VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"Codificación en Redes y Sistemas Distribuidos"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS (COMPUTACIÓN)

P R E S E N T A:

CARLOS IVÁN GUERRERO ROMÁN

DIRECTOR DE TESIS: DR. SERGIO RAJSBAUM

México, D.F. **2009.**

Índice general

Resumen

La cantidad de información que puede ser transmitida a través de una red de cómputo, se ha analizado a últimas fechas desde una nueva perspectiva, la de codificación en redes, lo que ha permitido contar con avances significativos en el área.

La información en una red de cómputo se manipula hoy en día del mismo modo que un bien inmutable[4]: es decir, paquetes de datos fluyen a través de canales de comunicación, entre nodos de una red de cómputo y son vistos de la misma manera que un fluido en un sistema de bombeo, o automóviles en un conjunto de avenidas y carreteras, hasta ahora son entidades incapaces de ser alteradas, solamente dirigidas, a lo más replicadas. Sin embargo, la información puede ser manipulada de muy diversas formas, mediante una gran variedad de funciones que toman como parámetros los paquetes de datos y efectúan combinaciones o modificaciones a éstos.

Los algoritmos de ruteo tradicionales se valen de una gama de funciones bastante limitada. Los dispositivos encargados de manipular los paquetes de datos a través de una red de cómputo, para que dichos paquetes alcancen su destino final, pueden básicamente aplicar 3 funciones: Elegir una conexión entre todas las posibles y enviar un paquete a través de dicho canal (Monodifusión [1]), elegir un conjunto de conexiones y enviar réplicas del paquete por cada canal (Multidifusión [2]), enviar réplicas del paquete por cada una de las conexiones disponibles (Difusión General [3]).

En investigaciones recientes, se ha analizado el empleo de nuevas funciones, capaces de combinar o modificar paquetes de manera que el tamaño de los mismos no se vea alterado (ver figura 1), pero que a la vez permitan un incremento en la cantidad de información que puede ser enviada a través de una red.

Este nuevo enfoque se presenta como una generalización de las técnicas de ruteo tradicionales, permitiendo a los nodos intermedios de una red la generación de nuevos paquetes, mediante la realización de operaciones algebraicas sobre los datos contenidos, proveyendo ventajas en su rendimiento y en la manera de visualizar diversos problemas vinculados al flujo de información a través de una red de cómputo, engendrando con ello una nueva perspectiva ante el problema de enrutamiento de paquetes, la perspectiva de codificación en redes.

[1] Transmisión Unicast
[2] Transmisión Multicast
[3] Transmisión Broadcast

Figura 1: *Tipos de Transmisión.* Una de las ventajas de la codificación de redes es que el tamaño de los paquetes no se altera al modificarlos o combinarlos mediante funciones. En el ejemplo sea $a = \{00111\}$ y $b = \{11010\}$, entonces el paquete codificado $a + b = \{11101\}$, conservando así su longitud dentro del campo finito $\mathbb{F}_{2^5}$.

En la actualidad existen algoritmos capaces de obtener en tiempo polinomial una solución factible para un problema de codificación en redes, lo que equivale a obtener un conjunto de funciones de codificación para cada uno de los nodos que conforman una red de cómputo, empleando un alfabeto de símbolos que permitirá la manipulación de los paquetes enviados a través de la red, de manera que sea posible para cada nodo de la red que requiera un subconjunto de los paquetes enviados, obtenerlos.

Dentro de este documento, se presentan un par de aplicaciones de la codificación en redes: primeramente, el análisis detallado de un algoritmo para la obtención de una solución factible al problema de codificación de redes, con el principal objetivo de contar con una visión global del panorama de la codificación en redes y posteriormente un algoritmo distribuido para la identificación de caminos disjuntos dentro de una gráfica dirigida, valiéndose para ello de la obtención de manera eficiente de un código de red para dicha gráfica, en donde, la principal aportación del empleo de la codificación en redes es el análisis de algoritmos distribuidos con novedosas características, como la simpleza y fácil adaptación a este tipo de problemas.

Capítulo 1

Introducción

1.1. Codificación en Redes

La propagación de información dentro de un conjunto de dispositivos de cómputo (computadoras, conmutadores, etc.) conectados mediante una red de comunicación ha sido un problema de relevancia que ha sido analizado dentro de la computación. Dado inicialmente un conjunto de dispositivos que cuentan con cierta información y un conjunto de dispositivos que requieren de dicha información (de manera total o parcial), el problema de propagación de información consiste en encontrar la manera de transmitir dicha información a través de la red de comunicación, para que los dispositivos que la requieren logren obtenerla en el menor tiempo posible y consumiendo la menor cantidad de recursos de la red.

De las soluciones propuestas hasta hoy, un considerable número de ellas, emplea la fragmentación de la información en paquetes, los cuales contienen parte de la información requerida, así como datos adicionales que permiten a los dispositivos involucrados en el envío de la información dirigir o encausar dichos paquetes hacia los dispositivos que requieren de ellos. Para llevar a cabo esta tarea, se han ideado y perfeccionado con el tiempo algoritmos para el enrutamiento de paquetes, la mayoría de los cuales basan su funcionamiento en algoritmos para el flujo de bienes dentro de una red de comunicación.

Uno de los inconvenientes emergentes ante este acercamiento, radica en que se analiza a la información como un ente material, es decir, solamente es posible redirigirlo, a lo más replicarlo, provocando con ello que las tasas máximas de propagación de información no fuesen alcanzables en una gran cantidad de ocasiones e incluso que dichas tasas máximas de propagación no fuesen siquiera calculables. Sin embargo, en fechas recientes, otro tipo de soluciones han sido planteadas para la

resolución del problema, dentro de las cuales se hace énfasis en la diferencia de que la información, en contraste con los bienes materiales, cuenta con características radicalmente diferentes como su posibilidad de combinación, cifrado y demás operaciones aplicables sobre ella que no habían sido explotadas del todo.

La codificación en redes es un área dentro de la cual se estudian los beneficios de extender las capacidades de los métodos de ruteo empleados actualmente. Este enfoque requiere contar con nodos pertenecientes a una red de cómputo, con la capacidad de enviar paquetes formados mediante combinaciones de los paquetes previamente recibidos y a cambio, los beneficios de este nuevo enfoque son entre otros: el incremento de la tasa de transferencia de información a través de la red y una mejor tolerancia a fallas de comunicación en la red de cómputo.

En el problema de Codificación en Redes se cuenta con un conjunto de dispositivos fuente que conocen uno o varios mensajes y un conjunto de dispositivos destino que requieren de dichos mensajes. Cada conexión en la red puede transmitir cualquier función lineal de los paquetes que recibe, siempre y cuando no exceda su capacidad de transmisión, y el conjunto de estas funciones representa una solución de Codificación de Redes, siendo dicha solución factible siempre y cuando los dispositivos fuente obtengan cada uno de los mensajes que requieren (Ver figura 1.1).

Figura 1.1: Ejemplo de red donde con una sesión de transmisión y ruteo es imposible que las 2 terminales reciban 2 mensajes de cada fuente en una misma ronda, mientras que con Codificación en Redes es factible

Existen en la actualidad implementaciones de algoritmos capaces de obtener códigos de red factibles para una gráfica dada. En especial, dentro de este documento, se analizará un algoritmo centralizado para la obtención de códigos de red factibles dada una gráfica, por otra parte, se empleará un algoritmo distribuido como base para mostrar la capacidad de la codificación en redes, no

solamente como una herramienta que incrementa tanto el desempeño de una red como la robustez de la misma, sino que además, aporta un novedoso enfoque para la resolución de diversos problemas que atañen al área de redes de cómputo. Finalmente, en el documento se presenta un algoritmo distribuido para la obtención de caminos disjuntos dentro de una red, empleando para ello la codificación en redes.

En general, el problema de la obtención de caminos disjuntos se puede describir de manera escueta como sigue: Dada una red de cómputo, un nodo fuente (s) y un nodo destino (t), encontrar un conjunto de h caminos desde s hasta t. Dentro de este documento, se analizará la variante de caminos de aristas disjuntas, dentro del cual dos caminos son disjuntos siempre y cuando no compartan aristas.

La solución propuesta en el documento funciona de la manera siguiente: primeramente se obtiene un código de red factible para la gráfica y gracias a que existen algoritmos distribuidos probabilísticos para tal tarea, cuya implementación resulta ser simple, es posible que una vez obtenido dicho código, se lleve a cabo un proceso de descubrimiento de caminos de aristas disjuntas, ayudándose para ello de los vectores de codificación que se asocian a cada una de las aristas de la red, basándose en el principio de que si un nodo recibe h mensajes linealmente independientes, existen al menos h caminos disjuntos.

1.2. Trabajos Vinculados

El establecimiento de multidifusión de información en una red de manera eficiente es otro de los problemas estudiados dentro de la codificación en redes. En él, un nodo s cuenta con h paquetes que son requeridos por un conjunto de nodos terminales T dentro de una red de comunicación.

En un artículo publicado a mediados del año 2000[5], Ahlswede presenta una nueva técnica de ruteo que trata el problema de alcanzar la tasa máxima de comunicación posible en una red de cómputo con multidifusión, lo cual no era posible en la mayoría de los casos mediante el empleo de las técnicas de ruteo tradicionales, en dicho artículo, se muestra que para ello se requiere emplear codificación en algunos nodos, lo que permitiría un ahorro del ancho de banda. Además, se muestra que es posible determinar la máxima tasa de comunicación en una red de cómputo y se propone un teorema que demuestra esta aseveración, dentro de este artículo se acuña el concepto de codificación en redes.

Posiblemente una de las aportaciones más importantes de la codificación en redes queda plasmada en un teorema dentro de este mismo artículo. Este teorema establece que si se permite a nodos intermedios de la red combinar sus flujos de información entrantes, entonces cada uno de los destinos será capaz de recibir información a una taza equivalente a la del escenario en donde utiliza un solo destino la red. Adicionalmente se establece que es suficiente para los nodos intermedios efectuar operaciones lineales, es decir, adiciones y multiplicaciones sobre un campo finito $\mathbb{F}_q$, a lo que se le denomina codificación lineal de redes.

El artículo de Ahlswede [5], generó un creciente interés por el estudio de la codificación en redes. Una de las principales incógnitas engendradas a partir de su publicación, fue la clase de codificación requerida en los nodos de una red para poder alcanzar mayores tasas de comunicación. En un artículo publicado en 2003 [6], se prueba que mediante el uso de funciones lineales para realizar la codificación de los paquetes enviados, es posible alcanzar la tasa máxima de comunicación en algunos casos en los que mediante codificaciones triviales de redes (ruteo tradicional), resulta imposible alcanzar, es decir, los paquetes salientes de un nodo, si son generados mediante combinaciones lineales de los paquetes entrantes a dicho nodo dentro la red, permiten un incremento en la tasa de transmisión.

Por otra parte, a fines del año 2003, se presenta un marco algebraico[7] para el estudio de las redes de computadoras y su capacidad máxima de transmisión de información, este marco permite un vínculo entre el problema de flujo de información en una red y el álgebra sobre un campo finito. Además se presenta un algoritmo que permite la obtención de una codificación de redes en tiempo polinomial, con lo que se torna factible una implementación de la codificación en redes dentro de un sistema real.

Un año después, al explorar la aplicabilidad de la codificación en redes dentro de una amplia gama de aplicaciones, se propone una taxonomía para problemas vinculados al flujo de información dentro de una red de cómputo [8], dicha clasificación consta de 3 categorías principales:

- Problemas con solución sin codificación en redes
- Problemas cuya solución toma un tiempo de orden polinomial empleando codificación en redes
- Problemas para los cuales la obtención de una solución que tome un tiempo de orden polinomial no es posible [1]

[1] Este tipo de problemas se denotan comúnmente como "problemas NP"

En este artículo, se muestra que existen limitaciones para la codificación en redes empleando únicamente combinaciones lineales de paquetes, por ejemplo, al momento de resolver diversos problemas de flujo de información dentro de una red de cómputo, no existen algoritmos que tomen tiempo de orden polinomial, para determinar el tamaño mínimo del alfabeto requerido.

Jaggi y Sanders en [2] presentan una serie de algoritmos polinomiales, tanto deterministas como aleatorios, que permiten la obtención de códigos factibles sobre redes sin ciclos, algunos de los cuales serán analizados con un mayor nivel de detalle a lo largo de este documento.

Respecto a la obtención de caminos de aristas disjuntas mediante un algoritmo distribuido, existen una gran variedad de soluciones propuestas, una de las más estudiadas ha sido la que atañe al problema de Menger, el cual se enuncia a continuación: Dada una gráfica $G = (V, E)$ [2], dos vértices $s, t \in V$, encontrar el mayor número de caminos disjuntos que conecten a s con t dentro de G [9]. Este problema cuenta con 4 posibles variantes ya que la gráfica G puede ser dirigida o no dirigida y los caminos pueden ser de aristas disjuntas [3] o de nodos disjuntos. Desafortunadamente, en la actualidad existen escasos algoritmos distribuidos eficientes para la resolución de este problema, es por ello que el empleo de codificación en redes dentro de esta área resulta de relevancia, ya que aporta un nuevo enfoque para la generación de algoritmos distribuidos capaces de resolver este tipo de problemas.

En [1] se muestra que en cualquier tipo de gráfica, el problema de encontrar el mayor número de caminos de aristas disjuntas es equivalente a la obtención del flujo máximo de la red si las aristas cuentan con capacidades unitarias, es decir, dada una gráfica en la cual cada arista puede transportar una unidad, se busca encontrar el número máximo de unidades que se pueden transmitir entre dos vértices s y t (flujo máximo).

Otro problema relacionado a la obtención de caminos disjuntos, ha sido la obtención de pares de caminos entre un par de vértices dentro de una gráfica dada, en [10] se propone una solución a este problema mediante un algoritmo distribuido que tiene complejidad de tiempo [4] $O(|V|D)$ y una complejidad de mensajes $O(|E||V| + |V|^2 D)$.

En [3] se propone un algoritmo centralizado para la obtención de pares de caminos disjuntos,

[2] donde V representa a un conjunto de vértices y E representa a un conjunto de aristas

[3] Un par de caminos resultan ser de aristas disjuntas si no comparten ninguna arista entre ellos

[4] $|V|$ representa la cardinalidad de V, es decir, el número de vértices de G y D es el diámetro de la gráfica, es decir, la máxima distancia en aristas entre dos vértices

el algoritmo, desarrollado por Suurballe y Tarjan, cuenta con una posible implementación distribuida (desarrollada dentro de este documento), la cual puede generalizarse para la obtención de $k-$caminos disjuntos entre un par de nodos de una gráfica dada.

1.3. Principales Aportaciones

Dentro de este documento se presentan principalmente dos aportaciones vinculadas estrechamente a la codificación en redes: la primera consta de una revisión técnica a un algoritmo que obtiene códigos factibles para una red dada, incluido un análisis a una mejora propuesta en [11] para un algoritmo centralizado, que permite la obtención de un código factible de manera más eficiente y se analiza una cota para el número de nodos de codificación empleados dentro del proceso de codificación.

El artículo analizado [11], presenta una solución al problema de encontrar un código de red factible para una gráfica acíclica, que cuente con el menor número posible de nodos de codificación, (*i.e.*, nodos que generan nuevos paquetes mediante operaciones algebraicas sobre paquetes recibidos a través de sus aristas entrantes), la solución propuesta se presenta mediante un algoritmo determinista, dentro del cual se transforma la red original en una nueva red con el número de aristas acotado, lo que permite encontrar un código de red factible en un menor tiempo.

Dentro del mismo artículo, se presenta un algoritmo que encuentra un código factible en un tiempo $O(|E|hk + |V|h^2k^2 + h^4k^3(k+h))$, donde k representa al número de nodos destino y h el número de paquetes, este tiempo mejora la cota del algoritmo más eficiente desarrollado por Jaggi [2], el cual encontraba un código factible en un tiempo $O(|E|hk + |V|h^2k^2(h+k))$.

Respecto a los nodos de codificación necesarios para encontrar un código de red factible, se prueba que su número se encuentra acotado por $O(h^3k^2)$.

Como segunda aportación, se presenta el empleo de la codificación en redes como una herramienta para la obtención de caminos de aristas disjuntas entre dos nodos de una gráfica. El algoritmo distribuido implementado, es analizado mediante tres medidas de desempeño: el tiempo que toma su ejecución, el número de mensajes enviados a lo largo de la ejecución y el tamaño de los mensajes en bits.

Como complemento, se presenta un cuadro comparativo entre diversos algoritmos que resuelven la misma problemática. Inicialmente, se analiza un algoritmo clásico para la resolución de problemas de manera distribuida, el cual consiste en el envío de la información de cada uno de los nodos de la red a un nodo central, quien se encarga de efectuar todas las operaciones requeridas para la obtención de los caminos de aristas disjuntas, mediante la ejecución de un algoritmo centralizado que resuelva dicho problema y finalmente, propaga la solución a través de la red.

Además se analiza un segundo algoritmo, basado en la solución de manera centralizada propuesta en [3], pero convertida en un algoritmo distribuido.

1.4. Estructura del Documento

En el capítulo 2, se presenta el modelo de cómputo empleado en el resto del documento y adicionalmente se muestran una serie de definiciones relacionadas con la codificación en redes, flujos de redes y teoría de gráficas, las cuales son necesarias para la comprensión de los diversos algoritmos presentados en las secciones subsecuentes. En el capítulo 3 se presentan primeramente una serie de algoritmos auxiliares, los cuales son empleados, ya sea por el algoritmo para la obtención de un código de red factible o por el algoritmo para la obtención de caminos disjuntos entre dos nodos de una gráfica, los cuales son presentados al final de dicho capítulo.

Finalmente, en el capítulo 4, se presentan los resultados y conclusiones alcanzadas a partir del trabajo realizado para la obtención de este documento y diversas vertientes relevantes para llevar a cabo trabajo futuro de investigación sobre ellas.

Capítulo 2

Modelo de Cómputo y Conceptos Empleados

Dentro de este capítulo, se define el modelo computacional empleado como base para la descripción de los diversos algoritmos analizados a lo largo de este documento, además, se presentan una serie de definiciones con el afán de permitir una comprensión más clara del funcionamiento de los principales algoritmos que componen a este documento.

Como marco inicial para comprender el funcionamiento de la codificación en redes [12], consideremos un dispositivo dentro de una red de cómputo, como por ejemplo un ruteador. Tradicionalmente, al enviar un paquete destinado a algún otro nodo de la red, el dispositivo simplemente lo replica. Con codificación en redes, existe la posibilidad de combinar uno o más paquetes recibidos por el dispositivo, generando así uno o más paquetes de salida.

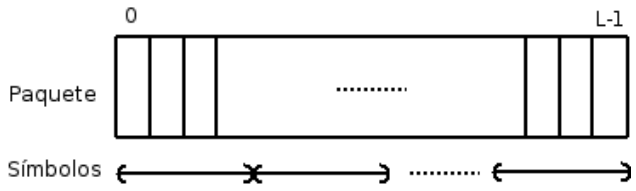

Figura 2.1: Relación entre un paquete de información y los símbolos empleados dentro del campo finito $\mathbb{F}_{2^s}$

Si consideramos que cada paquete consta de L bits (ver figura 1.1), cuando los paquetes que se desean combinar no cuentan con el mismo tamaño, el más pequeño es rellenado con ceros. Dentro de la codificación en redes, cada s bits de un paquete se interpretan como un símbolo sobre un campo

finito $\mathbb{F}_{2^s}$, por lo que cada paquete estará formado por L/s símbolos. Cada paquete saliente de un dispositivo de una red estará conformado como una combinación lineal de los paquetes recibidos, donde las operaciones de suma y producto se efectuarán sobre el mismo campo finito.

En general la estructura de los paquetes salientes de cada nodo estará conformada por dos elementos esenciales: un vector de codificación y un vector de información. El vector de codificación será empleado por los nodos destino para poder decodificar la información contenida en el segundo vector.

2.1. Modelo de Cómputo

Una red de cómputo consta de un conjunto de componentes capaces de transmitir y/o recibir información (computadoras, conmutadores, ruteadores, etc.) y de conexiones entre dichos componentes. Para el análisis de una red de cómputo, esta será modelada mediante una gráfica $G = (V, E)$ que consta de un conjunto V de vértices y un conjunto E de aristas. En ella suponemos que un componente está representado mediante un vértice $u \in V$ y cada uno de estos componentes, puede comunicarse con otro $v \in V$ si existe una arista, que representa una conexión entre ambos componentes en la red, $e = (u, v) \in E$, tal que el origen de e es u (expresado subsecuentemente como $S(e) = u$), su destino es v (expresado como $T(e) = v$).

Suponemos además que la comunicación entre vértices se presenta de manera síncrona, lo que permite que en cada unidad de tiempo un nodo envíe información solamente a sus nodos vecinos, a través de las conexiones que los comunican. Dichas conexiones cuentan con una capacidad asociada $U(e)$, la cual representa la cantidad máxima de información que puede fluir a través de la conexión en una sesión de transmisión. Además, suponemos que no se presentan fallas de pérdida ni modificación de mensajes enviados a través de la red y finalmente que el retraso en la transmisión para cada conexión es la misma en todos los casos, por lo que no es tomada en cuenta.

Dentro de este documento se analizarán gráficas dirigidas (en este tipo de gráficas las aristas cuentan con un nodo origen y un nodo destino, lo que implica un sentido en el flujo de datos a través de cada una de ellas), acíclicas (no existe un camino de un nodo a sí mismo) y se efectuará en ellas multidifusión de información por las siguientes razones:
Se emplearán *Gráficas dirigidas* puesto que es el tipo de gráficas en donde la codificación presenta mayores ventajas. Si empleamos codificación en redes dentro de gráficas no dirigidas, existen casos

en los que no nos ofrece beneficio alguno[11].
Las gráficas analizadas serán acíclicas, principalmente porque el manejo de ciclos dentro de la obtención de una codificación en redes, torna a este problema más complejo al resultar necesario el uso de diversos mecanismos como memoria en los nodos de codificación[13].
Finalmente, se empleará *multidifusión de información* por ser una generalización al contemplar la monodifusión, en el caso de contar con una transmisión a un solo nodo destino y las transmisiones de difusión general, en el caso de contar con una transmisión a el total de los vértices de la gráfica subyacente como destinos. Además, en trabajos previos[8] se muestra una clasificación natural para una gran cantidad de problemas relacionados con codificación lineal de redes. En esta clasificación, un problema es NP-completo o puede ser reducido a un problema de multidifusión.

2.2. Definiciones

A continuación se denotan una serie de conceptos empleados para la representación de problemas tanto de codificación de una red como de obtención de caminos de aristas disjuntas entre 2 nodos:

2.2.1. Campos Finitos

Debido a que las operaciones de codificación en redes se efectúan dentro de un campo finito, se presentan definiciones al respecto, con la intensión de contar con una idea nítida de su funcionamiento:

Definición 1. Un **campo** F es una estructura algebraica con 2 operaciones definidas sobre un conjunto de elementos, comúnmente denominadas suma (+) y producto (*), las cuales satisfacen las siguientes propiedades.

$\forall a, b, c \in F$:

- Cerradura sobre + y *. $a + b \in F, a * b \in F$
- Asociatividad de + y *. $a + (b + c) = (a + b) + c$ y $a * (b * c) = (a * b) * c$
- Conmutatividad de + y *. $a + b = b + a$ y $a * b = b * a$
- Distributivad de * sobre +. $a * (b + c) = (a * b) + (a * c)$
- Identidad de + y *. $a + 0 = a$ y $a * 1 = a$
- Inverso Aditivo. $a + (-a) = 0$

- Inverso Multiplicativo. $\forall a \in F : a \neq 0.\ \exists a^{-1} : a * (a^{-1}) = 1$

Definición 2. Un **campo finito** es un campo con la característica adicional de que el conjunto de elementos sobre el cual se efectúan las operaciones es finito.

Ejemplo 1. *El campo finito más pequeño consta solamente de 2 elementos:* $\{0, 1\}$*, se denota como* $\mathbb{F}_2$ *y puede ser definido mediante las siguientes tablas:*

$+$	0	1
0	0	1
1	1	0

$*$	0	1
0	0	0
1	0	1

Para un número primo p*, el conjunto de los números enteros módulo* p *es un campo finito, denotado como* $\mathbb{Z}_p = \{0, 1, ..., p-1\}$ *donde las operaciones son definidas realizando la operación en* $\mathbb{Z}_p$ *dividiendo por* p *y tomando el residuo.*

Dentro del área de la codificación en redes, los campos finitos empleados son de la forma $\mathbb{F}_{2^s}$, con $s \in \mathbb{N}$. Dada una cadena de s bits $b_{s-1}, ..., b_1, b_0$, ésta se puede interpretar como un polinomio $P(X)$ con coeficientes en $\mathbb{F}_s$:

$$b_{s-1}X^{s-1} + ... + b_2X^2 + b_1X + b_0$$

Para realizar una adición entre dos elementos dentro del campo finito, es necesario realizar adiciones entre los coeficientes de los polinomios dentro del campo $\mathbb{F}_s$.

Para efectuar multiplicaciones entre dos elementos, se multiplican los polinomios dentro del campo finito y el producto obtenido es dividido entre un polinomio irreducible [1] de grado s, de donde el residuo de la división, es el producto de la multiplicación.

Ejemplo 2. *Si deseamos sumar las cadenas de bits 10110110 y 1010011 dentro del campo finito* $\mathbb{F}_{2^8}$ *realizamos lo siguiente:*
Representamos las cadenas como polinomios siguiendo la fórmula antes mencionada:
10110110: $x^7 + x^5 + x^4 + x^2 + x$
1010011: $x^6 + x^4 + x + 1$
Sumamos los coeficientes de los polinomios dentro del campo finito:
$x^7 + x^6 + x^5 + x^2 + 1 = 11100101$

[1] Un polinomio irreducible es un polinomio que no puede ser factorizado como producto de dos polinomios más simples

Para efectuar la multiplicación, efectuamos el producto de los polinomios (las sumas de los coeficientes del polinomio se deben de efectuar igualmente dentro del campo finito):

$(x^7+x^5+x^4+x^2+x)*(x^6)$	$(x^{13}+x^{11}+x^{10}+x^8+x^7)$
$(x^7+x^5+x^4+x^2+x)*(x^4)$	$(x^{11}+x^9+x^8+x^6+x^5)$
$(x^7+x^5+x^4+x^2+x)*(x)$	$(x^8+x^6+x^5+x^3+x^2)$
$(x^7+x^5+x^4+x^2+x)*(1)$	$(x^7+x^5+x^4+x^2+x)$
	$(x^{13}+x^{10}+x^9+x^8+x^5+x^4+x^3+x)$

Dividimos el polinomio resultante entre un polinomio irreducible, para el caso del campo finito empleado podemos emplear el polinomio $x^8+x^4+x^3+x+1$. *Por simplicidad en la notación, se muestran en el siguiente ejemplo solamente los exponentes en los cuales el coeficiente no es cero, i.e. el polinomio* $x^8+x^4+x^3+x+1$ *se representa como (8 4 3 1 0).*

.	*(13 _ _ 10 9 8 _ _ 5 4 3 _ 1 _)*
*(8 4 3 1 0) * (5)*	*(13 _ _ _ 9 8 _ 6 5 _ _ _ _ _)*
.	*(_ _ _ 10 _ _ _ 6 _ 4 3 _ 1 _)*
.	
.	*(10 _ _ _ 6 _ 4 3 _ 1 _)*
*(8 4 3 1 0) * (2)*	*(10 _ _ _ 6 5 _ 3 2 _ _)*
.	*(_ _ _ _ _ 5 4 _ 2 1 _)*

Con lo que el producto es el polinomio $x^5+x^4+x^2+x$, *el cual representa a la cadena de bits 00110110*

2.2.2. Teoría de Gráficas

Vinculado de manera más estrecha a la obtención de caminos de aristas disjuntas, a continuación se presentan una serie de conceptos básicos sobre teoría de gráficas, definiciones y algoritmos empleados durante el documento.

Definición 3. Una **gráfica** G se encuentra definida por un conjunto V de vértices o nodos y un conjunto E de aristas o arcos, cuyos elementos son pares ordenados $(u,v) : u \in V, v \in V$.

Definición 4. Una **subgráfica** $G' = (V', E')$ de una gráfica $G = (V, E)$ cumple con $V' \subset V$ y además $E' \subset E$.

Definición 5. Un **paseo dirigido** en una gráfica G es una subgráfica que consta de una secuencia arbitraria de vértices $(i_1, i_2, ..., i_r) \in V$ y aristas $(e_1, e_2, ..., e_r) \in E$: $i_1 - e_1 - i_2 - e_2 - ... - i_{r-1} - e_r \breve{} 1 - i_r$ donde se tiene que $e_k = (i_k, i_{k+1}) \in E, \forall k : 1 \leq k \leq r-1$.

Generalmente un paseo dirigido se denota como una secuencia de vértices sin mención explícita de aristas, por lo que el caso anterior se puede representar como: $i_1 - i_2 - ... - i_r$ donde se tiene que $\forall k : i_k \in V$, igualmente es posible representarlo como una secuencia de aristas $e_1, e_2, ..., e_r$.

Definición 6. Un **camino dirigido** es un paseo dirigido donde no se repiten vértices, es decir no existen ciclos dentro del camino, definiendo un ciclo como un paseo en donde se repite al menos un vértice o una arista.

Definición 7. Un $(s-t)$ **camino** es un camino dirigido con origen en el nodo $s \in V$ y destino en el nodo $t \in V$.

Definición 8. Sea $i \in V$ un vértice cualquiera de G, la **lista de adyacencia** del nodo i denotado por $A(i)$ se define indistintamente como el conjunto de arcos que salen del nodo i:

$$Adyacencia(i) = A(i) = \{(i,j) \in E / j \in V\}$$

Definición 9. Dada una gráfica $G = (V, E)$, un par de **caminos** $\mathbb{P}_1 = \{e_{1,1}, e_{1,2}, ..., e_{1,n}\}$ y $\mathbb{P}_2 = \{e_{2,1}, e_{2,2}, ..., e_{2,m}\}$ tienen sus **aristas disjuntas** si $\mathbb{P}_1 \cap \mathbb{P}_2 = \Phi$.

Definición 10. Una gráfica $G = (V, E)$ es **acíclica** si para cualquier $v \in V$ no existe un camino que contenga a v como su vértice inicial y final.

Ahora, basándonos en las definiciones denotadas hasta el momento, podemos definir el primer problema a ser analizado dentro de este documento:

Problema 1. *Dada una gráfica $G = (V, E)$, un nodo fuente $s \in V$, un nodo destino $t \in V$, encontrar* h (s,t)*-caminos de aristas disjuntas entre s y t.*

Existen para el problema mencionado algunas variantes, en una de ellas, el valor h puede ser un parámetro requerido o bien, puede ser el valor máximo de caminos disponibles dentro de la gráfica dada.

2.2.3. Codificación en Redes

Debido a que la parte medular del documento se enfoca a la codificación en redes, se presentan un conjunto de definiciones y elementos necesarios para su comprensión, así como la representación de una red que será empleada a través del documento y el significado de un problema de codificación.

Definición 11. Para la representación de una **red de cómputo N** emplearemos una 2-tupla $N = \{G, U\}$ en donde los elementos mencionados representan lo siguiente:

- Una gráfica dirigida $G = (V, E)$, en donde los vértices $v \in V$ representan dispositivos de la red y las aristas $e = (u, v) \in E$ representan la existencia de una conexión del vértice $u \in V$ al vértice $v \in V$.

- Una función de capacidad $U : E \to \mathbb{Q}^+$ cuyo propósito es el de asignar a cada una de las aristas de la red de cómputo $e \in E$ un número positivo fraccionario ($q \in \mathbb{Q}^+$) que representa la cantidad máxima de información que puede viajar por cada arista de la red de cómputo.

Al emplear codificación en redes, las funciones de capacidad asociadas a cada una de las aristas de la gráfica subyacente a la red, suponemos cuentan con capacidad unitaria, lo cual dentro del modelo de cómputo definido al inicio de esta sección, en el cual se define una comunicación síncrona, permite que en cada transmisión se envíe un símbolo de algún campo finito $\mathbb{F}_q$. Inicialmente, cada nodo fuente emite un símbolo que es igualmente elemento del campo finito $\mathbb{F}_q$. En la práctica este modelo permite que cada arista transporte de manera confiable un símbolo por cada transmisión, si se emplea un alfabeto de mayor tamaño, $q = 2^m$, significará que la información será enviada en paquetes de m bits, los cuales serán procesados como símbolos dentro del campo finito $\mathbb{F}_q$, mediante operaciones sobre dicho campo.

Definición 12. Dado un alfabeto de símbolos Σ de un campo finito, una gráfica $G = (V, E)$, un nodo fuente $s \in V$ y un conjunto M de mensajes conocidos por s, donde $|M| = h$, una **función de codificación** $f_e : e \in E$ se define como:

- $f_e : \Sigma^h \to \Sigma^{u(e)}$ si $e(s, u)$, $u \in V$

- $f_e : \Sigma^{u_{in}(v)} \to \Sigma^{u(e)}$ si $e(v, u)$, $u \in V$ y $v \neq s$
  Donde $u(e)$ es la capacidad asociada a la arista e, $u_{in}(v) = \sum_{w:(w,v)\in E} u(w, v)$ y representa la capacidad total de las aristas de entrada del nodo v

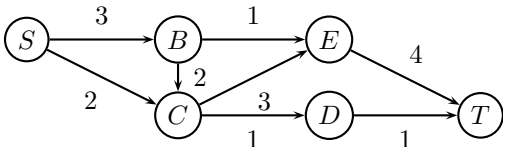

Figura 2.2: Funciones de codificación

Ejemplo 3. *A partir de la gráfica de la figura 2.2, se muestra la obtención de funciones de codificación en redes, en el cual los números asociados a cada una de las aristas representan su capacidad de transmisión, el conjunto de símbolos empleado es Σ, cuyo número inicial de elementos*

dependerá del tamaño de los paquetes empleados, que dentro de este ejemplo definiremos como $M = \{m_1, m_2, m_3, m_4, m_5\}$ *y las funciones de codificación asociadas a cada una de las aristas son:*
$f_{(S,B)} : \Sigma^5 \to \Sigma^3$
$f_{(S,C)} : \Sigma^5 \to \Sigma^2$
$f_{(B,C)} : \Sigma^3 \to \Sigma^2$
$f_{(B,E)} : \Sigma^3 \to \Sigma^1$
$f_{(C,D)} : \Sigma^4 \to \Sigma^1$
$f_{(C,E)} : \Sigma^4 \to \Sigma^3$
$f_{(D,T)} : \Sigma^1 \to \Sigma^1$
$f_{(E,T)} : \Sigma^4 \to \Sigma^4$

Definición 13. La **codificación de una red** N quedará determinada por los siguientes elementos:

- Un alfabeto Σ con los símbolos de un campo finito válidos dentro de la codificación propuesta
- Un conjunto de funciones de codificación $\mathbb{F}(N) = \{f_e | \forall e \in E\}$

Definición 14. Dada una gráfica $G = (V, E)$, un nodo inicial $s \in V$, un conjunto de nodos destino T y un conjunto de mensajes M conocidos por s, una **codificación de red es factible** si para cada $t \in T$ existe una función de decodificación $g_t : \Sigma^{u_{in}(t)} \to \Sigma^h$, que permita a cada nodo destino obtener decodificados los $|M| = h$ mensajes conocidos por s.

Definición 15. Dada una gráfica $G = (V, E)$, un nodo inicial $s \in V$, un conjunto de nodos destino T y un conjunto de mensajes M conocidos por s, una **codificación de red es mínima** con respecto a remosión de aristas si la codificación es factible y la remosión de cualquier arista de G impide la factibilidad de la codificación.

Definición 16. Dada una red de cómputo N y una codificación de red, una **arista de reenvío** $e = (v, w) \in E$ es aquella que cuenta con una función de codificación f_e que puede ser descompuesta en $u(e)$ funciones que mapean $\Sigma^{u_{in}(v)} \to \Sigma$, de manera que cada función dependa solamente de una variable, de lo contrario, se denominará a e como **arista de codificación**.

Definición 17. Un **nodo de codificación** $v \in V$ es aquel que cuenta con al menos una arista saliente $e = (v, w) \in E$ que es de codificación, de lo contrario v es un **nodo de reenvío**.

El siguiente problema que será analizado se enuncia a continuación:

Problema 2. *Un* ***problema de codificación de una red (PCR)*** *radica en obtener una codificación de red que permita a los nodos* $t \in T$ *obtener la información que requiere cada uno de ellos a partir de los siguientes parámetros:*

- *Una red de cómputo* $N = \{G(V, E), U\}$
- *Un nodo Fuente* s *que conoce los mensajes*
- *Un conjunto de nodos Destino* $T = \{T_1, ..., T_m\}$ *que requieren de al menos un mensaje*
- *Un conjunto* M *compuesto por* h *mensajes*

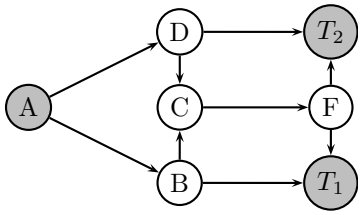

Figura 2.3: *Ejemplo de un problema de codificación.*

Ejemplo 4. *Problema de codificación de redes:*
Tomemos la red N *(ver figura 2.3), en donde asumimos que la función de capacidad asocia a cada arista una capacidad unitaria* $(\forall e \in E, u(e) = 1)$
El conjunto de $h = |M|$ *mensajes es* $M = \{a, b\}$
Un nodo fuente $s = A$ *donde* $S(a) = A, S(b) = A$
El conjunto de nodos destino $T = \{T_1, T_2\}$ *donde tanto* T_1 *como* T_2 *requieren de* a *y* b

Para resolver un problema de codificación de una red, es necesario hacer llegar a de cada uno de los nodos terminales o destino todos los paquetes requeridos por cada uno de ellos en una sola sesión de transmisión, es decir, cada nodo espera los paquetes requeridos por su función de codificación asociada y en el momento de recibirlos, ejecuta dicha función generando las salidas de cada una de sus conexiones.

Un conjunto de funciones $\mathbb{F}(N)$ *que resuelve el problema anterior es el siguiente:*

Arista	***Entrada***	***Salida***	***Función***
$A \to B$	a, b	a	$f(a,b) = a$
$A \to D$	a, b	b	$f(a,b) = b$
$B \to C$	a	a	$f(a) = a$
$B \to T_1$	a	a	$f(a) = a$
$C \to F$	a, b	$a + b$	$f(a,b) = a + b$
$D \to C$	b	b	$f(b) = b$
$D \to T_2$	b	b	$f(b) = b$
$F \to T_1$	$a + b$	$a + b$	$f(a+b) = a + b$
$F \to T_2$	$a + b$	$a + b$	$f(a+b) = a + b$

Y un conjunto de símbolos que da solución al problema es $\Sigma = \{0, 1\}$*, por lo que las operaciones se efectuarán sobre el campo finito* $\mathbb{F}_{2^n}$*, donde* $n = MAX(|a|, |b|)$*.*

Figura 2.4: Codificación final

Para la decodificación hay que recordar que las operaciones se efectúan sobre el campo finito $\mathbb{F}_2$ *por lo que es posible para el nodo* T_1 *por ejemplo obtener ambos paquetes:* a *y* b*, de la siguiente manera:*

T_1 *recibe* a *desde el nodo* B

T_1 *recibe* $a + b$ *desde el nodo* F

Por lo que al efectuar la adición de los 2 paquetes recibidos sobre el campo obtiene lo siguiente:

$a + (a + b) = 2a + b \equiv b(mod_2)$

Y es por ello que puede decodificar ambos paquetes.

En el ejemplo, una arista de codificación de la red es $\{C \to F\}$*, y un nodo de codificación es* $\{C\}$*, pues es el único en donde se realizan operaciones de combinación de paquetes no triviales.*

Por otra parte la codificación es mínima puesto que la remosión de cualquier arista impediría una codificación.

Notas sobre la codificación en redes en la práctica [12]

Como complemento al ejemplo anterior, a continuación se muestra a detalle la obtención de la codificación en redes.

Supongamos los mensajes $a : 0100110$ y $b : 10011100$.
Como se mencionó anteriormente, al ser los mensajes de longitudes diferentes, se complementa con ceros al mensaje más pequeño, por lo que $a : 00100110$ y $b : 10011100$
Para proseguir con la solución, el campo finito empleado será $\mathbb{F}_{2^8}$, puesto que los elementos de cada mensaje son binarios y cada mensaje cuenta con 8 elementos.

Cada uno de los paquetes que será enviado a través de la red consta de 2 vectores: un vector de codificación $\bar{g} = (g_1, g_2, ..., g_n)$ y un vector de información $\bar{M} = (m_1, m_2, ..., m_m)$ con la información codificada, n es el número de paquetes y m es el número de símbolos dentro de cada mensaje, para nuestro caso $n = 2$ y $m = 8$.

Para obtener un vector con la información codificada en cada una de las aristas de la red $\bar{X}_i : (x_1, x_2, ..., x_m)$ se empleará la siguiente ecuación: $\forall k \in [1; m] : x_k = \sum_{i=1}^{n} g_i M_k^i$, donde M_k^i representa al k-ésimo símbolo del i-ésimo mensaje.

Para todos los nodos exceptuando el fuente y el destino, es necesario calcular el vector de codificación $\bar{g}'_i = \sum_{j=1}^{m} h_j g_i^j$, para el cual se requiere de un vector de codificación local $\bar{h}$ y los vectores de codificación de cada uno de los mensajes recibidos $\{(\bar{g}^1, \bar{M}^1), (\bar{g}^2, \bar{M}^2), ..., (\bar{g}^m, \bar{M}^m)\}$, en donde $(\bar{g}^i, \bar{M}^i)$ es el par de vectores que conforman al i-ésimo paquete recibido por este nodo y g_i^j es el i-ésimo símbolo del j-ésimo vector de codificación.

Continuando con el ejemplo, para el nodo S contamos con lo siguiente:
En un principio, S conoce 2 mensajes, $a = (00100110)$ y $b = (10011100)$
Para el envío del mensaje a, requerimos 1 mensaje codificado por cada una de las aristas con origen en S:

Para la arista $S \rightarrow B$ tenemos lo siguiente:
$\bar{g} = (10)$
$\bar{M} = (m_1 m_2 ... m_8) = (00100110)$
A partir de estos datos, obtenemos el vector de información:
$x_k = \sum_{i=1}^{2} g_i M_k^i$

$x_1 = \sum_{i=1}^{2} g_i M_k^i$

$$x_1 = 1(0) + 0(1) = 0 \quad x_2 = 1(0) + 0(0) = 0$$
$$x_3 = 1(1) + 0(0) = 1 \quad x_4 = 1(0) + 0(1) = 0$$
$$x_5 = 1(0) + 0(1) = 0 \quad x_6 = 1(1) + 0(1) = 1$$
$$x_7 = 1(1) + 0(0) = 1 \quad x_8 = 1(0) + 0(0) = 0$$

Por lo que el vector de información $\bar{X} = (00100110)$ y por tanto, el mensaje enviado a través de la arista $S \rightarrow B$ será $[(10), (00100110)]$.
Similarmente para el envío del mensaje b por la arista $S \rightarrow D$:
$\bar{g} = (01)$
$\bar{X} = (10011100)$
Se obtiene al vector $[(01), (10011100)]$, para ser enviado a través de la arista.
Algo similar sucede con los nodos D y B puesto que no son nodos de codificación.

Sin embargo vale la pena analizar con un poco más de detalle al nodo C, el cual recibe los siguientes mensajes:
Desde $D : [(01), (10011100)]$
Desde $B : [(10), (00100110)]$
Para proseguir, se requiere de la creación de un vector de codificación local $h = (11)$
Con lo que se obtiene el nuevo vector de codificación $\bar{g}' = (g_1' g_2')$ para el mensaje:
$g_i = \sum_{j=1}^{2} h_j g_i^j$
$g_1 = \sum_{j=1}^{2} h_j g_1^j$
$g_1 = 1(0) + 1(1) = 1$
$g_2 = 1(1) + 1(0) = 1$
$\bar{g}' = (11)$
Y para el vector de información:
$x_k = \sum_{i=1}^{2} g_i' M_k^i$

$$x_1 = 1(0) + 1(1) = 1 \quad x_2 = 1(0) + 1(0) = 0$$
$$x_3 = 1(1) + 1(0) = 1 \quad x_4 = 1(0) + 1(1) = 1$$
$$x_5 = 1(0) + 1(1) = 1 \quad x_6 = 1(1) + 1(1) = 0$$
$$x_7 = 1(1) + 1(0) = 1 \quad x_8 = 1(0) + 1(0) = 0$$

Obteniendo el vector de información $\bar{X} = (10111010)$ y el mensaje a ser enviado será $[(11), (10111010)]$

Finalmente el nodo T_1 recibe los siguientes paquetes:
Desde $B : [(10), (00100110)]$

Desde $F : [(11), (10111010)]$

Y a partir de ellos deberá de ser capaz de regenerar los mensajes originales:

$$\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$$
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix}$$

En donde observamos que al sumar ambos mensajes se resuelve el sistema de ecuaciones y se obtienen los mensajes originales a y b dentro del nodo T_1. Similarmente se resuelve un sistema de ecuaciones dentro del nodo T_2 para obtener de igual forma los mensajes originales.

2.2.4. Flujos de Redes [1]

La obtención de caminos de aristas disjuntas entre un par de nodos en una gráfica, puede analizarse bajo la perspectiva de flujos de redes, es decir, la obtención de dichos caminos es equiparable a la obtención de un flujo entre el mismo par de vértices si asociamos una capacidad unitaria a cada arista. Es por ello que se torna relevante la comprensión de los flujos de redes dentro de este documento, para la comprensión de una serie de rutinas y algoritmos vinculados a la obtención de caminos de aristas disjuntas entre un par de vértices dentro de una gráfica.

Se presentan una serie de definiciones que atañen a los flujos de redes:

Definición 18. En una gráfica $G = (V, E)$, el **flujo** de una arista $(i, j) \in E$ se define como x_{ij} sujeto a:
$0 \leq x_{ij} \leq u_{ij}$, donde u_{ij} es la capacidad asociada a la arista (i, j).

Definición 19. Dada una gráfica $G = (V, E)$, Sea el conjunto $S \subseteq V$, $\bar{S} = V - S$ y el conjunto $K \subseteq E$. Definimos al **flujo de un conjunto de aristas** K como $f(K) = \sum_{e \in K} f(e)$.
Si las aristas de K son de la forma $S, \bar{S}$, entonces denotamos como $f^+(K)$ a $f(S, \bar{S})$ y a $f^-(K)$ como $f(\bar{S}, S)$.
Con esto definimos al **valor de un flujo** $val(f) = f^+(K) - f^-(K)$.

Definición 20. En una gráfica $G = (V, E)$, el **costo** c_{ij} de una arista $(i, j) \in E$ se define como una cantidad no negativa que equivale al esfuerzo de enviar una unidad de flujo por la arista (i, j).

Definición 21. El **flujo factible** v_{fact} de una red $G = (V, E)$, donde G es una gráfica dirigida con capacidades no negativas u_{ij} asociadas a cada arista $(i, j) \in E$ se define como un flujo v sujeto a las siguientes restricciones:

- $\forall (i,j) \in E : 0 \leq x_{ij} \leq u_{ij}$ donde x_{ij} es el flujo asociado a la arista $(i,j) \in E$
-

$$\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = \begin{cases} v & \text{si i=s} \\ 0 & \forall i \in V - \{s,t\} \\ -v & \text{si i=t} \end{cases}$$

Definición 22. El **flujo máximo** v_{max} de una red $G = (V, E)$ es el valor máximo dentro de todos los flujos factibles de G.

Se presenta una definición para una red residual, la cual permite medir el flujo de una red de manera incremental y será de utilidad como paso intermedio en diversos algoritmos.

Definición 23. [14] Dados una red $G = (V, E)$, una función capacidad $u : E \to Z^+$ y un flujo $x : E \to Z^+$ (donde Z^+ es el conjunto de número enteros positivos); definimos la **capacidad residual** como una función r sobre el conjunto de arcos definido mediante:

$$r : E \to Z^+ : (i,j) \to r(i,j) := r_{ij} = (u_{ij} - x_{ij}) + x_{ji}$$

Por definición $r_{ij} \geq 0$ y en términos prácticos r_{ij} representa el flujo adicional máximo que aún puede pasar a través de los arcos (i,j) y (j,i).

Definición 24. Dada una gráfica $G = (V, E)$ con capacidad u, un flujo x de $s \to t$, definimos y denotamos la **red residual** mediante:

$$G(x) = (V, \hat{E}) : \hat{E} = \{(i,j) \in E : r_{ij} > 0\}$$

La red residual $G(x)$ se construye de la siguiente manera:

1. Reemplazar cada arco $(i,j) \in E$ por 2 arcos:
   - (i,j) con costo c_{ij} y capacidad residual $r_{ij} = u_{ij} - x_{ij}$
   - (j,i) con costo $-c_{ij}$ y capacidad residual $r_{ji} = x_{ij}$
2. La red residual se formará solamente con las aristas con capacidad residual positiva

Ejemplo 5. *Obtención de una red residual.*
Dada la gráfica de la izquierda (ver figura 1.5), en la cual suponemos que las aristas que no existen

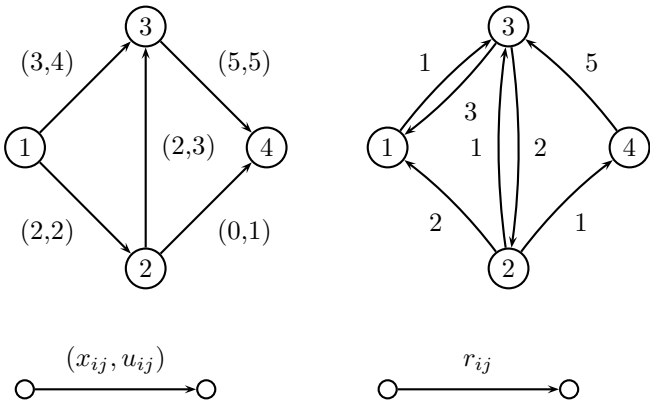

Figura 2.5: Izq: Gráfica original, Der:Gráfica residual asociada con capacidades residuales

dentro de dicha gráfica, pero que son necesarias para el cálculo de la red residual, tienen una capacidad y un flujo iguales a cero, la obtención de la gráfica residual asociada se realiza de la manera siguiente:

A partir de (1,2) $r_{12} = u_{12} - x_{12} = 2 - 2 = 0$, $r_{21} = x_{12} = 2$

A partir de (1,3) $r_{13} = u_{13} - x_{13} = 4 - 3 = 1$, $r_{31} = x_{13} = 3$

A partir de (2,3) $r_{23} = u_{23} - x_{23} = 3 - 2 = 1$, $r_{32} = x_{23} = 2$

A partir de (2,4) $r_{24} = u_{24} - x_{24} = 1 - 0 = 1$, $r_{42} = x_{24} = 0$

A partir de (3,4) $r_{34} = u_{34} - x_{34} = 5 - 5 = 0$, $r_{43} = x_{34} = 5$

Definición 25. Un **corte** de una gráfica $G = (V, E)$ denotado como $[S, \bar{S}]$ es una partición de V en 2 subconjuntos S y $\bar{S} = V - S$.

Definición 26. Un **corte s-t** de una gráfica $G = (V, E)$ denotado como $[S, \bar{S}]$ es una partición de V en 2 subconjuntos S y $\bar{S} = V - S$, tal que $s \in S$ y $t \in \bar{S}$.

Definición 27. Dado un corte $[S, \bar{S}]$, la **capacidad de un corte** es la suma de las capacidades de los arcos hacia delante en el corte[2]

$$U[S, \bar{S}] = \sum_{(i,j) \in (S, \bar{S})} u_{ij}$$

Definición 28. Dado un corte $[S, \bar{S}]$, la **capacidad residual de un corte** es la suma de las capacidades residuales de los arcos hacia delante en el corte

$$U_r[S, \bar{S}] = \sum_{(i,j) \in (S, \bar{S})} r_{ij}$$

[2]Arcos que van de una vértice $v \in S$ a un vértice $v' \in \bar{S}$

Definición 29. Un **corte mínimo entre** s **y** t es un corte cuya capacidad es mínima entre todos los posibles cortes entre s y t

Ejemplo 6. *Cortes sobre una gráfica*

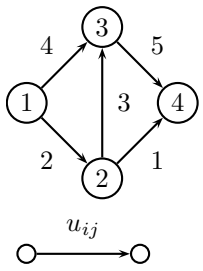

Figura 2.6: Gráfica empleada para el ejemplo de un corte dentro de una gráfica

Si definimos al nodo fuente s=1 y al nodo destino t=4 (ver figura 2.6), los posibles cortes $s-t$ *con la capacidad asociada a cada corte son los siguientes:*

S	S'	$U[S,S']$
1	*2,3,4*	*6*
1,2	*3,4*	*8*
1,3	*2,4*	*7*
1,2,3	*4*	*6*

Por lo que el corte mínimo s-t de la gráfica puede ser tanto $S = \{1\}, S' = \{2,3,4\}$ *o* $S = \{1,2,3\}, S' = \{4\}$

Definición 30. Dada una gráfica dirigida $G = (V, E)$ con costo c_{ij} y capacidad u_{ij} asociadas a cada arco $(i,j) \in A$, un **flujo de costo mínimo** $z'(x)$ se define como:

1. Minimizar $z(x) = \sum_{(i,j)\in E} c_{ij}x_{ij}$
2. Sujeto a las siguientes restricciones:
   - $\sum_{j:(i,j)\in E} x_{ij} - \sum_{j:(j,i)\in E} x_{ji} = b(i)\ \forall i \in V$
   - $0 \leq x_{ij} \leq u_{ij}\ \forall (i,j) \in E$

A cada nodo $i \in V$ se le asocia una cantidad $b(i)$ de la manera siguiente:

$$\begin{cases} b(i) > 0 & \text{si i es un nodo que provee} \\ b(i) < 0 & \text{si i es un nodo que demanda} \end{cases}$$

Definición 31. Dado un flujo f sobre una red N y dado un camino P, definimos al **incremento de flujo** como $i(e)$ y al incremento de flujo sobre el camino como $i(P)$ de la siguiente forma:
$i(P) = min_{e \in E(P)} i(e)$
$i(P) \geq 0$
$i(e) = c(e) - f(e)$ si e es una arista hacia delante en P
$i(e) = f(e)$ si e es una arista hacia atrás en P

Definición 32. Dado un flujo f sobre una red N, definimos a un **camino de incremento de flujo** como:

$$\hat{f}(e) = \begin{cases} f(e) + i(p) & \text{Si } e \text{ es un arco hacia delante} \\ f(e) - i(p) & \text{Si } e \text{ es un arco hacia delante} \\ f(e) & \text{Cualquier otro caso} \end{cases}$$

2.2.5. Propiedades de Flujos y Cortes

Teorema 3. *[15] Para cualquier flujo f y cualquier corte $K = (S, \bar{S})$ dentro de una red dada N, el valor del flujo es menor o igual que la capacidad del corte.*

$$val(f) \leq U[S, \bar{S}]$$

Demostración. Sabemos que $f^+(S) \leq U[K]$ y además $f^-(S) \geq 0$
Por lo tanto establecemos lo siguiente:
$val(f) + f^-(K) = f^+(K) \leq U[K]$
$val(f) \leq U[K] - f^-(K)$
Como $f^-(K) \geq 0$
Se cumple que $val(f) \leq U[S, \bar{S}]$ □

La multidifusión de información permite la transmisión simultánea de la misma información a múltiples destinos dentro de una red y los siguientes teoremas nos permitirán estudiar las condiciones necesarias para que dichas transmisiones permitan una taza dada de transferencia determinada. Para el caso de monodifusión de información, estas condiciones se conocen desde hace tiempo. Por ejemplo, el teorema de 'corte mínimo-flujo máximo', probado por Menger en el año de 1927, establece que el corte mínimo dentro de una red equivale al flujo máximo posible de transmisión dentro de esa misma red, teorema que extrapolándolo al terreno de la transmisión de información, resulta en que la taza máxima de comunicación en la red resulta ser equivalente también al corte mínimo, sin embargo, al contar con múltiples destinos, parecía ser que el mismo teorema era insuficiente, pero es aquí en donde la principal aportación de la codificación en redes nos muestra lo contrario.

Una definición formal de los teoremas citados es la siguiente:

Teorema 4. *[15] Un flujo f en una red dada N es máximo si y solo si N no contiene caminos de incremento de flujo.*

Teorema 5 (Corte-mínimo Flujo-máximo)**.** *[15] En cualquier red N, el valor del flujo máximo es igual a la capacidad del corte mínimo.*

Como teorema final, debido a que es uno de los más importantes resultados obtenidos dentro del área de la codificación en redes desde su aparición en el año 2000, tenemos lo siguiente:

Teorema 6. *[13] Sea $G = (V, E)$ una gráfica dirigida acíclica con capacidad unitaria asociada a cada una de sus aristas, h fuentes ubicadas dentro de un mismo vértice dentro de la gráfica y N nodos destino. Si asumimos que el valor del corte mínimo para cada nodo destino es h, entonces existe un esquema de multidifusión de información sobre un campo finito suficientemente grande $\mathbb{F}_q$, en el cual nodos intermedios de la red efectúan combinaciones lineales de los paquetes recibidos dentro de $\mathbb{F}_q$, que envía la información de las fuentes de manera simultánea a cada uno de los nodos destino con una taza igual a h.*

Gracias al teorema de corte-mínimo flujo-máximo, sabemos que existen exactamente h caminos de aristas disjuntas entre las fuentes y cada uno de los nodos destino. Es por ello que si cualquier destino, digamos R_j, se encuentra utilizando la red por sí solo, la información de cada una de las h fuentes puede ser enrutada hacia R_j a través de un conjunto de h caminos de aristas disjuntas. Sin embargo, cuando múltiples destinos utilizan la red de manera simultánea, sus conjuntos de caminos pueden traslaparse.
Es aquí donde la aportación del último teorema toma relevancia, pues establece que si se permite a nodos intermedios de la red combinar sus flujos de información entrantes, entonces cada uno de los destinos será capaz de recibir información a una taza equivalente a la del escenario en donde utiliza un solo destino la red.

Adicionalmente el teorema establece que es suficiente para los nodos intermedios efectuar operaciones lineales, es decir, adiciones y multiplicaciones sobre un campo finito $\mathbb{F}_q$, a lo que se le denomina codificación lineal de redes.

Capítulo 3

Algoritmos y Codificación en Redes

Dentro de la siguiente sección, se presenta el funcionamiento de dos algoritmos. El primero permite la obtención de un código de red factible para una gráfica dada y el segundo es un algoritmo distribuido para la obtención de un conjunto de caminos de aristas disjuntas entre un par de vértices. Es importante considerar que este par de algoritmos, requieren de una cantidad considerable de algoritmos y procedimientos adicionales, los cuales se presentan igualmente dentro de esta sección con el afán de contar con un panorama completo del funcionamiento de las aportaciones principales provistas dentro de este documento.

El algoritmo para la obtención de un código de red de manera eficiente, funciona *grosso modo* como a continuación se menciona:

El algoritmo resuelve un problema de codificación de una red [1], por lo que a partir de una red acíclica $\mathbb{N}(G, s, T, h)$, donde G es una gráfica dirigida, s es el nodo fuente, T es el conjunto de nodos destino y h es el número de paquetes que se desean enviar de s a cada $t \in T$, obtiene una codificación de red [2] mediante el siguiente procedimiento:

1. Construir una red auxiliar $\hat{\mathbb{N}}(\hat{G}, s, T, h)$ de manera que cuente con un número acotado de nodos, lo que permite la obtención de un límite sobre el número de nodos de codificación
2. Encontrar un código de red factible $\hat{\mathbb{F}}(\hat{\mathbb{N}})$ para $\hat{\mathbb{N}}$ (Aplicando el algoritmo de [2])
3. Encontrar un código de red $\mathbb{F}(\mathbb{N})$ para $\mathbb{N}$ que corresponda a $\hat{\mathbb{F}}(\hat{\mathbb{N}})$

El segundo algoritmo principal presentado en este documento, resuelve el problema de obtención de un conjunto de caminos de aristas disjuntas entre un par de vértices dentro de una gráfica, para

[1] Ver problema 2
[2] Ver definición 13

ello, se presenta un algoritmo que emplea codificación en redes para la resolución del problema y se compara con 2 algoritmos distribuidos que resuelven el problema definido con antelación (ver problema 1):

Algoritmo 1: Cada nodo de la red envía información sobre su vecindad al nodo fuente, una vez que el nodo fuente ha recibido información de cada uno de sus vecinos, ejecuta de manera local un algoritmo centralizado que resuelve el problema de encontrar h caminos disjuntos entre s y t, finalmente propaga la solución encontrada por el algoritmo centralizado por toda la red, de manera tal que cada nodo de la red conozca la solución al problema.

Algoritmo 2: Se basa en el algoritmo centralizado para encontrar pares de caminos disjuntos [3], el cual consta de 3 etapas:

- Encontrar un árbol de caminos más cortos con nodo raíz en el nodo fuente del problema s
- Transformar cada arista de manera que 2 caminos con el mismo vértice de inicio y final, sean modificados por la misma cantidad, lo que permitirá que el ordenamiento de dichos caminos de acuerdo a su longitud no se vea afectado por la transformación
- Obtener los caminos disjuntos en la gráfica con las longitudes modificadas

Algoritmo 3: Este algoritmo emplea Codificación en Redes para resolver el problema. Inicialmente se ejecuta un algoritmo distribuido para obtener un código de red, una vez que el nodo destino ha recibido h paquetes diferentes, envía mensajes para el descubrimiento de los caminos disjuntos hacia el nodo fuente.

A continuación, se presenta una descripción detallada de cada uno de los algoritmos mencionados previamente:

3.1. Algoritmo para obtener un código de red factible (Definición 14)

El algoritmo emplea 3 redes auxiliares con las propiedades mencionadas a continuación:

- $\mathbb{N}'(G', s, T, h)$
  - Todas las aristas en G' tienen capacidad unitaria
  - El número total de aristas en G' está acotado por $|V|hk$

- Cada nodo interno [3] $v \in G'$ tiene grado de entrada o de salida 1 [4]
- Cualquier conjunto de caminos con aristas disjuntas en G' es también de nodos disjuntos

- $\mathbb{N}^*(G^*, s, T, h)$
  - Se obtiene a partir de G' eliminando las aristas redundantes de manera que el número de nodos con grado de entrada mayor a 2 en G^* está acotado por $O(h^3k^2)$
- $\hat{\mathbb{N}}(\hat{G}, s, T, h)$
  - Se obtiene a partir de G^* contrayendo todos los nodos de grado 2 con lo que el número de aristas está acotado por $O(h^3k^2)$

Para la obtención del código de red para $\mathbb{N}$ se llevan a cabo los siguientes pasos:

1. Construir una red auxiliar $\hat{\mathbb{N}}(\hat{G}, s, T, h)$
2. Encontrar un código de red factible $\hat{\mathbb{F}}(\hat{\mathbb{N}})$ para $\hat{\mathbb{N}}$ (Aplicando el algoritmo de [2])
3. Encontrar un código de red $\mathbb{F}(\mathbb{N})$ para $\mathbb{N}$ que corresponda a $\hat{\mathbb{F}}(\hat{\mathbb{N}})$

3.1.1. Construcción de la red auxiliar $\hat{\mathbb{N}}(\hat{G}, s, T, h)$

El algoritmo para el primer punto consta de 4 procedimientos:

1. *Expand:* Encuentra para cada terminal un flujo de costo mínimo (ver definición 30), lo que permite la obtención de h caminos de aristas disjuntas entre el nodo fuente y cada uno de los destinos $t_i \in T$. Este conjunto de h caminos de aristas disjuntas entre s y t_i será nombrado como $\mathbb{P}_i$. Además, para asegurar que en la codificación se empleará el menor número posible de aristas, se sustituye cada nodo interno v de grado mayor a 3 por un objeto Γ_v.
   Este objeto Γ_v se construye de la siguiente manera:
   Sea E_v^{in} y E_v^{out} las aristas entrantes y salientes de v respectivamente.
   - Por cada arista $(x, v) \in E_v^{in}$ agregar un nodo x' a Γ_v y una arista (x, x')
   - Por cada arista $(v, y) \in E_v^{out}$ agregar un nodo y' a Γ_v y una arista (y', y)
   - Por cada camino $P_i \in \{\mathbb{P}_i | t_i \in T\}$, sea x' el nodo de Γ_v que corresponde a la arista $(x, v) \in P_i$ y y' el nodo en Γ_v que corresponde a la arista $(v, y) \in P_i$. Si Γ_v no incluye a la arista (x', y'), agregarla

[3] nodos $v \in V : v \neq s, v \notin T$

[4] El grado de un nodo (vértice) es el número de aristas conectadas a él. El $grado^+$ o grado de entrada de un nodo v es el número de aristas $(u, v) \in E$ y el $grado^-$ o grado de salida, es el número de aristas $(v, u) \in E$

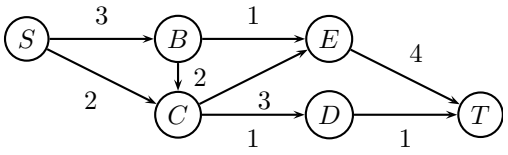

Figura 3.1: Substitución del nodo v por el objeto Γ_v

Este objeto permite que la gráfica generada cumpla con la propiedad de que cada nodo $v \in G'$ tiene grado de entrada o de salida 1, con lo que cada camino de aristas disjuntas será también de nodos disjuntos.

2. *Min-local:* Encuentra 2 conjuntos de h nodos disjuntos con la cualidad de que la subgráfica inducida por dichos conjuntos es mínima, es decir que la remosión de cualquier arista de la subgráfica impediría la codificación.

3. *Min-global:* Asegura un número de conexiones acotado dentro de la gráfica producida por $O(h^3k^2)$ donde h es el número de paquetes que se desean transmitir y k es el número de destinos a los cuales se les requiere enviar los paquetes

4. *Shrink:* Simplifica la red, formando otra mediante la contracción de los nodos de grado 2, específicamente aquellos que cuenten con grado de entrada 1 y grado de salida 1.

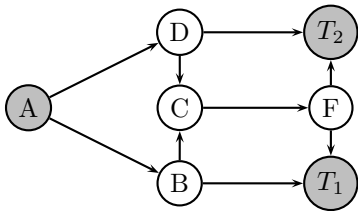

Figura 3.2: *Contracción de nodos*: El nodo v al contar con grado de entrada y salida 1, se torna irrelevante su análisis para la obtención de la codificación de la red original, por lo que es removido de la gráfica final.

3.1.2. Código Factible

Para el segundo punto basta con obtener una codificación para la red mediante la ejecución del algoritmo presentado en [2].

Este algoritmo, denominado LIF (Flujo de Información Lineal por sus siglas en inglés), cuenta como idea principal la preservación de la propiedad de multidifusión de información dentro de la red, la cual establece que el corte mínimo entre el nodo fuente s y cada nodo destino $t_i \in T$ es mayor o igual que h.

Para llevar a cabo este propósito, se seleccionan vectores de codificación que preserven la propiedad mencionada, visitando los nodos de codificación en orden topológico, es decir, asegurando que ningún nodo de codificación sea visitado antes que cualquiera de sus antecesores en la gráfica, de manera que el algoritmo preserva h mensajes linealmente independientes en cada conojunto de caminos hacia los nodos destino.

Se requiere de notación adicional para describir de manera más precisa el algoritmo, tarea que a continuación se llevará a buen término.
Sea $\mathfrak{C}$ el conjunto de nodos de codificación.
$T(\delta)$ el conjunto de nodos destino que emplean al nodo de codificación δ en alguno de sus caminos.
Cada nodo de codificación δ aparece en a lo más en un camino para cada destino t_i.
Sea $f^j_{\leftarrow}(\delta)$ el nodo predecesor del nodo de codificación δ a través del camino hacia t_j.

Para cada destino t_j, el algoritmo mantiene un conjunto $\mathfrak{C}_j$ de h puntos de codificación y un conjunto $B_j = \{c^j_1, ..., c^j_h\}$ de h vectores de codificación.
El conjunto $\mathfrak{C}_j$ mantiene una referencia al punto de codificación más recientemente visitado a través de cada uno de los h caminos desde el nodo fuente s hasta el destino t_j y el conjunto B_j almacena los vectores de codificación asociados.

Inicialmente, para todos los nodos destino t_j, el conjunto $\mathfrak{C}_j$ contiene a los nodos fuente de cada uno de los mensajes y el conjunto B_j contiene la base ortonormal $\{e_1, e_2, ..., e_h\}$, donde el vector e_i contiene un uno en la posición i y cero en el resto.

En la k-ésima iteración, el algoritmo asigna un vector de codificación $c(\delta_k)$ al nodo de codificación δ_k y reemplaza para todos los destinos $t_j \in t(\delta_k)$ lo siguiente:

- Al punto $f^j_{\leftarrow}(\delta)$ en $\mathfrak{C}_j$ con el punto δ_k
- Al vector asociado $c(f^j_{\leftarrow}(\delta))$ en B_j con $c(\delta_k)$

El algoritmo selecciona al vector $c(\delta_k)$ de manera que para cada destino $t_j \in t(\delta_k)$, el conjunto $(B_j/\{c(f^j_{\leftarrow}(\delta))\}) \cup c(\delta_k)$ forme una base. Algo a resaltar es que el vector $c(\delta_k)$ existe si el campo

finito empleado $\mathbb{F}_q$ tiene un tamaño mayor al número de destinos.

Cuando el algoritmo termina, el conjunto B_j contiene al conjunto de ecuaciones lineales que el nodo t_j necesita resolver para obtener los símbolos contenidos en los mensajes originales.

3.1.3. Código correspondiente a la red original

Finalmente, para encontrar un código $F^*(N^*)$, se procede de la manera siguiente:
Una vez ejecutado el algoritmo $Min-Global$, cada arista $e \in \hat{G}$ corresponde a un camino P en G^*. Para obtener F^*, la primera arista de P tiene la misma función de codificación que e en $\hat{F}$, el resto de los nodos en P solamente reenvían paquetes.
Para encontrar una codificación en la red original, analizaremos los procedimientos restantes:
Al ejecutar Expand, existen 2 tipos de aristas, las removidas y las sustituidas por Γ_v. Si la arista fue removida su función de codificación tendrá como resultado el elemento cero del campo finito Σ. Si el nodo v no fue reemplazado por Γ_v entonces todas las aristas $e(v,u)$ tendrán la misma codificación que en F^*.
Para el caso en el que v fue sustituido por Γ_v, definimos a $e' = (u', u)$ como la arista en G' que corresponde a $e(v,u)$. Un paquete enviado a través de e' tendrá la misma codificación que e.

A continuación se presentan los procedimientos empleados por el algoritmo:

Expand: Dada una gráfica G, un nodo fuente s, un conjunto de nodos destino T, y h el número de paquetes que se desean enviar de s a cada nodo destino, encuentra para cada nodo destino $t_i \in T$ un flujo de costo mínimo $s - t_i$ denominado θ

Algoritmo: Expand ($\mathbb{N}(G, s, T, h)$)

Asignar costo unitario $\forall e \in E$: $c_e = 1$;

foreach $t_i \in T$ **do**

  Encontrar un flujo θ_i de costo mínimo de $s - t_i$ de valor h;

end

foreach $e(u, v) \in E$ **do**

  if $MAX_{t_i \in T}\theta_i(e) > 0$ **then**

    Reemplazar e por $MAX_{t_i \in T}\theta_i(e)$ conexiones paralelas de capacidad unitaria $u - v$;

  else

    Remover e;

  end

end

foreach $t_i \in T$ **do**

  Encontrar un conjunto $\mathbb{P}_i$ de h caminos disjuntos entre s y t_i;

end

foreach $v \in V : v \neq s \& v \notin T \& grado(v) > 3$ **do**

  Reemplazar v por Γ_v;

end

La gráfica resultante es G';

return $\mathbb{N}' = (G', s, T, h)$

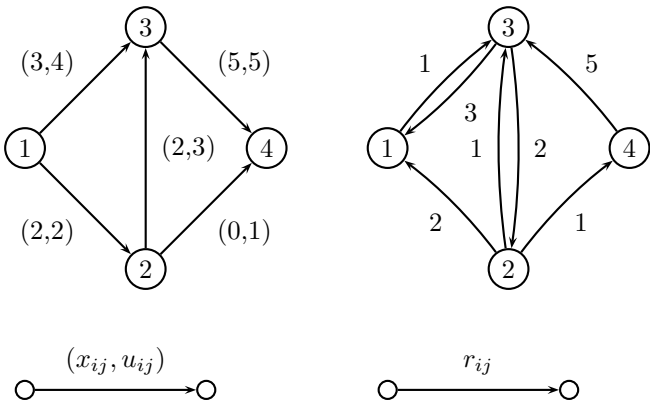

Figura 3.3: Gráfica obtenida después de ejecutar el algoritmo Expand (derecha) a partir de G (izquierda)

Demostración. Mostraremos que las propiedades de $\mathbb{N}'$ se cumplen al ejecutarse el algoritmo Expand:

- Todas las aristas en G' tienen capacidad unitaria
  En la línea 8 del algoritmo *Expand*, toda arista $e \in G$ es reemplazada por un conjunto de aristas $e \in G'$ de capacidad unitaria, con lo que aseguramos el cumplimiento de la propiedad.

- El número total de aristas en G' está acotado por $|V|hk$
  En la línea 14 del algoritmo *Expand*, se obtiene para cada terminal $t \in T$ un conjunto de h caminos disjuntos, como cada camino disjunto es un camino simple (sin nodos repetidos), el número de nodos en cada uno de ellos es a lo más $|V|$ y al no existir ciclos, el número de aristas es $O(|V|)$. Como existen h caminos disjuntos por cada terminal, el número de aristas en cada terminal está acotado por $O(|V| \cdot h)$ y al existir k terminales, las aristas en G' están acotadas por $O(|V|hk)$.

- Cada nodo interno $v \in G'$ tiene grado de entrada o de salida 1
  En la línea 16 del algoritmo *Expand*, todo nodo interno $v \in G$ con $grado(v) > 3$, es substituido en G' por un objeto Γ_v, el cual se construye de la siguiente manera:
  Sea E_v^{in} las conexiones entrantes a v y E_v^{out} las conexiones salientes de v.
  $\forall e = (x, v) \in E_v^{in}$ agregar un nodo x' a Γ_v y una arista (x, x')
  $\forall e = (v, y) \in E_v^{out}$ agregar un nodo y' a Γ_v y una arista (y', y)
  $\forall P_i \in \{\mathbb{P} | t_i \in T\}$, x' agregado por (x,v), y' agregado por (v,y), si Γ_v no tiene (x',y'), agregarlo.
  Por lo anterior, todos los nodos internos en G son substituidos por un objeto Γ dentro del cual los nodos x' tienen grado de entrada 1 y los nodos y' tienen grado de salida 1, por lo que la propiedad se cumple.

- Cualquier conjunto de caminos con aristas disjuntas en G' es también de nodos disjuntos
  Esta propiedad se deriva de la anterior ya que 2 caminos con aristas disjuntas no pueden compartir un nodo interno puesto que los nodos tienen grado de salida o entrada 1, lo que implicaría que comparten aristas llegando a una contradicción.

□

Min-local: Encuentra 2 conjuntos de h nodos disjuntos $\mathbb{P}_i^*$ y $\mathbb{P}_j^*$ que conectan $s - t_i$ y $s - t_j$ con la cualidad de que G_{ij}^*, la subgráfica inducida por $\mathbb{P}_i^*$ y $\mathbb{P}_j^*$ es mínima.

Para la demostración de las propiedades de las gráficas auxiliares subsecuentes, se emplearán los siguientes lemas:

Algoritmo: Min-Local

Sea G_{ij} la subgráfica de G inducida por las aristas de $\mathbb{P}_i$ y $\mathbb{P}_j$;
Asignar costo 0 a cada arista de $\mathbb{P}_i$ y costo 1 al resto de las conexiones;
Encontrar h caminos disjuntos $\mathbb{P}_j^*$ en G_{ij} entre $s - t_j$ con costo mínimo;
Sea G'_{ij} la subgráfica de G inducida por las aristas de $\mathbb{P}_i$ y $\mathbb{P}_j^*$;
Asignar costo 0 a cada arista de $\mathbb{P}_j^*$ y costo 1 al resto de las conexiones;
Encontrar h caminos disjuntos $\mathbb{P}_i^*$ en G'_{ij} entre $s - t_i$ con costo mínimo;
return $\mathbb{P}_i^*$ *y* $\mathbb{P}_j^*$

Figura 3.4: Algoritmo Min-Local

Lema 1. *[16] Sea $\mathbb{N}(G, s, T, h)$ una codificación de una red acíclica tal que $|T| = 2$, todas las aristas en G tienen capacidad unitaria, el grado total de cada nodo interno en G es a lo más 3 y G es mínima con respecto a remoción de aristas, entonces el número de nodos internos en G con grado de entrada mayor a 1 está acotado por h^3 y el número de nodos internos en G con grado de salida mayor a 1 está acotado por $h^3 + h$.*

Lema 2. *[11] Sea G_{ij}^* la subgráfica de G inducida por los conjuntos de caminos $\mathbb{P}_i^*$ y $\mathbb{P}_j^*$ y sea $\mathbb{N}_{ij}^*$ la codificación para la gráfica G_{ij}^* y las 2 terminales $\{t_i, t_j\}$. Entonces todas las aristas en $\mathbb{N}_{ij}^*$ son vitales.*

A partir de los cuales se deriva el siguiente teorema:

Teorema 1. *[11] Sea $\mathbb{N}^*(G^*, s, T, h)$ el problema de codificación obtenido por el algoritmo Min-Global, sea $\bar{V}^*$ el subconjunto de V^*/T que incluye a los nodos de grado de entrada ≥ 2 y sea $\bar{E}^*$ el conjunto de aristas que llegan a los nodos en $\bar{V}^*$. Entonces se cumple que $|\bar{E}^*| \in O(h^3k^2)$*

Demostración. Sea $E_i = \bigcup_{j=1}^{i} E(\mathbb{P}_j)$ donde $E(\mathbb{P}_i)$ es el conjunto de aristas de $\mathbb{P}_i$, la demostración del teorema anterior se realizará por inducción sobre i.
Caso Base: Cuando i=2, $|E_2|$ está acotado por h^3 empleando el lema 1
Inducción: Cuando i=k, probaremos que $|E_i| \leq h^3ki$. Sea $E_i^1 = E_i \cap E_{i-1}$ y $E_i^2 = E_i \setminus E_i^1$, por h.i. sabemos que $|E_i^1|$ está acotado por $h^3k(i-1)$, por lo que solamente se requiere encontrar una cota para E_i^2.
Para ello, definimos $E_{ij} = E(\mathbb{P}_i) \cup E(\mathbb{P}_j)$, empleando el lema 1 y 2, sabemos que E_{ij} está acotado por h^3, ahora probaremos que cada arista en $E_i^2 \in E_{ij}$ para $j \in \{1, ..., i-1\}$.
Existen 2 posibilidades para cada arista $e(u, v) \in E_i^2$:

1. $e \in E_i \setminus E_{i-1}$, lo que implica que $e \in E(\mathbb{P}_i)$, por lo que se encuentra en E_{ij}

```
1  Algoritmo: Min-Global
2  ℙ = φ;
3  for i=1 to k do
4  |  Asignar costo 0 a todas las conexiones de ℙ;
5  |  Asignar costo 1 al resto de las conexiones en G';
6  |  Encontrar un conjunto de h caminos con aristas disjuntas ℙ_i en G' entre s − t_i de costo
   |  mínimo;
7  |  if i > 1 then
8  |  |  for j=1 to i-1 do
9  |  |  |  ℙ_i, ℙ_j = Min_Local(ℕ', ℙ_i, ℙ_j);
10 |  |  end
11 |  end
12 |  ℙ = ⋃_{j=1}^{i} ℙ_j y E_i = ⋃_{j=1}^{i} E(ℙ_j);
13 end
14 G* es la subgráfica de G' inducida por las conexiones en E_k;
15 return ℕ*(G*, s, T, h)
```

Figura 3.5: El algoritmo asegura que el número de conexiones en G^* esté acotado por $O(h^3k^2)$

2. $e \in E_{i-1}$, existe otra arista $e'(w, v)$, ya que v tiene grado de entrada 2 y $e' \in E_i \setminus E_{i-1}$, por lo que e pertenece a E_{ij}

□

Para el algoritmo Shrink, únicamente se requiere probar que el número de aristas en N está acotado por $O(h^3k^2)$, por lo que el teorema 7 mostrado con antelación es suficiente.

Una vez analizado el funcionamiento del algoritmo para la obtención de un código de red factible, se presentan diversos algoritmos empleados de manera conjunta con el algoritmo antes mencionado.

3.1.4. Algoritmo para la descomposición de un flujo

El algoritmo descompone un flujo x factible en una gráfica dirigida, en un conjunto de flujos con nodo origen en s y nodo destino en t.
Recibe como parámetros de entrada, una gráfica dirigida, un nodo origen s, un nodo destino t y una función para el flujo $f : \forall e \in E \rightarrow \mathbb{Z}^+$ y obtiene como resultado un conjunto de caminos desde

Algoritmo: Shrink
foreach $v \in G^*$ **do**
  if $grado^+(v) = grado^-(v) = 1$ **then**
    Substituir (u,v) y (v,w) por (u,w) en $\hat{G}$;
  end
end
return $\hat{\mathbb{N}}(\hat{G}, s, T, h)$

Figura 3.6: El algoritmo forma una red auxiliar contrayendo nodos de grado 2

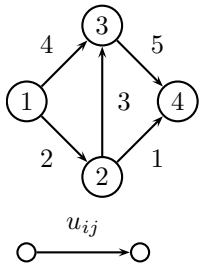

Figura 3.7: Gráfica obtenida después de ejecutar el algoritmo Shrink y haber encontrado un código factible (izquierda) y codificación final de la gráfica original G (derecha)

s hacia t que transportan al flujo x.

Demostración. Para la demostración de correctez del algoritmo, nos basamos en la veracidad del siguiente teorema:

Teorema 2 (Flow Decomposition Theorem)**.** *[1] Cada camino de flujo tiene una representación única como un conjunto de arcos con flujos no negativos y cada flujo sobre arcos de flujos no negativos puede ser representado como un camino de flujo con las siguiente propiedad:*
Cada camino dirigido con flujo positivo conecta un nodo déficit con un nodo exceso.

□

```
1  Algoritmo: Flow Decomposition(G,s,t,f)
2  foreach i ∈ N do
3  |  e(i) = Σ_{j:(j,i)∈A} x_ji − Σ_{j:(i,j)∈A} x_ij;
4  end
5  while ∀n ∈ N : e(n) = 0 do
6  |  P.add(s);
7  |  i=s;
8  |  while ∃(i, j) con flujo positivo do
9  |  |  P.add(j);
10 |  |  i = j;
11 |  end
12 |  k=P.last();
13 |  if k = t then
14 |  |  caminos.add(P);
15 |  end
16 |  f(P) = min{−e(s), e(k), min{x_ij : (i, j) ∈ P}};
17 |  e(s) = e(s) + f(P);
18 |  e(k) = e(k) − f(P);
19 |  foreach (i, j) ∈ P do
20 |  |  x_ij = x_ij − f(P);
21 |  end
22 |  P.clear();
23 end
24 return caminos
```

Figura 3.8: Algoritmo para la descomposición de un flujo en caminos de aristas disjuntas

3.1.5. Algoritmo para obtención de flujos de costo mínimo[1]

Dada una gráfica dirigida $G = (V, E)$, con un costo c_{ij} y una capacidad u_{ij} asociadas a cada arista $(i, j) \in E$, nodos fuente y nodos destino, el algoritmo obtiene los flujos de costo mínimo factibles dentro de la gráfica dada.

El algoritmo emplea los siguientes valores tanto para la demostración de su validez como para su funcionamiento:
Potencial de nodo: Es un número real $\pi(i)$ asociado a cada nodo $i \in V$
Costo reducido del arco (i, j)**:** $c_{ij}^{\pi} = c_{ij} - \pi(i) + \pi(j)$
Pseudoflujo: Es una función $x : E \to \mathbb{R}^+$ que satisface las restricciones siguientes:

- *Capacidad:* $0 \leq x_{ij} \leq u_{ij}$
- *No negatividad:* Dado el flujo x, la red residual $G(x)$ no contiene ciclos de costo negativo

Inequilibrio: $\forall$ pseudoflujo x, el inequilibrio del nodo i está definido por:

$$e(i) = b(i) + \sum_{j:(j,i)\in E} x_{ji} - \sum_{j:(i,j)\in E} x_{ij} \forall i \in V$$

El inequilibrio cumple con lo siguiente:

$$\begin{cases} e(i) > 0 & \text{si existe un exceso en } i \\ e(i) < 0 & \text{si existe un déficit en } i \\ e(i) = 0 & \text{si } i \text{ está balanceado} \end{cases}$$

Algoritmo: Successive Shortest Path

$x = 0$;

$\pi = 0$;

$\forall i \in V \ e(i) = b(i)$;

$Ex = \{i : e(i) > 0\}$;

$D = \{i : e(i) < 0\}$;

while $Ex \neq 0$ **do**

  Seleccionar un nodo $k \in Ex$ y un nodo $l \in D$;

  Determinar las distancias más cercanas $d(j)$ de s al resto de los nodos de $G(x)$ con respecto a los costos reducidos c_{ij}^{π};

  Definir P el camino más pequeño de k a l;

  Actualizar $\pi = \pi - d$;

  Definir $\delta = min[e(k), -e(l), min\{r_{ij} : (i,j) \in P\}]$;

  Aumentar en δ unidades el flujo de P;

  Actualizar $x, G(x), Ex, D$ y los costos reducidos c_{ij}^{π};

end

Ejemplo 1. *Ejecución del Algoritmo "Successive Shortest Path"*

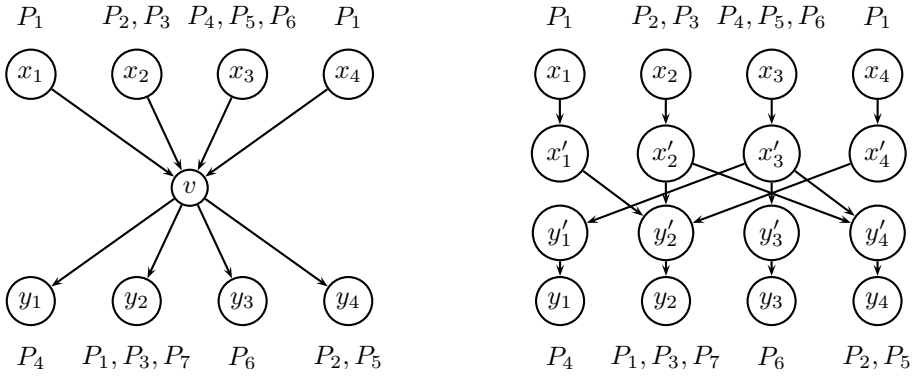

Figura 3.9: Gráfica inicial sobre la cual se ejecuta el algoritmo de Caminos más cortos

1. *Inicialización*

   El nodo 1 provee 4 paquetes a la red y el nodo 4 requiere de estos 4 paquetes, por lo que se tienen los siguientes valores:

   $b(1) = 4, b(4) = -4, b(2) = b(3) = 0$

$x = 0$; *//Inicializamos el pseudoflujo con valor de 0 en cada arista* $(i, j) \in A$

$x_{12} = x_{13} = x_{23} = x_{24} = x_{34} = 0$

$\pi(i) = 0$ *//Inicializamos el potencial de cada nodo con valor 0*

$\pi(1) = \pi(2) = \pi(3) = \pi(4) = 0$

Nota: Los valores $x = 0$ *y* $\pi(i) = 0$ *permiten que G(x=0) tenga los mismos valores que G.*

$E = \{1\}$ *y* $D = \{4\}$

2. *Iteración 1*

Seleccionamos $k = 1$ *y* $l = 4$

Obtenemos distancias de 1 al resto de los nodos $= \{0, 2, 2, 3\}$

$P = \{1 - 3 - 4\}$

Actualizamos los potenciales de cada nodo:

$\pi(1) = 0 - 0 = 0$

$\pi(1) = 0 - 2 = -2$

$\pi(1) = 0 - 2 = -2$

$\pi(1) = 0 - 3 = -3$

Calculamos $\delta = min[\{e(1), -e(4), min\{r_{ij} : (i, j) \in P\}]$

$\delta = min[4, 4, min\{2, 5\}] = 2$

Aumentamos 2 unidades de flujo a lo largo de P y actualizamos $x, G(x), E, D, e, c_{ij}^{\pi}$.

Para los valores de flujo:

$x_{12} = x_{23} = x_{24} = 0$ *ya que no han sido modificados.*

$x_{13} = x_{34} = 2$ *ya que se aumentan 2 unidades de flujo en P.*

Para los valores de inequilibrio:

$e(1) = b(1) + 0 - [x_{12} + x_{13}] = 4 - 2 = 2$

$e(2) = b(2) + [x_{12}] - [x_{23} + x_{24}] = 0$

$e(3) = b(3) + [x_{23} + x_{13}] - [x_{34}] = 0 + 2 - 2 = 0$

$e(4) = b(4) + [x_{24} + x_{34}] - 0 = -4 + 2 = -2$

Por lo que $E = \{1\}$ *y* $D = \{4\}$.

Para los valores de costo reducido:

$c_{12}^{\pi} = c_{12} - \pi(1) + \pi(2) = 2 - 0 - 2 = 0$

$c_{13}^{\pi} = c_{13} - \pi(1) + \pi(3) = 2 - 0 - 2 = 0$

$c_{23}^{\pi} = c_{23} - \pi(2) + \pi(3) = 1 + 2 - 2 = 1$

$c_{24}^{\pi} = c_{24} - \pi(2) + \pi(4) = 3 + 2 - 3 = 2$

$c_{34}^{\pi} = c_{34} - \pi(3) + \pi(4) = 1 + 2 - 3 = 0$

Para los valores de la red residual:

$r_{12} = u_{12} - x_{12} = 4 - 0 = 0$,$r_{21} = x_{12} = 0$

$r_{13} = u_{13} - x_{13} = 2 - 2 = 0, r_{31} = x_{13} = 2$

$r_{23} = u_{23} - x_{23} = 2 - 0 = 2, r_{32} = x_{23} = 0$

$r_{24} = u_{24} - x_{24} = 3 - 0 = 3, r_{42} = x_{24} = 0$

$r_{34} = u_{34} - x_{34} = 5 - 2 = 3, r_{43} = x_{34} = 2$

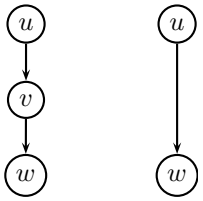

Figura 3.10: Gráfica después de la primera iteración

3. *Iteración no. 2*

Seleccionamos $k = 1$ *y* $l = 4$

Obtenemos distancias de 1 al resto de los nodos $= \{0, 0, 1, 1\}$

$P = \{1 - 2 - 3 - 4\}$

Actualizamos los potenciales de cada nodo:

$\pi(1) = 0 - 0 = 0$

$\pi(1) = -2 - 0 = -2$

$\pi(1) = -2 - 1 = -3$

$\pi(1) = -3 - 1 = -4$

Calculamos $\delta = min[2, 2, min\{4, 2, 3\}] = 2$

Aumentamos 2 unidades de flujo a lo largo de P *y actualizamos* $x, G(x), E, D, e, c_{ij}^{\pi}$.

Para los valores de flujo:

$x_{13} = 2, x_{24} = 0$ *ya que no han sido modificados.*

$x_{12} = 2, x_{23} = 2, x_{34} = 4$ *ya que se aumentan 2 unidades de flujo en* P.

Para los valores de inequilibrio:

$e(1) = b(1) + 0 - [x_{12} + x_{13}] = 4 - [2 + 2] = 0$

$e(2) = b(2) + [x_{12}] - [x_{23} + x_{24}] = 0 + 2 - 2 = 0$

$e(3) = b(3) + [x_{23} + x_{13}] - [x_{34}] = 0 + [2 + 2] - 4 = 0$

$e(4) = b(4) + [x_{24} + x_{34}] - 0 = -4 + [0 + 4] = 0$

Por lo que $E = \Phi$ *y* $D = \Phi$.

Para los valores de costo reducido:

$c_{12}^{\pi} = c_{12} - \pi(1) + \pi(2) = 2 - 0 - 2 = 0$

$c_{13}^{\pi} = c_{13} - \pi(1) + \pi(3) = 2 - 0 - 2 = 0$

$c^{\pi}_{23} = c_{23} - \pi(2) + \pi(3) = 1 + 2 - 3 = 0$

$c^{\pi}_{24} = c_{24} - \pi(2) + \pi(4) = 3 + 2 - 4 = 1$

$c^{\pi}_{34} = c_{34} - \pi(3) + \pi(4) = 1 + 3 - 4 = 0$

Para los valores de la red residual:

$r_{12} = u_{12} - x_{12} = 4 - 2 = 2, r_{21} = x_{12} = 2$

$r_{13} = u_{13} - x_{13} = 2 - 2 = 0, r_{31} = x_{13} = 2$

$r_{23} = u_{23} - x_{23} = 2 - 2 = 0, r_{32} = x_{23} = 2$

$r_{24} = u_{24} - x_{24} = 3 - 0 = 3, r_{42} = x_{24} = 0$

$r_{34} = u_{34} - x_{34} = 5 - 4 = 1, r_{43} = x_{34} = 4$

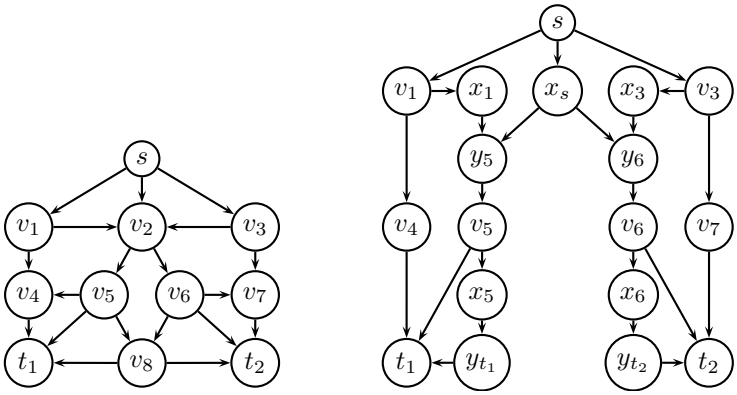

Figura 3.11: Gráfica después de la segunda iteración

Por lo que el flujo de costo mínimo es:

$x_{12} = 2, x_{13} = 2, x_{23} = 2, x_{24} = 0, x_{34} = 4$

3.1.6. Algoritmo para obtener un código de red factible[2]

Dada una gráfica dirigida $G = (V, E)$, un nodo fuente s y un conjunto de nodos destino T, se define h como el valor del corte mínimo entre s y cualquier nodo $t \in T$. El algoritmo para la obtención de un flujo lineal de información[5], construye códigos lineales para cada uno de los vértices en un tiempo $O(|E| \cdot |T| \cdot h^2)$, contando dicho código con las siguientes propiedades:

- Cualquier campo finito $\mathbb{F} \geq 2 \cdot |T|$ puede ser empleado para representar los símbolos de la codificación
- El nodo fuente recibe h símbolos como entrada

[5]LIF: Linear Information Flow

- Cada nodo destino puede reconstruir los h símbolos requeridos en tiempo $O(h^2)$

Algoritmo: LIF(V,E,s,T)

$h = min_{t \in T} min\{|C| : \text{c es un corte } s - t\}$;

Insertar nodo s' en V;

Insertar h aristas paralelas $\{e_1, e_2, ..., e_h\}$ de $s' \to s$ en E;

foreach i:$b(e_i) = [0^{i-1}, 1, 0^{h-i}]$ **do**

  foreach $t \in T$ **do**

    $C_t = \{e_1, ..., e_h\}$;

    $B_t = \{b(e_1), ..., b(e_h)\}$;

    foreach $c \in C_t$ **do**

      $a_t(c) = b(c)$;

    end

  end

end

foreach $v \in V - \{s'\}$ *en orden topológico* **do**

  foreach $e = (v, u)$ **do**

    Escoger una combinación lineal $b(e) = \sum_{p \in P(e)} m_e(p) b(p)$ tal que
    $\forall t \in T(e) : (B_t - \{b(f^t(e))\}) \cup \{b(e)\}$ sea linealmente independiente;

    foreach $t \in T(e)$ **do**

      $C'_t = (C_t - \{f^t(e)\}) \cup \{e\}$;

      $B'_t = (b_t - \{b(f^t(e))\}) \cup \{b(e)\}$;

      $a'_t(e) = (b(e) \cdot a_t(f^t(e)))^{-1} a_t(f^t(e))$;

      foreach $c \in C_t - \{f^t(e)\}$ **do**

        $a'_t(c) = a_t(c) - (b(e) \cdot a_t(c)) a'_t(e)$;

      end

      $(C_t, B_t, a_t) = (C'_t, B'_t, a'_t)$;

    end

  end

end

return $(h, \{m_e : e \in E\}, \{(C_t, a_t) : t \in T\}, \mathbb{F})$;

Figura 3.12: Algoritmo para la obtención de una Codificación en Redes a partir de la gráfica dada

3.2. Algoritmo Distribuido para Obtención de h Caminos Disjuntos

3.2.1. Modelo

Para este problema contamos con una colección de elementos de cómputo, los cuales serán representados mediante vértices en una gráfica dirigida acíclica $G = (V, E)$. Dentro de esta gráfica, deberá existir al menos un camino entre el nodo fuente s y cualquier otro nodo de la gráfica. Denotamos por n al número de nodos de la red, es decir, $|V| = n$, por m al número de aristas, por lo que $|E| = m$ y k es el tamaño máximo de un camino en la red. Cada arista $(v, u) \in E$ representa un canal de transmisión capaz de transmitir paquetes de información de v hacia u y paquetes de control entre $u \leftrightarrow v$ (en ambas direcciones). Suponemos además que cada nodo cuenta con un número de identificación único.

Por otra parte, cada nodo conoce su vecindario, pero no cuenta con ningún dato adicional sobre la topología de la red. El modelo de comunicación es síncrono, por lo que existen rondas de comunicación y en cada ronda solamente un mensaje puede ser enviado a través de una arista (v, u). Asumimos también que un mensaje es recibido de manera instantánea y que todos los canales de comunicación son confiables, es decir, si un paquete es enviado a través de una arista *e=(v,u)*, entonces el nodo u recibirá dicho paquete en la misma ronda que fue enviado.

Para la comparación de diversos algoritmos, se emplearán las siguientes medidas de eficiencia:

- *Complejidad temporal:* Es el número de rondas de comunicación requeridas para la ejecución del algoritmo
- *Complejidad de mensajes:* Es el número total de mensajes enviados durante la ejecución del algoritmo
- *Complejidad de bits:* Es la longitud total en bits de los mensajes enviados a través de la red durante la ejecución del algoritmo

El problema a resolver es el siguiente:

Problema 3. *Dada una red* $G = (V, E)$*, un nodo fuente* s*, un nodo destino* t*, encontrar* h (s, t)*-caminos de aristas disjuntas entre* s *y* t

3.2.2. Algoritmos Analizados

Se analizarán 3 algoritmos distribuidos que resuelven el problema de encontrar caminos disjuntos, definido con antelación (ver problema 1), descritos de manera general a continuación:

Algoritmo 1: Cada nodo de la red envía información sobre su vecindad al nodo fuente, una vez que el nodo fuente ha recibido información de cada uno de sus vecinos, ejecuta de manera local un algoritmo centralizado, el cual resuelve el problema de encontrar h caminos disjuntos entre s y t, finalmente propaga la solución encontrada por el algoritmo centralizado por toda la red, de manera tal que cada nodo de la red conozca la solución al problema.

Algoritmo 2: Se basa en el algoritmo centralizado para encontrar pares de caminos disjuntos [3], el cual consta de las siguientes etapas:

- Encontrar un árbol de caminos más cortos cuyo nodo raíz es el nodo fuente del problema, s
- Transformar cada arista de manera que la longitud de dos caminos con el mismo vértice de inicio y final haya sido modificado por la misma cantidad, lo que permitirá que el ordenamiento de dichos caminos de acuerdo a su longitud no se vea afectado por la transformación
- Obtener los caminos disjuntos en la gráfica con las longitudes modificadas
- Repetir el procedimiento por cada camino adicional desde el segundo paso

Algoritmo 3: Este algoritmo emplea Codificación en Redes para resolver el problema. Inicialmente se ejecuta un algoritmo distribuido para obtener un código de red, una vez que el nodo destino ha recibido h paquetes diferentes, envía mensajes para el descubrimiento de los caminos disjuntos hacia el nodo fuente.

En la figura 3.13 se presenta un cuadro comparativo entre los 3 algoritmos citados, en él, se muestra que el algoritmo más eficiente con relación al tiempo de ejecución medido en rondas resulta ser el algoritmo genérico, con el inconveniente de que la medición no toma en cuenta el tiempo de procesamiento, aunado a la desventaja de que dicha labor ocasiona un incremento de tráfico en la red hacia el nodo fuente.

Algoritmo	Complejidad de Tiempo	Complejidad de Mensajes	Complejidad de Bits
Genérico	$O(k) + \epsilon$	$O(\|E\|)$	$O(\|V\|^2)$
Suurballe-Tarjan	$O(h \cdot k)$	$O(\|E\| \cdot \|V\|)$	$O(k \cdot log\ n)$
Codificación en Redes	$O(k^2)$	$O(\|E\| \cdot \mathfrak{C})$	$O(h^2 + \|V\|)$

Figura 3.13: Comparación entre algoritmos

Comparando los 2 algoritmos restantes, tenemos que el algoritmo que emplea Codificación en Redes es más eficiente cuando $h > k$, es decir, cuando se requieren más caminos en una red cuyos nodos no se encuentran muy alejados entre sí.

Vinculado a la complejidad de mensajes, sabemos que siempre se cumple que $\mathfrak{C} \leq |V|$, por lo que el algoritmo de codificación en redes es más eficiente que el propuesto por Suurballe-Tarjan y finalmente, para la complejidad de bits, este último algoritmo resulta ser más eficiente, sin embargo, cabe resaltar que el desempeño del algoritmo de codificación en redes, resulta ser aceptable, puesto que los mensajes más comunes enviados durante la ejecución del algoritmo son de $O(h + |V|)$ bits, que en comparación con el algoritmo más eficiente resultan ser similares.

3.2.3. Algoritmo Genérico

La idea del algoritmo es la de implementar de la manera más sencilla posible un algoritmo centralizado como distribuido. Para ello, un nodo central es designado como responsable de todos los cálculos (generalmente el nodo fuente), de manera que solamente se requiera que cada nodo de la red de cómputo envíe información sobre sus aristas al nodo central designado, empleando un algoritmo de inundación [6] y cuando el nodo central haya recibido suficiente información sobre la topología de la red, comienza su trabajo de cómputo para obtener una solución al problema, la cual finalmente propagará a través de toda la red.

La complejidad temporal del algoritmo es $O(k)$ rondas, ya que el tiempo que toma a cada nodo envíar su información al nodo central es a lo más, el máximo número de rondas que separan a 2 nodos de la red, es decir $O(k)$. El tiempo que le toma al nodo central la obtención de una solución depende del algoritmo centralizado que ejecute, por ejemplo, el algoritmo ideado por Suurballe y Tarjan [17] toma $O(|E| \cdot k \cdot log|V|)$ instrucciones, pero debido a que el desempeño de los algoritmos se mide en términos de rondas, este valor es omitido durante los cálculos de desempeño, de cualquier manera, es importante considerarlo para comparaciones más detalladas.

La complejidad de mensajes es $O(|E|)$ ya que el nodo central envía inicialmente una petición de información a cada uno de sus vecinos y una vez que cualquier nodo de la red reciba dicha notificación, la propaga al resto de su vecindario y espera a que cada uno de sus vecinos le envíe la información requerida, empleando para ello la misma arista por la cual recibió la petición, este

[6] Un algoritmo de inundación es aquel que permite enviar mensajes a partir de un nodo fuente al resto de los nodos de la red

proceso toma a lo más $O(|E|)$ mensajes ya que cada arista es empleada a lo más 2 ocasiones, una para enviar la petición y otra para recibir la información.

La complejidad en bits del algoritmo es $O(|V|^2)$ ya que cada nodo deberá incluir en sus mensajes los nodos a los cuales se encuentra conectado y como cada nodo puede estár conectado a lo más a $|V|-1$ nodos, esto implica que cada paquete que contenga la información del vecindario de un nodo tiene tamaño $O(|V|)$ y como cada nodo debe esperar hasta que cada uno de sus vecinos le envíe su información, la máxima cantidad de mensajes que deberá incluir en su información es $O(|V|)$.

3.2.4. Algoritmo Suurballe-Tarjan Distribuido [3]

El algoritmo hace uso de una función de costo que asocia un número natural a cada una de las aristas de la gráfica dirigida acíclica, asumimos sin embargo que cada arista tiene asociado un costo unitario.

El algoritmo hace uso de 2 pasos preliminares:

1. Encontrar un árbol de distancias mínimas T con raíz en s. Dicho árbol contiene para cada vértice v, un camino más corto de s a v. Calcular $d(s,v)$, que es la distancia de s a v para cada vértice v
2. Transformar el costo de cada arista (v,w) definiendo $c'(v,w) = c(v,w) - d(s,w) + d(s,v)$

Esta transformación permite que 2 caminos con el mismo nodo inicial y final, transformen sus costos por la misma cantidad, con lo que el ordenamiento de dichos caminos por costo se vea inalterado, lo que significa, que podemos resolver el problema de los caminos más cortos para la gráfica transformada en costos y obtener una solución para la gráfca con los costos originales. El algoritmo se basa en el siguiente teorema, empleado en el algoritmo centralizado:

Teorema 4. *[3] Para cualquier vértice v, la longitud total de un par de caminos de longitud mínima con aristas disjuntas de $s \to v$ es $d_v(s,v)$, donde d_v denota la función de distancia en G_v, que es la gráfica formada a partir de G invirtiendo la dirección de todas las aristas que forman el camino en T desde s hasta v. La unión del par de caminos obtenido de s a v en T y el camino más corto obtenido de s a v en G_v descartando cada arista en un camino cuyo recorrido aparezca en el otro camino y agrupando las aristas restantes en caminos de s a t.*

A continuación se da una descripción de un algoritmo para la obtención de un par de caminos disjuntos de $s \to t$ [17]:

1. Realizar la siguiente transformación a la gráfica $G = (V, E)$: $c'(v, w) = c(v, w) - d(s, w) + d(s, v)$

2. Remover todos los arcos a través del camino más corto que se dirigen al nodo fuente. Invertir la dirección de las aristas restantes del camino más corto de manera que cada arista se dirija hacia el nodo fuente

3. Ejecutar nuevamente el algoritmo para la obtención de caminos más cortos de s a t en la gráfica modificada G_t

4. Transformar la gráfica original. Eliminar las aristas que se entrelazan en los caminos encontrados de $s \rightarrow t$, el par de caminos disjuntos de longitud mínima resulta

Algoritmo para la obtención de h caminos disjuntos de $s \rightarrow t$ [17]:

1. Encontrar un par de caminos de longitud mínima de aristas disjuntas entre s y t

2. Reemplazar las aristas de los caminos disjuntos por arcos dirijidos hacia el nodo fuente. Cambiar las longitudes de los arcos por longitudes negativas

3. Encontrar el camino más corto de $s \rightarrow t$ en la gráfica modificada G_t utilizando un algoritmo para distancias mínimas

4. Transformar la gráfica original y eliminar las aristas que se entrelacen

La complejidad temporal del algoritmo distribuido es:
Para la obtención del árbol de distancias mínimas $O(k)$. Para la transformación del costo en cada arista en el camino más corto de s a t $O(1)$ y como el proceso debe llevarse a cabo en h ocasiones, la complejidad del algoritmo entero es $O(h \cdot k)$.

La complejidad de mensajes del algoritmo distribuido es:
Para obtener el árbol de distancias mínimas $O(|V| \cdot |E|)$ [18], para transformar el costo en cada una de las aristas del camino más corto $O(1)$ mensajes y ya que este proceso debe realizarse para cada uno de los caminos, debe realizarse en $|V|$ ocasiones, por lo que el número de mensajes enviados al ejecutar el algoritmo es $O(|V| \cdot |E|)$.

La complejidad en bits de cada uno de los mensajes es para la obtención del árbol de caminos más cortos $O(log\ n)$ que corresponde a la máxima cantidad de bits necesarios para representar a un nodo de la red de cómputo, de manera que sea posible asignar un nodo padre a cada nodo, para

transformar el costo de cada arista es necesario que el nodo fuente reciba cada camino disjunto, por lo que el tamaño máximo de un mensaje en bits es $O(k \cdot log\ n)$.

3.2.5. Algoritmo empleando Codificación en Redes

Para que sea más sencillo el análisis del algoritmo de codificación en redes, es necesario realizar una transformación a la gráfica original, la cual permitirá la manipulación de ésta como una gŕafica por capas, denominada $L(G)$ en lo sucesivo (ver figura 3.14), en la cual L_i representa a la i-ésima capa de $L(G)$, se construye como a continuación se describe:

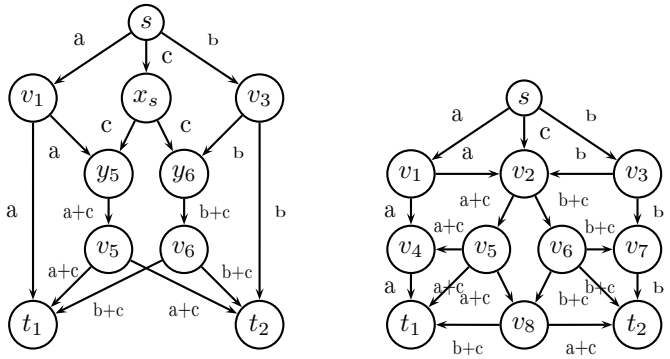

Figura 3.14: Ejemplo de una gráfica en capas L(G) (abajo) para la gráfica G (arriba)

1. El número de capas en la gráfica nueva $L(G)$ es $k+1$, que corresponde a la distancia máxima entre cualquier par de vértices en la gráfica original, además, cada capa cuenta con un número de identificación $i : 0 \leq i \leq k$

2. Cada capa en $L(G)$ contará con $|V|$ nodos

3. Una arista (u^i, v^{i+1}) existe en $L(G)$ si y solo si $u^i \in L_i$, $v^{i+1} \in L_{i+1}$, la arista (u, v) existe en la gráfica original G

4. Una arista (u^i, u^j) existe en $L(G)$ para $0 \leq i = j - 1 \leq k$ y para cada $u \in V$

Donde v^i denota al nodo v dentro de la capa i de $L(G)$.
La gráfica por capas $L(G)$ construida cuenta con las siguientes propiedades:

Lema 3. *$L(G)$ es una gráfica acíclica*

Demostración. Supongamos que existe un ciclo en $L(G)$, existen dos posibles escenarios:

- El ciclo se encuentra en la misma capa, lo cual no es posible ya que $L(G)$ no cuenta con aristas entre nodos de la misma capa
- El ciclo se encuentra entre capas adyacentes, lo cual no es posible ya que si existe una arista de un nodo en la capa L_i a un nodo en la capa L_{i+1}, entonces no es posible que exista una arista del nodo empleado en L_{i+1} al mismo nodo de la capa L_i

□

Lema 4. *Como existe siempre un camino entre s y v $\forall u \in V$, si existe un camino entre $u \in V$ y $v \in V$ en G, entonces existe un camino en $L(G)$ entre los nodos u^i y v^k*

Demostración. La prueba se efectuará por inducción en el número de nodos π de un camino, excluyendo en esta cuenta a los nodos fuente y destino:
Caso Base: Para $\pi = 0$ existe un camino en $L(G)$ del nodo u^i al nodo u^{i+1} y también si existe un camino del nodo u al nodo v en G, existe una arista (u^i, v^{i+1}) en $L(G)$ por la manera en que se construye la gráfica

Hipótesis Inductiva: Suponemos que para $\pi = r$, si existe un camino en G entre los nodos u y v con π nodos intermedios, entonces existe un camino en $L(G)$ de u^i a v^j con el mismo número de nodos intermedios

Paso Inductivo: Ahora probaremos para $\pi = r + 1$ que si existe un camino en G para cualquier par de nodos u y v con π nodos intermedios, también existe un camino en $L(G)$ de u^i a v^j con el mismo número de nodos intermedios. Para ello, sea $p = \{u - v_1 - v_2 - \ldots - w - v\}$ un camino de u a v en G, por H.I. sabemos que existe un camino en $L(G)$ de u^i a w^{j-1} y empleando el caso base sabemos que además existe siempre un camino de w^{j-1} a v^j ya que existe un camino en G de $w \rightarrow v$ □

Lema 5. *Cada camino en $L(G)$ entre cualquier nodo u^i y el nodo destino t^k tiene la misma longiud*

Demostración. La prueba se efectuará por inducción sobre la longitud de los caminos λ:
Caso Base: Si $\lambda = 1$, existe solamente un camino entre u^i y t^k ya que no se emplean aristas paralelas

Hipótesis Inductiva: Suponemos que para $\lambda = r$ cada camino entre u^i y t^k tiene la misma longitud

Paso Inductivo: Ahora probaremos que para $\lambda = r + 1$ la hipótesis es cierta:
Por H.I. cada camino de u^i a v^{k-1} tiene la misma longitud: $\lambda = r$ y por el caso base, cada camino de v^{k-1} a t^k tiene longitud 1, por lo que el total de las longitudes de todos los caminos permanece igual □

Corolario 5. *Si la gráfica G contiene h (s,t)-caminos disjuntos, el número de aristas salientes de cada capa $L^i \in L(G)$ es al menos h*

Descripción del Algoritmo

1. Ejecutar un algoritmo para encontrar un código de red factible (ver algoritmo LIF en la sección 3.1.6). En cada ronda del algoritmo, todos los nodos almacenarán información relacionada a los paquetes recibidos por cada una de sus aristas de entrada, esta información será empleada posteriormente para generar la gráfica de capas $L(G)$

2. Una vez que el nodo destino ha recibido h paquetes linealmente independientes, él envía un mensaje para descubrir los caminos independientes, cada mensaje tendrá asociado un identificador para el camino y el conjunto de vectores independientes empleado en cada arista de entrada al nodo destino

3. Cuando un nodo recibe un mensaje para descubrir un camino, espera hasta que cada nodo de codificación más cercano que él al nodo destino, haya efectuado su elección de arista que cumpla con la invariante de que el conjunto de vectores continúe siendo linealmente independiente. Una vez que todos los nodos más cercanos al nodo destino hayan efectuado su elección de arista para cada camino, el nodo en cuestión lleva a cabo su elección y envía un mensaje de notificación de la elección al resto de la red, de manera que el resto de los nodos actualicen la información de los nuevos vectores linealmente independientes

4. El algoritmo termina una vez que el nodo fuente recibe la descripción de los h caminos de aristas disjuntas

Mensajes Empleados

1. NETWORK_CODE(encoding vector ev, encoding nodes' table ent[nodeID, roundID]). Este mensaje será empleado durante la primera etapa del algoritmo para encontrar un código de red factible

2. PATH_DISCOVERY(pathID, current cut table cct[pathID, cv], delayed rounds table drt[roundID, nodeID]). Este mensaje será empleado durante le última etapa del alogritmo para etiquetar los diversos caminos disjuntos encontrados. La tabla de rondas de retraso será empleada para que cada nodo pueda verificar si es necesario que espere por el envío de una actualización en el corte de parte de un nodo de codificación

3. CUT_UPDATE(cut updates' table cut[pathID, encoding vector ev], nodeID, set of notified nodes snn[nodeID]). Este mensaje es enviado por un nodo de codificación para actualizar la información del corte actual, de manera que el algoritmo contenga suficientes vectores linealmente independientes para poder formar una bases sobre $\mathbb{F}^h$

4. ROUND_INC. Este mensaje es interno y se emplea para notificar a cada nodo que una nueva ronda ha comenzado

Estructuras de Datos

1. Layered Graph lg[round, sourceID, encoding vector ev]: Esta tabla contiene la información pertinente para mantener la estructura de la gráfica de capas en cada uno de los nodos, almacenando por cada arista entrante, su nodo fuente y el vector de codificación empleada para dicha arista

2. Routing Table rt[sourceID, pathID]: Esta tabla mantiene una relación de las aristas que ya cuentan con un camino asociado

3. Incoming Messages im: Contador del número de mensajes recibidos por el nodo

4. Current Cut cc[pathID, encoding vector ev]: Empleado para compartir la información del corte actual de la red de cómputo entre los diversos procesos del nodo

5. Waiting Vector wv[roundID, sourceID]: Empleada para compartir la información vinculada a las rondas en las cuales un nodo debe de esperar por un mensaje de actualización del corte (CUT_UPDATE)

6. ROUND: El número de ronda actual (actualizado de manera interna en cada nodo)

7. INCOMING_LINKS: Número de aristas de entrada del nodo

```
1 Procedimento:H-Disjoint Paths(number of disjoint paths h)
2 vector = getEncodingVectors(h) *;
3 for i=1 to h do
4 |   send(NETWORK_CODE(vector_i, Φ) to neighbor_i;
5 end
```

Figura 3.15: Procedimiento principal para encontrar h caminos disjuntos

* La función getEncodingVectors genera al menos h paquetes con vectores linealmente independientes, si $grado^{-}(v) > h$, entonces genera combinaciones lineales de los paquetes originales

```
1 begin upon ROUND_INC is received
  |  /* Para mantener la gráfica de capas, se necesita una arista de u_i a u_{i+1} */
2 |  ROUND = ROUND + 1;
3 |  if im > 0 then
4 |  |  lg.add(ROUND, v_ROUND, lg.encodingVectors());
5 |  end
6 end
```

Figura 3.16: Algoritmo para Incremento de Rondas

Análisis del Algoritmo

Para demostrar la correctez del algoritmo, emplearemos lo siguiente:

Lema 6. *[20] Para cualquier $e \in E$ y $t \in T(e)$, B_t es el conjunto que contiene $b(f^t_{\leftarrow}(e))$ y forma una base para $\mathbb{F}^h$. Entonces, con probabilidad $1/\mathbb{F}$, una elección aleatoria de los coeficientes m_e con base en los vectores de codificación entrantes falla en conservar la propiedad de que el nuevo conjunto de vectores forma también una base para $\mathbb{F}^h$.*

$T(e)$ Es el conjunto de nodos desino que emplean la arista e
$f^t_{\leftarrow}(e)$ es el predecesor de la arista e en un camino del nodo fuente al nodo destino t
$b(e)$ es el vector de codificación de la arista e

Primeramente, mostraremos como se preserva la gráfica de capas dentro del algoritmo distribuido.

La gráfica de capas cuenta con 2 tipos diferentes de aristas, las aristas entre el mismo nodo en capas contiguas y las aristas entre diferentes nodos. Cada nodo v, al incrementarse el número de ronda, almacena una nueva arista de v_{round} a $v_{round+1}$ (ver algoritmo de la figura 3.16) dentro de su estructura para la gráfica de capas, con lo que mantiene el mismo número de aristas entre un

mismo nodo que en una gráfica por capas.

Para las aristas entre diferentes nodos, cada vez que un mensaje NETWORK_CODE es recibido, cada nodo almacena una nueva arista dentro de su estructura para la gráfica por capas junto con el número de ronda en el cual el mensaje fue recibido, para poder conocer la capa a la cual pertenece dicha arista (ver figura 3.17, line 2), como cualquier otra arista no aporta información relevante para el algoritmo, son omitidas dentro de la estructura.

Solamente resta por mostrar como el algoritmo satisface la condición del lema 6.

Demostración. La demostración se realiza mediante inducción sobre el número de rondas r.
El algoritmo consta de 2 etapas principales, en la primera el nodo fuente envía paquetes en dirección al nodo destino para encontrar un código de red factible y en la segunda el nodo destino encuentra los caminos disjuntos valiéndose para ello de los vectores de codificación asociados a cada una de las aristas.

Primera Etapa

Caso Base: Cuando $r = 1$, el nodo fuente genera h paquetes $P = \{p_1, p_2, ..., p_k\}$ y envía un vector de codificación linealmente independiente por cada una de sus aristas de salida, de manera que si la gráfica contiene h (s, t)-caminos disjuntos, el nodo fuente enviará al menos h vectores linealmente independientes (ver figura 3.15).

Hipótesis Inductiva: Suponemos que el lema se cumple para $r = m$

Paso Inductivo: Ahora probaremos que el lema se cumple para $r = m + 1$.
Esta prueba se basa en la implementación de los procesos requeridos para la obtención de los vectores de codificación linealmente independientes para cada una de las aristas y se encuentra en [20].

Segunda Etapa

Caso Base: Cuando $r = 1$, durante la primera ronda después de que el nodo destino ha obtenido h vectores de codificación linealmente independientes, dicho nodo envía un mensaje para descubrimiento de caminos PATH_DISCOVERY por cada una de sus aristas de salida, dicho mensaje contiene un identificador para el camino que tendrá asociado y el conjunto de vectores linealmente independientes que forman una base para $\mathbb{F}^h$, condición que deberá de preservarse hasta que el nodo

fuente sea alcanzado.

Hipótesis Inductiva: Suponemos que el lema se cumple para $r = m$

Paso Inductivo: Probaremos que el lema se cumple para $r = m + 1$.
Para ello suponemos que existe un camino descubierto $P = \{e_1, ..., e_m\}$, por H.I. sabemos que el lema se satisface y que una ronda después existen los siguientes escenarios:
Si el nodo en cuestión no es un nodo de codificación, cuenta solamente con una arista de entrada y dicha arista será etiquetada con el identificador del camino recibido en el mensaje PATH_DISCOVERY, el cual reenviará por la arista etiquetada, incrementando con ello la longitud del camino en 1. Como el vector de codificación asociado a la nueva arista es igual al de la arista por donde se recibió el mensaje, el conjunto de vectores linealmente independientes sigue formando una base para $\mathbb{F}^h$.

Si el nodo en cuestión es un nodo de codificación, deberá de esperar hasta que cada nodo de codificación más cercano al nodo destino haya efectuado una elección de arista y haya enviado la notificación pertinente al resto de la red, de manera que al momento de que el nodo elija la nueva arista a ser agregada a su camino, la invariante de formar una base para $\mathbb{F}^h$ se preserve (ver 3.20, líneas 1,2). Una vez que el nodo ha recibido todas las actualizaciones esperadas, elige un vector linealmente independiente y reemplaza el vector anterior asociado al camino para finalmente enviar una notificación de modificación con un mensaje CUT_UPDATE a cada uno de sus vecinos, con lo que la base se preserva (ver 3.20). □

Análisis de desempeño del Algoritmo

La complejidad de tiempo del algoritmo es primeramente, para encontrar un código de red factible $O(k)$ y para el descubrimiento de los caminos disjuntos $O(k^2)$, ya que a lo más en cada ronda, un nodo deberá esperar por una notificación de actualización del corte, en general podemos decir que la complejidad es $O(k^2)$.

La complejidad en el número de mensajes empleados durante la ejecución del algoritmo es, para encontrar un código de red factible, $O(|E|)$, ya que a lo más cada arista transporta un mensaje hasta que h mensajes son recibidos por el nodo destino. Para la etapa de descubrimiento de caminos disjuntos, a lo más cada nodo de codificación envía $O(|E|)$ mensajes de forma que el resto de los nodos actualicen su información sobre el corte de la gráfica, por lo tanto, la complejidad total del algoritmo es $O(|E| \cdot \mathfrak{C})$, donde $\mathfrak{C}$ es el número de nodos de codificación en G.

La complejidad de bits depende del contenido de cada mensaje:
El mensaje de NETWORK_CODE se conforma por un vector de codificación $O(h)$, una tabla con los nodos de codificación descubiertos $O(|V|)$, en el caso de que cada nodo sea de codificación, por lo que su complejidad es $O(h + |V|)$.
El mensaje de CUT_UPDATE se conforma por una tabla con las actualizaciones efectuadas por un nodo, el cual es $O(h^2)$ ya que a lo más h actualizaciones pueden ser efectuadas en un nodo, una por cada camino y el tamaño de cada actualización es h, el número de vectores linealmente independientes, además cuenta con una tabla con los nodos que ya han recibido la notificación, con lo que la complejidad total es $O(h^2 + |V|)$.
El mensaje PATH_DISCOVERY consta de una tabla que contiene los vectores de codificación que forman la base para $\mathbb{F}^h$, la cual es $O(h^2)$, además cuenta con una tabla en la que se determinan las rondas en las que existen nodos de codificación, $O(\mathfrak{C})$, por lo que la complejidad total es $O(h^2+\mathfrak{C})$.

El algoritmo tiene una complejidad de bits final de $O(h^2 + |V|)$.

```
1  begin upon NETWORK_CODE⟨ev, ent⟩ is received from v_i
       /* Almacena el vector de codificación usado en la arista de la que proviene
          el mensaje                                                              */
2      lg.add(ROUND, v_i, ev);
3      im = im + 1;
       /* Si el nodo ha recibido un mensaje de todas sus aristas entrantes, crea
          sus mensajes de codificación                                            */
4      if im = INCOMING_LINKS then
5          if v is Destination Node then
               /* Actualiza los vectores con el corte actual y comienza el
                  descubrimiento de caminos                                       */
6              pathID = 0;
7              foreach data in lg do
8                  pathID = pathID + 1;
9                  cc.add(pathID, data.ev);
10             end
11             pathID = 0;
12             foreach data in lg do
13                 pathID = pathID + 1;
14                 send(PATH_DISCOVERY(pathID, cc, wv)) to data.source;
15                 rt.add(data.source, pathID);
16             end
17         end
18         if INCOMING_LINKS > 1 then
               /* Si se han recibido varios mensajes, el nodo puede ser de
                  codificación                                                    */
19             ent.add(ROUND, v);
20         end
           /* Obtiene los nuevos vectores de codificación para cada una de sus
              aristas de salida                                                   */
21         encoding vectors = getEncodingVectors(lg) *;
22         foreach data in encoding vectors do
23             send(NETWORK_CODE(data.encoding_vector, ent)) for data.target_node;
24         end
25     end
26 end
```


* Una descripción detallada de la función getEncodingVectors puede encontrarse en [19]. El propósito de la función es el de encontrar un nuevo conjunto de vectores linealmente independientes, uno por cada arista de salida del nodo

```
1  begin upon CUT_UPDATE⟨cut, nodeID, cnn⟩ is received from v_i
       /* Actualiza el corte actual y remueve al nodo de la lista de espera     */
2      foreach cut_ change in cut do
3          cc.replace(cut_change.pathID, cut_change.ev);
4      end
5      wv.remove(nodeID);
       /* Reenvía la actualización del corte a cada uno de sus vecinos que aún no
          lo han recibido                                                        */
6      cnn' = cnn ∪ neigbours;
7      foreach neighbor ∉ {cnn} do
8          send(CUT_UPDATE(cut, nodeID, cnn');
9      end
10     cnn = cnn';
11 end
```

Figura 3.18: Algoritmo para la notificación de modificacioes en el corte de la red

```
1  Procedimento:getWaitingList()
2  maxRound = lg.getMaximumRound();
3  waitingList = Φ;
4  foreach data in wv do
5  |  if data.roundID ≥ maxRound then
6  |  |  if data.sourceID ≠ v then
7  |  |  |  waitingList.add(data.sourceID);
8  |  |  end
9  |  end
10 end
   /* Para evitar ciclos infinitos, si el nodo cuenta con el menor identificador,
      no es necesario que espere                                              */
11 if waitingList.maxRound = maxRound AND
   waitingList.getNodesAtRound(maxRound).minID = v.id then
12 |  waitingList = Φ;
13 end
14 return waitingList;
```

Figura 3.19: Procedimiento para la obtención de la lista de nodos de codificación más cercanos al nodo destino

```
1  begin upon PATH_DISCOVERY⟨pathID, cct, drt⟩ is received from v_i
2      wv = drt;
       /* Verifica si no es necesario esperar por algún nodo de codificación    */
3      while wv.getWaitingList() ≠ Φ AND v is an encoding node do
           /* Si la lista de espera no está vacía, esperar una actualización de
              corte e intentar nuevamente                                        */
4          wait(CUT_UPDATE);
5      end
6      targets = getLinearlyIndependentTargets(cct, ld) *;
       /* Si existen varias posibilidades, actualizar el corte actual y notificar
          al resto de la red                                                     */
7      if targets > 1 then
8          cct.replace(pathID, lg.getEV(targets[0]));
9          send(CUT_UPDATE(cct, v, neighbors ∪ {v})) to every neighbor;
10         wv.remove(v);
11     end
       /* Continua con el descubrimiento de caminos                             */
12     rt.add(v_i, pathID);
13     send(PATH_DISCOVERY(pathID, cct, wv) through targets[0];
14 end
```

* Una descripción detallada de la función getEncodingVectors puede encontrarse en [19]. El propósito de la función es el de encontrar un nuevo conjunto de vectores linealmente independientes, uno por cada arista de salida del nodo

Figura 3.20: Algoritmo para encontrar h caminos disjuntos

Capítulo 4

Conclusiones y Trabajo Futuro

Este trabajo es un esfuerzo por dar a conocer la codificación en redes como una nueva posibilidad de ruteo de paquetes en una red de cómputo, que cuenta con la principal ventaja de ahorrar ancho de banda mediante el empleo de funciones de codificación lineales en algunos nodos de la red. Al ser este un tema relativamente nuevo dentro del área de redes de computadoras, el trabajo por realizar es extenso y la idea de presentar un algoritmo que permite encontrar codificaciones en redes, es primordialmente con el afán de impulsar la búsqueda de posteriores implementaciones, especialmente en el campo del cómputo distribuido, área que se encuentra poco explorada y dentro de la cual la codificación en redes podría aportar una cantidad de avances significativos.

Dentro de esta área de investigación, la de codificación en redes, aún queda trabajo pendiente por llevar a cabo, como por ejemplo en la obtención de algoritmos eficientes para el caso de gráficas no dirigidas en donde el empleo de codificación en redes para el caso de algoritmos distribuidos se torna más complejo debido a la presencia de ciclos, en especial la obtención de códigos de red.

Igualmente será de interés el estudio de problemas como la detección de ciclos en una red, la identificación de vértices de separación de una gráfica dada, la obtención de componentes separables dentro de una gráfica, entre otros problemas dentro de los cuales la codificación de redes es factible que arroje resultados interesantes si es empleada en la obtención de algoritmos distribuidos.

El principal propósito de la investigación futura radicará en desarrollar métodos y herramientas para aplicaciones en redes empleando codificación en redes, para ello, temas como los citados a continuación, se tornan ejemplos dentro de las cuales la codificación en redes podrá sin duda, efectuar aportaciones trascendentes:

- Análisis de factibilidad de algoritmos basados en codificación en redes para resolver problemas clásicos de redes de comunicación. En especial dentro del área de sistemas distribuidos existen una gran variedad de problemas dentro de los cuales la codificación en redes puede realizar aportes significativos.
- Análisis y diseño de métodos y protocolos para redes con capacidad de recuperación en caso de fallas en aristas. Una de las principales ventajas de la codificación en redes es su capacidad para recuperación de fallas en aristas de manera instantánea, es decir, sin la necesidad de retransmisión de paquetes.
- Desarrollo de algoritmos para Tomografía de Redes
- Desarrollo de protocolos de transmisión para redes dinámicas. Particularmente en redes inalámbricas, existen constantes cambios en la topología de la red debido al constante movimiento de los nodos de transmisión, la intensión en este rubro es el de generar protocolos basados en codificación de redes que permitan hacer frente a este tipo de contrariedades.

4.1. Recuperación de información en caso de fallas en canales de comunicación

Mediante el empleo de algoritmos distribuidos para la obtención de una codificación factible para una red de cómputo dada, la característica de recuperación de información en caso de fallas se encuentra inherente al algoritmo.
Un algoritmo distribuido trabaja permitiendo a cada nodo de la red la elección de los vectores de codificación de manera aleatoria (tomando en cuenta los vectores de codificación recibidos con antelación) y dichos vectores son enviados junto con los paquetes codificados, lo que permite a los nodos destino la decodificación de la información aún en caso de fallas en los canales de comunicación de la red de cómputo.
Diversos estudios al respecto resaltan el hecho de que el envío de los vectores de codificación junto con los paquetes de datos no son necesariamente un lastre, sino que pueden ser empleados para deducir información adicional sobre la ubicación precisa de fallas dentro de la red de cómputo. Esto se debe principalmente a que diversas fallas en las aristas de la red producen diversos arrivos de vectores de codificación a los nodos destino, por lo que estudiando las características de los patrones de error observados dentro de los vectores de codificación, es posible determinar los diversos tipos de fallas en las aristas presentes en una sesión de transmisión de información dentro de una red de cómputo (ver [21]).

Sin embargo es importante resaltar el hecho de que diversos patrones de error pueden ser indistinguibles, por lo que en general se analizan conjuntos de patrones de error (ver figura 4.1).

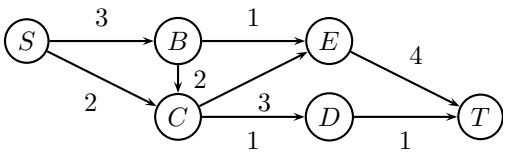

Figura 4.1: Ejemplo de Patrones de Error. Una falla en cualquiera de las aristas de los extremos izquierdo y derecho son indistinguibles para el nodo de la derecha

Existen trabajos previos en los que el análisis de los códigos de red obtenidos por los nodos destino permiten la deducción de posibles localizaciones de las fallas en los canales de comunicación así como de la ubicación de la pérdida de mensajes dentro de la red, además se han encontrado cotas para el tamaño del campo finito requerido para la generación de códigos de red más robustos, de manera tal que sea posible diferenciar con mayor grado de detalle entre eventos de fallas que se presenten en una sesión de transmisión.

Es pertinente resaltar el hecho de que aún queda trabajo por hacer en el área de redes inalámbricas, en donde el empleo de codificación en redes ha sido escuetamente analizado, además la detección de fallas en canales de comunicación es un rubro importante en la administración de redes y en algunas ocasiones es la principal causa de demoras, por lo que la codificación en redes resulta ser un tópico de alta relevancia en este tipo de labores.

4.2. Tomografía de Redes

El monitoreo o la inferencia de las características internas de una red mediante mediciones de los datos de entrada o salida de la red se conoce como tomografía de redes [21].
Existen diversas clasificaciones para las tomografías:

- Tipo de Medición Empleada
  - Mediciones Activas: Involucra el envío de mensajes para sondeo de las propiedades
  - Mediciones Pasivas: Cuya intención es inferir propiedades de la red analizando los mensajes existentes
- Alcance de los Mensajes Enviados

    - Unicast
    - Multicast
    - Broadcast
- Tipo de Red Analizada
    - Redes Alambradas
    - Redes Inalámbricas
    - Redes de Sensores

En general podemos establecer que la principal ventaja del empleo de tomografías de redes es el hecho de que ya no se depende mas de la cooperación entre diversas redes, es decir, mediante el empleo de una robusta red de monitoreo, muestreo de entradas y salidas e inferencias estadísticas, es posible la obtención de conocimiento relacionado a características internas de una red de comunicación.

En los primeros trabajos del área, se hacía especial énfasis en llevar a cabo muestreos de la red con un nodo fuente y múltiples destinos, lo que provocaba un incremento significativo del tráfico de mensajes en el vecindario del nodo fuente, ocasionando un bajo desempeño de este tipo de enfoques y una falta de escalabilidad de las soluciones a problemas que involucrasen redes de tamaño considerablemente mayor.

Es por ello que trabajos posteriores se han enfocado en la realización de muestreos en múltiples fuentes y múltiples destinos, pues se ha demostrado que este tipo de problemas puede ser reducido a tomografías sobre dos nodos fuente y dos nodos destino (ver [22], para detalles), con lo que el tráfico de mensajes cuenta con una mejor distribución sobre la red, pero trae como problemas la coordinación del muestreo y la combinación de los datos obtenidos para poder obtener información relevante sobre las características internas de la red.

4.3. Redes Dinámicas

En años recientes, ha emergido un creciente interés en el diseño y desarrollo de redes inalámbricas, redes que se han transformado poco a poco en elementos no solo importantes, sino indispensables para contar con una cobertura ubicua para la transferencia de información. Uno de los puntos más importantes en el diseño de redes inalámbricas es el incremento del desempeño de la red y la minimización del consumo de poder de los transmisores.

Recientemente se han efectuado investigaciones que atañen a las difusiones generales, pues se ha observado que por su naturaleza, las redes inalámbricas pueden valerse de los mensajes enviados para minimizar el número de transmisiones. En general, en los ambientes de redes inalámbricas, cada paquete es enviado a todo el vecindario del nodo transmisor de manera involuntaria, por lo que es posible emplear esta información, tanto de los paquetes recibidos como del conocimiento del entorno para poder llevar a cabo codificaciones oportunistas con el afán de reducir el número de transmisiones necesarias.

Existen trabajos que han buscado el mínimo número de transmisiones necesarias para satisfacer los requerimientos de información de cada uno de los clientes (ver [23] para más detalles). El problema tratado en el documento se vincula al siguiente ejemplo:

Basándose en la red de la figura 4.2, el nodo central, tiene como objetivo entregar algún subconjunto de los paquetes que conoce (p_1, p_2, p_3, p_4) a cada uno de sus clientes. Por su parte, cada cliente, debido a transmisiones previas, conoce un subconjunto de dichos paquetes (has) y requiere de otro subconjunto (wants). El problema consiste en encontrar el mínimo número de transmisiones requeridas para satisfacer a todos los clientes.

Para resolver este problema, se observa que sin el empleo de codificación en la red, se requiere de cuatro transmisiones, pues cada cliente requiere un paquete diferente, sin embargo, empleando codificación, el número de transmisiones requeridas es solamente dos: $p_1 + p_2 + p_3$ y $p_1 + p_4$, ya que con éstas, es posible que cada uno de los clientes logre decodificar el paquete requerido por cada uno de ellos.

Figura 4.2: Ejemplo de problema de comunicación de Difusión General de un solo salto

Dentro de este artículo se demuestra que en general, la obtención del mínimo número de transmisiones en redes inalámbricas de este tipo es un problema NP-Completo.

Con lo antes mostrado, encontramos que existen un gran número de tópicos dentro de las cuales, el empleo de la codificación en redes aún no ha sido explorado del todo.

Bibliografía

[1] R.K. Ahuja. *Network Flows.* Prentice Hall, 1993.

[2] S. Jaggi; P. Sanders; P. Chou; M. Effros; S. Egner; K. Jain; L. Tolhuizen. Polynomial Time Algorithms for Multicast Network Code Construction. *IEEE Transactions on Information Theory, Vol. 51, No. 5*, pages 1973–1982, 2005.

[3] R.E. Tarjan J.W. Suurballe. A Quick Method for Finding Shortest Pairs of Disjoint Paths. *Networks, Vol. 14, Issue 2*, pages 325–336, 1984.

[4] A. Rasala Lehman. *Network Coding.* PhD thesis, Department of Electrical Engineering and Computer Science, MIT, 2005.

[5] R. Ahlswede; N. Cai; S. Robert Li; R. Yeung. Network Information Flow. *IEEE Transactions on Information Theory, Vol. 46, No. 4*, pages 1204–1216, 2000.

[6] N. Cai S. Robert Li, R.Yeung. Linear Network Coding. *IEEE Transactions on Information Theory, Vol. 49, No. 2*, pages 371–381, 2003.

[7] M. Medard R. Koetter. An Algebraic Approach to Network Coding. *IEEE/ACM Transactions on Networking, Vol. 11, No. 5*, pages 782–795, Octubre, 2003.

[8] E. Lehman A. Rasala Lehman. Complexity Classification of Network Information Flow Problems. *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 142–150, 2004.

[9] D. Wagner U. Brandes. A Linear Time Algorithm for the Arc Disjoint Menger Problem in Planar Directed Graphs. *European Symposium on Algorithms*, pages 64–77, 1997.

[10] N. Shacham R.G. Ogier, V. Rutenburg. Distributed Algorithms for Computing Shortest Pairs of Disjoint Paths. *IEEE Transactions on Information Theory, vol. 32, no. 9*, pages 443–455, Marzo, 1993.

[11] J. Bruck M. Langberg, A. Sprintson. Network Coding: A Computational Perspective. *The 40th Annual Conference on Information Sciences and Systems*, pages 877–882, Marzo, 2006.

[12] J. Widmer C. Fragouli, J. Le Boudec. Network Coding: An Instant Premier. *ACM, SIGCOMM Computer Communication Review, Vol. 36*, pages 63–68, Enero, 2006.

[13] E. Soljanin C. Fragouli. *Network Coding Fundamentals.* NOW Publishers Inc., 2007.

[14] R. Perez Cupe. Algoritmo de etiquetas para el problema del flujo máximo. Encontrada en el sitio de la Universidad de Lima, Perú: http://fc.uni.edu.pe/publicaciones, 1995.

[15] U.S.R. Murty J.A. Bondy. *Graph Theory with Applications.* North-Holland, 1982.

[16] J. Bruck M. Langberg, A. Sprintson. The Encoding Complexity of Network Coding. *IEEE Transactions on Information Theory, Vol. 14*, pages 2386 – 2397, Junio, 2006.

[17] R. Bhandari. *Survivable Networks: Algorithms for Diverse Routing.* Kluwer Academic Publishers, 1999.

[18] N. Lynch. *Distributed Algorithms.* Ed. Morgan Kaufmann, 1996.

[19] P. Sanders S. Jaggi. Polynomial Time Algorithms for Multicast Network Code Construction. *Transactions on Information Theory, IEEE Volume 51, Issue 6*, pages 1973 – 1982, Junio, 2005.

[20] T. Ho; R. Koetter; M. Medard; D.R. Karger; M. Effros. The benefits of coding over routing in a randomized setting. *Information Theory, 2003. Proceedings. IEEE International Symposium on Volume , Issue , 29*, pages 442 –, Junio, 2003.

[21] T. Ho; B. Leong; Yu-Han Chang; Yonggang Wen; R.Koetter. Network Monitoring in Multicast Networks Using Network Coding. *International Symposium on Information Theory, ISIT 2005. Proceedings*, pages 1977–1981, Septiembre, 2005.

[22] R. D.Ñowak M. G. Rabbat, M. J. Coates. Multiple Source Internet Tomography. *Journal on Selected Areas in Communications, IEEE*, pages 2221–2234, Diciembre, 2006.

[23] A. Sprintson S. Y El Rouayheb, M. Asad R. Chaudhry. On the Minimum Number of Transmissions in Single-hop Wireless Coding Networks. *Information Theory Workshop, IEEE*, pages 120–125, Septiembre, 2007.