INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN ELECTRONICA, COMPUTACION, INFORMACION Y COMUNICACIONES

DISEÑO Y OPTIMIZACION DE REDES, MENTOR® Vs. PROGRAMACION LINEAL

T E S I S

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO ACADEMICO DE:

MAESTRO EN ADMINISTRACION DE LAS TELECOMUNICACIONES

POR

BOLIVAR ALONSO SOSA

MONTERREY, NUEVO LEON JUNIO 2005

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN ELECTRÓNICA,
COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

DISEÑO Y OPTIMIZACION DE REDES, MENTORII Vs. PROGRAMACION LINEAL

TESIS

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO ACADEMICO DE:

MAESTRO EN ADMINISTRACIÓN DE LAS TELECOMUNICACIONES

POR:

BOLIVAR ALONSO SOSA

MONTERREY, N. L. Junio 2005

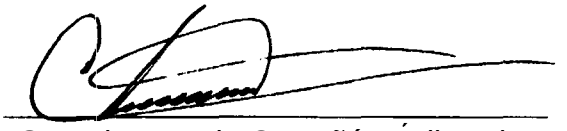
INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES DE MONTERREY

DIVISIÓN DE ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

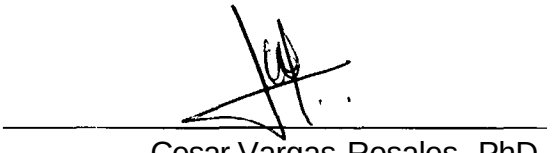
PROGRAMAS DE GRADUADOS EN ELECTRÓNICA, COMPUTACIÓN, INFORMACIÓN Y COMUNICACIONES

Los miembros del comité de tesis recomendamos que la presente tesis del Ing. Bolívar Alonso Sosa sea aceptada como requisito parcial para obtener el grado académico de Maestro en Administración de las Telecomunicaciones.

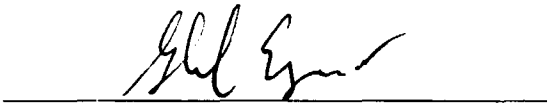
Comité de tesis:



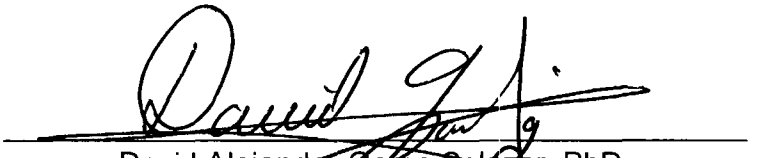
Gerardo Antonio Castañón Ávila, PhD.
Asesor



Cesar Vargas Rosales, PhD.
Sinodal



Gabriel Campuzano Treviño, PhD.
Sinodal



David Alejandro Garza Salazar, PhD.
Director del Programa de Graduados en Electrónica, Computación, Información y Comunicaciones.
Junio 2005

DISEÑO Y OPTIMIZACION DE REDES, MENTORII Vs. PROGRAMACION LINEAL

POR:

BOLIVAR ALONSO SOSA

TESIS

Presentada al Programa de Graduados en Electrónica, Computación, Información y Comunicaciones.

Este trabajo es requisito parcial para obtener el grado de Maestro
En Administración de las Telecomunicaciones

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY

Junio 2005

Dedicatoria

Con mucho cariño y amor para mi madre, mi padre y mis hermanas ya que sin ellos no seria lo que soy ahora; por su apoyo incondicional a lo largo de mi vida, porque siempre han estado y estarán en mi corazón.

Agradecimientos

A la Profa. Heriberta Sosa que a pesar de las adversidades está cada vez más firme que un roble y me ha enseñado el camino de la vida con gran sabiduría, en sus diferentes facetas de madre y amiga.

Al Diputado Miguel Alonso Raya el cual es para mí sinónimo de rectitud y le tengo un gran respeto no solo por ser mi padre sino porque sus logros son una inspiración para mi.

A mis hermanas la Ingeniera Haydé y la Lic. Tania que son mis tesoros más preciados y me han enseñado a sonreírle a la vida; son una inspiración para mí.

Al Ingeniero Albores por sus sabios consejos que fueron de gran ayuda.

A mi comité de tesis, por su gran apoyo para que ésta sea una realidad.

A mi asesor PhD. Gerardo Castañón Ávila que siempre me tendió la mano durante toda la tesis y siendo un ejemplo a seguir.

A mis amigos Angel y Ernesto que en todo momento me hacen ver lo bello de la vida.

A las nuevas amistades que hice en esta aventura a los cuales recuerdo con alegría y un cariño especial.

A todos mis cuates de Veracruz que siempre llevo en mi corazón con mucha alegría.

Resumen

Este documento exterioriza la inquietud sobre un buen diseño de redes de área metropolitana en base a los procesos como indica el titulo de esta tesis MENTOR II vs. Programación Lineal.

Para comenzar se presenta la problemática la cual nos explica porque la necesidad de realizar este documento, en el cual se plasma el crecimiento de las redes y la necesidad de crear nuevos y mejores diseños.

Esto es enfocado a la idea de tener una mejor topología de redes. Y debido a la demanda de obtener mejores diseños con el paso del tiempo han surgido una gran variedad de métodos y procedimientos para la creación de redes y de los mas importantes se encuentra el algoritmo heurístico Mentor II el cual presenta ciertas mejoras a la versión anterior pero no quiere decir que sea la mejor opción ya que como antes se menciona estos métodos se encuentran en constante cambio y evolución y todos con la finalidad de obtener diseños óptimos. Y si nos basamos en la idea de buscar diseños que nos den una topología óptima hay que hablar de la programación lineal la cual, así como mentor II, presenta diseños con un alto grado de confiabilidad.

Siendo estas dos formas para diseñar redes y para tener un concepto mas claro que ventajas o desventajas ofrecen se harán comparaciones en base a simulaciones hechas con ambos métodos.

Dentro de este documento como parte de la estructura se muestra una descripción sobre redes, y un poco más detallado y completa el funcionamiento del algoritmo Mentor II, el cual se programó en C++. También se incluyen los fundamentos de programación lineal (PL) así como del AMPL el cual es la estructura para el modelo presentado en PL.

Por último se presenta las simulaciones frente a frente con la finalidad de tener una mejor perspectiva y opinión de ambos sistemas en base a los resultados obtenidos. Siempre teniendo en mente el de generar mejores redes en cuanto a costos y a su vez que presenten un formato confiable para su desempeño.

índice

Listado de Graficas

Listado de Tablas

Capítulo 1

Introducción

El crecimiento de la comunicación de datos menciona Nicholas Osifchin (2000), el Internet y el comienzo de una convergencia que seguida de una desregulación mundial provocaron una nueva era en el establecimiento de una red y aumentaron el número de proveedores de servicio competitivos. La nueva era de las redes será capaz de transportar y procesar voz, video, datos y el tráfico de Internet a muy altas velocidades a un costo mas bajo que el actual. Esto junto con el rápido crecimiento del Internet, están llevando a una industria revolucionaria y cambiante para un entorno de negocios de las telecomunicaciones. Y como vemos en la Figura 1.1 con el pasar de los años la demanda de Internet ha aumentado considerablemente.

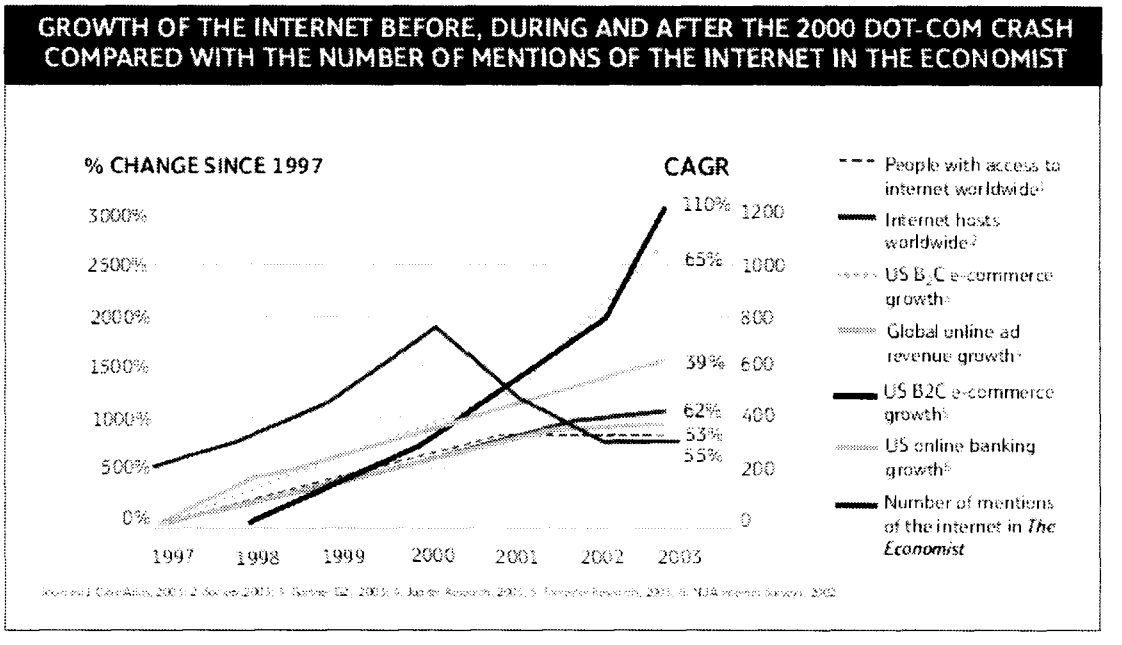


FIGURA 1.1 CRECIMIENTO DE INTERNET.
Tomado de Digital Networking Economy (White paper) 2004

El panorama mundial de la telecomunicación es marcado por una demanda creciente del ancho de banda, principalmente en el intercambio local, televisión por cable y redes corporativas privadas. Tal crecimiento es conducido por el Internet y usos del Intranet, servicios residenciales de banda ancha y servicios internacionales de alta velocidad de los datos (Pellegatta et. al, 2002).

El Sistema de fibra óptica, con sus altas tasas de la transmisión, están supliendo ya peticiones del mercado de telecomunicaciones. Los fabricantes y los portadores del sistema están mirando constantemente hacia las nuevas soluciones del transporte de datos para una transmisión a larga distancia, redes metropolitanas, así como para redes de acceso de banda ancha (Pellegatta et. al, 2002).

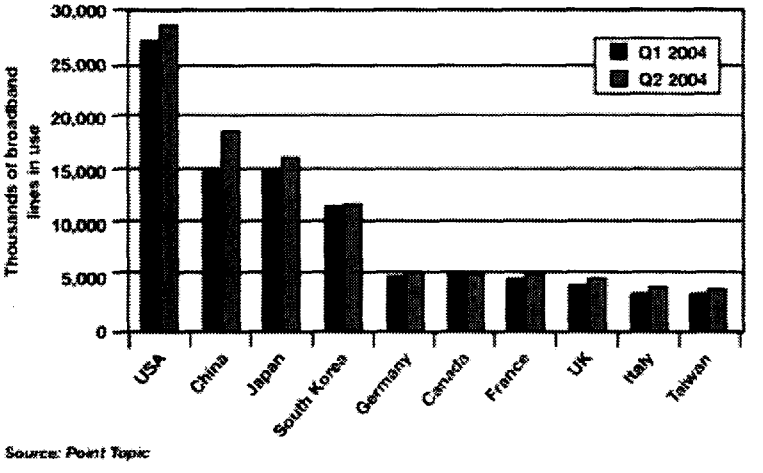


FIGURA 1.2. LÍNEAS EN USO DE BANDA ANCHA EN EL MUNDO 2004
Tomado de Optical Metro Networks and Beyond 2004

Las nuevas compañías competitivas están ofreciendo costos más bajos y están conduciendo a la introducción de nuevos servicios y por ende a un crecimiento notable en el número de líneas de banda ancha como se muestra en la figura 1.2. Las nuevas tecnologías están emergiendo, se están reduciendo las regulaciones del gobierno, y la industria se globaliza rápidamente, incluso compañías que han funcionado tradicionalmente en ambientes nacionales. El transporte eficiente de la información se está convirtiendo en un elemento dominante en la sociedad de hoy en día. Este transporte es apoyado por una infraestructura compleja de las comunicaciones que si está implementado y operado correctamente, es un elemento invisible para los usuarios finales. Estos usuarios finales parecen estar interesados sobre todo en servicios y costos solamente (Kazovsky et al, 1998).

Mientras que los nuevos servicios se desarrollan y las necesidades de los usuarios cambian, la industria debe adaptarse modificando las infraestructuras existentes o implementando nuevas redes. Por lo tanto desafían a los expertos de telecomunicaciones a crear mapas e itinerarios hacia las infraestructuras futuras. Esto es una tarea difícil debido al incompleto conocimiento de las tendencias en las demandas del usuario y cómo la tecnología se desarrollará (Kazovsky et al, 1998).

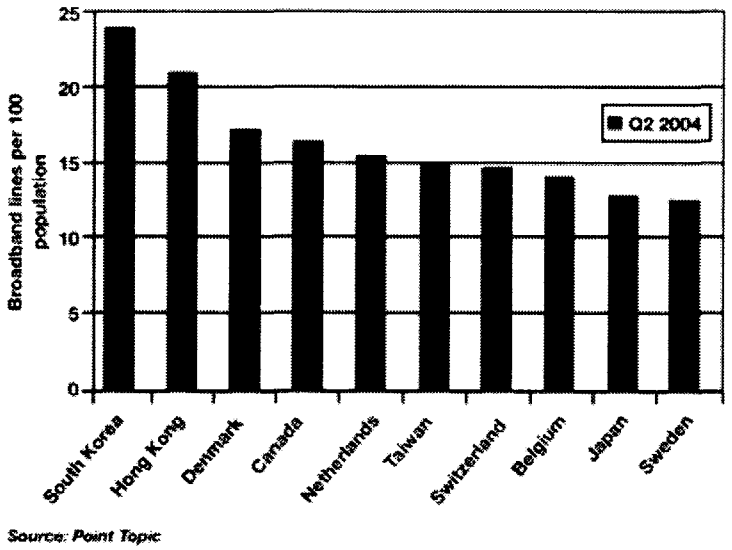


FIGURA 1.3 LÍNEAS DE BANDA ANCHA POR POBLACIÓN 2004
Tomado de Optical Metro Networks and Beyond 2004

Decina (2000) menciona que según la opinión de los usuarios cada vez más ven el Internet como medio importante para las comunicaciones, educación, reunión de la información, y comercio electrónico de mercancías y de servicios; por esta razón el uso de la líneas con velocidades mayores a 256 kilo bits por segundo (banda ancha) tiene una mayor demanda.

Existen dos posibilidades principales de la infraestructura de la información para la próxima década según Decina (2000): por un lado, la capacidad para construir un todo, de medios integrados, con seguridad, y una infraestructura rentable de la red con una trayectoria de transición de la situación actual a un objetivo a largo plazo; y, por el otro lado, el desarrollo de una plataforma de valor agregado de gran alcance de la creación y del despliegue de servicio.

Para el final de la próxima década dice Decina (2000), las aplicaciones de Internet constituirán una gran parte de los accesos del Internet en el mundo, con respecto a los accesos terminales para el uso humano.

1.1 Situación Problemática

El rápido crecimiento de la capacidad de la telecomunicación, conducido en parte por el vasto despliegue de la tecnología de la fibra óptica ha conducido a la creciente preocupación con respecto a la supervivencia de tales redes. En redes de comunicaciones, la supervivencia se define generalmente como el porcentaje del total del tráfico que sobrevive al fallo de alguna red en el peor de los casos. La mayoría de los modelos del diseño de redes propuestos hasta la fecha aseguran

indirectamente la supervivencia de la red invocando una restricción de la conectividad, para un número especificado de trayectorias entre cada par distinto de nodos en la red (SooMyung et al, 1999).

Con la integración de la computadora y de las tecnologías de comunicación y la rápida maduración de las técnicas ópticas de comunicación de la fibra, las redes de telecomunicación de hoy pueden proporcionar servicios rápidos y de alta calidad a las empresas mencionan Zhou y Subramaniam (2000).

¿Cómo prevenir la interrupción del servicio, y reducir la pérdida de servicio a un mínimo si la interrupción es inevitable?, Se convierte en una problemática crítica de acuerdo con Zhou y Subramaniam (2000); es decir, la supervivencia debe ser considerada al diseñar redes de telecomunicación. La supervivencia de una red se refiere a la capacidad de una red para proporcionar servicio continuo en la presencia de fallas.

La capacidad de una red de soportar y de recuperarse de fallas según Zhou y Subramaniam (2000), es uno de los requisitos más importantes de redes. Su importancia se magnifica en redes de fibra óptica con rendimientos de salida del orden de gigabits y de terabits por segundo.

Las organizaciones modernas en estos tiempos necesitan funcionar de manera rápida y mucho más eficiente para mantener su estado de competitividad. Existe una clara necesidad por las computadoras y sistemas de comunicaciones la cual permite a las personas poder trabajar juntos como si estuvieran dentro de la misma oficina a pesar de las distancias. La flexibilidad y aplicabilidad de estos sistemas se están viendo de manera muy vital en este mundo actual de negocios (Norris y Pretty, 2000).

La mayoría de las empresas operan en diferentes ubicaciones geográficas; la administración de estas organizaciones necesita tener un conocimiento actualizado de sus operaciones geográficamente dispersas para dar un mejor servicio a sus clientes, hacer frente a la competencia y vigilar de cerca las actividades críticas. Así las redes de telecomunicaciones permiten recolectar, procesar y distribuir de una manera veloz la información. También permiten tener servicios de valor agregado a las operaciones diarias de la empresa según Hernández (1994).

La demanda por obtener información de multimedia de las redes esta aumentando día con día. El diseñar algoritmos que distribuyan la información con garantías de calidad del servicio (QoS) en sistemas de tiempo real ha sido un problema de gran importancia para los investigadores (Huang, 2002).

Las fibras ópticas como medio de transmisión de redes se están convirtiendo rápidamente en el mayor medio de telecomunicaciones en redes MAN (Metropolitan Área Network) y WAN (Wide Área Network) debido a su gran

ancho de banda, su alta velocidad y su baja pérdida (Liu; Jaekel y Bandyopadhyay, 2002).

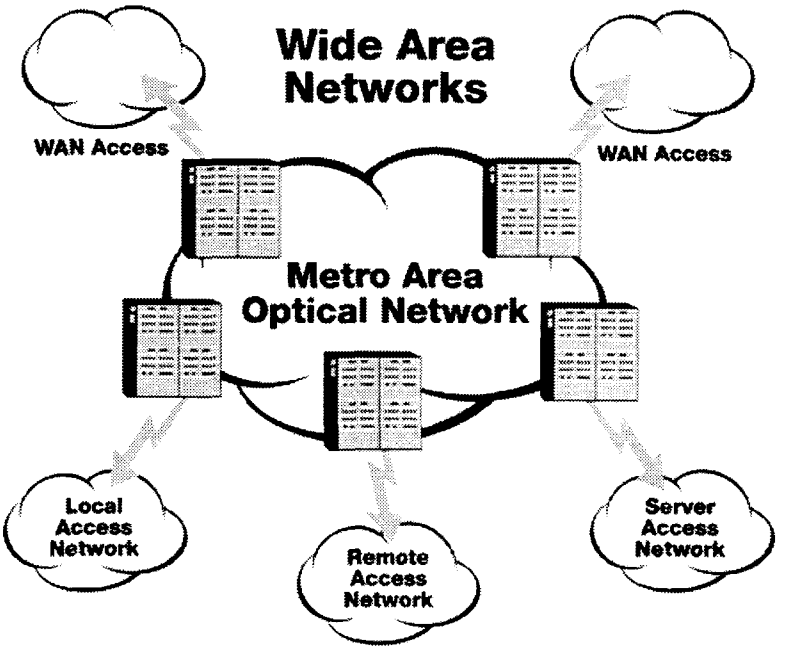


FIGURA 1.4 REDES LAN-MAN-WAN
Tomado de Building 10-Gig/DWDM MANs) 2000

El diseñador de la red debe aspirar a proveer una infraestructura de información que permita a la organización responder al mercado desarrollando fuentes disponibles, dondequiera que se encuentre. La red debe proveer fácilmente acceso seguro a la información dondequiera que esta se encuentre almacenada, y no debe requerir que el usuario final aprenda los detalles de la red cada vez que exista algún cambio. La figura 1.4 nos ilustra los elementos básicos de una red metropolitana y su relación con el acceso a larga distancia. Para tener una red se debe de tener una visión clara de los problemas claves (Norris y Pretty, 2000):

- Requisitos del usuario (servicios a soportar, características del tráfico, desempeño, capacidad de procesamiento, etc.);
- Necesidades empresariales (¿cuánto va a costar la red?, ¿cuan flexible y expandible necesita ser?, y ¿cómo encaja con las fuentes de redes existentes?);
- La manera de trabajar (un método sistemático para la combinación de usuario y empresa, necesidades y elecciones, integración y adaptación apropiada del equipo) (Norris y Pretty, 2000).

El diseño de redes es un poco compleja: una parte es científica, la otra concierne la habilidad o experiencia para diseñarlas. En muchas formas es lo mismo que cualquier otro diseño en el cual se establecen los requerimientos, encontrar los componentes relevantes, ajustarlos, integrar el conjunto en un todo, hacer pruebas sobre su funcionalidad y por último publicarlo para uso general (Norris y Pretty, 2000).

La planeación de las redes es algo complicado introduciendo varias limitantes como lo es el causado por los pros y contras de ciertos factores deseados. Muchos de los problemas en las tareas de administración y control de redes y la evaluación del desempeño deben ser estudiados más a fondo (Wei, 2003).

El diseño de una red es un reto interesante, demanda a los diseñadores a poner en la balanza las expectativas del desempeño con los costos de la red, capacidad, confiabilidad y utilización. Los operadores de las redes quieren mantener la utilización a niveles altos y los costos bajos. Las redes son una mezcla complicada de aplicaciones, protocolos, dispositivos, tecnologías de enlace, tráfico y algoritmos de ruteo. Generalmente hay miles de configuraciones posibles, y cada una con diferentes atributos de desempeño y costos (W. Bragg, 2000).

Las telecomunicaciones están evolucionando hacia un escenario caracterizado por la variedad de proveedores compartiendo fuentes y ofreciendo redes interconectadas. La evaluación del desempeño es una tarea fundamental para los operadores de las redes. Una razón es la búsqueda por un incremento en la satisfacción del cliente por medio de la detección rápida de fallas en la red; todo esto por medio de análisis estadísticos del comportamiento de la misma (Bertocco, Tittoto, Rizzi y Benetazzo, 2002).

Basándonos un poco en Norris y Pretty, (2000), establecen que para el proceso de diseño de redes, se deben de tener en cuenta cinco puntos importantes mostrados en la tabla comparativa 1.1.

1.1 Tabla Comparativa

Diseño de Redes
Autor: Norris y Pretty
Requerimientos -derivado de la declaración de que es lo que el cliente quiere de la red que se va a empezar a construir. Idealmente esto empieza desde la necesidad comercial. La declaración de lo que se requiere debe ser lo suficientemente detallada para que el diseño propuesto deba ser probado para su aceptación. Por consiguiente deberá contener declaraciones tanto cuantitativas como cualitativas. **Diseño de la arquitectura** -examinar los diferentes caminos en los cuales los requerimientos pueden ser conocidos y propuestos (usualmente después de presentar una lista corta al cliente) y cual opción debe ser adoptada. El diseño resultante debe dar una buena idea del tipo de tecnología a usarse.

Diseño de nivel superior -aquí muestra de manera anticipada el volumen del tráfico, los tipos y mezclas que pueden transmitirse en el diseño final. Muestra como la red será implementada en términos físicos y como fluirá el tráfico. También determina el número y la capacidad requerida de circuitos, terminales, etc.

Implementación del diseño -enfocándose en el aspecto lógico, así como en nombrar y direccionar las políticas del ruteo del tráfico. Verifica que los parámetros establecidos en los requerimientos sean factibles y que la capacidad correcta de los enlaces este disponible entre las localidades correspondientes.

Caso del negocio -esto cubre la parte capital y el actual gasto y deberá cubrir costos secundarios como lo es la actualización de software y hardware, mantenimiento general, licencias y servicios contratados.

Los procesos mostrados por Norris y Pretty en la parte de diseño de arquitectura y diseño de nivel superior son los procesos a destacar para este trabajo.

Dentro del diseño de las redes de telecomunicaciones encontramos ciertos aspectos y parámetros que son considerados importantes, tales como la optimización de la misma red tal y como se ha mencionado en párrafos anteriores.

Uno no diseña las redes de manera efectiva por suerte, existen técnicas a ser usadas, instrucciones a seguir, lecciones debido a la experiencia a ser aplicadas. Hay una gran variedad de ejemplos de redes que proveen competitividad y beneficios, enriquecen el trabajo de los usuarios diariamente, incluso contribuyen al entorno por medio de la reducción de un viaje innecesario. También existen ejemplos de redes que no han sido lo esperado o lo suficientemente efectivas como se esperaba, lo que ha llevado solamente a la pérdida de tiempo y dinero. (Norris y Pretty, 2000).

Los procedimientos o pasos para la búsqueda de soluciones óptimas a estos problemas de diseño de redes de telecomunicaciones son los llamados *Algoritmos* o metodologías de optimización. Por ejemplo la programación lineal es un procedimiento bien definido que encuentra un punto extremo óptimo o determina que el problema no es factible o que es ilimitado. Mucha de la investigación en centros de optimización gira en torno a encontrar, mejores y más eficientes algoritmos para resolver problemas (R. Evans y Minieka, 1992).

Otro de los autores menciona también como la importancia del algoritmo evolutivo (EAs) como una herramienta poderosa para la optimización en ingeniería, y esto ha sido mostrado ampliamente a través de los últimos años con una vasta cantidad de aplicaciones. Pero uno de los problemas al utilizar estos algoritmos (EAs) en aplicaciones prácticas es el cómo diseñar la función adecuada (Zeng, Ding y Kang, 2002).

En la tabla 1.2 vemos una recopilación de algunos algoritmos utilizados para la optimización de redes cada uno con sus características propias de un algoritmo único; aunque aquí sólo hay algunos, no quiere decir que son todos, existe una gran variedad de ellos.

1.2 Tabla de Algoritmos.

Algoritmos
> *Algoritmos Genéticos* Los algoritmos genéticos, son algoritmos de búsqueda robusta para optimización basados en los mecanismos de selección natural y genética. Presenta una búsqueda envolviendo a una población de solución de candidatos a través de operandos no determinísticos (Sayoud y Takahashi, 2001).
> *Algoritmo "Flow Deviation (FD)"* Este algoritmo encuentra un conjunto de rutas o trayectorias que minimizan el atraso, dada (a topología de la red y un conjunto de requerimientos. Se asume una teoría de colas para calcular el atraso en los enlaces. El problema se enfoca en el atraso en los enlaces, sin considerar el atraso en los nodos, ya que no influye en gran manera en el análisis que se describe (Beltrán, 1996).
> *Algoritmo de Bertsekas-Gallager (BG)* Otro algoritmo utilizado es el propuesto por el Doctor D. Bertsekas y R. Gallager (Beltrán, 1996). Este define el algoritmo en 2 formas: 1. Se mueve un par fuente-destino a la vez. 2. Se calcula la cantidad de flujo a mover directamente, en lugar de establecer una línea de búsqueda binaria. El algoritmo BG calcula la cantidad de flujo a mover mediante una aproximación de la segunda derivada d e atraso con respecto al flujo, y utilizando este valor como un factor de corrección.
> Algoritmo MENTOR El algoritmo MENTOR ("Mesh Network Topology Optimizaron and Routing) sirve para generar la red "backbone" con las redes de acceso de topología estrella. Inicializando uno de sus parámetros a un cierto valor, considera todos los nodos como conmutadores y genera la red de "backbone" únicamente (Beltrán, 1996).

Este estudio es importante ya que existen muchos y variados algoritmos como se muestran en esta tabla, el asunto aquí es el hecho de que cada algoritmo llega a cierto puntos de optimización y lo que se quiere lograr con este trabajo es

llegar a los puntos mas óptimos de estos algoritmos, es decir, llegar lo más cercano al mejoramiento en cuanto a desempeño, costos y confiabilidad de redes, pero estos algoritmos no solo trabajan sobre un tipo de localización de nodos, sino que pueden diseñar redes independientemente de su localización, es por ello que, es importante establecer y poder mostrar que el algoritmo Mentor II utilizado en este trabajo funciona bajo estas circunstancias.

1.2 Planteamiento del Problema

Al diseñar una red se debe tener en cuenta dos preguntas que se hacen constantemente: ¿Cuánto dinero se necesita invertir para tener una red con mayor utilización?, ¿qué tipo de mejoría de la red se puede comprar? Y ¿qué mejora se le puede hacer a la red? (Cahn, 1998)

La respuesta a ambas preguntas dependerá del costo del servicio básico de la red y de los componentes de la misma. Si los servicios están disponibles y son baratos, es posible que se decida no construir una red en absoluto sino simplemente utilizar la red pública. Si la red pública no satisface la necesidad o no ofrece lo necesario para la empresa como por ejemplo, garantías de rendimiento, en ese caso no habrá otra alternativa más que construir la red. En cualquier caso concentrarse en las técnicas de algoritmos nos ayudara a responder las siguientes preguntas: ¿Qué tan barata puede ser la red? Y ¿cómo debe de ser la red? (Cañan, 1998).

El problema de una transmisión de datos efectiva envuelve el diseño de sub-redes de comunicación. Un problema crítico en el diseño de redes es encontrar redes de comunicación que tanto el costo de las trayectorias seleccionadas entre par de nodos sea minimizado, y las limitantes de la capacidad de la red y su confiabilidad sean conocidas. El problema de diseño de redes da origen a un problema de optimización (Lo, 2000).

En el proceso de diseño de redes existen 2 factores importantes que deben ser tomados en cuenta: el rendimiento y confiabilidad. La confiabilidad en particular puede ser descrita de muchas maneras, existen 2 que son las más comunes. La confiabilidad del sistema está definido como la probabilidad de que el sistema este operando cierto tiempo. Esta medida de confiabilidad es de especial importancia para redes que deben estar operando 24 hrs.; como sería el caso por ejemplo de las redes de un hospital donde la vida de paciente podría depender de que cierto sistema permanezca funcionando. Por otro lado la accesibilidad es más adaptable para redes que deben operar en base a horarios en lugar de hacerlo continuamente. En una organización donde los negocios se hacen a través de las redes es necesario que funcionen la mayor parte del tiempo para minimizar pérdidas en los negocios (Sauvé y Silva, 2001).

Toda red tiene tres características: costo, desempeño, y confiabilidad. Ninguna de estas características es tan fácil de cuantificar. El diseñador debe presentar la red y decir que el costo es $100,000/mes, el promedio de retraso es 92 milisegundos, y la confiabilidad es de 0.9889; otro diseñador podría presentar la misma red y describir el costo de $99,720, el retraso de 97 milisegundos, y la confiabilidad como 0.991. Probablemente ninguno este equivocado. Ambos están haciendo su mejor esfuerzo. El desacuerdo podría ser porque ellos están utilizando diferentes tarifas y diferentes modelos de redes (Cahan, 1998).

En esta situación los autores (Arabas y Kozdrowski, 2001) manejan que los algoritmos evolutivos (EAs) en las telecomunicaciones han tenido un despunte bastante interesante, especialmente en el campo de diseño de redes, ruteo, asignación de frecuencias, planeación de capacidad, control de admisión y administración de redes entre otras.

Las redes reales son construcciones multidimensionales. Tienen costo, desempeño y confiabilidad. Por separado, ninguno de estos factores puede dar suficiente información para decidir el valor de la red. Si dos redes tienen diferente costo, la pregunta en la mente del diseñador deberá ser, ¿qué obtengo por dinero extra? (Cahan, 1998).

1.3 Objetivo

1.3.1 Objetivos Generales:

El objetivo de este trabajo es comparar los resultados del algoritmo MENTOR II con la solución óptima obtenida utilizando métodos de programación lineal a través de un paquete llamado CPLEX el cual resuelve la optimalidad de los problemas matemáticos que se le introduzcan.

El problema es el análisis de algoritmos para diseño de redes ópticas y la comparación de estos algoritmos con la solución "óptima" de la red utilizando técnicas de optimización de programación matemática o programación lineal. MENTOR II (Cahan, 1998) es uno de los algoritmos ampliamente conocidos para el diseño de redes de telecomunicaciones que optimiza el costo, eficiencia, y confiabilidad de la red. Este trabajo consiste en comparar el algoritmo MENTOR con la solución óptima obtenida utilizando métodos de programación lineal y un paquete de optimización llamado CPLEX.

CPLEX optimiza el problema matemático que se le da de entrada y da una solución muy aproximada a la solución óptima y en la mayoría de los casos encuentra la solución óptima. En este trabajo se hará un estudio estadístico comparativo de estas formas de diseñar redes telecomunicaciones (MENTOR vs.

técnicas de PL). El análisis estadístico pretende comparar una muestra significativa de diferentes topologías. La comparación se hará en base a costo, confiabilidad y eficiencia de las redes diseñadas por las dos técnicas.

Al querer definir qué es óptimo para este trabajo, se tomó en cuenta muchos factores tanto lo que se busca en la investigación así como la diversidad de conceptos de cada autor, dentro de los cuales tenemos algunos de ellos a continuación:

Podemos decir que optimalidad es tener redes en las cuales se presentan de manera confiable, con el desempeño establecido, buscando un costo adecuado a cada situación problemática presentada.

Usualmente en muchos casos no es fácil obtener la solución óptima o simplemente no se puede encontrar la solución óptima como concepto tradicional. En estos casos, la estrategia es escoger la "solución satisfactoria" en lugar de la "solución óptima" (Xianhai, Weidong y Dúo, 2003). Entendiendo como solución satisfactoria el hecho de ser una solución cercana o lo más posible a lo esperado en la investigación.

El problema en sí es la gran variedad de técnicas de optimización de redes y que todas ellas llegan a ciertos niveles de optimización y cada una busca su propio punto de optimización, es por ello que se busca mejorar estos rangos a un nivel de optimalidad algorítmica.

1.3.2 Objetivos específicos:

Analizar, utilizando métodos estadísticos, los diferentes tipos de topologías y determinar, con un número aceptable de muestras (topologías), el grado de confiabilidad del algoritmo MENTOR II con respecto a optimizar costo, retardo y confiabilidad.

Se entiende como algoritmo óptimo aquel que cumple con las características de optimización de redes, en el cual entran los rangos del desempeño de la red, la confiabilidad y los costos de la misma.

Muestra:

Distintas topologías clasificadas de acuerdo a:

- Localización de Nodos
- Tráfico
- Distancia
- Costo de Línea

Variables:

- Costo (tráfico, distancia, localización de nodos)
- Confiabilidad
- Retardo

Medición de variables:

Los instrumentos a utilizar para mejorar las redes estarán basados en dos métodos: el Algoritmo MENTOR II y programación lineal basándonos en CPLEX

Capítulo 2

Teoría de Grafos

En esta parte empezaremos a darle estructura a la teoría para el diseño de redes metropolitanas; dentro de lo básico encontramos los grafos, árboles y rutas mas cortas que nos ayudarán a determinar la mejor estructura de la red en base a los puntos importantes dentro del objetivo. Después entraremos un poco más en lo establecido de los parámetros como son los costos, el tráfico y la confiabilidad de la red. Y ya dentro del tercer capitulo veremos de manera mas explícita el algoritmo a utilizar, que en este caso es MENTOR II.

2.1 Grafos y Árboles.

Un grafo es un objeto matemático que se utiliza para representar circuitos, redes, etc. Los grafos son muy utilizados en computación, ya que permiten resolver problemas muy complejos.

Imaginemos que disponemos de una serie de ciudades y de carreteras que las unen. De cada ciudad saldrán varias carreteras, por lo que para ir de una ciudad a otra se podrán tomar diversos caminos. Cada carretera tendrá un costo asociado (por ejemplo, la longitud de la misma). Gracias a la representación por grafos podemos elegir el camino más corto que conecta dos ciudades, determinar si es posible llegar de una ciudad a otra, si desde cualquier ciudad existe un camino que llegue a cualquier otra, etc.

El estudio de grafos es una rama de las matemáticas muy importante. Estudiaremos primero sus rasgos generales y sus recorridos fundamentales, para tener una buena base que permita comprender los algoritmos que se pueden aplicar.

Un grafo consta de vértices (o nodos) y aristas (o arcos). Los vértices son objetos que contienen información y las aristas son conexiones entre vértices. Para representarlos, se suelen utilizar puntos para los vértices y líneas para las aristas.

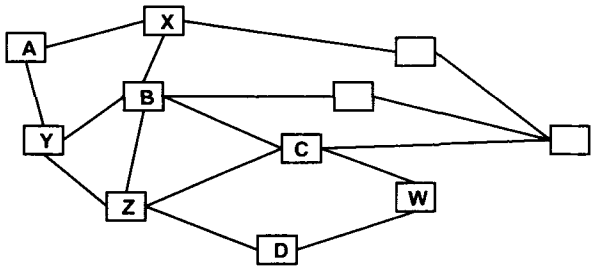


FIGURA 2.1 GRAFO.

- Un camino (walk) es una secuencia de nodos (n1, n... nk) en la que cada par de nodos adyacentes es un arco.

Un camino entre dos vértices es una lista de vértices en la que dos elementos sucesivos están conectados por una arista del grafo. Así, el camino AJLOE es un camino que comienza en el vértice A y pasa por los vértices J, L y O (en ese orden) y al final va del O al E. El grafo será conexo si existe un camino desde cualquier nodo del grafo hasta cualquier otro. Si no es conexo constará de varias componentes conexas.

- Una ruta (path) es un camino sin nodos repetidos

Camino (1, 2, 3, 4, 2) Ruta (1, 2, 3, 4)

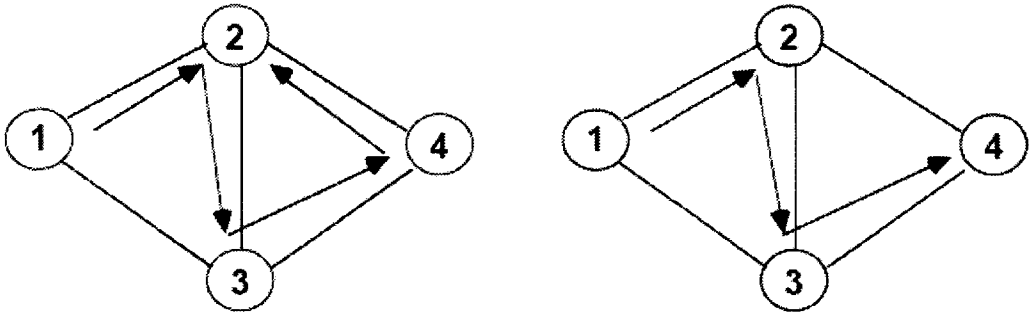


FIGURA 2.2 CAMINO (A) Y RUTA (B)
Tomado de Enrutamiento de amplia difusión (broadcast). (2000).

- **Longitud de un camino:** número de arcos del camino.

Un camino simple es un camino desde un nodo a otro en el que ningún nodo se repite. Si el camino simple tiene como primer y último elemento al mismo nodo se denomina ciclo. Cuando el grafo no tiene ciclos tenemos un árbol (ver figura 2.3). Varios árboles independientes forman un bosque. Un árbol de expansión de un grafo es una reducción del grafo en el que solo entran a formar parte el número mínimo de aristas que forman un árbol y conectan a todos los nodos.

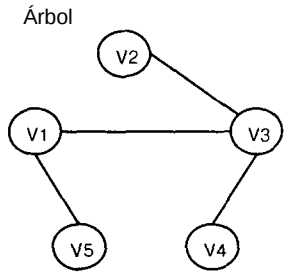


FIGURA 2.3 ARBOL

Las aristas son la mayor parte de las veces bidireccionales, es decir, si una arista conecta dos nodos A y B se puede recorrer tanto en sentido hacia B como en sentido hacia A: estos son llamados grafos no dirigidos. Sin embargo, en ocasiones tenemos que las uniones son unidireccionales. Estas uniones se suelen dibujar con una flecha y definen un grafo dirigido.

Cuando las aristas llevan un costo asociado (un entero al que se denomina peso) el grafo es ponderado. Una red es un grafo dirigido y ponderado.

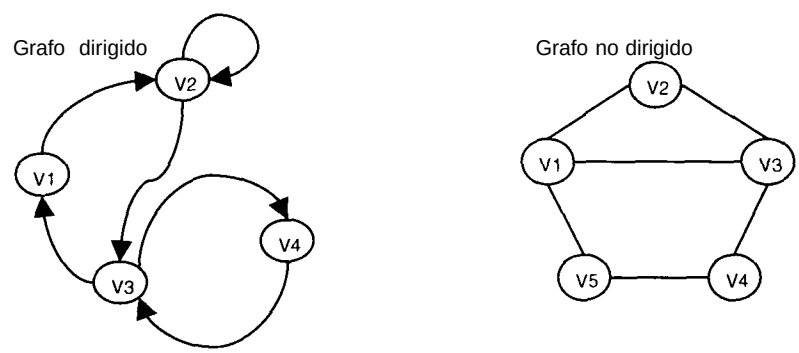


FIGURA 2.4 TIPOS DE GRAFOS

- Un ciclo es un camino (n1, n2... nk) con n1 = nk, k>3, y sin nodos repetidos excepto n1 = nk

Ciclo (1,2,4,3,1)

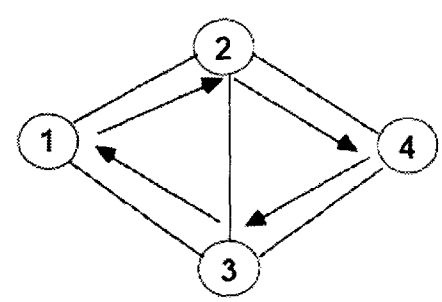


FIGURA 2.5 CICLO
Tomado de Enrutamiento de amplia difusión (broadcast). (2000).

- G' = (N',A') es un subgrafo de G = (N, A) si:
  > G' es un grafo

> N' es un subconjunto de N
> A' es un subconjunto de A

- Un subgrafo se obtiene eliminando nodos y arcos de un grafo:

- Nota: los arcos adyacentes a un nodo eliminado también deben ser eliminados

Grafo G Subgrafo G' de G

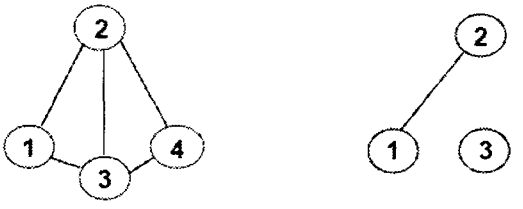


FIGURA 2.6 SUBGRAFOS
Tomado de Enrutamiento de amplia difusión (broadcast). (2000).

2.2 Generación de Población para los nodos.

Al tener una noción más clara de lo que son los grafos podemos continuar con la delineación de la red. Como ya sabemos, se planea desarrollar una red de área metropolitana (MAN) y cada nodo de la misma debe tener cierta población que a su vez genera un tráfico y un costo para el diseño de la red.

Ahora ya que tenemos esta información empezaremos a generar lo que es nuestra población, con la ayuda del posicionamiento de cada nodo, el cual se presenta a continuación.

En el mundo real existen 2 sistemas de coordenadas utilizadas, la vertical-horizontal (V&H) y latitud-longitud (L&L). Básicamente el sistema de coordenadas V&H permiten procesos computacionales de manera simplificada (Cahn, 1998).

Si dos ciudades tienen coordenadas (v_p/z,) y (v_2,/i,), entonces la distancia entre ellas es calculada para el propósito de las tarifas por la formula:

$$\bullet dist = ceil \wedge\overline{(dv*dv)/10 + (dh*dh)/\backslash 0^{1})} \qquad (2.1)$$

$$dist = ceil\left(\sqrt{\frac{dv^2}{10} + \frac{dh^2}{10}}\right) \qquad (2.2)$$

donde $dv = v_{,} - v$, y $dh = h_{-,} - h_{j}$. La función ceil regresa el número entero mas pequeño mayor o igual al número.

La distancia será utilizada para las tarifas de los circuitos entre dos localidades. Las tarifas de circuitos especifican que el cargo al consumidor es calculado utilizando una línea directa entre los puntos finales (Cahn, 1998).

Para estar dentro de los propósitos de esta tesis y poder presentar un trabajo consistente y tener un punto de referencia, se apoyó el esquema de coordenadas utilizados en el ejemplo mux1 del programa Deiite para diseño de redes, en el cual también se plantea una cantidad de población (Pop) constante, y las coordenadas Vertical (V) y horizontal (H) fijas. El cual se muestra en la tabla siguiente:

Nodo	Pop	V	H
1	1	6624	2555
2	1	5975	3690
3	1	7996	2543
4	1	7220	2715
5	1	6564	2394
6	1	5265	3232
7	1	5876	3163
8	1	8109	3692
g	1	7421	4044
10	1	7425	4467
11	1	7602	2480
12	1	6977	2gg5
13	1	6096	4900
14	1	6643	3268
15	1	6918	4483

Con lo anterior se crean 2 vectores, los cuales serán utilizados por el algoritmo MENTOR II. Continuaremos con la obtención del tráfico por medio de las fórmulas propuestas en la sección 2.4.

2.3 Generación del Tráfico de la red.

Las redes futuras deberán servir a nivel cliente, en lo que incluya paquetes de red, como por ejemplo Internet, el cual tiene un alto patrón de dinamismo en cuanto a conexión y una porción significativa de flujo de tráfico intermitente entre los pares comunicantes. Se requerirá para esto una mayor cantidad de conexiones y la reserva de ancho de banda para cada conexión para evitar un excesivo retraso en el ingreso de los ruteadores (Conferencia Internacional de Telecomunicaciones, 2003).

Generalmente, es el diseñador de la red quien determina como modelar el tráfico de cada población. Esto puede ser hecho tomando en cuenta registros personales, inventarios, información de la administración de red, o lo que se tenga a la mano (Cahn, 1998).

Para Cahn según Mendoza, si la población es homogénea, se puede utilizar un modelo basado en fenómenos físicos y establecer que el tráfico entre las poblaciones es una función de la población a alguna potencia dividida por la distancia elevada a otra potencia, en otras palabras:

$$Traf_a = a * - \frac{(Pop_i^{*} Pop_j)^{Pop\_Power}}{(dist(i,j))^{Dist\_Power}} \qquad (2.4)$$

Si se toma un modelo basado en la atracción gravitacional, tendríamos *Pop_Power* = 1 y *Dist_Power* = *2.* Si se considera un modelo basado en la teoría de magnetismo, entonces *Pop_Power* = 1 y *Dist_Power* = *3.* Cabe mencionar que se hace notar que existen redes en donde el tráfico es fuertemente dependiente de la distancia y otras redes en donde el tráfico tiene poca relación con la distancia. Si el tráfico es independiente de la población, entonces *Pop_Power* = *0.* Obsérvese que se esta utilizando *a* como un factor de escalamiento para colocar el tráfico en el rango que se desea (Cahn, 1998).

Al hacer este estudio se puede notar que en algunas poblaciones el valor numérico es de 7 dígitos y que la distancia entre ellas es de alrededor de 5 dígitos. Por lo tanto el producto de la población por la distancia tendría una magnitud de nueve dígitos mayores que la distancia, y el efecto de la distancia se perdería. Para normalizar los cálculos se define:

$$Pop\_\max = \max Pop_i \qquad (2.5)$$

Y

$$dist\_\max = \max dist(i,j) \qquad (2.6)$$

Además se definen dos pequeños números reales, positivos, Pop_{Iiff} y $Dist_{off}$. La finalidad de $Dist_{iiff}$ es evitar la división entre 0. El propósito de Pop_{off} es evitar que el tráfico hacia y desde pequeños nodos sea 0 (Cahn, 1998). Entonces tenemos que la función utilizada para la generación de tráfico es la siguiente:

$$Traf_{\alpha} = \frac{(Pop^{*} Pop \quad Y'''^{-Fiwr}}{[Pop\_max- \quad ''')} \qquad (2.7)$$

$$\left| \frac{}{dist\_\max} {}^{L-L\wedge-} + Dist_{off} \right|$$

2.4 Cálculo del Costo de la red.

El requerimiento de ancho de banda de una conexión típica de redes ópticas es referido a la demanda de tráfico. Es por ello que uno de los problemas en el diseño del "backbone" de la red es a causa de conectores ópticos (OCX) de diferentes niveles de ancho de banda y por ello se necesita minimizar la parte del costo de los puertos de conectores ópticos. (H. Zhu, K. Zhu, Zang, y Mukherjee, 2003).

El costo del diseño de la red dependerá del número de líneas a utilizar así como estas líneas varían en cuanto al ancho de banda y la distancia total recorrida entre los puntos de conexión; también esta el costo del equipo (enrutadores, switches, cross-connectors, etc.) que dependerá tanto del ancho de banda como del número de entradas y salidas de la topología a utilizar. Para esta investigación se utilizo lo que Cahn (1998) llama generador lineal de costo basado en distancias. El costo por kilómetro (o milla), se plantea de la siguiente manera.

Las tablas necesarias para el generador de costos son:

> Tabla con las coordenadas de localización de cada uno de los nodos en V&H.

> Tabla de tarifas como la que se muestra a continuación (Cahn, 1998).

Link-Type	Fixed_Cost	Dist_Cost	Dist1	Dist_Cost2
D64	900	24.35	300	0.95
F128	1950	4.80	100	3.00
T1	4500	10.25	200	7.00
E1	5550	12.50	200	9.50

Los resultados de este generador de costos es una matriz de tamaño *n*n .* Este algoritmo utiliza la siguiente condición para la generación de la tabla de costos.

$$C(iJ) = \begin{cases} [F + d*DC & d<Dist \\ [F + Disñ*DC + (d\text{-}Disñ)*DC2 & otherwise \end{cases} \quad \textbf{(2.8)}$$

Ya con esta información generada sabemos con exactitud del costo de conexión entre cada uno de los diferentes nodos de la red que se obtuvo en la sección anterior.

2.5 Diseño de Árboles.

Aquí veremos lo que son algunos algoritmos básicos que ocuparemos para encontrar rutas de los nodos dentro de un grafo ya sea por ruta mas corta, por costo, etc., que servirán de herramienta dentro de nuestro algoritmo de diseño de redes Mentor II.

De acuerdo con Mendoza (2004) los árboles son muy utilizados por un gran número de razones y serán utilizados como la base para muchos algoritmos y para técnicas de diseño y análisis. Primero, proveen conectividad sin agregar enlaces innecesarios. Segundo, proveen una ruta única entre cada par de nodos, se elimina el problema de ruteo (Ej., decidiendo como el tráfico debe fluir entre los nodos). Esto simplifica tanto la red como su diseño. Sin embargo, como los árboles están conectados en base a la ruta mas corta, también son muy poco confiables y robustos. Es por esto que las redes actuales se encuentran más interconectadas. Aún así, el diseño de una red parte de un árbol (Kershenbaum, 1993).

2.5.1 Minimum Spanning Trees (MST).

El árbol que se obtenga menciona Mendoza (2004) por lo general es arbitrario. Y por esta razón muchas veces es necesario encontrar el "mejor" árbol que se adapte a los requerimientos. Entonces, debemos asignar una longitud a cada enlace en la gráfica y buscar el árbol con la longitud total mínima. La "longitud" puede ser la distancia, el costo, o alguna medición de retraso o disponibilidad (Kershenbaum, 1993). Un árbol con el costo total mínimo es conocido como *minimum spannig tree.* Como definición se tiene:

> "Un grafo *G* es medido si existe un número real asociado con cada enlace. El peso de un enlace *e* será usualmente denotado como *w(e).* Y se denotara de manera usual este grafo *(G, w)*. Si *G'* es cualquier sub-grafo de *G,* entonces *w(G') = ^^w(e)"* (Cahn, 1998).

Se asumirá de acuerdo con Mendoza (2004) que el peso es una función *w* que denota el peso de los enlaces que es positivo, **w:£->¡** . Si *G* esta conectado, entonces se desea seleccionar el subgrafo conectado con el peso mínimo (Cahn, 1998).

En general, si el grafo no esta conectado, se desea encontrar el *minimum spannig forest,* el cual es un conjunto de enlaces de mínimo costo total el cual conecta el grafo lo mas posible.

2.5.2.1 Algoritmo de Prim.

En ocasiones es de mucha ventaja, especialmente si la red es muy densa, el considerar una aproximación alterativa para encontrar MST's. Este algoritmo es particularmente creado para implementaciones paralelas porque puede implementar operaciones utilizando vectores (Kershenbaum, 1993). El algoritmo de Prim puede describirse informalmente como:

- Empieza con un nodo en el árbol y todos los demás nodos fuera-del-árbol.
- Mientras existan nodos fuera-del-árbol,
-
    - Encuentra el nodo fuera-del-árbol que se encuentre más cercano al árbol.
    - Incorpora ese nodo al árbol y guarda el enlace que lo conecta con el árbol.

Existen diversas formas de explicar el algoritmo de Prim y lo mostrado por Jeltsch F. (2004) nos expresa a continuación un método de las etapas que debemos seguir:

Etapa 0: Escoge cualquier elemento r y S={r}, A, (siendo r la raíz del Árbol Expandido Mínimo(AEM), en donde A es el conjunto de las aristas, V conjunto de vértices.

Etapa 1: Hallar los arcos adyacentes más livianos de manera que un punto final pertenezca a S y el otro en V-S. Agregarlos a S y los más livianos a A. Ahora, la forma de cómo ir agregándolos es considerar el arco de menor peso, de manera que un punto pertenezca a S y el otro esté en V-S. Si V-S=A, entonces se detiene y entrega el AEM, de otra manera vuelve a la Etapai.

La idea antes mencionada se refiere a expandir el árbol actual agregando el arco más liviano y su punto final, tal como se aprecia en la figura, es decir,

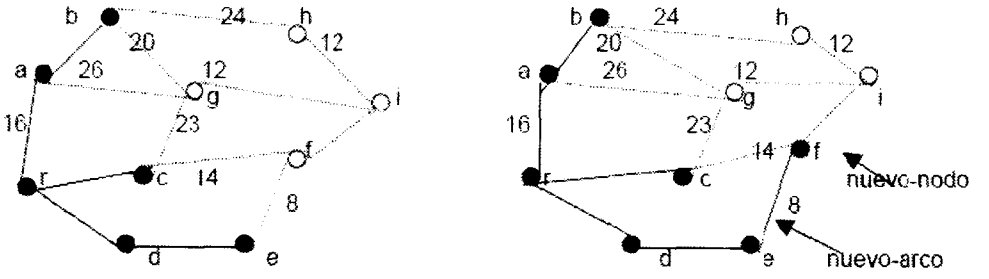


FIGURA 2.7 ARCOS.

Ejemplo de Prim:

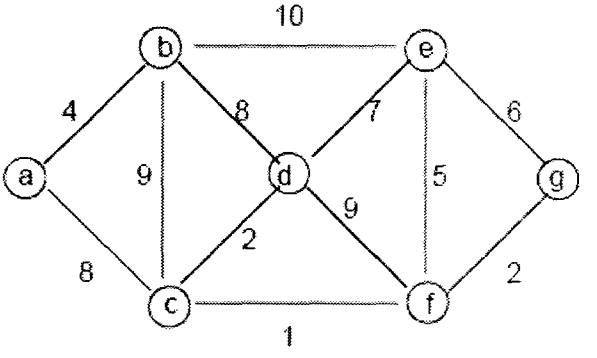


FIGURA 2.8 EJEMPLO DE PRIM.

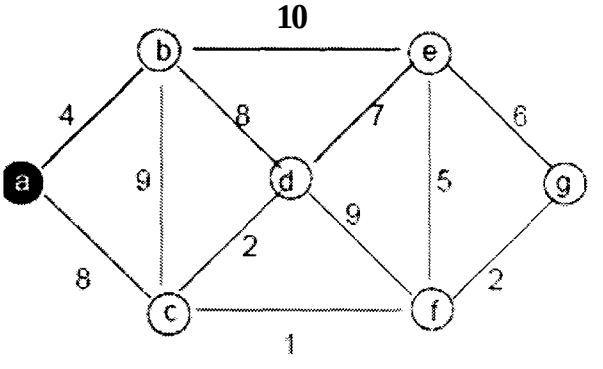


FIGURA 2.9 EJEMPLO 2 DE PRIM.

En la Fig. 2.8 se visualiza un grafo ponderado y conexo. En la Fig. 2.9 se visualiza la EtapaO. S={a}, V-S = {b, c, d, e, f, g}, con arco más liviano {a, b}.

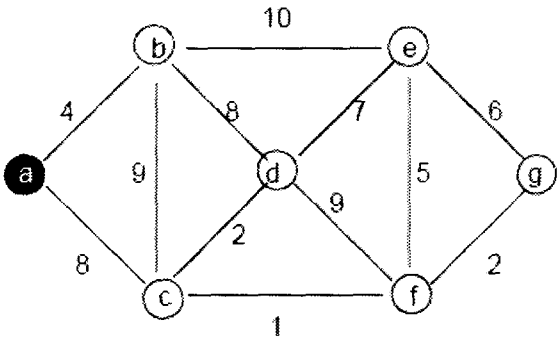


FIGURA 2.10 ETAPA 1.1 ANTES

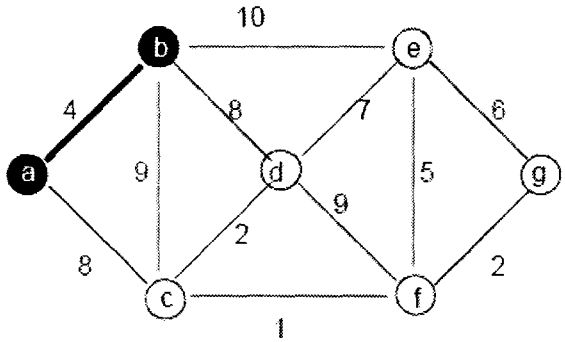


FIGURA 2.11 ETAPA 1.1 DESPUÉS.

En Fig. 2.10 se muestra la etapa1.1, "antes", con S={a}, V-S={b, c, d, e, f, 9}. A={ } y el arco más liviano {a, b}, mientras que en Fig. 2.11 se muestra la etapa1.1, después", en donde S={a, b}, V-S={c, d, e, f, g}, A={ {a, b} } y el arco más liviano es {b, d}, {a, c}. Continuando de esta manera llegamos a la Etapa1.6 "antes", con S={ a, b, c, d, f, g}, V-S= {e}, A={ {a, b}, {b, d}, {c, d}, {c, f}, {f, g}}, arco más liviano= {f, e}, como se muestra en Fig. 2.12. (Jeltsch F., 2004).

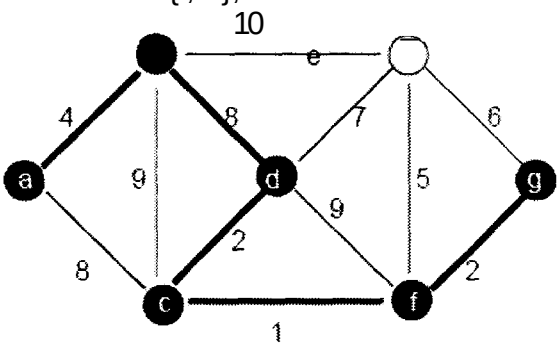


FIGURA 2.12 ETAPA 1.6 ANTES.

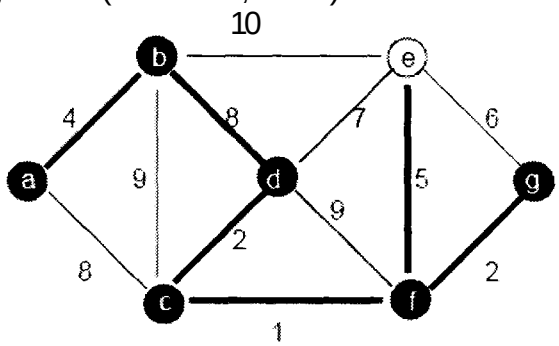


FIGURA 2.13 ETAPA 1.6 DESPUÉS.

Mientras que en Fig. 2.13 se tiene la Etapa1.6 "después" con S={ a, b, c, d, e, f, g}, V-S= {}, A={ {a, b}, {b, d}, {c, d}, {c, f}, {f, g}- {f, e}}. Así, de esta manera se logra finalmente obtener el árbol. (Jeltsch F., 2004).

2.5.3 Shortest Path Tree (STP).

El problema de encontrar la distancia más corta nos dice Mendoza (2004) juega un papel muy importante en el diseño y análisis de las redes. La mayoría de los problemas de ruteo pueden ser resueltos como problemas de distancias (rutas) mínimas. Una ves que se ha asignado "longitud" apropiada a cada uno de los enlaces (o arcos) en la red (Kershenbaum, 1993).

"Dado un grafo con pesos (G,w) y nodos n_1 y n_2, la ruta más corta de n_1 a n_2, es la ruta P tal que $\sum w(e)$ es mínima" (Cahn, 1998).

En el problema de la distancia mínima es lograr encontrar la ruta óptima (más corta) entre uno o más pares de nodos (Robertazzi, 1999). A continuación, se trabajara con modelo de grafo dirigido y se asumirá que l_{ij}, la distancia del arco entre cada par de nodos, i y j, es conocida. Estas distancias no necesitan ser simétricas. En al caso donde no exista un arco, se asume que l_{ij} es muy largo (más largo que n veces el mas largo de los arcos de al red). Inicialmente, se asume que l_{ij} es estrictamente positivo (Kershenbaum, 1993).

2.5.3.1 Algoritmo de Dijkstra.

Todos los algoritmos de distancias cortas se basan en la observación, la agrupación de las rutas mas cortas, esto es, si k, es parte de la ruta de i a j, entonces la ruta mas corta (i, j) debe ser la ruta más corta (i, k) seguida por la ruta mas corta (j, k) (Kershenbaum, 1993). Entonces, se puede encontrar la ruta mas corta utilizando la siguiente función recursiva:

$$d_{ij} = \min_k \left(d_{ik} + d_{kj}\right) \tag{2.9}$$

donde d_{ij} es la distancia mas corta de la ruta de i y j. La dificultad con esta aproximación es que debe de existir alguna manera para comenzar la función recursiva, dado que no comenzamos con ningún valor conocido en la parte derecha de la ecuación. Existen varias formas de hacer esto, cada una dando principio a diferentes algoritmos (Kershenbaum, 1993).

Una forma de proceder, como se muestra Dijkstra, es aceptable para encontrar la distancia mínima de un nodo, i, hacia todos los demás. Comienza por:

$$d_{ii} = 0 \tag{2.10}$$

y

$$d_{ij} = \infty \qquad \forall j \neq i \tag{2.11}$$

determinando

$$d_{ij} = l_{ij} \quad \forall j \quad \text{adyacente a } i \tag{2.12}$$

Entonces, encontrar el nodo, ./, con el mínimo d_{ir} y utilizarlo para tratar de mejorar las distancias de otros nodos, determinando:

$$d_{ik} = \longleftarrow \min \left(d_{ik}, d_{ij} + l_{ik} \right) \qquad (2.13)$$

En cada estado del proceso, el valor de d_{ik} es el estimado de la ruta mas corta de / a *k* y es, de hecho, las distancia de la ruta mas corta encontrada hasta el momento. Refiriéndose a d_{ik} como la etiqueta del nodo *k .* El proceso de utilizar un nodo para mejorar otros nodos es conocido como chequeo de nodos *(scanning node)* (Kershenbaum, 1993).

Una vez que un nodo es revisado se requiere que no se vuelva a etiquetar. Entonces, cada uno solo requiere ser examinado una vez. Si la etiqueta en un nodo cambia, debe ser nuevamente revisado (Kershenbaum, 1993).

Para hacer un poco más entendible la definición y el como funciona a continuación veremos una síntesis de los pasos a seguir junto con un ejemplo práctico de acuerdo con DaSilva (2002):

Objetivo: Dado un grafo G conectado y un nodo que sirve como raíz, encontraremos la ruta mas corta del árbol.

Hay que recordar que el algoritmo de Dijkstra sirve para encontrar la topología del árbol cuando ya se tiene una topología, es decir, encuentra el árbol (MST) y a la vez la ruta más corta en la red si se aplica a un conjunto de nodos entonces encuentra una topología tipo estrella.

Procedimiento:

1. Marcar todos los nodos como "sin examinar" y etiquetarlos con °° (infinito).
2. establecer la etiqueta del nodo raíz a 0 y su predecesor hacia el mismo.
3. Hacer lazos hasta que todos los nodos sean examinados
   3.1. Encontrar el nodo n con la etiqueta más pequeña.
   3.2. Marcar el nodo como "analizado".
   3.3. Examinar todos los nodos adyacentes, *m,* y checar si la distancia a la raíz a través de *n* es más corta que la etiqueta de m. Si es el caso, entonces actualizar la etiqueta y el predecesor.

Iteración 0: Aplicar el algoritmo de Dijkstra para encontrar la ruta más corta arraigada a D. Inicializar con el nodo D con etiqueta de valor 0 y el resto de los nodos con etiqueta °°.

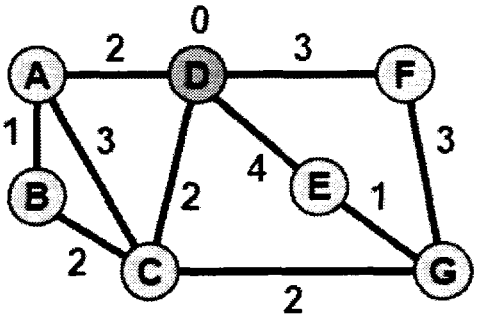


FIGURA 2.14 ITERACIÓN 0:

.

Iteración 1: La actual ruta mas corta para los nodos adyacentes A, C, E, y F es el enlace directo al nodo D.

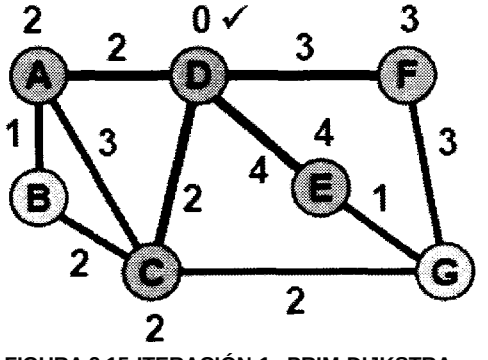


FIGURA 2.15 ITERACIÓN 1: PRIM-DIJKSTRA.

Iteración 2: Los nodos A y C tiene la menor etiqueta con valor de 2. Se toma el nodo C, ya que ofrece la trayectoria mas corta hacia el nodo D para los nodos B yG.

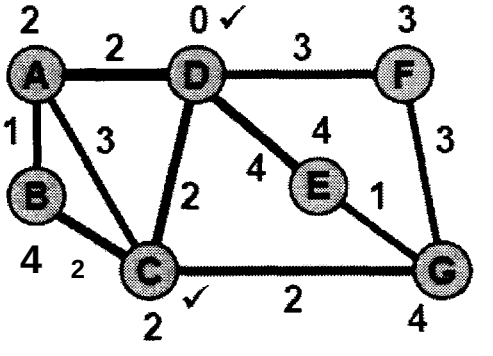


FIGURA 2.16 ITERACIÓN 2: PRIM-DIJKSTRA.

Iteración 3: El nodo A tiene la etiqueta más pequeña (2). Esto provee la ruta mas corta para el nodo B, por lo tanto el predecesor y la etiqueta de B son cambiados.

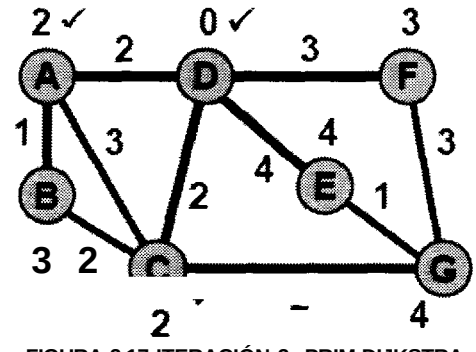


FIGURA 2.17 ITERACIÓN 3: PRIM-DIJKSTRA.

Iteración 4: Los nodos B y F tiene la misma etiqueta (3). Primero revisamos F. Esto nos da como resultado ninguna ruta nueva.

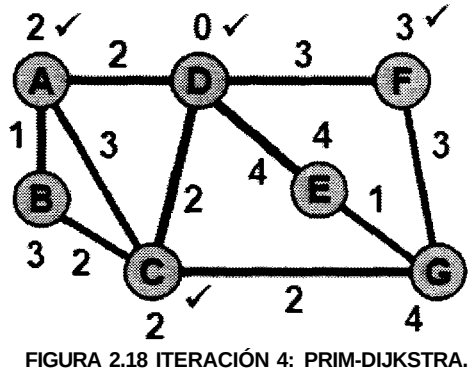


FIGURA 2.18 ITERACIÓN 4: PRIM-DIJKSTRA.

Iteración 5, 6 y 7: Los nodos B, E y G son revisados y no nos dan ninguna ruta más corta. El algoritmo termina cuando todos los nodos han sido revisados.

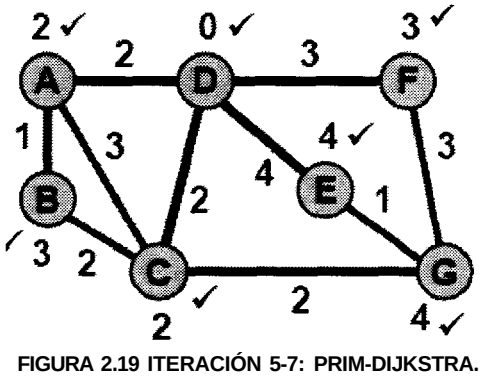


FIGURA 2.19 ITERACIÓN 5-7: PRIM-DIJKSTRA.

El algoritmo terminado ruteado hacia el nodo D, es mostrado a continuación:

La etiqueta de los nodos indican el peso de la trayectoria hacia el nodo D.

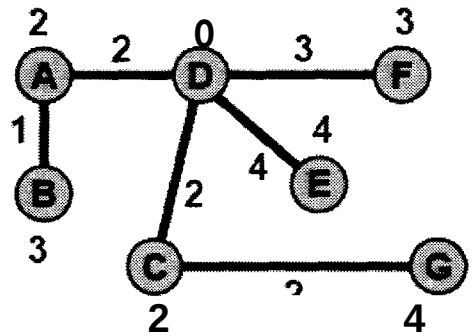


FIGURA 2.20 ITERACIÓN 8: PRIM-DIJKSTRA.

Este algoritmo puede parecer familiar nos menciona Mendoza (2004). Es casi idéntico al algoritmo de MSP de Prim. La única diferencia es que los nodos son etiquetados con una longitud de una ruta completa en vez de la longitud de un solo enlace. Nótese que también trabaja para grafos dirigidos, en donde el algoritmo de Prim solo trabaja con grafos no dirigidos. De manera estructural, sin embargo, los algoritmos son muy similares. La complejidad del algoritmo de Dijkstra, como el de Prim, es de $O(N^2)$ (Kershenbaum, 1993).

De acuerdo con Mendoza (2004), como el algoritmo de Prim, el algoritmo de Dijkstra es más confiable para redes densas. Su mayor defecto es que no toma ventaja de la diversidad de manera correcta y que es sólo apropiado para redes con longitudes de arco positivas (Kershenbaum, 1993).

2.5.4 Algoritmo Prim-Dijkstra.

Ambos algoritmos se encargan de construir árboles. El algoritmo de Prim trata de minimizar el costo de los enlaces al escoger enlaces pequeños. Sin embargo, las longitudes de ruta son demasiado largas para construir una red útil. El algoritmo de Dijkstra produce nodos en donde la mayoría del tráfico transita a través de un nodo central. Esto produce rutas mucho más cortas pero puede producir redes muy caras. Mediante el algoritmo Prim-Dijkstra es posible obtener soluciones intermedias que produzcan árboles en donde los enlaces no sean tan caros como en un árbol SPT (Shortest Path Tree) o las rutas tan largas como en un MST (Mínimum Spanning Tree), según nos dice Cahn (1998).

Nos dice Mendoza (2004) según Cahn (1998) que ambos algoritmos inician dando a cada nodo una etiqueta inicial, y en ciclos buscan entre los nodos para hallar aquél con la etiqueta más pequeña, incorporando al árbol al nodo con la etiqueta más pequeña y finalmente reetiquetando a todos los vecinos.

En ambos casos podemos en forma eficiente obtener al nodo vecino más cercano al reetiquetar los vecinos de cada nuevo nodo que se incorpora en el

árbol. La idea es construir un árbol al iniciar con un nodo e incorporando nodos al árbol al seleccionar aquél que tenga la mejor etiqueta. Para el árbol Prim-Dijkstra la etiqueta es:

$$\min_{neighbors} \; a * dist(root, neighbor) + dis(neighbor, node) \qquad (2.14)$$

$$0 < a < 1 \qquad (2.15)$$

Para dos valores de *a*, el significado en los árboles Prim-Dijkstra es claro: Si *a=0,* entonces se construye un árbol MST. Si *a='\,* entonces construimos un árbol SPT a partir de la raíz. Esto se observa al revisar las etiquetas usadas en el algoritmo de Prim y en el algoritmo de Dijkstra:

Algoritmo	Etiqueta
Prim's	^{min}„rf$_{1i}$A/$_{W}$rv *dist(node, neighbor)*
Dijkstra 's	$'^{m}$TM$_{n\text{ę}eif\text{ł},hho]}$, *(dist(root, neighbor) + dis(neighbor, node))*

Para valores intermedios se construirá un árbol que interpola entre un MST yunSTP(Cahn, 1998).

2.6 Diseño de Redes tipo MESH.

La topología de las redes será la manera en la cual los nodos de la red son interconectados, generalmente existe una asociación jerárquica con los nodos de la red y/o acoplamientos (Sharma, 1990).

Dada la ubicación geográfica de las entidades a comunicar (transmisoras y receptoras de información), así como el tráfico entre ellas, la topología lleva a cabo 3 tareas (Hernández, 1994):

- Decidir en qué lugares debe colocarse los nodos de red
- Decidir cómo conectar los nodos (estrella, árbol, anillo, malla etc.)
- Decidir cuántos enlaces se necesitan entre los nodos.

Según Aarón Kershenbaum (1993) el problema general de optimización de redes de tipo MESH para redes *backbone* es complejo. Involucra la selección de enlaces, la asignación de capacidades de los enlaces, y los requerimientos de *ruteo* en estos enlaces. Idealmente, todos los puntos anteriores son optimizados en común, con el fin de lograr redes de menores costos que cumplan con los objetivos de retraso y *troughput.*

Regresando al problema de determinar la topología para una red de tipo MESH. El objetivo es determinar qué nodos unir de manera directa. Esta decisión

no puede hacerse de manera sencilla y aislada según Aarón Kershenbaum (1993). Este proceso está muy relacionado a la decisión de qué velocidad deben tener los enlaces a usar y cómo hacer el *ruteo* del tráfico a través de la red.

La optimización de la topología de las redes de tipo MESH es claramente un problema complejo. Incluso problemas pequeños como la asignación de *ruteo* y capacidades son difíciles. Técnicas de programación integradora han sido utilizadas para ofrecer u obtener soluciones óptimas o lo más cercano a una solución óptima, pero para un número reducido de nodos (algunas docenas de nodos) (Gavish, 1990). Para problemas de tamaño práctico, la mayoría de las aproximaciones han sido heurísticas.

Las topologías de tipo MESH pueden producirse empezando con una topología, generalmente con una topología factible, y después realizar modificaciones locales agregando o eliminando enlaces. Es posible empezar con el algoritmo de *minimal spanning tree* con capacidad adecuada para soportar la carga completa a un nivel razonable de retraso. Es necesario hacer notar que en general, esta no es la red más económica dado que pueden existir un gran número de canales paralelos en cada uno de los enlaces. Estos enlaces son agregados a la red, buscando reducir el costo mientras se satisfacen los requerimientos de *troughputy* retraso (Kershenbaum, 1993).

De manera alternativa, existe la opción propuesta por Maruyama (1978) de añadir enlaces que minimicen la taza de cambio en el costo por un cambio en el retraso, esto es, procesar para cada enlace la reducción del retraso por el costo del enlace. Esto requiere de un re-enrutamiento del tráfico.

Otra posibilidad es empezar con un grafo completo y de ahí identificar el enlace a eliminar. Otra vez, es posible utilizar algoritmos de minimización de costos.

El análisis de las ramificaciones es muy utilizado. Se puede verificar el máximo número de saltos permitidos utilizando una red que satisfaga esta condición. Además tiene la propiedad de ser un procedimiento incremental. Entonces, puede ser utilizado para refinar la red que ya existe, mediante la adición o eliminación de algunos enlaces para ajustar los cambios en el tráfico en un cierto plazo sin tener que rediseñar la red por completo (Kershenbaum, 1993).

2.7 Teoría de Encolamiento para el Análisis de Retardo.

¿Qué sucede si dos sitios desean utilizar el enlace al mismo tiempo? Dos repuestas básicas: *coordinación* y *encolamiento.* Un fácil ejemplo de coordinación es *Token Ring.* Las LANs *token ring* involucra el movimiento del *token,* el cual otorga el permiso para transitar, dentro del anillo. Solo aquel que tenga el *token*

podrá transmitir. Esto es correcto en LANs donde las distancias son muy cortas y el tiempo de propagación es muy pequeño, pero no es muy bueno si las entidades que deben de ser coordinadas se encuentran a ciento o miles de kilómetros de distancia. Esto es porque el retraso de propagación del *token* para moverse por un anillo de 1000 millas de longitud seria:

$$\frac{1000}{186000} \tag{2.16}$$

o aproximadamente 5mseg. El tiempo que le toma transmitir un paquete de 1000bits a una velocidad de 16Mbps es:

$$\frac{1000}{16{,}000{,}000} \tag{2.17}$$

casi dos ordenes de magnitud menos. En el caso donde las distancias son muy largas, una aproximación más razonable es el agregar capacidad de almacenamiento al sistema y el utilizar una política de almacenamiento-y-envío para el ruteo de paquetes. Esto permite a múltiples paquetes el ocupar la red al mismo tiempo. Si se modela un enlace como un sistema de encolamiento, el objetivo es el comprender el retraso que los paquetes experimentan el pasar por el enlace. Importante es resaltar que el tiempo de servicio para un paquete de *n* bits en un enlace de velocidad *S* bps es *yí*, (Cahn, 1998).

La distribución del los tiempos de servicio para paquetes es una de las dos cosas que determinan el retraso a través del enlace. El otro aspecto es el proceso de arribo. El proceso de arribo es descrito por el espacio que existe entre el arribo de paquetes. Por lo tanto, el modelo depende de dos distribuciones de probabilidad: $p_{k,,grl,}(x)$(o *p(x)de* manera mas corta, el cual describe la distribución de de la longitud de lo paquetes, y $p_{imi,l\cdot an\cdot ivil,}(x)$ (o *g(x)óe* manera mas corta, que describe el proceso de inter-arribo. Si la probabilidad de distribución esta dada por la función:

$$_P(x) = ce^{\sim cx} \tag{2.18}$$

entonces es llamada una distribución exponencial. Si ambos procesos, *py q,* están distribuidos de manera exponencial, el sistema resultante de encolamiento es denotado como un encolamiento *MIMl\.*

El análisis de un encolamiento *MIM l* es particularmente simple y esta evocado a la derivación de bloqueo en el sistema telefónico. La tasa de arribo de paquetes al sistema de encolamiento es denotado como *A*, y la tasa de

procesamiento es denotada como *ju.* La distribución de la longitud de paquetes es entonces

$$p(x) = ¡le^{\wedge x} \tag{2.19}$$

y el espacio entre el arribo de paquetes es

$$q(x) = Áe^{\sim ÁX} \tag{2.20}$$

Debe hacerse notar que dado que los paquetes raramente tiene una longitud que no es un número entero de bits, esto no modela completamente la realidad. Se puede modelar este sistema con un modelo infinito de estados. Los estados son S_o, 5,, 5_, S_3,... Los estados y las transiciones entre los estados se muestran en la figura 2.21. S_o se refiere al estado donde actualmente los paquetes no están siendo servidos o almacenados, y S_n se refiere a un paquete que esta siendo transmitido y *n*-\ paquetes esperando por servicio en el encolamiento. Se asumirá que los paquetes son transmitidos en el orden en el que son recibidos (Cahn, 1998).

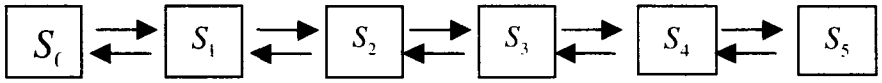


Figura 2.21. El espacio de estados para un encolamiento *M IM I\.*

El retraso del sistema puede ser calculado si se conocen las probabilidades, p_n, de estar en el estado S_n. Estas probabilidades estarán definidas mientras el sistema este en un estado estable definido por una región que cumple con *A < /u.* Claramente, si *A >ju*, los paquetes arriban mas rápidamente de lo que pueden ser transmitidos y la longitud del encolamiento tiende a °°.

Cuando el sistema se encuentra en cualquiera de los estados S^1_n, entonces la tasa a la que los paquetes arriban es *A* y la tasa en el cual están siendo servidos es //. La tasa de servicio es independiente del número de paquetes que se encuentran en el encolamiento dado que sólo hay un servidor. Las ecuaciones de estados muestran el flujo a través de de los estados 5„ y 5_{n+1} debe ser cero. Por lo tanto,

$$\lambda S_n = \mu S_{n+1} \tag{2.21}$$

y se define

$$P \equiv \frac{A}{\mu} \tag{2.22}$$

entonces

$$S_{H+i} = pS_n \tag{2.23}$$

Esto demuestra que

$$\begin{aligned} P\backslash &= PPo \\ Pi &= P'Po \end{aligned} \tag{2.24}$$

y en general

$$Pn = P''Po \tag{2-25}$$

Dado que p_n es la distribución de probabilidad, se puede ver que

$$\sum_{i=0} \text{Í}>, \text{A} > \sum_{i=0} \text{Í}>'' = 7^{\wedge} = 1 \tag{2.26}$$

por lo tanto

$$Po = l \sim P \tag{2-27}$$

y

$$p_n = (1-\rho)\rho^n \tag{2.28}$$

Estas probabilidades nos permiten calcular el tiempo de espera y el tiempo total en el sistema. El tiempo de espera esta definido como la cantidad de tiempo que un paquete espera en el encolamiento antes de que llegue a la cabeza de la línea y comience a ser transmitido. El tiempo total es el tiempo de espera más el tiempo de servicio. El tiempo de espera esta definido como

$$T_W = \frac{\rho / \mu}{1 - I7} \tag{2.29}$$

y el tiempo total entre entrar y salir del sistema es

$$T = \frac{1/\mu}{1-\rho} \tag{2.30}$$

Se define como el tiempo promedio del servicio T_s como el recíproco de la tasa de servicio. Esto es $T_s = 1/\mu$. Por lo tanto en ocasiones es de ayuda escribir

$$T = T_S + T_W \tag{2.31}$$

lo cual expresa el tiempo total como la suma del tiempo de servicio y el tiempo de espera.

Ya con los fundamentos claros propuestos en este capítulo, se tiene de manera más completa lo que se pretende hacer en esta tesis. Los modelos de generación de tráfico buscando obtener una relación entre la distancia, costo y el tráfico entre cada par de nodos; así como la relación a considerar entre la utilización del un enlace y el retraso del mismo.

Como parte primordial de esta tesis es la teoría para la creación de redes tipo MESH, la cual veremos en el siguiente capítulo con el algoritmo Mentor II.

2.8 Confiabilidad en la red.

Dada una red con sus respectivos nodos y links, la confiabilidad de la red esta dada por la probabilidad que los nodos funcionando estén conectados por links funcionando.

En esta definición estamos asumiendo que una falla parcial en la red todavía es capaz de hacer su trabajo. Pero esto no siempre es factible para todas las redes. Si la falla de la red todavía esta conectada pero puede soportar el 75% del tráfico, dos situaciones pueden suceder. Si la red tiene una prioridad de estructura, entonces el 25% del tráfico será bloqueado y el resto estará dando un servicio relativamente normal. Si no hay prioridad en el sistema, entonces en promedio 1 de cada 4 paquetes de transmisión será descartado y la red se volverá inservible para la mayoría de los usuarios. (Cahn, 1998).

Por lo regular, estamos interesados en diseñar redes que puedan soportar una falla en un link en el backbone y que no se desconecte. Por esta razón estamos encaminados a mirar el problema de diseñar redes confiables.

Capítulo 3

MENTORII.

Contando con las bases que se han visto en el capítulo anterior podemos entrar de lleno y con mayor referencia del algoritmo para el diseño de redes MENTOR *(MEsh Network Topology Optimization and Routing),* y así como los fundamentos y características mas importantes de este algoritmo.

El algoritmo MENTOR nos menciona Mendoza (2004) es el arquetipo para alta-calidad, baja-complejidad del algoritmo en el diseño del "backbone". MENTOR se mantiene en el ruteo y optimización topológica de redes. Divide los sitios en sitios de "backbone" y sitios finales; los sitios de "backbone" son pretendidos a ser puntos de acumulación para el tráfico y puntos en el ruteo. Este es un alto procedimiento de parametrizado que puede producir diferentes "backbone". Se continúa escogiendo sitios hasta cubrir su totalidad. El algoritmo procede a seleccionar el centro de cada esqueleto. Se selecciona el sitio del esqueleto con el momento más pequeño a ser medio en la red. Usa el árbol para determinar la secuencia de los pares. Por ultimo el algoritmo considera cada par solo una vez; se agrega un enlace que llevará el suficiente tráfico para justificar el mismo (Cahn, 1998).

El algoritmo MENTOR al inicializar uno de sus parámetros a un cierto valor, considera todos los nodos como conmutadores y genera la red del "backbone". El algoritmo calcula una mediana entre todos los conmutadores, posteriormente crea un árbol con características de bajo costo de conexión, y con caminos cortos a la mediana. El siguiente paso es adicionar enlaces directos entre conmutadores fuera de los enlaces del esqueleto; se agrega la cantidad de tráfico que justifique el colocar el enlace (Hernández, 1994).

Nos menciona Mendoza (2004) que la baja complejidad es lograda debido a que realiza ruteo implícito dentro de un procedimiento que evalúa enlaces. Existen otros procedimientos para el diseño de redes tipo MESH (malla), y algunos de estos podrían obtener soluciones que por pequeños porcentajes serían mejor que la solución de MENTOR. El algoritmo MENTOR es único, con la habilidad de encontrar topologías adecuadas de manera rápida y de poder realizar diferentes y alternas soluciones en un tiempo reducido, es esto lo que le da la gran ventaja (Kershenbaum, 1993).

Pero hasta ahora solo se ha mencionado lo que es MENTOR y ¿qué pasa con MENTORII? A continuación dentro de este trabajo nos daremos cuenta que los primeros 4 pasos de MENTOR y MENTORII son exactamente los mismos y solo varían con respecto a un algoritmo llamado Incremental Shortest Path (ISP)

que veremos mas adelante; todo esto se irá viendo poco a poco mientras se desarrolla el algoritmo como se presenta a continuación.

3.1 Principios de Diseño.

Este algoritmo es apropiado para el diseño de muchos tipos de redes de comunicación porque no se basa en características particulares de tecnología de redes o arquitectura, sino que se basa en los principios básicos del diseño de redes. Específicamente, trata de encontrar la red con las siguientes características (Kershenbaum, 1991):

1. Los requerimientos de tráfico son ruteados de manera relativamente directa. La noción de que sea casi directa es realizada en término de costos, pero dado que el costo esta relacionado con la distancia, la ruta será dirigida también por la distancia. Es importante que también considera que las rutas sean de un número reducido de enlaces.

2. Los enlaces tienen una utilización razonable. No es deseable que la utilización de un enlace sea tan elevada que el desempeño (pérdida, retraso) se vea afectado, y no es deseable que la utilización del enlace sea muy pequeña dado que no seria en costo.

3. Los enlaces de capacidad relativamente alta son utilizados, lo cual no permite beneficiarnos de la economía de escala generalmente presente en la relación entre la capacidad y el costo.

Tomando en cuenta los tres aspectos anteriores, la red ideal será aquella en la cual el tráfico es ruteado a través de enlaces directos entre fuente y destino. Estos enlaces serán de la capacidad más alta posible y serán utilizados con la mayor eficiencia.

Estos tres objetivos son, para algunos, contradictorios. Por ejemplo, el deseo de crear enlaces bien utilizados y de alta capacidad generalmente implica el traspaso de tráfico a rutas indirectas. No obstante, el algoritmo MENTORII enfrenta cada uno de los objetivos anteriores uno contra otro para obtener redes efectivas y de bajo costo (Kershenbaum, 1993).

Dentro de este algoritmo se basa en las tablas de trafico, costo, población de cada nodo que obtuvimos en el capitulo anterior, ya que así funciona este algoritmo. Estas tablas o matrices nos facilitan la labor y son la base del buen funcionamiento del algoritmo MENTORII.

3.2 Inicialización de MENTORII

El Algoritmo busca minimizar la cantidad

$$M_i = \sum_j C_{ij} w_j \qquad \textbf{(3.1)}$$

donde c_{ij} es el costo de conectar los nodos *I* y *j*, y w_j es el peso del nodo *j*, definido como el requerimiento total de salida y entrada del nodo *j*, esto es,

$$w_j = \sum_k r_{jk} + r_{kj} \qquad (3.2)$$

Así, la media es el nodo mejor capacitado para integrarle el tráfico, en el sentido en que el tráfico debe de ser desviado lo menos posible para llegar el centro de masa. Este paso es $O(N^2)$ por que solo implica formar *N* sumas de *N* elementos cada una y después buscar el valor mas pequeño de *N* (Kershenbaum, 1993).

3.3 Selección de los nodos backbone de la red.

El segundo paso es dividir los nodos entre los sitios de *backbone* y los sitios terminales. Los sitios *backbone* se tienen pensados para manejar y *rutear* el trafico de la red. Este es un proceso con muchos parámetros que puede producir muchas y diferentes redes de *backbone.* Existen varios algoritmos que hacen esto; pero el método original, conocido como *threshold clustering,* utiliza un peso limite WPARM para determinar el nodo *backbone* y un radio de costo alrededor del sitio *backbone* RPARM (Cahn, 1998). El peso normalizado es:

$$NW(N_i) = \frac{3}{C} - \qquad \textbf{(34)}$$

Donde C representa la capacidad del enlace ha ser utilizado. Los sitios que tengan suficiente tráfico, por ejemplo:

$$NW(N_i) > WPARM \qquad (3.5)$$

se enlistan como sitios de *backbone.* Todos aquellos nodos que no cumplan con el peso límite y se encuentren cerca de los sitios de *backbone* serán nodos terminales. A continuación Z estando cerca de un nodo de *backbone* significa que

$$\frac{Cost[Z][N]}{MAXCOST} < RPARM \qquad (3.6)$$

donde

$$MAXCOST = Maxfost\ [N_i\ ]\ [\ Nj\ ] \qquad (3.7)$$

Si seleccionamos todos los nodos que sobrepasen el límite de peso como sitios *backbone* y si aun hay nodos más lejanos de cualquier sitio *backbone* que

$$RPARM * MAXCOST \qquad (3.8)$$

se continua seleccionando nodos hasta que todos lo nodos sean cubiertos. Para hacer esto se asigna a cada nodo un mérito. El mérito es calculado de la siguiente manera. Cada nodo es asignado con coordenadas (x,y) en un plano cartesiano de dos dimensiones (Mendoza, 2004). Si se encuentra en Norte América, se pueden utilizar las coordenadas V&H. En base al mundo entero, se utilizan proyecciones para el mapeo del mundo, lejos de los polos, en un rectángulo (Cahn, 1998). Una vez que se tiene este sistema se pueden procesar el punto

$$(xctr, yctr) \qquad (3.9)$$

definido por

$$xcrt = \frac{\sum_{n \in N} x_n * Weight_n}{\sum_{/_i e/V} Weight_n} \qquad (3.10)$$

$$y_{Crt=} \frac{\sum_{n \in N} y_n * Weight_n}{\sum_{neN} Weight_n} \qquad (3\,1\,1)$$

Estas coordenadas no requieren estar en alguno de los nodos. Verdaderamente, si la mitad de los nodos están en Norte América y la otra mitad en Asia, *(xctr, yctr)* es muy probable que se encuentre en medio del océano. De

todas formas, este sitio es de mucha utilidad para medir que tal lejos está los nodos de centro de la red. Se define

$$dist\_crt_n = j(x_n - xcrt)^2 + (y_n - ycrtf \quad (3\text{-}12)$$

Y

$$\max\_ dist\_crt = \max_{n \in N\ y} j(x_{ii} - xcrt)^2 + (y_n - y\ crt)^2 \quad (3.13)$$

De manera similar se define

$$\text{wmax} = \max_{n \in N} Weight_n \quad (3.14)$$

Por ultimo se define

$$mérito <_n = \frac{1}{2}\left(\frac{\max\_ dist\_ crt - dist\_ crt}{\max\_ dist\_ crt}\right) J, + \frac{1}{2}\left(\frac{Weight}{Weight\_ \max}\right) \quad (3.15)$$

Esta función de mérito da un valor igual a la proximidad del nodo al centro y a su peso. De entre todos los sitios que no exceden el límite de peso WPARM y que no están dentro del radio RPARM de algún sitio *backbone,* se selecciona el nodo con mayor mérito y se enlista como *backbone* y se procede de la misma manera. Esto se realiza hasta que todos los sitios sean *backbone* o estén dentro del radio RPARM del algún sitio *backbone* (Cahn, 1998). Como se ilustra en la figura 3.1.

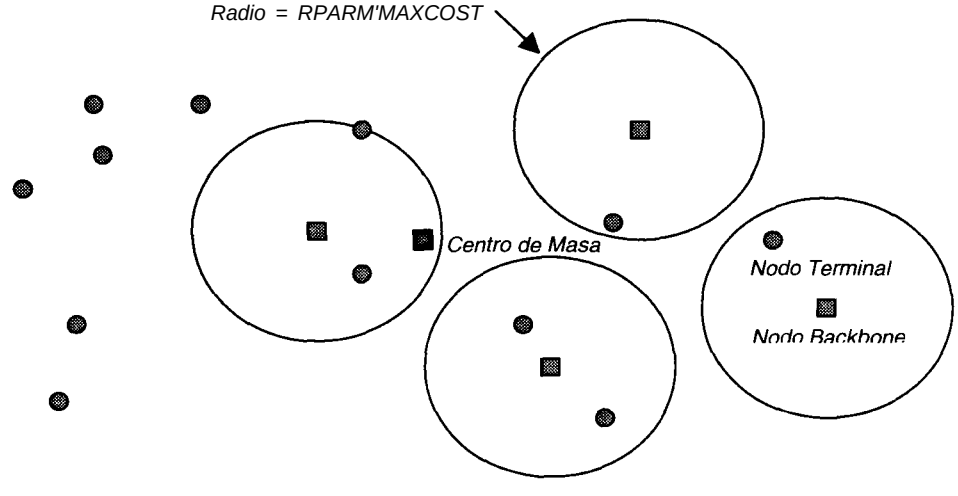


FIGURA 3.1 MÉTODO DE AGRUPACIÓN DEL ALGORITMO.

3.4 Selección de la Media.

Como segundo paso se selecciona el sitio de *backbone* con el menor momento para ser la media de la red. Como se muestra en la figura 3.2, ese será el nodo de *backbone* más cercano al centro de la masa de la red *(xctr,yctr)* (Cahn, 1998). El *momento* de un nodo *backbone* es definido como

$$Momento(n) = \bigwedge_{n'eN} dist(n,n')*Weight_n \quad (3.16)$$

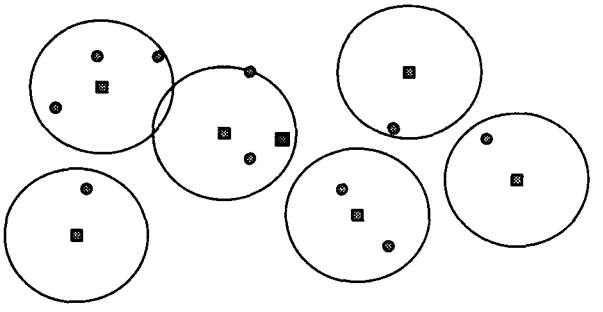


FIGURA 3.2 AGRUPACIÓN FINAL DE LOS BACKBONE CON SUS NODOS TERMINALES.

3.5 Construcción del árbol.

El siguiente paso es crear un árbol que interconecte los nodos. El árbol debe tener las siguientes características:

1. Realizar un ST *(spanning tree)* en los nodos de *backbone.* Cada nodo local es conectado directamente al nodo *backbone* más cercano (en términos de costo).

2. El árbol tiene enlaces cortos (como pude ser MST).

3. Las rutas en el árbol son cortas. En este sentido, es como un SPT (shortest path tree).

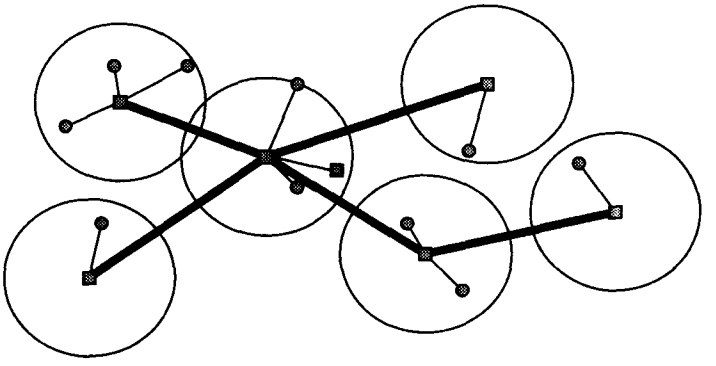


FIGURA 3.3 AGRUPACIÓN FINAL, YA INTERCONECTADOS TODOS LOS NODOS.

Si es posible encontrar un árbol con estas características, entonces existe la posibilidad de crear una red que satisfaga los tres objetivos principales ya antes mencionados. El árbol conglomera tráfico, permitiendo obtener un beneficio de la economía de escala. Cuenta con algunos enlaces directos. Es posible aproximar la capacidad del los enlaces para que sean utilizados de manera razonable, no muy alto ni muy bajo. Finalmente si los enlaces son cortos, y poco utilizados no se requerirá de una gran inversión (Kershenbaum, 1993).

El problema general de encontrar el árbol (Mendoza, 2004) que minimice la suma de las rutas más cortas entre los pares de nodos es en si mismo un problema difícil. Finalmente, este árbol no es la red deseada, pero parecida, es simplemente un punto de partida de la red. El objetivo es minimizar una combinación de la longitud del árbol (suma de las longitudes de los enlaces en el árbol) y la longitud de la ruta hasta tener una media (Kershenbaum, 1993).

Este problema puede ser aproximado con una simulación heurística nos comenta Mendoza (2004), que puede ser una modificación de los algoritmos Prim y Dijkstra. Obsérvese que los algoritmos Prim's y Dijkstra's, para MST (minimal spanning tree) y SPT (shortest path tree) respectivamente, son relativamente idénticas. En el algoritmo Prim's, empieza con un nodo designado / (en este caso la media) y procura marcar los demás nodos *j,* con:

$$L_j = d_{ij} \qquad (3\text{-}17)$$

donde d_{ij} es la distancia (o costo) entre los nodos / y *j.* Esto selecciona el árbol con enlaces cortos. En el algoritmo Dijkstra's, los nodos son marcados como:

$$L_j = L_i + d_{ij} \qquad \textbf{(3-18)}$$

Esto selecciona árboles con rutas cortas. Un híbrido conocido como Prim-Dijkstra puede ser creado para marcar nodos con:

$$L_j = aL_i + d_u \tag{3.19}$$

donde

$$0 < a < l \tag{3.20}$$

Por lo tanto

$$L_j = \alpha L_i + d_{ij} \quad \begin{cases} \alpha = 0 \rightarrow PRIM \\ \alpha = 1 \rightarrow Dijktra \end{cases} \tag{3.21}$$

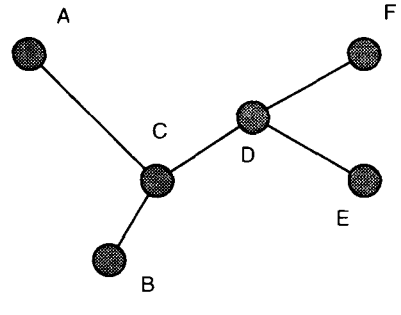
Entre mas grande sea *a,* mas atención se presta a la longitud de la ruta, y de manera opuesta a la longitud del árbol. Cuando el costo del enlace (A, B) no es mayor que la suma de los costos de los enlaces *(A, C)* y *(C, B),* para todo *(A, B y* C), con *a* igual a 1 se obtiene una estrella centrada en la media.

En la practica, la experiencia computacional con este algoritmo ha mostrado que valores de *a* entre 0.2 y 0.3 tienden a genera árboles con enlaces cortos y rutas cortas (Kershenbaum, 1993).

3.6 Secuencia entre pares de nodos.

El siguiente paso es utilizar el árbol para definir la secuencia de todos los pares de nodos. La secuencia no es única pero obedece la restricción de que no se obtiene la secuencia *(N_rN₁)* hasta que se obtiene la secuencia de todos los pares *(N\,N\)* tal que N_t y $N_{,}$ se encuentran en la ruta entre *N* y *N*. Es decir que se ordenaran los pares de nodos que se encuentren más separados por el numero de saltos entre fuente-destino, como ejemplo de una secuencia se muestra la figura 3.

3.



Secuencia	
AE	
AF	
BE	
BF	3hops
CE	
CF	
DA	
P)B	2 hops
AC	
BC	
...	
DF	1 hop

FIGURA 3.4 ÁRBOL CON SECUENCIA BASADA EN SALTOS.

De esta manera se consideran primero todos lo pares de nodos con la ruta mas larga, y así hasta analizar lo pares de nodos que se encuentran con una salto de distancia. En donde existirán

$$\binom{Total\_de\_nodos}{2} \tag{3.22}$$

secuencias, no existe una secuencia única. Dado que existen múltiples secuencias de nodos validos, el algoritmo procede por seleccionar uno y utilizarlo. Experimentos han demostrado que el tratar con diferentes secuencias no produce diseños significativamente diferentes.

3.7 Problemática con Enlaces Directos.

Supongamos que MENTOR agrega enlaces directos del enlace B al F para poder llevar el tráfico desde B a F, G, H e I. (DaSilva, 2002).

- > Ahora el "mínimo ruteo con salto" (mínimum hop routing) usará el enlace para llevar el tráfico de A y C a F, G, H e I.
- > Esto provocará un sobre flujo en el enlace.
- > El algoritmo MENTOR puede asumir que el enrutamiento es absolutamente diferente de la de la red actual.
- > No es factible atar el enrutamiento de la red en las suposiciones hechas por MENTOR.
- > El algoritmo modificado o Mentor II se ocupa de este problema (a expensas de una mayor complejidad).
- ➢

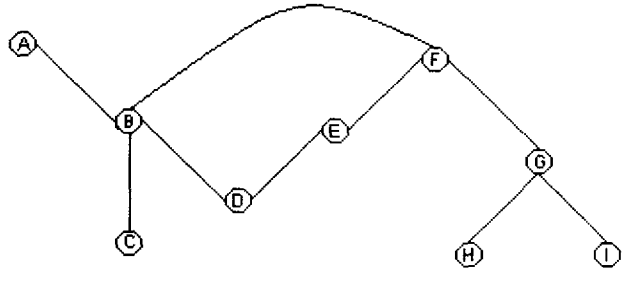


FIGURA 3.5 EJEMPLO DE COMPLEJIDAD CON ENLACES DIRECTOS.

En este trabajo no se habla detalladamente de las etapas de homming y enlaces directos ya que existen trabajos que con anterioridad hablan de manera clara de estos temas como el de Mendoza (2004). Aquí solo se muestra algunos argumentos del porque se hace una mejora al algoritmo, así como la finalidad de ser algoritmo MENTOR a MENTOR II. Aquí es donde se sustituye esta etapa del

MENTOR, y se incluye otro algoritmo llamado *Incremental Shortest Path* (ISP), el cual se verá a continuación.

3.8 Incremental Shortest Path (ISP).

La meta para el algoritmo ISP es identificar todos los pares que puedan usar un enlace en lugar de la ruta actual. (DaSilva, 2002).

* Inicialmente todas las rutas pasan a través del árbol, pero una vez que se añadieron los enlaces directos, el espacio de posibles rutas puede ser más complejo.

    - Supóngase que se mantienen dos matrices de dimensión *n x n.* Las distancias de rutas mínimas entre cada dos nodos se denota por la matriz:
        - (sp_dist) nxn - ruta-mas-corta entre 2 nodos
        - (sp_pred) nxn - Apuntadores a la ruta más corta

Esto denota a una matriz de apuntadores a nodos que mantienen todas las rutas más cortas a través de la red en forma simultánea. Cuando se inicia la ejecución de MENTOR-II, las distancias iniciales son aquellas que pasan por el árbol Prim-Dijkstra. (Cahn, 1998).

El significado de la matriz *sp_dist* es evidente, pero es necesario explicar la matriz *sp_pred.* La matriz *sp_pred[i][j]* contiene el penúltimo nodo (o el predecesor al último) en la ruta más corta que va del nodo "/" al nodo "/• (Cahn, 1998). Considérese el grafo mostrado en la siguiente figura:

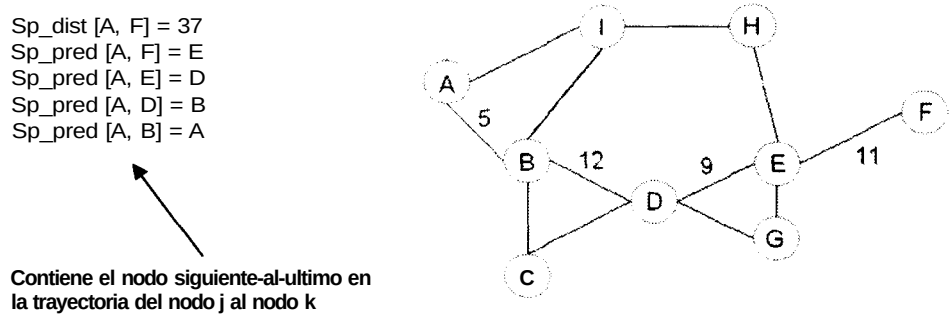


FIGURA 3.6 EJEMPLO PARA HACER LA MATRIZ DE DATOS SP_DIST Y SP.PRED.

Si se está considerando el enlace directo entre el nodo fuente "s" y el nodo destino "d", es importante que no suceda el "robo de tráfico" nos menciona Cahn, (1998). El robo de tráfico ocurre cuando se le da a un enlace una longitud corta de manera artificial. El tráfico fluiría a través del enlace porque éste tendría una longitud corta, pero la nueva ruta podría ser más cara que la ruta original. En

consecuencia, no se considerarán los flujos que se atraen a un enlace al darle a éste una longitud que es menor que su costo.

El algoritmo ISP construye dos listas en base a los siguientes puntos:

Al considerar si agregar un enlace directo entre fuente S y destino D, se construye un sjist y un djist

Un sitio es agregado a *sjist* si *Sp_dist[node,s] + ¡ength < sp_dist[node,d]*
Un sitio es agregado a *djist* si *Sp_dist[node,d] + length < sp_dist[node,s]*

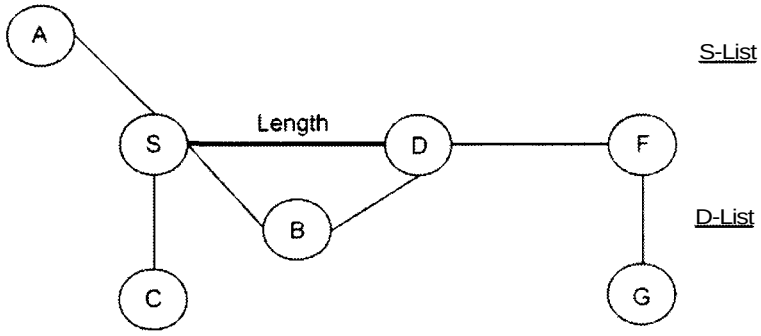


FIGURA 3.7 EJEMPLO PARA HACER LA MATRIZ DE DATOS S_LIST Y D_LIST.

Trabajar con todos los pares n_i en *sjist* y n_k en *djist.*
Si se cumple *Sp_dist[n*$_i$*,s] + length + sp_dist[d, n*] < sp_dist[n*$_i$, nj Entonces el par de nodos *(n*$_i$*, n*$_k$*)* cambiaran su trafico al enlace propuesto si se agrega a la red y se da una longitud de *"length".*

Para cada par *Pm,* computar la longitud máxima para que el tráfico utilice el nuevo enlace: *ReqJen(Pm) = sp_dist[n*$_{it}$ nj - *sp_dist[n*$_i$*,s] - sp_dist[d,* nj

Clasificar los pares por *reqjen* y obtener una secuencia ordenada.

Fijar la longitud del enlace a la longitud apropiada que atraerá el tráfico del par deseado.

3.9 Incorporación de enlaces "1-commodity".

Existe una debilidad en el algoritmo MENTOR-I I menciona Cahn, (1998) cuando el algoritmo de obtención de clusters escoge pocos nodos backbone. Supóngase que se fijan los siguientes valores:

WPARM = 100
y
RPARM=1

En la mayoría de las redes, no hay nodo que tenga 100 enlaces que valgan la pena con respecto a tráfico y tampoco nodo que se convierta en backbone al exceder el límite de peso. Si posteriormente se calcula el mérito, se escoge al primer nodo backbone. Pero como el valor de RPARM es 1, todos los demás nodos serán nodos terminales y el algoritmo termina solamente con un nodo backbone. Así, con un solo nodo backbone, no se habrían añadido enlaces directos, y habríamos obtenido una red estrella. (Cahn, 1998).

La estrella está bien a menos que exista una cantidad considerable de tráfico entre dos nodos terminales. En tal caso es un desperdicio el enrutar el tráfico a través del nodo central. Como consecuencia, se hace una desviación a través del par de nodos (ni y n_2) y se considera añadir una nueva clase de enlaces entre nodos terminales, que se denominan enlaces 1-commodity. A diferencia de los enlaces directos, estos enlaces tienen como propósito llevar sólo el tráfico entre 2 nodos. Estos se colocan en la red sólo si están suficientemente utilizados. (Cahn, 1998). Considérese el ejemplo de la siguiente figura:

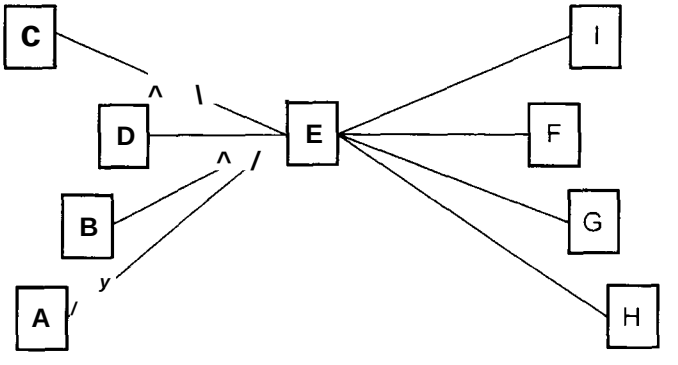


FIGURA 3.8 TOPOLOGÍA ESTRELLA.

En la figura se tiene una red estrella centrada en el nodo E. Supóngase que los enlaces tienen una capacidad de 2 unidades y que se desea cargar a los enlaces a lo más con 1 unidad de tráfico. Se añade un enlace entre cualquier par de nodos que haya acumulado suficiente cantidad de trafico para que un enlace sea utilizado de manera "apropiada", es decir, que utilizara enlaces suficiente para hacerlo costo efectivo.

Existen muchas maneras de hacer esta determinación. La forma más sencilla, y descrita en (Kershenbaum, 1991), es usar el parámetro *slack, s,.* Si la capacidad del enlace es C y la máxima utilización permitida del enlace es *p,* entonces la capacidad utilizable es de *pC.* Si el *slack* es *s*, entonces se añade un enlace *{A,B)* si $r_{AB} > pC(\text{-}s)$.

Entonces si el valor dei parámetro *slack = 0.1,* se podrían añadir 2 enlaces en paralelo entre *A* y *H* para llevar el tráfico directamente sin pasar por el nodo E. Si el valor del parámetro *slack=0.2,* entonces se podrían añadir 3 enlaces entre *H* e /. Si el valor del parámetro *slack=0.3* entonces también se podrían añadir 2 enlaces entre C e /. Todos estos enlaces pueden o no ser una buena idea. Todo depende de la matriz de tráfico menciona Cahn, (1998). De lo anteriormente expuesto, este paso de la etapa de MENTOR-II deba quedar como sigue:

Añadir enlaces directos 1-commodity entre aquellos pares que no se consideraron para la incorporación de enlaces directos si se ve que el tráfico en el enlace excederá $util_{min}$.

Si se ha añadido un enlace 1-commodity *"L",* se requiere darle una longitud que aseguré que llevará solamente el tráfico entre sus nodos extremos (Cahn, 1998). Esto es fácil de hacer con el ruteo de la distancia más corta (shortest-distance routing). Si todos los enlaces de la red tienen una longitud mayor a 1, entonces se fija:

$$Length(l) = sp\_dist\text{^} - 1 \qquad (323)$$

después de la incorporación de los enlaces directos. Esto ocurre en el paso 6 de la 3a. etapa de MENTOR-II, que bajo estas consideraciones diría ahora:

Dar una longitud a los enlaces 1-commodity tal que atraiga solamente al tráfico existente entre los nodos terminales correspondientes.

No siempre es posible decir esto, por lo que hay entonces 3 opciones:

- o No añadir enlaces 1-commodity,
- o Extender la incorporación de enlaces directos considerando no solo los pares backbones sino también a todos los pares de nodos, o
- o Añadir los enlaces y luego checar el ruteo.

Al terminar esta etapa, ahora tenemos que considerar los requerimientos en el backbone. En MENTOR-II se pondrán en secuencia sólo a los pares de nodos del backbone y no a todos los pares de nodos. Supongamos que ei y e2 son nodos terminales. Sea b_i el nodo backbone que es el predecesor a e_i en el árbol Prim-Dijkstra. Si bi = b_2 entonces el tráfico no entrará en el backbone. De otra forma colocamos el tráfico entre ei y e_2 a través de bi y b_2. (Cahn, 1998). Esto nos generará el total de tráfico que pasa a través de cada backbone para poder determinar nuestra utilización de la red.

3.10 Utilización.

Una vez realizados lo pasos anteriores es importante el conocer la utilización porcentual de cada uno de los enlaces que interconectan la red. La siguiente función de acuerdo con Cahn (1998) obtiene un acercamiento adecuado:

$$u = \frac{Traf[N,][N_2\}}{(n * C)} \tag{3.24}$$

donde *n* es le numero de enlaces entre cada par de nodos y C es la capacidad del enlace. Esta función es utilizada para mostrar la utilización de cada uno de los enlaces de cada par de nodos de las diferentes redes creadas. Y un promedio total de utilización de la red.

3.11 Simulaciones.

Como parte final aquí mostraremos los resultados obtenidos en cuanto al desarrollado e implementación del algoritmo Mentor II en el lenguaje de programación C++, basándonos en lo anteriormente mostrado dentro de los capítulos 2 y 3.

Para estos ejemplos se tomo en consideración lo expuesto por Cahn (1998) lo que se refiere a la red de 15 nodos, mostrados a continuación junto con sus coordenadas cartesianas, población (pop) y designación de nodos backbone (bb):

Nodo	BB	Poo	V	tt
1		1	6624	2555
2		1	5975	3690
3		1	7996	2543
4		1	7220	2715
5		1	6564	2394
6		1	5265	3232
7		1	5876	3163
8		1	8109	3692
9		1	7421	4044
10		1	7425	4467
11		1	7602	2480
12		1	6977	2995
13		1	6096	4900
14		1	6643	3268
15		1	6918	4483

Como se muestra en la tabla se tienen 5 nodos backbone determinados; esto es debido a que para un mejor manejo del algoritmo y un diseño mas practico de ejemplificar para este estudio, se dejo fijo el numero de backbone de la red a 5

sin permitir que el propio algoritmo lo decidiera para evitar una red muy grande o muy pequeña y que los tiempos de simulación no fueran tan tardados.

De manera gráfica los nodos de esta red se encuentran distribuidos de la siguiente manera como se ve en la figura 3.9.

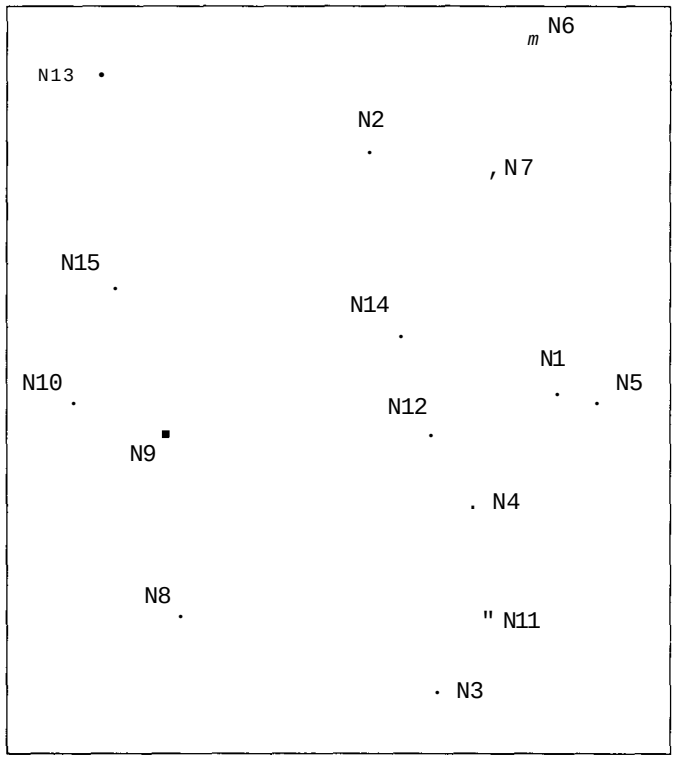


FIGURA 3.9 DISTRIBUCIÓN DE LOS NODOS UTILIZADOS PARA ESTA INVESTIGACIÓN.

Como guía del tipo de línea a manipular para la interconexión de los nodos se muestra en esta tabla:

Link_Type	Fixed_Cost	Dist_Cost	Dist_Cost2	Dist1
T1	1000	24.35	15	3500

Esto es con respecto a la tabla de costos que se tiene con lo estipulado en el capitulo 2 en la formula 2.8.

Dentro de la matriz de tráfico que se usó como lineamiento del buen funcionamiento de este algoritmo, se tomo como parámetro el hecho de que fuera full-duplex, esto quiere decir que cada par de nodos tiene un tráfico de 255000bps en ambas direcciones tanto de $(A, B)_{Traf} = 2550Qhps$ como de $(B,A)_{Tri_{.f}} = 255000bps.$ Esto se presenta en todos los casos.

En esta parte de las simulaciones del algoritmo Mentor II, se dividió en 3 diseños los cuales fueron los más significativos para este estudio, para cada uno de los parámetros importantes a cambiar. Y estos parámetros antes mencionados son los siguientes:

Variable :	Valor
RPARM	0.0
WPARM	0.0
Alpha	0.0
Snack	0.0

A continuación se procede con la primera simulación dentro de la cual se tienen los siguientes parámetros:

Siml	
Variable	Valor
RPARM	0.4
WPARM	1.0
Alpha	0.0
Slack	0.0

Y su respectiva simulación se presenta a continuación:

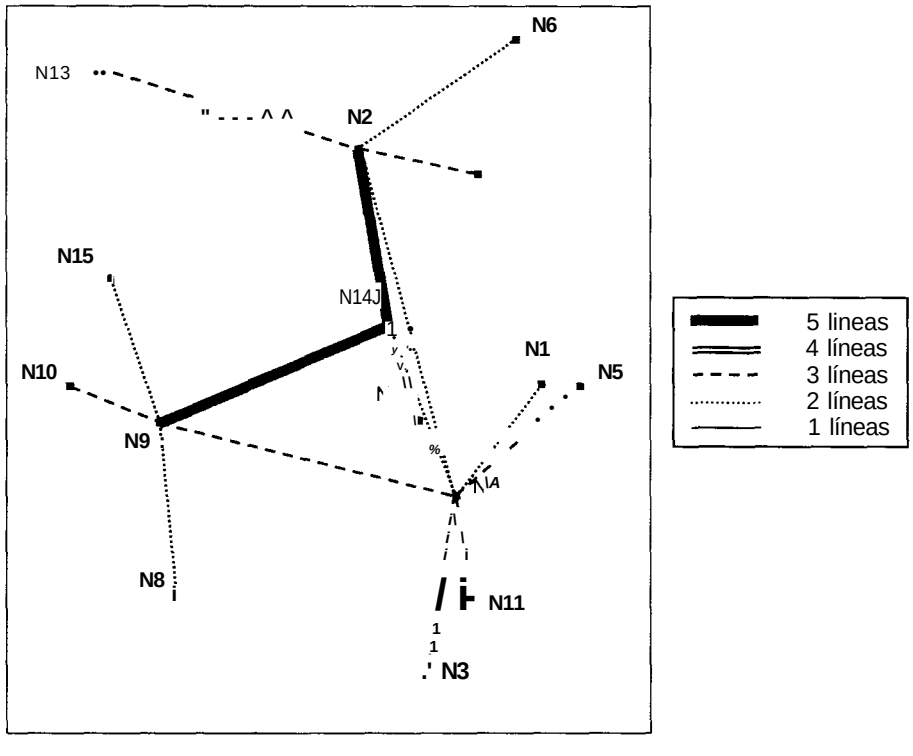


FIGURA 3.10 RESULTADO OBTENIDO DE LA PRIMERA PRUEBA (SIM I).

Esta red que elaboró el algoritmo Mentor II nos proporciona los siguientes datos:

Sim I	
Costo	329490
Útil. Prom. Total	0.7826
No. de líneas	45

Como vemos de la tabla tenemos que la utilización de nuestra red es de un 78.26% y como muestra la figura 3.10 se puede llegar a tener hasta 5 líneas por enlace lo cual nos muestra cual es la trayectoria mas congestionada de la red y se puede concluir que es nuestro esqueleto. Esta primera simulación está elabora en base al modelo de Prim-Dijkstra del algoritmo Mentor II, el cual es aproximadamente la parte intermedia del mismo. Así mismo se presenta también el esquema de la red solo mostrando la utilización por cada uno de los enlaces. Donde los valores en el recuadro pequeño muestra la utilización por enlace.

El esbozo de utilización de Sim I:

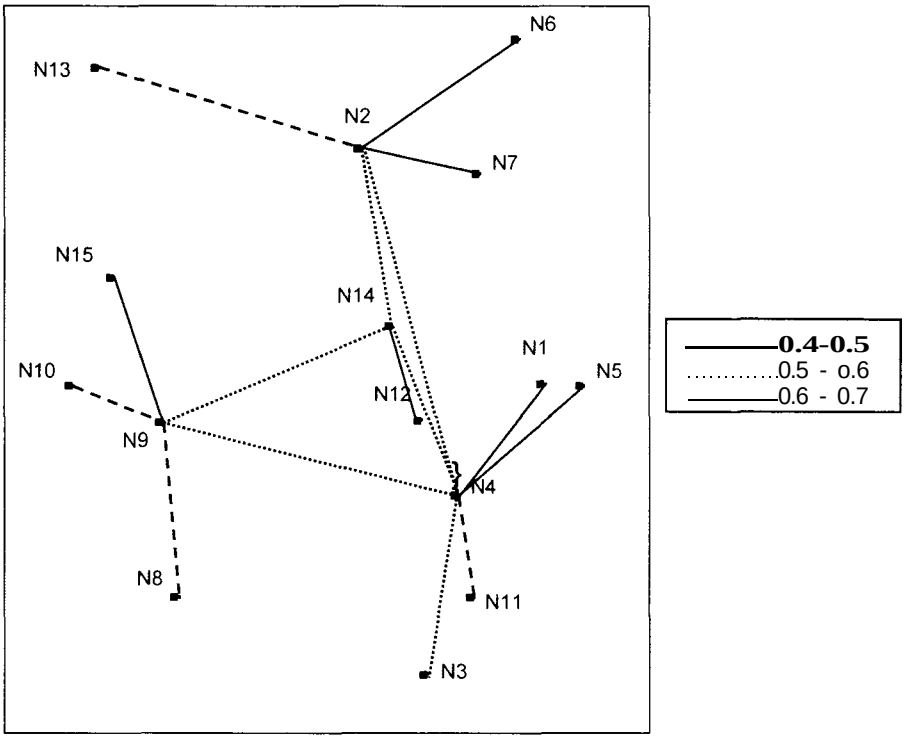


FIGURA 3.11 ESQUEMA DE UTILIZACIÓN DE SIM I.

Se puede observar en la figura anterior que existe un alto grado de utilización en las trayectorias centrales de la red el cual como ya antes se mencionaba representan la base de esta red.

Ahora vamos con la siguiente simulación del Mentor II, y si retomamos lo que dice Cahn (1998) sobre el fuerte impacto que generan el modificar cambios de valores tanto de *a* como de *slack.* Por ende, proseguimos a modificarlos y a darle los siguientes valores:

Sim II	
Variable	Valor
RPARM	0.4
WPARM	1.0
Alpha	0.0
Slack	0.2

Al solo cambiar el slack de 0 a 0.2 esto nos indica un incremento en la utilización mínima *($util_{min}$)* para poder establecer de manera mas clara la utilización de enlaces el cual es importante para Mentor II, pero también a su vez manteniendo el valor *a* = 0.

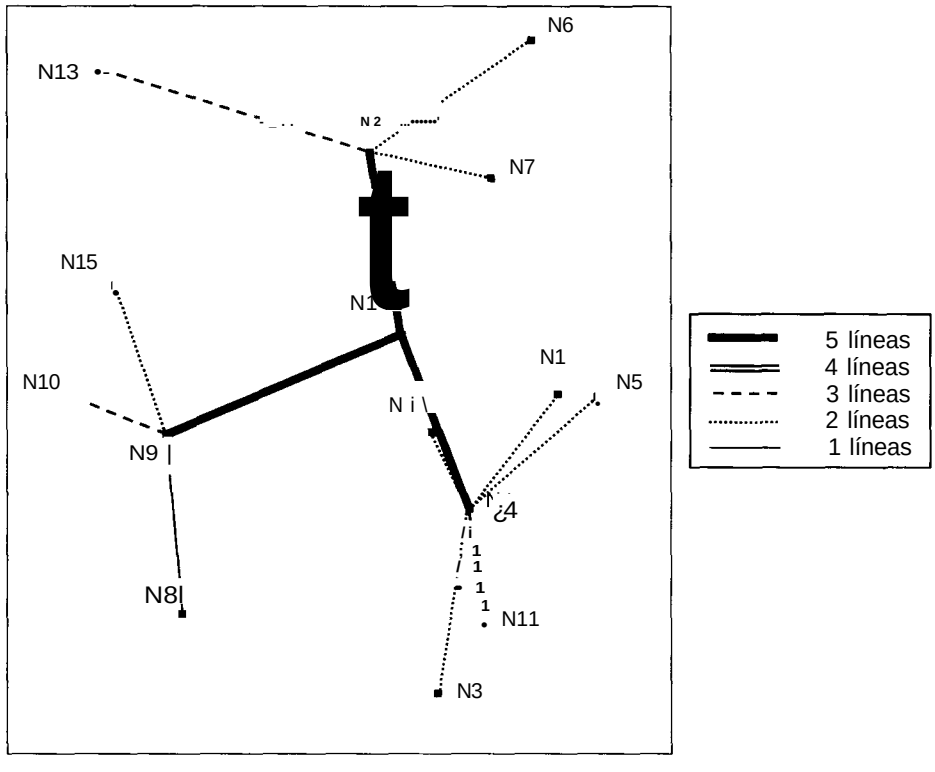


FIGURA 3.12 RESULTADO OBTENIDO DE LA PRUEBA SIM II.

Aquí en esta segunda prueba (figura 3.13) se observa que los cambios que realiza el algoritmo en el diseño sobresalen 2 importantes, dentro de los cuales como principal la reducción de costos total de la red en un 14.02% y a su vez en la disminución de líneas de un total de 45 a solo 39. Arrojando los siguientes datos:

Sim II	
Costo total	283265
Útil. Prom. Total	0.7617
No de líneas	39

También podemos analizar que la utilización casi no varia respecto a la primera prueba esto es un arma de doble filo ya que tanto se reducen costos en el diseño pero a su vez disminuye el tiempo de vida de la red ya que si tenemos en cuenta que en cuanto a recomendaciones sobre diseño de redes se estipula que la utilización debe de andar aproximadamente en un 50% de la capacidad total.

Así como en la primera prueba aquí tenemos la figura de utilización de esta segunda simulación de los enlaces (figura 3.14).

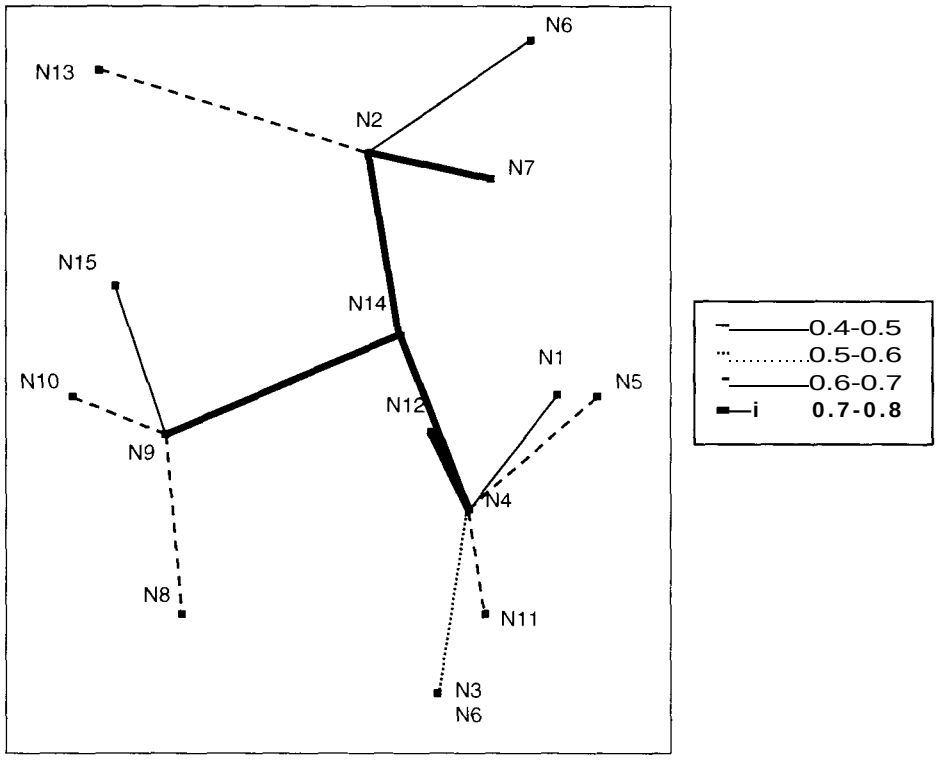


FIGURA 3.13 ESQUEMA DE UTILIZACIÓN DE SIM II CON SU RESPECTIVO CÓDIGO DE UTILIZACIÓN.

Por ultimo se variará el valor de alpha el cual nos menciona Cahn (1998) debe estar entre $0.2<er<0.3$, y es lo que variaremos en nuestra siguiente muestra; se incrementa *a* para la creación de enlaces cortos y directos, lo cual es muy importante ya que es lo que mejor hace este algoritmo de Mentor II. Entonces las variables quedan como sigue:

Sim III	
Variable	Valor
RPARM	0.4
WPARM	1.0
Alpha	0.2
Slack	0.2

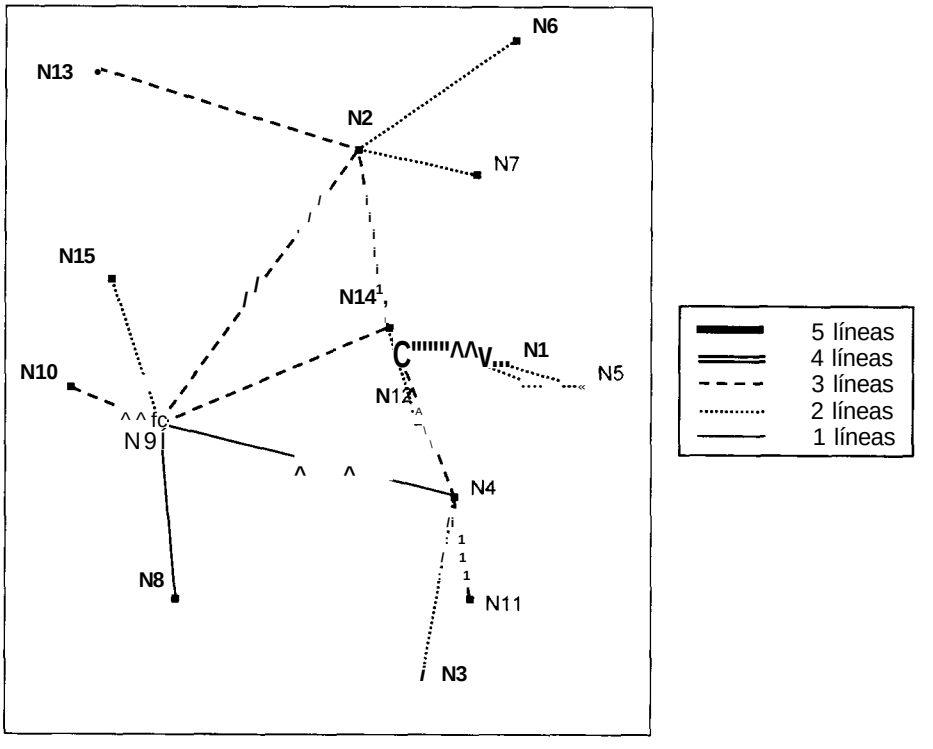


FIGURA 3.14 ÚLTIMA SIMULACIÓN DEL ALGORITMO, SIM III.

Los valores obtenidos de la última simulación fueron los siguientes:

SimtH	
Costo	272650
Útil. Prom. Total	0.7465
No. de líneas	37

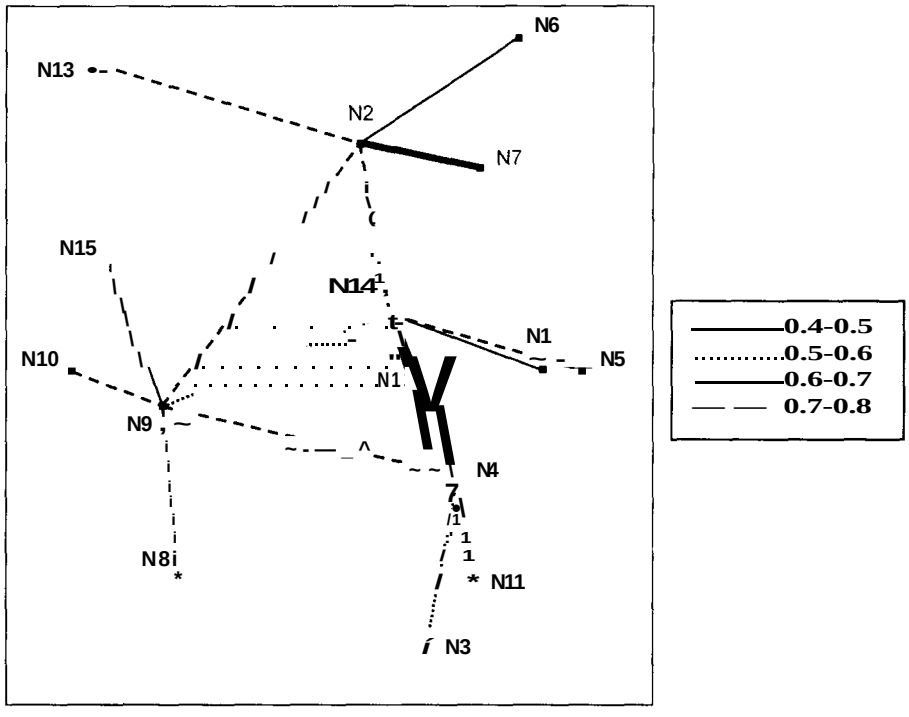


FIGURA 3.15 ESQUEMA DE UTILIZACIÓN DE LA ÚLTIMA PRUEBA CON SU CÓDIGO DE UTILIZACIÓN.

Como aportación final podemos constar de la figura y sus tablas la clara reducción en el costo de la red (figura 3.15) en comparación a la primera simulación elaborada en un 17.26%, la cual es una brecha significativa si hablamos de grandes corporativos. También una pequeña reducción en cuanto a utilización en comparación con la segunda prueba mostrada. Siendo este último diseño es más apropiado para esta situación. Y como ultima figura vemos el análisis de la utilización de enlaces de la red en la figura 3.16.

Con esto queda concluida la primera mitad de simulaciones para el algoritmo Mentor II y siguiendo lo señalado por los autores en los que se referencia este trabajo obtenemos una red mas confiable. A continuación entraremos a la otra mitad de esta investigación el cual contempla el desarrollo de simulaciones en base a programación lineal, la cual tendrá la misma topología a analizar como se hizo en esta sección.

Capítulo 4

Programación Lineal

En lo visto con anterioridad se mostró el funcionamiento del algoritmo heurístico MENTORII, con el cual pudimos obtener resultados para una red de 15 nodos. Con esto ya tenemos la primera mitad de este trabajo, y la otra mitad será el desarrollo y presentación del modelo matemático que será utilizado para el diseño de una red, teniendo en cuenta los mismos parámetros (nodos, tráfico, costos, etc.) que hemos venido utilizando en capítulos anteriores con el algoritmo MENTORII. Con la diferencia que en esta ocasión buscamos un resultado mucho más óptimo que los anteriores, esto se buscará mediante la utilización de la programación lineal, que veremos a continuación.

4.1 Optimización Matemática.

Las nuevas técnicas de optimización surgen diariamente, estimulado a menudo por penetraciones fascinadoras de otros campos. Algoritmos genéticos, por ejemplo, utiliza una analogía a la codificación del cromosoma y la selección natural "desarrolla" buenas soluciones de optimización. Las técnicas de la optimización desempeñan un papel en las redes neuronales artificiales usadas en la investigación de inteligencia artificial para el reconocimiento de patrón. La investigación de "Inteligencia del enjambre" usa agentes independientes del software para solucionar colectivamente varios problemas de optimización. (Dantzig, 2003).

Hoy en día, los métodos de la optimización se utilizan por todas partes en negocios, industrias, gobierno e informática. Un borde en la maximización de beneficios o la reducción al mínimo de costos puede significar a menudo la diferencia entre el éxito y la falla dentro del negocio. (Dantzig, 2003).

El proceso de la optimización se muestra esquemáticamente en la figura 4.1. Típicamente se comienza con un problema verdadero, lleno de detalles y de complejidades, algunos relevantes y otros no. De esto se extrae los elementos esenciales para crear un modelo, y se elije una técnica del algoritmo o de la solución para aplicarse a esta. En problemas prácticos, la computadora realizará los cálculos necesarios. Una parte importante de optimización aplicada acertada es habilidad en la determinación de cuál es y de cuál no es importante durante el proceso de la abstracción. (Chinneck, 2001).

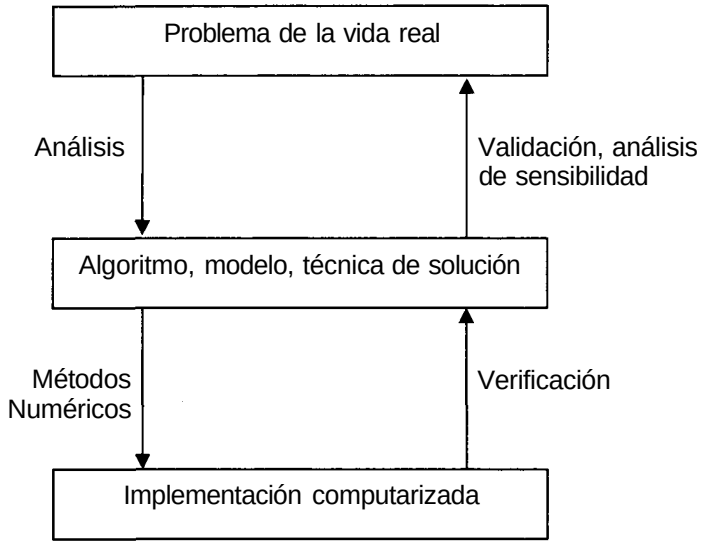


FIGURA 4.1 PROCESO DE OPTIMIZACIÓN.

Las flechas en la figura 1.1 indican el proceso normal del ciclo de optimización. Moviéndose desde el problema verdadero del mundo de algoritmos, al modelo, o la técnica de la solución que se le conoce como análisis. Es aquí donde toma lugar el principal trabajo de abstracción de detalles inaplicables y de centrarse en los elementos importantes. En muchos casos, el proceso del análisis es extremadamente valioso por sí mismo, aún sin realizar ninguna optimización subsiguiente. (Chinneck, 2001).

A partir de esto se extraen los elementos esenciales para crear un modelo, y seleccionar un algoritmo o técnica de solución a ser aplicada para dicho problema. En problemas prácticos, será una computadora quien se encargue de los cálculos necesarios (Mendoza, 2004).

La programación linear es el más aplicado extensamente posible de todos los métodos de optimización. La técnica se ha utilizado para optimizar diversas aplicaciones, incluyendo refinerías y plantas químicas, el mezclado de alimento de ganado, enrutamiento de aviones y hasta como programar sus equipos. Muchos de los problemas industriales de la asignación y de transporte se pueden optimizar con este método. El uso de la programación linear ha sido acertado, particularmente en casos de seleccionar el mejor sistema de variables cuando existen una gran cantidad de opciones correlacionadas. Un ejemplo típico es una refinería de petróleo donde los caudales de la corriente son muy grandes, y una mejora pequeña por la unidad del producto es multiplicada por un número muy grande para obtener un aumento significativo en el beneficio para la refinería. (Dantzig, 2003).

Programación Lineal "Programación", en este sentido, no implica programación computacional - es una palabra utilizada como "planear". En programación lineal todos los modelos del mundo real son procesos lineales, por lo tanto se puede pensar en "programación lineal" como "planeación mediante la utilización de modelos lineales" (Chinneck, 2001).

4.2 Programación Lineal.

Para la optimización que se requerirá, debe haber más de una solución disponible. En la figura 4.2, cualquier punto en la función es una solución, y porque la variable es valor-real, hay un número infinito de soluciones. Una cierta clase de proceso de optimización entonces se requiere para elegir la mejor solución entre miles de opciones. Lo que significa como "mejor" depende del problema actual: puede ser que signifique que la solución que proporciona sea la de mayor beneficio, o que eso consume el menor de un cierto recurso limitado, e.g. área en diseño de chip de computadora, o combustible en el diseño de la ruta de entrega. (Chinneck, 2001).

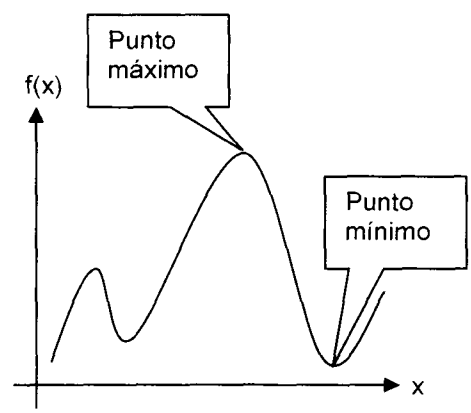


FIGURA 4.2 ESQUEMA DE MÁXIMO Y MÍNIMO DE UNA FUNCIÓN.
Tomado de (Mendoza, 2004).

La programación linear (LP) es la forma mas comúnmente aplicada de *optimización forzada.* La optimización forzada es mucho más dura que la optimización espontánea: ya que se tiene que encontrar el mejor punto de la función, pero también se tiene que respetar varias restricciones mientras que lo realiza. Por ejemplo, se debe garantizar que el punto óptimo no tiene un valor sobre o debajo de un límite especificado cuando sea substituido en una función dada de restricción. Las restricciones se relacionan generalmente con los recursos limitados. (Chinneck, 2001).

Los principales elementos de la optimización forzada son:

- **Variables** (también conocidas como *variables de decisión).* Los valores de las variables no son conocidos cuando se inicia el problema. Las variables usualmente representan cosas que se pueden ajustar o controla. La meta es encontrar los valores de las variables que suministren el mejor valor de la función objetivo

- **Función Objetivo.** Esta es la expresión matemática que combina las variables para expresar la meta. Se requerirá que la función objetivo se maximice o minimice.

- **Restricciones.** Estas son expresiones matemáticas que combinan las variables para expresar límites en las posibles soluciones y así reducir el espacio solución del problema.

- **Fronteras de variables.** Son muy extrañas las variables en un problema de optimización que pueden tomar cualquier valor desde menos infinito hasta más infinito. Por el contrario, las variables usualmente tienen límites (Mendoza, 2004).

En PL, todas las expresiones matemáticas de la función objetivo y las restricciones son lineales.

4.3 AMPL (Un Lenguaje de Programación Matemática).

AMPL por sus siglas en ingles A Mathematical Programming Language es un lenguaje que demuestra a CPLEX como ejecutar un procedimiento específico llamado scripting y a su vez permite al usuario expresar programas matemáticos de manera intuitiva. El AMPL puede y traduce esos programas, solo si son programas lineales o programas cuadráticos, hacia un formato que puede ser leído por un número bastante amplio de solucionadores o "solver" por así decirlo y dentro de estos se encuentra el programa CPLEX y el LPSOLVE. (Gentry, 2003)

AMPL nos da una manera de expresar la representación algebraica del modelo y de éstos valores para los parámetros de manera separada. Hace esto usando dos archivos independientes, un archivo modelo y un fichero de datos. El AMPL lee el modelo del archivo .MOD, los datos del archivo con extensión .dat y los pone juntos en un formato que el programa solucionador entienda. Entonces, entrega este caso de problema al solver, el cuál alternadamènte, soluciona el caso, y el programa regresa la solución al AMPL (Rajan, 2000). En la figura se muestra

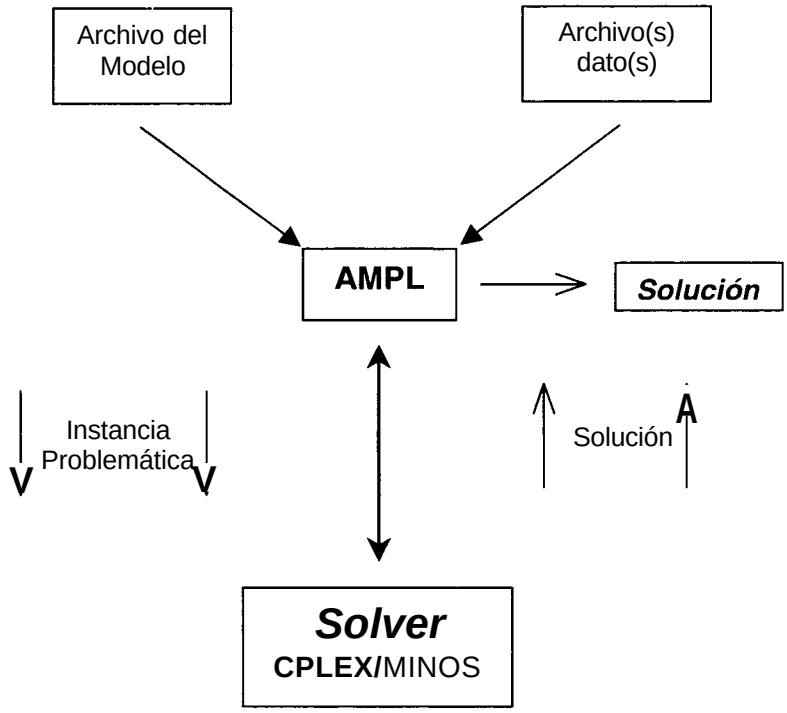


FIGURA 4.3 DIAGRAMA DEL FUNCIONAMIENTO DEL AMPL.

4.4 Modelo utilizado para la optimización de redes.

Así como se desarrollo el algoritmo Mentor II en el lenguaje C++ también se presenta un modelo en el lenguaje AMPL para poder tener los parámetros matemáticos que se necesita para dar solución al problema de diseño de redes, este modelo que se uso se encuentra como anexo de este trabajo. Y aquí mostraremos algunos lineamientos importantes que se siguieron para su desarrollo en base a lo propuesto y lo requerido:

- Los enlaces son pares de nodos sin un orden específico.
- Un arco es un par de nodos con un orden.
- Una ruta esta compuesta de arcos.
- El tráfico puede ser compartido entre diversas rutas.
- Supongamos que tenemos un enlace de 64k entre los puntos (A, *B)* esto accede a un flujo de 64,000bps dentro del arco *(A.B)* y 64,000bps en el arco *(B,A)* concurrentemente.

Con lo propuesto anteriormente podemos detallar las restricciones de nuestro modelo con los parámetros que a continuación se fijan:

Parámetros:

- *N* nodos,
- ***L* enlaces,**
- *E* arcos (los arcos tienen dirección),
- *D* dentro de *{N,N}* pares que demandan tráfico,
- *P* rutas,
- *PD{E)* dentro de *{P},* donde *PD[o,d]* denota el conjunto de rutas que pueden ser utilizadas para rutear la demanda *[o,d]*,
- *PE{E]* dentro de *{P},* donde *PE[e]* manifiesta el acumulado de trayectos que utiliza el arco e,
- *EQUIP* simboliza el tipo de línea a ser manipulada en los arcos,
- *dematid{D]*, *d[o,d]* muestra la demanda del par *[o,d]*,
- *Cost{EQUIP,L), Cost[eq,l]* **indica el costo por usar el tipo de enlace *eq* en el enlace** /,
- *Capacity{EQUIP}, Capacity[eq* **expresa la capacidad del enlace de tipo *eq*.**

4.4.1 Definición de las variables de Salida.

Lo antes propuesto es el fundamento para una mejor red en cuanto a costos. Y como parte complementaria es la determinación de variables a ser manipuladas en el modelo.

$$x\{P\} \geq 0 \qquad (4.1)$$

$$\text{y}\{£\} \geq 0 \qquad (4.2)$$

Esto muestra que la ecuación (4.1) establece que *x[p]* figura el flujo de la ruta *p*. De la misma forma la ecuación (4.2), *y[e]* exhibe el flujo en el arco *e.* (Mendoza, 2004)

Cabe mencionar que debemos tener una variable la cual nos muestre que ha sido situada una línea de conexión y a su vez en que arco, y para esto es la ecuación (4.3):

$$z\{EQUIP.E]\ binan\text{-} \tag{4.3}$$

en la cual c[^,e$^{>}$] = 1 solo si el tipo de enlace *eq* fue colocado en el arco *e;* de otra manera *z[eq,e] = 0.*

Igual que en la ecuación anterior, se necesita de una variable que nos muestra los mismos datos pero ahora para los enlaces *L*:

$$w\{EQUIP,L\}\ binan\text{-} \tag{4.4}$$

Mientras que si *w[eq,e]* = 1 quiere decir que el tipo de enlace fue colocado en el enlace /; sino *w[eq,l] = 0.*

Como ya tenemos declaradas nuestras variables de solución así como sus condiciones, solo nos falta determinar las limitantes de las mismas para poder consumar esta etapa de optimización de la red para esta investigación. Dado que:

- **Todas las peticiones** *{o,d)*

$$\sum_{p \in PD_{o,d}} x_p = demand_{od} \tag{4.5}$$

- Los flujos en arcos:

$$\text{.V, = Z } x_P \tag{4.6}$$

pe PE,

- Las capacidades en arcos:

$$>\text{V } \wedge^{eq \in EQUIP} \text{Z } Capacity\wedge * z_{e_i he} \tag{4.7}$$

- Para cada conexión y cantidad de enlaces:

$$\%, \wedge \text{V/} \tag{4-8}$$

- Para cada conexión y cantidad de enlaces:

$$\cdot \quad > - \tag{4.9}$$

Con estas bases ya contempladas para nuestro modelo solución solo nos queda determinar lo que viene siendo nuestra función objetivo el cual será MinCost y se personifica como sigue:

$$Minimizar \quad \pounds \underset{i'i/eEQUIP}{} \; Z^{Cost}{}_{iq}.i \; * \; \text{»«,/} \tag{4.10}$$
$$leL$$

4.4.2 Limitante del modelo.

Al querer realizar un modelo con una delineación basada en lo anterior y buscando obtener mejores resultados se hizo la propuesta de prácticamente obligar al modelo a buscar 6 rutas diferentes (o 6 saltos) para cada par de nodos y todas estas trayectorias tratando de ser el camino mas corto respetando los fundamentos de este trabajo en cuanto a diseño. Por ejemplo para el par de nodos (1,15):

Trayectoria	SPd
1 - 15	616
1 - 14 - 15	618
1 - 12 - 14 - 15	707
1 - 5 - 12 - 14 - 15	813
1 - 5 - 4 - 12 - 14 - 15	930
1 - 5 - 4 - 12 - 9 - 10 - 15	1054

Esto consecutivamente para todos los pares de nodos restantes en la red.

4.4.3 Archivo de datos.

En esta sección se muestra el planteamiento básico del modelo matemático en base a los datos a utilizar en el modelo. Tomando a /; nodos, y a su vez todos interrelacionados entre si y así como el hecho de que no hay trafico ni costo en el nodo (/z,.,/i,.). Los datos que se consideraron para esta parte se muestran en la sección de anexos de este trabajo. A continuación se muestra las cantidades en cuanto a limitantes del diseño con respecto a la cantidad de nodos *n* a utilizar, que para este caso fueron 15 y *h* viene siendo el número máximo de saltos.

Var.	Formula
N	77
L	$\frac{77*(1-77)}{2}$
E	77*(77-1)
D	/?*(77-I)
P	/?*(77*(77-I))
PD	77*(77-1)
PE	77*(77-1)
EQUIP	1 (T1)
Capacity	1 (1540000)
Cost	$\frac{77*(1-/7)}{2}$

4.5 Resultados de Programación Lineal.

En esta etapa de pruebas en base a Programación lineal se usaron los mismos parámetros que se aplicaron a los ejemplos del algoritmo Mentor II manteniendo de esta manera la distribución de nodos, también la distribución de trafico de 255000bps para cada uno de los nodos, y así como también se uso la misma tabla utilizada en el capitulo 2.8 para efectos de costo de la red con una línea T1 mostrándose a continuación

Link Type	Fixed_Cost	Dist_Cost	Dist_Cost2	Dist1
T1	1000	24.35	15	3500

En base a los parámetros establecidos en el apartado anterior de este trabajo se simuló el modelo así como también el archivo vinculado de datos agregado en el anexo A. A continuación se muestran los resultados que se encontraron:

SimCplex 1	
Costo total	336925
Útil Promedio total	0.4793
No de líneas	36

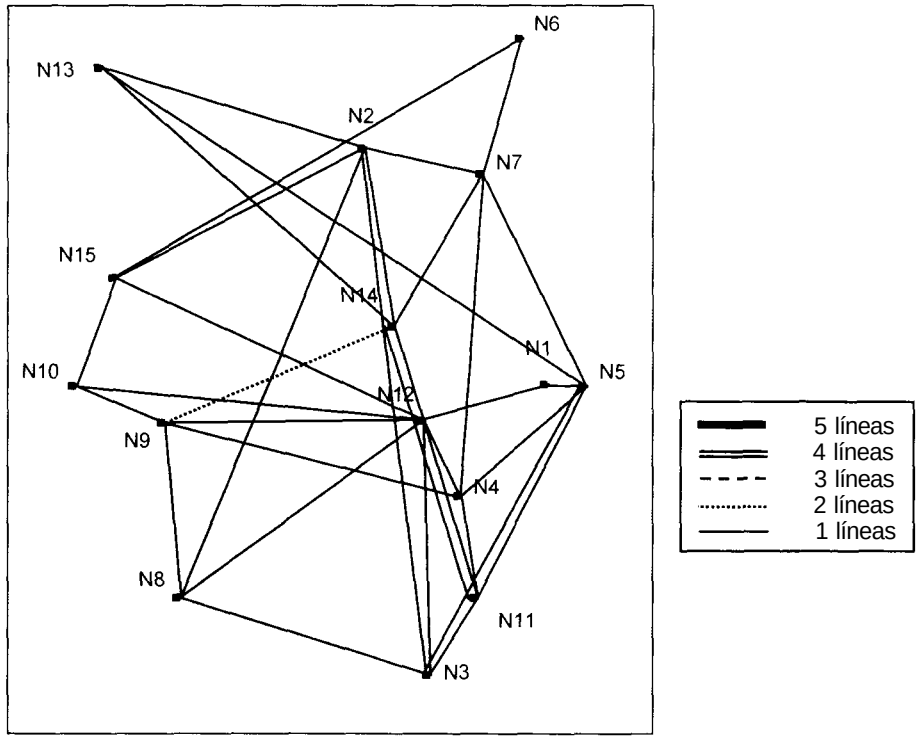


FIGURA 4.4 FIGURA DE LA RED DE SIMULACIÓN DE CPLEX.

Esta figura 4.4 nos enseña una mejora aceptable que presenta la programación lineal en cuanto a numero de líneas utilizadas en la red aunque el costo es relativamente mayor al primer resultado obtenido con Mentor II el cual incrementó en un 2.4% aunque no es mucho pero sigue siendo un alto costo.

En este ejemplo vemos una optimización en cuanto al número de líneas que va de 45 en Mentor II a 36 con PL. Y si continuamos observando la utilización promedio de la red es de 47.93% lo cual esta dentro de lo óptimo el nos indica que la utilización debe estar aproximadamente en un 50%.

Se busca mejorar este modelo presentado con el fin de buscar acercarnos a un diseño que cuente con mejoras en cuanto a costos de la red ya que el anterior no ofrece algo significativo. Esto se hará controlando el número de saltos para la ruta mas corta ya que la distancia de conexión entre los nodos esta directamente relacionada con el costo de conexión.

En seguida vemos el modelo propuesto como SimCplex 2 el cual se encuentra en el anexo B, presentado como sigue:

SimCplex 2	
Costo total	325799
Útil Promedio total	0.4286
No de líneas	41

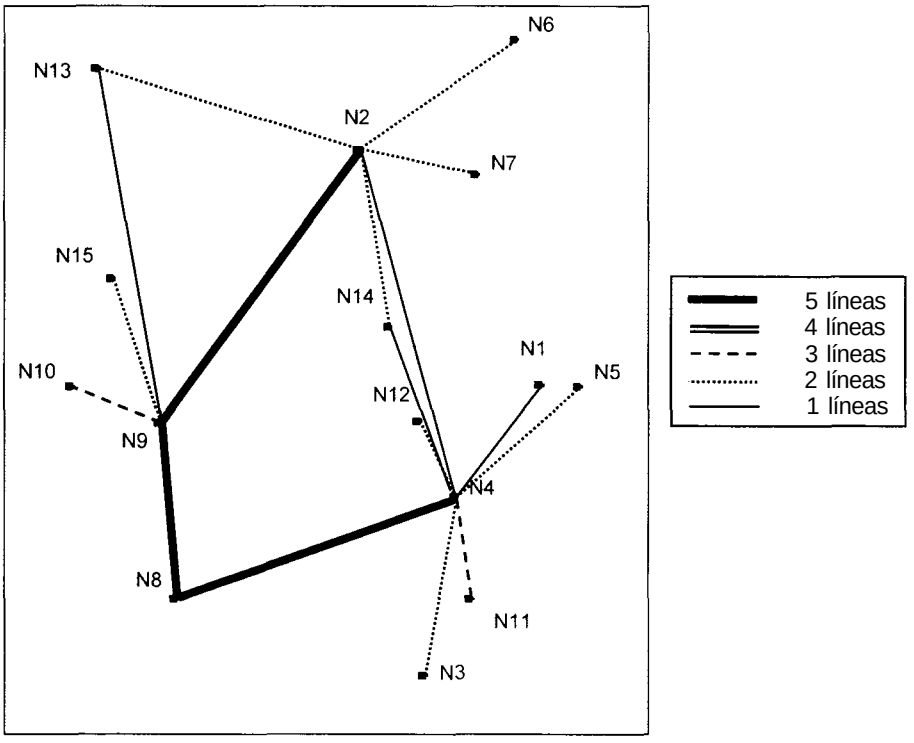


FIGURA 4.5 FIGURA DE LA SEGUNDA PRUEBA DE CONEXIÓN DE LA NUEVA PROPUESTA.

Como vemos ahora si existe una mejoría en el costo total de la red, comparado con la anterior, pero como consecuencia ahora tenemos un incremento en el número de líneas. Y debido a este aumento en las líneas la utilización de la red disminuyo notablemente y esta se encuentra dentro de lo óptimo. Aun y cuando comparemos este modelo con cualquiera de los 3 propuestos por Mentor II sigue mostrando una mayor confiabilidad Cplex en cada una de las líneas a pesar de su costo.

Ahora al querer tener un resultado cercano a lo óptimo trataremos de mejorar nuestro resultado anterior. Lo que se pretende es reducir los costos del modelo anterior. Esto se buscará utilizando las rutas del mejor modelo de mentor II; así mismo aprovechando los enlaces directos que nos den una mejor distribución de los enlaces en la red en base a los que tienen un alto grado de tráfico en el mismo.

Los resultados (Anexo C) de lo planteado se muestran en la siguiente figura:

SimCplex 3	
Costo total	212444
Útil Promedio total	0.79407
No de líneas	26

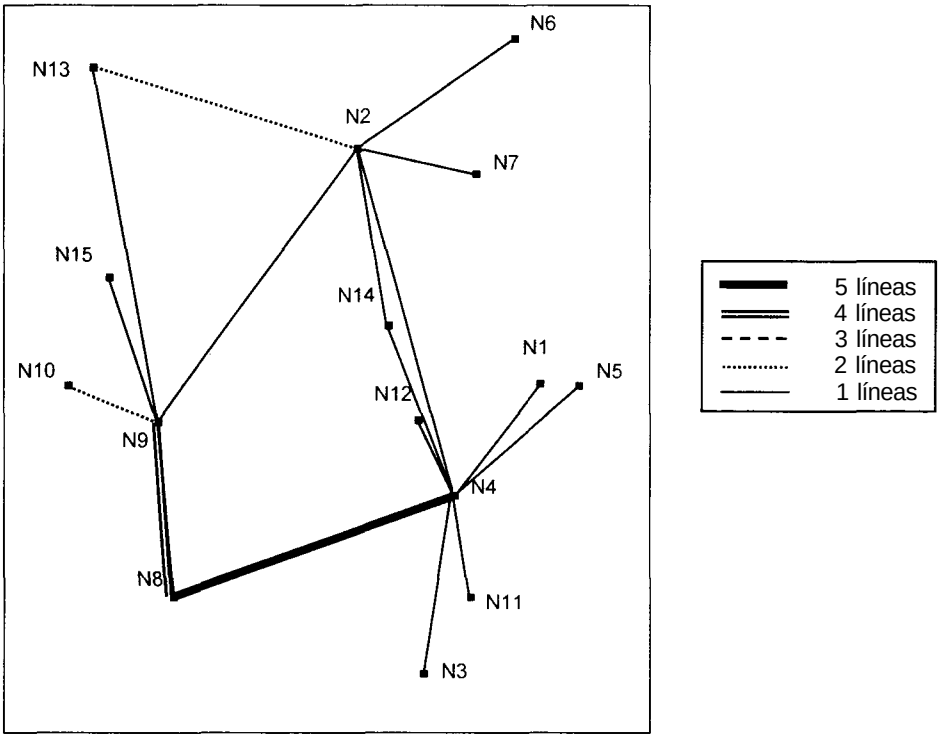


FIGURA 4.6 FIGURA DE MODELO USANDO ENLACES DIRECTOS.

Podemos observar en esta figura que se generan 2 especies de anillo entre los nodos 2, 14, 4,8, 9 y 13; esto se da en estos nodos específicamente debido al alto grado de utilización que presentaban al ser los nodos centrales por así decirlo y la creación de estos anillos distribuye de una mejor manera el trafico entre los nodos. Se puede observar una reducción en el costo total de la red en un 60% aproximadamente y a simple vista parece ser excelente pero esto conlleva a un aumento en a utilización de la red en un 200% prácticamente.

Capítulo 5

Conclusiones

En un principio se estableció la importancia de tener de manera clara y bien definida la planeación y diseño de redes para su optimización esto debido a la constante demanda de servicios de telecomunicaciones que son más globales y abundantes con el paso del tiempo.

Todo esto va generando nuevas y mejores corporaciones de telecomunicaciones, creando ambientes altamente competitivos y por ende las redes deben estar en constante actualización, buscando ofrecer calidad y confiabilidad de transferencia de información para poder complacer este entorno. Y aquí es donde el buen diseño de una red puede dar ese valor agregado a la industria.

Esto nos encamina a conocer más a fondo los diferentes métodos y formas para la creación de redes WAN con la finalidad de hacer mejoras a nivel de shortest path buscando la optimalidad en el diseño.

Tomando lo anterior y lo establecido en este trabajo tanto el algoritmo heurístico Mentor II y la programación lineal (PL) presentan estas características requeridas para el diseño de una red. Lo que prosigue es saber cual es la mejor opción al momento de diseñar la red.

Cabe mencionar que los valores de costos obtenidos en los diferentes diseños de Mentor II no son los más baratos sino que son puntos intermedios de diseño de redes en el cual el árbol del backbone no generará árboles completamente estrella ni árboles completamente Prim para evitar que la confiabilidad de la red fuese bajo.

A continuación se presenta la comparación que se genera en base a Mentor II y programación lineal presentados en esta tesis. Se consideraron los resultados que representan de manera más clara aspectos importantes en el diseño de redes propuestos en el objetivo.

5.1 Conclusiones: MENTOR II Vs. PL.

Los diagramas de las mejores redes generadas por ambos métodos se presentan en la figura 5.1:

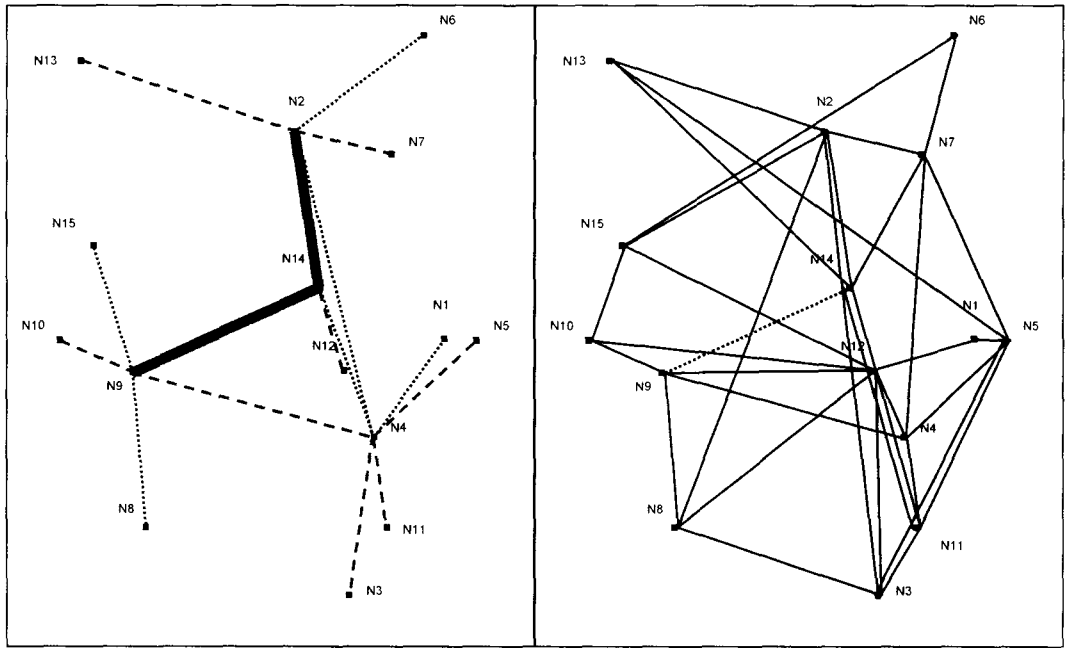


FIGURA 5.1 MENTORII VS. PL.

SimMentorII 1	
Costo	329490
Útil. Prom. Total	0.7826
No. De líneas	45
Confiabilidad	0.979

SimCplex	
Costo total	336925
Útil. Prom. Total	0.4793
No de líneas	36
Confiabilidad	0.999

NOTA: Los valores de confiabilidad obtenidos en los resultados por Delite son valores aproximados, que para propósitos de comparación de diseño cumplen con lo requerido.

Lo que se busca dentro de cualquier diseño es que presente una utilización que este dentro de los estándares propuestos en esta investigación así como un costo que sea relativamente bajo.

Si cotejamos ambos diseños como se ve en la figura 5.1 a primera instancia tenemos que ambos costos son casi los mismos lo cual hasta aquí los 2 diseños son muy similares. Pero si tomamos en cuenta el número de líneas utilizadas vemos una notable mejora la que presenta Cplex; dando en el diseño una reducción aproximada del 20%. Y en cuanto a utilización existe una diferencia bastante aceptable ya que Cplex se muestra alrededor del 50% mientras que

Mentor II nos da un valor elevado de casi un 80% lo cual es un rango de diferencia aastante significativo para el diseño.

Así mismo se obtuvo de cada diseño la confiabilidad que representa cada red llevando los modelos a uno de los mejores programas de diseño de redes, llamado Delite. Esto mediante la modificación de archivos del mismo programa para tener las redes dentro de este software; este cálculo de confiabilidad Delite lo realiza en base a un promedio de perdida de nodos si se quita un enlace en la red. Y como vemos la confiabilidad ofrecida por Cplex es un valor del 99.9% mientras que la red elaborada por Mentor II 97.9% esto nos dice que Cplex presenta una confiabilidad prácticamente del 100% en cuanto a diseño esto nos dice que el propuesto por Mentor II al tener una diferencia de 2.1% con respecto a Cplex nos da un valor de indisponibilidad considerable.

También se obtuvieron valores de retardo de ambos diseños los cuales se presentan a continuación:

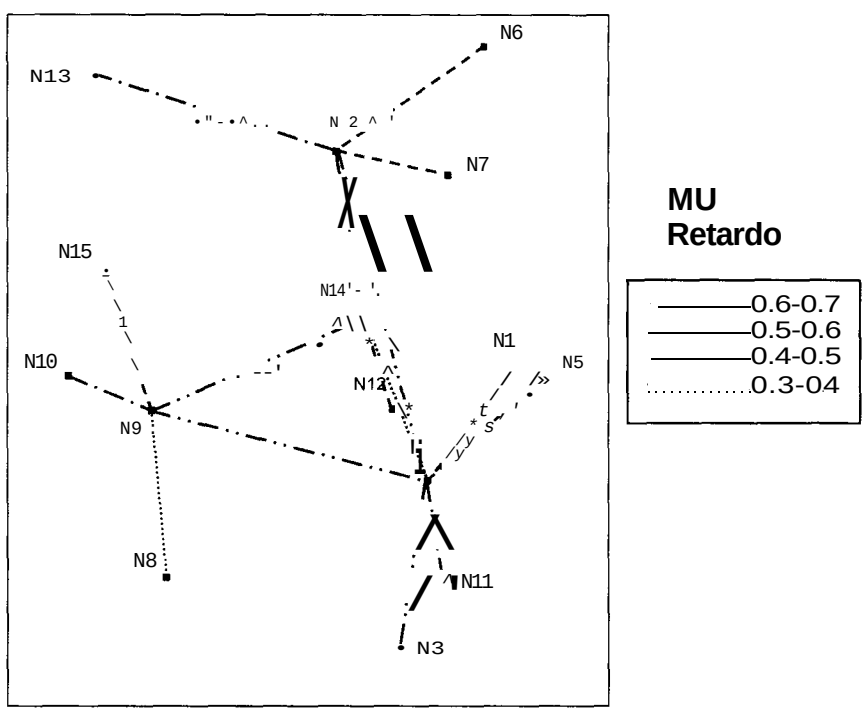


FIGURA 5.2 DIAGRAMA DE RETARDO DE SIM I DE MENTORII.

En esta figura 5.2 se ve claramente que el promedio de retardo para Mentor II es de 0.46 milisegundos mientras que el retardo presentado por el diagrama presentado por Cplex presentado en la figura 5.3 es de 0.34 lo cual representa una mejor solución.

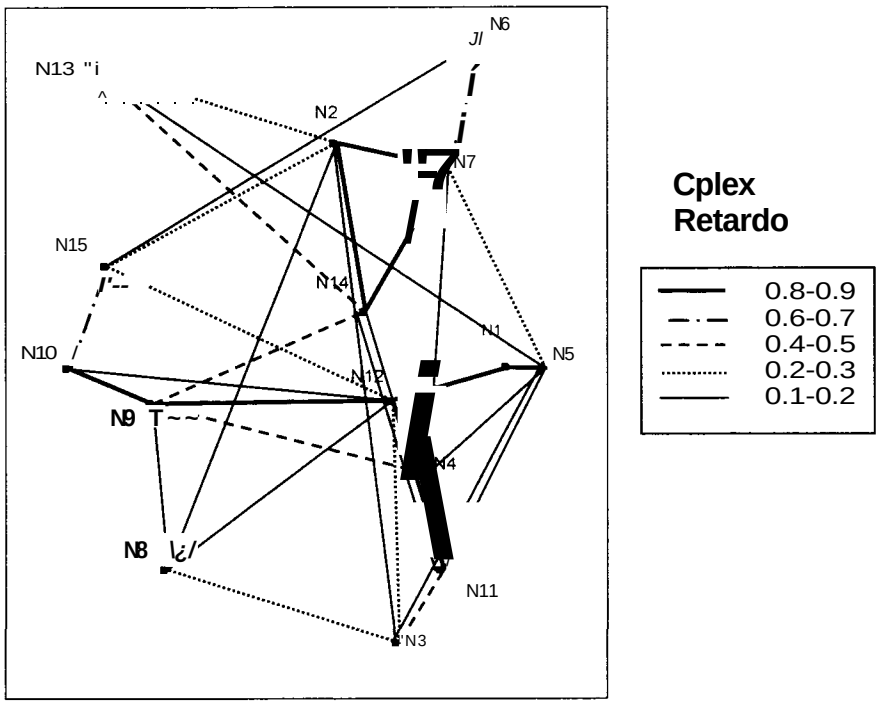


FIGURA 5.3 DIAGRAMA DE RETARDO DE SIM I DE CPLEX.

NOTA: Los cálculos de retardo de las redes son valores aproximados que obtuvimos con ayuda del programa DELITE, pero para efectos de comparación, si existe una diferencia efectiva. Es importante para futuros trabajos tomar esto en consideración.

Si observamos con mayor detenimiento vemos un alto grado de congestionamiento en los nodos 2, 14, 12 y 9 de ambos diseños por lo que debemos tener en cuenta que pueden existir mejores modelos para ambos procedimientos los cuales presenten un menor costo y que tenga mas variantes en cuanto al ruteo, y esto lo podemos ver en la siguiente comparación en donde tenemos los mejores diseños obtenidos por ambos métodos presentados a continuación.

Como se muestra en el diagrama 5.4 en ambos casos se generan 2 anillos entre los nodos con mayor trafico pero a diferencia de Mentor II con PL se crea un tercero el cual permite un mayor flujo de la información sin tanta carga lo cual permite una mejor distribución.

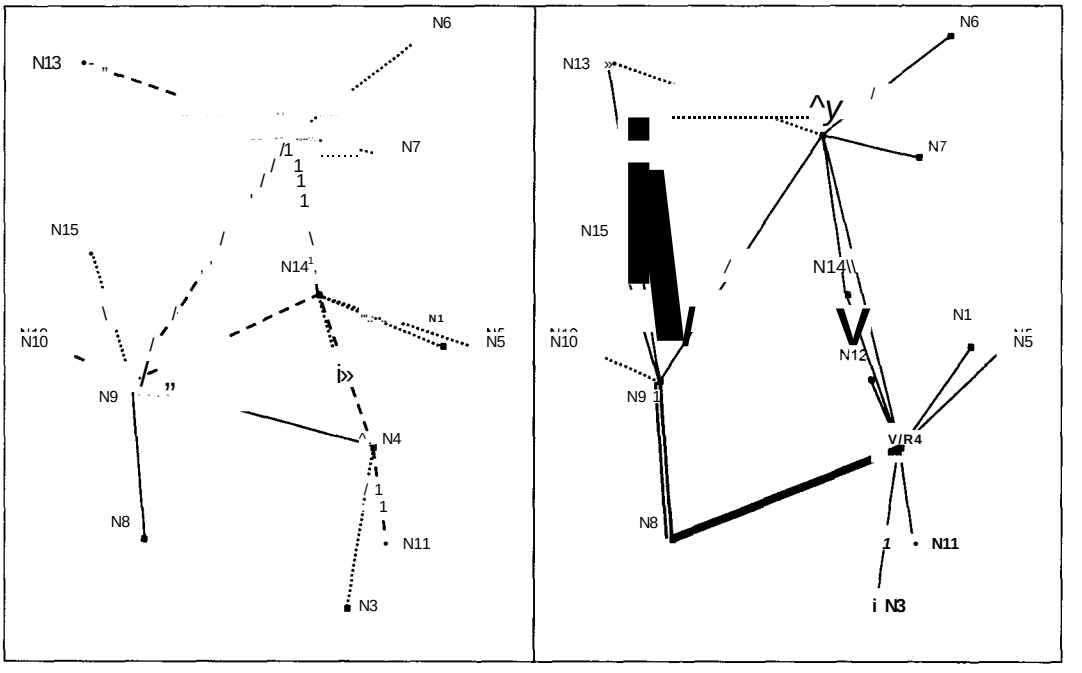


FIGURA 5.4 DISEÑOS FINALES DE MENTOR II Y PL RESPECTIVAMENTE.

SimIII	
Costo	272650
Útil. Prom. Total	0.7465
No. de líneas	37
Confiabilidad	0.9878

SimCplex 3	
Costo total	212444
Útil Promedio total	0.79407
No de líneas	26
Confiabilidad	0.9212

De estos diseños se observa que Cplex ofrece una solución más barata que Mentor II aunque se ve afectado en cuanto a utilización con respecto a Mentor II ya que éste último presenta una utilización de 74.65% mientras que Cplex nos da un valor más elevado llegando casi al 80% lo cual representa un problema en cuanto a ser un diseño óptimo. Si tomamos en cuenta el grado de confiabilidad ofrecida por ambos diseños Mentor II en su red nos da un valor de 98.78% y Cplex un 92.12% lo cual nos dice que con respecto a parámetros de utilización y confiabilidad Mentor II ofrece una mejor solución, pero Cplex presenta un diseño mas barato.

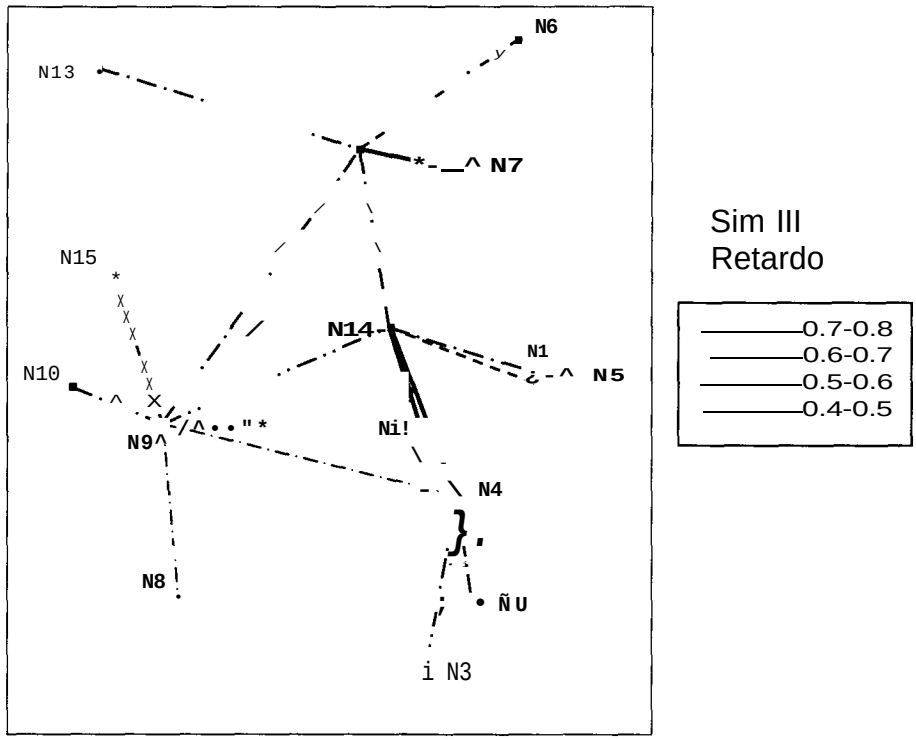


FIGURA 5.5 DIAGRAMA DE RETARDO DE SIM 3 DE MENTOR II.

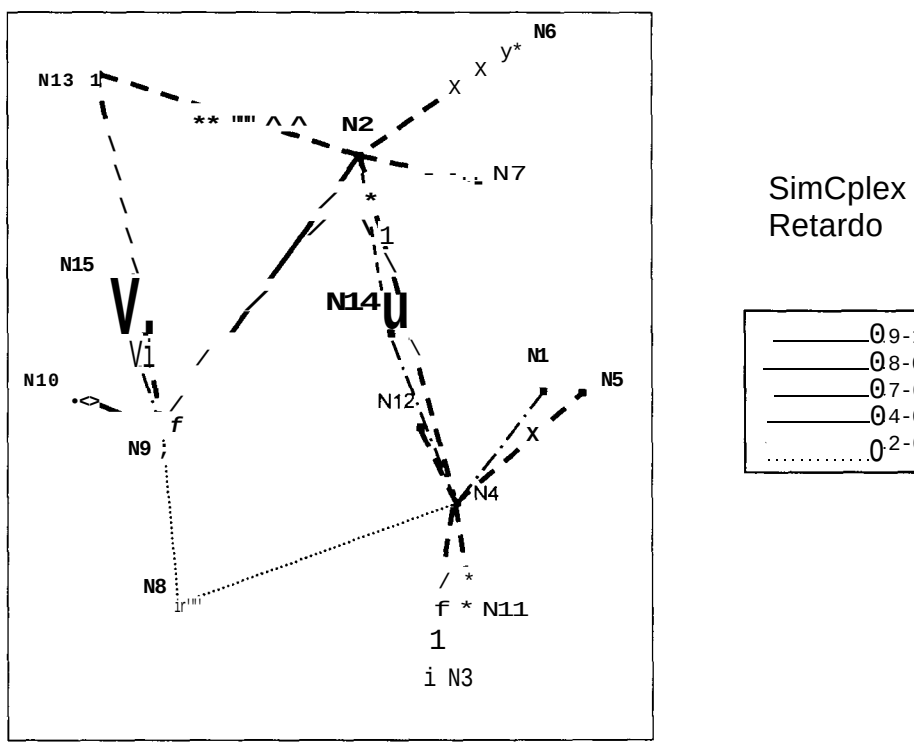


FIGURA 5.6 DIAGRAMA DE RETARDO DE SIM 3 DE CPLEX.

Si tomamos ahora los diseños con respecto a su retardo mostrados en las figuras 5.5 y 5.6 podemos ver que en el diseño de Mentor II obtenemos un retardo con un valor aproximado de 0.56 milisegundos con Cplex nos da 0.73 milisegundos siendo este un valor significativo. Ahora solo dependerá de las necesidades de cada empresa o diseñador en decidir que método utilizar de acuerdo a sus necesidades.

Lo que se podría decir a favor de Mentor II es el hecho de la creación de enlaces directos que permite un mejor acomodo de la red en comparación de la programación lineal esto debido a su algoritmo Incremental Shortest Path (ISP) que lo distingue de la versión anterior.

Por otro lado programación lineal maneja las conexiones de otra manera; esto es, busca todas las posibilidades existentes de conexiones entre los nodos hasta el caso de conectar todos contra todos aunque esto represente un alto grado de posibilidades de diseños; por lo tanto esto nos genera un margen bastante amplio en cuanto a resultados. Por esta razón se busco reducir el espacio solución a un numero finito de saltos en el diseño el cual se presento en el capitulo 4 mencionando un máximo de 6 saltos de cada uno de los pares de nodos.

Cabe mencionar que para programación lineal se recomienda empezar el diseño en base a una red ya establecida para a partir de ahí generar resultados óptimos de manera adecuada. Esto es debido a que podemos crear una amplia gama de soluciones con un alto número de enlaces directos y que no necesariamente sea lo óptimo.

5.2 Tiempos de simulación.

Como se dice comúnmente el "tiempo es oro" y por esta razón para diseños de redes es importante cuanto tiempo consume cada uno de estos procedimientos en sus respectivas simulaciones, y esto queda plasmado como sigue:

- Al trabajar con Mentor II es cuestión de segundos el que nos arroje un resultado aunque claro varia dependiendo de la velocidad de procesamiento de cada maquina.

- Con Programación lineal una simulación de un modelo planteado nos lleva en promedio máximo 30min.

- Con Mentor II si ya se tiene el algoritmo elaborado entonces solo es cuestión de capturar datos y ejecutar el algoritmo pero en caso de tener que

programar Mentor II y debido a su cierta complejidad nos tomaría un tiempo bastante considerable en desarrollarlo.

- Para PL elaborar el modelo matemático es cuestión de unos minutos en tenerlo listo; el problema es la creación de archivo de datos al tener que capturar toda la información necesaria ya que es un proceso un poco tardado debido a su importancia en cuanto a contenido. Y estos archivos de datos son los que llevan información específica de las limitantes del modelo por ello se desarrollo un programa que genera los archivos.

- Con Mentor II el crear un modelo requiere de varias simulaciones para buscar obtener un mejor modelo de diseño de redes y esto se hace de manera manual variando los diferentes parámetros ya antes mencionados.

- La ventaja de la programación lineal es que mientras ya se tenga una red inicial y el archivo de los datos sea confiable tendremos la certeza que nuestra primera simulación será la mas acertada.

Para terminar al ya tener una perspectiva más amplia del diseño de redes y el análisis detallado de estos 2 procedimientos podemos decir que no necesariamente estamos hablando de que uno sea mejor que el otro sino del hecho de las facilidades que ofrecen ambos diseños así como la posibilidad de complementarse uno con otro para así estar seguro de alcanzar un resultado bastante aceptable.

Anexo A

```
Set N := N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 N11 N12 N13 N14 N15;

setL := 1 2 34 5 6 7 89 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105;

set E:= 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189
190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210:

setD := (N1,N2)(N1,N3) (N1.N4) (N1.N5) (N1.N6) (N1.N7) (N1.N8) (N1.N9) (N1.N10) (N1.N11) (N1.N12) (N1.N13) (N1.N14) (N1.N15)
(N2.N1) (N2.N3) (N2,N4) (N2.N5) (N2.N6) (N2.N7) (N2.N8) (N2.N9) (N2.N10) (N2.N11) (N2.N12) (N2.N13) (N2.N14) (N2.N15) (N3,N1)
(N3.N2) (N3.N4) (N3.N5) (N3.N6) (N3,N7) (N3.N8) (N3.N9) (N3.N10) (N3.N11) (N3.N12) (N3.N13) (N3.N14) (N3.N15) (N4.N1) (N4.N2)
(N4.N3) (N4.N5) (N4.N6) (N4.N7) (N4.N8) (N4.N9) (N4.N10) (N4.N11) (N4.N12) (N4.N13) (N4.N14) (N4.N15) (N5.N1) (N5.N2) (N5,N3)
(N5.N4) (N5,N6) (N5.N7) (N5.N8) (N5.N9) (N5,N10) (N5.N11) (N5.N12) (N5.N13) (N5.N14) (N5.N15) (N6.N1) (N6.N2) (N6.N3) (N6.N4)
(N6.N5) (N6.N7) (N6.N8) (N6,N9) (N6.N10) (N6.N11) (N6.N12) (N6.N13) (N6.N14) (N6.N15) (N7.N1) (N7.N2) (N7.N3) (N7.N4) (N7,N5)
(N7.N6) (N7.N8) (N7.N9) (N7.N10) (N7.N11) (N7.N12) (N7.N13) (N7.N14) (N7.N15) (N8.N1) (N8.N2) (N8.N3) (N8.N4) (N8.N5) (N8.N6)
(N8.N7) (N8.N9) (N8.N10) (N8.N11) (N8.N12) (N8.N13) (N8.N14) (N8.N15) (N9.N1) (N9.N2) (N9.N3) (N9.N4) (N9.N5) (N9.N6) (N9.N7)
(N9.N8) (N9.N10) (N9.N11) (N9.N12) (N9.N13) (N9.N14) (N9,N15) (N10.N1) (N10.N2) (N10,N3) (N10.N4) (N10.N5) (N10.N6) (N10.N7)
(N10,N8) (N10.N9) (N10.N11) (N10.N12) (N10.N13) (N10.N14) (N10.N15) (N11.N1) (N11.N2) (N11.N3) (N11.N4) (N11.N5) (N11.N6)
(N11.N7) (N11,N8) (N11.N9) (N11.N10) (N11.N12) (N11.N13) (N11.N14) (N11.N15) (N12.N1) (N12,N2) (N12.N3) (N12.N4) (NI12.N5)
(N12.N6) (N12.N7) (N12.N8) (N12.N9) (N12.N10) (N12.N11) (N12.N13) (N12,N14) (N12.N15) (N13.N1) (N13.N2) (N13.N3) (N13.N4)
(N13.N5) (N13.N6) (N13.N7) (N13.N8) (N13.N9) (N13.N10) (N13.N11) (N13.N12) (N13.N14) (N13.N15) (N14.N1) (N14.N2) (N14.N3)
(N14.N4) (N14.N5) (N14.N6) (N14.N7) (N14.N8) (N14.N9) (N14.N10) (N14.N11) (N14.N12) (N14.N13) (N14.N15) (N15.N1) (N15,N2)
(N15.N3) (N15,N4) (N15.N5) (N15.N6) (N15.N7) (N15.N8) (N15.N9) (N15.N10) (N15.N11) (N15.N12) (N15.N13) (N15.N14);

set P := 1 2 34 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189
190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221
222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253
254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285
286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317
318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349
350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381
382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445
446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477
478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509
510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541
542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573
574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605
606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637
638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669
670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701
702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733
734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765
766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797
798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829
830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861
862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893
894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925
926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957
958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989
990 991 992 993 994 995 996 997 998 999 1000 1001 1002 1003 1004 1005 1006 1007 1008 1009 1010 1011 1012 1013 1014 1015 1016
1017 1018 1019 1020 1021 1022 1023 1024 1025 1026 1027 1028 1029 1030 1031 1032 1033 1034 1035 1036 1037 1038 1039 1040 1041
1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053 1054 1055 1056 1057 1058 1059 1060 1061 1062 1063 1064 1065 1066
1067 1068 1069 1070 1071 1072 1073 1074 1075 1076 1077 1078 1079 1080 1081 1082 1083 1084 1085 1086 1087 1088 1089 1090 1091
1092 1093 1094 1095 1096 1097 1098 1099 1100 1101 1102 1103 1104 1105 1106 1107 1108 1109 1110 1111 1112 1113 1114 1115 1116
1117 1118 1119 1120 1121 1122 1123 1124 1125 1126 1127 1128 1129 1130 1131 1132 1133 1134 1135 1136 1137 1138 1139 1140 1141
1142 1143 1144 1145 1146 1147 1148 1149 1150 1151 1152 1153 1154 1155 1156 1157 1158 1159 1160 1161 1162 1163 1164 1165 1166
1167 1168 1169 1170 1171 1172 1173 1174 1175 1176 1177 1178 1179 1180 1181 1182 1183 1184 1185 1186 1187 1188 1189 1190 1191
1192 1193 1194 1195 1196 1197 1198 1199 1200 1201 1202 1203 1204 1205 1206 1207 1208 1209 1210 1211 1212 1213 1214 1215 1216
1217 1218 1219 1220 1221 1222 1223 1224 1225 1226 1227 1228 1229 1230 1231 1232 1233 1234 1235 1236 1237 1238 1239 1240 1241
1242 1243 1244 1245 1246 1247 1248 1249 1250 1251 1252 1253 1254 1255 1256 1257 1258 1259 1260;
```

```
setPD[N1.N2] := 12 3456;
setPD[N1,N4] := 13 14 15 16 17 18;
setPD[N1,N6]:= 25 26 27 28 29 30;
setPD[N1,N8] := 37 38 39 40 41 42;
setPD[N1,N10] := 49 50 51 52 53 54;
setPD[N1,N12]:= 61 62 63 64 65 66;
setPD[N1,N14]:= 73 74 75 76 77 78;
setPD[N2,N1] := 85 86 87 88 89 90;
set PD[N2,N4] := 97 98 99 100 101 102;
set PD[N2,N6] := 109 110 111 112 113 114;
set PD[N2,N8] := 121 122 123 124 125 126;
setPD[N2,N10] := 133 134 135 136 137 138;
setPD[N2.N12] := 145 146 147 148 149 150;
set PD[N2.N14] := 157 158 159 160 161 162;
setPD[N3,N1]:= 169 170 171 172 173 174;
setPD[N3,N4]:= 181 182 183 184 185 186;
set PD[N3,N6] := 193 194 195 196 197 198;
set PD[N3.N8] := 205 206 207 208 209 210;
setPD[N3,N10] := 217 218 219 220 221 222;
setPD[N3,N12] := 229 230 231 232 233 234;
set PD[N3,N14] := 241 242 243 244 245 246;
set PD[N4,N1] := 253 254 255 256 257 258;
setPD[N4,N3] := 265 266 267 268 269 270;
set PD[N4,N6] := 277 278 279 280 281 282;
set PD[N4,N8] := 289 290 291 292 293 294;
setPD[N4,N10] := 301 302 303 304 305 306;
setPD[N4.N12] := 313314 315316317318;
set PD[N4,N14J := 325 326 327 328 329 330;
setPD[N5,N1] := 337 338 339 340 341 342;
set PD[N5,N3] := 349 350 351 352 353 354;
set PD[N5,N6] := 361 362 363 364 365 366;
set PD[N5,N8] := 373 374 375 376 377 378;
setPD[N5,N10] := 385 386 387 388 389 390;
setPD[N5,N12] := 397 398 399 400 401 402;
set PD[N5.N14] := 409 410 411 412 413 414;
set PD[N6,N1] := 421 422 423 424 425 426;
set PD[N6,N3] := 433 434 435 436 437 438;
set PD[N6,N5] := 445 446 447 448 449 450;
setPD[N6,N8] := 457 458 459 460 461 462;
setPD[N6.N10J := 469 470 471 472 473 474;
setPD[N6,N12] := 481 482 483 484 485 486;
setPD[N6,N14] := 493 494 495 496 497 498;
setPD[N7,N1] := 505 506 507 508 509 510;
set PD[N7,N3] := 517 518 519 520 521 522;
setPD[N7,N5]:= 529 530 531 532 533 534;
set PD[N7,N8] := 541 542 543 544 545 546;
setPD[N7,N10]:= 553 554 555 556 557 558;
setPD[N7,N12]:= 565 566 567 568 569 570;
setPD[N7,N14] := 577 578 579 580 581 582;
setPD[N8,N1] := 589 590 591 592 593 594;
set PD[N8,N3] := 601 602 603 604 605 606;
set PD[N8,N5] := 613614615616617618;
set PD[N8,N7] := 625 626 627 628 629 630;
setPD[N8,N10] := 637 638 639 640 641 642;
setPD[N8,N12] := 649 650 651 652 653 654;
set PD[N8,N14] := 661 662 663 664 665 666,
set PD[N9,N1] := 673 674 675 676 677 678;
set PD[N9,N3] := 685 686 687 688 689 690;
set PD[N9,N5] := 697 698 699 700 701 702;
set PD[N9,N7] := 709710711 712713714;
setPD[N9,N10] := 721 722 723 724 725 726;
setPD[N9,N12] := 733 734 735 736 737 738;
set PD[N9,N14] := 745 746 747 748 749 750;
setPD[N10,N1] := 757 758 759 760 761 762;
setPD[N10,N3] := 769 770 771 772 773 774;
set PD[N10,N5] := 781 782 783 784 785 786;
setPD[N10,N7] := 793 794 795 796 797 798;
setPD[N10,N9]:= 805 806 807 808 809 810;
setPD[N10,N12] := 817 818 819 820 821 822;
setPD[N10,N14] := 829 830 831 832 833 834;
setPD[N11.N1]:= 841 842 843 844 845 846;
setPD[N11,N3]:= 853 854 855 856 857 858;
setPD[N11,N5] := 865 866 867 868 869 870;
setPD[N11,N7] := 877 878 879 880 881 882;
setPD[N11,N9] := 889 890 891 892 893 894;
setPD[N11,N12] := 901 902 903 904 905 906;
setPD[N11,N14] := 913914 915916917918;
setPD[N12,N1] := 925 926 927 928 929 930;
setPD[N12,N3]:= 937 938 939 940 941 942;

set PD[N1.N3] := 789 10 11 12;
set PD[N1.N5] := 19 20 21 22 23 24;
set PD[N1.N7] := 31 32 33 34 35 36;
set PD[N1,N9] := 43 44 45 46 47 48;
set PD[N1.N11] := 55 56 57 58 59 60;
set PD[N1.N13] := 67 68 69 70 71 72;
set PD[N1,N15] := 79 80 81 82 83 84;
set PD[N2.N3] := 91 92 93 94 95 96;
set PD[N2,N5] := 103 104 105 106 107 108;
setPD[N2,N7]:= 115 116 117 118 119 120;
set PD[N2.N9] := 127 128 129 130 131 132;
set PD[N2.N11] := 139 140 141 142 143 144;
setPD[N2.N13] := 151 152 153 154 155 156;
setPD[N2.N15] := 163 164 165 166 167 168;
set PD[N3,N2] := 175 176 177 178 179 180;
setPD[N3,N5] := 187 188 189 190 191 192;
set PD[N3.N7] := 199 200 201 202 203 204;
set PD[N3.N9] := 211 212 213 214 215 216;
set PD[N3,N11] := 223 224 225 226 227 228;
setPD[N3.N13] := 235 236 237 238 239 240;
setPD[N3,N15]:= 247 248 249 250 251 252;
set PD[N4,N2] := 259 260 261 262 263 264;
set PD[N4.N5] := 271 272 273 274 275 276;
set PD[N4.N7] := 283 284 285 286 287 288;
set PD[N4,N9] := 295 296 297 298 299 300;
set PD[N4,N11] := 307 308 309 310 311 312;
setPD[N4.N13] := 319 320 321 322 323 324;
setPD[N4.N15] := 331 332 333 334 335 336;
setPD[N5,N2J := 343 344 345 346 347 348;
setPD[N5.N4] := 355 356 357 358 359 360;
set PD[N5.N7] := 367 368 369 370 371 372;
set PD[N5,N9] := 379 380 381 382 383 384;
setPD[N5,N11] := 391 392 393 394 395 396;
setPD[N5.N13] := 403 404 405 406 407 408;
setPD[N5.N15] := 415416417418419420;
setPD[N6,N2] := 427 428 429 430 431 432;
setPD[N6,N4] := 439 440 441 442 443 444;
set PD[N6.N7] := 451 452 453 454 455 456;
set PD[N6.N9] := 463 464 465 466 467 468;
setPD[N6,N11] := 475 476 477 478 479 480;
setPD[N6.N13] := 487 488 489 490 491 492;
setPD[N6.N15] := 499 500 501 502 503 504;
set PD[N7,N2] := 511 512 513 514 515 516;
set PD[N7,N4] := 523 524 525 526 527 528;
setPD[N7.N6] := 535 536 537 538 539 540;
setPD[N7.N9] := 547 548 549 550 551 552;
setPD[N7.N11) := 559 560 561 562 563 564;
setPD[N7,N13] := 571 572 573 574 575 576;
setPD[N7.N15] := 583 584 585 586 587 588;
setPD[N8.N2] := 595 596 597 598 599 600;
set PD[N8.N4] := 607 608 609 610 611 612;
set PD[N8,N6] := 619 620 621 622 623 624;
setPD[N8,N9] := 631 632 633 634 635 636;
setPD[N8.N11] := 643 644 645 646 647 648;
setPD[N8.N13] := 655 656 657 658 659 660;
setPD[N8.N15] := 667 668 669 670 671 672;
setPD[N9,N2] := 679 680 681 682 683 684;
set PD[N9,N4] := 691 692 693 694 695 696;
set PD[N9.N6] := 703 704 705 706 707 708;
set PD[N9.N8] := 715 716 717 718 719 720;
set PD[N9,N11] := 727 728 729 730 731 732;
setPD[N9,N13] := 739 740 741 742 743 744;
setPD[N9.N15] := 751 752 753 754 755 756;
setPD[N10,N2] := 763 764 765 766 767 768;
setPD[N10.N4J := 775 776 777 778 779 780;
setPD[N10.N6] := 787 788 789 790 791 792;
set PD[N10,N8] := 799 800 801 802 803 804;
set PD[N10,N11] := 811 812 813 814 815 816;
setPD[N10.N13] := 823 824 825 826 827 828;
setPD[N10.N15] := 835 836 837 838 839 840;
set PD[N11,N2] := 847 848 849 850 851 852;
set PD[N11.N4] := 859 860 861 862 863 864;
set PD[N11,N6] := 871 872 873 874 875 876;
set PD[N11,N8] := 883 884 885 886 887 888;
set PD[N11,N10] := 895 896 897 898 899 900;
set PD[N11.N13] := 907 908 909 910 911 912;
set PD[N11,N15] := 919 920 921 922 923 924;
setPD[N12.N2] := 931 932 933 934 935 936;
setPD[N12,N4] := 943 944 945 946 947 948;
```

```
setPD[N12,N5] := 949 950 951 952 953 954;
setPD[N12,N7] := 961 962 963 964 965 966:
setPD[N12,N9] := 973 974 975 976 977 978;
setPD[N12,N11] := 985 986 987 988 989 990;
setPD[N12,N14] := 997 998 999 1000 1001 1002;
setPD[N13,N1] := 1009 1010 1011 1012 1013 1014;
setPD[N13.N3] ;= 1021 1022 1023 1024 1025 1026;
setPD[N13.N5] ;= 1033 1034 10351036 1037 1038;
setPD[N13.N7] ;= 1045 1046 1047 1048 1049 1050;
setPD[N13.N9] ;= 1057 1058 1059 1060 1061 1062;
setPD[N13,N11] := 1069 1070 1071 1072 1073 1074;
setPD[N13.N14] := 1081 1082 1083 1084 1085 1086;
setPD[N14,N1] ;= 1093 1094 1095 1096 1097 1098;
setPD[N14,N3] := 1105 1106 1107 1108 1109 1110;
set PD[N14,N5] ;= 1117 1118 11191120 1121 1122;
set PD[N14,N7] := 1129 1130 1131 1132 1133 1134;
setPD[N14,N9] := 1141 1142 1143 1144 1145 1146;
setPD[N14,N11] := 1153 1154 1155 1156 1157 1158;
setPD[N14,N13] := 1165 1166 1167 1168 1169 1170;
setPD[N15.N1] ;= 1177 1178 1179 1180 1181 1182;
setPD[N15.N3] := 1189 1190 1191 1192 1193 1194;
setPD[N15,N5] := 1201 1202 1203 1204 1205 1206;
setPD[N15,N7]:= 1213 1214 1215 1216 1217 1218;
setPD[N15,N9]:= 1225 1226 1227 1228 1229 1230;
setPD[N15.N11] := 1237 1238 1239 1240 1241 1242;
setPD[N15,N13] := 1249 1250 1251 1252 1253 1254;

setPD[N12,N6] := 955 956 957 958 959 960:
setPD[N12,N8]:= 967 968 969 970 971 972;
setPD[N12,N10] := 979 980 981 982 983 984;
setPD[N12,N13] := 991 992 993 994 995 996;
setPD[N12,N15] := 1003 1004 1005 1006 1007 1008;
setPD[N13,N2] := 1015 1016 1017 1018 1019 1020;
setPD[N13,N4]:= 1027 1028 1029 1030 1031 1032;
setPD[N13.N6] := 1039 1040 1041 1042 1043 1044;
setPD[N13,N8] := 1051 1052 1053 1054 1055 1056;
setPD[N13.N10] := 1063 1064 1065 1066 1067 1068;
setPD[N13,N12] := 1075 1076 1077 1078 1079 1080;
setPD[N13,N15] := 1087 1088 1089 1090 1091 1092;
setPD[N14,N2]:= 1099 1100 1101 1102 1103 1104;
setPD[N14.N4]:= 1111 1112 1113 1114 1115 1116;
setPD[N14,N6] := 1123 1124 1125 1126 1127 1128;
setPD[N14.N8]:= 1135 1136 1137 1138 1139 1140;
setPD[N14,N10] := 1147 1148 1149 1150 1151 1152;
setPD[N14.N12] := 1159 1160 1161 1162 1163 1164;
setPD[N14.N15] := 1171 1172 1173 1174 1175 1176;
setPD[N15,N2]:= 1183 1184 1185 1186 1187 1188;
setPD[N15,N4] := 1195 1196 1197 1198 1199 1200;
setPD[N15,N6]:= 1207 1208 1209 1210 1211 1212;
setPD[N15,N8]:=12191220 1221 1222 1223 1224;
setPD[N15,N10] := 1231 1232 1233 1234 1235 1236;
setPD[N15,N12] := 1243 1244 1245 1246 1247 1248;
setPD[N15,N14] := 1255 1256 1257 1258 1259 1260;

setPE[   1]:=      1       344;
set PE[   2] :=    7;
set PE[   3] :=    9       13      56      149     162     184     347     348     352     356     366
                   371     372     377     383     390     393     399     408     412     485     498     516
                   552     568     581     856     861     940     944     971     1080;
set PE[   4] :=    19      3       4       5       6       10      11      12      14      15      16
                   17      18      27      29      30      32      34      35      36      39      40      41
                   42      45      46      47      48      52      53      54      57      58      59      60
                   62      63      64      65      66      70      71      72      75      76      77      78
                   82      83      84      102     104     105     106     107     108     113     114     118
                   119     120     150     189     190     226     227     228     263     268     269     270
                   272     273     274     275     276     294     309     310     311     312     444     447
                   448     449     450     456     486     527     528     530     531     532     533     534
                   539     564     569     570     582     614     615     616     617     618     683     684
                   690     694     696     698     699     700     701     702     708     713     714     726
                   731     737     738     744     748     755     768     779     780     782     783     784
                   785     786     798     810     816     821     822     833     840     852     866     867
                   868     869     870     934     935     941     942     945     946     947     948     950
                   951     952     953     954     959     964     972     978     983     988     989     990
                   996     999     1008    1034    1036    1037    1038    1102    1103    1104    1109    1110
                   1113    1114    1115    1116    1118    1119    1120    1121    1122    1126    1127    1128
                   1131    1132    1133    1140    1144    1145    1151    1152    1157    1158    1161    1163
                   1164    1170    1176    1200    1202    1203    1204    1205    1206    1229    1230    1236
                   1248    1260;
set PE[   5] :=    25;
set PE[   6] :=    31      2       26      69      345     360     363     368     406     1134;
set PE[   7] :=    37      374;
set PE[   8] :=    43      380;
setPE[   9]:=      49      386;
setPE[   10]:=     55      8       351     392     400     401     402     413     414;
setPE[   11] :=    61      20      21      28      33      38      44      50      51      74      81
                   101     131     132     138     144     148     161     185     216     232     233     245
                   246     264     282     287     288     298     300     305     306     314     315     327
                   328     329     330     336     346     353     357     365     370     375     376     378
                   381     384     387     388     389     394     398     407     411     418     419     420
                   432     455     468     497     515     540     551     558     580     612     653     654
                   666     857     862     882     893     900     904     917     918     1062    1086;
setPE[   12]:=     67      404;
setPE[   13]:=     73      22      223     24      68      80      160     186     234     316     317
                   318     354     358     359     364     369     395     396     405     41      417     431
                   496     514     579     749     756     834     858     863     864     905     906     936
                   960     965     966     977     984     1000    1001    1002;
set PE[   14] :=   79      416;
setPE[   15]:=     85      104;
setPE[   16] :=    91;
setPE[   17]:=     97      140;
setPE[   18] :=    103;
setPE[   19] :=    109     116     536     537     538     540     704     788     789     1040    1047
                   1208;
```

```
set PE[ 20] :=      115     36      86      87      90      96      100     101     102     105     108
          110     130     131     132     137     138     143     144     147     148     149     150
          158     159     160     161     162     168     204     258     276     281     286     288
          372     452     707     712     790     792     794     795     797     876     881     958
          963     966     1011    1012    1031    1036    1041    1042    1046    1061    1062    1068
          1074    1078    1080    1083    1086    1125    1130    1209    1214    1218;
setPE[ 21] :=       121;
set PE[ 22] : =     127     122     154     165     464     1090:
set PE[ 23] :=      133     470     554;
set PE[ 24] :=      139;
set PE[ 25] :=      145:
set PE[ 26] : =     151     69      70      71      72      240     322     323     406     407     408
          488     489     572     573     659     743     744     827     828     911     912     993
          994     996     1166    1167    1170    1252    1253    1254;
set PE[ 27] : =     157     18      88      89      92      93      94      95      98      99      106
          107     111     112     113     114     117     118     119     120     123     124     125
          126     128     135     136     141     142     146     153     155     156     164     166
          167     360     443     453     454     456     474     480     484     510     522     526
          528     534     557     563     567     570     578     588     1013    1014    1026    1030
          1037    1038    1055    1067    1073    1077    1082    1091    1092;
set PE[ 28] : =     163     129     134     152     467     471     472     490     500     501     550
          555     584;
set PE[ 29] :=      169;
set PE[ 30] : =     175;
set PE[ 31] :=      181     65      66      78      178     194     195     200     224     226     227
          228     230     243     342     401     402     414     846     870     903     906     916
          918     948     1116    1164;
set PE[ 32] :=      187     864;
set PE[ 33] :=      193;
set PE[ 34] :=      199;
set PE[ 35] :=      205     41      42      126     212     219     291     294     377     378     546
          719     720     804     884     970     971     972     1056    1139    1140    1146
          1224:
set PE[ 36] :=      211     218     237     348     249;
set PEJ 37] :=      217;
set PE[ 38] :=      223     24      60      170     171     172     173     174     179     180     182
          183     184     185     186     188     189     190     191     192     196     197     198
          201     202     203     204     206     207     208     209     210     213     214     215
          216     220     221     222     231     232     233     234     238     239     240     244
          245     246     250     251     252     312     318     330     396     593     594     600
          609     612     617     618     630     635     636     642     644     652     653     654
          660     665     666     672     750     929     930     953     954     987     990     1002
          1098    1122    1156    1158;
set PE[ 39] :=      229     177     225     242;
set PE[ 40] :=      235;
setPE[ 41] :=       241     176;
set PE[ 42] :=      247     236;
set PE[ 43] :=      253     107     108     120     171     190     226     232     245     268     272
          309     314     327     450     533     534     617     653     666     701     714     786
          842     867     935     951     959     964     996     1038    1104    1120    1128
          1133;
set PE[ 44] :=      259     848;
set PE[ 45] :=      265     24      60      94      318     330     396     434     435     518     854
          856     857     858     929     930     938     953     954     987     990     1002    1098
          1107    1122    1156    1158;
set PE[ 46] : =     271     21      22      89      90      172     216     233     234     246     254
          264     282     287     288     298     300     305     306     315     316     328     326
          426     509     510     591     593     654     676     677     749     756     761     762
          834     843     882     893     900     904     905     906     917     918     927     936
          960     965     966     977     984     1000    1014    1096    1181    1182;
set PE[ 47] :=      277     194     872;
set PE[ 48] : =     283     195     196     200     201     278     873     878:
set PE[ 49] :=      289     39      40      124     207     376     461     544     718     802     803
          968     1055    1137    1222;
set PE[ 50] :=      295     213     220     302     809     890     897     1143    1150    1175    1228
          1235;
setPE[ 51] :=       301;
set PE[ 52] :=      307     9       10      11      12      23      41      42      56      57      58
          59      95      96      125     126     140     142     143     144     224     225     257
          266     275     291     317     329     352     353     354     377     378     393     394
          395     436     437     438     462     476     477     478     479     480     519     520
          521     522     545     546     560     561     562     563     564     603     604     605
          606     645     646     647     648     687     688     689     690     719     720     728
          729     730     731     732     772     773     774     804     813     814     815     816
          928     939     940     941     942     952     969     970     971     972     986     988
          989     1001    1024    1025    1026    1056    1070    1071    1072    1073    1074    1097
          1108    1109    1110    1121    1134    1138    1139    1140    1146    1155    1157    1192
          1194    1224    1238    1239    1240    1241    1242;
```

```
set PE[ 53] :=      313     5       6       30      35      36      46      47      53      54      63
          64      65      72      76      77      78      83      84      149     150     162     173
          174     178     179     180     191     192     197     198     202     203     208     209
          210     214     215     221     222     227     228     230     231     238     239     240
          243     244     250     251     252     255     256     258     261     262     263     267
          269     270     273     274     276     279     280     281     284     285     286     290
          292     293     294     296     297     303     304     308     310     311     312     320
          321     322     323     324     326     332     333     334     335     339     340     341
          342     347     348     366     371     372     383     390     399     400     401     408
          412     413     414     485     486     498     516     552     568     569     570     581
          582     592     594     598     599     600     616     618     623     624     628     629
          630     634     635     636     640     641     642     650     651     652     659     660
          663     664     665     670     671     672     725     735     737     747     750     754
          820     822     832     839     844     845     846     850     851     852     855     868
          869     870     874     875     876     879     880     881     885     886     887     888
          891     892     894     898     899     902     903     909     910     911     912     915
          916     921     922     923     924     1080    1145    1152    1160    1162    1164    1230
          1247    1259;
set PE[ 54] :=      319     908;
set PE[ 55] :=      325     66      260     402     975     982     998     1007;
set PE[ 56] :=      331     920;
set PE[ 57] :=      337     87      88      89      90      101     131     132     138     144     148
          149     160     161     162     172     173     174     184     185     186     216     233
          234     246     254     255     256     257     258     264     282     287     288     298
          300     305     306     315     316     317     318     328     329     330     336     344
          345     346     347     348     351     352     353     354     356     357     358     359
          360     363     364     365     366     368     369     370     371     372     374     375
          376     377     378     380     381     382     383     384     386     387     388     389
          390     392     393     394     395     396     398     399     400     401     402     404
          405     406     407     408     410     411     412     413     414     416     417     418
          419     420     423     425     426     431     432     455     468     485     496     497
          498     506     508     509     510     514     515     516     540     551     552     558
          568     579     580     581     591     592     593     594     612     654     675     676
          677     678     749     756     760     761     762     834     843     844     845     846
          856     857     858     861     862     863     864     882     893     900     904     905
          906     917     918     926     927     928     929     930     936     960     965     966
          977     984     1000    1001    1002    1012    1013    1014    1062    1080    1086    1095
          1096    1097    1098    1134    1180    1181    1182;

set PE[ 58] :=      343;
set PE[ 59] :=      349     312;
set PE[ 60] :=      355     5       6       10      14      30      35      36      39      41      46
          47      53      54      57      63      72      76      83      84      102     150     339
          340     444     486     527     528     564     569     570     690     694     696     731
          737     779     780     816     822     941     942     945     946     972     988     989
          990     1110    1113    1114    1140    1145    1152    1157    1158    1162    1200
          1230;
setPE[ 61]:=        361     539;
set PE[ 62] :=      367     3       18      27      32      70      113     114     118     119     120
          263     362     456     683     684     708     713     714     744     768     852     934
          935     959     964     996     1102    1103    1104    1126    1127    1128    1131    1132
          1133    1170;
set PE[ 63] :=      373;
set PE[ 64] :=      379;
set PE[ 65] :=      385;
set PE[ 66] :=      391     64      65      66      77      78      226     227     228     268     269
          270     294     309     310     311     341     342     350     582     947     948     1115
          1116    1163    1164;
set PE[ 67] :=      397     4       11      15      29      34      40      42      45      48      52
          58      62      71      75      82      338     738     810     821     1144    1151    1161
          1176    1229    1236    1248;
set PE[ 68] :=      403;
set PE[ 69] :=      409     12      16      17      59      60      726     748     755     833     840
          978     983     999     1008    1260;
set PE[ 70] :=      415;
setPE[ 71]:=        421;
set PE[ 72] :=      427     452     453     454     456     464     470     471     488     500     512
          573;
set PE[ 73] :=      433;
set PE[ 74] :=      439     434     476:
set PE[ 75] :=      445     455;
set PE[ 76] :=      451     116     422     423     424     425     426     428     429     430     431
          432     435     436     437     438     440     441     442     443     444     446     447
          448     449     450     459     460     461     462     465     466     467     468     472
          473     474     477     478     479     480     482     483     484     485     486     489
          490     491     492     494     495     496     497     498     501     502     503     504
          1047;
set PE[ 77] :=      457;
set PE[ 78] :=      463;
```

```
set PE[ 79] :=       469;
set PE[ 80] :=       475;
set PE[ 81] :=       481;
set PE[ 82] :=       487;
set PE[ 83] :=       493       458;
set PE[ 84] :=       499;
set PE[ 85] :=       505       86        105       276       422       447       530       582       1011      1036;
set PE[ 86] :=       511       2         3         6         18        69        70        180       262       263       264
          323       345       348       360       406       428       443       467       472       474       480       484
          489       490       495       501       510       522       526       528       534       536       550       554
          555       557       563       567       570       572       578       584       588       682       683       684
          743       744       767       768       828       851       852       912       913       934       935       936
          994       996       1100      1101      1102      1103      1104      1167      1170      1188;
set PE[ 87] :=       517;
set PE[ 88] :=       523       435       436       440       477       518       519       560:
set PE[ 89] :=       529       87        101       131       132       138       144       148       149       160       161
          162       258       423       431       432       446       468       485       496       497       498       506
          514       515       516       540       551       552       558       568       579       580       581       1012
          1062      1080      1086:
set PE[ 90] :=       535       26        27        28        29        30        110       111       112       113       114
          195       196       197       198       278       279       280       281       282       362       363       364
          365       366       512       573       621       622       623       624       705       706       707       708
          790       791       792       873       874       875       876       956       957       958       959       960
          1041      1042      1043      1044      1124      1125      1126      1127      1128      1209      1210      1211
          1212;
setPEt 91] :=        541;
set PE[ 92] :=       547;
set PE[ 93] :=       553;
set PE[ 94] :=       559;
set PE[ 95] :=       565       159       430       437       441       478       482       513       520       524       538
          561;
set PE[ 96] :=       571;
set PE[ 97] :=       577       90        96        100       102       108       130       137       143       147       150
          158       168       424       425       426       429       438       442       444       448       449       450
          459       460       461       462       465       466       473       479       483       486       491       492
          494       502       503       504       507       508       509       521       525       527       531       532
          533       537       539       542       543       544       545       546       548       549       556       562
          564       566       569       574       575       576       585       586       587       1031      1061      1068
          1074      1078      1083;
set PE[ 98] :=       583;
set PE[ 99] :=       589       614;
setPE[100]:=         595;
setPE[101] ;=        601       593       594       600       609       612       617       618       630       635       636
          642       644       652       653       654       660       665       666       672       686       750       771;
setPE[ 102] ;=       607       591       592       598       603       616       623       628       634       640       641
          650       659       663       670       671;
setPE[103];=         613;
setPE[104]:=         619;
setPE[105]:=         625;
setPE[106]:=         631       212       219       596       605       606       610       638       639       646       647
          656       657       658       668       669       807       826       837       1060      1146      1227;
setPE[107]:=         637       611       632       633       648       742       753       1066      1233:
setPE[108] :=        643       599       602       624       629       651       664;
setPE[109]:=         649       590       597       604       608       615       622       627       645       662;
setPE[110]:=         655;
setPE[111]:=         661       620       621       626;
setPE[112] :=        667       723;
setPE[113] :=        673       698;
setPE[114]:=         679       596       704       1018      1185      1252;
setPE[115]:=         685       770       1023      1190      1191;
setPE[116] :=        691       687       725       728       747       754       772       776       813       832       839
          1259;
setPE[117] :=        697;
setPE[118] :=        703;
setPE[ 119] :=       709;
setPE[120]:=         715       122       209       210       292       686       723       742       750       753       771
          800       801       886       887       1052      1053      1054      1066      1220      1221      1233;
setPE[121] ;=        721       51        52        53        54        84        135       136       137       138       154
          155       156       165       166       167       168       218       219       220       221       222       249
          252       293       302       303       304       305       306       324       334       335       336       388
          389       390       419       420       473       474       492       503       504       556       557       558
          575       576       586       587       588       638       640       641       642       658       669       671
          678       681       695       696       702       707       712       716       732       736       738       741
          752       888       896       897       898       899       900       923       924       980       981       982
          983       984       995       1005      1006      1007      1008      1065      1067      1068      1090      1091
          1092      1148      1149      1150      1151      1152      1168      1169      1173      1174      1175      1176
          1232      1234      1235      1236      1251;
set PEí 122] : =     727       812;
```

```
setPE[123]:=    733     605     646     674     675     676     682     683     688     690     692
          694   699     706     711     719     724     726     729     731     743     744     746
          748   749     755     756     759     760     761     7 56    767     768     773     777
          779   784     791     792     796     797     802     804     814     816     818     827
          828   831     833     834     838     840     1020    1050    1085    1182    1187    1188
          1194  1198    1200    1205    1212    1217    1218    1223    1241    1245    1254    1258
          1260;
setPE[124]:=    739     656;
setPE[ 125] :=  745     606     610     611     647     648     677     680     684     689     693
          700   701     705     708     710     713     714     718     720     730     734     735
          737   762     765     774     778     780     785     786     798     803     815     819
          820   821     822     830     1019    1032    1043    1044    1048    1049    1079    1084
          1186  1199    1206    1210    1211    1215    1216    1242    1246    1247    1248    1253
          1256  1257;
setPE[126]:=    751     237     248     491     502     574     585     639     657     668     717
          722   748     825     826     836     837     1089    1172:
setPE[127] :=   757     782;
setPE[128] :=   763     788     794;
setPE[129]:=    769;
setPE[130]:=    775;
setPE[ 131] : = 781;
setPE[132]:=    787;
setPE[133]:=    793;
setPE[134]:=    799     293     716     717     826     837     888     1060    1227;
setPE[135]:=    805     48      129     299     300     384     467     550     611     632     648
          759   760     761     762     765     766     767     768     770     771     772     773
          774   776     777     778     779     780     784     785     786     791     792     796
          797   798     800     802     803     804     812     813     814     815     816     818
          819   820     821     822     825     827     828     830     831     832     833     834
          836   838     839     840     894     976     978     1018    1019    1020    1032    1044
          1049  1050    1054    1059    1079    1084    1085    1089    1182    1185    1186    1188
          1191  1194    1198    1199    1200    1205    1206    1211    1212    1216    1217    1218
          1221  1223    1226    1241    1242    1245    1246    1247    1248    1252    1253    1254
          1257  1258    1259    1260;
setPE[136]:=    811;
setPE[137]:=    817     758     783;
setPE[138]: =   823     1250    1251;
setPE[139] :=   829     696;
set PE[ 140] :=  835    84      154     155     156     165     166     167     168     249     252
          324   334     335     336     419     420     492     503     504     575     576     586
          587   588     633     658     669     671     678     681     790     795     801     806
          807   808     809     810     824     923     924     995     1005    1006    1007    1008
          1088  1090    1091    1092    1168    1169    1173    1174    1175    1176;
set PE[ 141] :=  841    170     189     866     952     953     954     1121    1122;
setPE[142]:=    847;
setPE[143]:=    853     8       9       10      11      12      41      42      65      66      78
          95    96      126     266     267     268     269     270     291     294     342     350
          351   352     353     354     377     378     401     402     414     436     437     458
          519   520     521     522     546     602     603     604     605     606     687     688
          689   690     719     720     772     773     774     804     846     864     870     884
          903   906     916     918     939     940     941     942     948     970     971     972
          1024  1025    1026    1056    1108    1109    1110    1116    1139    1140    1146    1164
          1192  1193    1194    1224;
setPE[144]:=    859     17      64      77              171     172     173     174     179     180
          182   190     191     192     196     197     198     201     202     203     204     207
          208   209     210     213     214     215     216     220     221     222     231     232
          233   234     238     239     240     244     245     246     250     251     252     341
          359   400     413     582     593     594     599     600     609     617     618     624
          629   630     635     636     642     651     652     653     654     660     664     665
          666   672     750     842     843     844     845     848     850     851     852     854
          855   867     868     869     872     873     874     875     876     878     879     880
          881   882     885     886     887     888     889     890     891     892     893     894
          897   898     899     900     902     904     905     908     909     910     911     912
          915   917     920     921     922     923     924     947     1115    1163;
setPE[145]:=    865     23      24      184     185     186     188     317     318     329     330
          612   856     857     858     861     862     863     928     930     1001    1002    1097
          1098  1134;
setPE[ 146]:=   871;
setPE[147] :=   877;
setPE[148]:=    883     125     206     462     545     969     1138;
set PE[ 149]:=  889     896;
setPE[150]:=    895;
setPE[151]:=    901     183     257     275     849     860     914;
setPE[152]:=    907;
setPE[153]:=    913;
setPE[154]:=    919;
setPE[155]:=    925     102     106     114     119     191     227     269     273     294     310
          338   339     424     444     449     456     507     527     528     532     539     564
          590   615     616     618     674     683     684     690     694     696     699     700
```

```
        702     708     713     726     731     744     748     755     758     759     768     779
        780     783     784     785     798     816     833     840     852     868     934     940
        941     949     945     950     971     972     978     983     988     999     1008    1037
        1094    1103    1110    1114    1115    1116    1119    1127    1132    1157    1158    1170
        1179    1200    1204    1206    1260;
setPE[156]:=    931;
setPE[157]:=    937     93      855     1106;
setPE[158]:=    943     11      12      15      16      18      21      22      23      24      40
        42      58      59      60      89      90      94      95      96      99      100     101
        107     108     120     124     125     126     142     143     144     183     185     225
        353     354     357     358     360     376     378     394     395     396     426     437
        438     441     442     443     450     461     462     478     479     480     509     510
        520     522     524     525     526     533     534     544     545     546     561     562
        563     604     605     606     608     610     611     612     645     646     647     648
        676     677     688     689     692     693     695     701     714     718     719     720
        729     730     732     749     756     761     762     773     774     777     778     786
        802     803     804     809     814     815     834     857     858     860     862     863
        869     927     928     929     935     936     938     939     951     952     953     959
        960     964     965     966     968     969     970     975     977     982     984     986
        987     996     998     1000    1001    1002    1007    1014    1024    1025    1026    1028
        1029    1030    1031    1032    1038    1055    1056    1075    1072    1073    1074    1096
        1097    1098    1104    1107    1108    1112    1120    1121    1122    1128    1133    1134
        1137    1138    1139    1143    1146    1150    1155    1156    1175    1181    1182    1192
        1193    1194    1196    1197    1198    1199    1222    1223    1224    1228    1235    1239
        1240    1241    1242;
setPE[159]:=    949     20      88      173     255     425     508     592     594     675     678
        760     844     926     1013    1095    1180;
setPE[160]:=    955;
setPE[161]:=    961     112     117     197     202     279     284     454     874     879     956
        1101;
setPE[162];=    967     38      123     208     290     375     460     543     885     1136;
setPE[163]:=    973     44      45      46      51      52      53      84      130     131     136
        137     138     156     167     168     209     214     216     221     252     296     298
        303     305     334     336     381     388     419     466     473     474     504     549
        556     557     576     587     588     635     640     642     671     808     810     886
        891     893     898     900     923     980     1005    1061    1062    1067    1068    1092
        1142    1144    1145    1149    1151    1152    1169    1174    1176    1229    1230    1234
        1236;
setPE[164]:=    979     50      387;
setPE[165]:=    985     17      141     267     308     359     1154;
setPE[166]:=    991     238     320     909;
setPE[167]:=    997     4       5       6       28      29      30      33      34      35      36
        47      48      58      71      72      74      75      76      77      78      81      82
        83      132     159     161     162     174     177     178     179     180     192     198
        203     204     210     215     222     228     239     240     242     243     244     245
        246     251     256     257     258     261     262     264     270     274     275     276
        280     281     282     285     286     287     288     292     293     297     299     300
        304     306     311     312     321     323     324     326     327     328     329     330
        333     335     340     341     342     346     347     348     365     366     370     371
        372     382     383     384     389     390     407     408     411     412     413     414
        418     420     430     432     455     468     497     498     513     515     516     538
        540     551     552     558     580     581     582     597     598     599     600     622
        623     624     627     711     724     725     743     746     747     750     754     766
        767     791     792     796     797     827     828     831     832     838     839     845
        846     849     850     851     869     870     875     876     880     881     882     887
        888     892     894     899     910     911     912     914     915     916     917     918
        922     924     930     932     933     942     946     947     948     954     957     958
        962     963     974     976     981     989     990     992     993     994     995     1004
        1006    1020    1050    1085    1086    1187    1188    1212    1217    1218    1254    1258
        1259;
setPE[168];=    1003    250     332     921;
setPE[169]:=    1009    1034;
setPE[170]:=    1015    1011    1012    1013    1014    1026    1030    1031    1036    1037    1038
        1040    1041    1046    1047    1055    1061    1062    1067    1068    1073    1074    1078
        1080    1082    1083    1086    1090    1091    1092;
setPE[171]:=    1021;
setPE[172];=    1027    1070;
setPE[173]:=    1033;
setPE[174]:=    1039;
setPE[175]:=    1045;
setPE[176]:=    1051;
setPE[177]:=    1057    1052;
setPE[178]:=    1063    1088    1089;
setPE[179]:=    1069;
setPE[180]:=    1075    1024    1028    1071:
setPE[181]:=    1081    1010    1025    1029    1035    1056    1072    1076;
setPE[182]:=    1087    1016    1017    1018    1019    1020    1022    1023    1032    1042    1043
        1044    1048    1049    1050    1053    1054    1058    1059    1060    1064    1065    1066
        1079    1084    1085;
```

```
setPE[183]:=    1093    113     118     150     192     228     270     274     275     311     312
        340     341     342     448     486     531     569     570     73~     738     810     821
        822     869     870     942     946     947     948     986     990     1010    1035    1102
        1109    1113    1118    1126    1131    1140    1144    1145    1151    1152    1161    1162
        1163    1164    1176    1178    1203    1229    1230    1236    1248;
setPE[184]:=    1099    4       5       36      71      72      176     177     179     204     240
        258     260     261     276     281     286     288     322     346     347     372     407
        408     429     430     431     432     513     514     515     516     537     538     548
        597     598     599     600     659     680     765     766     793     797     827     849
        850     876     881     911     932     958     963     966     993     1017    1019    1020
        1125    1130    1166    1184    1186    1187    1218    1253    1254;
setPE[185]:=    1105    92;
setPE[186]:=    1111    98      735     820     930     954     1160    1247;
setPE[187]:=    1117    174     256     257     845     846;
setPE[188]:=    1123    620;
setPE[189]:=    1129    6       28      30      33      34      35      111     180     198     203
        262     264     280     282     285     287     322     348     364     365     366     369
        370     371     453     455     621     622     623     624     626     627     628     629
        630     682     705     706     710     711     743     767     791     796     828     851
        876     880     882     912     933     936     957     960     962     965     994     1043
        1044    1048    1049    1050    1100    1124    1167    1188    1210    1211    1212    1215
        1216    1217;
setPE[ 190]:=   1135    458     459     542;
setPE[191]:=    1141    47      54      128     132     135     155     166     210     215     222
        292     293     297     304     306     324     335     382     383     389     390     420
        465     468     491     492     502     503     548     551     552     558     574     575
        585     586     634     636     641     887     888     892     899     924     974     975
        976     977     981     983     984     995     1006    1007    1008    1091    1148    1168
        1172    1173;
setPE[192]:=    1147    300;
setPE[193]:=    1153;
setPE[194]:=    1159    12      16      17      18      22      23      24      59      60      66
        88      89      90      93      94      95      96      99      100     102     106     107
        108     112     114     117     119     120     123     124     125     126     130     136
        137     141     142     143     146     147     156     167     168     186     234     316
        317     318     354     358     359     360     395     396     402     424     425     426
        438     442     443     444     449     450     454     456     460     461     462     466
        473     474     479     480     483     484     504     507     508     509     510     521
        522     525     526     527     528     532     533     534     539     543     544     545
        546     549     556     557     562     563     564     566     567     576     587     588
        606     610     611     647     648     677     678     684     689     693     695     696
        700     701     702     708     713     714     718     720     730     732     734     736
        762     774     778     780     785     786     798     803     808     809     815     819
        858     863     864     905     906     1013    1014    1025    1026    1029    1030    1031
        1032    1037    1038    1055    1056    1061    1067    1068    1072    1073    1074    1076
        1077    1078    1079    1092    1094    1095    1096    1097    1098    1101    1103    1104
        1106    1107    1108    1110    1112    1114    1115    1116    1119    1120    1121    1122
        1127    1128    1132    1133    1134    1136    1137    1138    1139    1142    1143    1146
        1149    1150    1154    1155    1156    1157    1158    1169    1170    1174    1175    1179
        1180    1181    1193    1197    1199    1204    1206    1222    1224    1228    1234    1235
        1240    1242    1244    1246;
setPE[195]:=    1165    68      239     321     405     660     910     992;
set PE[ 196] : =1171    48      80      81      82      153     164     251     299     333     384
        417     418     670     672     724     725     726     754     755     756     838     839
        840     894     922     976     978     1004;
setPE[197]:=    1177    1202;
setPE[198]:=    1183    681     707     712     764     789     790     1016    1042    1208    1209
        1214;
setPE[199]:=    1189    1022;
set PE[ 200] :=1195     1238;
setPE[201]:=    1201;
set PE[ 202] :=1207;
setPE[203] : =  1213;
set PE[ 204] :=1219     807;
set PE[ 205] :=1225     633     801     806     1023    1043    1048    1053    1058    1065    1066
        1190    1210    1215    1220    1232    1233    1251    1256;
setPE[206]:=    1231    48      129     134     249     384     467     471     472     550     555
        639     717     722     723     724     725     726     894     976     978     1018    1019
        1020    1032    1044    1050    1054    1059    1060    1064    1079    1084    1085    1182
        1185    1186    1187    1188    1191    1198    1199    1200    1205    1206    1211    1212
        1216    1217    1218    1221    1223    1226    1227    1241    1242    1245    1246    1247
        1248    1250    1252    1253    1254    1257    1258    1259    1260;
setPE[207]:=    1237;
set PE[ 208] :=1243     1192    1196    1239;
setPE[209]:=    1249    152     153     154     155     156     236     237     324     490     491
        492     574     575     576     657     658     740     741     742     824     825     826
        995     1168    1169;
```

```
setPE[210]:=      1255    678     695     702     732     736     738     808     809     810     1017
          1178    1179    1180    1181    1184    1193    1197    1203    1204    1222    1224    1228
          1229    1230    1234    1235    1236    1240    1244;
```

param demand :=
[N1.N2] 255000 [N1.N3] 255000 [N1.N4] 255000 [N1.N5] 255000 [N1.N6] 255000 [N1.N7] 255000 [N1.N8] 255000 [N1.N9] 255000 [N1.N10] 255000 [N1.N11] 255000 [N1.N12] 255000 [N1.N13] 255000 [N1.N14] 255000 [N1.N15] 255000 [N2.N1] 255000 [N2.N3] 255000 [N2.N4] 255000 [N2.N5] 255000 [N2.N6] 255000 [N2.N7] 255000 [N2.N8] 255000 [N2.N9] 255000 [N2.N10] 255000 [N2.N11] 255000 [N2,N12] 255000 [N2.N13] 255000 [N2.N14] 255000 [N2.N15] 255000 [N3.N1] 255000 [N3.N2] 255000 [N3,N4] 255000 [N3.N5] 255000 [N3.N6] 255000 [N3.N7] 255000 [N3.N8] 255000 [N3.N9] 255000 [N3.N10] 255000 [N3.N11] 255000 [N3,N12] 255000 [N3.N13] 255000 [N3.N14] 255000 [N3.N15] 255000 [N4,N1] 255000 [N4,N2] 255000 [N4.N3] 255000 [N4.N5] 255000 [N4.N6] 255000 [N4.N7] 255000 [N4.N8] 255000 [N4.N9] 255000 [N4.N10] 255000 [N4.N11] 255000 [N4.N12] 255000 [N4.N13] 255000 [N4.N14] 255000 [N4.N15] 255000 [N5.N1] 255000 [N5.N2] 255000 [N5.N3] 255000 [N5.N4] 255000 [N5.N6] 255000 [N5.N7] 255000 [N5.N8] 255000 [N5.N9] 255000 [N5.N10] 255000 [N5.N11] 255000 [N5.N12] 255000 [N5.N13] 255000 [N5.N14] 255000 [N5.N15] 255000 [N6.N1] 255000 [N6.N2] 255000 [N6.N3] 255000 [N6.N4] 255000 [N6.N5] 255000 [N6.N7] 255000 [N6.N8] 255000 [N6.N9] 255000 [N6.N10] 255000 [N6.N11] 255000 [N6.N12] 255000 [N6.N13] 255000 [N6.N14] 255000 [N6.N15] 255000 [N7.N1] 255000 [N7.N2] 255000 [N7.N3] 255000 [N7.N4] 255000 [N7.N5] 255000 [N7.N6] 255000 [N7.N8] 255000 [N7.N9] 255000 [N7.N10] 255000 [N7.N11] 255000 [N7.N12] 255000 [N7.N13] 255000 [N7.N14] 255000 [N7.N15] 255000 [N8.N1] 255000 [N8.N2] 255000 [N8.N3] 255000 [N8.N4] 255000 [N8.N5] 255000 [N8.N6] 255000 [N8.N7] 255000 [N8.N9] 255000 [N8.N10] 255000 [N8.N11] 255000 [N8.N12] 255000 [N8.N13] 255000 [N8.N14] 255000 [N8.N15] 255000 [N9.N1] 255000 [N9.N2] 255000 [N9.N3] 255000 [N9.N4] 255000 [N9.N5] 255000 [N9,N6] 255000 [N9.N7] 255000 [N9,N8] 255000 [N9,N10] 255000 [N9,N11] 255000 [N9.N12] 255000 [N9,N13] 255000 [N9.N14] 255000 [N9.N15] 255000 [N10.N1] 255000 [N10,N2J 255000 [N10.N3] 255000 [N10.N4] 255000 [N10,N5] 255000 [N10.N6] 255000 [N10.N7] 255000 [N10.N8] 255000 [N10.N9] 255000 [N10.N11] 255000 [N10.N12] 255000 [N10.N13] 255000 [N10.N14] 255000 [N10.N15] 255000 [N11.N1] 255000 [N11.N2] 255000 [N11.N3] 255000 [N11.N4J 255000 [N11.N5] 255000 [N11.N6] 255000 [N11.N7] 255000 [N11.N8] 255000 [N11.N9] 255000 [N11.N10] 255000 [N11.N12] 255000 [N11.N13] 255000 [N11.N14] 255000 [N11,N15] 255000 [N12.N1] 255000 [N12.N2] 255000 [N12.N3] 255000 [N12.N4J 255000 [N12.N5] 255000 [N12,N6] 255000 [N12.N7] 255000 [N12.N8] 255000 [N12.N9] 255000 [N12.N10] 255000 [N12.N11] 255000 [N12.N13] 255000 [N12.N14] 255000 [N12.N15] 255000 [N13.N1] 255000 [N13.N2] 255000 [N13.N3] 255000 [N13.N4] 255000 [N13,N5] 255000 [N13,N6] 255000 [N13.N7] 255000 [N13.N8] 255000 [N13.N9] 255000 [N13.N10] 255000 [N13.N11] 255000 [N13.N12] 255000 [N13.N14] 255000 [N13.N15] 255000 [N14.N1] 255000 [N14.N2] 255000 [N14.N3] 255000 [N14.N4] 255000 [N14.N5] 255000 [N14.N6] 255000 [N14.N7] 255000 [N14.N8] 255000 [N14.N9] 255000 [N14.N10] 255000 [N14.N11] 255000 [N14.N12] 255000 [N14.N13] 255000 [N14.N15] 255000 [N15.N1] 255000 [N15.N2] 255000 [N15.N3J 255000 [N15.N4] 255000 [N15.N5] 255000 [N15.N6] 255000 [N15.N7] 255000 [N15.N8] 255000 [N15.N9] 255000 [N15.N10] 255000 [N15.N11] 255000 [N15.N12] 255000 [N15.N13] 255000 [N15.N14] 255000;

Anexo B

```
setN := N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 N11 N12 N13 N14 N15;

setL := 1 2 34 5 67 89 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105;

setE := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189
190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210;

setD := (N1.N2) (N1.N3) (N1.N4) (N1.N5) (N1.N6) (N1.N7) (N1.N8) (N1.N9) (N1.N10) (N1.N11) (N1.N12) (N1.N13) (N1.N14) (N1.N15)
(N2.N1) (N2.N3) (N2.N4) (N2.N5) (N2.N6) (N2.N7) (N2.N8) (N2.N9) (N2.N10) (N2.N11) (N2.N12) (N2.N13) (N2.N14) (N2.N15) (N3.N1)
(N3.N2) (N3.N4) (N3.N5) (N3.N6) (N3.N7) (N3.N8) (N3.N9) (N3.N10) (N3.N11) (N3.N12) (N3.N13) (N3.N14) (N3.N15) (N4.N1) (N4.N2)
(N4,N3)(N4,N5)(N4,N6) (N4.N7) (N4.N8) (N4.N9) (N4.N10) (N4.N11) (N4.N12) (N4.N13) (N4.N14) (N4.N15) (N5.N1) (N5.N2) (N5.N3)
(N5.N4) (N5.N6) (N5.N7) (N5.N8) (N5,N9) (N5.N10) (N5.N11) (N5.N12) (N5.N13) (N5.N14) (N5.N15) (N6.N1) (N6.N2) (N6.N3) (N6.N4)
(N6.N5) (N6.N7) (N6.N8) (N6.N9) (N6.N10) (N6.N11) (N6.N12) (N6.N13) (N6.N14) (N6.N15) (N7.N1) (N7.N2) (N7.N3) (N7.N4) (N7.N5)
(N7.N6) (N7.N8) (N7.N9) (N7.N10) (N7.N11) (N7.N12) (N7.N13) (N7.N14) (N7.N15) (N8.N1) (N8.N2) (N8.N3) (N8.N4) (N8.N5) (N8.N6)
(N8.N7) (N8.N9) (N8.N10) (N8.N11) (N8.N12) (N8.N13) (N8.N14) (N8.N15) (N9.N1) (N9.N2) (N9.N3) (N9.N4) (N9.N5) (N9.N6) (N9.N7)
(N9.N8) (N9.N10) (N9.N11) (N9.N12) (N9.N13) (N9.N14) (N9.N15) (N10.N1) (N10.N2) (N10.N3) (N10.N4) (N10.N5) (N10.N6) (N10.N7)
(N10.N8) (N10.N9) (N10.N11) (N10.N12) (N10.N13) (N10.N14) (N10.N15) (N11.N1) (N11.N2) (N11.N3) (N11.N4) (N11.N5) (N11.N6)
(N11.N7) (N11.N8) (N11.N9) (N11.N10) (N11.N12) (N11.N13) (N11.N14) (N11.N15) (N12.N1) (N12.N2) (N12.N3) (N12.N4) (N12.N5)
(N12.N6) (N12.N7)(N12.N8)(N12,N9)(N12,N10) (N12.N11) (N12.N13) (N12.N14) (N12.N15) (N13.N1) (N13.N2) (N13.N3) (N13.N4)
(N13.N5) (N13.N6) (N13.N7) (N13.N8) (N13.N9) (N13.N10) (N13.N11) (N13.N12) (N13.N14) (N13.N15) (N14.N1) (N14.N2) (N14.N3)
(N14.N4) (N14.N5) (N14.N6) (N14.N7) (N14.N8) (N14.N9) (N14.N10) (N14.N11) (N14.N12) (N14.N13) (N14.N15) (N15.N1) (N15.N2)
(N15.N3) (N15.N4) (N15.N5) (N15.N6) (N15.N7) (N15.N8) (N15.N9) (N15.N10) (N15.N11) (N15.N12) (N15.N13) (N15.N14);

setP ;= 1 2 3 4 5 6 7 89 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189
190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221
222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253
254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285
286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317
318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349
350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381
382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445
446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477
478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509
510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538;

setPD[N1,N2]:= 1 2 3;
set PD[N1,N4] := 5;
setPD[N1.N6] := 7 8 9;
setPD[N1,N8] := 13 14 15;
setPD[N1,N10] := 19 20 21;
set PD[N1.N12] := 23 24 25;
set PD[N1,N14] := 29 30 31,
set PD[N2,N1] := 35 36 37;
set PD[N2,N4] := 41 42 43;
set PD[N2,N6] ;= 47;
set PD[N2,N8] := 49 50 51;
setPD[N2,N10] :=55 56 57;
set PD[N2.N12J := 61 62 63;
set PD[N2,N14] := 65 66 67;
setPD[N3.N1] :=71;
set PD[N3.N4] := 75;
set PD[N3,N6] := 77 78 79;
set PD[N3.N8] := 83 84 85;
setPD[N3,N10] := 89 90 91;
set PD[N3,N12] := 93 94 95;
set PD[N3,N14] := 99 100 101;
set PD[N4.N1] := 105;
setPD[N4,N3] := 109;
setPD[N4,N6] := 111 112 113;
setPD[N4,N8] := 117 118 119;
setPD[N4,N10] ;= 123 124 125;

set PD[N1,N3] ;= 4;
set PD[N1,N5] := 6:
setPD[N1,N7] ;= 10 11 12;
set PD[N1,N9] := 16 17 18;
set PD[N1,N11] := 22;
set PD[N1.N13] := 26 27 28;
set PD[N1,N15] := 32 33 34;
set PD[N2,N3] := 38 39 40;
set PD[N2,N5] ;= 44 45 46;
set PD[N2,N7] := 48:
set PD[N2.N9] := 52 53 54;
set PD[N2,N11] := 58 59 60;
set PD[N2,N13] := 64;
set PD[N2,N15] := 68 69 70;
setPD[N3,N2J :=72 73 74;
set PD[N3,N5] := 76;
set PD[N3.N7] := 80 81 82;
set PD[N3.N9] := 86 87 88;
set PD[N3,N11] := 92;
set PD[N3,N13] := 96 97 98;
set PD[N3.N15] := 102 103 104;
set PD[N4.N2] := 106 107 108;
setPD[N4,N5] := 110;
set PD[N4,N7] := 114 115 116;
set PD[N4,N9] := 120 121 122;
set PD[N4.N11] := 126;
```

```
setPD[N4,N12]:= 127 128 129:
setPD[N4,N14J := 133 134 135;
setPD[N5,N1] := 139;
setPD[N5,N3J : = 143;
set PD[N5,N6] := 145 146 147;
set PD[N5,N8] := 151 152 153:
setPD[N5,N10] := 157 158 159;
setPD[N5,N12] := 161 162 163;
setPD[N5,N14] := 167 168 169;
set PD[N6,N1J := 173 174 175;
set PD[N6,N3] := 177 178 179;
setPD[N6,N5] := 183 184 185;
set PD[N6,N8] := 187 188 189;
setPD[N6,N10] := 193 194 195;
setPD[N6,N12]:= 199 200 201;
set PD[N6.N14] := 203 204 205;
setPD[N7.N1] := 209 210 211;
set PD[N7,N3] := 213 214 215;
setPD[N7,N5] := 219 220 221;
set PD[N7,N8] := 223 224 225;
setPD[N7,N10] := 229 230 231;
setPD[N7,N12] := 235 236 237;
setPD[N7,N14] := 239 240 241;
setPD[N8,N1] := 245 246 247;
set PD[N8,N3] := 251 252 253;
setPD[N8,N5] := 257 258 259;
set PD[N8,N7] := 263 264 265;
setPD[N8,N10] := 267;
setPD(N8,N12] := 271 272 273;
set PD[N8,N14] := 277 278 279;
setPD[N9,N1] := 281 282 283;
set PD[N9,N3] := 287 288 289;
set PD[N9,N5] := 293 294 295;
setPD[N9,N7]:= 299 300 301;
setPD[N9,N10] := 303;
setPD[N9,N12] := 307 308 309;
setPD[N9,N14] := 313 314 315;
setPD[N10,N1] := 317 318 319;
setPD[N10,N3] := 323 324 325;
setPD[N10,N5] := 329 330 331;
setPD[N10,N7] := 335 336 337;
setPD[N10,N9] := 339;
setPD[N10,N12] := 343 344 345;
setPD(N10,N14] := 349 350 351;
set PD[N11,N1]:= 353;
setPD[N11,N3] := 357;
setPD[N11,N5] := 359;
setPD[N11,N7]:= 363 364 365;
set PD[N11.N9]:= 369 370 371;
set PD[N11,N12] := 375 376 377;
setPD[N11,N14] := 381 382 383;
setPD[N12,N1] := 387 388 389;
setPD[N12,N3] := 393 394 395,
setPD[N12.N5] := 399 400 401;
setPD[N12.N7] := 405 406 407;
setPD[N12,N9]:= 411 412413;
set PD[N12.N11] := 417 418 419;
setPD[N12,N14] :=423;
setPD[N13,N1] := 427 428 429:
setPD[N13.N3]:= 431 432 433;
setPD[N13,N5] := 437 438 439;
setPD[N13,N7] := 441;
setPD[N13,N9] := 445 446 447;
setPD[N13,N11] := 451 452 453;
setPD[N13.N14] := 457 458 459;
setPD[N14,N1]:= 463 464 465;
set PD[N14,N3] := 469 470 471;
setPD[N14.N5] := 475 476 477;
setPD[N14.N7]:= 481 482 483;
setPD[N14,N9] := 487 488 489;
setPD[N14,N11] := 493 494 495;
setPD[N14,N13] := 497 498 499;
setPD[N15,N1] := 503 504 505;
setPD[N15,N3]:= 509510511;
setPD[N15,N5]:= 515 516 517;
setPD[N15.N7] := 521 522 523;
setPD[N15,N9] := 525;
set PD[N15,N11] := 527 528 529;
setPD[N15,N13] := 533 534 535;

setPD[N4.N13] := 130 131 132;
setPD[N4,N15] := 136 137 138;
set PD[N5.N2] := 140 141 142;
set PD[N5,N4] : = 144;
set PD[N5,N7] := 148 149 150;
set PD[N5,N9J := 154 155 156;
set PD[N5,N11] := 160;
setPD[N5,N13] := 164 165 166;
setPD[N5,N15J := 170 171 172;
set PD[N6,N2] := 176;
set PD[N6.N4] : = 180 181 182;
set PD[N6,N7] := 186;
setPD[N6,N9] : = 190 191 192;
setPD[N6.N11] := 196 197 198.
setPD[N6.N13¡ := 202;
setPD[N6,N15]:= 206 207 208;
set PD[N7,N2] := 212;
setPD[N7,N4] : = 216217218;
set PD[N7,N6] := 222;
setPD[N7,N9] := 226 227 228;
setPD[N7,N11] := 232 233 234;
setPD[N7,N13] := 238;
setPD[N7,N15] := 242 243 244;
setPD[N8,N2]:= 248 249 250;
setPD[N8,N4¡ := 254 255 256;
set PD[N8,N6] := 260 261 262;
setPD[N8,N9] : = 266;
setPD[N8,N11] := 268 269 270;
setPD[N8,N13] := 274 275 276;
setPD[N8.N15] := 280;
set PD[N9,N2] := 284 285 286;
setPD[N9,N4] : = 290 291 292;
set PD[N9,N6] := 296 297 298;
set PD[N9,N8] := 302;
set PD[N9.N11] := 304 305 306;
setPD[N9,N13] := 310 311 312;
setPD[N9,N15]:= 316;
setPD[N10,N2]:= 320 321 322;
setPD[N10,N4] := 326 327 328;
setPD[N10,N6] := 332 333 334;
setPD[N10,N8J := 338;
setPD[N10,N11] := 340 341 342;
setPD[N10,N13] := 346 347 348;
setPD[N10,N15] := 352;
setPD[N11,N2] := 354 355 356;
set PD[N11.N4] := 358;
setPD[N11.N6]:= 360 361 362;
setPD[N11,N8] := 366 367 368;
set PD[N11.N10] := 372 373 374;
set PD[N11,N13] := 378 379 380;
setPD[N11,N15] := 384 385 386;
setPD[N12,N2] := 390 391 392;
setPDiN12,N4] := 396 397 398;
setPD[N12,N6] := 402 403 404;
setPD[N12,N8] := 408 409 410;
setPD[N12.N10] :=414415416:
setPD[N12,N13] := 420 421 422;
setPD[N12.N15] := 424 425 426;
setPD[N13,N2] := 430;
setPD[N13,N4] := 434 435 436;
setPD[N13.N6] := 440;
setPD[N13,N8] := 442 443 444:
setPD[N13,N10] := 448 449 450;
setPD[N13,N12] := 454 455 456;
setPD[N13,N15] := 460 461 462;
setPD[N14,N2] := 466 467 468;
set PD[N14,N4]:= 472 473 474;
setPD[N14,N6] := 478 479 480;
setPD[N14,N8] := 484 485 486;
setPD[N14,N10] := 490 491 492;
setPD[N14,N12] := 496;
setPD[N14,N15] := 500 501 502;
setPD[N15,N2] := 506 507 508;
setPD[N15,N4] := 512 513 514;
setPD[N15,N6] := 518 519 520;
setPD[N15,N8] := 524;
setPD[N15,N10] := 526;
setPD[N15,N12] := 530 531 532;
setPD[N15,N14] := 536 537 538;
```

```
setPE[  1]:=        ;
set PE[ 2] :=       ;
set PE[ 3] :=       1       2       3       4       5       6       7       8       9       10      11
            12      13      14      15      16      17      18      19      20      21      22      23
            24      25      26      27      28      29      30      31      32      33      34;
set PE[ 4] :=       ;
set PE[ 5] : =      ;
set PE[ 6] :=       ;
set PE[ 7] : =      ;
set PE[ 8] : =      ;
set PE[ 9] : =      ;
setPE[ 10] :=       ;
setPE[ 11] :=       ;
setPE[  12]:=       ;
setPE[  13]:=       ;

set PE[ 14] :=      ;
setPE[ 15] :=       ;
setPE[ 16] :=       ;
setPE[ 17] : =      35      38      41      44      49      52      55      58      62      63      66
            67      68      173     177     180     183     187     190     193     196     200     201
            204     205     206     209     213     216     219     223     226     229     232     236
            237     240     241     242     247     253     256     259     270     283     289     292
            295     306     319     325     328     331     342     388     394     400     410     413
            416     418     426     427     431     434     437     442     448     451     455     456
            458     459     460     464     470     473     476     486     489     492     494     502
            505     511     514     517     529;
setPE[ 18] :=       ;
setPE[ 19] : =      7       8       9       47      77      78      79      111     112     113     145
            146     147     222     260     261     262     296     297     298     333     334     335
            360     361     362     402     403     404     440     478     479     480     518     519
            520;
set PE[ 20] :=      10      11      12      48      80      81      82      114     115     116     148
            149     150     186     263     264     265     299     300     301     335     336     337
            363     364     365     405     406     407     441     481     482     483     521     522
            523;
setPE[ 21] :=       ;
set PE[  22] :=     ;
set PE[  23] :=     ;
set PE[  24] :=     ;
set PE[  25] :=     ;
set PE[  26] :=     26      27      28      64      96      97      98      130     131     132     164
            165     166     202     238     274     275     276     310     311     312     346     347
            348     378     379     380     420     421     422     497     498     499     533     534
            535;
set PE[ 27] :=      15      18      21      24      30      37      39      40      42      43      45
            46      50      51      53      54      56      57      59      60      61      65      69
            70      85      88      91      94      100     104     119     122     125     128     134
            138     153     156     159     162     168     172     174     175     178     179     181
            182     184     185     188     189     191     192     194     195     197     198     199
            203     207     208     210     211     214     215     217     218     220     221     224
            225     227     228     230     231     233     234     235     239     243     244     273
            279     315     351     368     371     374     376     382     386     428     429     432
            433     435     436     438     439     443     444     446     447     449     450     452
            453     454     457     461     462     532     538;
set PE[ 28] :=      ;
set PE[ 29] :=      ;
set PE[ 30] :=      ;
set PE[ 31] :=      71      72      73      74      75      76      77      78      79      80      81
            82      83      84      85      86      87      88      89      90      91      92      93
            94      95      96      97      98      99      100     101     102     103     104;
set PE[ 32] :=      ;
set PE[ 33] :=      ;
set PE[ 34] :=      ;
set PE[ 35] :=      ;
set PE[ 36] :=      ;
set PE[ 37] :=      ;
set PE[ 38] :=      ;
set PE[ 39] : =     ;
set PE[ 40] :=      ;
setPE[ 41] :=       ;
set PE[ 42] :=      ;
set PE[ 43] :=      35      36      37      71      105     139     173     174     175     209     210
            211     245     246     247     281     282     283     317     318     319     353     387
            388     389     427     428     429     463     464     465     503     504     505;
set PE[ 44] :=      1       7       10      15      18      21      24      26      30      34      72
            77      80      85      88      91      94      96      100     104     106     111     114
            119     122     125     128     130     134     138     140     145     148     153     156
```

159 162 164 168 172 248 260 263 273 274 279 284
296 299 309 310 315 320 332 335 345 346 351 354
360 363 371 374 376 378 382 386 391 392 403 404
406 407 421 467 468 479 480 482 483 498 499 506
518 521 532 533 538;
set PE[45] := 4 38 39 40 109 143 177 178 179 213 214
215 251 252 253 287 288 289 323 324 325 357 393
394 395 431 432 433 469 470 471 509 510 511;
set PE[46] := 6 44 45 46 76 110 183 184 185 219 220
221 257 258 259 293 294 295 329 330 331 359 399
400 401 437 438 439 475 476 477 515 516 517;
set PE[47] := ;
set PE[48] := ;
set PE[49] := ;
set PE[50] := 3 9 12 13 16 25 28 31 32 49 51
52 54 55 57 63 67 68 70 74 79 82 83
86 89 95 98 101 102 108 113 116 117 120 123
129 132 135 136 142 147 150 151 154 157 163 166
169 170 187 190 192 193 195 201 205 206 208 223
226 228 229 231 237 241 242 356 362 365 366 369
372 377 380 382 383 409 410 412 413 415 416 425
426 442 444 445 447 448 450 456 459 460 462 485
486 488 489 491 492 501 502;
set PE[51] := ;
set PE[52] := 22 58 59 60 92 126 160 196 197 198 232
233 234 268 269 270 304 305 306 340 341 342 417
418 419 451 452 453 493 494 495 527 528 529;
set PE[53] := ;
set PE[54] := ;
set PE[55] := 2 8 11 14 17 20 23 27 29 33 62
66 73 78 81 84 87 90 93 97 99 103 107
115 118 121 124 127 131 133 137 141 146 149 152
155 158 161 165 167 171 200 204 236 240 262 265
272 276 278 286 298 301 308 312 314 322 334 337
344 348 350 355 361 364 367 370 373 375 379 381
385 508 520 523 531 535 537,
set PE[56] := ;
set PE[57] := ;
set PE[58] := ;
set PE[59] := ;
set PE[60] := 139 140 141 142 143 144 145 146 147 148 149
150 151 152 153 154 155 156 157 158 159 160 161
162 163 164 165 166 167 168 169 170 171 172;
set PE[61] := ;
set PE[62] := ;
set PE[63] := ;
set PE[64] := ;
set PE[65] := ;
set PE[66] := ;
set PE[67] := ;
set PE[68] := ;
set PE[69] := ;
set PE[70] := ;
setPE[71] := ;
set PE[72] := 173 174 175 176 177 178 179 180 181 182 183
184 185 186 187 188 189 190 191 192 193 194 195
196 197 198 199 200 201 202 203 204 205 206 207
208;
set PE[73] := ;
set PE[74] := ;
set PE[75] := ;
set PE[76] := ;
set PE[77] := ;
set PE[78] := ;
set PE[79] := ;
set PE[80] := ;
setPE[81] := ;
set PE[82] := ;
set PE[83] := ;
set PE[84] := ;
set PE[85] := ;
set PE[86] := 209 210 211 212 213 214 215 216 217 218 219
220 221 222 223 224 225 226 227 228 229 230 231
232 233 234 235 236 237 238 239 240 241 242 243
244;
set PE[87] := ;
set PE[88] ;= ;
set PE[89] := ;
set PE[90] := ;

```
setPE[  91] :=  ;
set PE[  92] :=  ;
set PE[  93] :=  ;
set PE[  94] :=  ;
set PE[  95] :=  ;
set PE[  96] :=  ;
set PE[  97] :=  ;
set PE[  98] :=  ;
set PE[  99] :=  ;
setPE[100]:=  ;
setPE[ 101] :=  ;
setPE[102] :=  ;
setPE[103] :=  ;
setPE[104] : =  ;
setPE[105] :=  ;
setPE[106]:=  245   246   247   248   249   250   251   252   253   254   255
              256   257   258   259   260   261   262   263   264   265   266   267
              268   269   270   271   272   273   274   275   276   277   278   279;
setPE[107]:=  ;
setPE[108] :=  ;
setPE[109] :=  ;
setPE[110]:=  ;
setPE[111] :=  ;
setPE[112]:=  ;
setPE[113] :=  ;
setPE[114] :=  ;
setPE[115] :=  ;
setPE[116]:=  37   40   43   46   60   175   179   182   185   198   211
              215   218   221   234   245   248   250   251   254   257   260   262
              263   265   268   272   273   274   276   278   279   281   284   286
              287   290   293   296   298   299   301   304   308   309   310   312
              314   315   317   320   322   323   326   329   332   334   335   337
              340   344   345   346   348   350   351   389   392   395   398   401
              404   407   419   422   429   433   436   439   453   465   468   471
              474   477   480   483   495   499   503   506   508   509   512   515
              518   520   521   523   527   531   532   533   535   537   538;
setPE[117] :=  ;
setPE[118] :=  ;
setPE[ 119] :=  ;
setPE[120]:=  13   14   15   49   59   51   83   84   85   117   118
              119   151   152   153   187   188   189   223   224   225   302   338
              366   367   368   408   409   410   442   443   444   484   485   486
              524;
setPE[121]:=  19   20   21   55   56   57   89   90   91   123   124
              125   157   158   159   193   194   195   229   230   231   267   303
              372   373   374   414   415   416   448   449   450   490   491   492
              526;
setPE[122] :=  ;
setPE[123]:=  ;
setPE[124]:=  ;
setPE[125]:=  3   9   12   25   28   31   63   67   74   79   82
              95   98   101   108   113   116   129   132   135   142   147   150
              163   166   169   201   205   237   241   246   247   249   252   253
              255   256   258   259   261   264   269   270   271   275   277   282
              283   285   288   289   291   292   294   295   297   300   305   306
              307   311   313   318   319   321   324   325   327   328   330   331
              333   336   341   342   343   347   349   356   362   365   377   380
              383   456   459   504   505   507   510   511   513   514   516   517
              519   522   528   529   530   534   536;
setPE[ 126] : =  32   33   34   68   69   70   102   103   104   136   137
              138   170   171   172   206   207   208   242   243   244   280   316
              352   384   385   386   424   425   426   460   461   462   500   501
              502;
setPE[127]:=  ;
setPE[128] : =  ;
setPE[129] :=  ;
setPE[ 130] :=  ;
setPE[131]:=  ;
setPE[132]:=  ;
setPE[133]:=  ;
setPE[134] :=  ;
setPE[135]:=  317   318   319   320   321   322   323   324   325   326   327
              328   329   330   331   332   333   334   335   336   337   338   339
              340   341   342   343   344   345   346   347   348   349   350   351
              352;
setPE[136] :=  ;
setPE[137]:=  ;
setPE[138]:=  ;
setPE[139]:=  ;
```

setPE[140] := ;
setPE[141]:= ;
setPE[142] := ;
setPE[143] := ;
setPE[144]:= 353 354 355 356 357 358 359 360 361 362 363
364 365 366 367 368 369 370 371 372 373 374 375
376 377 378 379 380 381 382 383 384 385 386:
setPE[145] : = ;
setPE[146] : = ;
set PE[147] := ;
set PE[148] := ;
set PE[149] : = ;
setPE[150]:= ;
setPE[151]:= ;
setPE[152] : = ;
setPE[153] •= ;
setPE[154]:= ;
setPE[155]:= ;
setPE[156] : = ;
setPE[157] := ;
setPE[158]:= ;
setPE[159]:= ;
setPE[160] := ;
setPE[161] := ;
setPE[162] •= ;
setPE[163]:= ;
setPE[164] := ;
setPE[165]:= ;
setPE[166] := ;
setPE[167] := 387 388 389 390 391 392 393 394 395 396 397
398 399 400 401 402 403 404 405 406 407 408 409
410 411 412 413 414 415 416 417 418 419 420 421
422 423 424 425 426;
setPE[168] := ;
setPE[169] := ;
setPE[170]:= 427 428 429 430 431 432 433 434 435 436 437
438 439 440 441 442 443 444 445 446 447 448 449
450 451 452 453 454 455 456 457 458 459 460 461
462;
setPE[171] := ;
setPE[172]:= ;
setPE[173]:= ;
setPE[174] := ;
setPE[175] := ;
setPE[176] : = ;
setPE[177] := ;
setPE[178]:= ;
setPE[179] := ;
setPE[180] := ;
setPE[181] : = ;
setPE[182] := ;
setPE[183] := ;
setPE[184] := 2 3 8 9 11 12 27 28 73 74 78
79 81 82 97 98 107 108 112 113 115 116 131
132 141 142 146 147 149 150 165 166 247 249 253
256 259 261 262 264 265 270 275 276 283 285 286
289 292 295 297 300 301 306 311 312 319 321 325
328 331 333 334 336 337 342 347 348 355 356 361
362 364 365 379 380 388 390 394 397 400 402 405
410 413 416 418 420 426 464 466 470 473 476 478
481 486 489 492 494 497 502 505 507 511 514 517
519 520 522 523 529 534 535:
setPE[185]: = ;
setPE[186] := 36 39 42 45 51 54 57 59 70 174 178
181 184 189 192 195 197 208 210 214 217 220 225
228 231 233 244 246 252 255 258 269 282 288 291
294 305 318 324 327 330 341 387 391 393 396 399
403 406 409 412 415 417 421 425 428 432 435 438
444 447 450 452 462 463 467 469 472 475 479 482
485 488 491 493 498 501 504 510 513 516 528;
setPE[187] := ;
setPE[188] := ;
setPE[189] := ;
setPE[190] := ;
setPE[191] := 14 15 17 18 20 21 33 34 37 40 43
46 50 53 56 60 69 84 85 87 88 90 91
103 104 118 119 121 122 124 125 137 138 152 153
155 156 158 159 171 172 175 179 182 185 188 191
194 198 207 208 211 215 218 221 224 227 230 234

243 367 368 370 371 373 374 385 386 389 392 395
398 401 404 407 408 411 414 419 422 424 429 433
436 439 443 446 449 453 461 465 468 471 474 477
480 483 484 487 490;
setPE[192]:= ;
setPE[193]:= ;
setPE[194] := 23 24 25 61 62 63 93 94 95 127 128
129 161 162 163 199 200 201 235 236 237 271 272
273 307 308 309 343 344 345 375 376 377 454 455
456 496 530 531 532;
setPE[195]:= ;
setPE[196]:= ;
setPE[197]:= ;
setPE[198] •= ;
setPE[199] := ;
setPE[200] := ;
setPE[201]:= ;
set PE[202] := ;
setPE[203]:= ;
set PE[204] := ;
set PE[205] := 503 504 505 506 507 508 509 510 511 512 513
514 515 516 517 518 519 520 521 522 523 524 525
526 527 528 529 530 531 532 533 534 535 536 537
538;
set PE[206] : = ;
set PE[207] := ;
set PE[208] := ;
setPE[209]:= ;
setPE[210]:= ;

setEQUIP :=T1;

param demand :=
[N1.N2] 0 [N1.N3] 0 [N1.N4] 255000 [N1.N5] 0 [N1.N6] 0 [N1.N7] 0 [N1.N8] 0 [N1.N9] 255000 [N1.N10] 255000 [N1.N11] 0 [N1.N12] 0 [N1.N13] 0 [N1.N14] 0 [N1.N15] 0 [N2.N1] 0 [N2.N3] 255000 [N2.N4] 255000 [N2.N5] 0 [N2.N6] 0 [N2.N7] 0 [N2.N8] 0 [N2.N9] 255000 [N2.N10] 255000 [N2.N11] 255000 [N2.N12] 0 [N2.N13] 0 [N2.N14] 0 [N2.N15] 255000 [N3.N1] 0 [N3.N2] 255000 [N3.N4] 255000 [N3.N5] 255000 [N3.N6] 0 [N3.N7] 0 [N3.N8] 0 [N3.N9] 0 [N3.N10] 255000 [N3.N11] 255000 [N3.N12] 255000 [N3.N13] 0 [N3.N14] 0 [N3.N15] 0 [N4.N1] 255000 [N4.N2] 255000 [N4.N3] 255000 [N4.N5] 0 [N4.N6] 255000 [N4.N7] 0 [N4.N8] 0 [N4.N9] 0 [N4.N10] 255000 [N4.N11] 255000 [N4.N12] 0 [N4.N13] 0 [N4.N14] 0 [N4.N15] 0 [N5.N1] 0 [N5.N2] 0 [N5.N3] 255000 [N5.N4] 0 [N5.N6] 0 [N5.N7] 0 [N5.N8] 0 [N5.N9] 255000 [N5.N10] 0 [N5,N11]255000 [N5.N12] 255000 [N5.N13J 255000 [N5.N14] 255000 [N5.N15] 0 [N6.N1] 0 [N6.N2] 0 [N6.N3] 0 [N6.N4] 255000 [N6.N5] 0 [N6.N7] 255000 [N6.N8] 0 [N6.N9] 255000 [N6.N10] 0 [N6.N11] 0 [N6.N12] 0 [N6.N13] 255000 [N6.N14] 0 [N6.N15] 255000 [N7.N1] 0 [N7.N2] 0 [N7.N3] 0 [N7.N4] 0 [N7.N5] 0 [N7.N6] 255000 [N7.N8] 0 [N7.N9] 255000 [N7.N10] 0 [N7.N11] 255000 [N7.N12] 255000 [N7.N13] 255000 [N7.N14] 255000 [N7.N15] 0 [N8.N1] 0 [N8.N2] 0 [N8.N3] 0 [N8.N4] 0 [N8.N5] 0 [N8.N6] 0 [N8.N7] 0 [N8.N9] 0 [N8.N10] 0 [N8.N11] 0 [N8,N12] 255000 [N8.N13] 255000 [N8.N14] 0 [N8.N15] 255000 [N9.N1] 255000 [N9.N2] 255000 [N9.N3] 0 [N9.N4] 0 [N9.N5] 255000 [N9.N6] 255000 [N9.N7] 255000 [N9.N8] 0 [N9.N10] 0 [N9.N11] 0 [N9.N12] 255000 [N9.N13] 255000 [N9.N14] 255000 [N9.N15] 0 [N10.N1] 255000 [N10.N2] 255000 [N10.N3] 255000 [N10.N4] 255000 [N10.N5] 0 [N10.N6] 0 [N10.N7] 0 [N10.N8] 0 [N10.N9] 0 [N10.N11] 255000 [N10.N12] 0 [N10.N13] 255000 [N10.N14] 0 [N10.N15] 255000 [N11.N1] 0 [N11.N2] 255000 [N11.N3] 255000 [N11.N4] 255000 [N11.N5] 255000 [N11.N6] 0 [N11.N7] 255000 [N11.N8] 0 [N11.N9] 0 [N11.N10] 255000 [N11.N12] 0 [N11.N13] 0 [N11.N14] 255000 [N11.N15] 0 [N12.N1] 0 [N12.N2] 0 [N12.N3] 255000 [N12.N4] 0 [N12.N5] 255000 [N12.N6] 0 [N12.N7] 255000 [N12.N8] 255000 [N12.N9] 255000 [N12.N10] 0 [N12.N11] 0 [N12.N13] 255000 [N12.N14] 255000 [N12.N15] 0 [N13.N1] 0 [N13.N2] 0 [N13.N3] 0 [N13.N4] 0 [N13.N5] 255000 [N13.N6] 255000 [N13.N7] 255000 [N13.N8] 255000 [N13.N9] 255000 [N13.N10] 255000 [N13.N11] 0 [N13.N12] 255000 [N13.N14] 255000 [N13.N15] 0 [N14.N1] 0 [N14.N2] 0 [N14.N3] 0 [N14.N4] 0 [N14.N5] 255000 [N14.N6] 0 [N14.N7] 255000 [N14.N8] 0 [N14.N9] 255000 [N14.N10] 0 [N14.N11] 255000 [N14.N12] 255000 [N14.N13] 255000 [N14.N15] 0 [N15.N1] 0 [N15.N2] 255000 [N15.N3] 0 [N15.N4] 0 [N15.N5] 0 [N15.N6] 255000 [N15.N7] 0 [N15.N8] 255000 [N15.N9] 0 [N15.N10] 255000 [N15.N11] 0 [N15.N12] 0 [N15.N13] 0 [N15.N14] 0;
param Capacity := T1 1540000,

Anexo D

```
setN := N1 N2 N3 N4 N5 N6 N7 N8 N9 N10 N11 N12 N13 N14 N15;

setL := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 192021 222324252627282930 31 32333435363738394041 4243444546
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105;

setE := 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189
190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210;

setD := (N1,N2)(N1,N3)(N1,N4)(N1,N5) (N1.N6) (N1.N7) (N1.N8) (N1.N9) (N1.N10) (N1.N11) (N1.N12) (N1.N13) (N1.N14) (N1.N15)
(N2.N1) (N2.N3) (N2.N4) (N2.N5) (N2.N6) (N2.N7) (N2.N8) (N2.N9) (N2.N10) (N2.N11) (N2.N12) (N2.N13) (N2.N14) (N2.N15) (N3.N1)
(N3.N2) (N3.N4) (N3.N5) (N3.N6) (N3.N7) (N3.N8) (N3.N9) (N3.N10) (N3.N11) (N3.N12) (N3.N13) (N3.N14) (N3.N15) (N4.N1) (N4.N2)
(N4.N3) (N4.N5) (N4.N6) (N4.N7) (N4.N8) (N4.N9) (N4.N10) (N4.N11) (N4.N12) (N4.N13) (N4.N14) (N4.N15) (N5.N1) (N5.N2) (N5.N3)
(N5.N4) (N5.N6) (N5.N7) (N5.N8) (N5.N9) (N5.N10) (N5.N11) (N5.N12) (N5.N13) (N5.N14) (N5.N15) (N6.N1) (N6.N2) (N6.N3) (N6.N4)
(N6.N5) (N6.N7) (N6.N8) (N6.N9) (N6.N10) (N6.N11) (N6.N12) (N6.N13) (N6.N14) (N6.N15) (N7.N1) (N7.N2) (N7.N3) (N7.N4) (N7.N5)
(N7.N6) (N7.N8) (N7.N9) (N7.N10) (N7.N11) (N7.N12) (N7.N13) (N7.N14) (N7.N15) (N8.N1) (N8.N2) (N8.N3) (N8.N4) (N8,N5) (N8.N6)
(N8,N7)(N8,N9)(N8,N10)(N8,N11) (N8.N12) (N8.N13) (N8.N14) (N8.N15) (N9.N1) (N9.N2) (N9.N3) (N9.N4) (N9.N5) (N9,N6) (N9.N7)
(N9.N8) (N9.N10) (N9.N11) (N9.N12) (N9,N13) (N9.N14) (N9.N15) (N10,N1) (N10.N2) (N10.N3) (N10.N4) (N10.N5) (N10.N6) (N10.N7)
(N10.N8) (N10.N9) (N10.N11) (N10.N12) (N10.N13) (N10.N14) (N10,N15) (N11.N1) (N11.N2) (N11,N3) (N11.N4) (N11.N5) (N11.N6)
(N11.N7) (N11.N8) (N11,N9) (N11.N10) (N11.N12) (N11.N13) (N11,N14) (N11,N15) (N12,N1) (N12.N2) (N12.N3) (N12.N4) (N12.N5)
(N12.N6) (N12.N7) (N12.N8) (N12.N9) (N12.N10) (N12.N11) (N12.N13) (N12.N14) (N12.N15) (N13.N1) (N13.N2) (N13.N3) (N13.N4)
(N13.N5) (N13.N6) (N13.N7) (N13.N8) (N13.N9) (N13.N10) (N13.N11) (N13.N12) (N13.N14) (N13.N15) (N14.N1) (N14.N2) (N14.N3)
(N14.N4) (N14,N5) (N14,N6) (N14,N7) (N14,N8) (N14.N9) (N14.N10) (N14.N11) (N14.N12) (N14.N13) (N14.N15) (N15.N1) (N15,N2)
(N15,N3) (N15,N4) (N15,N5) (N15,N6) (N15.N7) (N15.N8) (N15.N9) (N15.N10) (N15.N11) (N15.N12) (N15.N13) (N15.N14);

setP:= 1 2 34 5 6 7 89 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125
126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157
158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189
190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221
222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253
254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285
286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317
318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349
350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381
382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413
414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445
446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477
478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509
510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541
542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571
572 573;

set PD[N1,N2] := 1 2 3;
setPD[N1,N4] := 7 8 9;
setPD[N1,N6] := 11 12 13;
set PD[N1,N8] := 17 18 19;
setPD[N1,N10] := 23 24 25;
setPD[N1,N12]:=29;
setPD[N1,N14] := 33;
set PD[N2,N1] := 37 38 39;
set PD[N2,N4] := 44 45 46 47;
set PD[N2,N6] := 51;
set PD[N2,N8] := 53 54 55;
setPD[N2,N10] := 59 60 61;
set PD[N2,N12] := 66 67 68;
set PD[N2,N14] := 70 71 72;
set PD[N3,N1] := 76 77 78;
set PD[N3,N4] := 83;
set PD[N3,N6] := 87 88 89 90;
set PD[N3,N8] := 95 96 97;
set PD[N3,N10] := 101 102 103;
setPD[N3,N12] := 105 106 107;
set PD[N3,N14] := 112 113 114;
setPD[N4,N1] := 118 119 120;
set PD[N4,N3] := 125;

set PD[N1.N3] ;= 4 5 6:
setPD[N1,N5] := 10;
setPD[N1,N7] := 14 15 16;
set PD[N1,N9] := 20 21 22;
set PD[N1,N11] := 26 27 28;
set PD[N1,N13] := 30 31 32;
set PD[N1.N15] := 34 35 36;
set PD[N2,N3] := 40 41 42 43;
set PD[N2.N5] := 48 49 50;
set PD[N2,N7] := 52;
set PD[N2,N9] ;= 56 57 58;
set PD[N2,N11] := 62 63 64 65;
set PD[N2,N13] ;= 69;
set PD[N2,N15] := 73 74 75;
set PD[N3,N2] ;= 79 80 81 82;
set PD[N3,N5] ;= 84 85 86;
set PD[N3,N7] ;= 91 92 93 94;
set PD[N3,N9] := 98 99 100;
set PD[N3,N11] := 104;
set PD[N3,N13] := 108 109 110 111;
set PD[N3,N15] := 115 116 117;
set PD[N4,N2] := 121 122 123 124;
set PD[N4,N5] .= 126 127 128;
```

```
set PD[N4,N6J := 129 130 131 132 133,
set PD[N4,N8] := 139 140 141;
set PD[N4,N10] := 145 146 147;
setPD[N4,N12] := 149 150 151:
set PD[N4,N14] := 157 158 159;
setPD[N5,N1] := 163;
set PD[N5,N3] := 167 168 169;
set PD[N5,N6] ;= 173 174 175;
setPD[N5.N8J ;= 179 180 181;
set PD[N5,N10J := 185 186 187;
setPD[N5,N12] := 191;
setPD[N5,N14] := 195;
setPD[N6,N1] := 199 200 201;
set PD[N6.N3] := 203 204 205 206;
setPD[N6,N5]:=211 212 213;
setPD[N6,N8]:=215 216 217;
set PD[N6,N10] := 221 222 223;
set PD[N6,N12] := 228 229 230;
set PD[N6,N14] := 232 233 234;
set PD[N7,N1] := 238 239 240;
set PD[N7,N3] := 242 243 244 245;
set PD[N7,N5] := 249 250 251;
set PD[N7,N8] := 253 254 255 256;
set PD[N7,N10] := 260 261 262;
set PD[N7,N12J := 267 268 269;
set PD[N7,N14] := 271 272 273;
set PD[N8,N1] := 277 278 279;
set PD[N8,N3] := 283 284 285;
set PD[N8,N5] := 289 290 291;
set PD[N8,N7] := 295 296 297;
setPD[N8,N10] := 299;
set PD[N8,N12J := 303 304 305:
setPD[N8,N14] := 309 310 311;
setPD[N9,N1] :=313 314 315;
set PD[N9,N3] := 319 320 321:
set PD[N9,N5] := 325 326 327;
set PD[N9,N7] := 331 332 333;
setPD[N9,N10] := 335;
set PD[N9,N12] := 339 340 341;
set PD[N9,N14] := 345 346 347;
set PD[N10,N1] := 349 350 351;
set PD[N10,N3] := 355 356 357;
set PD[N10,N5J := 361 362 363;
set PD[N10,N7] := 367 368 369;
setPD[N10,N9] :=371;
set PD[N10.N12] := 375 376 377;
set PD[N10,N14] := 381 382 383;
setPD[N11,N1] :=385 386 387;
setPD[N11,N3] := 392;
setPD[N11,N5] := 394 395 396;
setPD[N11,N7] := 401 402 403 404;
set PD[N11.N9] := 408 409 410;
setPD[N11,N12] :=414415416;
setPD[N11,N14] := 421 422 423;
setPD[N12,N1] :=427;
set PD[N12,N3] := 431 432 433;
set PD[N12,N5] :=437;
set PD[N12,N7] := 441 442 443;
set PD[N12,N9] := 447 448 449:
set PD[N12,N11] := 453 454 455;
setPD[N12,N14] :=459;
setPD[N13.N1] :=463 464 465;
set PD[N13,N3] := 467 468 469 470;
set PD[N13,N5J := 475 476 477;
setPD[N13,N7] := 479;
setPD[N13,N9] := 483 484 485;
set PD[N13,N11] :=489 490 491 492;
set PD[N13,N14] := 496 497 498;
setPD[N14,N1] := 502;
set PD[N14,N3] := 506 507 508;
setPD[N14,N5] := 512;
set PD[N14,N7] := 516 517 518;
set PD[N14,N9J := 522 523 524;
setPD[N14,N11] := 528 529 530;
set PD[N14,N13] := 532 533 534;
set PD[N15,N1] := 538 539 540;
set PD[N15,N3] := 544 545 546;
set PD[N15,N5J := 550 551 552;
set PD[N15.N7] := 556 557 558;
set PD[N4.N7] := 134 135 136 137 138;
set PD[N4,N9] := 142 143 144;
set PD[N4,N11] := 148;
setPD[N4,N13] := 152 153 154 155 156;
set PD[N4.N15] := 160 161 162;
set PD[N5,N2] := 164 165 166;
set PD[N5.N4) := 170 171 172;
set PD[N5,N7] := 176 177 178;
set PD[N5.N9] := 182 183 184;
setPD[N5,N11] := 188 189 190;
set PD[N5,N13] := 192 193 194;
set PD[N5.N15] := 196 197 198;
setPD[N6,N2J := 202;
set PD[N6,N4] := 207 208 209 210;
setPD[N6,N7] := 214;
set PD[N6.N9] := 218 219 220;
set PD[N6.N11] := 224 225 226 227;
setPD[N6.N13] :=231;
set PD[N6,N15] := 235 236 237;
setPD[N7.N2] :=241;
set PD[N7,N4] := 246 247 248;
set PD[N7,N6] := 252;
set PD[N7,N9] := 257 258 259;
setPD[N7,N11J :=263 264 265 266;
setPD[N7,N13J:=270;
set PD[N7,N15] := 274 275 276;
set PD[N8.N2] := 280 281 282;
set PD[N8,N4] := 286 287 288;
set PD[N8.N6) := 292 293 294;
set PD[N8,N9] := 298;
setPD[N8,N11] := 300 301 302;
set PD[N8,N13] := 306 307 308;
setPD[N8,N15]:=312;
setPD[N9,N2]:=316 317 318;
set PD[N9,N4] := 322 323 324;
set PD[N9,N6] := 328 329 330;
set PD[N9,N8] := 334;
setPD[N9,N11] := 336 337 338;
set PD[N9,N13] := 342 343 344;
setPD[N9,N15]:=348;
set PD[N10,N2].:= 352 353 354;
set PD[N10.N4] := 358 359 360;
set PD[N10,N6] := 364 365 366;
setPD[N10.N8] := 370;
set PD[N10.N11] := 372 373 374;
set PD[N10,N13] := 378 379 380;
setPD[N10.N15] := 384;
setPD[N11.N2] := 388 389 390 391;
setPD[N11.N4]:= 393;
set PD[N11.N6] := 397 398 399 400;
set PD[N11 ,N8] := 405 406 407;
set PD[N11,N10] := 411 412 413;
set PD[N11,N13] := 417 418 419 420:
set PD[N11,N15] := 424 425 426,
set PD[N12,N2] := 428 429 430;
set PD[N12,N4) := 434 435 436;
set PD[N12,N6] := 438 439 440;
set PD[N12.N8] := 444 445 446;
set PD[N12,N10] := 450 451 452;
setPD[N12,N13] := 456 457 458;
setPD[N12.N15J := 460 461 462;
setPD[N13,N2J :=466:
setPD[N13,N4] :=471 472 473 474;
setPD[N13.N6]:=478;
set PD[N13,N8]:= 480 481 482;
setPD[N13,N10] := 486 487 488;
set PD[N13,N12] := 493 494 495;
setPD[N13,N15] := 499 500 501;
set PD[N14.N2] := 503 504 505;
set PD[N14,N4] :=509510511;
set PD[N14,N6] := 513 514 515;
set PD[N14.N8] := 519 520 521;
set PD[N14,N10] := 525 526 527;
setPD[N14.N12] ;=531;
set PD[N14.N15] := 535 536 537;
set PD[N15,N2] := 541 542 543;
set PD[N15,N4] := 547 548 549;
set PD[N15,N6] := 553 554 555;
setPD[N15,N8]:=559;
```

```
setPD[N15,N9] := 560;                                   setPD[N15,N10] := 561;
setPD[N15,N11] := 562 563 564;                          set PD[N15,N12] := 565 566 567;
set PD[N15.N13] := 568 569 570;                         set PD[N15.N14] := 571 572 573;

setPE[   1]:=          ;
set PE[  2] :=         ;
set PE[  3] ;=         ;
set PE[  4] ;=         ;
set PE[  5] ;=         ;
set PE[  6] :=         ;
set PE[  7] :=         ;
set PE[  8] :=         ;
set PE[  9] :=         ;
setPE[  10] :=         ;
setPE[  11] :=         ;
setPE[  12] :=         ;
setPE[  13] :=         1       2       3       4       5       6       7       8       9       10      11
               12      13      14      15      16      17      18      19      20      21      22      23
               24      25      26      27      28      29      30      31      32      33      34      35
               36;

setPE[  14] :=         ;
setPE[  15] ;=         ;
setPE[  16] :=         ;
setPE[  17] :=         ;
setPE[   18]:=         ;
setPE[  19] :=         11      12      13      51      87      88      89      90      129     130     131
               132     133     253     292     293     294     328     329     330     364     365     366
               397     398     399     400     438     439     440     478     513     514     515     553
               554     555;
set PE[  20] :=        14      15      16      52      91      92      93      94      134     135     136
               137     138     214     295     296     297     331     332     333     367     368     369
               401     402     403     404     441     442     443     479     516     517     518     556
               557     558;
setPE[  21] :=         ;
set PE[  22] :=        6       9       22      24      28      35      38      39      41      42      45
               47      49      50      53      56      59      62      65      67      68      71      72
               73      97      100     103     117     144     147     162     172     181     184     186
               190     197     200     201     204     205     208     210     212     213     215     218
               221     224     227     229     230     233     234     235     239     240     243     244
               247     249     251     252     254     257     260     263     266     268     272     273
               274     407     410     413     426     433     436     446     449     451     455     461
               464     465     468     469     472     474     476     477     480     483     486     489
               492     494     495     497     498     499     508     511     521     524     526     530
               536;
set PE[  23] :=        ;
set PE[  24] :=        ;
set PE[  25] :=        ;
set PE[  26] :=        30      31      32      69      108     109     110     111     152     153     154
               155     156     192     193     194     270     306     307     308     342     343     344
               378     379     380     456     457     458     532     533     534     568     569     570;
set PE[  27] :=        37      43      44      46      48      54      55      57      58      60      61
               63      64      66      70      74      75      78      86      107     114     120     128
               151     159     199     203     206     207     209     211     216     217     219     220
               222     223     225     226     228     232     236     237     238     242     245     246
               248     250     255     256     258     259     261     262     264     265     267     271
               275     276     278     285     288     290     302     304     310     314     321     324
               326     338     340     346     350     357     360     362     374     376     382     387
               396     416     423     463     467     470     471     473     475     481     482     484
               485     487     488     490     491     493     496     500     501     539     546     549
               551     564     566     572;
set PE[  28] :=        ;
set PE[  29] :=        ;
set PE[  30] :=        ;
setPE[  31] :=         76      77      78      79      80      81      82      83      84      85
               86      87      88      89      90      91      92      93      94      95      96      97
               98      99      100     101     102     103     104     105     106     107     108     109
               110     111     112     113     114     115     116     117;
set PE[  32] :=        ;
set PE[  33] :=        ;
set PE[  34] :=        ;
set PE[  35] :=        ;
set PE[  36] :=        ;
set PE[  37] :=        ;
set PE[  38] :=        ;
set PE[  39] :=        ;
set PE[  40] :=        ;
set PE[  41] :=        ;
set PE[  42] ;=        ;
```

set PE[43] :=
set PE[44] :=
set PE[45] : = 4 5 6 41 42 43 125 167 168 169 203
204 205 206 242 243 244 245 283 284 285 319 320
321 355 356 357 392 431 432 433 467 468 469 470
506 507 508 544 545 546;
set PE[46] :=
set PE[47] :=
set PE[48] :=
set PE[49] :=
set PE[50] := 3 13 16 18 21 25 32 36 55 58 61
75 77 78 79 82 85 86 87 89 91 93 95
98 101 106 107 108 110 113 114 115 119 120 122
123 127 128 130 132 135 137 139 142 145 150 151
153 155 158 159 160 166 175 178 180 183 187 194
198 217 221 237 262 276 386 387 388 391 395 396
397 399 401 403 405 408 411 415 416 417 419 422
423 424 430 440 443 445 448 452 462 482 485 488
501 505 518 520 523 527 534 537;
set PE[51] :=
set PE[52] := 26 27 28 63 64 65 104 148 188 189 190
224 225 226 227 263 264 265 266 300 301 302 336
337 338 372 373 374 453 454 455 489 490 491 492
528 529 530 562 563 564;
set PE[53] :=
set PE[54] :=
set PE[55] := 39 50 68 72 76 84 88 90 92 94 96
97 99 100 102 103 105 109 111 116 117 118 121
124 126 129 131 133 134 136 138 140 141 143 144
146 147 149 152 154 156 157 161 162 201 213 230
234 240 252 273 279 282 291 294 297 305 308 311
315 318 327 330 333 341 344 347 351 354 363 366
369 377 380 383 385 389 390 394 398 400 402 404
406 407 409 410 412 413 414 418 420 421 425 426
465 477 498 540 543 552 555 558 567 570 573;
set PE[56] :=
set PE[57] :=
set PE[58] :=
set PE[59] :=
set PE[60] :=
setPE[61] :=
set PE[62] :=
set PE[63] :=
set PE[64] :=
set PE[65] :=
set PE[66] :=
set PE[67] :=
set PE[68] :=
set PE[69] := 163 164 165 166 167 168 169 170 171 172 173
174 175 176 177 178 179 180 181 182 183 184 185
186 187 188 189 190 191 192 193 194 195 196 197
198;
set PE[70] :=
setPE[71] :=
set PE[72] := 199 200 201 202 203 204 205 206 207 208 209
210 211 212 213 214 215 216 217 218 219 220 221
222 223 224 225 226 227 228 229 230 231 232 233
234 235 236 237;
set PE[73] :=
set PE[74] :=
set PE[75] :=
set PE[76] :=
set PE[77] :=
set PE[78] :=
set PE[79] :=
set PE[80] :=
set PE[81] :=
set PE[82] :=
set PE[83] :=
set PE[84] :=
set PE[85] :=
setPE[86]:= 238 239 240 241 242 243 244 245 246 247 248
249 250 251 252 253 254 255 256 257 258 259 260
261 262 263 264 265 266 267 268 269 270 271 272
273 274 275 276;
set PE[87] :=
set PE[88] :=
set PE[89] :=
set PE[90] :=

setPE[91] :=
set PE[92] :=
set PE[93] :=
set PE[94] :=
set PE[95] :=
set PE[96] :=
set PE[97] :=
set PE[98] :=
set PE[99] :=
setPE[100]:=
set PE[101] :=
setPE[102] :=
setPE[103]:=
setPE[104]:=
setPE[105]:=
setPE[106]:= 277 278 279 280 281 282 283 284 285 286 287
288 289 290 291 292 293 294 295 296 297 298 299
300 301 302 303 304 305 306 307 308 309 310 311
312;
setPE[107]: =
setPE[108]:=
setPE[109] :=
setPE[11O]:=
set PE[111] :=
setPE[112]:=
setPE[113]:=
setPE[114]:= 2 3 12 13 15 16 31 32 78 79 81
86 87 **90** 91 94 107 108 111 114 120 122 124
128 130 133 135 138 151 153 156 159 165 166 174
175 177 178 193 278 280 285 288 290 292 295 302
304 306 310 314 316 321 324 326 328 331 338 340
342 346 350 352 357 360 362 364 374 376 378 382
387 388 390 396 397 401 404 416 417 423 429 430
439 440 442 443 457 458 504 505 514 515 517 518
533 539 541 546 549 551 553 556 564 566 568 572;
setPE[115]:=
setPE[116]:= 5 6 8 9 27 28 39 41 43 45 46
50 62 64 68 72 168 169 171 172 189 190 201
204 206 208 209 213 224 226 230 234 240 243 245
247 248 252 263 265 269 273 279 282 283 286 291
294 300 305 308 311 315 318 319 322 327 330 333
336 341 344 347 351 354 355 358 363 366 372 377
380 383 432 433 435 436 454 455 465 468 470 472
473 477 489 491 495 498 507 508 510 511 529 530
540 543 544 547 552 555 558 562 567 570 573;
setPE[117]:=
setPE[118]:=
set PE[119] :=
setPE[120] := 17 18 19 53 54 55 95 96 97 139 140
141 179 180 181 215 216 217 254 255 256 334 370
405 406 407 444 445 446 480 481 482 519 520 521
559;
setPE[121]:= 23 24 25 59 60 61 101 102 103 145 146
147 185 186 187 221 222 223 260 261 262 299 411
412 413 450 451 452 486 487 488 525 526 527 561;
setPE[122]: =
setPE[123] :=
setPE[124]:=
setPE[125]: = 38 42 47 65 67 71 77 82 85 89 93
106 110 113 119 123 127 132 137 150 155 158 200
205 210 227 229 233 239 244 249 266 268 272 277
281 284 287 289 293 296 301 307 309 313 317 320
323 325 329 332 337 339 343 345 349 353 356 359
361 365 368 373 375 379 381 386 391 395 399 403
419 422 464 469 474 492 494 497 538 542 545 548
550 554 557 563 565 569 571;
setPE[126]:= 34 35 36 73 74 75 115 116 117 160 161
162 196 197 198 235 236 237 274 275 276 312 348
384 424 425 426 460 461 462 499 500 501 535 536
537;
setPE[127]:=
setPE[128]:=
setPE[129]:=
setPE[130]: =
setPE[131]:=
setPE[132] :=
setPE[133] :=
setPE[134] :=

setPE[135]:= 349 350 351 352 353 354 355 356 357 358 359
360 361 362 363 364 365 366 367 368 369 370 371
372 373 374 375 376 377 378 379 380 381 382 383
384;
setPE[136] : = ;
setPE[137]:= ;
setPE[138]:= ;
setPE[139]:= ;
set PE[140] := ;
setPE[141] := ;
set PE[142] := ;
setPE[143]:= ;
set PE[144] := 385 386 387 388 389 390 391 392 393 394 395
396 397 398 399 400 401 402 403 404 405 406 407
408 409 410 411 412 413 414 415 416 417 418 419
420 421 422 423 424 425 426;
setPE[145]:= ;
set PE[146] := ;
setPE[147]:= ;
set PE[148] := ;
setPE[149] := ;
setPE[150] := ;
setPE[151]:= ;
setPE[152]:= ;
setPE[153]:= ;
setPE[154]:= ;
setPE[155]:= ;
setPE[156]:= ;
setPE[157] := ;
setPE[158]:= ;
setPE[159]:= ;
setPE[160] := ;
setPE[161] := ;
setPE[162] := ;
setPE[163]:= ;
setPE[164] := ;
setPE[165]:= ;
setPE[166]:= ;
setPE[167]:= 427 428 429 430 431 432 433 434 435 436 437
438 439 440 441 442 443 444 445 446 447 448 449
450 451 452 453 454 455 456 457 458 459 460 461
462;
setPE[168]:= ;
setPE[169]:= ;
setPE[170]:= 463 464 465 466 467 468 469 470 471 472 473
474 475 476 477 478 479 480 481 482 483 484 485
486 487 488 489 490 491 492 493 494 495 496 497
498 499 500 501;
setPE[171]:= ;
setPE[172]:= ;
setPE[173]:= ;
setPE[174] := ;
setPE[175]:= ;
setPE[176]:= ;
setPE[177]:= ;
setPE[178]:= ;
setPE[179]:= ;
setPE[180]:= ;
setPE[181]: = ;
setPE[182]:= ;
setPE[183]:= 37 38 39 76 77 78 118 119 120 163 199
200 201 238 239 240 277 278 279 313 314 315 349
350 351 385 286 387 427 463 464 465 502 538 539
540;
setPE[184]:= 1 6 9 11 14 19 22 24 28 30 35
80 82 88 89 92 93 97 100 103 109 110 117
121 123 129 131 132 134 136 137 141 144 147 152
154 155 162 164 169 172 173 176 181 184 186 190
192 197 281 282 294 296 297 307 308 317 318 329
330 332 333 343 344 353 354 365 366 368 369 379
380 389 391 398 399 402 403 407 410 413 418 419
428 433 436 438 441 446 449 451 455 456 461 503
508 511 513 516 521 524 526 530 532 536 542 543
554 555 557 558 569 570;
setPE[185]:= ;
setPE[186]:= 3 4 7 13 16 18 21 25 26 32 36
40 42 44 55 58 61 63 65 75 166 167 170
175 178 180 183 187 188 194 198 203 205 207 210
217 220 223 225 227 237 242 244 246 249 256 259

262 264 266 276 284 285 287 288 301 302 320 321
323 324 337 338 356 357 359 360 373 374 430 431
434 440 443 445 448 452 453 458 462 467 469 471
474 482 485 488 490 492 501 505 506 509 515 518
520 523 527 534 537 545 546 548 549 563 564:
setPE[187]:= 10 48 49 50 84 85 86 126 127 128 211
212 213 250 251 252 289 290 291 348 361 362 363
394 395 396 437 475 476 477 512 550 551 552;
setPE[188] := ;
setPE[189] := ;
setPE[190] := ;
setPE[191] := 2 5 8 12 15 17 20 23 27 31 34
43 46 54 57 60 64 74 81 90 94 96 99
102 116 124 133 138 140 143 146 156 161 165 168
171 174 177 179 182 185 189 193 196 206 209 216
219 222 226 236 245 248 255 258 261 265 275 390
400 404 406 409 412 420 425 429 432 435 439 442
444 447 450 454 457 460 470 473 481 484 487 491
500 504 507 510 514 517 519 522 525 529 533;
setPE[192] := ;
setPE[193]:= ;
setPE[194] := ;
setPE[195] := ;
setPE[196]:= ;
setPE[197]:= ;
setPE[198]:= ;
setPE[199] := ;
set PE[200] := ;
setPE[201] := ;
setPE[202]:= ;
set PE[203J := ;
setPE[204] := ;
set PE[205] := 538 539 540 541 542 543 544 545 546 547 548
549 550 551 552 553 554 555 556 557 558 559 560
561 562 563 564 565 566 567 568 569 570 571 572
573;
setPE[206] := ;
setPE[207] := ;
set PE[208] := ;
setPE[209] := ;
setPE[210] := ;

#setEQUIP:=T1;

param demand :=
[N1,N2]0[N1,N3] 0 [N1.N4] 255000 [N1.N5] 0 [N1.N6] 0 [N1.N7] 0 [N1.N8] 0 [N1.N9] 255000 [N1.N10] 255000 [N1.N11] 0[N1,N12]0 [N1.N13] 0 [N1.N14] 0 [N1.N15] 0 [N2.N1] 0 [N2.N3] 255000 [N2.N4] 255000 [N2.N5] 0 [N2.N6] 0 [N2.N7] 0 [N2.N8] 0 [N2.N9] 255000 [N2.N10] 255000 [N2.N11] 255000 [N2.N12] 0 [N2.N13] 0 [N2.N14] 0 [N2.N15] 255000 [N3.N1] 0 [N3.N2] 255000 [N3.N4] 255000 [N3.N5] 255000 [N3.N6] 0 [N3.N7] 0 [N3.N8] 0 [N3.N9] 0 [N3.N10] 255000 [N3.N11] 255000 [N3.N12] 255000 [N3.N13] 0 [N3.N14] 0 [N3.N15] 0 [N4.N1] 255000 [N4.N2] 255000 [N4.N3] 255000 [N4.N5] 0 [N4.N6] 255000 [N4.N7] 0 [N4.N8] 0 [N4.N9] 0 [N4.N10] 255000 [N4.N11] 255000 [N4.N12] 0 [N4.N13] 0 [N4.N14] 0 [N4.N15] 0 [N5.N1] 0 [N5.N2] 0 [N5.N3] 255000 [N5.N4] 0 [N5.N6] 0 [N5.N7] 0 [N5.N8] 0 [N5.N9] 255000 [N5.N10] 0 [N5,N11]255000 [N5.N12] 255000 [N5.N13] 255000 [N5.N14] 255000 [N5.N15] 0 [N6.N1] 0 [N6.N2] 0 [N6.N3] 0 [N6.N4] 255000 [N6.N5] O [N6.N7] 255000 [N6.N8] O [N6.N9] 255000 [N6.N10] O [N6.N11] O [N6.N12] O [N6.N13] 255000 [N6,N14] O [N6.N15] 255000 [N7.N1] O [N7,N2] O [N7.N3] O [N7.N4] O [N7.N5] O [N7.N6] 255000 [N7,N8] O [N7.N9] 255000 [N7.N10] O [N7.N11] 255000 [N7.N12] 255000 [N7.N13] 255000 [N7.N14] 255000 [N7.N15] O [N8,N1] O [N8,N2] O [N8.N3] O [N8,N4] O [N8.N5] O [N8.N6] O [N8.N7] O [N8,N9] O [N8.N10] O [N8,N11] O [N8.N12] 255000 [N8.N13] 255000 [N8.N14] O [N8,N15] 255000 [N9.N1] 255000 [N9.N2] 255000 [N9.N3] O [N9.N4] O [N9.N5] 255000 [N9.N6] 255000 [N9.N7] 255000 [N9.N8] O [N9.N10] O [N9.N11] O [N9.N12] 255000 [N9.N13] 255000 [N9.N14] 255000 [N9.N15] O [N10.N1] 255000 [N10.N2] 255000 [N10.N3] 255000 [N10.N4] 255000 [N10.N5] O [N10.N6] O [N10.N7] O [N10.N8] O [N10.N9] O [N10.N11] 255000 [N10.N12] O [N10.N13] 255000 [N10.N14] O [N10.N15] 255000 [N11.N1] O [N11.N2] 255000 [N11.N3] 255000 [N11.N4] 255000 [N11.N5] 255000 [N11.N6] 0[N11,N7]255000[N11,N8]0[N11,N9]0[N11,N10]255000[N11.N12]0[N11,N13]0[N11.N14]255000[N11,N15]0[N12.N1]0[N12,N2]0 [N12.N3] 255000 [N12.N4] O [N12.N5] 255000 [N12.N6] O [N12.N7] 255000 [N12.N8] 255000 [N12.N9] 255000 [N12.N10] O [N12.N11] O [N12.N13] 255000 [N12.N14] 255000 [N12.N15] O [N13.N1] O [N13.N2] O [N13.N3] O [N13.N4] O [N13.N5] 255000 [N13.N6] 255000 [N13.N7] 255000 [N13.N8] 255000 [N13.N9] 255000 [N13.N10] 255000 [N13.N11] O [N13.N12] 255000 [N13.N14] 255000 [N13.N15] O [N14.N1] O [N14.N2] O [N14.N3] O [N14.N4] O [N14,N5] 255000 [N14.N6] O [N14.N7] 255000 [N14.N8] O [N14.N9] 255000 [N14.N10] O [N14.N11] 255000 [N14.N12] 255000 [N14.N13] 255000 [N14.N15] O [N15.N1] O [N15.N2] 255000 [N15.N3] O [N15.N4] O [N15.N5] O [N15.N6] 255000 [N15.N7] O [N15.N8] 255000 [N15,N9] O [N15.N10] 255000 [N15.N11] O [N15.N12] O [N15.N13] O [N15.N14] 0;

#param Capacity := T1 1540000;

Bibliografía

Arabas, J. y Kozdrowski, S. "Applying an Evolutionary Algorithm to Telecommunication Network Design", IEEE Transactions on Evolutionary Computation, Volume 5, Issue 4, Agosto. 2001 Páginas: 309 - 322.

Beltrán, J. "Ruteo Óptimo para el Diseño de Redes Mediante Algoritmos Genéticos", Tesis QA402.5 .B4 ITESM, 1996.

Bertocco, M., y cois. "Statistical Analysis of Measurements for Telecommunication-Network Troubleshooting", IEEE Instrumentation and Measurement, Volume 52, Issue 4, Agosto. 2003 Paginas: 1048 - 1053.

Bragg, W. "Which network design tool is right for you", IEEE IT Professional Volume 2, Issue 5, Sept.-Oct. 2000 Páginas: 23 - 32.

BT, "Digital Networking Economy". White paper. British Telecommunications pie, 2004.

Cahn, Robert, "Wide Área Network Design . Concepts and Tools for Optimizaron", Morgan Kaufmann Publishers Inc, 1998.The Morgan Kaufmann Series in Networking,

Chen, Kuang-Chien, et al. "A Resynthesis Approach for Network Optimization" IEEE 28th ACM/IEEE Design Automation Conference, June 17-21, 1991 Páginas: 458-463.

Chien, S., Takahashi, K. y Marjumder, S. "Performance of wavelength convertible multi-hop optical network under deflection routing with a delay-line optical buffer", IEEE Proceedings of the 2002 4th International Conference on Transparent Optical Networks, Volume 1, 21-25 April 2002 Páginas: 46 - 48.

Chinneck, John W. "Practical Optimization: a Gentle Introduction". Carleton University, Departamento de Ingeniería en computacional y de sistemas, 2001. URL: http://www.sce.carleton.ca/faculty/chinneck/po.html (Accesada en Marzo, 2005).

Dantzig, George Bernard y Mukud N. Thapa, "Linear programming 2: theory and extensions ", Springer Series in Operation Research. Edición 2003. ISBN: 0-387-98613-8.

DaSilva, Luiz A. y Midkiff, Scott F. "Network Design: Introduction to Graph Theory", Bradley Department of Electrical and Computer Engineering Virginia Polytechnic Institute and State University, 2002.

Decina, Maurizio. "Service and Technology Trends in the Next Decade", Proceedings International Zurich Seminar on Broadband Communications, Feb. 2000 Páginas: 109 2000.

Dongyun Zhou y Suresh Subramaniam, "Survivability in Optical Networks", IEEE Network, Volume 14, Issue 6, Nov.-Dic.2000 Páginas:16 - 23.

Evans, J. y Minieka, E. "Optimizaron Algorithms for Networks and Graphs". Editorial Marcel Dekker, segunda edición, ISBN 0-8247-8602-5, 1992.

Fourer, Robert. .AMPL: A Mathematical Programming Language., Managemet Science 36,1990.

Gangxiang, S., et al. "Designing WDM Optical Network for Reliability Routing Light Paths Efficiently for Path Protection". Sf.

Gentry, Sommer. "How to use AMPL/Cplex". Basado en "AMPL: A Modeling Language for Mathematical Programming, Fourer, R. 2003.

Gavish.B. .Backbone network design tools with economic tradeoff., ORSA Journal on Computing, 2:236-252, 1990.

Han, J., McMahon G. y Sugden S. "A Model for Nonlinear Problems in Optimal Design of Communication Networks", IEEE Proceedings. Ninth International Conference on Computer Communications and Networks, 2000. Oct 16-18. 2000 Páginas: 516 - 519.

Hernández, H. "PORT: Proceso para la Optimización de Redes de Telecomunicaciones", Tesis 045.62 TEC.121 ITESM, 1994.

Huang, Shin-Hsu y Wang, Chu-Liao. "An effective Floorplan-based Power Distribution Network Design Methodology under Reliability Constraints". IEEE International Symposium on Circuits and Systems, ISCAS 2002. Volume 1. Mayo 26-29, 2002. Páginas: I-353 -1-356.

Huang, Tzu-Lun y Lee, D.T. "A Distributed Multicast Routing Algorithm for Real-Time Applications in Wide Área Networks" , IEEE I-SPAN '02. Proceedings International Symposium on Parallel Architectures, Algorithms and Networks, 2002. Mayo 22-24. Páginas: 295 - 300.

Jeltsch F., Eric. 'Diseño y Análisis de Algoritmos con Java", Tutorial Universidad de La Serena Chile, 2004.

Kazovsky, Leonid G., Giok-Djan Khoe, M. Oskar van Deventer. "Future telecommunication networks: Major trend projections", IEEE Communications Magazine, Volume 36, Issue 11, Nov. 1998 Páginas: 122 - 127.

Kershenbaum, Aaron, Kermani, and Grover. "Mentor: An algorithm for mesh network topological optimization and routing". IEEE. Transactions on Communications. Volume: 39, Issue: 4, Abril 1991. Páginas: 503-513.

Kershenbaum, Aaron, Telecommunications Networks Design Algorithms., New York: MacGraw-Hill, 1993.

Kumaran, K. et al. "Efficient Algorithms for Location and Sizing Problems in Network Design", IEEE GLOBECOM '01 Global Telecommunications Conference, 2001.Volume 4, Noviembre 25-29. 2001 Páginas: 2586 - 2590.

Lightscape Networks. "Optical Metropolitan Networks and Beyond". White paper, 2004.

Liu, Z. y Bandyopadhyay S. "A Genetic Algorithm for Optimization of Logical Topologies in Optical Networks". IEEE Proceedings International Parallel and Distributed Processing Symposium, IPDPS 2002. Abril 15-19, 2002 Páginas: 202 - 209.

Lo, Chi-Chun y Chang, Wei-Hsin. "A Multiobjective Hybrid Genetic Algorithm for the Capacitated Multipoint Network Design Problem", IEEE Transactions on Systems, MAN, and Cybernetics—Part B: Cybernetics, Vol. 30, No. 3, 2000.

Maier, G., y cois. "Design and Cost Performance of the Multistage WDM-PON Access Networks", IEEE Journal of Lightwave Technology, Vol. 18, No. 2, Volume 18, Issue 2, Feb. 2000 Páginas:125 - 143.

Maruyama, K; .Designing reliable packet-swithced Communications networks., Proceedings of the IEEE ICCC, pages 493-498, 1978.

Modiano, Eytan. "Enrutamiento de amplia difusión (broadcast)", Massachusetts Institute of Technology Department of Aeronautics and Astronautics 2000.

Mendoza, Eduardo "Diseño y Optimización de Redes: Mentor vs. Programación Lineal" Tesis QA76.9.A43 M45, ITESM 2004.

Osifchin, Nicholas, "A Telecommunications Buildings/Power Infrastructure in a New Era of Public Networking", International Power Strategies 2000.

Pellegatta M., Monguzzi M., Mazzarese A. y Zuchetti A., "Fiber Networks Maintenance in the all Optical Network Era", IEEE/IFIP Symposium 2002 Network Operations and Management NOMS. Abril 2002 Páginas: 855 - 868.

Rajan, D. "Introduction to AMPL A Tutorial", Basado en AMPL: A Modeling Language for Mathematical Programming by R. Fourer, D. M. Gay, and B. W. Kernighan (1993: Boyd & Fraser), 2000.

Ramírez-Rosado, I. y Bernal-Agustín, J. "Reliability and Costs Optimization for Distribution Networks Expansión Using an Evolutionary Algorithm", IEEE Transactions on Power Systems, Vol. 16, No. 1, 2001.

Saradhi, C. y cois. "Distributed Network Control for Establishing Reliability-Constrained Least-Cost Lightpaths in WDM Mesh Networks", IEEE Proceedings Eighth International Symposium on Computers and Communication (ISCC 2003), vol.1, Junio 30, Julio 3, 2003 Páginas: 678 - 683.

Sauvé, J., Silva, F. "Availability Considerations in Network Design", IEEE Proceedings. 2001 Pacific Rim International Symposium on Dependable Computing, Dic. 2001 Páginas: 119- 126.

Sayoud, H., Takahashi K. y Vaillant B. "A Genetic Local Tuning Algorithm for a Class of Combinatorial Networks Design Problems", IEEE Communications Letters, Vol. 5, No. 7, 2001

Sharma, R. "Network topology optimization : the art and science of network design" Publisher: New York Van Nostrand Reinhold, TK5105.5 .S427, 1990.

Song, K. y Somani, A. "Fault Tolerant ATM Backbone Network Design Considering Cell Loss Rates and End-to-End Delay Constraints", IEEE 13th Annual International Phoenix Conference on Computers and Communications, 1994, Abril 1994 Página: 119.

Steiglitz, K., Weiner, P. y Kleitman, D. "The Design of Minimum-Cost Survivable Networks" IEEE Transactions on Circuit Theory, Vol. CT-16, No. 4, 1996.

Universidad de Chile, Departamento de Computación http://www.dcc.uchile.cl/web/channel.html (Accesada, Marzo 2005).

Van Norman, H. "LAN/WAN Optimization Techniques", Editorial Artech House, primera edición, ISBN 0-89006-617-5, 1992.

Wei, W. "Network Design Issues for a Terabit Optical Internet", IEEE Communications Engineer, Volume 1, Issue 2, Abril/Mayo 2003 Páginas: 38-41.

Xianhai, T., Weidong, J. y Dúo, Z. "Application Multicriterion Satisfactory Optimizaron Computer Networks Design", IEEE Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Agosto 2003 Páginas: 660 - 664.

Young SooMyung, Hyung-joon Kim, Dong-wan Tcha. "Design of communication Networks with survivability constraints", 1999.

Zeng, Y. S., Ding, L. X. y Kang, L. S. "An Evolutionary Algorithm of Contracting Search Space Based on Partial Ordering Relation for Constrained Optimization Problems", IEEE Proceedings. Fifth International Conference on Algorithms and Architectures for Parallel Processing. Oct. 2002 Páginas: 76 - 81.

Zhang, H., y cois. "Analysis and Comparison of Optimization Algorithm for Network Flow Control". IEEE Proceedings of the 41st Conference on Decisión and Control, Volume 1, Dic. 2002 Páginas: 1129-1134.

Zhao, D., Yao S. y Zhou M., "An Architecture with Reliability Support for Wide-Area Distributed Applications", IEEE Proceedings. Fifth International Conference on Algorithms and Architectures for Parallel Processing. Oct. 2002 Páginas: 269-272.

Zhu, H., et al. "Cost-Effective WDM Backbone Network Design With OXCs of Different Bandwidth Granularities", IEEE Journal on Selected Áreas in Communications, Volume 21, Issue 9, Nov. 2003 Páginas: 1452 - 1466.