UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Propiedades Topológicas y Algorítmicas de la Gráfica (n, k)-Estrella

T E S I S

QUE PARA OBTENER EL TÍTULO DE:
LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:
EMMANUEL VITE SEVILLA

DIRECTOR DE TESIS:
DRA. MARÍA DE LUZ GASCA SOTO

2015

Ciudad Universitaria, D. F.

1. Datos del Alumno
Vite
Sevilla
Emmanuel
22 27 83 44
Universidad Nacional Autónoma de México
Facultad de Ciencias
Ciencias de la Computación
302012916

2. Datos del tutor
Dra.
María de Luz
Gasca
Soto

3. Datos del Sinodal 1
Dr.
Sergio
Rajsbaum
Gorodezky

4. Datos del Sinodal 2
Mat.
Ilán Abraham
Goldfeder
Ortiz

5. Datos del Sinodal 3
Dr.
José David
Flores
Peñaloza

6. Datos del Sinodal 4
M. en C.
Carlos
Zerón
Martínez

7. Datos del trabajo escrito
Propiedades Topológicas y Algorítmicas de la Gráfica (n, k)-Estrella
123p
2015

Índice general

Índice de figuras

Índice de cuadros

Introducción

Al incrementarse los requerimientos de alto desempeño computacional, el desarrollo de procesadores en paralelo ha ido aumentando en años recientes. Este campo genera investigación abundante en ámplias áreas de las Ciencias de la Computación y Teoría de Gráficas. Sin embargo, a pesar de estos esfuerzos, la sobrecarga de comunicación en los sistemas de cómputo es uno de los elementos más afectados en el desempeño del sistema. A fin de dar solución a este problema, se han propuesto redes de interconexión que ofrecen una solución interesante, y actualmente se están convirtiendo en omnipresentes en los sistemas digitales. En esencia, el problema básico a resolver es simple: dado un conjunto de procesadores (o nodos de una gráfica), definimos un conjunto de enlaces (o aristas de una gráfica) tal que se puedan comunicarse a costo mínimo. Obviamente, tal simplificación del problema no hace justicia a este campo de investigación; sin embargo, su dominio de soluciones ha dado lugar a mucha investigación en las décadas pasadas. De ahí que, dependiendo de la aplicación en cuestión, comunicar puede significar muchas cosas diferentes [39].

Parte integral de cualquier sistema distribuido es el diseño de la red de interconexión que usará ya que determinará significativamente su desempeño. Es conveniente, por tanto, que las siguientes propiedades: simetría, un grado pequeño, un diámetro bajo, tolerancia a fallos, conexidad y los algoritmos de enrutamiento estén presentes, pues hablan de una red de interconexión eficiente.

Por ejemplo: el grado de una gráfica es una medida del costo de la red, el diámetro es una medida de la demora de la comunicación entre los nodos, la simetría por vértices y la simetría por aristas permite minimizar la congestión del envío de mensajes enrutados en la gráfica. De igual manera, las trayectorias y ciclos son fundamentales para el diseño de algoritmos simples con bajo costo de comunicación en cualquier red de interconexión. La tolerancia a fallos junto con un algoritmo simple de enrutamiento generará trayectorias de longitud mínima

entre cualesquiera dos vértices, permitiendo que el desempeño en términos de retardo de comunicación sea óptimo.

Como primer acercamiento planteemos lo siguiente. Dado un conjunto de n procesadores (vértices) en una red de interconexión, podemos conectarlos todos entre sí mediante enlaces (aristas); como resultado tendremos una gráfica completa. Cada procesador podrá enviar y recibir datos directamente, pues dispondrá de $n-1$ vecinos para ello. Aunque parece conveniente, este tipo de red no resulta factible en la práctica ya que resulta ser costosa y poco realista para todos los valores de n. Como consecuencia se buscan topologías que permitan al conjunto de nodos por medio de los enlaces comunicarse a costo mínimo.

La red de interconexión más popular es el hipercubo [38]. En 1987 Akers *et al.* [1] proponen la red de interconexión n-estrella como una alternativa competitiva al hipercubo. Para el año de 1995 Chiang y Chen [15], definieron una generalización de la red n-estrella agregando un nuevo parámetro, k, que controla la cantidad de nodos en la topología. A esta nueva topología la denominaron (n,k)-**estrella**.

Desde su descripción misma, la red de interconexión n-estrella se propone como serio contendiente al hipercubo. No obstante, la red de interconexión (n,k)-estrella tiene ventajas relevantes sobre la n-estrella. Su mayor ventaja, la escalabilidad, la hace más versátil que la red n-estrella, pues cuenta con un parámetro más, a saber k. La mayor dificultad práctica que tiene la gráfica n-estrella es la restricción sobre el número de nodos, pues se requieren las permutaciones de n elementos, por lo cual, al pretender escalar la red entre $n!$ y $(n+1)!$ encontramos que existe una enorme diferencia, y sin embargo, para la gráfica (n,k)-estrella se necesitan las permutaciones de n en k. Dicho de otra manera: uno puede enfrentarse a la elección de cualquiera de las dos redes de interconexión, o con muy pocos nodos disponibles o demasiados nodos disponibles.

Al ser una generalización de la n-estrella, la gráfica (n,k)-estrella hereda algunas de sus propiedades logrando ventajas significativas en ello, así como nuevas propiedades. Aunado a que la red es escalable, también se pueden establecer otras relaciones entre estas dos topologías; por ejemplo: el isomorfismo. Cuando $k=n-1$, la gráfica $(n,n-1)$-estrella es isomorfa a la n-estrella; también para $k=1$ se tiene que la gráfica (n,k)-estrella es isomorfa a la gráfica completa. Otra propiedad importante es la estructura jerárquica de la (n,k)-estrella, ya que permite descomponer la gráfica en subgráficas que serán isomorfas entre ellas; gracias a ello es posible aprovechar esta estructura para el desarrollo de algoritmos de enrutamiento y difusión de la información.

El objetivo del presente material es estudiar la topología de interconexión (n,k)-estrella, profundizando en las propiedades y características descritas an-

teriormente; realizar un análisis comparativo con otras topologías o redes de interconexión; observar cómo se elimina la restricción sobre el número de nodos de la gráfica n-estrella y cómo preserva algunas de las propiedades de la gráfica, tales como: la simetría en nodos, la estructura jerárquica, la tolerancia máxima a fallos y la ruta más corta. En general, mostrar que la red de interconexión (n, k)-estrella posee excelentes propiedades topológicas que la hacen una elección atractiva. La discusión se basará, principalmente, en los artículos escritos por *Wei-Kuo Chiang* y *Rong-Jaye Cheng* titulados *The* (n, k)*-star graph: a generalized star graph* [15] y *Topological properties of the* (n, k)*-star graph* [16].

Examinaremos la red (n, k)-estrella desde dos puntos de vista: la Teoría de Gráficas y los Algoritmos. El presente trabajo está organizado de la siguiente manera:

- En el capítulo 1 exponemos algunos antecedentes sobre la computación paralela, así como un acercamiento breve a las redes de interconexión y conceptos, que serán necesarios en la discusión del presente material.

- El capítulo 2 introduce anotaciones necesarias sobre cómo nuestro modelo de red de interconexión ejecutará sus tareas. Se detallan algunos conceptos sobre el estudio de Redes de Interconexión y se presentan algunas redes tales como: la red completa, el arreglo lineal, la malla, el árbol binario, la pirámide, el entrelazado perfecto, el hipercubo, la n-estrella y la (n, k)-estrella. Finalmente se hace un breve comparativo de estas topologías.

- En el capítulo 3 consideraremos las propiedades topológicas de la red de interconexión (n, k)-estrella. Hablaremos de la estructura jerárquica, el isomorfismo, la simetría y la tolerancia a fallos, la distancia y el diámetro de la gráfica $S_{n,k}$.

- En el capítulo 4 revisaremos algunos algoritmos de comunicación para la gráfica (n, k)-estrella. Estudiaremos el enrutamiento en la red, así como, algoritmos de difusión de la información.

- Después presentaremos las conclusiones del presente trabajo.

- Finalmente se complementa el escrito con dos apéndices. El primero enuncia conceptos básicos sobre Teoría de Gráficas. El segundo presenta conceptos necesarios sobre Álgebra.

Capítulo 1

Antecedentes

Existen buenas razones por las cuales estudiar computación paralela. Pensemos en el siguiente ejemplo: un equipo de cirujanos desea ver en una pantalla especial una imagen reconstruida del cuerpo de un paciente en preparación para cirugía. Deben ser capaces de girar la imagen a voluntad, obtener una vista en sección transversal del órgano y luego observar los detalles nítidamente, para posteriormente realizar en un simulador la cirugía; todo ello sin tocar al paciente. Sin embargo, para poder llevar a cabo esta empresa se necesita como mínimo una velocidad de procesamiento de 10^{15} operaciones por segundo para observar los detalles nítidamente [2].

Este ejemplo es representativo de aplicaciones que necesitan computadoras que puedan procesar grandes cantidades de datos y, si esto no es posible, por lo menos hacerlo dentro de un periodo razonable de tiempo.

En los últimos años ha habido un espectacular incremento en la velocidad de procesamiento de las computadoras lo que ha permitido el aumento en tamaño y alcance de los problemas que podrán resolverse. Desafortunadamente este incremento tiene un límite establecido por las leyes físicas que impiden el incremento en la velocidad de procesamiento, al menos con los materiales actualmente usados [2].

Así como ha aumentado la velocidad de procesamiento, también la demanda de los problemas a resolver. Por esta razón la necesidad de disponer de computadoras más eficientes y rápidas ha provocado que se dé énfasis a la computación paralela, pues ha supuesto un cambio en la forma tradicional en que se realizaban las cosas.

En la parte final de la década de 1940 un grupo de investigadores (*principal-*

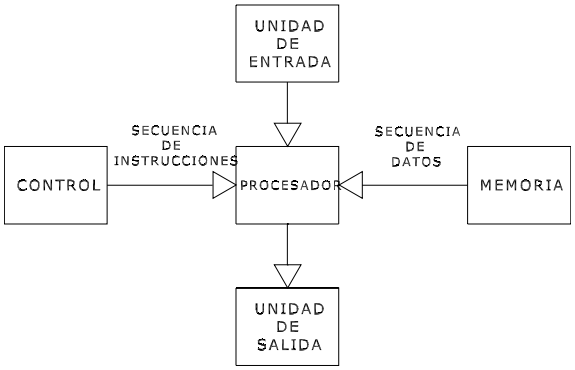


Figura 1.1: Computadora secuencial

mente John Von Neumann) de la Universidad de Princeton propuso un diseño que marcó el comienzo de la era moderna de la computación. Actualmente, la inmensa mayoría de las computadoras en uso lo sigue, más o menos igual al modelo propuesto. Este diseño original de una computadora consiste esencialmente de una unidad de procesamiento, o procesador, que ejecuta una única secuencia de instrucciones en una única secuencia de datos.

Comúnmente, se acepta que este modelo es el modelo idóneo para el análisis y diseño de algoritmos secuenciales. Las operaciones típicas para este modelo incluyen operaciones básicas de aritmética y lógica, así como lectura y escritura en memoria.

La secuencia de instrucciones, antes mencionada, es el programa, que le dice al procesador cómo resolver cierto problema. La secuencia de datos, es un ejemplar del problema a resolver. En cada paso durante el cómputo, la unidad de control emite una instrucción que opera sobre un *datum* (un *datum* puede ser un simple número o, más complejo, varios números) obtenido de la unidad de memoria. Cabe señalar que el procesador cuenta con una pequeña unidad de memoria local constituida de registros de tamaño fijo para un mejor rendimiento computacional. Además, están conectados a unidades de entrada y salida para permitir la comunicación con el mundo real. Este modelo de cómputo es conocido como **computadora secuencial** (serial o convencional). La figura 1.1 ilustra el modelo.

En cambio, el procesamiento paralelo difiere de la forma clásica de hacer cómputo, a saber en la gran variedad del soporte existente. La historia, en esta materia, resalta el progreso vertiginoso en el desarrollo de arquitecturas y software para estas computadoras y su influencia en el desarrollo de aplicaciones modernas.

Desafortunadamente, el cómputo paralelo adolece de un modelo algorítmico

ampliamente aceptado; debido a que el rendimiento de los algoritmos paralelos obedece a factores complejamente interrelacionados que dependen de la máquina en estudio. Entre los factores encontramos: concurrencia computacional, asignación del procesador y la programación (o calendarización) de la comunicación y sincronización [32].

En las últimas décadas, para alcanzar la meta de resolver un problema, se han usado varios procesadores (dos o más), también llamados unidades de procesamiento, en una **computadora paralela**. El objetivo es dividir el problema en subproblemas con el fin de resolverlos simultáneamente por un procesador diferente, y así, que los procesadores se comuniquen unos con otros para poder intercambiar resultados parciales. Después, los resultados se combinan para dar respuesta al problema original. Esto contrasta con el modelo de computadoras secuenciales, las cuales toman el problema a resolver esperando que cada parte del problema se resuelva para poder continuar y dar una solución.

Dicho esto, entenderemos que una *computadora paralela* es una colección de procesadores, de algún tipo, que se interconectan para permitir la coordinación de sus actividades y el intercambio de datos. Se asume que todos los procesadores tienen poca distancia uno del otro y que su función principal es resolver conjuntamente un problema dado.

Esta visión puede extenderse, pues dicho por Jack Dongarra *et al.* en [23], la computación paralela es más que una estrategia para lograr alto desempeño, es la percepción de cómo la computación puede escalar sin problemas de un solo procesador a un poder de cómputo virtualmente ilimitado. Los sistemas paralelos están compuestos de procesadores, memoria jerarquizada y redes de interconexión.

Por desgracia, el escalamiento de una aplicación para mejorar su rendimiento no ha igualado el gran incremento en la velocidad de procesamiento y la carga de programación para estas máquinas sigue siendo pesada. Esto es particularmente problemático porque el enfoque de “escalabilidad sin problemas” no puede lograrse sin tener aplicaciones que escalen automáticamente a la par con el incremento del número de procesadores en la topología en cuestión. Además, para que esto suceda, las aplicaciones tienen que ser programados para que sean capaces de explotar el paralelismo de forma eficaz. Por lo tanto, la responsabilidad de lograr la visión de paralelismo escalable cae en el desarrollador de la aplicación, pues es el responsable de que la aplicación se acople a la arquitectura adecuadamente y que esté diseñada para el aumento de procesadores en su topología.

El desempeño eficiente de una aplicación en un equipo paralelo no depende únicamente de la velocidad de procesamiento, sino de la capacidad del sistema de memoria para alimentar de datos al procesador. En el nivel lógico, un sistema

de memoria, posiblemente consista de múltiples niveles de memorias caché, éstas toman una solicitud de una palabra en memoria y devuelve un bloque de datos de tamaño b que contiene la palabra solicitada después de l nanosegundos. Aquí, a l se le conoce como la *latencia* de la memoria y la velocidad a la que los datos pueden ser surtidos desde la memoria al procesador se le conoce como el *ancho de banda* del sistema de memoria. Ilustremos.

Supongamos que una manguera de bomberos está conectada a una toma de agua. El agua sale dos segundos después al final de la manguera, entonces la latencia del sistema es de dos segundos. Ahora bien, una vez que se inicia el bombeo de agua, el flujo de agua es a razón de un litro por segundo, entonces el ancho de banda del sistema es a razón de un litro sobre segundo. Así que, si tenemos que apagar un incendio de inmediato, podríamos requerir de una latencia menor del sistema y un aumento en la presión sobre el agua. Ahora bien, si requerimos apagar incendios más grandes, es necesario contar con un manguera más ancha y larga. Esta analogía funciona bien para sistemas de memoria. Como vemos aquí, la latencia y el ancho de banda juegan un papel crítico en la determinación del rendimiento del sistema de memoria.

Una de las maneras más fáciles de mejorar el rendimiento de un sistema informático es simplemente replicar computadoras enteras y agregar una forma de comunicación de datos entre equipos independientes.

1.1. Modelos de cómputo paralelo

Con mente inquisitiva, podemos plantearnos las siguientes preguntas: ¿cómo está organizada una computadora paralela? ¿cómo se comunican los procesadores? ¿se ejecutan los mismos o diferentes programas en los procesadores? ¿operan síncrona o asíncronamente? Todas estas preguntas tienen su respuesta en el campo de la computación paralela. Sin embargo, a diferencia de lo que ocurre con las computadoras secuenciales donde parece que las preguntas tienen una única respuesta ya establecida, la computación paralela admite una amplia variedad de posibles diseños y, por ende, respuestas. Más adelante revisaremos algunos modelos de cómputo paralelo.

Primeramente, se tiene que un **modelo de cómputo paralelo** es una abstracción que describe una computadora paralela. Consideremos el siguiente modelo de cómputo paralelo.

- *Ejemplo.* El método más sencillo, desde el punto de vista del hardware, es el modelo de **memoria distribuida**. El enfoque aquí es utilizar diferentes

equipos conectados por una red. El modelo de programación típico consiste en procesos separados en cada equipo que se comunica mediante el envío de mensajes. Ésta es la forma clásica de computación paralela que se remonta a cuando las computadoras eran las personas con las calculadoras y los mensajes eran escritos en pedazos de papel. Los sistemas de memoria distribuida son las computadoras paralelas más comunes, porque son las más fáciles de implementar. ▲

No obstante, pretendiendo dar respuesta a las preguntas planteadas, ignoraremos los detalles irrelevantes de la implementación intentando capturar las características más importantes del modelo en cuestión. Hoy en día, con base a una multitud de modelos, existen varios sistemas de clasificación debido a la gran diversidad de computadoras paralelas. Para representar esta diversidad, y dependiendo en qué nivel estemos del problema, los investigadores han propuesto modelos abstractos que capturan las principales características de los componentes físicos y los mecanismos lógicos para estudiar y analizar el comportamiento de las aplicaciones paralelas. Estos criterios incluyen el número y el tipo de procesadores, la interconexión entre los procesadores, los correspondientes esquemas de comunicación entre los procesadores, el control y la sincronización global y las operaciones de entrada y salida. Existen tres modelos de abstracción que son usualmente distinguidos: los modelos que contemplan la arquitectura, los modelos computacionales y los modelos de programación.

Es necesario contar con un clasificación sencilla para su estudio. Clasificar estos modelos no es fácil, no obstante, una manera práctica de hacerlo es mediante la generalización de dos “grandes familias”: los modelos de memoria compartida y los modelos de redes de interconexión. Mediante los modelos de memoria compartida y redes de interconexión permitimos que los procesadores se comuniquen entre sí. Esto es necesario para solucionar cualquier problema no trivial en una computadora paralela.

Antes de abordar estos modelos, introduciremos una clasificación general de sistemas computacionales.

1.1.1. Clasificación de arquitecturas informáticas

Introducimos una clasificación básica de arquitecturas informáticas [29]. Las cuatro clasificaciones se basan sobre el número de instrucciones concurrentes (o controles) y los flujos de datos disponibles en la arquitectura, visto por el procesador durante la ejecución del programa. Dependiendo de si hay uno o varios de

estos flujos, las computadoras pueden dividirse en cuatro clases:

- **Instrucción Individual, Secuencia Individual de Datos (SISD[1]).**
  La computadora SISD es la generalización de una máquina secuencial. En esta clase de computadoras no existe el paralelismo, ni en las instrucciones ni en el flujo de datos.

- **Instrucción Múltiple, Secuencia Individual de Datos (MISD[2]).**
  La computadora MISD contiene n procesadores con su propia unidad de control y una unidad de memoria común compartida por todos ellos. El paralelismo se consigue mediante varias unidades funcionales que ejecutan diferentes operaciones sobre los mismos datos. Sin embargo, en la práctica, hasta ahora no se conoce la implementación de esta clase de computadora.

- **Instrucción Individual, Flujo Múltiple de Datos (SIMD[3]).**
  La computadora SIMD contiene n procesadores idénticos con su propia memoria local para almacenar datos. Una unidad central controla el flujo único de instrucciones para el funcionamiento de todos los procesadores. Al funcionar sincrónicamente todos los procesadores ejecutan la misma instrucción en cada paso en un elemento diferente de los datos. Las computadoras SIMD son mucho más versátiles que las computadoras MISD.

- **Instrucción Múltiple, Flujo Múltiple de Datos (MIMD[4]).**
  Esta computadora contiene múltiples procesadores autónomos que ejecutan simultáneamente diferentes instrucciones sobre diferentes datos; cada procesador tiene su propia unidad de control y memoria local. Por lo tanto, todos los procesadores están potencialmente ejecutando diferentes programas en datos diferentes, mientras resuelven subproblemas de un solo problema. Esto hace que las computadoras MIMD sean más potentes que las otras tres clases de computadoras.

Sin embargo, hemos de hacer la siguiente observación: las arquitecturas paralelas y los modelos de programación no son independientes uno del otro. Aunque la mayoría de las arquitecturas paralelas pueden soportar los principales paradigmas de programación, puede que no sean capaces de hacerlo con el desempeño

[1] Siglas en inglés de *Single Instruction, Single Data Stream (SISD)*.
[2] Siglas en inglés de *Multiple Instruction, Single Data Stream (MISD)*.
[3] Siglas en inglés de *Single Instruction, Multiple Data Stream (SIMD)*.
[4] Siglas en inglés de *Multiple Instruction, Multiple Data Stream (MIMD)*.

suficiente para ser rentables. Es por ello necesario resaltar que una parte importante de cualquier arquitectura paralela es la característica de simplificar los procesos de construcción (incluyendo la compilación), las pruebas y el ajuste de la aplicación. De este modo, encontramos que algunas arquitecturas paralelas se diseñan pensando en un modelo de programación paralela particular; mientras que otras ofrecen poco o ningún soporte al modelo de programación paralela. Además, hemos de agregar que algunos modelos de cómputo paralelo pueden ser usados para predecir el desempeño de un algoritmo paralelo, teniendo en cuenta la topología subyacente de la red de interconexión.

1.2. Memoria compartida

Pensemos en la siguiente analogía: un tablero de anuncios. La gente, utiliza el tablero para recibir comunicados, para compartir anuncios y para colocar información. De la misma manera, el modelo de **memoria compartida** permite a todos lo procesadores recibir datos, compartir información y colocar los resultados de sus cómputos en la memoria global: un área en la cual todos lo procesadores tienen acceso, la comparten. Sin embargo, es necesario aclarar que los procesadores del sistema trabajan todos sincrónicamente y ejecutan la misma secuencia de instrucciones sobre diferentes datos y, para que los procesadores se comuniquen entre sí, es necesario que hagan uso de la memoria. Este modelo ofrece una infraestructura atrayente para el desarrollo de técnicas algorítmicas en el cómputo paralelo.

Éste es un enfoque más complejo que une estrechamente a todos los procesadores mediante la colocación de toda la memoria en un solo espacio físico de direcciones y con soporte para direcciones virtuales. Dicho de otra forma, los datos estarán disponibles para todos los procesadores a través de la carga y almacenamiento de instrucciones que soporte la arquitectura en cuestión.

Como ya se mencionó, los procesadores operan sincrónicamente y ejecutan la misma secuencia de instrucciones sobre diferentes datos. Cada paso de cómputo consiste de las siguientes tres fases: la fase de lectura, en la cual cada procesador copia un *datum* de una localidad de memoria en sus registros. Una fase de cómputo, en la cual cada procesador, si se requiere, ejecuta una operación aritmética o lógica sobre un dato almacenado en sus registros. La fase de escritura, en la cual cada procesador, si lo requiere, copia un *datum* de sus registros y lo pone en una localidad de la memoria compartida. Los procesadores comparten un reloj común, pero pueden ejecutar diferentes instrucciones en cada ciclo [27].

La clase de computadoras paralelas que funcionan bajo este esquema son co-

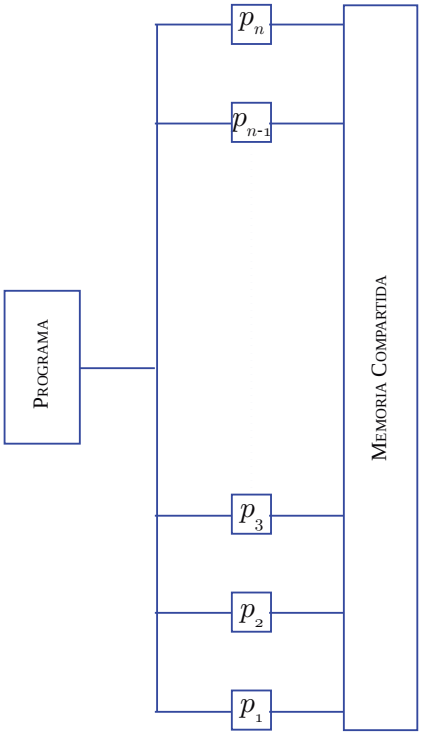


Figura 1.2: Memoria compartida

nocidas también como: **Máquinas Paralelas de Acceso Aleatorio (PRAM[5])**. Como se muestra en la figura 1.2 posee n procesadores idénticos $p_1, p_2, \ldots, p_n$ y una memoria compartida de acceso común para todos estos procesadores.

Ahora bien, dado que no existe un modelo universal que represente la gran diversidad de los sistemas paralelos, se conoce que el modelo de referencia fundamental es el modelo PRAM diseñado para computadoras paralelas bajo el esquema de memoria compartida [27].

Gran parte del trabajo teórico en complejidad de la computación paralela se ha dado en el estudio del modelo de memoria de acceso aleatorio paralelo. Diferentes versiones de PRAM varían en los detalles de cómo se accede a la memoria mediante hilos de ejecución o procesos que acceden a la misma localidad de memoria. Si bien este modelo es de valiosa ayuda en la comprensión de los límites de los algoritmos paralelos, el modelo PRAM representa una abstracción que no puede ser implementada de manera eficiente en la práctica.

[5]Siglas de *Parallel Random Access Machine*.

1.2.1. Clasificación de equipos PRAM

En teoría, todos los procesadores tienen el mismo tiempo para acceder, leer y escribir en memoria. Gracias al conjunto de instrucciones que puede ejecutar una computadora PRAM, se consigue acceso simultáneo de los procesadores a una misma localidad de memoria compartida. En función de cómo acceden a la misma localidad de memoria, hay cuatro diferentes formas de clasificar los equipos PRAM.

- **Lectura Exclusiva, Escritura Exclusiva (EREW[6]).**
  En este modelo, sólo un procesador puede leer desde cualquier posición de la memoria a la vez y sólo un procesador puede escribir en cualquier posición de la memoria a la vez. En otras palabras, cuando se ejecuta esta instrucción, n procesadores pueden leer simultáneamente o escribir en n distintas direcciones de memoria. En esta clase de equipos, el acceso a una ubicación de memoria es exclusiva. No se permiten lectura o escritura concurrente. De una PRAM, es el modelo más débil; dando mínima concurrencia de acceso a la memoria.

- **Lectura Exclusiva, Escritura Concurrente (ERCW[7]).**
  Este modelo tiene la capacidad de que sólo un procesador puede leer de una celda de memoria, pero varios procesadores pueden escribir en una celda de memoria a la vez. Sin embargo, en la práctica, no se considera este tipo de acceso a la memoria.

- **Lectura Simultánea, Escritura Exclusiva (CREW[8]).**
  En este modelo, múltiples procesadores puede leer una celda de memoria, pero sólo uno puede escribir a la vez.

- **Lectura Simultánea, Escritura Simultánea (CRCW[9]).**
  Las computadoras PRAM bajo el esquema CRCW permiten a múltiples procesadores leer o escribir en la misma posición de memoria a la vez. Este es el modelo más potente de una PRAM. Permitir el acceso de lectura simultáneo no crea discrepancias semánticas en el programa. Sin embargo, el

[6] Siglas en inglés de *Exclusive Read, Exclusive Write (EREW)*
[7] Siglas en inglés de *Exclusive Read, Concurrent Write (ERCW)*
[8] Siglas en inglés de *Concurred Read, Exclusive Write (CREW)*
[9] Siglas en inglés de *Concurred Read, Concurrent Write (CRCW)*

acceso a escritura concurrente en una ubicación de memoria requiere arbitraje. Cuando se produce la instrucción *CW*, una pregunta aparece: ¿Qué sucede cuando varios procesadores intentan escribir contenidos diferentes en la misma celda de memoria? A fin de resolver este conflicto regulamos la escritura concurrente.

- **Prioridad *CW***: todos los procesadores están organizados en una lista jerarquizada de prioridades predefinidas y únicamente el procesador con la prioridad más alta tiene éxito al permitírsele escribir en la celda de memoria, el resto fallan.
- **Común *CW***: sólo se permite si los procesadores están tratando de escribir el mismo valor; de lo contrario, es una operación ilegal.
- **Aleatorio *CW***: el procesador que escribe su valor es elegido por un proceso aleatorio.
- **Arbitraria *CW***: se permite que un procesador arbitrario ejecute la operación de escritura, el resto fallan.

1.3. Redes de interconexión

Los modelos de cómputo paralelo basados en memoria compartida poseen una región física de memoria donde se lleva a cabo la comunicación entre los procesadores. No obstante, en las **redes de interconexión** cada procesador contiene su propia memoria y se conecta con otros procesadores vía enlaces directos. Estos enlaces permiten la comunicación bidireccional, por lo que, conectados dos procesadores por un enlace pueden intercambiar datos simultáneamente. El modelo consiste de n procesadores que forman un red conectada, sus procesadores se conectan por enlaces bidireccionales. La comunicación entre procesadores se efectúa mediante el intercambio de mensajes entre los procesadores y todos los procesadores efectúan la misma secuencia de instrucciones sobre diferentes datos.

A diferencia de los modelos basados en memoria compartida, el modelo basado en **redes de interconexión** captura en su diseño la comunicación al incorporar la interconexión entre los procesadores en la topología misma, entendiendo por ello que se dispone de una red para interconectar, es decir, mantener la conexión recíproca entre los nodos de la red. La interconexión es la conexión física y lógica de los elementos de la red, es una comunicación recíproca (de igual correspondencia) efectuada entre dos a más elementos de la red, sea ésta fija o temporal.

Según Dally y Towles [19], una red de interconexión es un sistema programable que transporta datos entre terminales.

Durante mucho tiempo la comprensión de una topología que permitiera alto rendimiento fue importante para los programadores y desarrolladores de algoritmos. Esta situación se reflejó tanto en la literatura como en los modelos de programación paralela. Recientemente, las redes han mejorado al punto que para muchos usuarios la topología de la red ya no es un factor importante en el rendimiento. Este aparente desinterés se da por la uniformidad en la topología proveniente del aumento de ancho de banda dentro de la red. Sin embargo, para beneficiarnos de una topología más rápida, el cómo los procesadores se intercomunican se convierte en una factor todavía más importante al diseñar algoritmos y modelos de programación. La congestión en la red todavía puede ser un problema si el rendimiento de la red no se escala con el número de nodos para el procesamiento.

Para hacer frente a este problema, algunos sistemas de memoria compartida han optado por utilizar redes que conectan cada procesador con cada sistema de memoria. Para números pequeños de procesadores y memorias, una conexión directa entre cada procesador y la memoria es posible (se requieren p^2 conexiones para p dispositivos). Para un mayor número de procesadores se puede utilizar una red menos compleja.

En el siguiente capítulo se ahondará sobre las redes de interconexión.

1.4. Algoritmos paralelos

Los sistemas de cómputo paralelo necesitan de algoritmos para llevar a cabo sus tareas. Entendemos que un **algoritmo paralelo** es un método para resolver un problema computacional en un modelo de computación paralela. Esto permite que el problema se rompa en partes pequeñas para resolverse simultáneamente. Existen varios criterios para evaluar la calidad de un algoritmo paralelo. Nos enfocamos específicamente en el tiempo de ejecución, el número de procesadores y el costo [7].

1.4.1. Análisis de un algoritmo paralelo

Se define una **unidad de tiempo** como el tiempo requerido por un procesador para: ejecutar una operación aritmética, ejecutar una operación lógica, obtener acceso a la memoria, leer o escribir un *datum* o enviar un dato a su vecino.

Tiempo de ejecución. Mediremos el tiempo de ejecución de un algoritmo paralelo contando el número de unidades de tiempo que le toma desde el momento que empieza la ejecución del algoritmo hasta que termina. Hemos de notar que un algoritmo paralelo tiene varias operaciones básicas que podrán estar ejecutándose en una misma unidad de tiempo. El número de operaciones puede ser igual al número de procesadores activos durante la misma unidad de tiempo en ejecución. Usaremos $t(n)$ para expresar el tiempo de ejecución que le toma a un algoritmo paralelo resolver un problema de tamaño n.

Una buena apreciación de la velocidad de un algoritmo paralelo es la comparación de su solución con la mejor solución secuencial posible. La aceleración[10] de un algoritmo paralelo que resuelve un problema computacional es igual al cociente de dos tiempos de ejecución, la mejor solución secuencial posible dividida por la solución del algoritmo paralelo. Cuanto mayor es el cociente, mejor es el algoritmo paralelo.

Número de procesadores. Prácticamente por razones económicas una importante consideración es el número de procesadores requerido por un algoritmo paralelo. Suponga que dos algoritmos diseñados para resolver un mismo problema en un mismo modelo de cómputo tienen idénticos tiempos de ejecución. Si uno de los algoritmos necesita menos procesadores para resolver el problema, éste es menos caro para ejecutarlo y el preferido. Se expresa usualmente el número de procesadores como una función del tamaño del problema a resolver. Para un problema de tamaño n, el número de procesadores está dado por $p(n)$.

Costo. El costo de un algoritmo secuencial se define como un límite superior sobre el número total de operaciones ejecutadas colectivamente por los procesadores. Si un algoritmo resuelve un problema de tamaño n en $t(n)$ unidades de tiempo con $p(n)$ procesadores, entonces el costo está dado por $c(n) = p(n) \times t(n)$. El costo es de utilidad en la evaluación de lo caro que es la ejecución de un algoritmo paralelo. Más importante aún, es una medida frecuentemente usada con propósitos de comparación con respecto a una solución secuencial. La *eficiencia* de un algoritmo paralelo para un problema dado es el coeficiente dado por el tiempo de ejecución de la mejor solución secuencial posible del problema dividido entre el costo del algoritmo paralelo. Cuanto mayor es la eficiencia, mejor es el algoritmo paralelo.

[10] Se usará aceleración como traducción del idioma inglés de *speedup*.

Difusión de la información

El intercambio de datos, entre procesadores, puede tener un impacto significativo en la eficiencia de los programas paralelos, pues introduce el retardo de la interacción durante su ejecución. Los algoritmos paralelos a menudo exigen acciones de comunicación coordinadas que implican múltiples procesos. Estas operaciones globales pueden ser implementadas por un programador. Sin embargo, para mayor comodidad y para permitir implementaciones óptimas, existe un conjunto de funciones colectivas especializadas en la comunicación, es decir, operaciones de uso común que permiten acciones de comunicación coordinadas [23]. Estas funciones son las siguientes:

- *Barrier*. Sincronizar todos los procesos.
- *Difusión de la información*. Enviar datos desde un proceso a todos los procesos.
- *Gather*. Recopilar datos de todos los procesos en un solo proceso.
- *Operaciones de reducción*. Suma, multiplicación, y así sucesivamente de datos distribuidos.

Todas estas operaciones se ejecutan de manera colectiva.

Mediante un solo proceso hay que enviar datos a todos los procesos o a un subconjunto de ellos. Esta operación se conoce como **difusión de la información**[11]. Inicialmente, el proceso que tiene los datos es el único que necesita difundir información. Al terminar el procedimiento, existen varias copias del dato.

Este modelo de difusión de la información consiste de n procesadores con m localidades de memoria compartida y una unidad de acceso a la memoria. Esta instrucción permite a todos los procesadores escribir en todas las localidades de la memoria compartida simultáneamente. Es decir, en la que cada procesador p_i difunde un *datum* d_1 y una etiqueta q_1, con $1 \leq i \leq n$, con destino a todas las m localidades de memoria.

[11] Usaremos difusión de la información como traducción de *broadcasting*.

Capítulo 2

Redes de interconexión

Un **modelo** es física, lógica o matemáticamente una representación de una entidad del mundo real. Su propósito es simular cierto fenómeno por lo que permite un estudio sistemático del mismo. Los modelos pueden ser usados para describir ciertos sistemas conceptuales. Esta capacidad provee a los modelos una concreta manifestación de los procesos que está destinado a representar [2].

En ciencias de la computación, los modelos de cómputo sirven para ambos propósitos. Primero, se utilizan para describir entidades reales, llamadas computadoras. Como tal, un modelo de computación es una versión simplificada de una ya existente o una contemplada pieza de maquinaria. Con tales intentos logramos capturar las características esenciales de la máquina e ignoramos los detalles de la implementación. El modelo, por lo tanto, da una descripción abstracta que es simple de entender teóricamente y de fácil manipulación matemática. Segundo, los modelos de cómputo son usados como herramientas para pensar en el problema y expresar algoritmos. Por ello, la principal razón de ser de los modelos es ayudarnos a entender la computación. Algunos ejemplos de modelos tempranos de computación son los autómatas, la máquina de Turing, las gramáticas formales y las funciones recursivas. Modelos más recientes son: máquina de acceso aleatorio, máquina paralela de acceso aleatorio, redes de interconexión, etcétera.

En el presente trabajo nos centraremos en las redes de interconexión, pues permitirá proporcionar un marco teórico apropiado para el estudio de la red de interconexión (n, k)-estrella. Algunos ejemplos de redes de interconexión son: los sistemas informáticos, las redes de computadoras, los sistemas de comunicación y los sistemas de transportación. En el caso de los sistemas informáticos, los componentes podrán ser los procesadores, las unidades de almacenamiento y los dispositivos de I/O. El objetivo de un sistema informático es básicamente trans-

formar un conjunto de información de entrada en un conjunto de información de salida, visto éste como resultado.

2.1. Conceptos básicos

En la computación paralela una de las maneras de comunicarse entre los procesadores es mediante una red de interconexión; como ya se mencionó en el capítulo anterior, entendemos que se dispone de una red para interconectar, es decir, mantener la conexión recíproca entre los nodos de la red. Por lo cual, se tiene que la interconexión es la conexión física y lógica de los elementos de la red, es una comunicación recíproca (de igual correspondencia de uno a otro) efectuada entre dos a más elementos de la red sea esta fija o temporal. Según Dally y Towles [19], una red de interconexión es un sistema programable que transporta datos entre terminales.

Uno de los criterios más importante para la clasificación de las redes de interconexión se basa en la rigidez de los enlaces entre los nodos, siendo estos enlaces estáticos o dinámicos. Una red estática se caracteriza porque su topología queda establecida de forma definitiva cuando se instala el sistema, su única posibilidad de expansión es crecer. En contraste, una red dinámica puede variar de topología bien durante el curso de la ejecución de los procesos o bien entre la ejecución de los mismos. Por otra parte, las redes pueden ser jerárquicas o no; lo son si están formadas de una serie de niveles, con diferente número de nodos que además son simétricos en su tratamiento [4]. En la figura 2.1 observamos un esquema general de una red de interconexión paralela y estática.

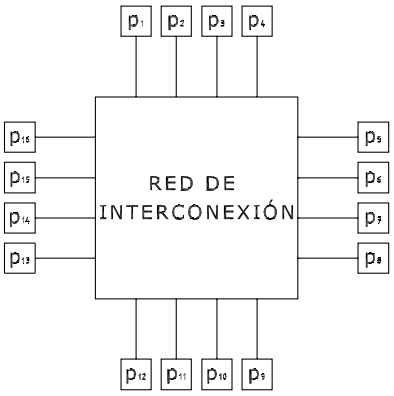


Figura 2.1: Red de interconexión paralela

Así, tomamos en cuenta los principales problemas encontrados al emplear este modelo de computación: el diseño, el análisis, su uso y la rentabilidad que conlleva hacer uso de una red de interconexión [39]. Consideramos, a propósito, hacernos algunas preguntas sobre este modelo de computación:

- ¿Qué forma deberá tener la red?
- ¿Puede un procesador comunicarse inmediatamente con todos sus vecinos?
- ¿Cuál es el tamaño de un mensaje que un procesador puede transmitir en un momento?
- ¿Cuánto tiempo le toma a un procesador inicializar la transmisión?
- ¿Cuánto tiempo le toma a un mensaje viajar entre dos procesadores vecinos?
- ¿Cuánto le toma a un procesador recibir un mensaje?
- ¿Cuánto le toma a un procesador escribir o leer un mensaje en su memoria local?
- ¿Cuánto le toma a un mensaje viajar del procesador p_i al procesador p_j?
- ¿Las rutas son dinámicas o estáticas?
- ¿Los procesadores operan síncrona o asíncronamente?
- ¿Qué tipo de procesador es usado por una red de interconexión?

Todas estas preguntas son apropiadas al diseñar la red de interconexión y tienen diversas respuestas. Algunas, sin embargo, son más importantes que otras y sirven para distinguir plenamente los diferentes modelos empleados en la interconexión.

A continuación listamos algunos conceptos y consideraciones importantes ligados con las redes de interconexión.

Número constante de vecinos. Podemos decir que habrá un número constante de vecinos para la comunicación o que estará en función de n, el número total de procesadores. Se asume que un procesador puede enviar o recibir datos de un número constante de vecinos en una misma unidad de tiempo.

Tamaño constante del mensaje para la transmisión. Cada mensaje que un procesador desea transmitir es considerado como un dato en memoria de tamaño fijo.

Si m datos serán enviados de un procesador a otro, entonces se requerirán de m transmisiones.

Tiempo constante para inicializar la comunicación. Supongamos que un procesador tiene x vecinos (donde x es constante o está en función de n). Con el fin de seleccionar uno de sus vecinos para la transmisión el procesador necesita un tiempo, el cual es una función de x.

Tiempo constante por enlace. Si el enlace que conecta a dos procesadores tiene longitud l, entonces el tiempo para atravesar el enlace es una función de l. Por simplicidad, supondremos que un dato puede viajar de cualquier procesador a otro en tiempo constante.

Tiempo constante para acceder a memoria. Si un procesador es la fuente (o el destino) de un dato que debe leer (o escribir) en la memoria; este paso tomará tiempo, se asume que éste será constante.

Almacenamiento y retransmisión de comunicación. Cuando un procesador p_i desea enviar un dato d al procesador p_j, al que no está conectado, se usa el siguiente esquema: primero p_i envía el dato a uno de sus vecinos, p_k, ahora p_k recibe y almacena d y posteriormente retransmite a uno de sus vecinos, p_l, este proceso continua hasta que se alcanza p_j; donde cada dato d es de tamaño constante y no existe diferencia si este esquema es usado en otro esquema en el cual la ruta ha sido previamente establecida.

Comunicación estática. A menos que se indique lo contrario, se asume que las rutas serán predefinidas. Es decir, la dirección del procesador destino es usada para encontrar la ruta más corta desde el procesador fuente. Por lo cual, es necesario diseñar un algoritmo. El algoritmo define lo que requiere la ruta cuando un procesador p_i desea enviar un dato al procesador p_j en cualquier paso dado. se asumirá que todos los procesadores conocen el número total de procesadores, así como la topología de la red. Además, ningún procesador se aísla, por lo que siempre habrá una ruta entre el procesador origen y el procesador destino.

Operación sincrónica. Asumiremos que cualquier procesador trabajará sincrónicamente. En un paso, que requerirá tiempo constante, un procesador puede recibir datos de un número constante de vecinos, realizar cálculos y enviar datos a un número constante de vecinos. Cada procesador mantiene una copia del algoritmo y todos los procesadores ejecutan este algoritmo en modo *lockstep*[1]. El algoritmo puede indicar que sólo un subconjunto está activo, usando los procesadores indicados. Los procesadores ejecutan el mismo paso al mismo tiempo.

Procesadores. Cada procesador en la red puede ser visto como una RAM; ésta

[1] *Lockstep* es una forma de estar sincrónizados.

puede ejecutar un número básico de operaciones ariméticas y lógicas y tener acceso a una memoria de acceso local. Cada operación requiere un tiempo constante para ser ejecutada. Sin embargo, cada procesador tendrá un número especial de registros llamados puertos, que permiten la comunicación con sus vecinos. Cada puerto está físicamente conectado por un enlace a un puerto de otro procesador.

En una red de interconexión, no hay memoria compartida; en su lugar, cada procesador cuenta con su propia unidad de memoria local y se conecta con otros procesadores mediante enlaces directos entre ellos. Los enlaces son bidireccionales; es decir, dos procesadores conectados por un enlace pueden intercambiar datos de forma simultánea. Por lo cual, la estructura matemática adecuada y conveniente es una gráfica no dirigida $G = (V, A)$ y es empleada para describir una red de interconexión, donde cada procesador p_i es un vértice en V y si hay un enlace entre dos procesadores p_i y p_j en la red de interconexión, entonces la arista (p_i, p_j) existe entre los correspondientes vértices A de la gráfica. En esta tesis usaremos los términos red de interconexión y gráfica de forma indistinta.

2.2. Topologías básicas

Introduciremos algunas redes de interconexión conocidas. Supondremos que todas las redes tienen n procesadores o nodos, centrando nuestra atención en las redes de interconexión con enlaces estáticos.

Primeramente se tiene que el grado de una red de interconexión es el grado máximo de todos los procesadores que hay en la red, (véase la definición A.3).

2.2.1. Red completa

La red de interconexión más obvia y general es la gráfica donde cada nodo en ella está conectado directamente con todos los otros $n-1$ nodos de la gráfica. Esta red de interconexión es conocida como **gráfica completa** o K_n. Ésta es la red más poderosa por su amplia versatilidad; el grado de K_n es $n-1$ y el diámetro es 1. Como se puede observar hay $n(n-1)/2$ enlaces en la red, lo que la hace inviable en la práctica por dos razones: el costo asociado con el número total de aristas y el límite físico de conexiones que puede tener el procesador.

Afortunadamente, un pequeño subconjunto de pares conectados suele ser suficiente para obtener algoritmos eficientes en la mayoría de aplicaciones. En algunos modelos el número de procesadores vecinos es constante, mientras que en

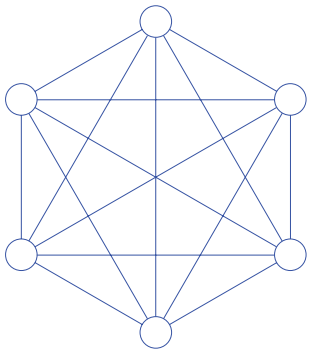


Figura 2.2: Red completa con 6 Procesadores K_6

otros es una función de n, donde n es el número de procesadores. La figura 2.2 ilustra la gráfica completa K_6.

2.2.2. Arreglo lineal

El **arreglo lineal** es la más sencilla y fundamental red de interconexión. En ésta, los n procesadores forman un arreglo unidimensional. Cada procesador p_i, con $1 < i < n$, está conectado con dos vecinos llamados p_{i-1} y p_{i+1}, exceptuando los procesadores p_1 y p_n, estos sólo tienen un vecino. El grado máximo del arreglo lineal es 2 y el diámetro es $O(n)$. La figura 2.3 presenta una arreglo lineal de cinco procesadores.

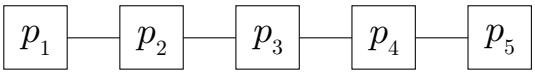


Figura 2.3: Arreglo lineal

Un arreglo lineal especial, es la red de interconexión *ring*, tal que p_1 y p_n están conectados, por lo que todos los procesadores tienen exactamente dos vecinos.

2.2.3. Mallas

La **mallas** son redes bidimensionales que son obtenidas del arreglo de los procesadores en una matriz de $r \times s$. El procesador en el renglón i y columna j es denotado por $p_{i,j}$, donde $1 \leq i \leq r$ y $1 \leq j \leq s$. Los vecinos de $p_{i,j}$ son $p_{i-1,j}$, $p_{i+1,j}$, $p_{i,j-1}$ y $p_{i,j+1}$, si existen. Los procesadores en la frontera de las columnas y renglones tienen al menos cuatro vecinos. El grado máximo de la malla es 4 y el

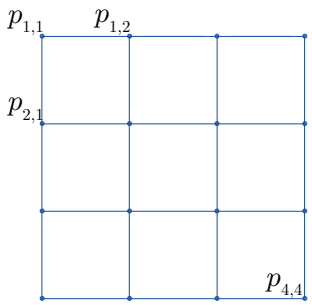


Figura 2.4: Malla M de 4×4

diámetro es $O(r+s)$ ya que la distancia desde $p_{1,1}$ a $p_{r,s}$ es $r-1+s-1 = r+s-2$. La figura 2.4 muestra una malla de 4×4.

Si en una malla la dimensión d de la matriz es mayor que 2, la red de interconexión es conocida como **malla d-dimensional**. En ésta cada procesador está conectado con dos procesadores en cada dimensión excepto con los procesadores que están en la frontera de la malla. Por lo cual, el grado máximo de la malla d-dimensional será $2 \times d$.

2.2.4. Árbol

En la red de interconexión, de **árbol**, los procesadores forman un árbol completo. De tal manera que el árbol tiene d niveles, numerados de 0 a $d-1$, y $n = 2^d - 1$ nodos cada uno de los cuales es un procesador, como se muestra en la figura 2.5.

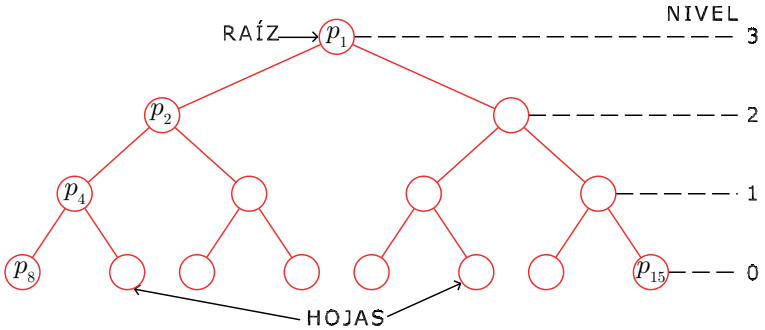


Figura 2.5: Árbol binario

Cada procesador en el nivel i está conectado por un enlace de comunicación

bidireccional a su padre en el nivel $i+1$ y a sus dos hijos en el nivel $i-1$. El procesador raíz no tiene padre y las hojas no tienen hijos. En éste modelo no hay memoria compartida. En su lugar, la memoria está distribuida entre los procesadores, cada uno de los cuales tiene una porción igual. Cada procesador almacena una copia del mismo programa. Sin embargo, no todos los procesadores ejecutan la misma instrucción al mismo tiempo, sólo ocurre cuando un procesador está activo; esto pasa si recibe datos de uno o más procesadores. Sólo la raíz y las hojas se pueden comunicar con el mundo real.

2.2.5. Pirámides

Una pirámide unidimensional se obtiene al agregar interconexiones bidireccionales en los procesadores ubicados en el mismo nivel de un árbol binario; y en cada nivel las interconexiones forman un arreglo lineal. Este concepto puede extenderse para dimensiones de orden superior.

Por ejemplo, una pirámide de dos dimensiones consiste de $(4^{d+1}-1)/3$ procesadores distribuidos entre $d+1$ niveles. Todos los procesadores en el mismo nivel están interconectados por una malla. Hay 4^d procesadores en el nivel 0 (también llamado base) organizados en una malla de $2^d \times 2^d$. Sólo hay un procesador en nivel d, también llamado cima. En general; un procesador en el nivel i, además de

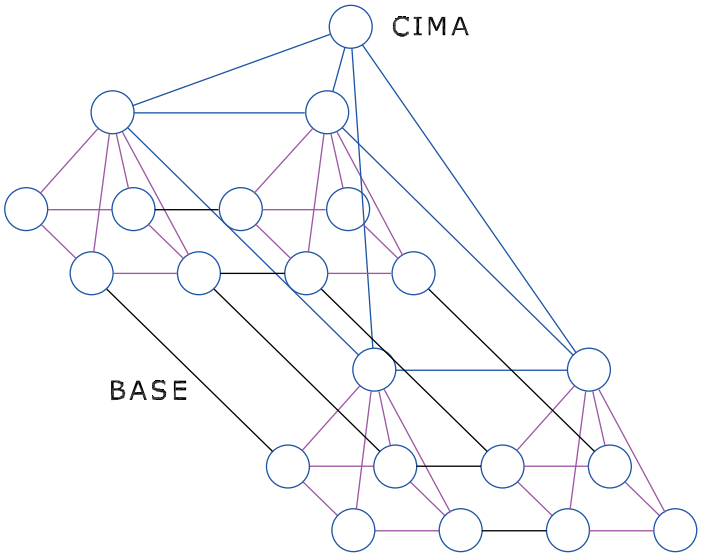


Figura 2.6: Red de interconexión pirámide

estar conectado a sus cuatro vecinos en el mismo nivel, tiene conexiones a cuatro hijos en el nivel $i-1$ con $i \geq 1$ y a uno de los padres en el nivel $i+1$, con $i \leq d-1$. El diámetro de la pirámide es $2d-1$. La pirámide se muestra en la figura 2.6.

2.2.6. Entrelazado perfecto

En este modelo de red de interconexión el parámetro n es una potencia de 2 y etiqueta a n procesadores como $p_0, p_1, p_2, ..., p_{n-1}$. En el entrelazado perfecto[2] las aristas entre p_i y p_j se obtienen de la siguiente manera:

$$j = \begin{cases} 2i & \text{si } 0 \leq i \leq n/2-1, \\ 2i+1-n & \text{si } n/2 \leq i \leq n-1. \end{cases}$$

Asimismo, se usará la representación binaria para explicar la estructura de la red de interconexión entrelazado perfecto. La representación binaria de j se obtiene desplazando cíclicamente el nodo original i una posición a la izquierda; lo que significa que el procesador $p_{i_{n-1}i_{n-2}...i_0}$ está conectado al procesador $p_{i_{n-2}i_{n-3}...i_0i_{n-1}}$. Por ejemplo, cuando $n=8$, existe un sólo sentido entre la arista del procesador p_{001} a p_{010}. Una variación de entrelazado perfecto es la red de interconexión *shuffle-exchange*. En este modelo cambiamos los enlaces unidireccionales por los bidireccionales. Además agregamos el intercambio de aristas en la red, que son aristas bidireccionales enlazadas a todos los procesadores etiquetados y cuya etiqueta es par con sus sucesores.

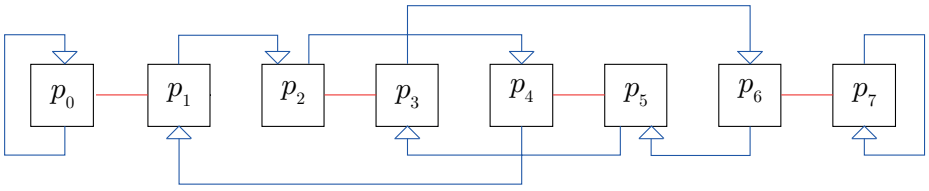


Figura 2.7: Red entrelazado perfecto

La figura 2.7 es una representación alternativa de la red de interconexión entrelazado perfecto, en la que se da un mapeo a partir del conjunto de procesadores. Esta representación explica el origen del nombre de la red; cuando una baraja de cartas se parte en dos grupos del mismo tamaño, una mezcla perfecta se obtiene al barajear (entrelazar) las cartas en los dos grupos.

[2] Se usará Entrelazado Perfecto como traducción de *Perfect Shuffle*.

2.2.7. Hipercubo

En términos generales la red de interconexión hipercubo está vinculada con procesadores paralelos y se basa en la red binaria del d-cubo introducida bajo diferentes nombres como: cubo cósmico, n-cubo binario y n-cubo booleano. Sea $n = 2^d$, con $d \geq 0$; para usar una representación binaria de i etiquetamos todos los procesadores p_i de la red con $0 \leq i \leq n-1$. Para cada procesador p_i existen exactamente d vecinos, tal que los vecinos del procesador p_i serán los procesadores p_j y donde la representación binaria de los índices i y j difieran exactamente en un bit. Se tiene que d es la dimensión de la red de interconexión, por lo que la red de interconexión es conocida como d-**hipercubo**, con d la dimensión de éste, o d-**cubo**. El grado y diámetro de un d-cubo es d. La figura 2.8 muestra los primeros hipercubos $Q_1,\ Q_2$ y Q_3 de dimensión 1, 2 y 3.

Actualmente, la más popular y no trivial red de interconexión en uso es el hipercubo. La razón obedece a que la red d-cubo posee una gran cantidad de procesadores; tantos como $n = 2^d$ y un grado pequeño, $d = \log n$. Esto logra que una gran cantidad de procesadores puedan estar interconectados usando escasos enlaces para la comunicación (d enlaces por cada procesador), mientras que al mismo tiempo mantiene el retardo de la comunicación entre los procesadores al mínimo. Aunado a esto, el d-cubo es una topología completamente simétrica por lo que minimiza los problemas de congestión en la red; esto logra el uso de procesadores idénticos, pues cada vértice juega un papel idéntico en la red de interconexión. De este modo la estructura del hipercubo puede ser descompuesta recursivamente en hipercubos de dimensión menor. Otras características del hipercubo incluyen un diámetro pequeño y la propiedad de ser una gráfica regular. Además posee algoritmos de enrutamiento óptimos y sencillos y con buena tolerancia de fallos, que enrutan mensajes entre los procesadores a través de la ruta más corta [1].

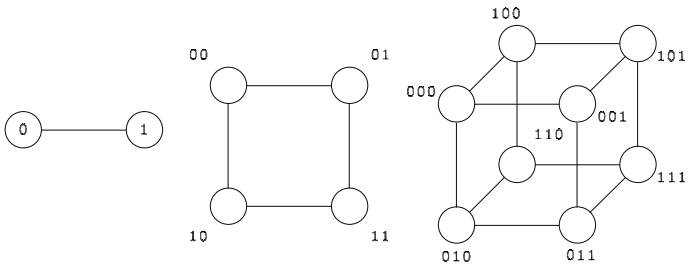


Figura 2.8: Hipercubos con $d = 1, 2, 3$

Construcción recursiva del hipercubo

Una propiedad importante del d-cubo es que puede construirse recursivamente a partir de cubos de dimensión menor: consideremos dos $(d-1)$-cubos, cuyos vértices igualmente están numerados de 0 a 2^{d-1}, al unir cada nodo del primer $(d-1)$-cubo con un nodo del segundo $(d-1)$-cubo con el mismo número, se obtiene un d-cubo. Para reetiquetar los nodos basta con concatenar un 0 con cada nodo p_i del primer hipercubo y concatenar un 1 seguido de cada p_i en el segundo; recordemos que cada nodo p_i es una representación binaria de los dos nodos similares en cada $(d-1)$-cubo. Esto se ilustra para $d = 4$ en la figura 2.9; para obtener 4-cubo, basta unir los nodos interiores del 3-cubo con los nodos exteriores del 3-cubo.

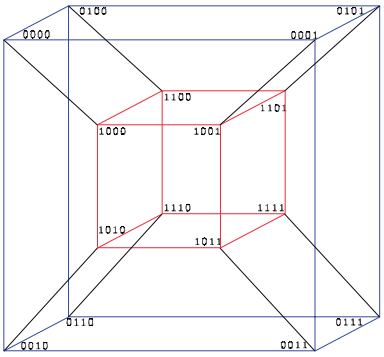
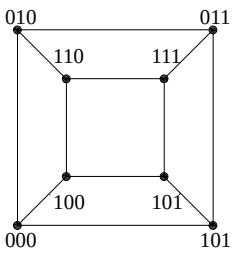


Figura 2.9: Vista 3D del 4-cubo

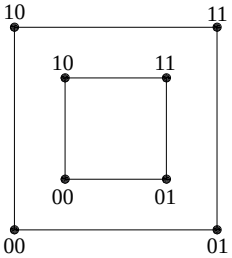
La separación de un d-cubo donde todos los nodos cuyo bit inicial es cero y donde todos los nodos cuyo bit inicial es uno producen dos subgráficas tales que cada nodo de la primera subgráfica está conectado con un nodo de la segunda subgráfica. Si eliminamos las aristas entre estas subgráficas, obtenemos dos $(d-1)$-cubos disjuntos. A la operación de dividir el d-cubo en dos $(d-1)$-cubos, de manera que todos los nodos de los dos $(d-1)$-cubos estén en correspondencia unívoca, la denominaremos *rompimiento*[3]. El rompimiento sugerido da mayor importancia al primer bit. Sin pérdida de generalidad, para una numeración dada, un rompimiento equivale a separar la gráfica en dos subgráficas obtenidas de tomar en cuenta todos los nodos cuyo bit i-ésimo es cero y cuyo bit i-ésimo es uno. Esto se conoce como rompimiento a lo largo de la i-ésima dirección de la red. Puesto que hay d bits, también hay d direcciones.

Proposición 2.1. Existen d diferentes maneras de romper un d-cubo, es decir

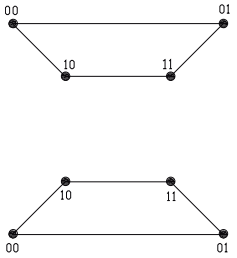
[3]Usaremos *romper* como traducción literal de *tearing*.



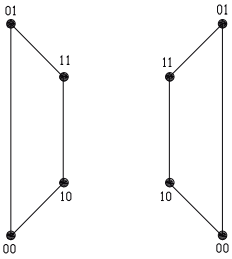
(a) Gráfica 3-cubo



(b) Rompimiento: 1er. bit



(c) Rompimiento: 2o. bit



(d) Rompimiento: 3er. bit

Figura 2.10: Rompimiento del hipercubo

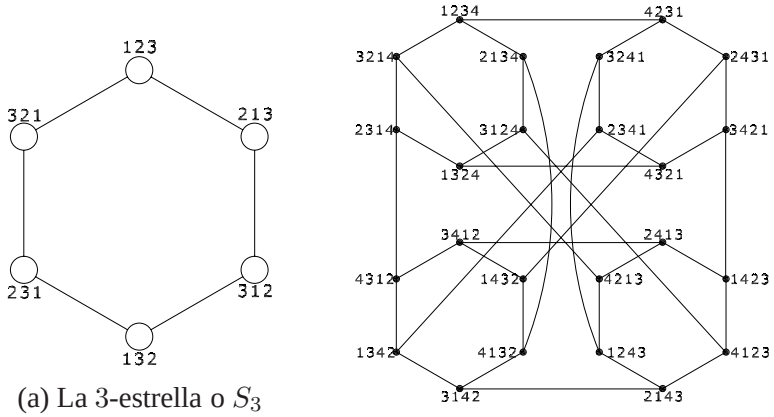
de dividir en dos $(d-1)$-subcubos de manera que sus respectivos vértices están conectados de forma unívoca [38].

Ilustrando este hecho en la figura 2.10, notamos el rompimiento del hipercubo en i-ésima dirección de éste. Cada rompimiento consiste en la división de la gráfica d-cubo en dos subgráficas, una donde las etiquetas de los nodos tienen un uno en la posición i-ésima y otra donde las etiquetas tienen un cero en la posición i-ésima.

2.2.8. n-Estrella

Para obtener una red de interconexión n-**Estrella**[4] empezamos por dar un entero n y un conjunto de enteros $\{1, 2, \ldots, n\}$. Cada procesador corresponde a las distintas permutaciones de los n símbolos. Por lo cual, el número de procesadores en una red de interconexión n-estrella es $n!$ Un procesador p estará conectado con $n-1$ vecinos que se obtendrán del intercambio del primer símbolo de p con i-ésimo símbolo, $2 \leq i \leq n$. Llamaremos a éstas, aristas de dimensión $n-1$.

[4]Usaremos n-*estrella* como traducción literal de n-*star*.



(a) La 3-estrella o S_3

(b) La 4-estrella o S_4

Figura 2.11: Ejemplos de la n-estrella

Denotamos esta red por S_n o n-estrella. Por ejemplo, si $n = 4$, el procesador p_{1234} estará conectado con p_{2134}, p_{3214} y p_{4231} por dos aristas bidireccionales. Las siguientes figuras 2.11a y 2.11b muestran S_3 y S_4 respectivamente.

La red de interconexión estrella es una atractiva alternativa al modelo hipercubo. Las propiedades en la n-estrella puede compararse favorablemente con el hipercubo, incluyendo las propiedades de simetría, la tolerancia a fallos, entre otras. Más adelante se hará un breve comparativo entre las redes de interconexión mencionadas.

2.2.9. (n, k)-Estrella

Chiang y Chen definieron una generalización de la n-Estrella al agregar un nuevo parámetro más, pues su función es controlar la cantidad de nodos (procesadores) en la topología. Esta nueva topología es conocida como (n, k)-**Estrella**[5] [15].

Tomemos n y k como dos enteros tales que $1 \leq k \leq n - 1$. Por simplicidad $\langle n \rangle = \{1, 2, \ldots, n\}$ y $\langle k \rangle = \{1, 2, \ldots, k\}$.

Definición 2.1 (Gráfica (n, k)-Estrella). Una (n, k)-estrella, denotada por $S_{n,k}$, se rige por dos enteros: n y k, donde $1 \leq k < n$. El conjunto de nodos de $S_{n,k}$ es denotado por $V(S_{n,k}) = \{p_1 p_2 \ldots p_k | p_i \in \langle n \rangle$ y $p_i \neq p_j$ para $i \neq j\}$ el conjunto

[5] Se usará (n, k)-*estrella* como traducción literal de (n, k)-*star*

de todas las permutaciones de k elementos tomados de $\langle n \rangle$. Los vecinos del nodo $p = p_1 p_2 \dots p_i \dots p_k$ se definen como sigue:

1. $p_i p_2 \dots p_{i-1} p_1 \dots p_{i+1} p_k$ a través de una arista de dimensión i, denominadas i-aristas con $2 \leq i \leq k$. Estamos intercambiando p_1 con p_i.

2. $x p_2 \dots p_i \dots p_k$ a través de una arista de dimensión 1, denominadas 1-aristas con $x \in \langle n \rangle - \{p_i \mid 1 \leq i \leq k\}$.

La figura 2.12 muestra la $(4, 2)$-estrella o $S_{4,2}$. En ella las líneas punteadas indican las i-aristas (en este caso $i = 2$ para $S_{4,2}$) y todas las líneas continuas indican 1-aristas de la gráfica.

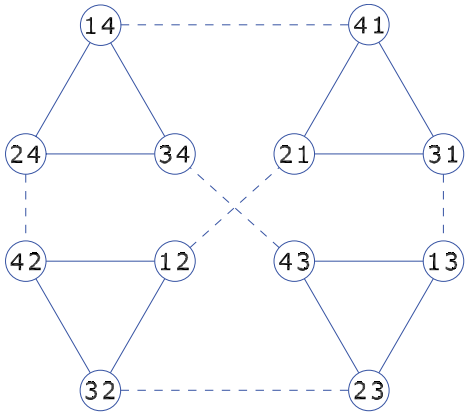


Figura 2.12: La $(4, 2)$-estrella

El número de nodos de una (n, k)-estrella es $n!/(n-k)!$. Claramente, cuando se diseña la red de interconexión en una computadora paralela, la gráfica (n, k)-estrella permite más flexibilidad que la n-estrella.

2.3. Comparando topologías

A partir de las propiedades matemáticas de la gráfica subyacente de la red física, podemos comparar las propiedades que se esperan de la red. Existen algunos criterios que nos permiten comparar qué topología es aparentemente más adecuada para cierta aplicación. Veamos algunos:

- **Grado máximo.** Éste es un importante criterio para evaluar una topología y debe ser considerado prudentemente. Desde un punto de vista teórico,

mientras el grado de la red sea más grande es mejor; caso contrario para aplicaciones prácticas, donde se prefiere un grado menor.[6]

- **Diámetro.** Dado que los procesadores necesitan comunicarse entre ellos mismos y que el tiempo para un mensaje de un procesador a otro depende de la distancia entre ellos, una red de interconexión con diámetro pequeño es mejor que una red con diámetro mayor.

- **Longitud de los enlaces.** Aunque los modelos antes mencionados son objetos abstractos, algunos de estos pueden representar computadoras paralelas para ser implementadas. Desde este enfoque, una red de interconexión es más deseable que otra si es más eficiente, más conveniente y más entendible. Un criterio particular es la longitud del enlace máximo de la red. Por ejemplo, en un arreglo lineal y una malla, todos sus enlaces tienen longitud constante, sin tener en cuenta en número de procesadores, mientras que todas las otras redes de interconexión mencionadas en las secciones 2.2.4-2.2.9 poseen enlaces que dependen del número de procesadores que tienen. Una red cuyos enlaces tienen longitud constante usualmente es más fácil y más eficiente de implementar.

Comparemos las redes de interconexión mencionadas en las secciones 2.2.2-2.2.9 con respecto a los dos primeros criterios, grado y diámetro. Se asume que el número de procesadores por red de interconexión es $O(n)$, donde n es suficientemente grande. Además $n = m!$ para la estrella.

Como se muestra en la tabla 2.1 es díficil dar un ganador. Por ejemplo, la malla es superior con respecto al grado del hipercubo, pues es el grado es constante. Por otro lado, el hipercubo posee un diámetro pequeño comparado con la malla. De igual manera podemos seguir comparando las redes de interconexión; cada una tendrá sus ventajas y sus propias desventajas y algunas son más adecuadas para ciertos problemas.

Sin embargo, aparte de estos tres criterios, hay que tomar en cuenta otros. Dado que el diseño de una red de interconexión es una parte integral de cualquier sistema paralelo o distribuido de computación [16], el escoger una red de interconexión particular determinará significativamente el desempeño del sistema. Una topología eficiente usualmente posee la siguientes propiedades: número elevado de vértices para permitir un paralelismo masivo, fuertemente conexa, regular, nodos simétricos, un algoritmo sencillo de enrutamiento, entre otros.

[6] Se usará grado para referirnos al grado máximo de la red siempre y cuando el contexto lo permita.

Topología	**Grado máximo**	**Diámetro**
Arreglo Lineal	2	$O(n)$
Mallas	4	$O(n^{1/2})$
Árboles Binarios	3	$O(\log n)$
Pirámides	9	$O(\log n)$
Entrelazado Perfecto	3	$O(\log n)$
Hipercubo	$O(\log n)$	$O(\log n)$
Estrella	n-1	$O(m)$

Cuadro 2.1: Comparando topologías [2].

Por otro lado, las aplicaciones para las cuales está destinada una red de interconexión también juegan un rol importante en la selección de la red. Uno desea tener una red de interconexión que sea la mejor para resolver un problema particular. Por ejemplo una malla es apropiada para problemas que involucran matrices mientras que un árbol está equipado para problemas de búsqueda, digamos en una base de datos. No obstante, los árboles tiene una debilidad particular en la raíz; ya que la raíz es un potencial cuello de botella, pues todas las comunicaciones entre los subárboles izquierdo y derecho deben pasar por la raíz. Alternativamente, uno quiere una red de interconexión que sea capaz de resolver una amplia variedad de problemas. Una de ellas es el hipercubo. En el presente trabajo pretendemos mostrar que la red de interconexión (n, k)-estrella es capaz de resolver una gran variedad de problemas.

Capítulo 3

Propiedades topológicas de la gráfica (n, k)-estrella

Cada red de interconexión posee características intrínsecas y por ello revisaremos la literatura escrita sobre la red de interconexión (n, k)-estrella o $S_{n,k}$, se usará indistintamente cualquiera de estas dos notaciones. Examinando la red como una gráfica, obtendremos algunas particularidades de la red. Haremos uso de la definición 2.1 de la subsección 2.2.9.

Sabemos que una gráfica G es k-regular si el grado de todo vértice es k; una gráfica regular es una que es k-regular para alguna k. Por ejemplo, la red de interconexión completa es regular, lo mismo que d-cubo. En particular tenemos el siguiente resultado:

Proposición 3.1. La gráfica $S_{n,k}$ es regular y de grado $n - 1$ [15].

Daremos una demostración exhaustiva de esta proposición. Para verlo claramente recurriremos al proceso de construcción de la gráfica (n, k)-estrella.

Consideremos el siguiente problema[1]. Una estrella con cinco vértices numerados del 1 al 5 y cinco etiquetas correspondientes a las primeras letras del alfabeto latino: A, B, C, D y E. Éstas han sido colocadas de forma aleatoria sobre los vértices de la estrella, como se puede ver en la figura 3.1. A través de un serie de intercambios deseamos colocar las etiquetas alfabéticamente, es decir, la etiqueta A corresponde al vértice 1, la etiqueta B corresponde al vértice 2, etcétera.

[1] Las ideas generales son tomadas de Akers *et al* [1].

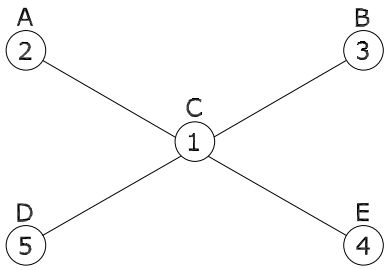


Figura 3.1: Estrella

Dado que deseamos colocar las etiquetadas en su lugar correspondiente, el problema se ajusta a las permutaciones de los símbolos A, B, C, D y E. Es decir, las permutaciones obtenidas de los símbolos representarán todos los posibles estados por los que podrá pasar una etiqueta, con el objeto de llegar a la configuración deseada. Por ejemplo, la configuración inicial mostrada en la figura 3.1 es CABED y la configuración final es ABCDE; es decir, la permutación identidad. Así que deseamos encontrar la mínima secuencia de permutaciones tal que empezando con CABED terminemos con ABCDE, y no sólo eso, sino que además las permutaciones intermedias pueden ser obtenida de la inmediata anterior por el intercambio del primer símbolo con cualquiera de los otros cuatro símbolos.

Observamos inmediatamente que este problema puede generalizarse a n marcas y, el estado de cada marca puede representarse por las permutaciones de los n símbolos. En lo que corresponde a los movimientos permitidos ocurre que se intercambia el primer símbolo con cualquiera de los otros símbolos. Por lo tanto, cualquier permutación se puede transformar en una cualquiera de las $n - 1$ permutaciones en un movimiento.

Demostración. Ahora bien, necesitamos hacer un ajuste, pues el problema se adecúa perfectamente a las permutaciones de n elementos[2]. Sin embargo, sabemos por la definición 2.1, que la (n, k)-estrella posee dos parámetros n y k que le dan mayor flexibilidad; por lo que buscamos las permutaciones de n elementos[3] tomados de k en k. Sabemos que la gráfica (n, k)-estrella posee $n!/(n - k)!$ vértices [15]; de ahí que el problema lo veríamos como la colocación en sus respectivas posiciones de todas las permutaciones de n elementos tomados de k en k (véase la definición B.17). Así que, por la

[2]La red de interconexión n-estrella se define como las permutaciones de n elementos.

[3]Durante el desarrollo de este trabajo $\langle n \rangle$ denotará el conjunto $\{1, 2, \ldots, n\}$ de símbolos y, $|X|$ representará la cardinalidad de un conjunto X.

definición de (n, k)-estrella, conocemos que si un vértice difiere de otro en el primer elemento de la etiqueta existirá un arista entre ellos; por lo que tenemos dos casos: el caso en que difieran únicamente en el primer elemento de la etiqueta la cual será una 1-arista, y el caso en que difieran en el primer elemento de la etiqueta y en la i-ésima posición de ésta, la cual será una i-arista. Además cada vértice está conectado con $n-k$ 1-aristas y una i-arista, para cada i, con $2 \leq i \leq k$, éstas últimas son $k-1$. Sumando las 1-aristas y las i-aristas tenemos que el grado de cada vértice de la (n, k)-estrella es $n-1$, y esto es válido para cualquier configuración (o permutación) inicial, por lo que el grado de todos los vértices de la gráfica $S_{n,k}$ es $n-1$ y esto indica que es una gráfica regular. □

Este resultado es importante. No sólo nos dice de cuantos enlaces disponemos en la red, sino que además, permite ajustar los parámetros n y k para un uso apropiado de los recursos disponibles o necesarios. Esto indica que el grado de una gráfica puede verse como una medida del costo de la red de interconexión. Mientras más grande el grado, mayor el costo. Por lo que se prefiere contar con un grado pequeño y manejable que con un grado mayor en la red.

3.1. Estructura jerárquica

La estructura jerárquica de la gráfica (n, k)-estrella es una de las propiedades más importantes de la $S_{n,k}$. Podemos explotar esta propiedad en varios algoritmos, en particular, en los de enrutamiento. Cuando hablamos de estructura jerárquica, nos referimos al orden y la disposición de la gráfica. Por ejemplo, si tenemos una gráfica $S_{n,k}$ podemos generar subgráficas, que también son gráficas (n', k')-estrella.

Debido a las siguientes razones, una red de interconexión con la propiedad de poseer una estructura jerárquica es atrayente:

1. Proporcionan una buena capacidad de expansión. Es decir, con el tamaño creciente de los sistemas de cómputo, las alteraciones de configuración en software como en hardware para la comunicación de cada vértice puede ser minimizada.

2. En comparación con algunas redes de interconexión no jerárquicas, se pueden integrar más vértices utilizando el mismo número de enlaces [34].

Definición 3.1 (Subgráfica $S_{n-1,k-1}(i)$). Sea $S_{n-1,k-1}(i)$ la subgráfica de $S_{n,k}$ inducida por todos los vértices que tienen el mismo último símbolo i en su etiqueta, se escoge i de 1 a n.

De la definición 2.1, fácilmente podemos observar que la $S_{n-1,k-1}(i)$ es una gráfica $(n-1, k-1)$-estrella definida por los enteros $\{1, 2, \ldots, n\} - \{i\}$. En otras palabras, cada subgráfica $S_{n-1,k-1}(i)$ es isomorfa a $S_{n-1,k-1}$. De está propiedad, $S_{n,k}$ puede ser descompuesta en n gráficas $S_{n-1,k-1}$, denotadas con $S_{n-1,k-1}(i)$ y con $1 \leq i \leq n$ [15].

- *Ejemplo.* En la figura 3.2 observamos la estructura jerárquica de la gráfica $(4, 3)$-estrella. Si seleccionamos el último símbolo de cada vértice con el mismo valor i, observamos que contiene cuatro subgráficas $(3, 2)$-estrellas generadas por cada valor de n; las cuales son: $S_{3,2}(1)$, $S_{3,2}(2)$, $S_{3,2}(3)$ y $S_{3,2}(4)$. Con los rellenos zigzag, guiones, césped y cruces identificamos las subgráficas $S_{3,2}(1)$, $S_{3,2}(2)$, $S_{3,2}(3)$ y $S_{3,2}(4)$ respectivamente. Recordando lo que pasa con el d-cubo, al romperlo se generan dos subgráficas $(d-1)$-cubo. Algo similar sucede aquí, al fijar el último símbolo, por ejemplo el uno, se genera la subgráfica $S_{3,2}(1)$ que contiene al conjunto de vértices $V(S_{3,2}(1))$ con las etiquetas 32, 23, 43, 34, 24 y 42. Lo mismo se replica con el resto de las subgráficas inducidas por último símbolo. Esta descomposición puede realizarse de forma recursiva en cada $S_{n-1,k-1}(i)$ para obtener subgráficas más pequeñas. ▲

Lema 3.1. La gráfica $S_{n,k}$ puede descomponerse en n subgráficas $S_{n-1,k-1}(i)$, $1 \leq i \leq n$ y cada subgráfica $S_{n-1,k-1}(i)$ es isomorfa a $S_{n-1,k-1}$ [15].

Demostración. Si removemos el último símbolo de todos los nodos de la gráfica en $S_{n-1,k-1}(n)$, obtenemos una $S_{n-1,k-1}$. Además, $S_{n-1,k-1}(n)$ es isomorfa a $S_{n-1,k-1}$. Por lo que, sólo es necesario demostrar que $S_{n-1,k-1}(i)$, con $1 \leq i \leq n-1$ y la $S_{n-1,k-1}(n)$ son isomorfas. Definimos una función biyectiva μ_3 en $\langle n \rangle$:

$$\mu_3(i) = n; \qquad \mu_3(n) = i; \qquad \mu_3(x) = x, \text{ para } x \in \langle n \rangle - \{i, n\}.$$

Definimos también una biyección M_3 dada por:

$$M_3(p_1 p_2 \ldots p_k) = \mu_3(p_1)\mu_3(p_2) \ldots \mu_3(p_k)$$

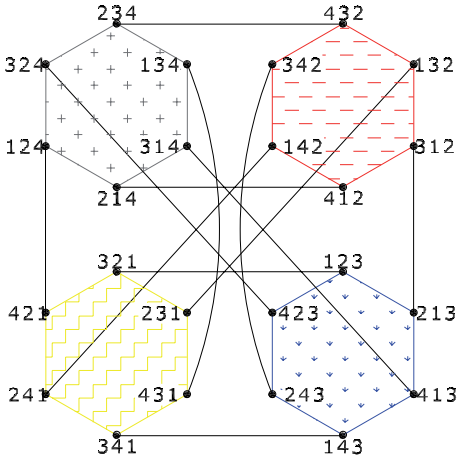


Figura 3.2: Descomposición en 4 subgráficas de la $(4,3)$-estrella

para un nodo $p = p_1p_2 \dots p_k$ en $S_{n,k}$.

Entonces podemos afirmar que, M_3 transforma los nodos de $S_{n-1,k-1}(i)$ en los nodos de $S_{n-1,k-1}(n)$ y además preserva adyacencias. □

Debemos notar que la dimensión en la cual fijamos los símbolos para obtener las subgráficas $S_{n-1,k-1}$, no tiene que ser necesariamente el último entero de la etiqueta de la permutación; como se mencionó en la definición 3.1. Esto, en general, puede ser definido como las subgráficas $S^i_{n-1,k-1}(j)$ para ser una subgráfica $S_{n-1,k-1}$ tal que todos los vértices en ellas tengan el mismo símbolo j en la i-ésima posición, con $2 \leq i \leq k$ y $1 \leq j \leq n$. Formalizado este hecho tenemos el siguiente resultado:

Proposición 3.2. Existen $k-1$ maneras diferentes de descomponer a una gráfica $S_{n,k}$ en n subgráficas $S_{n-1,k-1}$ ajenas por nodos, tal que las subgráficas son: $S^i_{n-1,k-1}(j)$, para $2 \leq i \leq k$ y $1 \leq j \leq n$ [29].

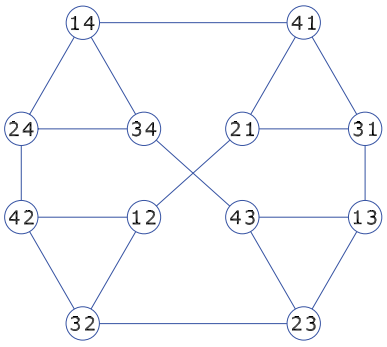
• *Ejemplo.* Para descomponer la gráfica $(4,2)$-estrella en n subgráficas $S_{n-1,k-1}$ ajenas por nodos, de acuerdo con la proposición anterior, existe una manera diferente de obtener cuatro subgráficas ajenas por nodos; pues son tantas como n, pero por cada dimensión i. En la figura 3.3b se indican las subgráficas $S_{n-1,k-1}$ que se obtienen al descomponer la gráfica $(4,2)$-estrella. Al eliminar el último símbolo de cada etiqueta obtenemos las subgráficas: $S_{3,1}(1)$, $S_{3,1}(2)$, $S_{3,1}(3)$ y $S_{3,1}(4)$. La subgráfica $S_{3,1}(1)$ nos indica que ella misma

se forma al eliminar el símbolo j de valor 1 de todas las etiquetas correspondientes a la posición $i = 2$; la cual en la figura 3.3b corresponde a la parte superior derecha. A simple vista observamos que a esta subgráfica le hace falta el símbolo 1; razón por la cual se denomina $S_{3,1}(1)$. Así que la gráfica $(4, 2)$-estrella puede descomponerse en cuatro subgráficas $S_{3,1}(j)$. En la figura 3.3b, se eliminaron los símbolos correspondientes de cada etiqueta, así como las aristas que las unían de la figura 3.3a, para observar la descomposición de la gráfica $(4, 2)$-estrella. ▲

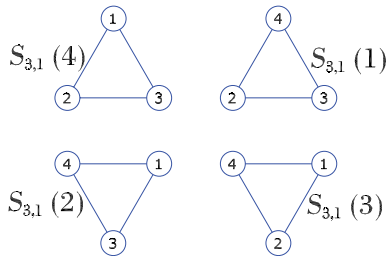
En la figura 3.3 observamos la única descomposición de la gráfica $(4, 2)$-estrella. Sin embargo, para hacer más clara la Proposición 3.2 considere la gráfica $(4, 3)$-estrella.

• *Ejemplo.* En la figura 3.4a tenemos una representación alternativa de la gráfica $(4, 3)$-estrella (veáse la figura 3.2). Recurriendo a la proposición anterior, podemos descomponer la $(4, 3)$-estrella en dos formas diferentes.

Conociendo el valor de k, i comienza a iterar con $i = 2$; lo cual nos genera cuatro subgráficas $S_{n-1,k-1}$ de dimensión dos, una por cada valor de j; tales subgráficas son: $S^2_{3,2}(1)$, $S^2_{3,2}(2)$, $S^2_{3,2}(3)$ y $S^2_{3,2}(4)$. En la figura 3.4b observamos las subgráficas. Lo mismo sucede cuando $i = 3$. Se descompone en cuatro subgráficas: $S^3_{3,2}(1)$, $S^3_{3,2}(2)$, $S^3_{3,2}(3)$ y $S^3_{3,2}(4)$ de dimensión tres. En la figura 3.4 podemos observarlas. Las primeras, más éstas últimas nos producen dos formas de descomponer la gráfica $(4, 3)$-estrella ajenas por nodos. ▲

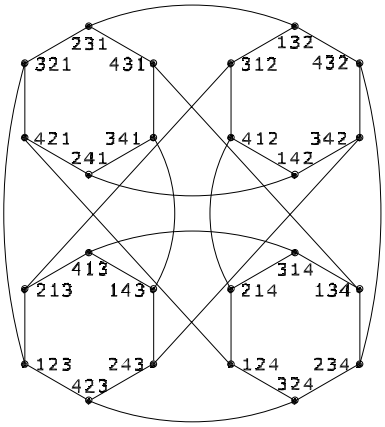


(a) Gráfica $(4, 2)$-estrella

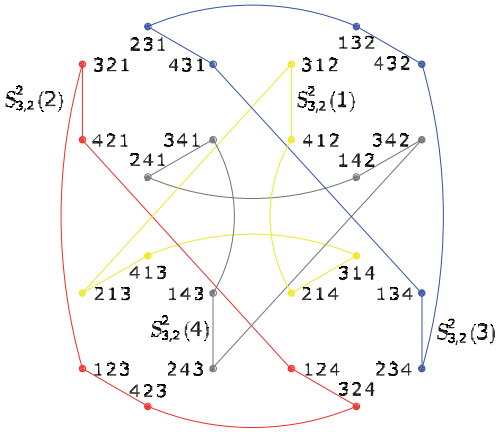


(b) Subgráficas $S_{n-1,k-1}(j)$

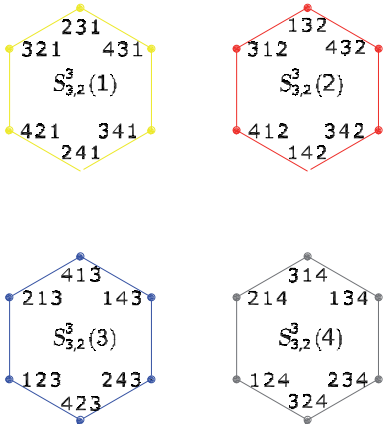
Figura 3.3: Descomposición de la gráfica $(4, 2)$-estrella



(a) Gráfica $(4, 3)$-estrella



(b) Subgráficas con $i = 2$



(c) Subgráficas con $i = 3$

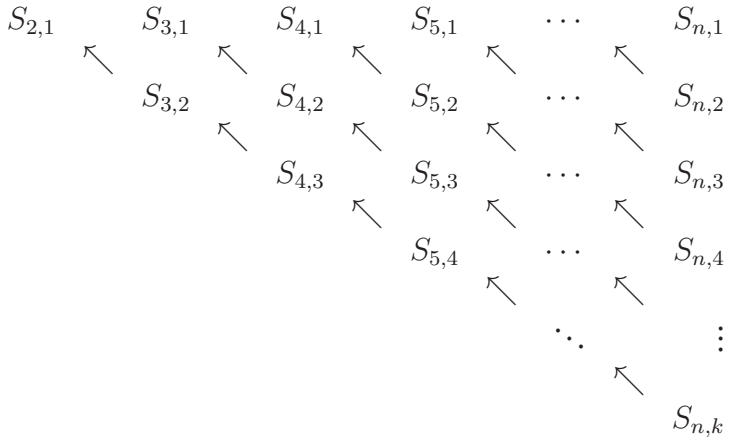
Figura 3.4: Descomposición en subgráficas de la $(4, 3)$-estrella

Nótese que en las figura 3.4b y 3.4c, en las subgráficas representadas, se eliminan las aristas de dimensión i correspondientes a cada caso; sin que ello signifique que la gráfica quedará así, sólo se ejemplifica la propiedad. En el presente material, mientras no se indique lo contrario, usaremos la descomposición de la gráfica (n, k)-estrella en subgráficas $S_{n-1,k-1}(i)$ en la última dimensión del vértice.

La descomposición de la gráfica $S_{n,k}$ en subgráficas $S_{n-1,k-1}(i)$ inducidas por el último símbolo de cada etiqueta es una descomposición que puede realizarse de forma recursiva en cada subgráfica $S_{n-1,k-1}(i)$ para obtener subgráficas más pequeñas. Pensemos en la gráfica $S_{5,4}$. Aplicando la descomposición en el último símbolo obtenemos cinco subgráficas $S_{4,3}$. Haciendo lo mismo, se inducen cuatro

subgráficas $S_{3,2}$. Por último se infieren tres subgráficas $S_{2,1}$; nuestro caso base. De este ejemplo, podemos reparar brevemente en la estructura jerárquica de nuestro caso de estudio, la (n, k)-estrella. Cada gráfica $S_{n,k}$, por definición, contendrá en su estructura interna subgráficas $S_{n,k}$ más pequeñas que serán ajenas por nodos.

El siguiente diagrama nos permite observar el proceso recursivo en la generación de subgráficas (n, k)-estrella, así como la estructura jerárquica de la red. Las flechas nos indican a la vez la descomposición que resultaría del último símbolo, así como las subgráficas que contiene en su estructura una $S_{n,k}$ cualquiera. Pensemos en la gráfica $S_{5,3}$ que aparece en el diagrama. La $S_{5,3}$, contiene en su estructura cinco gráficas $S_{4,2}$, además, si descomponemos la gráfica, como lo indica la flecha tendremos cinco subgráficas $S_{4,2}$.



La generación de la siguiente gráfica (n, k)-estrella puede partir del conocimiento previo de que la gráfica $S_{n,k}$ será parte de la estructura de la gráfica $S_{n+1,k+1}$. Es decir, en su estructura interna contendrá $n + 1$ copias de la gráfica $S_{n,k}$ y, no sólo eso, sino que además contendrá las aristas que unen a cada una de esas $n + 1$ copias. El siguiente resultado, indica la cantidad de aristas que existen entre cada una de esas subgráficas.

Lema 3.2. Existen $(n-2)!/(n-k)!$ k-aristas entre cualesquiera dos subgráficas $S_{n-1,k-1}(i)$ y $S_{n-1,k-1}(j)$; cada uno de los nodos de $S_{n-1,k-1}(i)$ está conectado exactamente con un nodo de $S_{n-1,k-1}(j)$ [15].

Demostración. Sabemos que para cualquier par de nodos, digamos: $jp_2 \dots p_{k-1}i \in S_{n-1,k-1}(i)$ e $ip_2 \dots p_{k-1}j \in S_{n-1,k-1}(j)$, con $i \neq j$, existe una k-arista que los conecta entre si.

Cada $p_2p_3 \dots p_{k-1}$ es una única permutación de $k-2$ símbolos distintos elegidos de $n-2$ símbolos de $\langle n \rangle - \{i, j\}$; ya que por la definición de permutaciones, cada $p_2p_3 \dots p_{k-1}$ es una única variación del orden de los elementos de $\langle n \rangle - \{i, j\}$ pues no admite repetición.

Así que, para obtener todas las k-aristas que conectan a las subgráficas $S_{n-1,k-1}(i)$ y $S_{n-1,k-1}(j)$ tomamos todas las permutaciones de $n-2$ símbolos en grupos de $k-2$. Por lo tanto existen k-aristas dadas por $(n-2)!/((n-2)-(k-2))! = (n-2)!/(n-k)!$ entre cualesquiera dos subgráficas $S_{n-1,k-1}(i)$ y $S_{n-1,k-1}(j)$. □

Del resultado anterior haremos la siguiente observación: sea $S_{n,k}$ una gráfica (n,k)-estrella. El resultado se aplica a gráficas $S_{n,k}$ que contengan entre sus subgráficas k-aristas. Es decir, si $k = 1$, por la definición 2.1, esta gráfica sólo contiene 1-aristas y no hay en ellas k-aristas.

Dado que no existe una forma única para respresentar una gráfica, es necesario encontrar, en la medida de lo posible, la mejor representación gráfica de ésta para su estudio. En nuestro caso particular, por el lema anterior, la gráfica $S_{n,k}$ puede representarse gráficamente por la unión, entre sus subgráficas, por medio de k-aristas dentro del intervalo de 2 hasta k, para los casos que aplique. Sin embargo, como se verá en la sección siguiente, se puede usar la estructura de una gráfica completa para facilitar su construcción. De hecho, la gráficas $S_{n,k}$ del primer renglón del diagrama anterior son todas gráficas completas.

Además, la diagonal inferior del diagrama corresponde con las gráficas isomorfas a la red n-estrella. Es patente, por tanto, que la (n,k)-estrella posee una escalabilidad superior a ésta última. Si mantenemos fijo n siempre obtendremos $n-1$ variantes de la gráfica $S_{n,k}$.

Gracias a esta estructura jeráquica de la gráfica (n,k)-estrella podemos realizar descomposiciones y explotar esta propiedad en los algoritmos de enrutamiento y comunicación, que revisaremos en el siguiente capítulo.

3.2. Isomorfismo en $S_{n,k}$

Analizaremos en la gráfica (n,k)-estrella el concepto de isomorfismo; éste pretende captar la idea de poseer la misma estructura o forma. Se establecen relaciones isomorfas entre las gráficas, o bien dadas por la descomposición de la misma, o debido a poseer la misma estructura que otra gráfica conocida.

Lema 3.3. La gráfica (n, k)-estrella $S_{n,n-1}$ es isomorfa a la gráfica n-estrella S_n [15].

Demostración. Para probar que $S_{n,n-1}$ y S_n son isomorfas, es necesario quitar el último símbolo de todos los nodos de S_n, luego entonces, obtendremos una $S_{n,n-1}$ que cumple la definición 2.1. Para probar que existe el isomorfismo, definimos una biyección M_2 de los nodos de S_n a los nodos de $S_{n,n-1}$ dada por:

$$M_2(p_1p_2 \dots p_{n-1}p_n) = p_1p_2 \dots p_{n-1}$$
$$\text{para un nodo } p = p_1p_2 \dots p_{n-1}p_n \text{ de } S_n$$

Además, M_2 preserva adyacencias. Sean p y q dos nodos adyacentes por una i-arista en S_n. Si $2 \leq i \leq n-1$ entonces $M_2(p)$ y $M_2(q)$ serán adyacentes con una i-arista en $S_{n,n-1}$; por otra parte, si $i = n$ ocurre que $M_2(p)$ y$M_2(q)$ serán adyacentes con una 1-arista en $S_{n,n-1}$. Por lo tanto $S_{n,n-1}$ y S_n son isomorfas gracias a la función de biyección M_2. □

En la definición 2.1 se especifica cómo construir una (n, k)-estrella, sin embargo haremos uso de una definición alternativa (ligeramente diferente) que permitirá realizar algunas observaciones importantes sobre la gráfica (n, k)-estrella, además que nos servirán para las siguientes propiedades y resultados.

Definición 3.2 (Gráfica (n, k)-estrella). La gráfica $S_{n,k}$, con $1 \leq k \leq n$, estará regida por dos parámetros n y k. El conjunto de vértices de $S_{n,k}$ consiste de todas las permutaciones de k elementos elegidos del conjunto base $\{1, 2, \dots, n\}$. Dos vértices $p_1, p_2, \dots, p_k$ y $q_1, q_2, \dots, q_k$ serán adyacentes si ocurre que:

1. si existe una r en el intervalo $2 \leq r \leq k$ tal que $p_1 = q_r$, $p_r = q_1$ y $p_i = q_i$ para $i \in \{1, 2, \dots, k\} - \{1, r\}$ y,
2. $p_i = q_i$ para $i \in \{2, \dots, k\}$, con $p_1 \neq q_1$ [14].

Comparando las dos definiciones podemos hacer algunas observaciones importantes. Dado un vértice $p_1, p_2, \dots, p_n$ en la gráfica, por la regla 1, tendrá $k-1$ vecinos generados por el intercambio de p_1 con p_i para $2 \leq i \leq k$. Este mismo vértice, vía la regla 2, tendrá $n-k$ vecinos generados por el intercambio de p_1 con cada elemento en $\{1, 2, \dots, n\} - \{p_1, p_2, \dots, p_k\}$. La regla 1 también es la regla para determinar la k dimensión de la gráfica (n, k)-estrella.

Lema 3.4. La gráfica $S_{n,1}$ es isomorfa a la gráfica completa K_n [13]

Demostración. Para demostrar la existencia del isomorfismo es necesario hacer algunas observaciones sobre estas dos topologías.

Recordemos que la gráfica completa K_n posee n vértices incidentes todos entre sí, de tal manera que el grado $\delta(v)$ de un nodo v cualquiera es $n-1$.

Por otra parte, por la definición 3.2, un nodo en la gráfica $S_{n,1}$ será igual a $p = p_i$, con $1 \leq i \leq n$; pues son las permutaciones de n elementos tomados de uno en uno. Es decir, un nodo $p = p_1, p_2, \ldots, p_k$ cualquiera será p_i pues $k = 1$ y las permutaciones de $P_{n,1} = n$.

Definimos una biyección M_3 de los nodos de $S_{n,1}$ a K_n dada por:

$$M_3(p_i) = k_i \text{ para un nodo } p = p_i \text{ de } S_{n,1}, \text{ con } 1 \leq i \leq n.$$

La función M_3 preserva adyacencias. La regla uno nos indica que un nodo p tiene $k-1$ vecinos; sin embargo, la gráfica $S_{n,1}$ no posee vecinos vía esta regla, pues $k = 1$. Por la regla dos, sabemos que un nodo p tendrá $n-k$ vecinos, es decir, existirán $n-1$ vecinos en la gráfica $S_{n,1}$. Todos los $n-1$ vecinos serán adyacentes, pero no entre sí. Por lo que, p y q dos nodos adyacentes en $S_{n,1}$ serán adyacentes vía la función $M_3(p)$ y $M_3(q)$ en K_n. Por lo tanto, la gráfica $S_{n,1}$ es isomorfa a la gráfica completa K_n. □

Los siguientes resultados describen otras propiedades elementales de la topología (n,k)-estrella.

Teorema 3.1. Sea $\{p_2, p_3, \ldots, p_k\} \subset \langle n \rangle$. Sea G la subgráfica de $S_{n,k}$ inducida por los vértices de la forma $qp_2 \ldots p_k$ donde $q \in \langle n \rangle - \{p_2, p_3, \ldots, p_k\}$. Entonces G es isomorfa a K_{n-k+1} la gráfica completa de $n-k+1$ vértices [14].

Teorema 3.2. Sea $\{p_1, p_2, \ldots, p_k\} \subset \langle n \rangle$, con $k \geq 3$. Sea G la subgráfica de $S_{n,k}$ inducida por los vértices cuyas etiquetas son las permutaciones $p_1, p_2, \ldots, p_k$. Entonces G es isomorfa a la gráfica S_k [14].

El siguiente resultado fue estudiado previamente en este trabajo, sin embargo resulta útil mencionarlo.

Teorema 3.3. Sea G la subgráfica de $S_{n,k}$, con $k \geq 2$ inducida por los vértices cuyas etiquetas contengan el mismo símbolo en la k-ésima posición. Entonces G es isomorfa a $S_{n-1,k-1}$ [14].

3.3. Simetría en (n, k)-Estrella

Por ser red de interconexión, la gráfica (n, k)-estrella posee una correspondencia recíproca en la disposición de sus nodos y aristas. Cuando una gráfica es simétrica por vértices se verá igual desde cualquier vértice, por otro lado, si es simétrica por aristas desde cualquier arista se verá igual. En la presente sección estudiaremos la disposición de los nodos y aristas en la red (n, k)-estrella.

La simetría juega un papel importante en el equilibrio de la carga de trabajo de la red y el enrutamiento. Esto puede simplificar el enrutamiento, ya que todos los nodos comparten el mismo itinerario de la red, por lo cual se pueden usar las direcciones de la ruta con respecto a la posición relativa. Por otro lado, al ser la red simétrica por aristas, permite mejorar el equilibrio de carga de trabajo de la red, pues no hay razón de favorecer la comunicación por medio de una arista sobre otra.

Definición 3.3 (Gráfica simétrica por nodos). Sea G una gráfica, se dice que es simétrica por nodos si y sólo si para cualquier par de nodos u y v existe un automorfismo de la gráfica que mapea u en v [16].

Definición 3.4 (Gráfica simétrica por aristas). Sea G una gráfica, se dice que es simétrica por aristas si y sólo si para cualesquiera par de aristas x e y, existe un automorfismo de la gráfica que mapea x en y [16].

Tales propiedades de simetría en una gráfica G son muy importantes para una red de interconexión. Por ejemplo, una gráfica simétrica por nodos permite ver a todos los procesadores sin distinción alguna. Además, una gráfica simétrica por nodos o por aristas, minímiza la congestión de los vértices o las aristas cuando se enrutan los mensajes en la gráfica.

Teorema 3.4. La gráfica (n, k)-estrella es simétrica por nodos [15].

Demostración. Es necesario demostrar que para dos nodos cualesquiera en la gráfica $S_{n,k}$ existe un automorfismo que mapea un nodo en el otro. Usaremos una función auxiliar que permitirá aplicar un mapeo inyectivo sobre todas las k-permutaciones de n. Sean $p = p_1p_2 \dots p_k$ y $q = q_1q_2 \dots q_k$ dos nodos en $S_{n,k}$; sean $P = \{p_1, p_2, \dots, p_k\}$ y $Q = \{q_1, q_2, \dots, q_k\}$.

Se tiene que $|P| = |Q|$ entonces:

$$|Q - P| = |Q| - |P \cap Q| = |P| - |P \cap Q| = |P - Q|.$$

Ahora, definimos una función biyectiva μ_1 en $\langle n \rangle$:

- $\mu_1(p_i) = q_i$ para $1 \leq i \leq k$, es decir, para $p_i \in P$;
- $\mu_1(x) = y$, un mapeo biyectiva para $x \in Q - P$ y $y \in P - Q$.
- $\mu_1(z) = z$, para $z \in \langle n \rangle - P \cup Q$.

Sea M_1 una función biyectiva en $S_{n,k}$:

$$M_1(t) = \mu_1(t_1)\mu_1(t_2)\ldots\mu_1(t_k) \text{ para toda } t = t_1t_2\ldots t_k \text{ en } S_{n,k}$$

Por lo cual, M_1 mapea el nodo p en el nodo q. Además, M_1 es un automorfismo porque si dos nodos s y t son adyacentes en $S_{n,k}$, entonces sus imágenes $M_1(s)$ y $M_1(t)$ estarán conectados por una arista. Es decir, sea $s = s_1s_2\ldots s_i\ldots s_k$ entonces t será $t = s_is_2\ldots s_1\ldots s_k$ (intercambiando s_1 con s_i lo que genera i-vecino) o que ocurra que $t = t_1s_2\ldots s_i\ldots s_k$, con $t_1 \in \langle n \rangle - \{s_1 | 1 \leq i \leq k\}$, es decir, un 1-vecino. Consideremos que:

$$M_1(s) = \mu_1(s_1)\mu_1(s_2)\ldots\mu_1(s_i)\ldots\mu_1(s_k)$$

y

$$M_1(t) = \mu_1(t_1)\mu_1(s_2)\ldots\mu_1(s_i)\ldots\mu_1(s_k)$$

o

$$M_1(t) = \mu_1(s_i)\mu_1(s_2)\ldots\mu_1(s_1)\ldots\mu_1(s_k),$$

entonces $M_1(s)$ y $M_1(t)$ son adyacentes. Por lo tanto, la gráfica (n, k)-estrella es simétrica por nodos. □

Se tiene que no todas las gráficas (n, k)-estrella, $S_{n,k}$, son simétricas por aristas. La gráfica (n, k)-estrella posee dos tipos de aristas (las 1-aristas y las i-aristas), esto le impide ser simétrica para cualquier par de aristas en $S_{n,k}$. Sin embargo, al poseer estas aristas la (n, k)-estrella, ocurre que son simétricas entre aristas del mismo tipo. Las siguientes proposiciones lo muestran.

Lema 3.5. Cada 1-arista en $S_{n,k}$ es simétrica con cualquier otra 1-arista [16].

Demostración. Mostraremos que para cualquier par de 1-aristas en $S_{n,k}$ existe un automorfismo que mapea una arista en otra. Sea (p, q) la 1-arista definida por $p = p_1p_2\ldots p_k$ y $q = q_1p_2\ldots q_k$. Sea (r, s) cualquier otra 1-arista en $S_{n,k}$ dada por $r = r_1r_2\ldots r_k$ y $s = s_1r_2\ldots s_k$. Definimos $P = \{p_i | 1 \leq i \leq k\}$, $R = \{r_i | 1 \leq i \leq k\}$ como dos subconjuntos de $\langle n \rangle$.

Sea μ_1 una función biyectiva en $\langle n \rangle$ dado por:

(i) $\mu_1(p_\alpha) = r_\alpha$ para $1 \leq \alpha \leq k$.

(ii) $\mu_1(q_1) = s_1,$ con $q_1 \notin P$ y $s_1 \notin R$.

(iii) $\mu_1(r_x) = p_y$, una función uno a uno para:

$$r_x \in R - P \cup \{q_1\} \text{ y } p_y \in P - R \cup \{s_1\}.$$

Nótese que:

$$|R-P\cup\{q_1\}| = |R|-|P\cap R|-1 = |P|-|P\cap R|-1 = |P-R\cup\{s_1\}|.$$

(iv) $\mu_1(z) = z$ para $z \in \langle n\rangle - P \cup R \cup \{q_1\}$.

Sea M_1 una función uno a uno en $S_{n,k}$ definida por:

(i) $M_1(t) = \mu_1(t_1)\mu_1(t_2)\ldots\mu_1(t_k)$ para toda $t = t_1t_2\ldots t_k \in S_{n,k}$.

Por la función M_1 se tiene $M_1(p) = r$ y $M_1(q) = s$. La función M_1 es un automorfismo en $S_{n,k}$, esto se debe a que si dos nodos u y v son adyacentes a una i-arista con $1 \leq i \leq k$ entonces $M_1(u)$ y $M_1(v)$ serán adyacentes con la i-arista. Por lo tanto, la gráfica $S_{n,k}$ es simétrica por 1-aristas. □

Lema 3.6. Cada i-arista en $S_{n,k}$ es simétrica con cualquier otra j-arista [16].

Demostración. Mostraremos que para cualquier i-arista y j-arista en $S_{n,k}$, con $2 \leq i, j \leq k$, existe un automorfismo que mapea una arista en otra. Sea (p, q) la i-arista, definida por $p = p_1p_2\ldots p_i\ldots p_k$ y $q = p_ip_2\ldots p_1\ldots q_k$. Sea (r, s) la j-arista con $i < j$ en $S_{n,k}$ dada por $r = r_1r_2\ldots r_j\ldots r_k$ y $s = r_jr_2\ldots r_1\ldots r_k$. Definimos $P = \{p_i|1 \leq i \leq k\}$, $R = \{r_i|1 \leq i \leq k\}$ como dos subconjuntos de $\langle n\rangle$.

Sea μ_2 una función biyectiva en $\langle n\rangle$ dada por:

(i) $\mu_2(p_\alpha) = r_\alpha$ para $1 \leq \alpha \leq k,$ con $\alpha \neq i$ y $\alpha \neq j$.

(ii) $\mu_2(p_i) = r_j$.

(iii) $\mu_2(p_j) = r_i$.

(iv) $\mu_2(r_x) = p_y$, una función biyectiva para:

$$r_x \in R - P \text{ y } p_y \in P - R.$$

Nótese que:

$$|R - P| = |R| - |P \cap R| = |P| - |P \cap R| = |P - R|.$$

(v) $\mu_2(z) = z$ para $z \in \langle n \rangle - P \cup R$.

Sea M_2 una función biyectiva en $S_{n,k}$ definida por:

(i) $M_2(t) = \mu_2(t_1)\mu_2(t_2)\ldots\mu_2(t_k)$ para toda $t = t_1t_2\ldots t_k$ en $S_{n,k}$.

Sea E_{ij} una función biyectiva en $S_{n,k}$ definida por:

(i) $E_{ij}(t) = t_1t_2\ldots t_{i-1}t_jt_{i+1}\ldots t_{j-1}t_it_{j+1}\ldots t_k$, para toda $t = t_1t_2\ldots t_k$ en $S_{n,k}$.

Sea F una función uno a uno definida por la composición M_2 seguida de E_{ij}, esto es $F = E_{ij} \circ M_2$ en $S_{n,k}$. Notamos que $F(p) = r$ y $F(q) = s$. Verificamos que la función F es un automorfismo en $S_{n,k}$.

Esto obedece a que si dos nodos u y v son adyacentes por una h-arista entonces $F(u)$ y $F(v)$ serán adyacentes por medio de:

a) otra h-arista si $h \neq i$ y $h \neq j$; o

b) una j-arista si $h = i$; o

c) una i-arista si $h = j$.

Por lo tanto, la gráfica $S_{n,k}$ es simétrica por i-aristas. □

3.4. Tolerancia a fallos

Informalmente, se tiene que la tolerancia a fallos[4] es la capacidad de la red de interconexión para funcionar eficazmente ante la presencia de diversos fallos; es decir, es la capacidad de la red para llevar a cabo sus tareas (entregar paquetes entre los nodos no defectuosos de la red) en la presencia de uno o más fallos.

Para un análisis completo de nuestro caso de estudio se considerará la tolerancia a fallos de la red, ya que esta propiedad es una medida del desempeño de la misma [34]. En sistemas de cómputo paralelo con una alta probabilidad de fallar esta propiedad es una característica deseable, dado que permite mantener la interconexión entre los nodos de la red. A este respecto, han sido propuestos diversos diseños tolerantes a fallos para arquitecturas multiprocesador basados en modelos teóricos de gráficas. Esto impulsa a tomar decisiones para el diseño de la red. Por

[4]Se usará *tolerancia a fallos* como traducción literal de *fault tolerant*.

ejemplo, si una red en particular sólo tolerara una falla, no se podría usar enrutamiento simple; ya que ante la presencia de una falla la red podría desconectarse o desconectar las aristas que inciden en el nodo que falla. Es preferible, por tanto, que una red se degrade grácilmente ante las fallas, es decir, que el desempeño de la red se reduzca gradualmente con el número de fallas.

Para muchos sistemas ser tolerante a fallos en un sólo nodo suele ser suficiente; ya que si la probabilidad de que falle un nodo cualquiera es baja, la de que se presenten múltiples fallas es extremadamente baja, permitiendo que la red de interconexión pueda continuar funcionado. Sin embargo, para sistemas de alta disponibilidad no es suficiente, por lo que los componentes defectuosos deben ser reemplazados antes de que las fallas se presenten.

Se pueden establecer dos modelos de fallos, a saber, el modelo estático y el modelo dinámico. El primero supone la existencia de fallos en la red; se conocen previamente las fallas desde el momento en que arranca el sistema. Se tiene un conocimiento global y exacto de las fallas en la red. El segundo asume que los fallos pueden aparecer en cualquier momento a lo largo de la operación del sistema. En este caso, el conocimiento de los fallos puede ser global en toda la red o local hacia los nodos adyacentes. Sin importar el modelo empleado, en todo momento nos referimos a fallos permanentes, es decir fallos que permanecen hasta que son reparados, desestimando aquellos que son transitorios por su naturaleza [35].

Complementariamente a los anteriores modelos de fallos, cabe distinguir el modelo de fallos individual y el modelo de fallos por bloque. El modelo de fallos individual supone la aparición de fallos aislados en los nodos o aristas de la red. Por su parte el modelos de fallos por bloque etiqueta componentes de la red como fallidos cuando en realidad no lo están. El principal inconveniente de este modelo es la innecesaria capacidad de procesamiento que demanda.

El empleo de modelos tolerantes a fallos permite que la red de interconexión siga operando a pesar de la presencia de fallos. Sin embargo, la red continuará con la operación del sistema hasta que el fallo sea reparado, mientras tanto, el sistema podría tener un menor rendimiento.

Definimos formalmente la tolerancia a fallos de una red de interconexión.

Definición 3.5 (Tolerancia a fallos). La tolerancia a fallos de una gráfica G se define como el número máximo f tal que si se eliminan cualesquiera f nodos de G, la subgráfica resultante todavía está conectada [15]. Se tiene que es tolerante a f-fallos siempre que f sea máximo.

Este concepto resulta familiar, ya que está íntimamente relacionado con la noción de conexidad en Teoría de Gráficas, por lo cual puede observarse que la

tolerancia a fallos es algo menos que conexidad. Si todos los vecinos de un vértice elegido son eliminados de G entonces el vértice elegido quedará disconexo del resto de la gráfica. De este modo, la tolerancia a fallos de una gráfica puede ser a lo sumo de $\delta_G - 1$, donde δ_G es el grado de la gráfica. Una gráfica cuya tolerancia a fallos es exactamente $\delta_G - 1$ se dice que es *máximamente tolerante a f-fallos*.

Consideremos lo siguiente: dado que varias fallas pueden presentarse, ¿esto hará que los procesadores se aíslen entre sí? o ¿pueden seguir funcionado coherentemente? Si la red de interconexión es tolerante a f-fallos podrán ser eliminados f o menos vértices que f y permanecer conexa. Sin embargo, si ha ocurrido un fallo las trayectorias entre los vértices que sean generadas mediante un algoritmo de enrutamiento permitirán se mantenga la conexión entre los nodos incluso si una o más rutas fallan, siempre y cuando éstas no sean mayor que f. Por otro lado, si las fallas se incrementan la red puede partirse de tal manera que no existan trayectorias entre algunos nodos particulares.

La gráfica n-estrella es tolerante a fallos, por está razón pertenece a una familia de gráficas jerárquicas [13, 1]. La (n, k)-estrella preserva algunas propiedades de la gráfica n-estrella, entre otras el ser máximamente tolerante a f-fallos [15]. Se analizará en las siguientes líneas como la gráfica (n, k)-estrella actúa ante la presencia de fallos.

Analicemos el siguiente teorema. Denotaremos la tolerancia a fallos de una gráfica G cualquiera como: $f(G)$.

Teorema 3.5. La tolerancia a fallos de la gráfica (n, k)-estrella es $n - 2$ [15].

Demostración. Afirmamos que:

$$f(S_{n,k}) \geq f(S_{n-1,k-1}) + 1 \text{ para } 2 \leq k \leq n.$$

Demostraremos eso probando que la gráfica $S_{n,k}$ permanece conexa incluso si $f(S_{n-1,k-1}) + 1$ de sus nodos fallan. Consideremos dos casos:

(i) En primer lugar supongamos que todos los nodos defectuosos estarán contenidos en una subgráfica inducida $S_{n-1,k-1}(i)$.

Conocemos que la tolerancia a fallos de la subgráfica $S_{n-1,k-1}$ es $f(S_{n-1,k-1})$, la subgráfica $S_{n-1,k-1}(i)$ podría desconectarse debido a las $f(S_{n-1,k-1}) + 1$ fallas. Sin embargo, cada nodo no defectuoso de $S_{n-1,k-1}(i)$ tiene una k-arista que lo une con otra subgráfica inducida $S_{n-1,k-1}(j)$, con $j \neq i$. Todas las demás subgráficas $S_{n-1,k-1}$ permanecen conexas ya que no tienen fallas. Por lo cual, la gráfica $S_{n,k}$ permanece conexa.

(ii) Suponemos que las $f(S_{n-1,k-1}) + 1$ fallas se distribuyen entre más de una de las subgráficas inducidas $S_{n-1,k-1}(i)$ de $S_{n,k}$. Dado que hay a lo más $f(S_{n-1,k-1})$ fallas en cualquier subgráfica inducida, cada subgráfica inducida $S_{n-1,k-1}(i)$ permanece conexa.

Necesitamos probar que para cualesquiera dos de las n subgráficas inducidas $S_{n-1,k-1}(i)$ que son, con $1 \leq i \leq n$, permanecerán conexas entre sí. Por el lema 3.2, cada $S_{n-1,k-1}(i)$ tiene $(n-2)!/(n-k)!$ nodos adyacentes en cada una de las subgráficas inducidas $S_{n-1,k-1}(j)$, con $j \neq i$.

Si consideramos cada $S_{n-1,k-1}(i)$ de $S_{n,k}$ como un hipernodo, entonces la gráfica resultante es una gráfica completa de n nodos. Para desconectarla tendríamos que eliminar $n-1$ de sus nodos. Por lo tanto, si $S_{n,k}$ se desconecta debido a las $f(S_{n-1,k-1}) + 1$ fallas, entonces:

$$f(S_{n-1,k-1}) + 1 \geq (n-1)!/(n-k)!$$

No obstante:

$$f(S_{n-1,k-1}) + 1 < \delta_G(S_{n-1,k-1}) + 1$$

$$= (n-2) + 1 = n - 1 \leq (n-1)!/(n-k)!, \text{ para } 2 \leq k \leq n,$$

lo cual no puede ser, así que $S_{n,k}$ permanece conexa.

Basado en lo anterior tenemos la siguientes relaciones:

$$f(S_{n,k}) \geq f(S_{n-1,k-1}) + 1$$

y la condición inicial:

$$f(S_{n-k+1,1}) = n - k - 1,$$

ya que $S_{n-k+1,1}$ es isomorfa a la gráfica completa de $n-k+1$ nodos.

Por consiguiente, $f(S_{n,k}) \geq n-2$. El grado de una $S_{n,k}$ es $n-1$, lo cual implica que $f(S_{n,k}) \leq n-2$. Por lo tanto, $f(S_{n,k}) = n-2$. □

Es necesario mencionar, que algunas redes de interconexión han sido estudiadas y se han obtenido cotas que nos informan sobre su posible comportamiento ante la presencia de fallos, sin embargo, esto no quiere decir que no se pueda encontrar una cota mejor o máxima a las ya dadas. Este es el caso de la gráfica

(n, k)-estrella. Aunque la cota parece ser poca, no quiere decir que el sistema no seguirá funcionando, sino más bien, que no se poseerá de ciertas trayectorias que permitan la comunicación entre los nodos, e incluso puede que no suceda, pues puede darse el caso de que no sea necesario comunicarse con los nodos que han fallado En caso contrario, el rendimiento del sistema empeorará.

Capítulo 4

Algoritmos para la gráfica (n, k)-estrella

Los algoritmos paralelos a menudo exigen acciones de comunicación coordinadas que implican múltiples procesos [23]. Basado en las permutaciones de sus símbolos, la red (n, k)-estrella proporciona a los nodos que la componen canales de comunicación que serán usados por los algoritmos de enrutamiento y comunicación. Estos algoritmos, hacen uso de las propiedades topológicas de la gráfica $S_{n,k}$; al ser diseñados con estas ideas en mente, permiten alcanzar los objetivos planteados, a saber, que la red haga uso eficiente de los recursos que dispone. Para ello, hablaremos de enrutamiento y difusión de la información en la gráfica (n, k)-estrella.

4.1. Enrutamiento

De acuerdo con Sheng-I Yeh *et al.* en [40], el enrutamiento se refiere al envío de mensajes desde un nodo fuente a un nodo destino a través de algunos nodos intermedios dentro de la red de interconexión, procurando que la trayectoria de enrutamiento sea tan corta como sea posible.

El enrutamiento es un mecanismo, realizado por software o por hardware, que dirige los mensajes entre el procesador origen y el procesador destino durante la comunicación en una red de interconexión. El modo de enrutar comprende dos aspectos esenciales: el algoritmo de enrutamiento y el control de flujo en el enrutamiento. El algoritmo de enrutamiento efectúa la elección de la trayectorias

cuando existen varias posibles y gestiona los conflictos que puedan surgir entre los mensajes que quieren tomar la misma trayectoria. Normalmente se busca un algoritmo de enrutamiento que sea óptimo, es decir, que conduzca los mensajes por la trayectoria más corta. El algoritmo de enrutamiento depende de la topología de la red. El control de flujo se refiere al modo de propagación física de la información. Este material se centrará en el modo de enrutamiento de la información.

En una red de interconexión uno de los principales problemas que encuentra el enrutamiento es el bloqueo,[1] lo que puede llegar a inutilizar la red. Es necesario elegir modos de enrutamiento y topologías que resuelvan estos conflictos antes de ejecutar una aplicación.

La topología de una red de interconexión está representada por un gráfica en la que cada nodo ocupa la posición de un procesador y cada enlace ocupa el canal de comunicación entre dos vértices cualesquiera. Cuando los vértices o aristas de una red de interconexión fallan, es decir, cuando no es posible acceder a ellos, algunas trayectorias pueden no estar disponibles y el enrutamiento o la comunicación por medio de ellas puede necesitar más conexiones o no lograr el envío de datos a sus destinos. Es en esos casos cuando se necesita un enrutamiento tolerante a f-fallos a fin de mantener el rendimiento del sistema. Basándose en la información acerca de los tipos de errores, el enrutamiento tolerante a fallos puede ser clasificado como:

- **Método global**: cada procesador en el sistema conoce la ocurrencia de todas las fallas y la mejor trayectoria posible de un procesador a otro se conoce desde el principio. Un asunto importante y no trivial es cómo conseguir la información global.

- **Método local**: cada nodo conoce sólo el estado de sus nodos vecinos y enlaces. La trayectoria de enrutamiento entre dos nodos no se conoce con antelación. Este método puede aplicarse por ensayo y error, usando el algoritmo de búsqueda a profundidad, a fin de encontrar trayectorias óptimas.

- **Método global limitado**. La información acerca de los errores ocurridos alrededor de los vecinos de un nodo es recopilada y procesada para obtener una guía de la distribución de las fallas ocurridas en el sistema. Esta guía es útil para el enrutamiento, por lo que la longitud de cualquier trayectoria puede ser calculada, caso contrario al método global que se conoce de

[1] Se define *bloqueo* como el conjunto de procesos o hilos de ejecución en un sistema concurrente que compiten por los recursos del sistema o la comunicación necesaria entre ellos.

antemano esa longitud. La recopilación de la información no es tan difícil como con el método basado en el conocimiento global de la información y tampoco hay mucha sobrecarga de mensajes ya que sólo se recopilará la información necesaria para llevar acabo el proceso.

Es necesario diseñar un algoritmo de enrutamiento simple para enviar mensajes de un vértice a otro. En realidad, es conveniente que el algoritmo produzca una trayectoria de longitud mínima entre cualesquiera dos vértices. Sin embargo, si consideramos que la tolerancia a fallos de la red (n, k)-estrella es f será necesario un algoritmo de enrutamiento tolerante a al menos f-fallos [40]. Es decir, un algoritmo que ante la presencia de f fallos pueda establecer una ruta entre todos los nodos funcionales [1].

Enrutamiento en la gráfica n-estrella

Para facilitar la comprensión de la red de interconexión (n, k)-estrella, revisaremos brevemente cómo es que la red de interconexión n-estrella lleva a cabo el envío de mensajes entre sus nodos. Como sabemos la red de interconexión n-estrella es una atractiva alternativa al hipercubo y un caso particular de la (n, k)-estrella. Esta red se compara favorablemente con el hipercubo en varios aspectos. Por ejemplo, el grado de la S_n es $n - 1$, es decir, sublogarítmica en el número de nodos de la S_n, mientras que un hipercubo con $\Theta(n!)$ nodos tiene grado $\Theta(\log n!) = n \log n$, es decir, logarítmica en el número de nodos [37].

En la gráfica n-estrella se etiqueta a cada nodo por una permutación distinta de n símbolos. Para dos nodos cualesquiera en S_n, si la etiqueta del nodo p_i se puede obtener a partir del nodo p_j mediante el intercambio del primer símbolo con otro símbolo de p_j, con $i \neq j$, entonces existe una arista que conecta a estos nodos.

Se sabe que la gráfica n-estrella es simétrica por vértices como por aristas. La simetría por vértices de la gráfica n-estrella permite que el enrutamiento entre dos nodos arbitrarios reduzca el enrutamiento de un nodo arbitrario al nodo identidad, $I_n = 123 \ldots n$. Por lo tanto, el diámetro $D(S_n)$ de S_n, que es la longitud de la trayectoria más larga de entre todas las trayectorias más cortas entre I_n y todos los nodos de S_n, resulta ser $D(n) = max\{x \in V(S_n) : d(x, I)\}$, donde $d(x, y)$ indica la distancia desde el nodo x al nodo y.

A continuación describimos un ejemplo representativo para examinar cómo enrutar dentro de la red de interconexión n-estrella. Como se ha señalado, es suficiente con generar una trayectoria desde una permutación arbitraria a la permutación identidad, en otras palabras, ordenar la permutación arbitraria hasta obtener la

permutación identidad. Por ejemplo, considere la permutación 64725831 con ocho símbolos. Emplearemos un algoritmo glotón. Observamos que el símbolo 6 en la primera posición, puede moverse a su posición correcta intercambiando 6 con 8. Esto genera la permutación 84725631. Una vez más, repetimos el procedimiento y obtenemos 14725638. Ahora estamos atrapados, pues 1 se encuentra en la posición correspondiente, la primera. En consecuencia, ahora debemos dar un paso más para mover 1 a cualquier posición no ocupada por un símbolo correcto. En este caso podríamos intercambiar 1 con 4, generando la permutación 41725638. Ahora, nuevamente siguiendo con el enfoque anterior obtenemos 21745638 y posteriormente 12745638. Una vez más, tenemos que realizar un paso adicional para colocar 1 en una posición no ocupada por el símbolo correcto. Por lo tanto, intercambiando 1 con 7 obtenemos 72145638. Volviendo de nuevo al enfoque anterior obtenemos 32145678 y como movimiento final se genera 12345678. Nótese que tomó ocho movimientos ordenar esta permutación. Esto significa que en la gráfica S_n hemos mostrado una trayectoria de longitud ocho entre 64725831 y la permutación identidad.

De este ejemplo representativo, tomemos en cuenta que sólo hay dos reglas involucradas en la búsqueda de estas trayectorias:

1) si el símbolo 1 está en la primera posición, hay que moverlo a cualquier posición no ocupada por un símbolo correcto, y

2) si el símbolo X (cualquier símbolo que no sea 1) está en la primera posición, hay que moverlo a su posición correcta.

No sólo seguir estas reglas asegurarán una trayectoria de longitud mínima, sino que también nos permitirá calcular la longitud de la trayectoria. En [1], se muestra que este algoritmo siempre encontrará la trayectoria más corta de p_i a I_n en S_n.

Sin embargo, es necesario hacer la siguiente observación para alcanzar nuestros objetivos. Recordemos que una permutación puede representarse por un conjunto de ciclos, es decir, conjuntos de símbolos cíclicamente ordenados con la propiedad de que la posición deseada por el símbolo en cuestión es la posición ocupada por el siguiente símbolo en el ciclo. La permutación 64725831, considerada anteriormente, consiste de los ciclos (681), (42), (73) y (5). Nótese que un símbolo en su posición correcta aparece en un ciclo con un único elemento.

4.1.1. Estructura de ciclo

Antes de presentar el algoritmo de enrutamiento en la red de interconexión (n,k)-estrella, primero debemos resolver el problema de enrutamiento entre cualesquiera dos nodos de la red $S_{n,k}$. Para tal fin, haremos uso de un nodo arbitrario p y el nodo identidad I_k, es decir, la permutación canónica de los símbolos $1 \leq i \leq k$. Por otro lado, será necesario definir una estructura de ciclo para cada etiqueta de cada nodo en la gráfica (n,k)-estrella similar a la ya conocida estructura cíclica de permutaciones en la gráfica n-estrella.

Dado que una permutación de n símbolos puede ser vista como un conjunto de ciclos ordenados críticamente (véase la subsubsección B.3), se establece que la posición deseada de cada símbolo en el ciclo sea la que esté ocupada por el siguiente elemento en el ciclo. Previamente, se define la estructura de la (n,k)-estrella similar a la n-estrella. Después, usaremos está representación para describir un algoritmo de enrutamiento en la gráfica (n,k)-estrella [16].

Para simplificar la exposición, afirmamos que un símbolo que pertenezca a $\langle n \rangle - \langle k \rangle$ será un símbolo externo, debido a que no pertenece a la etiqueta de nodo identidad o nodo destino. En caso contrario, un símbolo que este contendido en $\langle k \rangle$ será conocido como símbolo interno, pues pertenece al conjunto de símbolos del nodo identidad. Salvo que se indique lo contrario, usaremos C'_i para denotar un ciclo externo que posee un elemento externo y C_i para indicar un ciclo interno que posee elementos internos. Además, m'_i para representar el número de elementos en el ciclo externo C'_i, e igualmente m_i para indicar el número de elementos en el ciclo interno C_i.

Sea una permutación $p = p_1 p_2 \dots p_k$ un nodo en $S_{n,k}$, donde p_i representa el símbolo en la posición i del nodo. Si el símbolo i está en la posición correcta, es decir, $p_i = i$, entonces será invariante; ya que durante el proceso de enrutamiento éste no cambiará de posición. En lo que sigue, omitiremos todos los invariantes de la estructura de ciclo, ya que no se moverán durante el enrutamiento más corto.

Construiremos la estructura de ciclo de un nodo p como sigue:

Primero, para cada símbolo externo $x_{m'_i}$ en p se construye un ciclo externo $C'_i = (x_1, x_2, \dots, x_{m'_i})$, con la particularidad que la posición deseada x_j en p es la que está en manos de x_{j+1} para $1 \leq j \leq m'_i - 1$, tal que todos los x_j son símbolos internos. Nótese que un ciclo externo puede ser de longitud uno, consistiendo únicamente de un símbolo externo tal que la posición es requerida por un símbolo interno sin usar en p. Por otra parte, un ciclo externo no puede contener más de un símbolo externo en el ciclo, ya que no existe una transposición que los conecte (no existe un desplazamiento cíclico que los una). Adicionalmente, para el ciclo

externo C'_i definimos el símbolo deseado d_i, cuya posición deseada está en manos del primer elemento del ciclo, x_1. Cuando hayamos construido todos los ciclos externos para cada símbolos externos de p, el resto serán todos ellos símbolos internos.

Segundo, se construyen todos los ciclos internos para el resto de símbolos, siguiendo la misma construcción usada para los ciclos de la gráfica n-estrella. El ciclo interno $C_i = (x_1, x_2, \ldots, x_{m_i})$ de p, tendrá la particularidad que la posición x_{j+1} en p es deseada por x_j para $1 \leq j \leq m_i - 1$ y la posición x_1 es la deseada por x_{m_i}. Para simplificar la exposición, diremos que si existe un ciclo que contenga a p_1, elegiremos a p_1 como el primer elemento de ciclo ya que cualquier desplazamiento cíclico de la secuencia de símbolos se permite dentro de cada ciclo.

En la representación cíclica de un nodo p, los ciclos pueden estar en cualquier colocación. De tal manera que la estructura de ciclo de un nodo p con α ciclos internos y β ciclos externos puede ser representado como:

$$C(p) = C_1 C_2 \ldots C_\alpha C'_1 C'_2 \ldots C'_\beta, \text{ con } \beta \geq 0.$$

Observemos la estructura de ciclo de gráfica (n, k)-estrella a través de unos ejemplos.

• *Ejemplo.* Considere los nodos $p = 2968134$ y $q = 7369824$ en la gráfica $S_{9,7}$.

1. La estructura de ciclo del nodo p es: $C(p) = C_1 C'_1 C'_2$ donde los ciclos son $C_1 = (3, 6),\ C'_1 = (2, 9, 1)$ y $C'_2 = (4, 8)$. Los símbolos deseados de C'_1 y C'_2 son $d_1 = 5$ y $d_2 = 7$, respectivamente.
2. La representación de ciclo del nodo q es: $C(q) = C_1 C'_1 C'_2$ donde los ciclos son $C_1 = (2, 3, 6),\ C'_1 = (7, 4, 9)$ y $C'_2 = (8)$. Los símbolos deseados de C'_1 y C'_2 son $d_1 = 1$ y $d_2 = 5$, respectivamente.

▲

4.1.2. Enrutamiento en la gráfica (n, k)-estrella

El enrutamiento de un nodo arbitrario p al nodo identidad I_k puede lograrse por el movimiento de los símbolos internos a su posición correcta y el intercambio de los símbolos externos con los símbolos deseados. Es en este momento que deben corregirse los ciclos de $C(p)$ uno por uno. Sin embargo, si existe un ciclo, ya sea interno o externo, que contenga a p_1, escogeremos a p_1 como primer elemento del ciclo y será el primero en corregirse. Además, para reducir la distancia

de enrutamiento corregiremos el símbolo externo de un ciclo externo (si existe) mediante el intercambio de éste con el símbolo deseado de cualquier otro ciclo externo o el intercambio de este con el símbolo correspondiente deseado (en otro caso).

Gracias a la simetría de la gráfica $S_{n,k}$, cualquier nodo puede ser mapeado al nodo identidad por el renombre de sus símbolos. Para el enrutamiento entre cualesquiera dos nodos s y t, se define una función de renombre M, tal que el nodo destino t es mapeado en el nodo identidad, es decir, $M(t) = I_k$. A continuación, se obtiene que todas las rutas entre los nodos s y t en la gráfica original serán isomorfas a aquellas entre $M(s)$ y el nodo identidad I_k [16].

Sin pérdida de generalidad, asumiremos que el nodo identidad será el nodo destino para el problema del enrutamiento entre dos nodos de gráfica $S_{n,k}$.

Definimos el procedimiento para corregir un ciclo interno, denotado por $C_i = (x_1, x_2, \ldots, x_{m_i})$. Si $x_1 = p_1$ movemos inmediatamente x_1 a su posición correcta (que está en manos de x_2), mientras que x_2 se intercambia a la primera posición. Entonces x_2 es colocado en su posición correcta (que está en manos de x_3); seguimos este proceso hasta que $x_{m_{i-1}}$ esté colocado correctamente. Nótese que cada elemento de un ciclo, excepto el primer elemento del ciclo, se puede mover a la primera posición como resultado de la corrección de elementos previos en la representación de ciclo. Además, si $x_{m_i} = 1$, su corrección puede considerarse como resultado de la corrección de x_{m_i-1}. Si $x_1 \neq p_1$, requerirá un paso adicional para llevar a x_1 a la primera posición de la etiqueta, dado que un símbolo debe estar antes en la primera posición para llevar a éste a su posición correcta. Entonces movemos $x_1, x_2, \ldots, x_{m_i}$ a su posición correcta en el mismo orden. De esta manera, el número de pasos para corregir cada ciclo interno C_i de longitud m_i, con $1 \leq i \leq \alpha$, está dado por:

$$\begin{cases} m_i - 1 & \text{si } p_1 \in C_i, \text{ y} \\ m_i + 1 & \text{si } p_1 \notin C_i. \end{cases}$$

Por otro lado, para corregir un ciclo externo de p, movemos los símbolos internos a sus posiciones correctas empleando un argumento similar al anterior. Sin embargo, cuando el único símbolo externo está ubicado en la primera posición, lo intercambiamos con el símbolo deseado de cualquier otro ciclo externo no corregido (si existe) o en otro caso con él mismo. Más precisamente, C'_s denota la primera corrección incompleta del ciclo externo. Únicamente cuando todos los ciclos externos, aparte de C'_s, se han corregido completamente, el símbolo externo en la primera posición es intercambiado con d_s. A través de esta estrategia todos los ciclos se corrigen sucesivamente; de forma tal que los pasos adicionales para

intercambiar el primer elemento de estos ciclos distintos de C'_s a la primera posición se reducen, por lo que, el número de pasos necesarios para corregir los β ciclos externos es:

$$\begin{cases} m' + \beta - 1 & \text{si } p_1 \in C'_j \text{ para algún } j, \text{ ó} \\ m' + \beta + 1 & \text{si } p_1 \notin C'_j \ \forall j, \end{cases}$$

donde $m' = \sum_{j=1}^{\beta} m'_j$.
Mediante los siguientes ejemplos observaremos como se corrigen los ciclos.

• *Ejemplo.* Considere los nodos p y q de la gráfica $S_{9,7}$ usados anteriormente.

1. Primero corregimos los ciclos externos. Para nuestro caso los ciclos externos son: $C'_1 = (2, 9, 1)$ y $C'_2 = (4, 8)$ a lo largo de la trayectoria:

$$2968134 \rightarrow_2 9268134 \rightarrow_1 7268134 \rightarrow_7 4268137 \rightarrow_4$$
$$8264137 \rightarrow_1 5264137 \rightarrow_5 1264537.$$

   Continuando con el procedimiento, el ciclo interno $C_1 = (3, 6)$ se corrige por la trayectoria:

$$1264537 \rightarrow_6 3264517 \rightarrow_3 6234517 \rightarrow_6 1234567.$$

2. Considere la corrección del nodo $q = 7369824$. Siguiendo la estrategia anterior, lo primero que corregimos son los ciclos externos. Para nuestro caso los ciclos externos son: $C'_1 = (7, 4, 9)$ y $C'_2 = (8)$ a lo largo de la trayectoria:

$$7369824 \rightarrow_7 4369827 \rightarrow_4 9364827 \rightarrow_1 5364827 \rightarrow_5$$
$$8364527 \rightarrow_1 1364527.$$

   Continuando con el procedimiento, obtenemos los ciclos internos, en nuestro caso corresponde con: $C_1 = (3, 6)$ para ser corregido por la trayectoria:

$$1364527 \rightarrow_6 2364517 \rightarrow_2 3264517 \rightarrow_3 6234517 \rightarrow_6$$
$$1234567.$$

▲

Nótese que los subíndices nos indican la dimensión de la arista que se utilizó en la trayectoria elegida. Y los símbolos finales de cada una de las aristas nos indican en qué subgráfica inducida nos estamos moviendo, teniendo presente que existen muchas posibles representaciones de la red.

Si reiteradamente usamos una de las siguiente tres reglas llegaremos al nodo identidad I_k no importa el nodo p que escojamos de la gráfica (n, k)-estrella. Gracias a estas reglas el algoritmo de enrutamiento puede ser descrito:

R1) si el símbolo 1 se encuentra en la primera posición, intercambiamos éste con cualquier otro símbolo que no esté en su posición correcta,

R2) si el símbolo i, con $1 < i \leq k$, está en la primera posición, movemos a éste a su posición correcta o intercambiamos a éste con cualquier otro símbolo perteneciente a otro ciclo interno y,

R3) si el símbolo externo perteneciente al ciclo C'_1 se encuentra en la primera posición, entonces intercambiamos éste con el símbolo deseado d_j de cualquier otro ciclo C'_j no corregido.

De la regla R2, cuando un símbolo interno está en la primera posición, lo movemos a su posición correcta o lo intercambiamos con cualquier símbolo interno perteneciente a otro ciclo interno. Además, por la regla R3, todos los ciclos externos distintos al ciclo que contiene a p_1 pueden ser corregidos en cualquier orden. De esta manera, para todo símbolo i, $1 \leq i \leq n$, en la primera posición tendrá más de una elección para enrutarse. Esta regla es especialmente útil para el enrutamiento ante la presencia de fallos.

Siguiendo las reglas de enrutamiento antes mencionadas, todos los ciclos se corrigen consecutivamente, de tal manera que los pasos para intercambiar el primer elemento de estos ciclos aparte de C'_1 a la primera posición se reducen.

Desarrollaremos algunos ejemplos mostrando el comportamiento de los conceptos presentados.

• *Ejemplo.* Considere el nodo $p = 4635$ en la gráfica $S_{7,4}$ y el nodo identidad $I_4 = 1234$.

El nodo p enviará información al nodo destino I_4, es necesario corregir este nodo. Primero, encontramos los ciclos del nodo p. es $C(p) = C'_1C'_2$ donde los ciclos son $C'_1 = (4, 5)$ y $C'_2 = (6)$. Los símbolos deseados de cada ciclo externo son: para C'_1, $d_1 = 1$ y para C'_2, $d_2 = 2$. Observemos que no

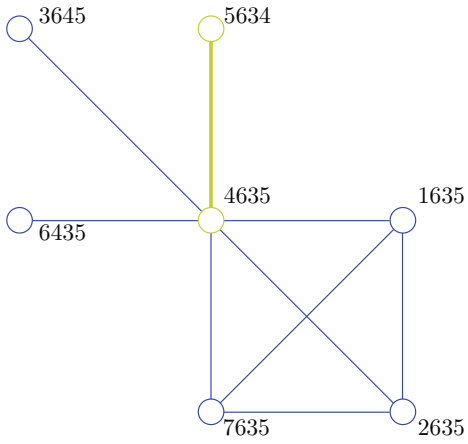


Figura 4.1: Vecinos del nodo $p = 4635$.

contamos con ciclos internos, pero sí contamos con un símbolo invariante, este durante la corrección del nodo p no se moverá, el símbolo es 3.

Obtenida ya la representación por ciclos del nodo p, corregimos primeramente los ciclos externos por medio de la trayectoria:

$$4635 \rightarrow_4 5634 \rightarrow_1 2634 \rightarrow_2 6234 \rightarrow_1 1234.$$

Por no contar con ciclos internos, el procedimiento de corrección termina. La figura 4.1 muestra los vecinos del nodo p y la primer elección de la trayectoria. ▲

Consideremos el siguiente ejemplo donde el nodo es un ciclo externo.

• *Ejemplo.* Considere el nodo $p = 314$ en la gráfica $S_{4,3}$ y el nodo identidad $I_3 = 123$.

Corregimos el nodo p. Primeramente, representamos al nodo p por medio de la estructura de ciclo; dando como resultado que: $C(p) = C'_1$ donde el ciclo es $C'_1 = (1, 3, 4)$.

El símbolo deseado del ciclo externo C'_1, es $d_1 = 2$. Observemos que no contamos con ciclos internos.

Obtenida la estructura de ciclo del nodo p, corregimos primeramente el ciclo externo por medio de la trayectoria:

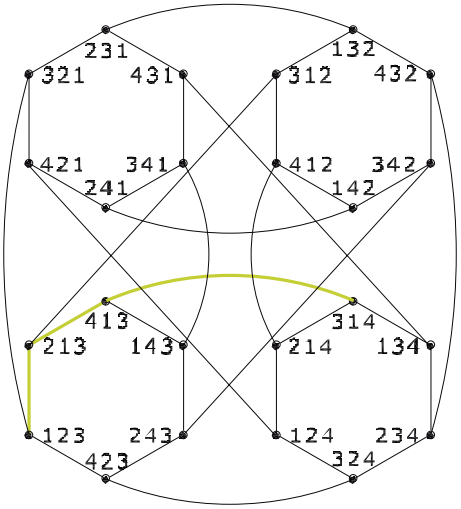


Figura 4.2: Trayectoria más corta entre $p = 314$ y $I_3 = 123$.

$$314 \rightarrow_3 413 \rightarrow_1 213 \rightarrow_2 123.$$

Por no contar con ciclos internos, el procedimiento de corrección termina. La figura 4.2 muestra la elección de la trayectoria. ▲

Enrutamiento general

Sin embargo, ¿qué sucede si deseamos enrutar el nodo p con el nodo q, con la salvedad que el nodo q no es el nodo origen I_k? En esta sección se describe el enrutamiento general de la gráfica (n, k)-estrella basándonos en [17].

Como ya sabemos, las trayectorias en la gráfica (n, k)-estrella pueden ser representadas por ciclos. Afirmamos que el nodo origen $p = p_1p_2 \dots p_k$ y el nodo $q = q_1q_2 \dots q_k$ contendrán un ciclo $C = (s_1, s_2, \dots, s_l)$, donde $C \subset \{p_i \mid 1 \leq i \leq k\}$ y l representa la longitud del ciclo. Un símbolo externo es aquel símbolo sin usar en la etiqueta del nodo destino q, en otro caso es un símbolo interno. Si todos los símbolos de C son símbolos internos de q, diremos que C es un ciclo interno y lo denotaremos con $C_i = (s_1, s_2, \dots, s_{l_i})$. Si el ciclo C contiene un símbolo externo de q diremos que es un ciclo externo. Deberá existir un símbolo deseado d del símbolo s_l, donde d pertenece a q, uno por cada ciclo externo. Denotaremos

estos ciclos C con $C'_i = (s_1 s_2 \dots s_{l_i})$. El índice i indicará la cantidad de ciclos internos o externos en p. Nótese que no existe más de un símbolo externo por ciclo, por la construcción misma de la trayectoria.

Para obtener la trayectoria más corta, la regla de enrutamiento sin fallos sigue el orden de los símbolos contenidos en los ciclos para enrutar el mensaje.

• *Ejemplo.* Suponga que desea enrutar el nodo $p = 21745$ al nodo destino $q = 14526$ contenidos en $S_{7,5}$.

Los ciclos son:

$$\begin{pmatrix} 1 & 4 & 5 & 2 & 6 \\ 2 & 1 & 7 & 4 & 5 \end{pmatrix} = C_1 C'_1 = (2, 4, 1)(5, 7)$$

El símbolo deseado $d_1 = 6$. Procedemos a enrutar el nodo p al nodo q:

$$21745 \to_4 41725 \to_2 14725 \to_5 54721 \to_3$$
$$74521 \to_1 64521 \to_5 14526.$$

Sin embargo, para este caso en la gráfica $S_{7,5}$ no es la única solución posible:

$$21745 \to_5 51742 \to_3 71542 \to_1 61542 \to_5$$
$$21546 \to_4 41526 \to_2 14526.$$

La longitud de las trayectorias encontradas es seis. ▲

Consideremos los siguientes ejemplos; nuevamente usaremos las gráficas antes analizadas:

• *Ejemplo.* Considere enrutar el nodo $p = 2635$ al nodo destino $q = 5634$ que pertenecen a la gráfica $S_{7,4}$.

Construimos la estructura de ciclo del nodo p, sin embargo, nótese que los nodos comparten dos símbolos que son invariantes durante la obtención de la trayectoria más corta. No contiene ciclos internos.

$$\begin{pmatrix} 5 & 6 & 3 & 4 \\ 2 & 6 & 3 & 5 \end{pmatrix} = C'_1 = (5, 2)$$

El símbolo deseado $d_1 = 4$.Procedemos a enrutar el nodo p al nodo q.

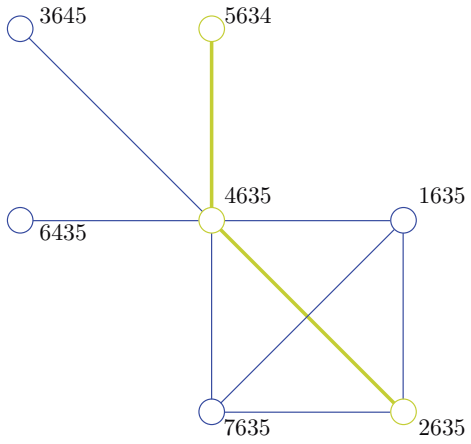


Figura 4.3: Trayectoria solución de $u = 2635$ a $v = 5634$

$$2635 \rightarrow_1 4635 \rightarrow_4 5634.$$

La longitud del ciclo C_1' es dos. La figura 4.3 muestra la trayectoria más corta entre los nodos p y q. ▲

El siguiente ejemplo posee dos posibles trayectorias como solución.

• *Ejemplo.* Sea la gráfica $S_{4,3}$ y sean el nodo origen $p = 234$ y el nodo destino $q = 321$ dos nodos contenidos en $S_{4,3}$.

Obtenemos la estructura de ciclo del nodo origen u:

$$\begin{pmatrix} 3 & 2 & 1 \\ 2 & 3 & 4 \end{pmatrix} = C_1C_1' = (2,3)(4) \text{ y } d_1 = 1$$

Primera trayectoria más corta entre los nodos p y q.

$$234 \rightarrow_2 324 \rightarrow_3 423 \rightarrow_1 123 \rightarrow_3 321.$$

Segunda trayectoria más corta entre los nodos p y q.

$$234 \rightarrow_3 432 \rightarrow_1 132 \rightarrow_3 231 \rightarrow_2 321.$$

La longitud de las trayectorias es cuatro. La figura 4.4 muestra las trayectorias solución. La primera de ellas se representa por el color azul, la segunda por el color amarillo. ▲

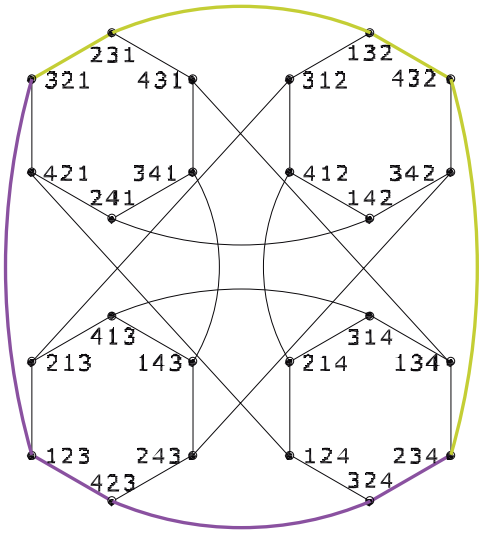


Figura 4.4: Trayectorias entre $u = 234$ y $v = 321$.

4.1.3. Distancia y diámetro en la gráfica (n, k)-estrella

Recordemos la definición de distancia.

Definición 4.1 (Distancia). La distancia $d(u, v)$ entre los vértices u y v en $V(G)$ es la longitud de la $uv-$trayectoria más corta en G.

El número de ciclos $C(p)$ se representa por c un entero, donde todos los ciclos externos se consideran como uno sólo en el recuento. Sabemos que α representa los ciclos internos de p y β los ciclos externos de p. Si $\beta > 0$, entonces $c = \alpha + 1$; si $\beta = 0$, entonces $c = \alpha$. Definimos e como el número de elementos externos de p, es decir, entonces $e = \beta$ ya que por cada símbolo externo de p se construye un ciclo externo. Supongamos que m denota el número de símbolos fuera de lugar de p con respecto al nodo identidad I_k, es decir, la cantidad de elementos en p que no ocupan la posición correcta con respecto al nodo identidad. Se establece la siguiente relación:

$$m = \Sigma_{i=1}^{\alpha} m_i + \Sigma_{j=1}^{\beta} m'_j = \Sigma_{i=1}^{\alpha} m_i + m'.$$

Así que m representa la suma de todos los elementos de cada ciclo interno más la suma de todos los elementos de cada ciclo externo, no obstante, al reducir la expresión tenemos que es igual a la suma de todos los elementos de cada ciclo interno más el número de pasos requeridos para corregir β ciclos externos.

Teorema 4.1. La distancia $d(p, I_k)$ del nodo p al nodo identidad I_k en la gráfica $S_{n,k}$ es:

$$d(p, I_k) = \begin{cases} c + m + e & \text{si } p_1 = 1, \\ c + m + e - 2 & \text{si } p_1 \neq 1. \end{cases}$$

Demostración. Los haremos por casos:

a) Si $p_1 = 1$. No existe un ciclo que contenga a p_1.
Ahora bien, si $\beta = 0$, entonces:
$d(p, I_k) = \Sigma_{i=1}^{\alpha}(m_i + 1) = \alpha + m = c + m + e$.
Si $\beta > 0$, entonces:
$d(p, I_k) = \Sigma_{i=1}^{\alpha}(m_i + 1) + \{\Sigma_{i=1}^{\beta}(m'_i + 1) + 1\} = \alpha + m + \beta + 1 = c + m + e$.

b) Si $p_1 \neq 1$. Existe un ciclo que contiene a p_1.
Ahora bien, si $\beta = 0$, entonces $p_1 \in C_j$ para algún j, y
$d(p, I_k) = (m_j - 1) + \{\Sigma_{i=1}^{\alpha}(m_i + 1) - (m_j + 1)\} =$
$\alpha + m - 2 = c + m + e - 2$.
Si $\beta > 0$ y p_1 está en C_j para algún j, entonces
$d(p, I_k) = (m_j - 1) + \{\Sigma_{i=1}^{\alpha}(m_i + 1) - (m_j + 1)\} + \{\Sigma_{i=1}^{\beta}(m'_i + 1) + 1\} =$
$m + \alpha + \beta - 1 = c + m + e - 2$.
Si $\beta > 0$ y p_1 pertenece a C'_j para algún j, se necesita de $\{\Sigma_{i=1}^{\beta}(m'_i + 1) - 1\}$ movimientos para corregir todos los ciclos externos de p. De ahí que,
$d(p, I_k) = \Sigma_{i=1}^{\alpha}(m_i + 1) + \{\Sigma_{i=1}^{\beta}(m'_i + 1) - 1\} =$
$\alpha + m + \beta - 1 = c + m + e - 2$.

□

• *Ejemplo.* Calcularemos la distancia para nuestros ejemplos antes utilizados.

- Para la gráfica $S_{7,4}$, la distancia entre el nodo $p = 4635$ y el nodo $I_4 = 1234$ está dada por:

$$d(p, I_4) = c + m + e - 2 = 1 + 3 + 2 - 2 = 4.$$

- Para la gráfica $S_{4,3}$, la distancia entre el nodo $p = 314$ y el nodo $I_3 = 123$ está dada por:

$$d(p, I_3) = c + m + e - 2 = 1 + 3 + 1 - 2 = 3.$$

▲

Corolario 4.1. La distancia $d(p, I_n)$ de un nodo p al nodo identidad I_n en la gráfica n-estrella está dada por:

$$d(p, I_n) = \begin{cases} c + m & \text{si } p_1 = 1, \\ c + m - 2 & \text{si } p_1 \neq 1. \end{cases}$$

Demostración. Basado en el lema 3.3, la distancia del nodo $p = p_1 p_2 \dots p_n$ a $I_n = 12 \dots n$ en la gráfica n-estrella S_n es equivalente a la distancia entre $p' = p_1 p_2 \dots p_{n-1}$ y $I_{n-1} = 12 \dots (n-1)$ en $S_{n,n-1}$. Se tiene que los siguientes símbolos están asociados a $p(p')$ con respecto al nodo $I_n(I_{n-1})$; $c(c')$ indica el número de ciclos, $m(m')$ indica los símbolos fuera de lugar y $e(e')$ indica los símbolos externos. Existen dos posibles escenarios:

i) Para $p_1 = 1$. Si $p_n = n$, entonces $c' = c$, $m' = m$ y $e' = 0$; por consiguiente:
$d(p, I_n) = d(p', I_{n-1}) = c + m$.
Si $p_n \neq n$, sucede que $c' = c$, $m' = m - 1$ y $e' = 1$; de este modo,
$d(p, I_n) = d(p', I_{n-1}) = c + m$.

ii) Para $p_1 \neq 1$. Si $p_n = n$, entonces $c' = c$, $m' = m$ y $e' = 0$; por consiguiente
$d(p, I_n) = d(p', I_{n-1}) = c + m - 2$.
Si $p_n \neq n$, sucede que $c' = c$, $m' = m - 1$ y $e' = 1$; de este modo,
$d(p, I_n) = d(p', I_{n-1}) = c + m - 2$.

Queda entonces demostrado. □

Parte importante del desempeño de una red de interconexión es qué tan costoso es enviar un mensaje. Una medida del desempeño es el diámetro. Nos permite darnos una idea de lo que podría pasar.

Teorema 4.2. El diámetro $D(S_{n,k})$ de $S_{n,k}$ está dado por:

$$D(S_{n,k}) = \begin{cases} 2k - 1 & \text{si } 1 \leq k \leq \lfloor \frac{n}{2} \rfloor, \\ k + \lfloor \frac{n-1}{2} \rfloor & \text{si } \lfloor \frac{n}{2} \rfloor + 1 \leq k \leq n - 1. \end{cases}$$

Demostración. Recordemos que el diámetro está dado por:

$$D(S_{n,k}) = \textit{máx}\,\{d(p, q) | p,\, q \in S_{n,k}\}$$

Consideremos los siguientes dos casos:

i) Para $1 \leq k \leq \lfloor \frac{n}{2} \rfloor$. El diámetro se obtiene de una de las siguientes dos maneras:

   i.i) $p_1 = 1$, $c = 1$, $m = k - 1$ y $e = k - 1$; y

   i.ii) $p_1 \neq 1$, $c = 1$, $m = k$ y $e = k$.

   Por lo tanto, el diámetro, $D(S_{n,k}) = 2k - 1$.

ii) Para $\lfloor \frac{n}{2} \rfloor + 1 \leq k \leq n - 1$.

   ii.i) Si n es impar, el diámetro se obtiene por medio de $p_1 = 1$, $c = (2k - n - 1)/2 + 1$, $m = k - 1$ y $e = n - k$; por consiguiente, $D(S_{n,k}) = k + \lfloor \frac{n-1}{2} \rfloor$.

   ii.ii) Si n es par, el diámetro se obtiene de las siguientes dos maneras: 1) $p_1 = 1$, $c = (2k - n)/2$, $m = k - 1$ y $e = n - k$; 2) $p_1 \neq 1$, $c = (2k - n)/2 + 1$, $m = k$ y $e = n - k$.
   Por lo tanto, $D(S_{n,k}) = k + \lfloor \frac{n-1}{2} \rfloor$.

□

Gracias a la relación de isomorfismo que existe entre las topologías (n, k)-estrella y n-estrella, podemos inferir el diámetro de la gráfica S_n.

Corolario 4.2. El diámetro $D(S_n)$ de una gráfica n-estrella es $\lfloor \frac{3(n-1)}{2} \rfloor$.

Demostración. Dado que la gráfica S_n es isomorfa a $S_{n,n+1}$, se tiene que el diámetro $D(S_n) = D(S_{n,n-1})$. □

La tabla 4.1 muestra un breve comparativo entre las redes de interconexión n-estrella y (n, k)-estrella. Examinamos las propiedades de tamaño, grado y diámetro. El tamaño indica el número de nodos en la gráfica.

4.1.4. Trayectorias disjuntas por nodos

Tanto el diámetro como la distancia entre los nodos de la red, permiten evaluar la interconexión de la red. Sin embargo, contar con trayectorias disjuntas por nodos permite interconectar los nodos de la red a pesar del posible congestionamiento o fallo de la red. Un conjunto de trayectorias serán disjuntas por nodos si los únicos nodos que comparten las trayectorias son el nodo fuente y el nodo destino. Si esto ocurre en la red de interconexión, permitirá acelerar la transferencia de grandes cantidades de datos y proporciona rutas alternativas en caso de fallos

Gráficas	**Tamaño**	**Grado**	**Diámetro**
$S_{4,2}$	12	3	3
$S_{5,2}$	20	4	3
$S_4 = S_{4,3}$	24	3	4
$S_{6,2}$	30	5	3
$S_{5,3}$	60	4	5
$S_5 = S_{5,4}$	120	4	6
$S_{6,3}$	120	5	5
$S_{6,4}$	360	5	6
$S_6 = S_{6,5}$	720	5	7
$S_{n,k}, 1 \leq k \leq \lfloor \frac{n}{2} \rfloor$	$\frac{n!}{(n-k)!}$	$n-1$	$2k-1$
$S_{n,k}, \lfloor \frac{n}{2} \rfloor + 1 \leq k \leq n-2$	$\frac{n!}{(n-k)!}$	$n-1$	$k + \lfloor \frac{n-1}{2} \rfloor$
$S_n = S_{n,n-1}$	$n!$	$n-1$	$\lfloor \frac{3n-3}{2} \rfloor$

Cuadro 4.1: Comparando S_n y $S_{n,k}$

debido a que un nodo o a un enlace fallen en la construcción de la trayectoria. Podemos decir que el número de tales trayectorias proporciona una medida de tolerancia a fallos de la red.

Sea $S_{n-1,k-1}(i)$ una subgráfica de $S_{n,k}$ inducida por todos los nodos que poseen el mismo último símbolo i en su etiqueta, para $1 \leq i \leq n$. Claramente, $S_{n-1,k-1}(i)$ y $S_{n-1,k-1}$ son isomorfas. Las subgráficas inducidas $S_{n-1,k-1}(i)$ tienen conjuntos disjuntos de vértices y, por lo tanto, forman una partición del conjunto de nodos de la $S_{n,k}$.

Revisemos el siguiente lema que relaciona la distancia entre las particiones de la gráfica $S_{n,k}$.

Lema 4.1. Dado un nodo $p \in S_{n-1,k-1}(u)$, existe un nodo $q \in S_{n-1,k-1}(v)$, para cada v, $1 \leq v \leq n$, tal que la $d(p, q) \leq 2$, donde $d(p, q)$ denota la distancia entre p y q [16].

Demostración. Consideremos el nodo $p = p_1 p_2 \dots p_{k-1} u$ en la gráfica inducida $S_{n-1,k-1}(u)$. Si $u = v$ entonces se tiene que $p = q$ y por lo cual $d(p, q) = 0$. En otro caso, si $p_1 = v$, entonces $q = u p_2 \dots p_{k-1} v$ y por lo cual $d(p, q) = 1$. Si $p_i = v$, para alguna i, $1 \leq i \leq k$, entonces $q = u p_2 \dots p_1 \dots p_{k-1} v$ y $d(p, q) = 2$, además, resulta ser una trayectoria de p a q. Por otra parte, si $p_i \neq v$ para toda i, con $1 \leq i \leq k$, entonces $q = u p_2 \dots p_{k-1} v$ y $d(p, q) = 2$

además, la trayectoria de p a q está dada por:
$p = p_1p_2 \dots p_{k-1}u \rightarrow_1 vp_2 \dots p_{k-1}u \rightarrow_k q = up_2 \dots p_{k-1}v$. □

Ahora bien, el siguiente resultado nos indica la cantidad de trayectorias disjuntas por nodos y su longitud en la red de interconexión (n, k)-estrella. Este resultado muestra el desempeño de la red de interconexión ante posibles fallas.

Teorema 4.3. Sean s y t dos nodos distintos de la gráfica $S_{n,k}$, existen $n-1$ trayectorias disjuntas por nodos entre ellas mismas y la longitud de estas trayectorias es a lo más $D(S_{n,k}) + 3$, donde $D(S_{n,k})$ es el diámetro de la red $S_{n,k}$ [16].

Demostración. Sin pérdida de generalidad, suponemos que $s \in S_{n-1,k-1}(u)$ y que $t \in S_{n-1,k-1}(v)$. Definimos la notación $s \Rightarrow t$ como las trayectorias entre los nodos s y t generadas por las reglas de enrutamiento. Se probará el teorema considerando los siguientes dos casos:

i) $u = v$, es decir, u y v se encuentran en la misma $S_{n-1,k-1}(u)$.
Haciendo uso del lema 4.3, primero encontramos trayectorias disjuntas por nodos de longitud al menos 2 de s a $s_i \in S_{n-1,k-1}(i)$, con $1 \leq i \neq u \leq n$, y las $n-1$ trayectorias disjuntas por nodos de longitud al menos 2 de t a $t_i \in S_{n-1,k-1}(i)$, con $1 \leq i \neq u \leq v$. Sea P_i denota una trayectoria, $s \Rightarrow s_i \Rightarrow t_i \Rightarrow t$, donde $s_i \Rightarrow t_i$ tiene la trayectoria más corta entre s_i y t_i. Es decir, $s_i \Rightarrow t_i$ es una trayectoria de enrutamiento en $S_{n-1,k-1}(i)$. Podemos decir que las trayectorias $P_i, 1 \leq i \neq u \leq n$, construidas sobre nodos disjuntos, tienen longitud a lo más $D(S_{n-1,k-1}) + 4 \leq D(S_{n,k}) + 3$.

ii) $u \neq v$; es decir, s y t se encuentran en diferentes gráficas inducidas $S_{n-1,k-1}$.
Por un argumento similar al de (i), podemos construir trayectorias P_i disjuntas por nodos $(n-2)$: $s \Rightarrow s_i \Rightarrow t_i \Rightarrow t$ con $i \in \langle n \rangle - \{u, v\}$. Solamente es necesario mostrar cómo se construye la otra trayectoria disjunta por nodos. Sea $s = p_1p_2 \dots p_{k-1}u$ y $t = q_1q_2 \dots q_{k-1}v$. Para aclarar, asumimos que $p_k \neq q_{k-1}$. Entonces las $n-1$ trayectorias disjuntas por nodos se construyen como:

$$P_{uv} : s \rightarrow x_1 \rightarrow x_2 \rightarrow x_3 \Rightarrow x_5 \rightarrow x_4 \rightarrow t,$$

donde el nodo x_1 tiene a v como primer símbolo en su etiqueta, el nodo x_2 está conectado con el nodo x_1 a través de una k-arista, el nodo x_3

está conectado con el nodo x_2 a través de $(k-1)$-arista, el nodo x_4 tiene a u como primer símbolo , el nodo x_5 está conectado con el nodo x_4 a través $(k-1)$-arista. Además, la trayectoria P_{uv} está contenida entre $S_{n-1,k-1}(u)$ y $S_{n-1,k-1}(v)$, por lo que P_{uv} es otra trayectoria disjunta por nodos. Nótese que ambos nodos x_3 y x_5 tiene a uv como sus dos últimos símbolos; esto es, $x_3 \Rightarrow x_5$ están contenidos en una $S_{n-2,k-2}$ como los dos últimos símbolos. Por lo cual, la longitud de la trayectoria P_{uv} es a lo más $D(S_{n-2,k-2}) + 5 \leq D(S_{n,k}) + 2$. Si $p_k = q_{k-1}$, podemos escoger cualquier i, con $2 \leq i \leq (k-2)$, y reemplazar $k-1$ en el argumento superior con i.

□

Hasta este momento hemos hablado de algunas propiedades topológicas en la red $S_{n,k}$. Sin embargo, podemos extendernos a otras. Se dice que la conexidad por nodos de una red G se define como el mínimo número de nodos que al ser removidos resulta en una gráfica disconexa o trivial. Tomando en cuenta este concepto, se define el fallo por diámetro.

Definición 4.1 (Fallo por diámetro). El fallo por diámetro D_f de una red G con conexidad $\kappa(G)$, se define como el diámetro máximo de cualquier red obtenida de remover $\kappa(G) - 1$ nodos en la red.

Los siguientes resultados se obtienen del teorema de Menger.

Corolario 4.3. La conexidad por nodos $\kappa(S_{n,k})$ de la gráfica (n, k)-estrella es $n-1$ [16].

Corolario 4.4. El fallo por diámetro $D_f(S_{n,k})$ de la gráfica (n, k)-estrella es a lo más $D(S_{n,k}) + 3$ [16].

Para una discusión sobre estos resultados puede consultarse [16].

4.2. Difusión de la información

La comunicación entre los procesadores se lleva a cabo mediante el envío de mensajes a través de los enlaces disponibles en la red de interconexión. En algunas ocasiones necesitamos que todos los procesadores conozcan la misma información, digamos el valor x. Para poder llevar a cabo esta tarea es necesario usar

la operación *difusión de la información*, que es la operación que un procesador ejecuta al enviar mensajes a otros procesadores, es decir, es un algoritmo que le provee soporte a la red de interconexión [32]. Esta operación envía un paquete a lo largo de una sola trayectoria, en lugar de transmitir simultáneamente a través de varias trayectorias [19].

Existen dos posibles esquemas de comunicación entre nodos: el esquema a un solo puerto[2] y el esquema a todo puerto[3]. El modelo a un solo puerto permite a un procesador enviar (o recibir) a lo más un *datum* de longitud fija a (o desde) uno de sus vecinos en una unidad de tiempo. Por otro parte, en una unidad de tiempo un procesador puede enviar (o recibir) un datum de longitud fija a (o desde) todos sus vecinos en el modelo a todo puerto [30].

Sin embargo, ¿cuánto le toma al algoritmo enviar la información? Como sucede en algunas redes, si el grado es constante, el tiempo requerido para la difusión de la información a todos los vecinos es constante. Minimizar esta constante en otras redes será el principal problema en cuestión.

El problema de la difusión de la información vecinal se define como el envío de un mensaje de tamaño fijo desde un nodo origen a todos sus vecinos donde en una unidad de tiempo, un nodo puede enviar o recibir exactamente de uno de sus vecinos un dato de tamaño constante [30]. Por decirlo así, se ejecuta un paso del modelo a todo puerto en un modelo a un sólo puerto.

El siguiente teorema nos indica la cota inferior del problema difusión de la información vecinal para una red de interconexión con grado d:

Teorema 4.4. Cualquier algoritmo de difusión de la información vecinal en una red de interconexión con grado d requiere $\Omega(\log d)$ de tiempo de ejecución [29].

De acuerdo con el teorema anterior podemos llegar a las siguientes conclusiones: para la red de interconexión S_n, el mejor tiempo de ejecución que podemos obtener es $\log(n-1)$. Cuando hablamos de la $S_{n,k}$, el teorema anterior nos dice que el límite inferior para el problema de difusión de la información entre los nodos de la $S_{n,k}$ también es $\Omega(\log n)$, ya que el grado de $S_{n,k}$, es $n-1$.

Por otra parte, comparado con la difusión de la información vecinal, el problema de difusión de la información implica que un vértice desea enviar un mensaje de tamaño constante a todos los vértices en la red. Para el modelo a un solo puerto, el problema de difusión de la información tiene una cota inferior de $\Omega(\log N)$, donde N es el número total de vértices en la red de interconexión.

[2]Usaremos a un solo puerto como traducción literal de *single-port*

[3]Usaremos a todo puerto como traducción literal de *all-port*

Teorema 4.5. Bajo el modelo a un solo puerto, cualquier algoritmo de difusión de la información en una gráfica con N vértices requiere tiempo de ejecución de $\Omega(\log N)$ [29].

Aplicando el teorema anterior nuevamente a las redes n-estrella y (n, k)-estrella, se obtiene las siguientes observaciones: dado que la S_n tiene $n!$ nodos en la gráfica, el algoritmo óptimo para el problema de difusión de la información en la n-estrella necesita $O(\log n!) = O(n \log n)$ de tiempo [29].

En los últimos años se han desarrollado algoritmos para la difusión de la información en la red n-estrella; la idea que subyace en estos algoritmos es usar su estructura, ya que la S_n se puede descomponer en n subgráficas inducidas S_{n-i} en $O(\log n)$ de tiempo, el nodo fuente envía la información a un nodo en cada una de las $S_{n-1}(i)$ subgráficas, donde $1 \leq i \leq n$. Ahora cada subgráfica posee un nodo con la información. Esta operación se lleva a cabo de forma recursiva para cada subgráfica.

Recientemente, el problema de difusión de la información en la red de interconexión (n, k)-estrella ha sido estudiado, y se han obtenido algoritmos de tiempo $O(nk)$. Aplicando el teorema anterior, el límite inferior para este problema en un solo puerto para la $S_{n,k}$ es $\Omega(\log(n!/(n-k)!)) = \Omega(k \log n)$.

No obstante, hasta el momento sólo hemos hablado de un modelo de difusión de la información, a saber, a un solo puerto. Hablar sobre el problema de difusión de la información en redes de interconexión bajo el modelo de todo puerto, además del tiempo de ejecución, requiere una consideración adicional: el tráfico, es decir, el número total de mensajes intercambiados. Esto significa que es deseable minimizar tanto el tiempo como el tráfico. Minimizar el tráfico en la red implica minimizar la redundancia, es decir, el número de veces que un nodo recibe la misma información. Este problema de difusión de la información ha sido considerado antes y los algoritmos cuyos tiempos de ejecución son proporcionales al diámetro de la (n, k)-estrella se han obtenido utilizando árboles de generadores [33].

Haciendo uso de un esquema general, proponemos lo siguiente: denotemos a los procesadores por $p_1, p_2, \ldots, p_n$. Para que los p_n procesadores conozcan el valor de x efectuaremos los siguientes pasos:

1) p_1 lee x y se lo comunica a p_2.

2) Paralelamente, p_1 y p_2 difunden x en p_3 y p_4 respectivamente.

3) Simultáneamente, p_1, p_2, p_3 y p_4 difunden x en p_5, p_6, p_7 y p_8 respectivamente, y así sucesivamente.

Paso 1;
lee el valor en D;
lo almacena en su propia memoria, y;
lo escribe en $A(1)$.;
Paso 2;
for *i=1* **to** *max* **do** mark i;
for $i = 0$ **to** $\log(n-1)$ **do in parallel**
  // en paralelo
  for $j = 2^i + 1$ **to** 2^{i+1} **do**
    Procesador p_j ;
    lee el valor de $A(j - 2^i)$;
    lo almacena en su propia memoria, y;
    lo escribe en $A(j)$.;
  end
end

Algoritmo 1: Algoritmo general de difusión de la información

El proceso continúa hasta que todos los procesadores conocen el valor de x. A medida que el número de procesadores que reciben x se duplica en cada paso, la difusión de x a todos los n procesadores requiere $\log n^2$ pasos. El algoritmo 1 esquemátiza lo antes dicho.

Dicho formalmente, sea D una localidad de memoria en la cual hay un dato

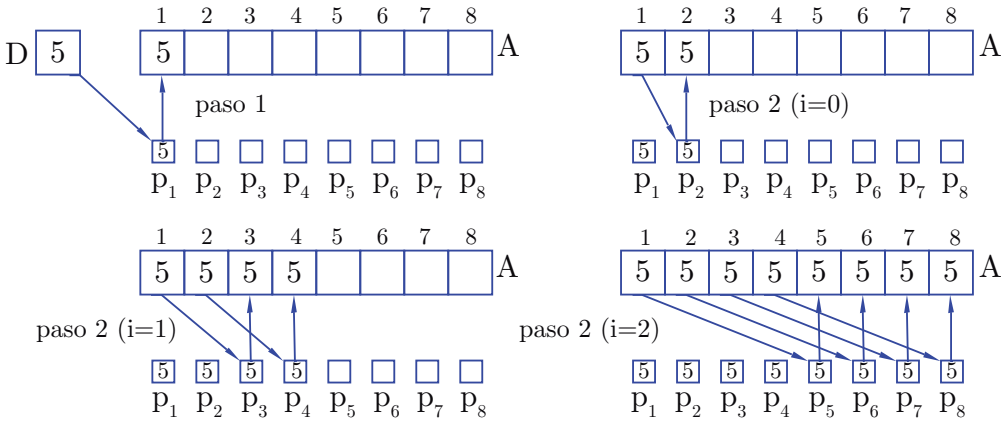


Figura 4.5: Algoritmo de difusión de la información

almacenado que los n procesadores en la red necesitan en un momento dado durante la ejecución de un algoritmo. El procedimiento supone la presencia de A un arreglo de tamaño n en memoria. El arreglo inicialmente está vacío y es empleado por el procedimiento como un espacio de trabajo para difundir el contenido de D entre los procesadores. La posición i-ésima se denota por $A(i)$.

El funcionamiento del algoritmo de difusión de la información se ilustra en la figura 4.5, para $n = 8$ y $D = 5$. Cuando el algoritmo termina, todos los procesadores han almacenado el valor de D en sus respectivas memorias locales. Dado que el número de procesadores que ha leído D se duplica en cada iteración, el procedimiento termina en $O(\log n)$ de tiempo. El espacio en memoria requiere un arreglo de longitud n.

Aunque es cierto que este algoritmo general hace el trabajo, para nuestros propósitos, en las siguientes secciones se planteará un algoritmo que haga uso de las propiedades de la red de interconexión (n, k)-estrella.

4.2.1. Bajo el modelo de a *un solo puerto*

Revisaremos la difusión información en la red (n, k)-estrella bajo el modelo de a un solo puerto en esta sección. Iniciamos revisando la difusión de la información vecinal en la gráfica $S_{n,k}$. Después, estudiamos la difusión de la información en la gráfica $S_{n,k}$.

Usaremos la notación $i*$ para representar un nodo cuyo primer símbolo es i, y análogamente, $*i$ un nodo cuyo símbolo final es i. Esta notación será útil para identificar nodos en la red y descomponer la gráfica $S_{n,k}$ en subgráficas.

Difusión de la información vecinal en $S_{n,k}$

Debido a que la red de interconexión (n, k)-estrella es simétrica por vértices, sin pérdida de generalidad, el nodo origen es el nodo identidad $I_k = 123 \ldots k$. Para nuestros fines, obtendremos para el nodo identidad I_k sus aristas vecinas. Para el nodo I_k, sus i-aristas vecinas son:

$$\begin{array}{c} 21345 \ldots k \\ 32145 \ldots k \\ 42315 \ldots k \\ \vdots \\ k2345 \ldots 1 \end{array}$$

Ahora bien, para el nodo I_k, sus 1-aristas vecinas son:

$$
\begin{array}{c}
(k+1)234\ldots k \\
(k+2)234\ldots k \\
\vdots \\
n234\ldots k
\end{array}
$$

Partiendo de este desglose general para el nodo I_k, el algoritmo se basará en las siguientes observaciones de las propiedades estructurales que posee la red (n,k)-estrella. El teorema 3.1 establece la relación existente entre las gráficas $S_{n,k}$ y K_n. Será necesario la siguiente definición.

Definición 4.2. **(Clan)** Sea $G = (V, A)$ una gráfica. W es un subconjunto de V tal que cada vértice de W es adyacente a los demás vértices de W.

Observación 4.1. Para cualquier $m \neq 1$, $S_{m,1}$ es un clan K_m; es decir, una gráfica completa de tamaño m.

Observación 4.2. En $S_{n,k}$, para cualesquiera nodos u, v y todas sus i-aristas vecinas forman un clan K_{n-k+1}.

Observación 4.3. Suponemos que $i < j$. Para cualesquiera dos i-aristas $i * k = i23\ldots(i-1)1(i+1)\ldots k$ y $j*k = j23\ldots(j-1)1(j+1)\ldots k$ vecinas del nodo identidad $I_k = 12\ldots k$ que están contenidas en el mismo ciclo de longitud 6 si ocurre que:

$$
\begin{array}{l}
123\ldots i\ldots j\ldots k \leftrightarrow \\
i23\ldots 1\ldots j\ldots k \leftrightarrow \\
j23\ldots 1\ldots i\ldots k \leftrightarrow \\
123\ldots j\ldots i\ldots k \leftrightarrow \\
i23\ldots j\ldots 1\ldots k \leftrightarrow \\
j23\ldots i\ldots 1\ldots k \leftrightarrow
\end{array}
$$

El símbolo $\leftrightarrow$ representa un enlace (arista) bidireccional entre dos nodos. El ciclo sólo contiene i-aristas. De hecho,la observación anterior también es verdadera cuando $k+1 \leq j \leq n$.

Observación 4.4. Para cualquier i-arista $i * k = i23\ldots(i-1)1(i+1)\ldots k$ y 1-arista $j * k = j23\ldots k$ incidentes al nodo identidad $I_k = 12\ldots k$, donde $k+1 \leq j \leq n$, que están contenidas en el mismo ciclo de longitud 6 si ocurre que:

$$
\begin{aligned}
&123\ldots(i-1)i(i+1)\ldots k \leftrightarrow \\
&i23\ldots(i-1)1(i+1)\ldots k \leftrightarrow \\
&j23\ldots(i-1)1(i+1)\ldots k \leftrightarrow \\
&123\ldots(i-1)j(i+1)\ldots k \leftrightarrow \\
&i23\ldots(i-1)j(i+1)\ldots k \leftrightarrow \\
&j23\ldots(i-1)i(i+1)\ldots k \leftrightarrow
\end{aligned}
$$

Además, el ciclo antes descrito contiene 1-aristas e i-aristas.

Observación 4.5. Si construimos dos ciclos por medio de las observaciones 4.3 y 4.4 (uno de cada uno) con distintas aristas $2 \leq i_1, j_1, i_2, j_2 \leq n$, serán disjuntos excepto en que compartirán el nodo identidad.

Note que las observaciones 4.3, 4.4 y 4.5 permiten dimensionar el nodo identidad junto con los $n-1$ vecinos como una gráfica completa en el sentido de que dos nodos estarán conectados por una trayectoria de longitud constante.

Sobre la base de las observaciones anteriores y por cada paso que demos duplicamos el número de vecinos con el mensaje. Partiendo de esta idea, usaremos un conjunto de ciclos disjuntos de tamaño constante para dar un algoritmo simple de difusión de la información vecinal en la red $S_{n,k}$.

Inicialmente, el nodo fuente es el único con el mensaje. En un solo paso, envía el mensaje a través del enlace directo a uno de sus vecinos. Ahora, dos nodos tienen el mensaje y, turnándose, envían el mensaje a otros dos vecinos del nodo fuente de tal manera que el nodo fuente envía su mensaje a un vecino en un solo paso y el vecino que acaba de recibir el mensaje en el paso anterior envía el mensaje a otro vecino del nodo fuente a través de una trayectoria de longitud cuatro que está contenida en un ciclo de longitud seis. El número de nodos con el mensaje es ahora cuatro (el nodo fuente y tres de sus vecinos) además, estos cuatro nodos envían el mensaje a otros cuatro vecinos del nodo fuente de la misma manera. Es decir, los tres vecinos del nodo fuente junto con el mensaje envían el mensaje a otros tres vecinos del nodo fuente por trayectorias disjuntas de longitud cuatro que están contenidas en tres ciclos disjuntos de longitud seis y el nodo fuente envía su mensaje a un vecino directamente. El algoritmo continúa hasta que todos los vecinos del nodo de origen reciben el mensaje.

Una posible implementación se da en el algoritmo 2, suponiendo que los vecinos de nodo identidad están ordenados de tal manera que $213\ldots k$ es el primero, $321\ldots k$ es el segundo, y así sucesivamente.

```
Difusión(n);
N = 1 ; // El número actual de nodos con el mensaje
for i = 0 to ⌈log(n/2)⌉ do
    if 1 ≤ n − N ≤ 3 then
        OrigenEnvíaM;
        stop;
    else in parallel
        NodoReenvíaM;
        N ← 2 ∗ N;
    end
end
```

Algoritmo 2: Difusión(n)

El proceso `OrigenEnvíaM`, es la acción que el nodo origen ejecuta al enviar a los nodos que no han recibido el mensaje por medio de un enlace directo, es decir, los nodos $2^i + j$, $1 \leq j \leq n - N$.

Posteriormente, se ejecuta el proceso `NodoEnvíaM`, esta acción indica que cada nodo u que posea el mensaje lo envíe al nodo $u + 2^i$. Si el nodo $u + 2^i$ existe se le asigna a $N \leftarrow 2 * N$. Sin embargo, hemos de decir que el nodo fuente efectúa esto a través de sus enlaces directos, mientras los demás, por trayectorias de la forma: $u* \rightarrow (u+2^i)* \rightarrow 1* \rightarrow u* \rightarrow (u+2^i)*$ (un vecino del nodo fuente) y de longitud 4 contenidas en ciclos disjuntos.

Desde un punto de vista práctico, muchas otras implementaciones también son posibles desde el nodo fuente y sus vecinos partiendo del supuesto que se tiene una gráfica completa.

Este algoritmo funciona correctamente debido a que los ciclos de longitud seis utilizados en el enrutamiento son disjuntos (excepto en el nodo fuente). En cuanto al tiempo de ejecución, consideramos en primer lugar el caso en que $n \mod 2^{\lfloor \log n \rfloor} > 3$. En este caso, se necesitan $\lceil \log n \rceil$ pasos, donde cada paso requiere un enrutamiento de longitud cuatro, excepto para el primer paso en el que el nodo fuente envía su mensaje directamente al nodo 2. Así,

$$\begin{aligned} t(n) &= 4\lceil \log n \rceil - 3 \\ &= 4\lceil \log n \rceil - 4\log 2 + 1 \\ &= 4\lceil \log(n/2) \rceil + 1. \end{aligned}$$

El análisis del otro caso es similar, por lo que el tiempo de ejecución es:

$$t(n) = \begin{cases} 4\lfloor \log(n/2) \rfloor + 1 + x & \text{si } 1 \leq x = n \mod 2^{\lfloor \log n \rfloor} \leq 3, \\ 4\lceil \log(n/2) \rceil + 1 & \text{en otro caso.} \end{cases}$$

Lo cual es $O(\log n)$.

Note que cuando n es relativamente pequeño, es mejor, por simplicidad, que el nodo fuente envíe su mensaje a cada uno de sus $n-1$ vecinos en cada unidad de tiempo, requiriendo $n-1$ pasos para alcanzar el objetivo.

Difusión de la información en $S_{n,k}$

El problema de la difusión de la información ha sido estudiado anteriormente para la red de interconexión (n, k)-estrella, donde se han obtenido algoritmos con tiempo de ejecución $O(kn)$ [30].

Con el algoritmo que se propone para la difusión de la información vecinal antes expuesto, la difusión de la información en la (n, k)-estrella se puede hacer fácilmente. Una vez más, sin pérdida de generalidad, se supone que el nodo identidad I_k requiere transmitir un mensaje. En los siguientes párrafos se describe el funcionamiento del algoritmo.

Dado que k es igual a 1 y que en general estamos ante la presencia de un ciclo, ejecutamos el algoritmo `AlgoritmoGDI`; esta acción indica la necesidad de ejecutar un algoritmo general de difusión de la información, para ello, podemos hacer uso del algoritmo 1 visto anteriormente.

En caso contrario, dado que la red no es un ciclo, haremos uso de una de las propiedades topológicas de la (n, k)-estrella, a saber, su estructura jerárquica. Procederemos de la siguiente manera. Definimos un nuevo procedimiento, `OrigenEnviaMV`, cuya acción a realizar consiste de enviar el mensaje del nodo fuente a los vecinos de este. Es necesario recordar, que los vecinos de un nodo cualquiera en la red (n, k)-estrella están enlazados por medio de 1-aristas e i-aristas. Por medio de esos enlaces el procedimiento `OrigenEnviaMV` se comunicará con los nodos vecinos del nodo origen o fuente. Estos nodos son de la forma: $2 * k,\ 3 * k,\ \ldots,\ (k-1) * k$, y $k * 1$ (todas las i-aristas vecinas, con $2 \leq i \leq k$) y de la forma $(k+1) * k,\ (k+2) * k,\ \ldots$, y $n * k$ (todas las 1-aristas). Al terminar el proceso todos estos nodos poseen el mensaje. Ahora, mediante el procedimiento `VecinosReenvianMV`, los nodos que ya poseen el mensaje reenvían el mensaje (excepto $k*1$) a sus nodos vecinos de dimensión k, es decir: $*2,\ *3,\ \ldots,\ *n$, de tal manera que todas las subgráficas $S_{n-1,k-1}(i)$ tienen un nodo con el mensaje.

En este momento volvemos a ejecutar el algoritmo en cada una de las subgráficas inducidas, hasta abarcar toda la red. Una posible implementación se describe en el algoritmo 3.

```
Difusión(S_{n,k});
if k = 1 then // es un ciclo
    AlgoritmoGDI;
else en paralelo
    OrigenEnviaMV;
    VecinosReenvianMV;
    En paralelo for 1 ≤ i ≤ n do
        Difusión(S_{n-1,k-1}(i));
```

Algoritmo 3: Difusión($S_{n,k}$)

Analizaremos el tiempo de ejecución de este algoritmo.

Sea $t(n,k)$ el tiempo de ejecución para la difusión de la información en la (n,k)-estrella, entonces obtenemos $t(n)$ de la siguiente manera:

$$\begin{aligned}
t(n,k) &= C\log n + t(n-1,k-1)\\
&= C\log n + C\log(n-1) + t(n-2,k-2)\\
&\vdots\\
&= C\log n + C\log(n-1) + \ldots + C\log(n-k+2) + t(n-k+1,1)\\
&= C\log n + C\log(n-1) + \ldots + C\log(n-k+2)\\
&\quad + C_1\log(n-k+1)\\
&= O(\log(n!/(n-k)!))\\
&= O(k\log n),
\end{aligned}$$

que es óptima en vista del límite inferior $\Omega(\log(n!/(n-k)!))$.

La clave para el algoritmo de difusión de la información es el algoritmo de difusión de la información vecinal que primero envía el mensaje a $n-1$ vecinos del nodo fuente que son de la forma $2*k$, $3*k$, $\ldots$, $(k-1)*k$, $k*1$, $(k+1)*k$, $(k+2)*k$, $\ldots$, y $n*k$. La idea es que en lugar de empezar con los vecinos del nodo fuente, estos $n-1$ nodos pertenecen a un árbol binario con raíz en el nodo

fuente. También vale la pena señalar que existe un árbol binomial con raíz en el nodo fuente (para cualquier nodo, debido a la simetría por vértices de la gráfica n-estrella) que contiene nodos de estas formas [30].

4.2.2. Bajo el modelo a *todo puerto*

La difusión de la información mediante el modelo a todo puerto en la red de interconexión (n, k)-estrella ya ha sido considerada y se han desarrollado algoritmos óptimos cuyos tiempos de ejecución son proporcionales al diámetro de la red; esto se logra utilizando árboles generadores. Presentamos un enfoque distinto para este problema. Estableceremos el mínimo conjunto dominante en la red de interconexión para transmitir el mensaje a todos los nodos de manera que ningún nodo reciba el mismo mensaje en más de una ocasión. El tiempo de ejecución es $O(k)$, que es óptimo y, sin duda, práctico.

Para nuestros propósitos, es necesario definir qué es un conjunto dominante.

Definición 4.3. **(Conjunto dominante)** Sea $G = (V, A)$ una gráfica. Un conjunto dominante de vértices en G, es un conjunto $V' \subseteq V$ tal que cada vértice de G pertenece a V' o tiene un vecino en V'.

Hacemos la siguiente observación:

Sea $D_{n,k}$ el mínimo conjunto dominante de $S_{n,k}$. Dado que la gráfica es una gráfica regular de grado $n-1$, y cada vértice de un mínimo conjunto dominante domina en sí mismo y hasta $n-1$ de sus vecinos, tenemos $|D_{n,k}| \geq (n!/(n-k)!)/n$, es decir, $D_{n,k}$ contiene al menos $(n-1)/(n-k)!$ vértices.

Sea D el conjunto de todos los nodos de la forma $i*$. Claramente, cualquier nodo en la gráfica $S_{n,k}$ es adyacente a un nodo de esta forma. Además, el número de nodos de esta forma es $(n-1)!/((n-1)-(k-1))! = (n-1)!/(n-k)!$ Por lo tanto, D es el mínimo conjunto dominante de la $S_{n,k}$.

Con estas ideas presentes, podemos diseñar un algoritmo de difusión de la información simple bajo el esquema a todo puerto de la $S_{n,k}$ obteniendo el mínimo conjunto dominante de la red. El algoritmo 4 muestra esta opción.

El algoritmo se desempeña de la siguiente manera:

Primeramente, como casos particulares de la ejecución, se define el procedimiento `FuenteEnviaMDos` cuya acción consiste en que el nodo fuente envíe el mensaje a su vecino mediante la arista de dimensión dos, ya que n es igual a 2. En otro caso, si nos encontramos ante una gráfica completa, se define el procedimiento `FuenteEnviaMUno`, el cual mediante los enlaces de dimensión uno envían el mensaje a todos sus vecinos.

```
Algoritmo Difusión(S_{n,k});
if n = 2 then
    FuenteEnviaMDos;
else if k = 1 then
    FuenteEnviaMUno;
else
    Difusión(S_{n-1,k-1}(k));
    NodoEnviaMK;
    NodoEnviaMT;
end
```

Algoritmo 4: Algoritmo de difusión ($S_{n,k}$)

Segundo, la difusión de la información tomará una subgráfica inducida por la dimensión k-esima. Por medio de ella, el procedimiento `NodoEnviaMK` logrará que cada nodo de la forma $*k$ perteneciente $S_{n,k}(k)$ envíe su mensaje a su $k*$ vecino a través de la k dimensión. Observamos por tanto, que el conjunto de nodos de la forma $k*$ es un mínimo conjunto dominante de la red de interconexión. Finalmente, cada nodo en el conjunto dominante envía su mensaje a través de todas las dimensiones, excepto la k-ésima dimensión.

Es fácil observar que mediante el uso del algoritmo, cada nodo recibe el mensaje exactamente una vez, por lo tanto no hay redundancia de mensajes. En cuanto al tiempo de ejecución, sea $t(n,k)$ es el tiempo necesario para difundir la información en la red $S_{n,k}$, entonces tenemos:

$$t(n,k) = \begin{cases} 1 & n = 2 \text{ o } k = 1 \\ t(n-1,k-1)+2 & \text{en otro caso.} \end{cases}$$

Resolviendo nos da que $t(n,k) = 2k = O(k)$.

Conclusiones

En esta tesis se ha estudiado la red de interconexión (n,k)-estrella, que es una generalización interesante de la red n-estrella, hemos analizado propiedades topológicas y algorítmicas convenientes de la gráfica y mostrado algoritmos paralelos que podrían funcionar en esta red. La red de interconexión n-estrella principió como una elección competitiva al d-cubo, y en el presente trabajo se mostró que la (n,k)-estrella conserva las mismas propiedades irresistibles de su predecesora. La mayor parte de la discusión se basó en los artículos escritos por *Wei-Kuo Chiang* y *Rong-Jaye Cheng* titulados *The* (n,k)*-star graph: a generalized star graph* [15] y *Topological properties of the* (n,k)*-star graph* [16]; sin dejar de mencionar los siguientes artículos que aportaron material significativo: [1, 14, 30].

En concreto, hemos examinado los siguientes puntos:

- los conceptos y las topologías básicas sobre redes de interconexión y un breve comparativo entre éstas,
- la estructura jerárquica de la gráfica (n,k)-estrella. Esta propiedad le permite descomponerse en n subgráficas $S_{n-1,k-1}(i)$, $1 \leq i \leq n$ y cada subgráfica $S_{n-1,k-1}(i)$ es isomorfa a $S_{n-1,k-1}$. Y no sólo eso, sino que existen $k-1$ maneras diferentes de descomponer a una gráfica $S_{n,k}$ en n subgráficas $S_{n-1,k-1}$ ajenas por nodos; para $2 \leq i \leq k$ y $1 \leq j \leq n$ tales subgráficas son: $S^i_{n-1,k-1}(j)$,
- el isomorfismo de las gráficas n-estrella y (n,k)-estrella. Además se enfatizó que la gráfica (n,k)-estrella $S_{n,n-1}$ es isomorfa a la gráfica n-estrella S_n y la gráfica $S_{n,1}$ es isomorfa a la gráfica completa K_n,
- la simetría de la gráfica (n,k)-estrella. Afirmamos que la gráfica (n,k)-estrella es simétrica por nodos y, sin embargo, no es simétrica por aristas. No

obstante, la gráfica es simétrica con 1-aristas e i-aristas,

- la tolerancia a fallos de la gráfica (n, k)-estrella. Descubrimos que la gráfica es tolerante a fallos. En concreto, la tolerancia a fallos de la gráfica (n, k)-estrella es $n - 2$,

- la estructura de ciclo en la gráfica (n, k)-estrella. El diseño de esta estructura permite desarrollar un algoritmo de enrutamiento en la gráfica,

- la distancia y el diámetro de la red. La distancia $d(p, I_k)$ de un nodo p al nodo identidad I_k en la gráfica (n, k)-estrella está dada por:

$$d(p, I_k) = \begin{cases} c + m & \text{si } p_1 = 1, \\ c + m - 2 & \text{si } p_1 \neq 1, \end{cases}$$

  y el diámetro $D(S_{n,k})$ de $S_{n,k}$ está dado por:

$$D(S_{n,k}) = \begin{cases} 2k - 1 & \text{si } 1 \leq k \leq \lfloor \frac{n}{2} \rfloor, \\ k + \lfloor \frac{n-1}{2} \rfloor & \text{si } \lfloor \frac{n}{2} \rfloor + 1 \leq k \leq n - 1, \end{cases}$$

- la descomposición de la gráfica (n, k)-estrella en trayectorias disjuntas por nodos. En particular, la gráfica $S_{n,k}$ se descompone en $n-1$ trayectorias disjuntas por vértices y cada trayectoria tiene longitud conocida de $D(S_{n,k})+3$,

- el conjunto dominante. Se describió cómo generar el mínimo conjunto dominante de la gráfica $S_{n,k}$,

- el desarrollo de algoritmos de enrutamiento y comunicación en la red, en particular hablamos de difusión de la información vecinal. Tal algoritmo se aplica únicamente bajo el modelo a un solo puerto. Como sabemos, los algoritmos tienen complejidad computacional óptima,

- el diseño de un algoritmo de difusión de la información bajo el modelo a todo puerto. Este algoritmo se basa en encontrar un mínimo conjunto dominante en la red (n, k)-estrella.

Chiang y Cheng en [15], mostraron que con respecto al costo (costo es igual al diámetro por el grado de un vértice), la (n, k)-estrella es superior a una gráfica arreglo. Además, como sucede con otras redes de interconexión, la $S_{n,k}$ puede crecer exponencialmente en sus vértices (con respecto a n y k).

A través del análisis hecho, hemos supuesto que en una unidad de tiempo, un procesador puede enviar o recibir un dato desde o hacia uno y sólo uno de sus vecinos bajo el modelo a un solo puerto. De la misma manera, hemos dicho que un procesador puede enviar o recibir un dato hacia o desde todos sus vecinos bajo el modelo a todo puerto. Tomando en cuenta esto, podemos proponer una clasificación de la red (n, k)-estrella en dos familias de acuerdo al esquema de difusión de la información que emplee:

1. que en una unidad de tiempo, un procesador que pertenezca a la red $S_{n,k}$, sólo pueda transmitir como máximo un mensaje de longitud fija, y

2. que en una unidad de tiempo, un procesador puede transmitir un mensaje de longitud arbitraria a uno o desde todos sus vecinos en la red $S_{n,k}$.

En el capítulo 4, se estudió cómo difundir la información bajo los modelos a un solo puerto y a todo puerto dentro de la (n, k)-estrella. Con la excepción del algoritmo de difusión que utiliza un mínimo conjunto dominante para alcanzar la meta, los algoritmos están diseñados para que la gráfica $S_{n,k}$ se ejecute bajo el modelo a un solo puerto.

En esta tesis sólo hemos estudiado algunos de los problemas de la red de interconexión (n, k)-estrella, muchos de los problemas relacionados con esta topología permanecen abiertos. Algunos de estos problemas se describen a continuación:

- Queda mucho por investigar sobre cómo la red de interconexión (n, k)-estrella puede adaptarse a otras redes. El concepto clave es *encaje*. Si podemos hacer que otra red de interconexión, ampliamente aceptada, pueda encajar dentro de la (n, k)-estrella, tanto sus propiedades topológicas y algorítmicas, ya desarrolladas, robustecerían esta red. Algunas ejemplos de redes conocidas son: árboles binarios completos, hipercubos; y así sucesivamente.

- Un problema es mejorar el algoritmo de ordenación en la gráfica $S_{n,k}$. Por ejemplo, el algoritmo de ordenación en n-estrella [3] alcanza mejor complejidad de tiempo que los algoritmos desarrollados hasta ahora para la (n, k)-estrella.

Finalmente, como se muestra en esta tesis, así como en otros trabajos de investigación acerca de la red (n, k)-estrella, es sabido que esta red es una alternativa atrayente a la red de interconexión n-estrella. Sin embargo, aún existen algunos algoritmos importantes en la gráfica (n, k)-estrella cuyas actuaciones no se corresponden con los algoritmos ya desarrollados para la gráfica n-estrella. A pesar

de ello, las investigaciones recientes sobre la red de interconexión (n, k)-estrella, exhiben que es una elección competitiva sobre su predecesora; razón por la cual capta la atención por sus robustas propiedades topológicas y algorítmicas.

Apéndice A

Elementos de teoría de gráficas

Presentamos conceptos básicos de Teoría de Gráficas utilizados en la exposición de este texto.

Definición A.1 (Gráfica). Una gráfica $G = (V, A)$ consiste de un par de conjuntos: $V(G)$ y $A(G)$. Se tiene que V es un conjunto finito de *vértices* (o *nodos*) de la gráfica y $A(G)$ un conjunto de pares de vértices, llamados *aristas* (o *enlaces*) de G.

Definición A.2 (Adyacencia). Si $a = (u, v)$ es una arista en G se dice que los vértices u y v son *adyacentes*, o bien, que a *incide* en u y que a incide en v; o bien que u y v son *vecinos*.

Para el caso de las aristas, se tiene que dos aristas serán *adyacentes* al incidir en un mismo vértice.

El número de vértices de una gráfica se denomina *orden* y se denota $|V(G)|$. El número de aristas en G se denomina *tamaño* usando la notación $|A(G)|$. Acorde a su orden una gráfica será finita o infinita. En nuestro caso hablaremos de gráficas finitas. La *vecindad* V_G de un vértice v en G es el conjunto de todos los vértices adyacentes a él, es decir: $V_G(v) = \{u \in V(G) \mid (u, v) \in A(G)\}$.

• *Ejemplo.* La figura A.1 representa la gráfica G cuyos atributos son:

$$V(G) = \{1, 2, 3, 4, 5, 6, 7\}, \text{ los vértices de } G,$$

$$A(G) = \{(1, 5), (2, 5), (3, 4), (5, 7)\}, \text{ las aristas de } G,$$

$$a = (1, 5) \text{ indica que los vértices 1 y 5 están conectados por } a,$$

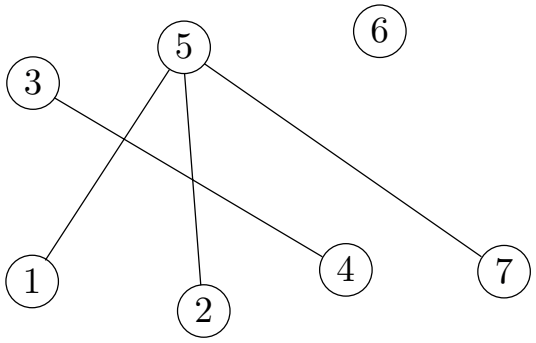


Figura A.1: Gráfica G

$$|V(G)| = 7,\ |A(G)| = 4,\ \text{el orden y tamaño de } G \text{ respectivamente},$$
$$N_G(5) = \{1, 2, 7\} \text{ es la vecindad de } 5.$$

▲

Hablaremos de algunas gráficas que poseen propiedades especiales. Una gráfica *nula* es aquella que no posee vértices ni aristas. Una gráfica *vacía* es aquella que no posee aristas. Una gráfica *trivial* es la que tiene un vértice y ninguna arista. Una gráfica *completa* es aquella en la que para cualesquiera par de vértices $u,\ v \in V(G)$ distintos están unidos por una arista, se denota por K_n. Se tiene que el *complemento,* denotado por G^c, de una gráfica G es una gráfica tal que: $V(G) = V(G^c)$, si a es adyacente a b en G entonces a no será adyacente a b en G^c, y si a no es adyacente a b en G entonces a sí será adyacente a b en G^c, nótese que $G \cup G^c = K_n$. Afirmamos que dos gráficas G_1 y G_2 serán *disjuntas* si no tienen un vértice en común y serán disjuntas por aristas si no poseen una arista en común. En la figura A.1 tenemos la gráfica G, afirmamos que la arista $(3, 4)$ es disjunta del vértice 6 si cada una de ellas fuera una gráfica.

A continuación exponemos algunas definiciones utilizadas frecuentemente en el presente trabajo.

Definición A.3 (Grado). Sea $G = (V, A)$ una gráfica y $v \in V(G)$. El grado de v, denotado por $\delta_G(v)$, es el número de aristas de G que inciden en v [8].

El **grado mínimo** de G se define como:

$$\delta(G) = \text{mín}\{\delta(v) \mid v \in V(G)\}.$$

El **grado máximo** de G se define como:

$$\Delta(G) = \text{máx}\{\delta(v) \mid v \in V(G)\}.$$

El grado de interconexión de una red es el grado máximo de la red. Para la gráfica G de la figura A.1 el grado $\delta_G(4) = 1$, el grado mínimo es $\delta(G) = 0$ y el grado máximo $\Delta(G) = 3$.

Definición A.4 (k-regular). G es k-regular, si para todo vértice v de G se cumple que $\delta_G(v) = k$.

Definición A.5 (Subgráfica). Se tiene que $G' = (V', A')$ es subgráfica de $G = (V, A)$ si $V' \subset V$ y $A' \subset A$ y G' es por sí misma una gráfica. Se denota como $G' \subset G$, es decir, G contiene a G'.

Si $G' \subseteq G$ y G' contiene todas las aristas $a = (x, y) \in A(G)$ con $x,\ y \in V'$, entonces G' es una subgráfica inducida de G; se dice que V' induce o genera G' en G y se escribe $G[V']$.

Si $U \subseteq V(G)$ y es cualquier conjunto de vértices entonces $G[U]$ representa la gráfica en U cuyas aristas son aristas de G con ambos extremos en U.

Sea R un subconjunto no vacío de $A(G)$. La subgráfica de G cuyo conjunto de vértices es el conjunto de aristas cuyos extremos están en R y cuyo conjunto de aristas es R se conoce como la subgráfica G inducida por R y se representa por $G[U]$.

Definición A.6 (Gráfica bipartita). Se tiene que una gráfica es bipartita si el conjunto de vértices G, puede partirse en dos conjuntos V_1, V_2 (con $V(G) = V_1 \cup V_2$ y $V_1 \cap V_2 = \emptyset$)tales que cada arista de G tiene un extremo en V_1 y el otro en V_2, de modo que los vértices de la misma partición no pueden ser adyacentes [42].

Es fácil verificar que la gráfica G de nuestros ejemplos es bipartita. Véase la figura A.2. Con el color rojo y azul señalamos las particiones de la gráfica.

Definición A.7 (Camino). Sea G una gráfica. Un *camino* es una secuencia finita no vacía de vértices y aristas $\mathcal{C} = v_0a_1v_1a_2v_2 \ldots a_kv_k$ tal que para $1 \leq i \leq k$ los extremos de a_i serán los vértices v_{i-1} y v_i. Un uv-*camino* es un camino del vértice u al vértice v.

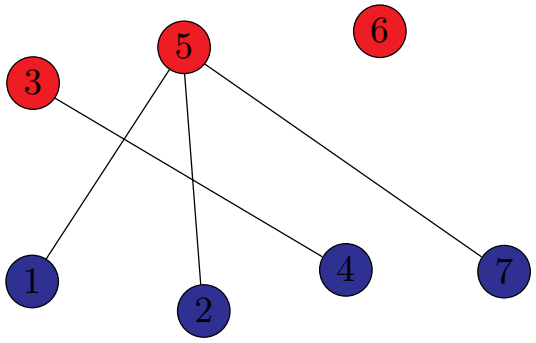


Figura A.2: Gráfica bipartita

El entero k es la *longitud* del camino $\mathcal{C}$, denotado por $l(\mathcal{C})$. Es decir, la longitud de un uv-camino $\mathcal{C}$ es igual a la cardinalidad de las aristas que posee. Por otro lado, se tiene que un paseo en G es un camino que no repite aristas.

Definición A.8 (Trayectoria). Una trayectoria es un camino que no repite vértices en $\mathcal{C}$, es decir, $v_i \neq v_j$ con $i \neq j$. Más aún, si existe una trayectoria entre los vértices u, v tendremos una uv-trayectoria.

Definición A.9 (Ciclo). Un ciclo C es una trayectoria $v_0v_1v_2 \dots v_l$ tal que la $l(\mathcal{C}) \geq 3$ y $v_0 = v_l$ y los vértices v_i, con $0 < i < l$, son distintos unos de otros.

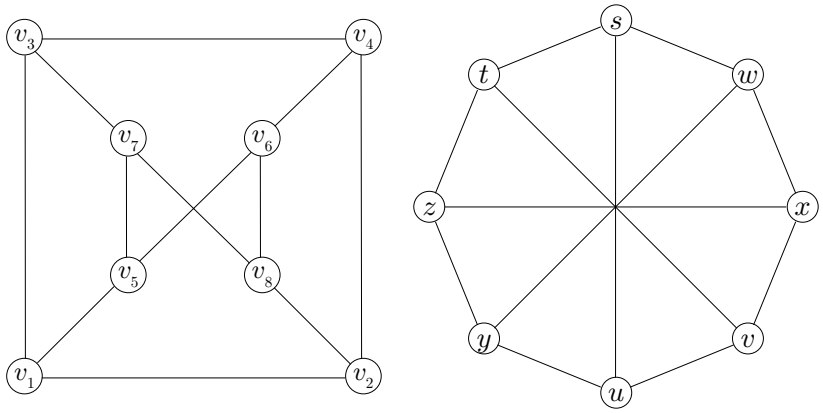
Definición A.10 (Diferencia). Suponemos que U es un conjunto no vacío de $V(G)$. La subgráfica inducida $G[V \setminus U]$ representado por $G - U$ es la subgráfica que se obtiene de G mediante la eliminación de los vértices en U junto con sus aristas incidentes.

Suponemos que R es un conjunto no vacío de $A(G)$. La subgráfica inducida de G con conjunto de aristas $G[V \setminus R]$ representado por $G - R$ es la subgráfica que se obtiene de G mediante la eliminación de las aristas de R.

Teorema A.1 (Menger)**.** Sea $G = (V, A)$ una gráfica y $A,\ B \subset V$. Se tiene que el mínimo número de vértices que separan A de B en G es igual al número máximo de trayectorias disjuntas $A - B$ en G.

Definición A.11 (Conexa). G es conexa si para cualesquiera dos vértices distintos u, v existe una uv-trayectoria en G. En otro caso, es disconexa.

Definición A.12 (Conexidad por vértices). Afirmamos que G es k-conexa para $k \in \mathbb{N}$ si $|V(G)| > k$ y $G - X$ es conexa para cualquier conjunto $X \subset V(G)$, tal que $|V(X)| < k$. En otras palabras, no hay dos vértices de G que estén separados por menos de k de otros vértices.



(a) Gráfica G (b) Gráfica H

Figura A.3: Gráficas isomorfas por medio de θ

Podemos decir que la conexidad por vértices o simplemente conexidad de G, denotado por $\kappa(G)$, es el mínimo número de vértices tal que al removerlos resulta en una gráfica disconexa o trivial.

Definición A.13 (Conexidad por aristas). Sea G una gráfica. Si $|V(G)| > 1$ y $G - F$ es conexa para cada conjunto $F \subset A(G)$ de menos de l aristas, entonces G es llamada l-conexa por aristas.

El entero mayor l tal que G es l-conexa es la conexidad por arista $\lambda(G)$ de G, se tiene que es el mínimo número de aristas que, al quitarlas de G, resulta en una gráfica disconexa o trivial.

Definición A.14 (Diámetro). El diámetro de una gráfica G se define como las distancia máxima entre todo par de vértices en G [8].

Definición A.15 (Isomorfismo). Se tiene que G es isomorfa a H si existe una función biyectiva $f : V(G) \to V(H)$ tal que para $\{u, v\} \subset V(G)$, la arista $uv \in A(G) \Leftrightarrow f(u)f(v) \in A(H)$ [22].

Claramente las gráficas isomorfas tendrán el mismo orden y tamaño. Incluso se pueden agregar tantas propiedades como se desee.

• *Ejemplo.* Observe las figuras A.3a y A.3b; se tiene que son isomorfas pues se establece por la biyección dada por θ:

$$\begin{aligned} \theta(v_1) &= s & \theta(v_5) &= w \\ \theta(v_2) &= t & \theta(v_6) &= x \\ \theta(v_3) &= u & \theta(v_7) &= y \\ \theta(v_4) &= v & \theta(v_8) &= z \end{aligned}$$

▲

Definición A.16 (Automorfismo). Un automorfismo en una gráfica G es un isomorfismo de la gráfica consigo misma [34].

Es decir, un automorfismo de G es una permutación de $V(G)$, esto es, un isomorfismo de G en G; como se mencionó, puede considerarse como una permutación en V, que conserva adyacencias, y que el conjunto de dichas permutaciones forman un grupo $\Gamma(G)$, el grupo del automorfismo de G bajo la operación de composición de vértices. Afirmamos que una gráfica G es *transitiva por vértices*, si para cada par de vértices $u,\ v \in V(G)$ existe un automorfismo que manda a u en v. Una gráfica será *transitiva por aristas* si para cada par $e,\ f \in A(G)$ existe un automorfismo que mapea e en f.

Definición A.17 (Hipergráfica). Una hipergráfica H es una pareja (V, A) de conjuntos disjuntos, donde los elementos de A, de cualquier cardinalidad, son subconjuntos no vacíos de V, es decir el conjunto potencia de V [22].

Una hipergráfica es la generalización de un gráfica en la cual una arista puede conectar cualquier cantidad de vértices. Se tiene que V es el conjunto de vértices (hipernodos) de H y A será las hiperaristas de H.

Apéndice B

Elementos de álgebra

El objetivo de este anexo es presentar los conceptos básicos de Álgebra que son utilizados durante la exposición de los temas tratados en este texto.

B.1. Conjuntos

Se presenta un breve resumen de las nociones de teoría de conjuntos que se emplearán con frecuencia en este texto.

Un conjunto es una colección de objetos bien definida, nombrando los elementos o miembros del conjunto. El adjetivo *bien definido* significa que debe ser posible determinar si un elemento dado se encuentra en el conjunto bajo escrutinio o no [42].

Si A denota un conjunto y x es un elemento, a menudo es conveniente escribir: $x \in A$, como abreviatura de la afirmación de que x es un elemento de A. En caso contrario se dirá x es un elemento que no pertenece al conjunto A ($x \notin A$).

Definición B.1 (**Subconjunto**)**.** Sean A y B dos conjuntos. Decimos que B es un subconjunto de A, si cada elemento de B es también un elemento de A.

Usemos la notación $B \subset A$ siempre que B sea un subconjunto de A. Así pues, $B \subset A$ si y sólo si, $x \in B$ implica que $x \in A$. Dos conjuntos son iguales si contienen los mismos elementos ($A = B$).

Definición B.2 (**Operaciones en conjuntos**)**.** Sean A y B dos conjuntos. Las principales operaciones entre conjuntos son:

- Intersección. $A \cap B = \{x | x \in A \text{ y } x \in B\}$.
- Unión. $A \cup B = \{x | x \in A \text{ ó } x \in B\}$.
- El complemento de B respecto de A. $A - B = \{x \in A | x \notin B\}$[6].
- Sea X el conjunto universal. El complemento de A es: $A^c = \{x | x \in X, x \notin A\}$.

Nótese que el complemento del conjunto se define respecto al conjunto universal del cual se están tomando los conjuntos.

Definición B.3 (Vacío). Al conjunto que no contiene elementos se le llama *vacío* y se denota mediante el símbolo $\emptyset$. Si A y B son conjuntos que no tienen elementos en común entonces se dice que A y B son ajenos o que no son intersecables.

Se tiene que un par ordenado es el conjunto $(a, b) = \{\{a\}, \{a, b\}\}$. Este conjunto contiene dos elementos, los cuales, a su vez, son conjuntos, un elemento es $\{a\}$, el cual contiene a a como único elemento; el otro elemento es el conjunto $\{a, b\}$. Se puede probar que los pares ordenados (a, b) y (a', b') son iguales si y sólo si $a = a'$ y $b = b'$, propiedad fundamental de los pares ordenados [41].

Definición B.4 (Producto cartesiano). Si A y B son dos conjuntos, el conjunto de todos los pares ordenados cuyo primer componente pertenece a A y el segundo componente pertenece a B recibe el nombre de *producto cartesiano*, es decir:

$$A \times B = \{(a, b) | a \in A \text{ y } b \in B\}.$$

Definición B.5 (Conjunto Potencia). Dado un conjunto S, el conjunto potencia $\mathcal{P}(S)$ es el conjunto de todos los subconjuntos del conjunto S.

Si S tiene n elementos, entonces su conjunto potencia tiene 2^n elementos, es decir, si $|S| = n$, entonces $|\mathcal{P}(S)| = 2^n$ [42].

B.2. Funciones

El concepto de función es de gran importancia en las matemáticas. Esto se debe a que casi cualquier situación de la vida diaria es susceptible de ser interpretada como una función.

Si a cada objeto de un conjunto corresponde un único objeto de un segundo conjunto, entonces esta correspondencia se llama función y se denota por f. [6, 28].

Definición B.6 (Función). Sean A y B conjuntos (no necesariamente distintos). Una función de A a B es un conjunto f de pares ordenados en $A \times B$ con la propiedad de que si (a, b) y (a, b') son elementos de f, entonces $b = b'$. Al conjunto de todos los elementos de A que pueden aparecer como primeros miembros de los elementos de f se llama dominio de f y se denota por $D(f)$. El codominio de f es B. Al conjunto de todos los elementos de B que puedan aparecer como segundos miembros de los elementos de f es llamado rango de f, o el conjunto de valores de f, y es denotado por $R(f)$. En el caso en que $D(f) = A$, con frecuencia se dice que f mapea A en B o es un mapeo, de A en B y se escribe $f : A \to B$.

Si (a, b) es un elemento de una función f, entonces es común escribir $b = f(a)$ o $f : a \mapsto b$ en vez de $(a, b) \in f$. Por lo general se hace referencia al elemento b como el valor de f en el punto a, o la imagen de f bajo el punto a [6].

Existe otra manera de visualizar una función como una transformación de una parte del conjunto A hacia una parte del conjunto B. En estos términos, cuando $(a, b) \in f$, nos imaginamos a f como si tomará el elemento a del subconjunto $D(f)$ de A y lo transformará o mapeará en un elemento $b = f(a)$ del subconjunto $R(f)$ de B.

Definición B.7 (Composición). Sea f una función con dominio $D(f)$ en A y rango $R(f)$ en B y sea g una función con dominio $D(g)$ en B y rango $R(g)$ en C, tal que $R(f) \subseteq D(g)$. La composición de $g \circ f$ es la función desde A a C dada por:

$g \circ f = \{(a, c) \in A \times C |$ existe un elemento $b \in B$ tal que $(a, b) \in f$ y $(b, c) \in g\}$.

Definición B.8 (Función inyectiva). Podemos decir que f es inyectiva, o uno a uno, si y sólo si las dos relaciones $f(a) = b$ y $f(a') = b$ implican que $a = a'$. Alternativamente, f es inyectiva si y sólo si a, a' pertenecen al $D(f)$ y $a \neq a'$, entonces $f(a) \neq f(a')$.

Definición B.9 (Función inversa). Sea f una función con dominio $D(f)$ en A y rango $R(f)$ en B. Si $g = \{(b, a) \in B \times A : (a, b) \in f\}$ entonces g es una inyección con dominio $D(g) = R(f)$ en B y rango $R(g) = D(f)$ en A. La función g se llama función inversa de f y se denota por f^{-1}.

La función inversa se puede interpretar desde el punto de vista de un mapeo. Si f es inyectiva, mapea distintos elementos de $D(f)$ hacia distintos elementos de $R(f)$. De tal manera, que cada elemento b de $R(f)$ es la imagen de f de un único elemento a en $D(f)$. La función inversa f^{-1} mapea el elemento b hacia este elemento único a.

Definición B.10 (Función suprayectiva). Sea f una función. Sea $D(f) \subseteq A$ el dominio y el rango $R(f) \subseteq B$. Se dice que f es suprayectiva, o que f mapea sobre B, cuando el rango $R(f) = B$. Si f es suprayectiva, se puede decir que f es una suprayección.

Al definir una función es importante especificar el dominio de la función y el conjunto en el cual se adoptan los valores. Una vez hecho esto se puede averiguar si la función es o no suprayectiva.

Definición B.11 (Función biyectiva). Sea f una función con dominio $D(f)$ en A y rango $R(f)$ en B se dice que es biyectiva si es inyectiva (uno a uno) y si es suprayectiva (es decir, aplica $D(f)$ sobre B). Si f es biyectiva, se puede decir que f es una biyección.

Un conjunto B es finito si es vacío o si hay una biyección con dominio B y rango en un segmento inicial de $\mathbb{N}$ (los números naturales). Si no existe tal función, el conjunto es infinito. Si hay una biyección de B sobre $\mathbb{N}$, entonces el conjunto B es numerable (o enumerable). Si un conjunto es finito o es numerable, se dice que es contable.

Definición B.12 (Función máximo entero). La función máximo entero, denotada por $\lfloor\ \rfloor$, es la función cuyo dominio son los números reales y con regla de correspondencia: $\lfloor x \rfloor$ tal que, es el máximo entero no mayor a x.

Definición B.13 (Función mínimo entero). La función mínimo entero, denotada por $\lceil\ \rceil$, es la función cuyo dominio son los números reales y con regla de correspondencia: $\lceil x \rceil$ tal que, es el mínimo entero no inferior a x.

B.3. Combinatoria

El Cálculo Combinatorio prescinde de la cualidades de los objetos, pero no del orden en que se encuentran los mismos [9]. Primeramente, para nuestros propósitos, se enunciará la definición del factorial de un número.

Se llama factorial de un número n, entero y positivo, al producto de los primeros n números naturales consecutivos desde 1 hasta n, representado por $n!$. Formalmente queda:

Definición B.14 (Factorial). Sea $n \in \mathbb{Z}$ tal que $n \geq 0$. Definimos el factorial de n, denotado por $n!$, por inducción, como sigue:

1. $0! = 1$
2. $n! = (n-1)!n,\ n \geq 1.$

La regla del producto se define de la siguiente manera: si una actividad puede efectuarse en k pasos sucesivos y si el paso 1 puede efectuarse en n_1 maneras, el paso dos puede efectuarse en n_2 maneras,..., el paso k puede realizarse de n_k maneras, entonces la actividad puede hacerse en $n_1 \cdot n_2 \cdots n_k$ maneras.

Si de un conjunto $A = \{a, b, c\}$ se forman *subconjuntos* tomando elementos dos a dos, el nuevo conjunto se llama *arreglo* [36]. Se representa por: $A_{3,2}$.

De lo anterior se puede establecer la siguiente definición.

Definición B.15 (Arreglos). Los arreglos de n objetos son los distintos *grupos* que pueden formarse con ellos, de modo que entren k de esos objetos en cada grupo y cada uno de estos se diferencie de los demás o en los elementos (uno por lo menos) o en el orden de colocación de los mismos.

El número de arreglos de n elementos tomados de k en k se representa por $A_{n,k}$. El índice n indica el número de elementos de que se dispone; el índice k, los elementos que hay en cada grupo. Los arreglos de un orden cualquiera se forman colocando a la derecha de cada grupo, uno por uno, los elementos restantes que no entran en el arreglo.

• *Ejemplo.* Sean los elementos a, b, c y d.

$$\begin{aligned} A_{4,1} &= a\ b\ c\ d; \\ A_{4,2} &= ab\ ac\ ad\ ba\ bc\ bd; \quad ca\ cb\ cd\ da\ db\ dc; \\ A_{4,3} &= abc\ abd\ acb\ acd\ adb\ adc; \quad bac\ bad\ bca\ bcd\ bda\ bdc; \\ &\quad\ cab\ cad\ cba\ cbd\ cda\ cdb; \quad dab\ dac\ dba\ dbc\ dca\ dcb; \end{aligned}$$

Éstos son todos. Si a cualquier arreglo de orden k que exista se le suprime el último elemento, queda uno de orden $k-1$ que estará ya considerado y que nos habrá servido en el proceso anterior de formación. ▲

El número de componentes del *grupo* es menor que el número de elementos del conjunto original, es decir $k < n$. En general, el número de arreglos es:

$$A_{n,k} = n(n-1)(n-2)(n-3)\ldots(n-k+1)$$

Es decir, el número de arreglos de grado k que se pueden formar con n objetos es igual al producto de tantos factores consecutivos decrecientes, a partir del índice n, como unidades tiene el índice k. El último término puede escribirse como: $[n-(k-1)]$.

Una vez aclarado el concepto de factorial. Si se multiplica y divide la fórmula por $(n-k)!$ tenemos que;

$$(n-k)! = \underbrace{(n-k)[(n-k)-1][(n-k)-2][(n-k)-3]\cdots 3*2*1}_{n-k \text{ factores}}$$

Entonces puede escribirse:

$$A_{n,k} = \frac{n!}{(n-k)!} = n(n-1)(n-2)(n-3)\ldots(n-k+1)$$

Cuando los arreglos son los grupos que comprenden a todos los n elementos del conjunto original tomados de n en n, entonces se les llama permutaciones. En otras palabras, si los *grupos* que se forman importa el orden, no se repiten los elementos y entran todos los elementos, es una permutación.

Dicho esto, consideremos dos tipos de permutaciones:

Definición B.16 (Permutaciones). Las permutaciones de n objetos son los distintos grupos que con ellos pueden formarse; de tal modo que en cada grupo entren todos esos objetos. Se tiene que son los arreglos de esos n objetos tomados de n en n. Sólo pueden diferenciarse unos grupos de otros en el orden de colocación de sus elementos.

• *Ejemplo.* Las permutaciones de los elementos, $a,\ b,\ c$ son:

$$abc \quad acb \quad bac \quad bca \quad cab \quad cba$$

▲

El número de permutaciones se representa por el símbolo P_n, en el cual el subíndice n indica el número de elementos que dispone el conjunto. Para obtener las permutaciones P_n, basta formar las $A_{n,n}$, teniendo presente que en cada grupo entran todos los elementos.

Si en la fórmula de arreglos: $A_{n,k}$ se hace $n = k$ queda como:

$$A_{n,n} = \frac{n!}{(n-k)!} = P_n = \frac{n!}{(n-n)!} = \frac{n!}{0!} = n! \quad \therefore \quad P_n = n!$$

Permutaciones $P_{n,k}$

Un arreglo ordenado de k elementos de un conjunto que contiene n elementos distintos recibe el nombre de *permutación* de n elementos tomados de k en k y se denota mediante $P_{n,k}$ donde $k \leq n$. Por lo que, una permutación de objetos implica ordenamiento [42]. En realidad una permutación es un arreglo.

Definición B.17 (Permutaciones $P_{n,k}$). Sea $n \geq k$. Se denomina a los distintos grupos formados por k elementos como las permutaciones de n elementos tomados de k en k de manera que: no todos los elementos del conjunto original entran en los grupos formados por k elementos, importa el orden de colocación de cada uno de los elementos y ningún grupo formado se repite en sus elementos.

La fórmula para calcularlas es:

$$\begin{aligned} P_{n,k} &= n(n-1)(n-2)\cdots(n-k+1) \\ &= \frac{n!}{(n-k)!} \end{aligned}$$

El primer elemento de una permutación puede elegirse de n maneras en un conjunto que tiene n elementos. Habiendo elegido el primer elemento para la primera posición de la permutación, el segundo elemento puede elegirse en $(n-1)$ maneras, pues son $(n-1)$ elementos que quedan en el conjunto. Siguiendo la idea, hay $(n-2)$ maneras de elegir el tercer elemento, y así en lo sucesivo. Por último, hay $n-(k-1) = n-k+1$ maneras de elegir el k-ésimo elemento. Consecuentemente, por la regla del producto, hay: $n(n-1)(n-2)\cdots(n-k+1)$ maneras de elegir los n elementos del conjunto dado.

Obsérvese que si tomamos las permutaciones de $P_{6,5} = P_{6,6}$, es decir, en general ocurre que:

$$\begin{aligned} P_{n,n-1} &= \frac{n!}{[n-(n-1)]!} \\ &= \frac{n!}{(n-n+1)!} \\ &= \frac{n!}{1!} \\ &= n! \end{aligned} \qquad \begin{aligned} P_{n,n} &= \frac{n!}{(n-n)!} \\ &= \frac{n!}{0!} \\ &= \frac{n!}{1} \\ &= n! \end{aligned}$$

$$\therefore P_{n,n-1} = P_{n,n}$$

Sin embargo, debe aclararse que, cualitativamente, los grupos que se forman son diferentes. Los grupos del primer miembro contienen $n-1$ elementos mientras que los del segundo miembro contienen n elementos en los grupos que se forman.

Cuando k es igual a n el número de posibles resultados es todas las diferentes permutaciones de $P_{n,n} = n(n-1)\ldots 1 = n!$

Propiedades de las permutaciones

En el cálculo combinatorio y sus numerosas aplicaciones se trabaja constantemente con funciones de conjuntos finitos en conjuntos finitos.

• *Ejemplo.* Sea A el conjunto que consta de los elementos 1,2,3 y B el formado por las letras a, b, c, d. Si, como ejemplo, la función $f : A \to B$ está dada por: $f(1) = a, f(2) = d, f(3) = c$, será muy cómoda, en lo que sigue, la notación:

$$f = \begin{pmatrix} 1 & 2 & 3 \\ a & d & c \end{pmatrix}$$

En el primer renglón escribimos todos los elementos del dominio y debajo de cada uno de ellos todos los elementos de rango que les corresponde según la función. ▲

Designaremos de aquí en adelante con I_n al conjunto de los n primeros números naturales, es decir: $I_n = \{1, 2, \ldots, n\}$.

De los elementos de I_n formaremos el conjunto de todas las permutaciones R_n. Una permutación de un conjunto I_n es una función biyectiva de I_n en I_n.

Una forma de escribir una permutación, digamos σ, consiste en darle un formato de matriz de dos filas, situando en la primera fila los elementos del dominio y en la segunda las imágenes correspondientes a los elementos (aunque no es necesario que estén ordenados, se prefiere por comodidad).

Recordemos que R_n es el conjunto de todas las permutaciones de I_n. Sabemos que hay $n!$.

• *Ejemplo.* Las permutaciones de $I_3 = \{1, 2, 3\}$ son:

$$\begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}$$

▲

Si $\sigma : I_n \to I_n$ es una permutación, acostumbramos escribir:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & \dots & n \\ \sigma(1) & \sigma(2) & \sigma(3) & \dots & \sigma(n) \end{pmatrix}$$

Se llama permutación *identidad* a la que permutación en la que sus elementos se siguen en el orden natural (alfabético si son letras, el de la serie natural si son números o tiene subíndices numéricos). En una permutación cualesquiera dos elementos forman una inversión si el mayor está antes que el menor, en caso contrario, no forman una inversión.

Más formalmente, $\sigma(i)$ y $\sigma(j)$ forman una *inversión* si $i < j$ y $\sigma(i) > \sigma(j)$. Para definir cuando una permutación es par o impar, es útil el concepto de inversión. En la permutación:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 7 & 1 & 5 & 6 & 2 & 4 \end{pmatrix}$$

Se tiene que los elementos 7 y 1 (del segundo renglón) forman una inversión. También los elementos 3 y 1; también 6 y 2. En cambio 3 y 7 no forman una inversión, ni 1 y 5, ni 3 y 6.

Una permutación es de *clase* par o impar según el número de inversiones que en ella existan, sea par o impar.

Así pues, en la permutación σ hay 11 inversiones. Obsérvese que para contar cuántas inversiones hay en una permutación basta contar cuántos números mayores que 1 preceden a 1, cuantos mayores que 2 preceden a 2, etcétera, hasta $n-1$. Se dice que una permutación es par si tiene un número par de inversiones y que es impar si tiene un número impar de inversiones.

Teorema B.1. Si en una permutación se cambian entre si dos elementos, la permutación cambia de clase [18].

Teorema B.2. Entre las $n!$ permutaciones de n elementos la mitad son de clase par y la otra mitad de clase impar [9].

Transposiciones. Consideremos una permutación σ de I_n; al transponer (intercambiar) dos de los elementos del segundo renglón se obtiene otra permutación. Por ejemplo, al transponer 2 y 3 en σ obtenemos τ:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 4 & 2 & 5 \end{pmatrix} \qquad \tau = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 3 & 5 \end{pmatrix};$$

al transponer 1 y 5 en σ' obtenemos τ':

$$\sigma' = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 7 & 1 & 5 & 6 & 2 & 4 \end{pmatrix} \qquad \tau' = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 7 & 5 & 1 & 6 & 2 & 4 \end{pmatrix}.$$

En estos casos se tiene que la permutación que resulta se ha obtenido de la permutación dada, mediante una transposición. Las transposiciones afectan la paridad de una permutación. Al examinar la permutación σ' de arriba y la permutación τ' obtenida mediante una transposición vemos que σ' es impar y que τ' es par. Al contar las inversiones de 1 con los demás elementos en σ y en τ observamos que hay las mismas excepto la del 1 con 5 que en σ' no forman inversión y en τ' sí. Todas las demás inversiones son las mismas.

Inversa de una permutación. Sea σ una permutación de $I_n = \{1, 2, \ldots, n\}$, l es decir una función biyectiva de $\sigma : I_n \to I_n$. Entonces la función inversa $\sigma' : I_n \to I_n$ es también una función biyectiva, es decir, σ' es también una permutación.

En otras palabras si $\sigma \in R_n$ entonces $\sigma' \in R_n$ también.

Recordemos cómo se define la inversa de una función biyectiva: $\sigma(i) = k$ si y solamente si $\sigma'(k) = i$.

En particular, si escribimos σ en la forma:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & \ldots & n \\ \sigma(1) & \sigma(2) & \sigma(3) & \ldots & \sigma(n) \end{pmatrix}$$

entonces $\sigma' =$ es

$$\begin{pmatrix} \sigma(1) & \sigma(2) & \sigma(3) & \dots & \sigma(n) \\ 1 & 2 & 3 & \dots & n \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & \dots & n \\ \sigma'(1) & \sigma'(2) & \sigma'(3) & \dots & \sigma'(n) \end{pmatrix}.$$

• *Ejemplo.* Considere la permutación σ:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 4 & 5 & 3 \end{pmatrix}$$

entonces

$$\sigma' = \begin{pmatrix} 2 & 1 & 4 & 5 & 3 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 5 & 3 & 4 \end{pmatrix}$$

▲

Lo observado anteriormente es cierto en general. Podemos asociar a cada permutación σ su inversa σ', con lo que obtenemos una función de R_n a R_n.

• *Ejemplo.* Dado el conjunto ordenado $\{1, ..., 8\}$ podemos expresar una permutación σ sobre éste mediante una matriz de correspondencias:

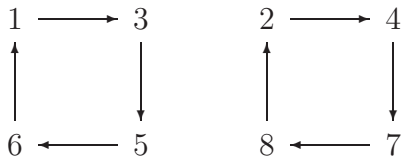
$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 4 & 5 & 7 & 6 & 1 & 8 & 2 \end{pmatrix}$$

Es biyectiva pues existe σ^{-1} de forma que $\sigma \circ \sigma^{-1} = \sigma_I$. Para ello, en primer lugar intercambiamos las filas y reordenamos las columnas, de modo que los elementos del dominio queden ordenados de forma natural.

$$\sigma^{-1} = \begin{pmatrix} 3 & 4 & 5 & 7 & 6 & 1 & 8 & 2 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 6 & 8 & 1 & 2 & 3 & 5 & 4 & 7 \end{pmatrix}$$

$$\sigma(\sigma^{-1}(n)) = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix}$$

Observe la representación gráfica de la permutación σ que nos muestra los ciclos de longitud 4 que contiene.



▲

Un ciclo es una permutación que intercambia cíclicamente sus elementos y fija los restantes. Para encontrar un ciclo en una permutación cualquiera usaremos el siguiente procedimiento:

(i) empezamos con cualquier elemento. Lo escribimos, a su derecha escribimos su imagen y continuamos hasta que se complete un ciclo,

(ii) escogemos cualquier elemento no contenido en el primero, y repetimos el paso anterior hasta contemplar el segundo ciclo, y

(iii) el proceso continua hasta que la permutación σ quede representada por ciclos disjuntos.

• *Ejemplo.* Usando el ejemplo y aplicando el procedimiento anterior, la permutación sigma σ queda expresada por:

$$\sigma = (1\,3\,5\,6)(2\,4\,7\,8)$$

La descomposición en ciclos no es única, tenemos los siguientes resultados equivalentes:

$$\sigma = (1\,3\,5\,6)(2\,4\,7\,8) = (2\,4\,7\,8)(1\,3\,5\,6) = (8\,2\,4\,7)(6\,1\,3\,5)$$

▲

La descomposición canónica de una permutación como producto de ciclos se obtiene colocando en primer lugar de cada ciclo el número más pequeño del mismo. Posteriormente se procede a la colocación de los ciclos, colocando primero el ciclo cuyo primer elemento sea menor. Frecuentemente, suelen omitirse los ciclos de longitud 1.

Bibliografía

[1] S. B. Akers, D. Harel, and B. Krishnamurthy. The Star Graph: An Attractive Alternative to the n-cube. In *Pro. International Conference on Parallel Processing*, pages 393–400, St. Charles, Illinois, August 1987.

[2] S. G. Akl. *The design and analysis of parallel algorithms*. Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[3] S. G. Akl and T. Wolff. Efficient sorting on the star graph interconnection network. *Telecommunication Systems*, 10(1-2):3–20, 1998.

[4] G. S. Almasi and A. Gottlieb. *Highly Parallel Computing*. The Benjamin/Commings Publishing Company, Inc., Redwood City, California, 2nd. edition, 1994.

[5] R. G. Bartle. *The Elements of Real Analysis*. John Wiley & Sons, Inc., New York, U.S.A., 2a. edition, 1976.

[6] R. G. Bartle. *Introducción al Análisis Matemático*. Limusa, México, D.F., 1989.

[7] J. Błazewicz, K. Ecker, B. Plateau, and D. Trystram, editors. *Handbook on Parallel and Distributed Processing*. Springer, Herdelberg Germany, 2000.

[8] J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. Elsevier Science Publishing Co. Inc., New York, 1976.

[9] G. M. Bruno. *Álgebra: Curso Superior*. Bruno, Madrid, 13 edition, 1984.

[10] A. García-Máynes C. and R. Mancio T. *Álgebra y Geometría*. Editorial Porrúa México, 2005.

[11] G. Chartrand. *Introductory Graph Theory*. New York: Dover Publication, 1977.

[12] G. Chartrand and O. R. Oellermann. *Applied an Algorithmic Graph Theory*. International Series in Pure and Applied Mathematics. Mcgraw-Hill College, 1992.

[13] Y. Y. Chen, D. R. Duh, T. L. Ye, and J. S. Fu. Weak-vertex-pancyclicity of (n, k)-star graphs. *Theoretical Computer Science*, 396(1–3):191–199, 2008.

[14] E. Cheng and M. J. Lipman. Unidirectional (n, k)-star graphs. *Journal of Interconnection Networks*, 3(1 & 2):19–34, 2002.

[15] W. K. Chiang and R. J. Chen. The (n, k)-star graph: A generalized star graph. *Information Processing Letters*, 56:259–264, 1995.

[16] W. K. Chiang and R. J. Chen. Topological properties of the (n, k)-star graph. *International Journal of Foundations of Computer Science*, 9(2):235–248, 1998.

[17] C-W. Chiu, C-B. Yang, K-S. Huang, and C-T. Tseng. A fault-tolerant routing algorithm with safety vectors on the (n, k)-star graph. In *ISPAN*, pages 34–39, Leganés Madrid, 2009. IEEE Computer Society.

[18] H. Cárdenas, E. Lluis, F. Raggi, and F. Tomás. *Álgebra Superior: conjuntos y combinatoria, introducción al álgebra lineal, estructuras numéricas, polinomios y ecuaciones*. Trillas, México, D.F., 1990.

[19] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers, San Francisco, California, 2004.

[20] K. Day and A. Tripathi. Arrangement graphs: A class of generalized star graphs. *Inf. Process. Lett.*, 42(5):235–241, jul 1992.

[21] M. H. DeGroot. *Probability and Statistics*. Addison Wesley, United States of America, 1975.

[22] R. Distel. *Graph Theory*. New York, New York, Springer-Verlag Heidelberg, 2000.

[23] J. Dongarra, I. Foster, G. Fox, W. Gropp, K. Kennedy, L. Torczon, and A. White. *Sourcebook of Parallel Computing*. Morgan Kaufmann Publishers an imprint of Elsevier Science, San Fracisco, CA (USA), 2003.

[24] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection networks : an engineering approach*. Morgan Kaufmann, San Francisco, 2003.

[25] J. B. Fraleigh. *Álgebra abstracta: Primer curso*. Sistemas Técnicos de Edición, México, D.F., 1998.

[26] L. I. Goloviná. *Álgebra Lineal y algunas de sus aplicaciones*. Editorial Mir, tercera edition, 1986.

[27] A. Grama, A. Gupta, G. Karypis, and V. Kumar. *Introducction to Parallel Computing*. Addison Wesley, England, 2a. edition, 2003.

[28] N. B. Haaser, J. P. La Salle, and J. A. Sullivan. *Análisis Matemático 1: Curso de Introducción*. Trillas, México, 1995.

[29] L. He. Properties and algorithms of the (n, k)-Star Graphs. Master of science, Faculty of Mathematics and Science, Brock University. St. Catharines, Ontario, 2008.

[30] L. He, K. Qiu, and Z.Z. Shen. Neighbourhood broadcasting and broadcasting on the (n, k)-star graph. In A. Bourgeois and S. Q. Zheng, editors, *Algorithms and Architectures for Parallel Processing*, volume 5022 of *Lecture Notes in Computer Science*, pages 70–78. Springer Berlin Heidelberg, 2008.

[31] M. R. HoseinyFarahabady and H. Sarbazi-Azad. The grid-pyramid: A generalized pyramid network. *The Journal of Supercomputing*, 37(1):23–45, 2006.

[32] J. JáJá. *An Introduction to Parallel Algorithms*. Addison-Wesley, Reading, Massachusetts, 1992.

[33] L. Jinli, C. Manli, X. Yonghong, and Y. Shaowen. Optimum broadcasting algorithms in (n, k)-star graphs using spanning trees. In Keqiu Li, Chris Jesshope, Hai Jin, and Jean-Luc Gaudiot, editors, *Network and Parallel Computing*, volume 4672 of *Lecture Notes in Computer Science*, pages 220–230. Springer Berlin Heidelberg, 2007.

[34] H. Lin-Hsing and L. Cheng-Kuan. *Graph Theory and Interconnection Networks*. Taylor & Francis, Boca Ratón, 2009.

[35] J. M. Montaña, Gómez M. E, A. Robles, J. Flich, P. López, and J. Duato. Líneas de investigación en tolerancia a fallos. In *XIV Jornadas de Paralelismo*, Leganés Madrid, 2003.

[36] S. Munúzuri O. *Notas de análisis combinatorio*. Universidad Nacional Autónoma de México, 2002.

[37] K. Qei and S. G. Akl. On some properties of the star graph. *VLSI Design*, 2(4):389–396, 1995.

[38] Y. Saad and M. H. Schultz. Topological properties of hypercubes. *IEEE Trans. Comput.*, 37(7):867–872, July 1988.

[39] I. D. Scherson and A. S. Youssef, editors. *Interconnection networks for high-performance parallel computers*. IEEE Computer Society, Los Alamitos, California: IEEE Computer Society, 1994.

[40] Y. Sheng-I, Y. Chang-Biau, and C. Hon-Chan. Fault-tolerant routing on the star graph with safety vectors. In *International Symposium on Parallel Architectures, Algorithms and Networks*, page 301, Makati City, Metro Manila, Philippines, 2002. IEEE Computer Society.

[41] M. Spivak. *Calculus*. Editorial Reverté, Barcelona, España, 3a. edition, 2012.

[42] T. Veerarajan. *Matemáticas Discretas con Teoría de Gráficas y Combinatoria*. Mc. Graw Hill/Interamericana, 2008.

[43] Y. Weihua, L. Hengzhe, and G. Xiaofeng. A kind of conditional fault tolerance of (n, k)-star graphs. *Information Processing Letters*, 110(22):1007–1011, oct 2010.

Índice alfabético