u^* \cup \{v\}$ que $\forall s_v \in S_v^{min}$, $s_v \not\prec s_u^* \cup \{v\}$. Este calendario $s_u^* \cup \{v\}$ es aquel que por definición pertenezca a S_v^*. Esto quiere decir que, sea v un vértice cualquiera, para todo vértice vecino previo u de v ($u \in \overleftarrow{N}(v)$), si un calendario $s_u \cup \{v\}$ pertenece a S_v^{min} entonces s_u pertenece a S_u^*.

De modo que, sea $s_{v_f}^*$ un calendario óptimo ordenado en $S_{v_f}^*$, éste se puede descomponer recursivamente en un calendario óptimo parcial de algún vértice vecino anterior en unión con el vértice v_f, es decir, $s_{v_f}^* = s_v^* \cup \{v_f\}$, donde s_v^* pertenece a S_v^*, está ordenado y es prefijo de $s_{v_f}^*$.

Proposición 4.3 *Sea* $s_{v_f}^* = \{v_1, v_2, \cdots, v_{(nm)}\}$ *un calendario óptimo ordenado en* $S_{v_f}^*$ *tal que* $\{v_1, v_2, \cdots, v_{nm}\}$ *representa el orden de la secuencia de cada uno de los* nm *vértices del calendario* $s_{v_f}^*$*, donde* $v_1 = v_0$ *y* $v_{nm} = v_f$*. Entonces,* $s_{v_f}^*$ *se puede descomponer recursivamente como* $s_{v_f}^* = s_{v_{(nm-1)}}^* \cup \{v_{nm}\}$ *donde* $s_{v_{(nm-1)}}^* \in S_{v_{(nm-1)}}^*$*,* $s_{v_{(nm-1)}}^* = s_{v_{(nm-2)}}^* \cup \{v_{nm-1}\}$ *donde* $s_{v_{(nm-2)}}^* \in S_{v_{(nm-2)}}^*$*,* $\cdots$*,* $s_{v_2}^* = s_{v_1}^* \cup \{v_2\}$ *donde* $s_{v_1}^* \in S_{v_1}^*$ *y* $s_{v_1}^* = \{v_0\}$*. Además, cada calendario* $s_{v_k}^*$*,* $k \in \{1, 2, \cdots, (nm-1)\}$*, está ordenado y es prefijo de* $s_{v_(k+1)}^*$.

Demostración

Suponga que para algún calendario óptimo parcial ordenado $s_{v_{(k+1)}}^*$ tal que $s_{v_f}^* = s_{v_{(k+1)}}^* \cup \{v_{(k+2)}, v_{(k+3)}, \cdots, v_{(nm)}\}$ donde $v_{(nm)} = v_f$, no existe un calendario óptimo parcial ordenado $s_{v_k}^*$ en $S_{v_k}^*$ tal que $s_{v_{(k+1)}}^* = s_{v_k}^* \cup \{v_{(k+1)}\}$ donde $s_{v_k}^*$ está ordenado y es prefijo de $s_{v_{(k+1)}}^*$. Esto pudiera ocurrir en cuatro casos:

Caso 1: $s_{v_k}^*$ **no está ordenado.** Dado que la descomposición se realiza partiendo de un calendario $s_{v_f}^*$ ordenado, cualquier descomposición de éste y subsecuentes donde sólo se retira el último elemento, serán ordenadas. Por lo tanto $s_{v_k}^*$ está ordenado.

Caso 2: $s^*_{v_k}$ no es prefijo de $s^*_{v_{(k+1)}}$. Dado que la descomposición se realiza partiendo de un calendario $s^*_{v_f}$ ordenado, cualquier descomposición de éste y subsecuentes donde sólo se retira el último elemento, serán prefijos. Por lo tanto $s^*_{v_k}$ es prefijo de $s^*_{(k+1)}$.

Caso 3: $\forall s^*_{v_k} \in S^*_{v_k}, s^*_{v_{(k+1)}} \neq s^*_{v_k} \cup \{v_{(k+1)}\}$. Este caso implicaría que el calendario s_{v_k}, tal que $s^*_{v_{(k+1)}} = s_{v_k} \cup \{v_{(k+1)}\}$, pertenece al conjunto $S_{v_k} \setminus S^*_{v_k}$. Sin embargo, dado que por definición, los calendarios $s^*_{v_k}$ en $S^*_{v_k}$ son aquellos que para todo $w \in \overrightarrow{N}(v_k)$ (entre ellos el vértice $v_{(k+1)}$), $\max_w(T_{max}(f^{T+}(s^*_{v_k} \cup \{w\}))) = \max_w(T_{max}(f^{T+}(s'_{v_k} \cup \{w\})))$ donde $s'_{v_k} \in S^{min}_{v_k}$, o $\max_w(T_{max}(f^{T+}(s^*_{v_k} \cup \{w\}))) < \max_w(T_{max}(f^{T+}(s'_{v_k} \cup \{w\})))$ donde $s'_{v_k} \in S_{v_k} \setminus S^{min}_{v_k}$. De modo que si $s^*_{v_{(k+1)}} = s_{v_k} \cup \{v_{(k+1)}\}$, entonces, siempre se podrá construir un calendario $s'^*_{v_{(k+1)}} = s^*_{v_k} \cup \{v_{(k+1)}\}$ donde $s'^*_{v_k} \in S^*_{v_k}$ tal que para todo $w \in \overrightarrow{N}(v_{(k+1)})$, $\max_w(T_{max}(f^{T+}(s'^*_{v_{(k+1)}} \cup \{w\}))) \leq \max_w(T_{max}(f^{T+}(s^*_{v_{(k+1)}} \cup \{w\})))$ y además que $s'^*_{v_{(k+1)}} \prec s^*_{v_{(k+1)}}$. Dado que $s^*_{v_k}$ debe ser aquel calendario tal que ningún otro calendario en $S^{min}_{v_k}$ lo domine, entonces $s'^*_{v_{(k+1)}} \prec s^*_{v_{(k+1)}}$ implica una contradicción y por lo tanto $\exists s^*_{v_k} \in S^*_{v_k} | s^*_{v_{(k+1)}} = s^*_{v_k} \cup \{v_{(k+1)}\}$.

Caso $S^*_{v_k} = \emptyset$. Dado que es posible construir al menos un calendario de coordinación no decreciente que conecte v_0 con cualquier otro vértice, entonces $S_{v_k} \neq \emptyset$, lo que quiere decir que hay al menos un elemento en S_{v_k}, por lo tanto $S^*_{v_k}$ debe contener al menos ese elemento también. Por lo tanto $S^*_{v_k} \neq \emptyset$.

En cualquiera de los cuatro casos posibles se llega a una contradicción, con lo cual la proposición queda demostrada.

■

En la Proposición 4.3 se garantiza que el calendario de coordinación óptimo ordenado $s^*_{v_f}$ en $S^*_{v_f}$ se puede descomponer en calendarios óptimos parciales. De manera que existe alguna forma de construir $s^*_{v_f}$ a partir de calendarios óptimos parciales ordenados.

Proposición 4.4 *Para todo vértice v, el calendario óptimo (parcial o total) $s^*_v \in S^*_v$ se puede calcular usando la Ecuación 29.*

Demostración

Se demuestra por inducción en k.

Caso base. Cuando $v = v_0$, $s_v^* = \{v_0\}$ lo cual es correcto dado que v_0 es el vértice origen y no existe algún otro calendario para éste, es decir, $|S_{v_0}| = 1$. Por lo tanto $|S_{v_0}^*| = 1$ y $|S_{v_0}^{min}| = 1$.

Hipótesis inductiva. Suponga que para todo vértice vecino previo v_k de $v_{(k+1)}$, es decir, $\forall v_k \in \overleftarrow{N}(v_{(k+1)})$, el calendario de coordinación óptimo parcial ordenado $s_{v_k}^*$ es correcto si se calcula mediante la Ecuación 29.

Paso inductivo. Utilizando la hipótesis anterior, observe que $\forall v_k \in \overleftarrow{N}(v_{(k+1)})$, $\forall s_{v_k} | s_{v_k} \cup \{v_{(k+1)}\} \in S_{v_{(k+1)}}$, $\forall w \in \overrightarrow{N}(v_k)$,

$$\max_w(T_{max}(f^{T+}(s_{v_k}^* \cup \{w\}))) \leq \max_w(T_{max}(f^{T+}(s_{v_k} \cup \{w\}))),$$

por lo tanto $\forall v_k \in \overleftarrow{N}(v_{(k+1)})$, $\forall s_{v_k} | s_{v_k} \cup \{v_{(k+1)}\} \in S_{v_{(k+1)}}$, $\forall w \in \overrightarrow{N}(v_{(k+1)})$,

$$\max_w(T_{max}(f^{T+}(s_{v_k}^* \cup \{v_{(k+1)}\} \cup \{w\}))) \leq \max_w(T_{max}(f^{T+}(s_{v_k} \cup \{v_{(k+1)}\} \cup \{w\}))).$$

De modo que $\forall v_k \in \overleftarrow{N}(v_{(k+1)})$, $\forall s_{v_{(k+1)}} \in S_{v_{(k+1)}}$, $\forall w \in \overrightarrow{N}(v_{(k+1)})$,

$$\arg\min_{s_{v_{(k+1)}}}(\max_w(T_{max}(f^{T+}(s_{v_{(k+1)}} \cup \{w\})))) = \arg\min_{s_{v_k}^* \cup \{v_{(k+1)}\}}(\max_w(T_{max}(f^{T+}(s_{v_k}^* \cup \{v_{(k+1)}\} \cup \{w\})))).$$

Por lo tanto

$$S_{v_{(k+1)}}^{min} = \arg\min_{s_{v_k}^* \cup \{v_{(k+1)}\}}(\max_{w \in \overrightarrow{N}(v_{(k+1)})}(T_{max}(f^{T+}(s_{v_k}^* \cup \{v_{(k+1)}\} \cup \{w\})))),$$

lo cual es coherente con la Ecuación 30. Finalmente, se busca aquel $s_{v_{(k+1)}}^* \in S_{v_{(k+1)}}^{min}$ tal que $\forall s_{v_{(k+1)}} \in S_{v_{(k+1)}}^{min}$, $s_{v_{(k+1)}} \prec s_{v_{(k+1)}}^*$, el cual será aquel $s_{v_{(k+1)}}^*$ que por definición pertenece a $S_{v_{(k+1)}}^*$ y que satisface la Ecuación 29. Por lo tanto utilizando la Ecuación 29, para todo

vértice v, se puede calcular un calendario óptimo $s_v^* \in S_v^*$.

■

Por las Proposiciones 4.3 y 4.4, si $k = n$ y si se calcula $s_{v_f}^*$ con la Ecuación 29, el calendario $s_{v_f}^*$ será aquel que $T_{max}(f^{T+}(s_{v_f}^*)) = T_{opt}$.

4.4.4. Análisis del tiempo de ejecución del Algoritmo 3

Los procedimientos del algoritmo GANTT-T y el algoritmo GANTT-T+ son similares si a costo computacional se refiere. La principal diferencia radica en el proceso de ordenamiento de robots en el agrupamiento que realiza el algoritmo GANTT-T+. Dado que el ordenamiento se lleva a cabo en una sola iteración del algoritmo (Iteración 1), el costo computacional de ordenar cada uno de los n/k grupos de k robots es $\frac{n}{k} \cdot k \log k = n \cdot \log k$ pasos.

Proposición 4.5 *El algortimo GANTT-T+ tiene un costo computacional de* $O(n(nm)^{(k+1)} \log_k(n))$ *pasos.*

Demostración

El costo computacional del algoritmo GANTT-T es $T(n) = O(n(nm)^{(k+1)} \log_k(n))$ (Proposición 4.2). Como el algoritmo GANTT-T+ es equivalente al algoritmo GANTT-T más el proceso de ordenamiento, se puede decir que el costo computacional del algoritmo GANTT-T+ es $T(n) = O(n(nm)^{(k+1)} \log_k(n) + n \log k) = O(n(nm)^{(k+1)} \log_k(n))$ pasos. ■

4.4.5. Factor de aproximación

Sea T_{opt} el valor óptimo del tiempo máximo de finalización (*makespan*) para un caso de RPC.

Proposición 4.6 *El algoritmo GANTT-T+ tiene un factor de aproximación de* $(n/k) \cdot T_{opt}$.

Demostración

Dado un conjunto de robots simples A con rutas preespecificadas y una constante k que determina el tamaño de las agrupaciones, considere una primer iteración en el algoritmo GANTT-T+. Al final de ésta se obtiene un conjunto A' con $\frac{n}{k}$ robots compuestos coordinados de manera óptima. Esto implica que el tiempo máximo de finalización óptimo T_{opt} entre todos los robots simples en A debe ser mayor que cualquier tiempo máximo de finalización entre todos los robots simples que conforman A'_i, por cada $A'_i \in A'$, es decir,

$$T_{opt} \geq T_{max}(f^{T+}(s \in A'_i)), \quad A'_i \in A',$$

similar a la Cota 1 (Sección 3.5). Finalmente, observe que si cada uno de los $\frac{n}{k}$ robots compuestos que pertenecen a A' recorren de manera secuencial su ruta compuesta, el algoritmo GANTT-T+ produce una solución donde todos los robots compuestos habrán finalizado el recorrido de sus rutas compuestas en un tiempo no mayor a

$$\frac{n}{k} \cdot T_{opt} \geq \sum_{A'_i \in A} T_{max}(f^{T+}(s \in A'_i))$$

unidades de tiempo. ■

4.4.6. Discusión

El algoritmo GANTT-T+ particiona el conjunto A en n/k conjuntos de k robots cada uno, diviendo así el problema de tamaño n en n/k problemas de tamaño k (algoritmo GANTT), ordena los robots en las agrupaciones en caso de ser agrupaciones de robots simples, para cada agrupación de k robots calcula el calendario de coordinación adelgazado que minimice su T_{max} (algoritmo GANTT-T). El proceso de ordenamiento le permite al algoritmo GANTT-T+ encontrar un mismo calendario de coordinación para una misma agrupación de robots simples independiente del orden en la agrupación. Con la Ecuación 29 es posible encontrar un calendario tal que $T_{max} = T_{opt}$ cuando $k = n$. Sin embargo, si se cambia la combinación de los robots que conforman las agrupaciones, el resultado no es necesariamente el mismo que con una combinación distinta. De modo que el resultado lo determinan las agrupaciones y no el orden de los robots en éstas.

Capítulo 5. Experimentación y resultados

En el capítulo anterior se hizo una descripción detallada de los algoritmos propuestos. En este capítulo se describen los experimentos realizados y se presentan los resultados obtenidos de la experimentación.

5.1. Casos de prueba

Para evaluar el desempeño de los algoritmos propuestos, tomando ventaja de la equivalencia de los problemas (RPC y JSP), se utilizan casos de prueba conocidos en la literatura para el JSP. Los casos de prueba se obtuvieron de la *OR-Library*, introducida en (Beasley, 1990), la cual es un compendio de casos de prueba para números problemas de investigación de operaciones. En total son 82 casos de prueba, divididos en:

- 5 casos abz5-abz9 (Adams *et al.*, 1988),
- 3 casos ft06, ft10, y ft20 (Fisher y Thompson, 1963),
- 40 casos la01-la40 (Lawrence, 1984),
- 10 casos orb01-orb10 (Applegate y Cook, 1991),
- 20 casos swv01-swv20 (Storer *et al.*, 1992),
- 4 casos yn1-yn4 (Yamada y Nakano, 1992).

Cada uno de los casos de prueba consta de un número *n* de rutas (tareas), una para cada robot, y de un número *m* de segmentos de ruta (operaciones). Cada segmento de ruta indica la región (máquina) que el robot debe recorrer para completar su ruta. Además, para cada segmento de cada ruta se indica el tiempo de recorrido (tiempo de procesamiento) que requiere el robot correspondiente para atravesar la región. Para cada caso de prueba se busca encontrar un calendario de tiempo máximo de finalización (*makespan*) mínimo (ver Sección 3.2). Los casos de prueba presentan tamaños $(n \times m)$ variados, entre los cuales hay: 10×5, 10×10, 10×15, 20×10, 20×20 y 50×10, entre otros. Un mayor tamaño supone un reto mayor para encontrar una solución óptima o cercana al óptimo.

5.2. Experimento 1: GANTT vs. GANTT-T vs. GANTT-T+

Para el primer experimento, se eligieron tres casos de prueba de los 82 disponibles. Los casos de prueba elegidos son: abz05, ft10 y la16, cada uno de ellos consta de $n = 10$ rutas de $m = 10$ segmentos cada uno. La elección de estos tres casos se deriva de evidencia empírica de los resultados obtenidos en pruebas preliminares, donde alguna mejora en los resultados de dos de los casos impactaba de manera negativa en el resultado del tercero.

El experimento se realiza para comparar el desempeño de los tres algoritmos propuestos (ver Capítulo 4) entre sí. Dado que el resultado de los algoritmos propuestos depende del orden del conjunto de entrada A de robots, por cada caso de prueba seleccionado (abz05, ft10 y la16), se realizan $\binom{n}{n/2}$ corridas con ordenamientos fijos distintos para A. Los ordenamientos de A son cada combinación de $\binom{n}{n/2}$ elementos de A concatenando su complemento; y los índices son:

(1,2,3,4,5,6,7,8,9,10)
(1,2,3,4,6,5,7,8,9,10)
⋮
(1,2,3,4,10,5,6,7,8,9)
(1,2,3,5,6,4,7,8,9,10)
⋮
(5,7,8,9,10,1,2,3,4,6)
(6,7,8,9,10,1,2,3,4,5).

Este proceso se repite variando el tamaño de agrupamiento de robots, las pruebas se realizan con valores de $k \in \{2, 3, 4, 5\}$. Entonces, con cada uno de los tres algoritmos se realizan 252 ejecuciones, por 4 tamaños distintos de agrupamientos, por 3 casos de prueba. En total se realizan $(3 \cdot 3 \cdot 4 \cdot 252) = 9072$ ejecuciones para este experimento.

Las Tablas 3, 4 y 5 muestran los resultados obtenidos de las ejecuciones del experimento. En la primera fila se muestra el nombre del caso de prueba y entre paréntesis el valor de la solución óptima T_{opt}. En la primer columna, k indica el tamaño de las agrupa-

ciones de los robots. Las columnas subsecuentes etiquetadas como GANTT, GANTT-T y GANTT-T+, se subdividen en cuatro columnas cada una. Las subdivisiones se etiquetan como x_{min}, x_{max}, $\lfloor \bar{x} \rfloor$ y σ, éstas indican el menor T_{max} encontrado, el mayor T_{max} encontrado, el piso del promedio y la desviación estándar, respectivamente. En los resultados, se puede observar que el GANTT-T+ es significativamente superior a los algoritmos GANTT y GANTT-T.

Tabla 3: Comparación de los resultados obtenidos por algoritmos propuestos para el caso de prueba abz05 y valores de $k \in \{2, 3, 4, 5\}$, resaltando con negritas el mejor resultado encontrado.

abz05 ($T_{opt} = 1234$)												
	GANTT				GANTT-T				GANTT-T+			
k	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ
2	1927	2782	2347	168.33	1325	1693	1496	65.81	1263	1447	1356	34.96
3	1718	2377	2020	130.52	1291	1536	1408	50.07	**1249**	1409	1326	31.65
4	1798	2603	2181	159.22	1287	1598	1424	52.67	1261	1391	1319	28.71
5	1757	2245	2045	90.07	1336	1594	1454	50.09	1280	1428	1347	32.42

Tabla 4: Comparación de los resultados obtenidos por algoritmos propuestos para el caso de prueba ft10 y valores de $k \in \{2, 3, 4, 5\}$, resaltando con negritas el mejor resultado encontrado.

ft10 ($T_{opt} = 930$)												
	GANTT				GANTT-T				GANTT-T+			
k	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ
2	1550	2074	1773	82.68	1019	1256	1145	44.80	988	1149	1066	32.13
3	1373	1725	1546	61.22	1025	1208	1100	37.11	**966**	1125	1046	28.27
4	1434	1812	1604	71.75	997	1193	1089	31.60	978	1101	1031	23.72
5	1340	1649	1506	49.23	1013	1206	1104	33.39	979	1118	1044	27.74

Tabla 5: Comparación de los resultados obtenidos por algoritmos propuestos para el caso de prueba la16 y valores de $k \in \{2, 3, 4, 5\}$, resaltando con negritas el mejor resultado encontrado.

la16 ($T_{opt} = 945$)												
	GANTT				GANTT-T				GANTT-T+			
k	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ
2	1478	2075	1801	104.21	1001	1255	1117	46.13	981	1140	1057	31.12
3	1379	1838	1578	79.82	1006	1203	1080	32.70	**979**	1145	1031	25.84
4	1418	1970	1628	112.25	989	1194	1078	35.09	**979**	1084	1021	21.81
5	1464	1772	1617	62.64	1011	1195	1091	34.82	984	1136	1032	24.65

En las Figuras 23, 24 y 25 se puede observar que los valores $\lfloor \bar{x} \rfloor$ de los primeros dos algoritmos son superados por aquellos del GANTT-T+. Se observa también que para el

GANTT-T+, conforme se aumenta el valor de k, el promedio ($\lfloor \bar{x} \rfloor$) disminuye, teniendo un repunte para cuando $k = 5$.

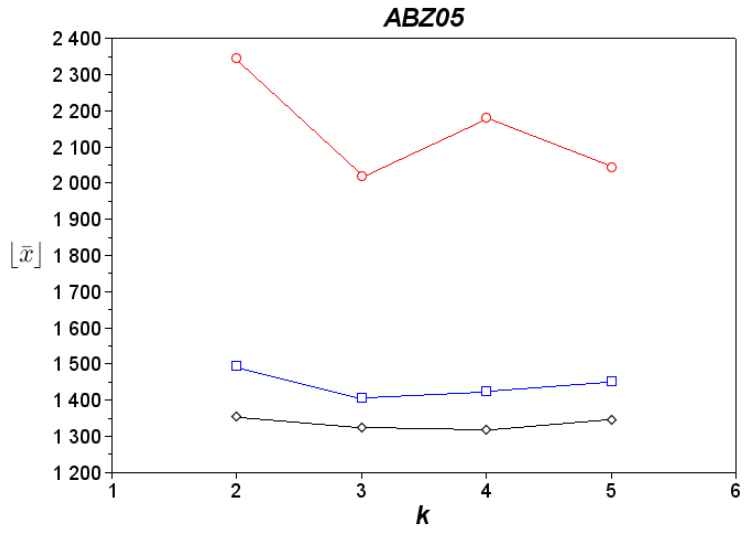


Figura 23: Valores promedio del tiempo máximo de finalización ($\lfloor \bar{x} \rfloor$) para distintos valores de k de las $\binom{n}{n/2}$ corridas de los algoritmos GANTT (◯), GANTT-T (□) y GANTT-T+ (◊) del caso abz05 (mostrado en la Tabla 3).

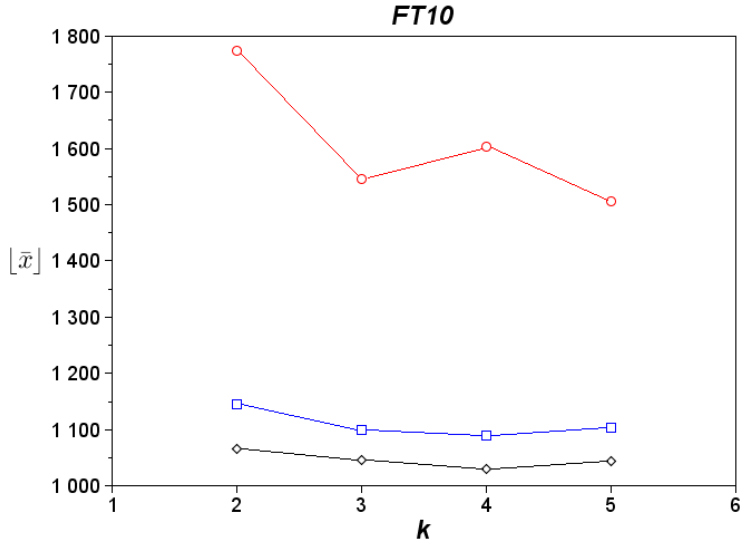


Figura 24: Valores promedio del tiempo máximo de finalización ($\lfloor \bar{x} \rfloor$) para distintos valores de k de las $\binom{n}{n/2}$ corridas de los algoritmos GANTT (◯), GANTT-T (□) y GANTT-T+ (◊) del caso ft10 (mostrado en la Tabla 4).

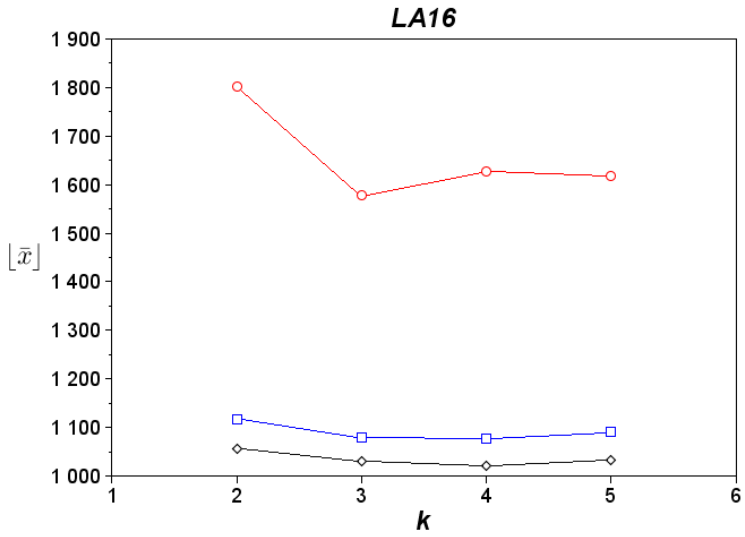


Figura 25: Valores promedio del tiempo máximo de finalización ($\lfloor \bar{x} \rfloor$) para distintos valores de k de las $\binom{n}{n/2}$ corridas de los algoritmos GANTT (◯), GANTT-T (□) y GANTT-T+ (◊) del caso la16 (mostrado en la Tabla 5).

5.3. Experimento 2: GANTT-T+, $k = n$

Para el segundo experimento, se realiza una evaluación del GANTT-T+ con seis casos de prueba: ft06, la01-la05. La intención de este experimento es evaluar el desempeño del algoritmo con un valor de $k = n$. La elección de estos seis casos se debe a que éstos corresponden a los casos de prueba más pequeños entre los 82 casos de prueba disponibles. El objetivo de este experimento es medir el desempeño del GANTT-T+ al calendarizar los movimientos de todos los robots de manera simultánea. La Tabla 6 muestra los casos de prueba analizados, la primer columna presenta el identificador del caso, la segunda su tamaño $(n \times m)$, la tercera el valor óptimo, la cuarta el valor logrado por el algoritmo GANTT-T+ y la última columna muestra el tiempo de ejecución para cada caso. Los detalles de la máquina donde fue corrido el algoritmo se describen en el Experimento 3.

La Figura 26 se muestra un diagrama de Gantt, del calendario óptimo para el caso ft06, que ilustra el esquema de orden y tiempo en el que los segmentos de ruta que cada robot debe recorrer.

Tabla 6: Comparación de los resultados obtenidos de los casos de prueba más pequeños cuando $k = n$.

Caso	$n \times m$	T_{opt}	GANTT-T+ $(k = n)$	$CPU(s)$
ft06	6×6	55	55	2.8
la01	10×5	666	666	3474
la02	10×5	655	655	3465
la03	10×5	597	597	3337
la04	10×5	590	590	3424
la05	10×5	593	593	3389

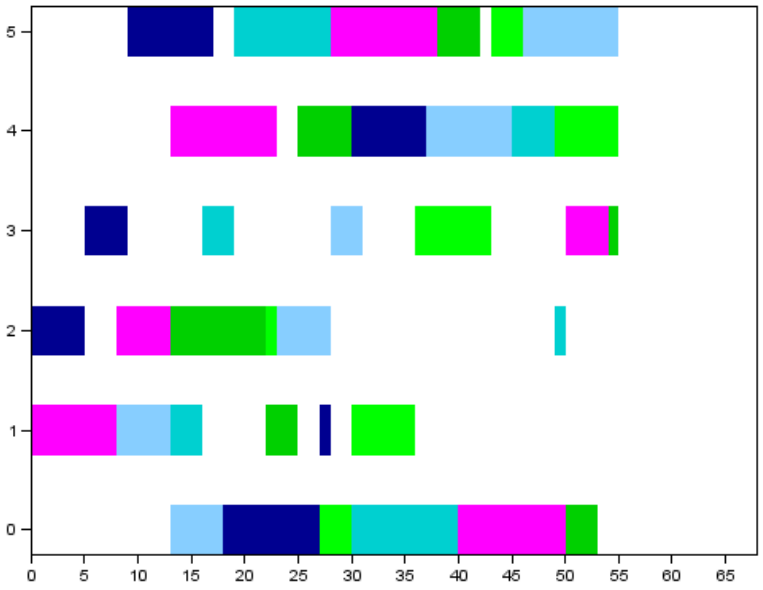


Figura 26: Solución óptima del caso ft06.

5.4. Experimento 3: GANTT-T+, 82 casos

Para el tercer experimento, se realiza una evaluación del GANTT-T+ con los 82 casos de prueba. En este experimento, una prueba consiste en correr el GANTT-T+ con 250 permutaciones aleatorias del orden de los robots y valores de $k \in \{2, 3\}$, donde $3 \leq \lceil \log n \rceil$ para n con cada caso de prueba (con excepción del caso ft06). Las Tablas 7 a 13 resumen los resultados experimentales de los 82 casos de prueba. En las tablas se enlista, el nombre del caso, el tamaño del caso ($n \times m$), la mejor solución conocida (MSC), el tamaño de agrupamiento (k), la mejor solución encontrada (x_{min}), la peor (x_{max}), el promedio ($\lfloor \bar{x} \rfloor$),

la desviación estándar (σ), el error relativo (ER) y el tiempo de cómputo en segundos. El error relativo se calcula de la siguiente manera

$$ER = \frac{x_{min} - MSC}{MSC} \cdot 100. \tag{31}$$

Los resultados experimentales muestran que pocos casos de prueba obtuvieron valores de error relativo (ER) por encima del $15\,\%$. Concretamente, los casos con $ER > 15\,\%$ para alguno de los valores establecidos de k, son: abz08 $(17.05\,\%)$ con $k = 2$, abz08 $(15.81\,\%)$ con $k = 2$, abz09 $(16.79\,\%)$ con $k = 2$, la29 $(15.12\,\%)$ con $k = 2$, swv07 $(15.37\,\%)$ con $k = 2$, swv08 $(16.05\,\%)$ con $k = 2$ y swv09 $(16.01\,\%)$ con $k = 2$. Estos casos representan el $7.31\,\%$ de los 82 casos de prueba y el $4.26\,\%$ si tomamos en cuenta ambos valores de k para cada caso. Más aún, ninguno de ellos supera un error relativo del $17.05\,\%$. Los resultados experimentales muestran además, tiempos de ejecución similares para casos de prueba del mismo tamaño. Por ejemplo, para casos de tamaño 10×10, la mayor diferencia entre cualesquier par de tiempos de cómputo es menor a 1 segundo con $k = 2$ y de 7 segundos con $k = 3$. Esto pudiera ser importante si se requiere de una cierta certeza del tiempo que tomará calcular una solución.

Adicionalmente, en las Tablas 7 a 13, se puede observar que el GANTT-T+ pudo encontrar la mejor solución conocida de un caso de prueba en el conjunto FT (33.3 %) y de 12 casos de prueba en el conjunto LA (33.3 %). Recuerde que el GANTT-T+ es determinista, por lo que cualquier solución óptima calculada, dada una cierta entrada, se puede recalcular si se repite la entrada.

En la Figura 27 se muestra cómo aumenta el tiempo de ejecución conforme aumenta el tamaño del problema. Se grafica el tiempo de CPU medido en segundos contra el tamaño del problema denotado por $n \in \{10, 15, 20, 30, 50\}$ y $m = 10$ fijo. Se grafican los valores, mostrados en las Tablas 9, 10 y 12, de los casos de prueba la16-20, la21-25, la16-30, la31-35 y swv11-15 con $k = 2$. Se puede observar que los tiempos de ejecución con un mismo tamaño de n se encuentran agrupados.

De manera similar, la Figura 28 muestra cómo aumenta el tiempo de ejecución conforme aumenta el tamaño del problema. Se grafica el tiempo de CPU medido en segundos

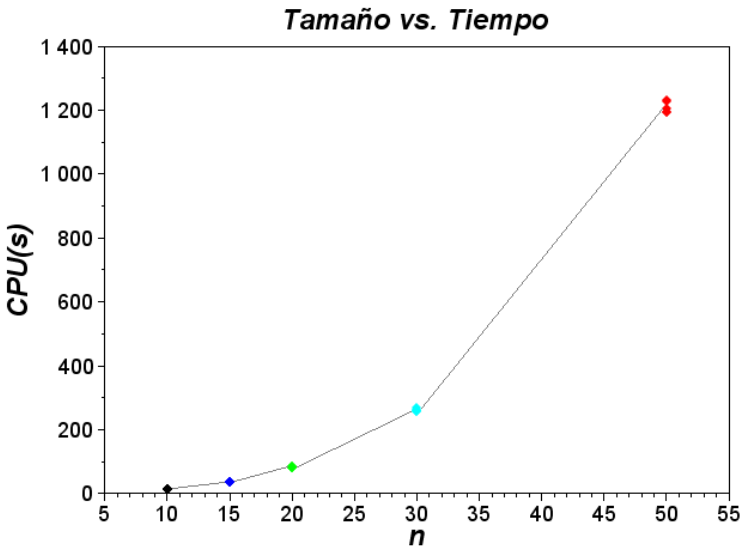


Figura 27: Valor experimental del tiempo de CPU medido en segundos para casos de prueba de distintos tamaños ($m \times n$) con $n \in \{10, 15, 20, 30, 50\}$, $m = 10$ fijo y $k = 2$.

contra el tamaño del problema denotado por $m \in \{5, 10, 15, 20\}$ y $n = 20$ fijo. Se grafican los valores, mostrados en las Tablas 7, 9 y 13, de los casos de prueba la11-15, la26-30, abz07-09 y yn01-04 con $k = 2$. Se puede observar que los tiempos de ejecución con un mismo tamaño de m se encuentran agrupados.

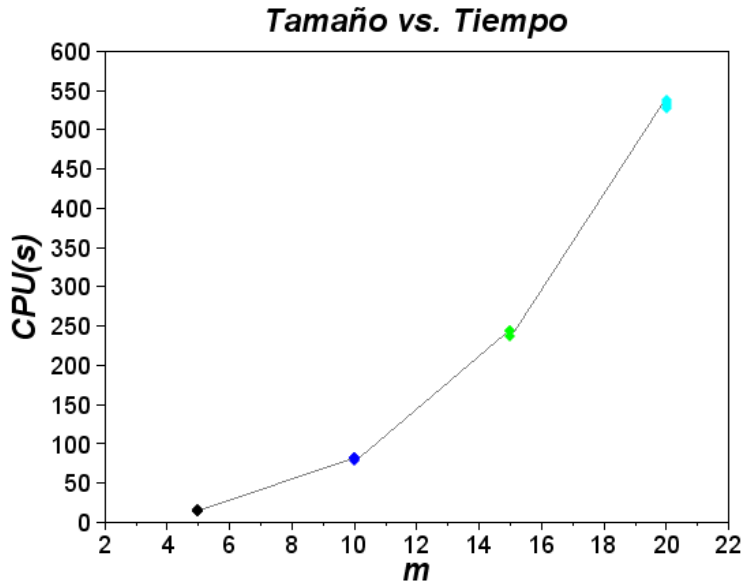


Figura 28: Valor experimental del tiempo de CPU medido en segundos para casos de prueba de distintos tamaños ($m \times n$) con $m \in \{5, 10, 15, 20\}$, $n = 20$ fijo y $k = 2$.

Se puede observar en ambas Figuras 27 y 28, que el crecimiento en tiempo es de orden cuadrático ($k = 2$) tanto en tamaño n como en m.

Tabla 7: Resultados del GANTT-T+ con valores de $k = 2$ y 3, para los casos de prueba del conjunto ABZ.

ABZ									
Caso	$n \times m$	MSC	k	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	ER	$CPU(s)$
abz05	10×10	1234	2	1272	1461	1357	36.92	3.07 %	11.5
			3	1253	1422	1330	32.18	1.53 %	142
abz06	10×10	943	2	960	1117	1030	27.26	1.80 %	11.7
			3	958	1082	1016	24.84	1.59 %	146
abz07	20×15	656	2	749	822	783	15.50	14.17 %	236
			3	736	813	763	12.11	12.19 %	7967
abz08	20×15	645	2	755	848	806	16.65	17.05 %	243
			3	747	814	783	13.29	15.81 %	8011
abz09	20×15	661	2	772	877	830	18.66	16.79 %	244
			3	759	852	804	15.50	14.82 %	8080

Tabla 8: Resultados del GANTT-T+ con valores de $k = 2$ y 3, para los casos de prueba del conjunto FT.

FT									
Caso	$n \times m$	MSC	k	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	ER	$CPU(s)$
ft06	6×6	55	2	55	65	57	2.19	0.0 %	1.4
			3	55	59	57	1.32	0.0 %	2.3
ft10	10×10	930	2	996	1146	1065	30.92	7.09 %	15.8
			3	964	1124	1044	27.65	3.65 %	143
ft20	20×5	1165	2	1230	1413	1314	34.69	5.57 %	15.8
			3	1203	1354	1281	29.65	3.26 %	167

Los experimentos se realizaron en una Mac Pro Intel Xeon Dual Core 2.6 Ghz, memoria RAM de 64GB. Sin embargo, se observó que sólo el $5.54\,\%$ del CPU estaba en uso.

Tabla 9: Resultados del GANTT-T+ con valores de $k = 2$ y 3, para los casos de prueba del conjunto LA.

LA$^{1/2}$									
Caso	$n \times m$	MSC	k	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	ER	$CPU(s)$
la01	10×5	666	2	666	832	721	31.82	0.0 %	2.9
			3	666	794	705	29.77	0.0 %	13.2
la02	10×5	655	2	676	875	774	33.66	3.20 %	2.5
			3	666	816	750	27.17	1.67 %	13.1
la03	10×5	597	2	630	753	680	22.80	5.52 %	2.5
			3	614	726	669	21.40	2.84 %	13
la04	10×5	590	2	614	759	675	26.81	4.06 %	3
			3	609	729	662	26.15	3.22 %	13.2
la05	10×5	593	2	593	671	599	11.78	0.0 %	2.3
			3	593	649	596	8.56	0.0 %	13.4
la06	15×5	926	2	926	1017	942	21.68	0.0 %	7.1
			3	926	1032	938	17.65	0.0 %	17.8
la07	15×5	890	2	890	1014	944	27.41	0.0 %	7.5
			3	890	1023	935	27.92	0.0 %	17.8
la08	15×5	863	2	864	1037	929	27.87	0.11 %	6.9
			3	863	1021	923	29.51	0.0 %	17.5
la09	15×5	951	2	951	1063	980	25.40	0.0 %	6.8
			3	951	1045	977	22.53	0.0 %	17.4
la10	15×5	958	2	958	1044	971	19.22	0.0 %	7.3
			3	958	1021	965	13	0.0 %	17.6
la11	20×5	1222	2	1222	1332	1243	23.58	0.0 %	14.9
			3	1222	1290	1230	13.36	0.0 %	161
la12	20×5	1039	2	1039	1190	1080	26.94	0.0 %	14.4
			3	1039	1127	1058	17.05	0.0 %	161
la13	20×5	1150	2	1150	1279	1181	23.58	0.0 %	15
			3	1150	1257	1164	16.90	0.0 %	159
la14	20×5	1292	2	1292	1345	1293	7.13	0.0 %	14.7
			3	1292	1337	1292	2.91	0.0 %	161
la15	20×5	1207	2	1233	1452	1322	36.78	2.15 %	14.4
			3	1215	1387	1290	31.77	0.66 %	161
la16	10×10	945	2	981	1163	1049	33.15	3.80 %	11.3
			3	963	1113	1027	25.91	1.90 %	140
la17	10×10	784	2	803	944	863	25.83	2.16 %	11.4
			3	797	911	847	24.02	1.68 %	140
la18	10×10	848	2	880	1044	960	30.11	3.77 %	11.2
			3	861	1023	940	28.15	1.53 %	140
la19	10×10	842	2	871	1057	943	29.84	3.44 %	11.2
			3	864	1016	923	27.98	2.61 %	141
la20	10×10	902	2	922	1100	994	31.97	2.21 %	11.2
			3	914	1062	977	29.60	1.33 %	140

Tabla 10: Resultados del GANTT-T+ con valores de $k = 2$ y 3, para los casos de prueba del conjunto LA.

LA$^{2/2}$									
Caso	$n \times m$	MSC	k	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	ER	$CPU(s)$
la21	15×10	1046	2	1165	1350	1242	32.85	11.37 %	35.1
			3	1147	1314	1230	30.45	9.65 %	169
la22	15×10	927	2	1045	1226	1117	34.64	12.72 %	35.6
			3	998	1205	1100	34.67	7.65 %	171
la23	15×10	1032	2	1055	1236	1151	33.15	2.22 %	35.2
			3	1054	1255	1139	31.65	2.13 %	171
la24	15×10	935	2	1024	1222	1121	36.26	9.51 %	35.9
			3	1005	1221	1109	32.36	7.48 %	169
la25	15×10	977	2	1082	1240	1158	30.49	10.74 %	35.7
			3	1067	1229	1150	30.38	9.21 %	168
la26	20×10	1218	2	1324	1548	1431	37.40	8.70 %	82.6
			3	1314	1481	1386	29.28	7.88 %	1911
la27	20×10	1235	2	1382	1606	1464	36.02	11.90 %	82
			3	1339	1506	1425	28.83	8.42 %	1893
la28	20×10	1216	2	1342	1589	1455	41.16	10.36 %	79
			3	1325	1527	1408	36.31	8.96 %	1883
la29	20×10	1157	2	1332	1525	1422	35.56	15.12 %	81
			3	1305	1450	1378	28.19	12.79 %	1918
la30	20×10	1355	2	1425	1630	1515	37.43	5.16 %	80
			3	1400	1564	1475	28.62	3.32 %	1928
la31	30×10	1784	2	1800	2017	1904	40.67	0.89 %	266
			3	1789	1967	1861	30.93	0.28 %	9597
la32	30×10	1850	2	1914	2013	2002	40.53	3.45 %	257
			3	1873	2055	1956	33.84	1.24 %	9429
la33	30×10	1719	2	1748	1966	1845	40.05	1.68 %	257
			3	1719	1944	1800	33.75	0.0 %	9277
la34	30×10	1721	2	1813	2012	1908	38.29	5.34 %	259
			3	1773	1977	1868	34.15	3.02 %	9528
la35	30×10	1888	2	1930	2196	2069	51.45	2.22 %	260
			3	1914	2160	2010	44.49	1.37 %	9569
la36	15×15	1268	2	1379	1601	1474	35.62	8.75 %	103
			3	1348	1559	1463	37.14	6.30 %	691
la37	15×15	1397	2	1500	1697	1595	33.87	7.37 %	111
			3	1507	1647	1578	29.31	7.87 %	686
la38	15×15	1196	2	1337	1525	1422	33.98	11.78 %	105
			3	1334	1499	1411	34.68	11.53 %	684
la39	15×15	1233	2	1361	1574	1461	40.96	10.38 %	106
			3	1351	1576	1441	37.55	9.57 %	700
la40	15×15	1222	2	1336	1520	1414	32.99	9.32 %	104
			3	1343	1482	1402	30.14	9.90 %	704

Tabla 11: Resultados del GANTT-T+ con valores de $k = 2$ y 3, para los casos de prueba del conjunto ORB.

ORB									
Caso	$n \times m$	MSC	k	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	ER	$CPU(s)$
orb01	10×10	1059	2	1125	1307	1217	35.30	6.23 %	11.4
			3	1127	1299	1190	27.88	6.42 %	143
orb02	10×10	888	2	934	1102	1001	31.26	4.84 %	11.4
			3	921	1054	982	25.70	3.71 %	141
orb03	10×10	1005	2	1072	1242	1156	31.55	6.66 %	11.4
			3	1060	1241	1125	28.39	5.47 %	141
orb04	10×10	1005	2	1071	1229	1139	32.92	6.56 %	11.2
			3	1038	1206	1114	30.18	3.28 %	140
orb05	10×10	887	2	919	1072	995	28.67	3.60 %	11.2
			3	916	1065	997	26.23	3.30 %	140
orb06	10×10	1010	2	1069	1294	1164	37.62	5.84 %	11.3
			3	1042	1243	1138	35.46	3.16 %	138
orb07	10×10	397	2	417	491	453	14.24	5.03 %	11.3
			3	414	474	443	12.43	4.28 %	140
orb08	10×10	899	2	940	1091	1011	26.73	4.56 %	11.2
			3	927	1056	988	22.23	3.11 %	139
orb09	10×10	934	2	980	1144	1042	28.51	4.92 %	11.1
			3	958	1109	1022	28.92	2.56 %	139
orb10	10×10	944	2	994	1167	1075	33.08	5.29 %	11.3
			3	965	1133	1049	31.13	2.22 %	143

Tabla 12: Resultados del GANTT-T+ con valores de $k = 2$ y 3, para los casos de prueba del conjunto SWV.

SWV									
Caso	$n \times m$	MSC	k	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	ER	$CPU(s)$
swv01	20×10	1407	2	1605	1820	1696	38.28	14.07 %	80
			3	1556	1730	1647	34.39	10.58 %	1864
swv02	20×10	1475	2	1638	1807	1710	35.46	11.05 %	81
			3	1592	1764	1665	28.54	7.93 %	1884
swv03	20×10	1398	2	1584	1797	1683	38.28	13.30 %	82
			3	1570	1740	1637	31.00	12.30 %	1913
swv04	20×10	1470	2	1675	1871	1756	36.81	13.94 %	82
			3	1617	1786	1703	34.92	10 %	1883
swv05	20×10	1424	2	1627	1829	1719	35.46	14.25 %	79
			3	1590	1749	1665	29.22	11.65 %	1877
swv06	20×15	1678	2	1914	2191	2056	50.91	14.06 %	241
			3	1899	2096	1984	37.10	13.17 %	8009
swv07	20×15	1600	2	1846	2026	1938	34.17	15.37 %	233
			3	1792	1947	1868	30.90	12 %	7988
swv08	20×15	1756	2	2038	2274	2143	42.03	16.05 %	243
			3	1997	2160	2079	32.65	13.72 %	7941
swv09	20×15	1661	2	1927	2115	2019	37.45	16.01 %	243
			3	1858	2034	1952	33.65	11.86 %	7966
swv10	20×15	1754	2	2006	2228	2097	37.62	14.36 %	2431
			3	1954	2114	2027	29.90	11.40 %	7929
swv11	50×10	3005	2	3347	3672	3487	55.36	11.38 %	1193
			3	3316	3597	3431	50.29	10.34 %	15323
swv12	50×10	3038	2	3385	3684	3524	48.79	11.42 %	1226
			3	3364	3604	3476	46.78	10.73 %	15194
swv13	50×10	3146	2	3474	3786	3624	53.29	10.42 %	1192
			3	3416	3750	3577	59.43	8.58 %	15081
swv14	50×10	2968	2	3258	3580	3413	52.51	9.77 %	1232
			3	3253	3519	3370	50.92	9.60 %	14987
swv15	50×10	2940	2	3303	3591	3440	52.58	12.34 %	1206
			3	3297	3557	3399	49.49	12.14 %	15068
swv16	50×10	2924	2	2924	3071	2946	28.70	0.0 %	1202
			3	2924	3029	2933	17.21	0.0 %	15190
swv17	50×10	2794	2	2794	2974	2875	36.97	0.0 %	1202
			3	2794	2972	2854	33.48	0.0 %	15130
swv18	50×10	2852	2	2852	3039	2898	37.69	0.0 %	1187
			3	2852	2979	2878	29.39	0.0 %	14835
swv19	50×10	2843	2	2887	3207	3026	54.45	1.54 %	1204
			3	2880	3116	2988	48.58	1.30 %	14903
swv20	50×10	2823	2	2823	3078	2932	43.21	0.0 %	1173
			3	2829	3077	2909	39.09	0.21 %	15173

Tabla 13: Resultados del GANTT-T+ con valores de $k = 2$ y 3, para los casos de prueba del conjunto YN.

YN									
Caso	$n \times m$	MSC	k	x_{min}	x_{max}	$\lfloor \bar{x} \rfloor$	σ	ER	$CPU(s)$
yn01	20×20	884	2	998	1119	1048	20.75	12.89 %	538
			3	982	1084	1023	16.65	11.08 %	22579
yn02	20×20	907	2	1027	1149	1079	20.33	13.23 %	527
			3	1005	1094	1049	15.68	10.80 %	22984
yn03	20×20	892	2	1021	1125	1066	19.93	14.46 %	547
			3	988	1090	1036	17.74	10.76 %	22498
yn04	20×20	968	2	1090	1269	1177	25.65	12.60 %	523
			3	1094	1201	1145	19.06	13.01 %	22532

5.5. Discusión

Los resultados del Experimento 1 fueron los esperados. Se esperaba que las soluciones producidas por el GANTT-T+, en general, superaran aquellas producidas por los algoritmos GANTT y GANTT-T. Se esperaba también observar que al aumentar el tamaño de la agrupación (k), el tiempo máximo de finalización (T_{max}) disminuiría. Esto se observó, en general, para valores de $k \in \{2, 3, 4\}$. Cuando $k = 5$, se observa un repunte de T_{max}, esto se puede atribuir a que para los casos abz05, ft10 y la16, $k = 5$ es $n/2$ y es probable que ambos calendarios de $n/2$ robots no hayan dejado suficiente espacio de tiempo libre entre segmentos de una misma región para poder combinar ambos calendarios. Debido a que para valores de $k > 3$, k es mayor a $\log n$ para los casos mencionados, debido al tiempo requerido no fue posible probar con valores de k mayores a 5.

Los resultados del Experimento 2 muestran soluciones óptimas para cada caso de prueba considerado, utilizando el GANTT-T+. Los resultados son congruentes con lo enunciado teóricamente en la Proposición 4.5. A pesar del tiempo exponencial requerido, el GANTT-T+ debe ser capaz de encontrar la solución óptima cuando $k = n$.

El Experimento 3 se diseña para probar el desempeño del GANTT-T+ con varios casos de prueba distintos y comparar las soluciones obtenidas con la mejor solución conocida (u óptima) de cada caso. Los resultados experimentales donde la mejor solución conocida es igual a la solución del GANTT-T+ para $k = 2$ o 3, fueron inesperados. A pesar de que no necesariamente se pierde optimalidad al construir calendarios a partir de otros calendarios, no se esperaba encontrar soluciones óptimas para valores de $k \in \{2, 3\}$. Las Figuras 29 y 30 muestran un histograma de frecuencias del error relativo (ER) de los resultados experimentales para valores de k de 2 y 3, respectivamente.

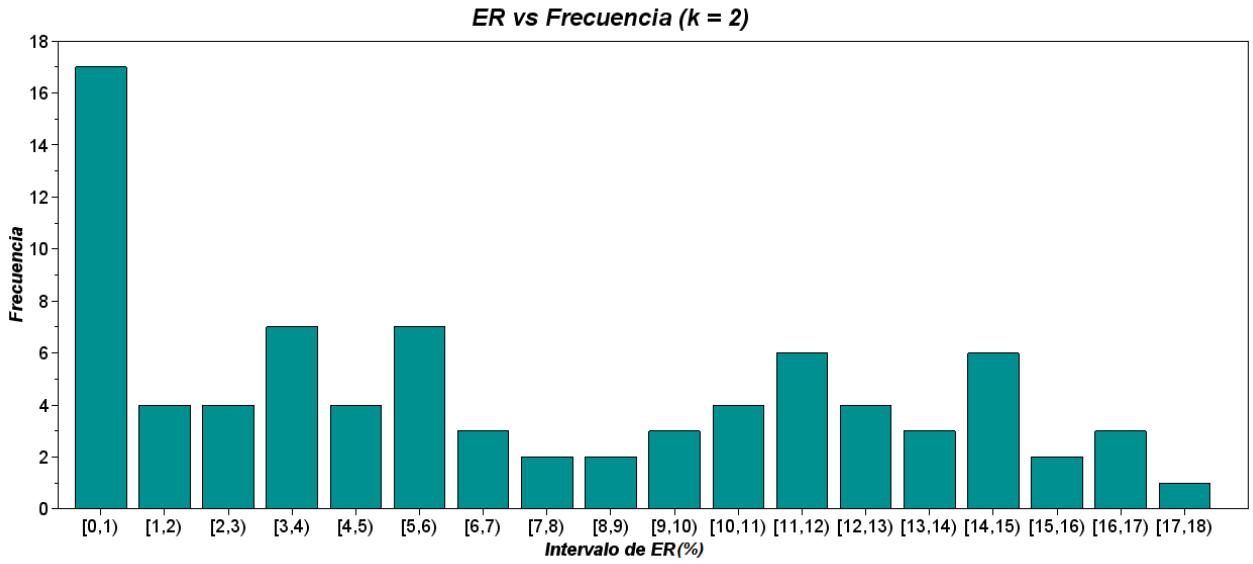


Figura 29: Histograma de frecuencias del error relativo (ER) de los resultados experimentales del GANTT-T+ con los 82 casos de prueba para $k = 2$.

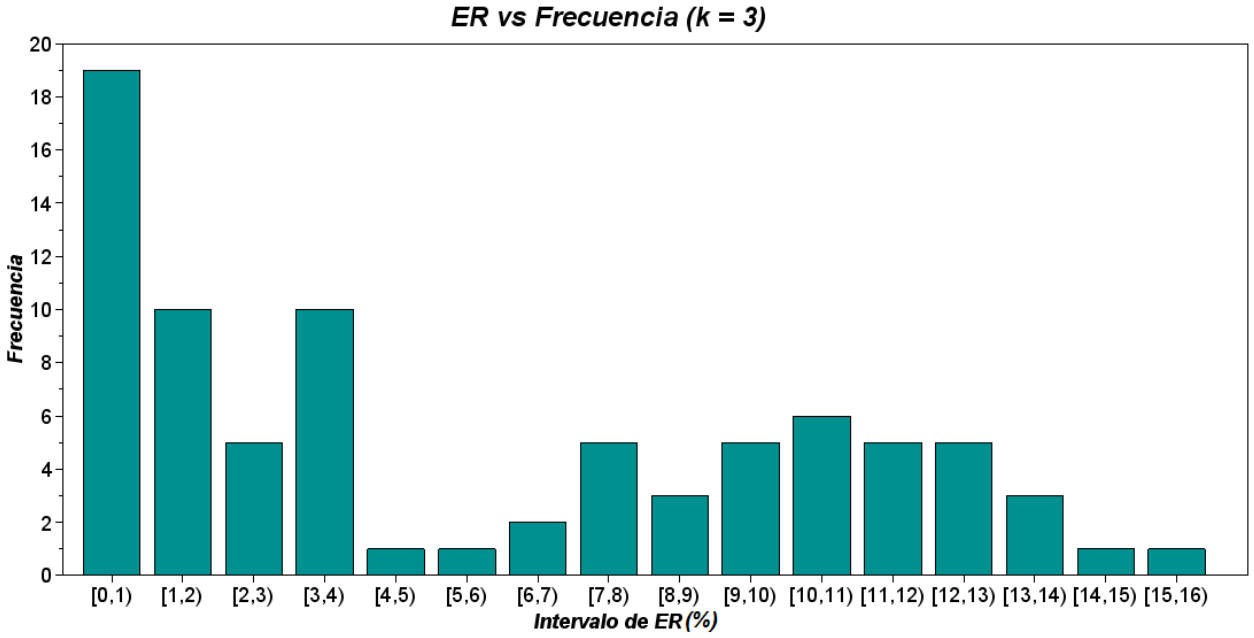


Figura 30: Histograma de frecuencias del error relativo (ER) de los resultados experimentales del GANTT-T+ con los 82 casos de prueba para $k = 3$.

Adicionalmente, las soluciones obtenidas por el GANTT-T+ pueden servir como semi-

lla para heurísticas que utilizan soluciones iniciales como entrada. Como una muestra de ello, se implementó un algoritmo genético cuya población inicial se construye a partir de soluciones generadas por al GANTT-T+. La representación del individuo es una secuencia de segmentos de ruta (calendario de coordinación). Sea I un individuo, la función de aptitud se define como $T_{max}(f^{T+}(I))$. En la Tabla 14 se muestran los parámetros y operadores del algoritmo genético. En 30 ejecuciones del algoritmo genético para los casos de prueba abz05, ft10, y la16 se encontraron los siguientes valores ($ER\,\%$) de solución: 1239 ($0.40\,\%$), 938 ($0.86\,\%$) y 946 ($0.10\,\%$), respectivamente. Estas soluciones corresponden a la mejor solución encontrada para cada caso a lo largo de las 30 ejecuciones. Cabe mencionar que estos resultados son alentadores para experimentar con esquemas que puedan utilizar resultados del GANTT-T+ como semilla.

Tabla 14: Parámetros y operadores del algoritmo genético.

Parámetro	Valor / Operador
Tamaño población	$\mu = 250$
Cruzamiento (prob.)	Cruce uniforme sobre un punto (1)
Mutación (prob.)	Mutación por intercambio con la siguiente casilla (.5 por casilla)
Descendencia	$\lambda = 2$
Selección	Ruleta
Reemplazo	Se quedan los mejores μ de $\mu + \lambda$
No. generaciones	50'000

Capítulo 6. Conclusiones y trabajo a futuro

6.1. Sumario

La coordinación de rutas de robots, propuesto por O'Donnell y Lozano-Pérez (1989), es un enfoque desacoplado que busca resolver parte del problema para la planificación de movimientos. En este enfoque se debe coordinar los movimientos de múltiples robots a lo largo de trayectorias (ruta y velocidad) pre-especificadas. Además, cada ruta se encuentra fragmentada en segmentos de ruta. Se desea que los robots recorran y completen su ruta en el menor tiempo posible. El objetivo es encontrar aquel calendario que especifique el tiempo de inicio de recorrido de trayectoria de cada robot sin que éste colisione con los demás, minimizando el máximo tiempo de finalización entre todos los robots. En un modelo donde la aceleración de los robots no está acotada, la velocidad es constante y donde las áreas de inicio y fin de cada segmento de ruta son libres de colisión, la coordinación de rutas es equivalente al problema ampliamente conocido como *Job Shop Scheduling* (JSP por sus siglas en inglés). De modo que el objetivo se traduce en encontrar aquel calendario de tiempo de inicio de los segmentos de ruta para cada robot tal que el tiempo máximo de finalización del último segmento sea mínimo.

En el presente trabajo de tesis, se explora la semejanza entre ambos problemas y se propone una técnica de agrupamiento de robots basado en diagramas de coordinación *n*-dimensionales para encontrar soluciones para casos del JSP.

Se proponen tres algoritmos determinísticos de calendarización implementando la técnica de agrupamiento de robots, concretamente: el GANTT, el GANTT-T, y el GANTT-T+. Los tres algoritmos realizan un procedimiento iterativo de agrupamiento y calendarización de robots, donde cada uno tiene características particulares. El GANTT busca maximizar el recorrido paralelo de segmentos de ruta que no pertenezcan a una misma región, en un mismo intervalo de tiempo. El GANTT-T implementa un adelgazador que permite calcular el tiempo de inicio de recorrido de un segmento de ruta; que inicie (el recorrido) tan pronto como el segmento que lo precede, en su misma ruta, haya sido recorrido y la región a recorrer haya sido liberada. El GANTT-T+ implementa un adelgazador mejorado en donde se calcula el mínimo tiempo de inicio de un segmento de ruta; que

inicie (el recorrido) tan pronto como sea posible, respetando la restricción de precedencia de los segmentos de ruta. Se demuestra un factor de aproximación de $n/k \cdot T_{opt}$ para el GANTT-T+, lo que lo convierte en un algoritmo de aproximación.

Se diseñan tres experimentos que utilizan casos de prueba del JSP conocidos en la literatura, de los cuales la mejor solución conocida (u óptima) está reportada. El primer experimento se diseña para comparar los algoritmos propuestos entre si. El segundo experimento se diseña para comprobar empíricamente el factor de aproximación, de GANTT-T+, en el caso particular cuando el tamaño de agrupamiento de robots es igual al número de robots; esto es cuando $k = n$. El tercer experimento se diseña para medir el desempeño de GANTT-T+ con cada caso de prueba con distintos tamaños de agrupamiento k de $O(\log n)$, comparando los valores de la solución obtenida con los valores de la mejor solución conocida.

6.2. Conclusiones

Con base en el trabajo de tesis realizado, se concluye lo siguiente:

1. Ambos problemas, la coordinación de rutas y el problema *Job Shop Scheduling*, no sólo son similares sino equivalentes bajo ciertas suposiciones. Específicamente, cuando la aceleración de los robots no está acotada, la velocidad es constante y donde las áreas de inicio y final de cada segmento de ruta están libres de colisión.

2. La coordinación de rutas, construyendo un calendario a través de un diagrama de coordinación *n*-dimensional, requiere de tiempo exponencial; sin embargo, es posible particionar el problema realizando n/k calendarios de n/k diagramas de coordinación *k*-dimensionales en tiempo polinomial. Esto es cierto para alguna constante k. Posteriormente, es posible combinar los n/k calendarios, en tiempo polinomial, y así obtener un calendario que coordine las rutas de todos los robots.

3. El algoritmo GANTT-T+ propuesto mostró ser, experimentalmente, superior a los algoritmos GANTT y GANTT-T (también propuestos en este trabajo) en cuanto a calidad de solución producida.

4. Evidencia teórica y experimental apuntan a que el algoritmo GANTT-T+ se comporta como un algoritmo exacto de tiempo exponencial, cuando $k = n$.

5. Los resultados experimentales muestran que el algoritmo GANTT-T+ tuvo un error relativo (ER) promedio del $5.48\,\%$ con $k = 3$ y del $6.84\,\%$ con $k = 2$, sobre 82 casos de prueba conocidos (Sección 5.1).

6. Los resultados de la literatura para el conjunto de casos de prueba LA (el conjunto de casos más grande) muestran valores de ER promedio de $3.67\,\%$ (Adams *et al.*, 1988), $1.87\,\%$ (Binato *et al.*, 2002), $0.41\,\%$ (Gonçalves *et al.*, 2005) y $0.00\,\%$ (Pardalos y Shylo, 2006), mientras que el algoritmo GANTT-T+ tiene un ER promedio de $3.65\,\%$ para $k = 3$. No obstante, el algoritmo GANTT-T+ recibe como parámetro de entrada el tamaño de agrupamiento k deseado. A pesar del incremento en tiempo de ejecución, de manera general, a mayor tamaño de agrupamiento k menor es el valor ER promedio.

7. Resultados experimentales muestran que para casos de prueba del mismo tamaño ($n \times m$), el algoritmo GANTT-T+ presenta tiempos de ejecución similares. Siendo que, para casos de prueba de tamaño 10×10, la mayor diferencia observada entre cualesquier par de tiempos es menor a un segundo cuando $k = 2$ y de siete segundos cuando $k = 3$. En cuanto al tiempo de ejecución experimental y el tiempo de ejecución teórico del algoritmo GANTT-T+ (Proposición 4.5), se observa congruencia entre ellos.

6.3. Trabajo a futuro

Los posibles temas a estudiar a partir de este trabajo son:

1. Algunas heurísticas para el *Job Shop Scheduling*, utilizan soluciones previamente construidas como parte de los argumentos de entrada. Se sugiere comparar el desempeño de heurísticas que requieran soluciones del problema como argumento, usando el algoritmo GANTT-T+ como “semillero”. Al hacer esto, idealmente, se esperaría una mejora en la calidad de las soluciones y/o una disminución del tiempo de ejecución de dichas heurísticas. En la Sección 5.5 se describe brevemente un

pequeño experimento en el que se utilizan soluciones generadas por el algoritmo GANTT-T+ para inicializar la población de un algoritmo genético.

2. Se tiene evidencia experimental que sugiere que los algoritmos propuestos tienen problemas para construir calendarios de coordinación a partir de ciertos diagramas de coordinación de robots compuestos. Por ejemplo, en cada caso de prueba del Experimento 1 (Capítulo 5), cuando $k = 5$ se puede ver un aumento en el promedio del tiempo máximo de finalización, en comparación con valores de k menores a 5. Se sugiere explorar el uso de alguna política de decisión que permita calendarizar de manera más holgada en etapas intermedias (donde existen varios agrupamientos de robots) y calendarizar de manera más compacta en la última etapa (donde existe un sólo agrupamiento de robots).

3. El presente trabajo, no toma en cuenta una aceleración acotada ni segmentos de ruta donde exista riesgo de colisión en sus puntos de inicio y final. Se sugiere estudiar las implicaciones que conllevan el tomar en cuenta dichas restricciones y estudiar su relación con las restricciones *no-wait*, *blocking swap* y *blocking no-swap* del problema *Job Shop Scheduling*. Posteriormente, realizar una adaptación de los algoritmos propuestos para lidiar con dichas restricciones.

4. Por último, se sugiere hacer un análisis con mayor detalle del peor caso para el algoritmo GANTT-T+, esto para poder proponer un factor de aproximación más ajustado. Si bien el factor de aproximación propuesto es de $(n/k) \cdot T_{opt}$, los resultados experimentales sugieren que el factor de aproximación es logarítmico como lo es en el caso del problema del viajante.

Lista de referencias

Abichandani, P., Benson, H. Y., y Kam, M. (2008). Multi-vehicle path coordination under communication constraints. En: *American Control Conference, 2008*. IEEE, pp. 650–656.

Adams, J., Balas, E., y Zawack, D. (1988). The shifting bottleneck procedure for job shop scheduling. *Management science*, **34**(3): 391–401.

Akella, S. y Hutchinson, S. (2002). Coordinating the motions of multiple robots with specified trajectories. IEEE, Vol. 1, pp. 624–631.

Applegate, D. y Cook, W. (1991). A computational study of the job-shop scheduling problem. *ORSA Journal on computing*, **3**(2): 149–156.

Beasley, J. E. (1990). Or-library: Distributing test problems by electronic mail. *The Journal of the Operational Research Society*, **41**(11): 1069–1072.

Binato, S., Hery, W., Loewenstern, D., y Resende, M. (2002). A grasp for job shop scheduling. En: *Essays and surveys in metaheuristics*. Springer, pp. 59–79.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., *et al.* (2001). *Introduction to algorithms*, Vol. 2. MIT press Cambridge.

Erdmann, M. y Lozano-Pérez, T. (1987). On multiple moving objects. *Algorithmica*, **2**(1-4): 477–521.

Fisher, H. y Thompson, G. L. (1963). Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules. pp. 225–251.

Garey, M. R., Johnson, D. S., y Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, **1**(2): 117–129.

Gonçalves, J. F., de Magalhães Mendes, J. J., y Resende, M. G. (2005). A hybrid genetic algorithm for the job shop scheduling problem. *European journal of operational research*, **167**(1): 77–95.

Hartenberg, R. S. y Denavit, J. (1964). *Kinematic synthesis of linkages*. McGraw-Hill. pp. 68–87.

Howden, W. (1968). The sofa problem. *The Computer Journal*, **11**(3): 299–301.

Latombe, J.-C. (1991). *Robot Motion Planning*. Kluwer Academic Publishers. Norwell, MA, USA.

Lavalle, S. M. y Hutchinson, S. A. (1998). Robots Having Independent Goals. **14**(6): 912–925.

Lawrence, S. (1984). *Supplement to resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques*. GSIA, Carnegie-Mellon University. Pittsburgh, PA, USA.

Leven, D. y Sharir, M. (1987). Planning a purely translational motion for a convex object in two-dimensional space using generalized voronoi diagrams. *Discrete & Computational Geometry*, **2**(1): 9–31.

O'Donnell, P. y Lozano-Pérez, T. (1989). Deadlock-free and collision-free coordination of two robot manipulators. *Proceedings, 1989 International Conference on Robotics and Automation*, pp. 484–489.

Olmi, R., Secchi, C., y Fantuzzi, C. (2008). Coordination of multiple agvs in an industrial application. En: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, pp. 1916–1921.

Pardalos, P. y Shylo, O. (2006). An algorithm for the job shop scheduling problem based on global equilibrium search techniques. *Computational Management Science*, **3**(4): 331–348.

Parker, L. E. (2009). Path Planning and Motion Coordination in Multiple Mobile Robot Teams. pp. 1–24.

Peng, J. y Akella, S. (2005). Coordinating multiple robots with kinodynamic constraints along specified paths. *The International Journal of Robotics Research*, **24**(4): 295–310.

Pinedo, M. L. (2012). *Scheduling: theory, algorithms, and systems*. Springer. pp. 247–251.

Reif, J. y Sharir, M. (1994). Motion planning in the presence of moving obstacles. *Journal of the ACM (JACM)*, **41**(4): 764–790.

Sahni, S. y Cho, Y. (1979). Complexity of scheduling shops with no wait in process. *Mathematics of Operations Research*, **4**(4): 448–457.

Schrijver, A. (1998). *Theory of linear and integer programming*. John Wiley & Sons.

Sharir, M. (1989). Algorithmic motion planning in robotics. *Computer*, **22**(3): 9–19.

Siegwart, R., Nourbakhsh, I. R., y Scaramuzza, D. (2011). *Introduction to autonomous mobile robots*. MIT press. pp. 1–10.

Siméon, T., Laumond, J.-P., y Nissoux, C. (2000). Visibility-based probabilistic roadmaps for motion planning. *Advanced Robotics*, **14**(6): 477–493.

Siméon, T., Leroy, S., y Laumond, J.-p. (2002). Path Coordination for Multiple Mobile Robots : A Resolution-Complete Algorithm. **18**(1): 42–49.

Storer, R. H., Wu, S. D., y Vaccari, R. (1992). New search spaces for sequencing problems with application to job shop scheduling. *Management science*, **38**(10): 1495–1509.

Yamada, T. y Nakano, R. (1992). A genetic algorithm applicable to large-scale job-shop problems. En: *Parallel Problem Solving from Nature 2, PPSN-II, Brussels, Belgium, September 28-30, 1992*. pp. 283–292.

Yamada, T. y Nakano, R. (1997). Job shop scheduling. *IEEE control Engineering series*, pp. 134–134.

Yannakakis, M., Papadimitriou, C., y Kung, H. T. (1979). Locking policies: Safety and freedom from deadlock. En: *Foundations of Computer Science, 1979., 20th Annual Symposium on*. pp. 286–297.