

**UNIVERSIDAD DEL BÍO-BÍO
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL**

**PROFESOR GUÍA:
SR. CARLOS OBREQUE NIÑEZ**



UNIVERSIDAD DEL BÍO-BÍO

**UN PROCEDIMIENTO OPTIMAL PARA RESOLVER EL MEDIAN
SHORTEST PATH PROBLEM**

**TRABAJO DE TITULACIÓN PRESENTADO EN CONFORMIDAD A LOS REQUISITOS PARA
OBTENER EL TÍTULO DE INGENIERO CIVIL INDUSTRIAL**

Concepción, 10 de Abril de 2008

GERMÁN ENRIQUE PAREDES BELMAR

DEDICATORIA

Dedico este trabajo a mis padres, Enrique y Elvira, que con su esfuerzo, su incondicional apoyo y amor, hicieron posible el cumplimiento de esta meta; a mi abuelo Luis (Q.E.P.D.) que siempre me apoyó y aconsejó; y a Dios, que siempre me guía y acompaña en este hermoso camino.

AGRADECIMIENTOS

Quiero darle las gracias al Profesor Carlos Obreque, por su confianza, entrega, paciencia y sus valiosos consejos que me entregó durante todo el desarrollo del presente Proyecto de Título.

También agradezco a todas las personas (amigos, familiares, compañeros, profesores) que me entregaron ánimo y apoyo durante la realización de este trabajo, y a lo largo de todos estos años de estudio.

RESUMEN

Sea $G = (N, A)$ un grafo conexo, donde N es el conjunto de nodos y A el conjunto de arcos. Se consideran conocidos dos nodos de N : el nodo origen y nodo destino. Cada arco de A tiene un costo de construcción y se conoce la distancia más corta entre cada par de nodos de la red. El Median Shortest Path Problem (MSPP) consiste en localizar un path (camino) entre el nodo origen y el nodo destino, llamado path principal, de tal manera que todos los otros nodos de la red, que no están sobre este path, sean asignados a partir del nodo más cercano que se encuentre sobre el mismo path principal.

El MSPP es un problema multiobjetivo con trade-off entre el costo total del path principal y la accesibilidad a este path. El objetivo del costo consiste en la suma de todos los costos (o longitudes) de los arcos que conforman el path principal, entre el nodo origen y el nodo destino, y el objetivo de accesibilidad es medido en términos del tiempo (o distancia) hacia el path principal, definida como la suma de todas las distancias desde el path principal a todos los nodos que no pertenecen a este path. Estos dos objetivos están en conflicto porque mientras más grande es el costo del path principal más pequeño es el tiempo de viaje desde el path a los demás nodos de la red y viceversa.

En este trabajo se propone un procedimiento para detectar arcos que no forman parte de ninguna solución no inferior. Se propone un modelo de programación lineal entera binaria para determinar soluciones no inferiores del MSPP en forma óptima. Además, se resolvió el MSPP con una formulación basada en flujo multicommodity, con el objetivo de comparar resultados. Se presenta una red de 30 nodos y 108 arcos dirigidos para mostrar el procedimiento que se propone en este trabajo. Se exponen también los resultados de las experiencias computacionales realizadas.

Índice de Contenidos

Capítulo 1: GENERALIDADES.....	1
1.1 Introducción	1
1.2 Origen del Tema.	1
1.3 Justificación del Tema.	1
1.4 Objetivos.....	4
1.4.1 Objetivo General	4
1.4.2 Objetivos Específicos.....	4
1.5 Alcances y Ámbito de Estudio.	4
1.6 Metodología	5
Capítulo 2: LA INVESTIGACIÓN DE OPERACIONES Y LA PROGRAMACIÓN LINEAL	6
2.1 La Investigación de Operaciones.....	6
2.1.1 Reseña Histórica de la Investigación de Operaciones	6
2.2 Clasificación de los problemas de Investigación de Operaciones	7
2.3 La Metodología de la Investigación de Operaciones	8
2.4. La Programación Lineal.....	10
2.5 La Programación Lineal Entera.	12
2.5.1 El Algoritmo Branch and Bound.	13
Capítulo 3: PROGRAMACIÓN MULTIOBJETIVO.....	16
3.1 Formulación del Problema General de Programación Multiobjetivo	16
3.1.1 Formulación del Problema	16
3.1.2 No Inferioridad.....	17
3.2 Clasificación de Métodos de Programación Multiobjetivo	19
3.2.1 Una Categorización de Técnicas.....	19
3.3 Técnicas para Generar Soluciones No Inferiores	21
3.3.1 El Método de los Pesos.....	21
3.3.2 El Método de las Restricciones.	22
3.3.3 El Método de Estimación de Conjuntos No Inferiores (NISE)	23
Capítulo 4: MODELOS DE REDES.....	25
4.1 Introducción	25
4.2 Definiciones de Redes.....	25
4.3 Minimun Spanning Tree Problem (Problema del Árbol de Extensión Mínima)	26
4.4 El Problema de la Ruta Más Corta	28
Capítulo 5: AMPL: A MATHEMATICAL PROGRAMMING LANGUAGE	30
5.1 Introducción	30
5.2 El Lenguaje AMPL	31
5.3 Entorno AMPL para MSDOS	33
5.4 Resolución de Problemas en AMPL Plus	34

Capítulo 6: EL MEDIAN SHORTEST PATH PROBLEM (MSPP).....	36
6.1 Revisión Bibliográfica.....	36
6.2 Formulación Matemática del MSPP.....	38
6.2.1 Formulación del MSPP mediante Programación Lineal Entera Binaria....	40
6.2.2 Formulación mediante Flujo Multicommodity del MSPP	42
Capítulo 7: RESOLUCIÓN DEL MEDIAN SHORTEST PATH PROBLEM	46
7.1 Ejemplo de Aplicación del MSPP.....	46
7.1.1 Datos del Modelo	46
7.1.2 Aparición y Eliminación de Subtours en el MSPP utilizando el Método de Planos Cortantes.....	50
7.1.3 Eliminación de Arcos Secundarios.....	54
7.2 Búsqueda de las Soluciones No Inferiores del MSPP	56
7.2.1 Obtención del Path Hamiltoniano.....	57
7.2.2 Obtención de la Ruta Más Corta	59
7.2.3 Obtención de las Soluciones No Inferiores y la Curva de <i>Trade-Off</i>	61
7.2.4 Aplicación del Método NISE para obtener Soluciones No Inferiores.....	62
7.3 Soluciones No Inferiores para la red de 30 nodos	66
7.3.1 Solución No-Inferior N°4.....	68
7.3.2 Solución No Inferior N°7	69
7.3.3 Selección de Soluciones No Inferiores.....	70
7.4 Experiencias Computacionales para la red de 30 nodos	73
7.4.1 Experimento 1: Obtención de SNI sin reducción de arcos secundarios	73
7.4.2 Experimento 2: Obtención de SNI con reducción de arcos secundarios...	74
7.4.3 Experimento 3: Obtención de SNI mediante la Formulación Basada en Flujo Multicommodity	76
7.4.4 Análisis de los Resultados de los Experimentos	78
7.5 Procedimiento propuesto para resolver el MSPP	80
7.6 Resultados Computacionales de los Problemas Test.....	82
7.6.1 Red de 21 Nodos	85
7.6.2 Red de 30 Nodos	89
7.6.3 Red de 50 Nodos	93
7.6.4 Red de 75 Nodos	97
7.6.5 Red de 100 Nodos	100
Capítulo 8: CONCLUSIONES Y FUTURAS INVESTIGACIONES	103
8.1 Conclusiones	101
8.2 Futuras Investigaciones.....	105
BIBLIOGRAFÍA	106

Índice de Figuras

Figura 1.1: Una solución del MSPP sobre la red de 21 nodos de Current <i>et al.</i> (1987).....	2
Figura 2.1: Clasificación de la Investigación de Operaciones.....	8
Figura 2.2: Metodología de la Investigación de Operaciones.....	9
Figura 3.1: Interpretación gráfica de no inferioridad para una región factible arbitraria en el espacio objetivo.....	18
Figura 3.2: Relaciones entre las categorías de los métodos.....	20
Figura 4.1: Red de 7 nodos y 10 arcos dirigidos.....	26
Figura 4.2: El Minimum Spanning Tree.....	27
Figura 5.1: Funcionamiento de AMPL, Holmes (1992).....	33
Figura 5.2: Entorno de comandos de AMPL para MSDOS.....	34
Figura 7.1: Red de 30 Nodos y 108 Arcos Dirigidos.....	48
Figura 7.2: Aparición de subtours en el MSPP.....	50
Figura 7.3: Efecto de la aplicación del método de planos cortantes.....	53
Figura 7.4: Eliminación de Arcos Secundarios para el MSPP.....	55
Figura 7.5: El Path Hamiltoniano para la red de 30 nodos.....	58
Figura 7.6: Ruta Más Corta para la red de 30 nodos.....	60
Figura 7.7: Gráfico del Path Hamiltoniano (PH) y la Ruta Más Corta (RMC).....	61
Figura 7.8: Pendiente de la recta entre el PH y la RMC.....	63
Figura 7.9: Obtención de una Solución No Inferior a partir del PH y la RMC.....	64
Figura 7.10: Obtención de una nueva Solución No Inferior	65
Figura 7.11: Representación Gráfica del Conjunto de Soluciones No Inferiores.....	67
Figura 7.12: Solución No Inferior N°4.....	68
Figura 7.13: Solución No Inferior N°7.....	69
Figura 7.14: Selección de Soluciones No Inferiores para el MSPP.....	70
Figura 7.15: Búsqueda de SNI en un intervalo mediante el Método de las Restricciones.....	72
Figura 7.16: Tiempos de CPU de las SNI en el Experimento 1.....	74
Figura 7.17: Tiempos de CPU de las SNI en el Experimento 2.....	75
Figura 7.18: Tiempos de CPU de las SNI en el Experimento 3.....	77
Figura 7.19: Comparación de Tiempos de CPU de los Experimentos Realizados.....	79
Figura 7.20: Tiempos de CPU para la red de 21 nodos, Origen: 2, Destino: 19.....	85
Figura 7.21: Tiempos de CPU para la red de 21 nodos, Origen: 4, Destino: 8.....	86
Figura 7.22: Tiempos de CPU para la red de 21 nodos, Origen: 17, Destino: 12.....	87
Figura 7.23: Comparación de tiempos totales de CPU para la red de 21 nodos.....	88
Figura 7.24: Tiempos de CPU para la red de 30 nodos, Origen: 8, Destino: 17.....	89
Figura 7.25: Tiempos de CPU para la red de 30 nodos, Origen: 15, Destino: 28.....	90
Figura 7.26: Tiempos de CPU para la red de 30 nodos, Origen: 18, Destino: 9.....	91
Figura 7.27: Comparación de tiempos totales de CPU para la red de 30 nodos.....	92
Figura 7.28: Tiempos de CPU para la red de 50 nodos, Origen: 3, Destino: 46.....	93
Figura 7.29: Tiempos de CPU para la red de 50 nodos, Origen: 33, Destino: 24.....	94
Figura 7.30: Tiempos de CPU para la red de 50 nodos, Origen: 33, Destino: 24.....	95

Figura 7.31: Comparación de tiempos totales de CPU para la red de 50 nodos.....	96
Figura 7.32: Tiempos de CPU para la red de 75 nodos, Origen: 4, Destino: 75.....	97
Figura 7.33: Tiempos de CPU para la red de 75 nodos, Origen: 21, Destino: 52.....	98
Figura 7.34: Comparación de tiempos totales de CPU para la red de 75 nodos.....	99
Figura 7.35: Tiempos de CPU para la red de 100 nodos, Origen: 40, Destino: 97....	100
Figura 7.36: Tiempos de CPU para la red de 100 nodos, Origen: 17, Destino: 66....	101

Índice de Tablas

Tabla 3.1: Un ejemplo de no inferioridad.....	18
Tabla 7.1: Costos $C(i, j)$ de los arcos de la red de 30 nodos.....	47
Tabla 7.2: Demandas de cada nodo para la red de 30 nodos generadas aleatoriamente.....	49
Tabla 7.3: Matriz de Distancias Mínimas para la red de 30 nodos.....	49
Tabla 7.4: Arcos Secundarios Candidatos.....	55
Tabla 7.5: Soluciones No Inferiores encontradas con el método NISE.....	66
Tabla 7.6: Búsqueda de tres SNI en un intervalo mediante el Método de las Restricciones.....	72
Tabla 7.7: Resultados del Experimento 1.....	73
Tabla 7.8: Resultados del Experimento 2.....	75
Tabla 7.9: Resultados del Experimento 3.....	77
Tabla 7.10A: Resultados para la red de 21 Nodos, Origen: 2; Destino: 19.....	85
Tabla 7.10B: Resultados para la red de 21 Nodos, Origen: 2; Destino: 19.....	85
Tabla 7.11A: Resultados para la red de 21 Nodos, Origen: 4; Destino: 8.....	86
Tabla 7.11B: Resultados para la red de 21 Nodos, Origen: 4; Destino: 8.....	86
Tabla 7.12A: Resultados para la red de 21 Nodos, Origen: 17; Destino: 12.....	87
Tabla 7.12B: Resultados para la red de 21 Nodos, Origen: 17; Destino: 12.....	87
Tabla 7.13A: Resultados para la red de 30 Nodos, Origen: 8; Destino: 17.....	89
Tabla 7.13B: Resultados para la red de 30 Nodos, Origen: 8; Destino: 17.....	89
Tabla 7.14A: Resultados para la red de 30 Nodos, Origen: 15; Destino: 28.....	90
Tabla 7.14B: Resultados para la red de 30 Nodos, Origen: 15; Destino: 28.....	90
Tabla 7.15A: Resultados para la red de 30 Nodos, Origen: 18; Destino: 9.....	91
Tabla 7.15B: Resultados para la red de 30 Nodos, Origen: 18; Destino: 9.....	91
Tabla 7.16A: Resultados para la red de 50 Nodos, Origen: 3; Destino: 46.....	93
Tabla 7.16B: Resultados para la red de 50 Nodos, Origen: 3; Destino: 46.....	93
Tabla 7.17A: Resultados para la red de 50 Nodos, Origen: 33; Destino: 24.....	94
Tabla 7.17B: Resultados para la red de 50 Nodos, Origen: 33; Destino: 24.....	94
Tabla 7.18A: Resultados para la red de 50 Nodos, Origen: 15; Destino: 38.....	95
Tabla 7.18B: Resultados para la red de 50 Nodos, Origen: 15; Destino: 38.....	95
Tabla 7.19A: Resultados para la red de 75 Nodos, Origen: 4; Destino: 75.....	97
Tabla 7.19B: Resultados para la red de 75 Nodos, Origen: 4; Destino: 75.....	97
Tabla 7.20A: Resultados para la red de 75 Nodos, Origen: 21; Destino: 52.....	98
Tabla 7.20B: Resultados para la red de 75 Nodos, Origen: 21; Destino: 52.....	98
Tabla 7.21A: Resultados para la red de 100 Nodos, Origen: 40; Destino: 97.....	100
Tabla 7.21B: Resultados para la red de 100 Nodos, Origen: 40; Destino: 97.....	100

Tabla 7.22A: Resultados para la red de 100 Nodos, Origen: 17; Destino: 66.....	101
Tabla 7.22B: Resultados para la red de 100 Nodos, Origen: 17; Destino: 66.....	101

Índice de Anexos

ANEXO A: Algoritmo de Generación de Números Aleatorios	
ANEXO B. Algoritmo de Generación de la Matriz de Rutas Más Cortas para el MSPP	
ANEXO C. Algoritmo de Resolución del MSPP mediante la Formulación Binaria	
ANEXO D. Algoritmo de Resolución del MSPP mediante la Formulación basada en Flujo Multicommodity	

Capítulo 1: GENERALIDADES

1.1 Introducción

En este capítulo se explica el origen del presente Trabajo de Titulación y se entregan las razones por las cuales fue necesario desarrollarlo. También se dan a conocer los objetivos de este estudio, tanto el objetivo general como los objetivos específicos, el alcance o ámbito y la metodología que se utilizó para alcanzar los objetivos propuestos.

1.2 Origen del Tema

El Tema desarrollado fue propuesto por el académico del Departamento de Ingeniería Industrial, señor Carlos Obreque Niñez, en respuesta a la necesidad de realizar una investigación acerca del problema de redes denominado en la literatura como el Median Shortest Path Problem, que consiste en localizar un path principal entre un nodo origen y un nodo destino de tal manera de minimizar el costo de construcción del camino principal y las distancias desde cada nodo, que no se encuentra en el camino principal, a un nodo que se encuentre sobre este camino.

1.3 Justificación del Tema

El problema de estudio del presente Proyecto de Título, se sitúa dentro del campo de la Programación Lineal Entera. Es importante hacer notar que el MSPP es un problema que posee dos objetivos que están en conflicto: por una parte considera el objetivo de minimizar costos de construcción de un path, y por otra parte, minimizar el tiempo (ó longitud) total de viaje requerido por la demanda cada nodo que no esté en el path principal para alcanzar al nodo más cercano que si se encuentre en el path principal. Este problema entrega como solución, un conjunto de

caminos mediante una curva de trade-off entre el costo de construcción del path principal y los costos del usuario (accesibilidad, costos sociales, etc.) de manera que el tomador de decisión pueda elegir cuál será la ruta óptima a seguir de acuerdo a su presupuesto.

En la Figura 1.1 se muestra el MSPP en forma gráfica, aplicado a la red de 21 Nodos de Current *et al.* (1987), donde el Nodo Origen es el Nodo 2 y el Nodo Destino el 19. El path principal es denotado con flechas continuas (color rojo) mientras que las asignaciones a los nodos que no están en el path principal, están marcadas con flechas segmentadas (color azul). Los pesos sobre los arcos corresponden a las distancias entre el par de nodos.

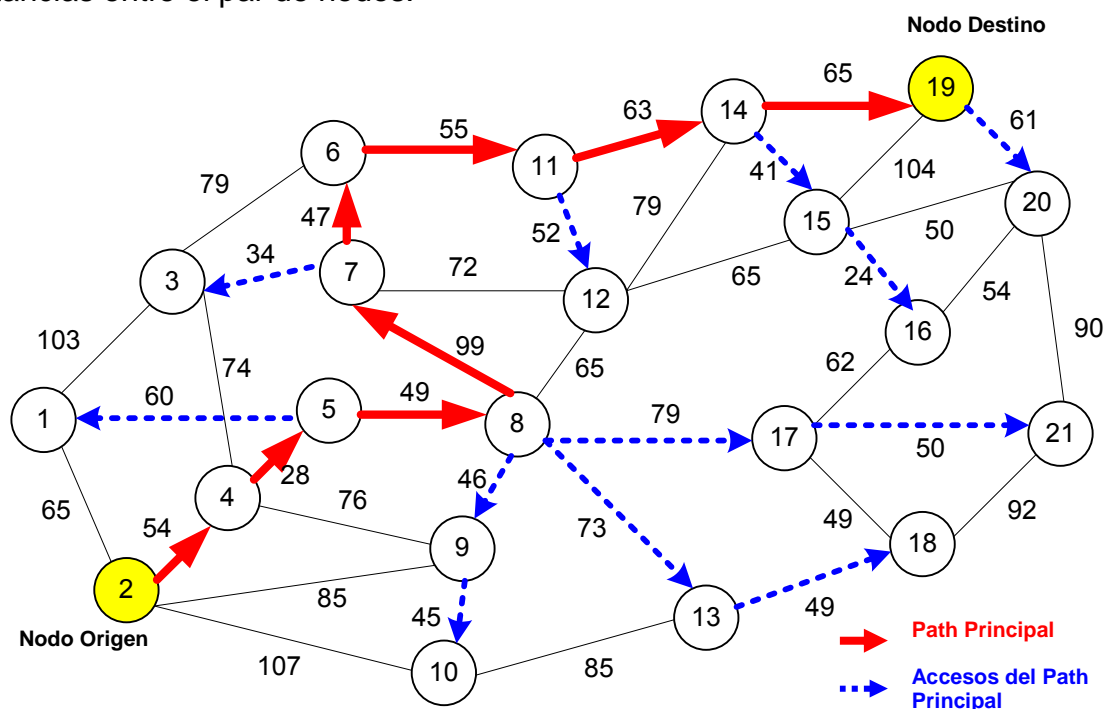


Figura 1.1: Una solución del MSPP sobre la red de 21 nodos de Current *et al.* (1987)

Como es usual, en los problemas multiobjetivo, no existe una única solución óptima, debido a que existen objetivos en conflicto. El concepto de solución óptima es reemplazado por soluciones eficientes, o soluciones no inferiores (SNI), Labbé *et al.* (1998); por lo tanto, la necesidad del estudio del MSPP radica en la aplicación del

una o más técnicas para la resolución de problemas multiobjetivo, generando alternativas eficientes de solución para los tomadores de decisiones.

El MSPP es aplicable en muchos problemas de diseño de redes de transporte, especialmente en aquellos en que los costos del usuario y del operador del sistema son factores importantes. Por ejemplo, si se desea conectar 2 grandes ciudades con una nueva línea del metro, el costo total de las líneas del metro es un factor importante para el operador del sistema. Por otra parte, el acceso a la nueva línea del metro con demandas en ciudades intermedias también es un factor importante debido a que representa un costo de los usuarios del sistema y, por lo tanto, podrían afectar la demanda total del sistema. En algunas situaciones (proyectos sociales por ejemplo), el objetivo principal puede ser la satisfacción de los usuarios, medido en términos de accesibilidad. Entonces, el MSPP podría ser utilizado para generar configuraciones de líneas de alternativas por donde debería pasar el metro.

Otras aplicaciones de este problema incluyen, también, problemas de transporte tales como la localización de autopistas y el diseño de rutas para aerolíneas, y por otra parte, problemas de telecomunicaciones y problemas de distribución de fluidos.

1.4 Objetivos

1.4.1 Objetivo General

El objetivo principal del presente estudio consiste en modelar y resolver el MSPP, utilizando Programación Lineal Entera.

1.4.2 Objetivos Específicos

- Proponer distintos modelos matemáticos para el MSPP.
- Determinar un método para resolver problemas multiobjetivo para aplicarlo al MSPP.
- Resolver el MSPP utilizando el software AMPL (A Mathematical Programming Language).
- Implementar y resolver el modelo propuesto aplicado a diferentes problemas test de otros artículos de la literatura y problemas obtenidos de Internet.

1.5 Alcances y Ámbito de Estudio

Este Proyecto de Título se sitúa dentro del ámbito de la investigación académica, con el objetivo de aplicar conocimiento en la resolución de un problema difícil de resolver. Específicamente, se trabajó en el campo de la modelación y resolución de problemas mediante Programación Lineal Entera.

En este estudio se considera la aplicación de una o más técnicas multiobjetivo para la resolución del MSPP aplicado a diferentes problemas test.

1.6 Metodología

La metodología utilizada para alcanzar los objetivos propuestos se basa en la investigación y el aprendizaje a través de herramientas tales como cursos dictados por expertos, textos relacionados con el presente trabajo (textos de Investigación de Operaciones, Programación Multiobjetivo, Lenguaje de Programación Matemática AMPL, Programación Lineal Entera, Publicaciones Científicas, entre otros) y sitios de Internet. También se recurrió a entrevistas periódicas con el profesor Guía.

Este trabajo se divide en una serie de capítulos donde los contenidos se agrupan de acuerdo a características similares. Cada uno de estos capítulos es imprescindible en el logro de los objetivos propuestos.

Para el análisis del Estado del Arte del MSPP, se recurrió principalmente a información existente en distintas publicaciones realizadas por investigadores en revistas científicas.

Para la obtención de información acerca de la Investigación de Operaciones, sus aplicaciones y sus distintas ramas de conocimiento, se investigó en libros clásicos y en sitios de Internet.

La descripción del software que se utilizó (AMPL), se realizó principalmente haciendo uso del Manual de Usuario del mismo.

Una vez realizado lo anterior se modeló y resolvió el MSPP mediante la formulación binaria del problema utilizando el método de planos cortantes para la eliminación de subtours, y mediante la formulación basada en flujo multicommodity con el fin de comparar los resultados obtenidos por ambas formulaciones.

Capítulo 2: LA INVESTIGACIÓN DE OPERACIONES Y LA PROGRAMACIÓN LINEAL

En este capítulo se presenta una breve descripción de la Investigación de Operaciones y la Programación Lineal dando a conocer algunos datos históricos, definiciones fundamentales y sus aplicaciones, con el fin de mostrar el ámbito de estudio del presente Proyecto de Título.

2.1 La Investigación de Operaciones

Existen varias definiciones de lo que es la Investigación de Operaciones. Josa (2007) propone las siguientes definiciones:

- La ciencia que estudia el modelado de sistemas probabilísticos y determinísticos que se originan en la vida real desde un punto de vista de la toma de decisiones óptimas.
- El estudio de cómo formular modelos matemáticos para problemas complejos de administración e ingeniería y cómo analizarlos para tener una visión de las posibles soluciones.
- Un enfoque científico para la toma de decisiones ejecutivas que consiste en:
 - a) el arte de modelar situaciones complejas;
 - b) la ciencia de desarrollar técnicas de solución para resolver dichos modelos;
 - c) la capacidad de comunicar efectivamente los resultados.

2.1.1 Reseña Histórica de la Investigación de Operaciones

El concepto de Investigación de Operaciones nació durante la primera guerra mundial en Inglaterra entre los años 1914 y 1915, cuando F.W. Lanchester intentó tratar cuantitativamente las operaciones militares, obteniendo ecuaciones que

relacionaban el resultado de una batalla en función de la fuerza numérica relativa de los combatientes y de su capacidad relativa de fuego. Lanchester modeló una situación que involucraba opciones estratégicas, y después probó ese modelo contra la situación real. Éste procedimiento es el que los investigadores de esta ciencia han venido practicando desde entonces, Chediak (2003).

Después de la segunda guerra mundial, las técnicas desarrolladas empezaron a ser usadas en la planeación de los negocios. En 1950 se organizó la Operations Research Society of America (ORSA) y The Institute of Management Science (TIMS). Desde la década de los 70's las dos sociedades realizan publicaciones con trabajos y artículos relacionados con problemas operacionales en el uso de la ciencia administrativa y la investigación de operaciones.

Actualmente, existen muchas situaciones de distinta naturaleza que pueden ser representadas matemáticamente por un modelo de Programación Lineal, por ejemplo, en finanzas, medicina, transportes, etc.

2.2 Clasificación de los problemas de Investigación de Operaciones

Esta clasificación atiende más bien al tipo de modelo donde encaja el problema. En algunos casos se tendrá que ajustar el problema con un Modelo Determinístico, en el cual todos los datos importantes del mismo se suponen conocidos, pero en otros, algunos de estos datos se consideran inciertos y normalmente vienen dados por una probabilidad por lo que será necesaria la utilización de un Modelo Probabilístico. Sin embargo, existen modelos que conviene tratar como Híbridos de estas dos categorías. En la Figura 2.1 se hace una agrupación aproximada, a grandes rasgos, de los diferentes tipos de problema dentro de la categoría a la que pertenecen, Josa (2007):

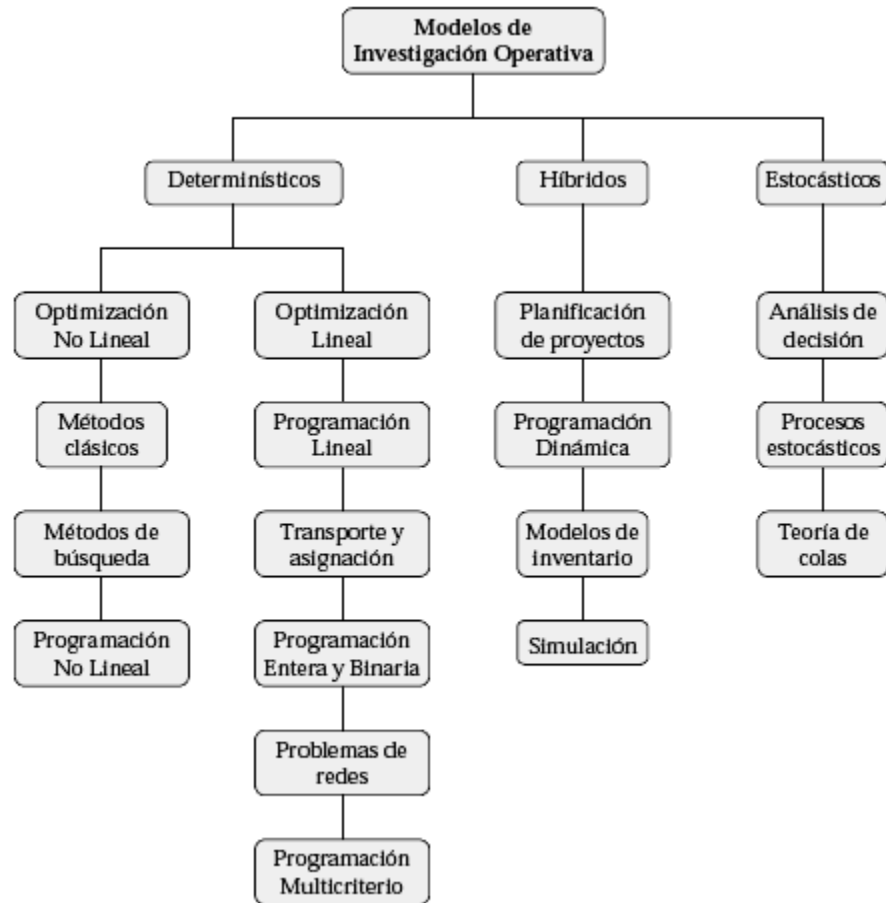


Figura 2.1: Clasificación de la Investigación de Operaciones

2.3 La Metodología de la Investigación de Operaciones

En su forma más simple, la Investigación de Operaciones puede considerarse como un procedimiento que consta de cuatro pasos, tal como se muestra en la Figura 2.2, Josa (2007):

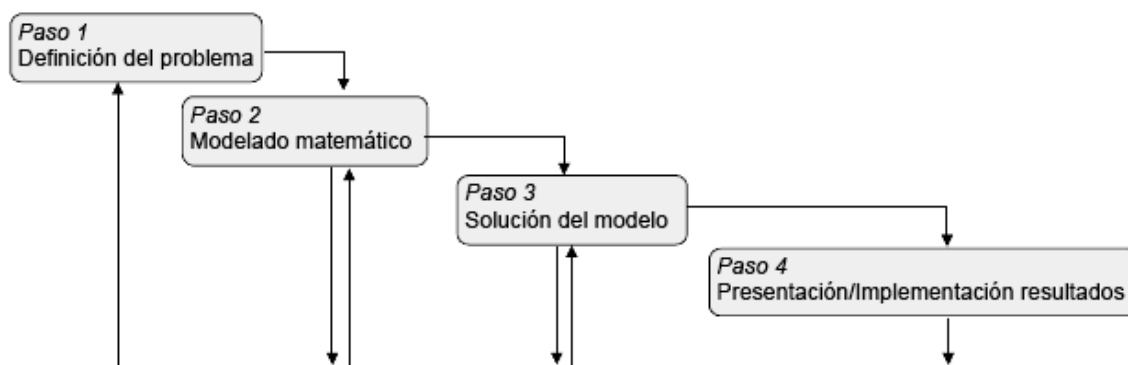


Figura 2.2: Metodología de la Investigación de Operaciones

- *Paso 1, Definición del problema.* Quizás la parte más importante de todo el proceso sea la definición del problema. Una respuesta incorrecta a una pregunta correcta no suele tener consecuencias fatales, ya que se pueden hacer revisiones y explorar otras alternativas; sin embargo, la respuesta correcta a una pregunta incorrecta puede ser desastrosa. Es importante que el problema esté claramente definido antes de invertir una gran cantidad de trabajo y energía en resolverlo.
- *Paso 2, Modelamiento Matemático.* Es un procedimiento que reconoce y verbaliza un problema para posteriormente cuantificarlo transformando las expresiones verbales en expresiones matemáticas. El modelamiento matemático es un arte, que mejora con la práctica. El proceso del modelado matemático consta de cuatro pasos:
 1. Identificar las variables de decisión.
 2. Identificar la función objetivo.
 3. Identificar las restricciones.
 4. Traducir los elementos anteriores a un modelo matemático.

- *Paso 3, Resolución del modelo.* Aceptado ya el modelo matemático que mejor describe la situación en estudio, se aplican los algoritmos y métodos matemáticos diseñados para su resolución. Las etapas de resolución del modelo son las siguientes:
 1. Elegir la técnica de resolución adecuada.
 2. Generar las soluciones del modelo.
 3. Comprobar/validar los resultados.
 4. Si los resultados son inaceptables, revisar el modelo matemático.
 5. Realizar análisis de sensibilidad.

- *Paso 4, Presentación/Implementación de los resultados.* Éste es el paso final dentro del proceso. Los pasos de este proceso son los siguientes:
 1. Preparar informes y/o presentaciones.
 2. Vigilar el proceso de implementación de los resultados.

2.4. La Programación Lineal

En 1947, George Dantzig creó un método eficaz, el algoritmo Simplex, para resolver problemas de Programación Lineal (PL). En una encuesta de la revista Fortune, de 500 empresas, el 85% de las que contestaron dijeron que habían utilizado la PL, Winston (1994).

Desde su aparición la PL ha demostrado que es una herramienta muy efectiva dentro de la Investigación de Operaciones. Un factor importante en el uso de esta técnica es el gran avance que han tenido los programas computacionales, que permiten resolver problemas extensos de PL en muy poco tiempo.

La Programación Lineal (PL) es una herramienta para resolver problemas de optimización que se caracterizan por tener como Función Objetivo y Restricciones combinaciones lineales de las Variables de Decisión. La principal ventaja radica en que existe un algoritmo eficiente (Simplex) para resolver este tipo de modelos.

Un problema de PL se podría representar matemáticamente de la siguiente forma, Fernández (2000):

$$\text{Optimizar } f(x_1, x_2, \dots, x_n) = c_1x_1 + c_2x_2 + \dots + c_nx_n \quad (2.1)$$

Sujeto a las siguientes restricciones:

$$g_j(x_1, x_2, \dots, x_n) = a_{j1}x_1 + a_{j2}x_2 + \dots + a_{jn}x_n = b_j \quad (2.2)$$

Donde:

c_i, a_{ji} y b_j son constantes conocidas, con $i = 1, 2, \dots, n$ y $j = 1, 2, \dots, m$

$$\text{Además } x_i \geq 0 \text{ para todo, } i = 1, 2, \dots, n \quad (2.3)$$

La expresión (2.1) se conoce como Función Objetivo. La expresión (2.2) se conoce como las Restricciones del modelo, y (2.3) se denominan las Restricciones de no negatividad. Las variables x_i se conocen como las Variables de Decisión. Los coeficientes c_i son los denominados coeficientes de costos; los coeficientes a_{ji} se conocen como los Coeficientes Tecnológicos y los b_j son los coeficientes de Recursos/Requerimientos, Fernández (2000).

Como se puede observar, todas las variables y restricciones son lineales.

La PL es un método sencillo y poderoso en su aplicación. La aplicación a nuevos tipos de problemas ha generado importantes investigaciones teóricas en esa dirección. Gracias al desarrollo de la tecnología, y por ende, los computadores y software, han surgido nuevos problemas de gran envergadura que ahora son posibles de resolver, Chediak (2003).

2.5 La Programación Lineal Entera

En algunas situaciones que pueden representarse con modelos lineales, se encuentra que sólo tienen sentido aquellas soluciones de la región factible en las que todas o algunas de las variables de decisión sean números enteros. Estas situaciones pueden representarse mediante modelos matemáticos ligeramente diferentes de la programación lineal. En este contexto, un problema de programación lineal entera es un problema de programación lineal con la restricción adicional de que algunas de las variables deben tomar valores enteros. Cuando todas las variables deben tomar valores enteros, se trata de un problema de programación lineal entera puro, en caso contrario, se dice que es mixto. Una variable es binaria si sólo puede tomar los valores 0 y 1. De este modo, se tienen tres tipos de variables, Sallán *et al.* (2002):

- Variables reales
- Variables enteras
- Variables binarias

La posibilidad de utilizar variables enteras o binarias amplía notablemente las posibilidades de modelación matemática. El precio por una mayor versatilidad de la herramienta es el de una mayor complejidad en la resolución del modelo. Esta complejidad se debe a los siguientes hechos, Sallán *et al.* (2002):

- Para las variables enteras del modelo, los vértices de la región factible no tienen porqué ser números enteros. En consecuencia, la solución óptima se encontrará en el interior de la región factible, por lo que el método Simplex, empleado de forma directa, no proporcionará la solución óptima.
- A diferencia del problema con variables reales, el número de soluciones de un modelo de programación lineal es finito, por lo que podría plantearse la posibilidad de encontrar la solución mediante la exploración de todas las soluciones posibles. Sin embargo, el número de soluciones a explorar para un problema mediano puede ser muy elevado: en principio, para un problema con n variables enteras debemos explorar 2^n soluciones (excluyendo quizás algunas descartadas por las restricciones).

Se han desarrollado metodologías que permiten explorar de manera más eficiente que la mera enumeración del conjunto de soluciones posibles. Gran número de estas metodologías emplean la lógica del branch and bound (ramificar y acotar), y están incorporadas a la mayoría de programas informáticos que resuelven modelos lineales.

2.5.1 El Algoritmo Branch and Bound

Un primer paso para solucionar un problema de programación lineal entera es resolver, mediante el método Simplex, el problema lineal asociado. Se trata de un problema lineal con la misma función objetivo y restricciones que el modelo original, pero al que se ha relajado la condición de que todas o algunas de las variables de decisión sean enteras. Si la solución así obtenida es entera, se habrá encontrado la solución del modelo de programación lineal entera. En caso contrario, la solución así obtenida es una primera aproximación a la solución del modelo, Sallán *et al.* (2002).

En el procedimiento del algoritmo branch and bound se trata de ir añadiendo restricciones al programa lineal asociado hasta encontrar la solución entera óptima. Para ello se procede en dos pasos: ramificación (branch) y acotamiento (bound).

2.5.1.1 Ramificación

Se trata de añadir restricciones al modelo que fuercen a que una de las variables sea entera. Esto se consigue añadiendo una de estas dos restricciones para que alguna de las variables x_{Bi} que no sea entera en la solución obtenida hasta el momento, Sallán *et al.* (2002):

- Redondeo por defecto: se impone que la variable x_{Bi} sea inferior o igual a la parte entera del valor de esa variable en el óptimo del problema lineal x_{Bi} con las restricciones hasta el momento. Esto equivale a añadir la restricción:

$$x_i \leq E(x_{Bi}) \quad (2.4)$$

- Redondeo por exceso: se impone que la variable x_i sea mayor o igual al entero inmediatamente superior al valor x_{Bi} . Esto equivale a añadir la restricción:

$$x_i \geq E(x_{Bi}) + 1 \quad (2.5)$$

En seguida, se deberá proceder a resolver mediante el método Simplex dos problemas lineales. El primero, además de las restricciones que pudiera tener de etapas anteriores, llevará incorporada las restricciones de redondeo por defecto incorporada en esta etapa. El segundo se diferenciará del primero en que incluirá la restricción de redondeo por exceso, en vez de la de redondeo por defecto.

2.5.1.2 Acotamiento

Al hacer la ramificación, se encuentran dos alternativas posibles para obtener la solución entera. Los valores óptimos de la función objetivo Z^* de cada uno de los programas lineales resueltos en la etapa anterior serán una cota superior de las posibles soluciones que se puedan obtener mediante posteriores ramificaciones a partir de ese modelo. En consecuencia, sólo tendrá sentido continuar con el procedimiento de ramificación a partir del problema lineal que tenga mayor Z^* de los dos (para el problema de máximo). Siguiendo la otra ramificación, se obtendrán soluciones enteras con valores de Z^* inferiores, con toda seguridad, a los que se obtendrían con la otra ramificación, y por tanto subóptimos, Sallán *et al*, (2002).

Este hecho marca, además, cuándo se debe detener la exploración de soluciones enteras: cuando el problema escogido tenga una solución con todas las variables enteras. Por ser el valor de la función objetivo una cota superior del óptimo, cualquier otra solución entera que se pudiera explorar sería subóptima, y por lo tanto no vale la pena seguir explorando Sallán *et al*, (2002).

La solución óptima del modelo de programación entera no necesariamente es la obtenida en la última etapa, sino la solución entera con mayor valor de función objetivo de las obtenidas en el proceso.

Capítulo 3: PROGRAMACIÓN MULTIOBJETIVO

3.1 Formulación del Problema General de Programación Multiobjetivo

En esta sección, se da a conocer la forma general de un problema de programación multiobjetivo. Para la realización de este capítulo, se utilizaron extractos del libro “Multiobjective Programming and Planning” de Jared Cohon (1978).

3.1.1 Formulación del Problema

La programación multiobjetivo trata con problemas de optimización con dos o más funciones objetivos. El problema de programación multiobjetivo difiere de la optimización clásica (un solo objetivo) sólo en la expresión de sus respectivas funciones objetivos. El problema general de optimización multiobjetivo con n variables de decisión, m restricciones y p objetivos es:

$$m \quad Z(x_1, x_2, \dots, x_n) = [Z_1(x_1, x_2, \dots, x_n), Z_2(x_1, x_2, \dots, x_n), \dots, Z_p(x_1, x_2, \dots, x_n)] \quad (3.1)$$

Sujeto a:

$$g_i(x_1, x_2, \dots, x_n) \leq 0, \quad i = 1, 2, \dots, m \quad (3.2)$$

$$x_j \geq 0, \quad j = 1, 2, \dots, n \quad (3.3)$$

Donde $Z(x_1, x_2, \dots, x_n)$ es la función multiobjetivo y $Z_1(\cdot), Z_2(\cdot), \dots, Z_p(\cdot)$ son las p funciones objetivos individuales. Notar que las funciones objetivos en (3.1) no están sumadas, multiplicadas o combinadas de ninguna manera.

3.1.2 No Inferioridad

En los problemas con sólo un objetivo, la finalidad de la solución es identificar la solución óptima: la solución (ó soluciones) factibles que provee el mejor valor para la función objetivo. Notar que cuando existen óptimos alternativos, el valor óptimo de la función objetivo es único. Sin embargo, esta noción de optimalidad debe ser descartada para problemas multiobjetivo, debido a que la solución que maximiza un objetivo, no maximizará, en general, cualquiera de los otros objetivos. Lo que es óptimo en término de uno de los p objetivos es usualmente no óptimo para otros $p-1$ objetivos.

La optimalidad juega un importante rol en la solución de problemas con un solo objetivo. Esto permite que analistas y tomadores de decisión restringir su atención hacia una solución única o bien hacia un muy pequeño subconjunto de soluciones del gran conjunto de soluciones factibles. Un concepto llamado No Inferioridad servirá de manera similar, pero menos limitada para el propósito de los problemas multiobjetivo.

La idea de no inferioridad es muy similar al concepto de dominancia. La no inferioridad es llamada “no-dominancia” por algunos programadores matemáticos, “Eficiencia” por otros, y “Optimalidad de Pareto” por algunos economistas. En la Tabla 3.1 se muestran tres soluciones en un problema con dos objetivos. La alternativa C es dominada por las alternativas A y B debido a que ambas alternativas producen más de ambos objetivos, Z_1 y Z_2 . La solución es dominada y por lo tanto es llamada “inferior”. Las soluciones que no son dominadas se llaman “no inferiores”. De este modo, por ejemplo, las alternativas A y B en la Tabla 3.1 son soluciones no inferiores.

Alternativa	Z_1	Z_2	
A	10	11	No Inferior
B	12	10	No Inferior
C	9	8	Inferior

Tabla 3.1: Un ejemplo de no inferioridad

De una manera más formal, la no inferioridad puede ser definida de la siguiente manera:

“Una solución factible para el problema de programación multiobjetivo es no inferior si no existe alguna solución factible que mejore el rendimiento en un objetivo sin causar degradación mínima en otro objetivo”, Cohon (1978).

Esta definición es más fácil de entender gráficamente. Un conjunto de alternativas arbitrarias y factibles se muestra en la Figura 3.1. El área dentro de la forma es factible y acotada. Notar que los ejes de éste gráfico son los objetivos Z_1 y Z_2 . El gráfico se refiere al “espacio objetivo” para distinguirlo del gráfico de “espacios de solución”. El área factible en la Figura 3.1 se llama la región factible dentro del espacio objetivo.

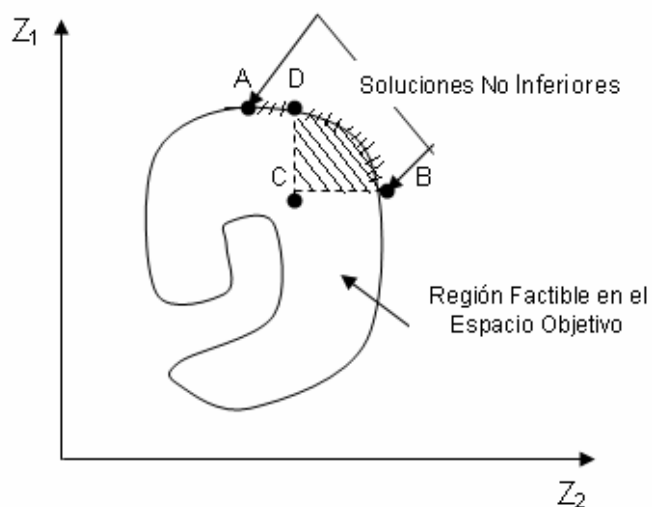


Figura 3.1: Interpretación gráfica de no inferioridad para una región factible arbitraria en el espacio objetivo

La definición de no inferioridad puede ser utilizada para encontrar soluciones no inferiores en la Figura 3.1. Primero, todas las soluciones interiores deben ser inferiores para que siempre se pueda encontrar una solución factible que mejore el rendimiento en ambos objetivos simultáneamente. Considerar que el punto C en la Figura 3.1 es una solución inferior. La alternativa B entrega más Z_1 sin decrecer la cantidad de Z_2 . En forma similar, D entrega más Z_2 sin decrecer Z_1 . De hecho, cualquier alternativa en el área achurada hacia el nor-oeste de C, domina a la alternativa C.

3.2 Clasificación de Métodos de Programación Multiobjetivo

La base para la clasificación es el rol del analista en el proceso de planeamiento el que está implicado cuando alguna técnica es usada. Después que las técnicas se categorizan, la aplicabilidad de los métodos se discute en variados contextos de tomas de decisiones, requerimientos computacionales son evaluados, y la evaluación de los métodos es considerada.

3.2.1 Una Categorización de Técnicas

Las características del proceso de toma de decisiones son los flujos de información en el proceso y el contexto de la toma de decisiones. Los flujos de información son importantes, debido a que ellos determinan el rol que el analista debe jugar en el proceso de planeamiento. El contexto de toma de decisiones define el objetivo de análisis.

Se considerarán dos tipos de flujos de información: desde el tomador de decisiones al analista (“top-down”) y desde el analista al tomador de decisiones (“bottom-up”). En el primer caso, el flujo desde el tomador de decisiones ocurre cuando éste específicamente articula preferencias de soluciones que deberían ser

identificadas. El analista, en el segundo caso, tiene los resultados acerca del conjunto de soluciones no inferiores.

El contexto del tomador de decisiones es especificar las preferencias para el análisis del problema. La configuración anterior es proyectada para incluir aquellas situaciones en las que existe un único tomador de decisiones, o bien, un grupo de tomadores de decisiones que comparten similares objetivos y preferencias, quienes deberán tomar una decisión acerca del problema con varios objetivos en conflicto.

El rango de métodos analíticos se segmenta en tres categorías, dependiendo de las preferencias de la toma de decisiones, para las que ellas son las más convenientes en los flujos de información que su uso requiere. Las categorías son técnicas de generación, métodos que incorporan preferencias y métodos de toma de decisión múltiples. La Figura 3.2 explica las relaciones entre los métodos:

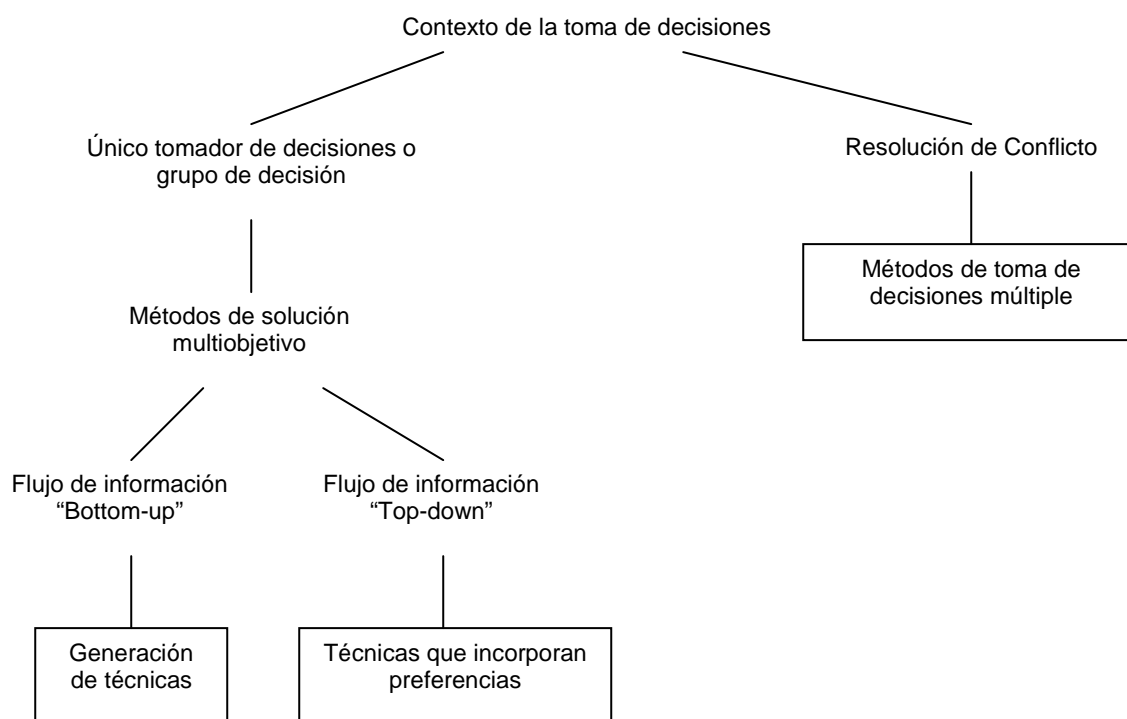


Figura 3.2: Relaciones entre las categorías de los métodos

3.3 Técnicas para Generar Soluciones No Inferiores

En esta sección se presentan algunos métodos para generar soluciones no inferiores desde un modelo de formulación multiobjetivo.

Los analistas están forzados a concentrar sus esfuerzos en la formulación y evaluación de alternativas, y cuando los resultados son reportados, ellos necesitan no recomendar sólo una alternativa. Los analistas, por cierto, deben buscar por ellos mismos una posición considerablemente defendible y cómoda, mostrando un buen rango de alternativas.

Una virtud adicional que puede ser exigida para generar aproximaciones, es su relativa generalidad hacia los procesos de decisión. Ellos pueden ser usados en procesos caracterizados por varios tomadores de decisión, por relativamente inaccesibles tomadores de decisión, o por la incertidumbre en lo que se refiere a quiénes son los tomadores de decisión.

La mayor debilidad de la generación de métodos es la sensibilidad del número de objetivos. Varios objetivos (usualmente más de tres) causan dos problemas: alta carga computacional y la complejidad de los resultados mostrados.

3.3.1 El Método de los Pesos

Ponderar los objetivos para obtener soluciones no inferiores es la técnica más antigua de solución multiobjetivo. El método sigue directamente de las condiciones de no inferioridad desarrolladas por los investigadores Kuhn and Tucker.

Suponer, por ejemplo, que se tiene un problema de localización de una estación de bomberos que tiene dos objetivos: maximizar el valor de propiedad de la instalación (medido en pesos) dentro de S kilómetros y maximizar la población dentro

de S kilómetros de la instalación. Los objetivos del valor de propiedad y la población serían llamados Z_1 y Z_2 , respectivamente. Los dos objetivos están en conflicto ya que las áreas comerciales son caracterizadas por un alto valor de propiedad y bajas poblaciones, mientras que las áreas residenciales poseen más personas y menor valor de propiedad. Dado que la estación de bomberos no puede ser localizada en toda el área dentro de S kilómetros, los valores máximos de Z_1 y Z_2 no puede ser obtenidos simultáneamente. La función objetivo del problema es:

$$\text{Maximizar } Z = [Z_1, Z_2] \quad (3.4)$$

Ahora, si alguien estuviera dispuesto a articular un juicio de valor en el que una persona tiene un valor de w pesos, entonces el problema multiobjetivo podría ser reducido a un problema con un solo objetivo. La especificación de w , el que es llamado *peso* en el objetivo Z_2 (población), es equivalente a la identificación de un deseable *trade-off* entre Z_1 y Z_2 . La ecuación (2.11) puede ser rescrita como:

$$\text{Maximizar } Z(w) = Z_1 + w Z_2 \quad (3.5)$$

Notar que ahora la función objetivo tiene una sola dimensión y es denotada por $Z(w)$, lo que implica la dependencia de una nueva función objetivo en el valor del peso w .

3.3.2 El Método de las Restricciones

El método de las restricciones es probablemente la técnica de generación más atractiva e intuitiva. Esta opera a través de la optimización de un objetivo mientras todos los otros son constantes con algún valor.

En el problema de la localización de una estación de bomberos, suponer ahora que en lugar de articular un peso de la población, los tomadores de decisión

establecen una restricción, en la que el mínimo de personas L puede estar en los S kilómetros de la instalación. Entonces el modelo matemático restringido es:

$$\text{Maximizar } Z_h(x_1, x_2, \dots, x_n) \quad (3.6)$$

Sujeto a:

$$\begin{aligned} Z_k(x_1, x_2, \dots, x_n) &\geq L_k \\ K &= 1, 2, \dots, h-1, h+1, 1, \dots, p \end{aligned} \quad (3.7)$$

3.3.3 El Método de Estimación de Conjuntos No Inferiores (NISE)

El método NISE (Noninferior Set Estimation) fue desarrollado por Cohon *et al.* (1978) para converger rápidamente en una buena aproximación del conjunto de soluciones no inferiores.

El método NISE es desarrollado para problemas con dos objetivos. Se asume que la región factible es un conjunto convexo y que las funciones objetivo son lineales.

El método NISE opera a través de la búsqueda de un número de puntos extremos no inferiores y la evaluación de las propiedades de los segmentos de línea entre ellos. Suponer que dos puntos extremos no inferiores han sido encontrados; entonces el segmento de línea entre ellos es factible y puede o no ser no inferior. Si el segmento de línea es no inferior, entonces cualquier movimiento fuera de la línea es infactible. Si el segmento de línea es inferior, entonces hay puntos no inferiores en dirección exterior.

El modelo matemático asociado al método NISE es el siguiente:

$$\text{Maximizar } Z(x_1, \dots, x_n; i; i+1) = \frac{Z_2(S_i) - Z_2(S_{i+1})}{Z_1(S_{i+1}) - Z_1(S_i)} Z_1(x_1, \dots, x_n) + Z_2(x_1, \dots, x_n) \quad (3.8)$$

Donde:

S_i es el punto no inferior con el mayor valor de Z_2 .

S_{i+1} es el punto no inferior con el menor valor de Z_2 .

Capítulo 4: MODELOS DE REDES

En el presente capítulo se presentan algunas definiciones básicas de los modelos de redes y sus aplicaciones. También se exponen dos problemas clásicos de redes: el Minimum Spanning Tree y el Problema de la Ruta más Corta.

4.1 Introducción

Un problema de redes puede ser visualizado prácticamente en los siguientes casos:

- Diseño de una tubería de agua, que es conectada de diferentes fuentes y que debe abastecer a una ciudad.
- Determinación de la ruta más corta entre dos ciudades, considerando los caminos existentes.
- Determinación de flujo de costo mínimo de campos petrolíferos a refinerías y finalmente a centros de distribución.

Un problema de redes, por lo general, posee un gran número de variables y restricciones, lo que permite el desarrollo de algoritmos altamente eficientes, que en muchos casos están basados en la teoría de la programación lineal.

4.2 Definiciones de Redes

Una red consta de un conjunto de nodos conectados por arcos. Asociado a cada arco se tiene un flujo de algún tipo. La notación estándar para describir una red G es $G = (N, A)$, donde N es el conjunto de nodos y A el conjunto de arcos. En general, el flujo de un arco está limitado por su capacidad que puede ser finita o infinita. Se dice que un arco está dirigido si permite un flujo positivo en una dirección,

y cero flujo en la dirección opuesta. Una red dirigida es una red con todos sus arcos dirigidos, Taha (1998).

En la Figura 4.1 se muestra una pequeña red de 7 nodos y 10 arcos que se describe como:

$$N = \{1, 2, 3, 4, 5, 6, 7\}$$

$$A = \{(1, 2), (1, 3), (2, 3), (2, 5), (3, 4), (4, 6), (5, 4), (5, 6), (5, 7), (6, 7)\}$$

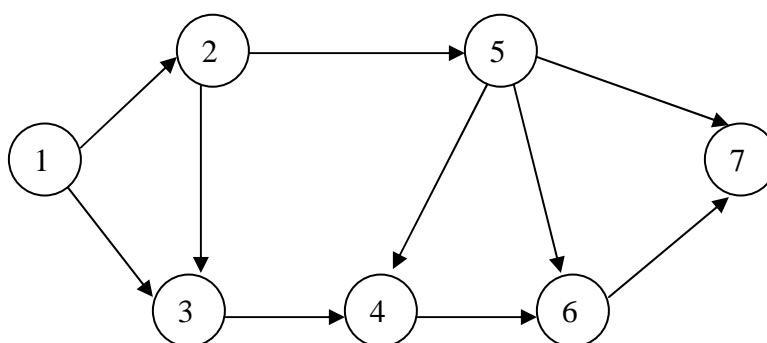


Figura 4.1: Red de 7 nodos y 10 arcos dirigidos

Un path o camino es una secuencia de arcos que conectan dos nodos sin considerar la orientación de los arcos individuales. Un path forma un ciclo si conecta un nodo consigo mismo.

4.3 Minimum Spanning Tree Problem (Problema del Árbol de Extensión Mínima)

El problema *Minimum Spanning Tree* (MST) consiste en encontrar las conexiones más “eficientes” entre todos los nodos a la red, las que no deben incluir ningún ciclo.

En la Figura 4.2 se muestra mediante un grafo el MST libre de subtours, donde el punto de partida (raíz) es el nodo 1.

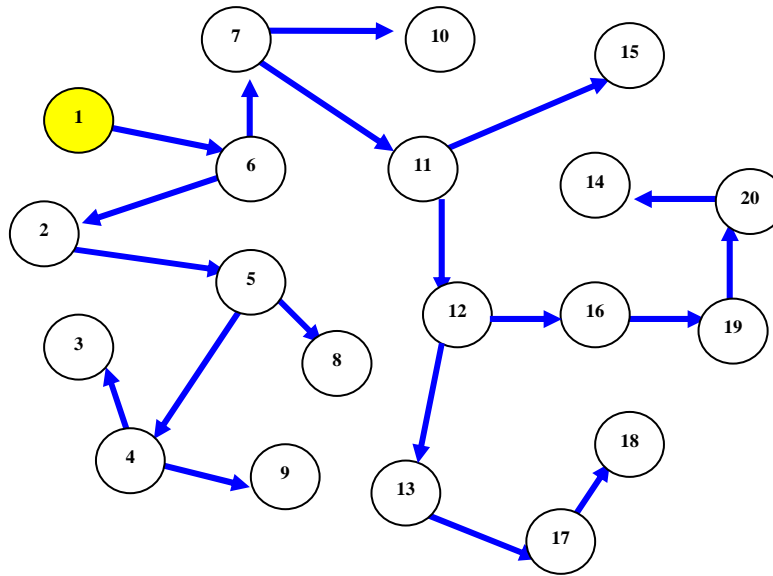


Figura 4.2: El Minimum Spanning Tree

El MST se puede modelar matemáticamente de la siguiente forma:

Variables de Decisión:

$$x_{i,j} = \begin{cases} 1, & \text{si la ruta pasa por el arco } (i, j), \\ 0, & \text{si no.} \end{cases}$$

Función Objetivo:

$$\text{Minimizar } Z = \sum_{(i,j) \in A} c_{i,j} \cdot x_{i,j} \quad (4.1)$$

Restricciones:

$$\sum_{j \in N: (r,j) \in A} x_{r,j} \geq 1 \quad (4.2)$$

$$\sum_{j \in N: (i,j) \in A} x_{i,j} = 1 \quad \forall j \in N \setminus \{r\} \quad (4.3)$$

$$\sum_{(i,j) \in A: i \in S, j \in S} x_{i,j} \leq |S| - 1 \quad \forall S \subseteq N \quad (4.4)$$

Donde:

N : conjunto de nodos de la red

A : conjunto de arcos de la red

S : conjunto de nodos de la red, subconjunto de N .

r : nodo raíz del MST

$c_{i,j}$ costos de los arcos (i,j) .

La restricción (4.2) asegura que exista al menos un arco que salga desde el nodo raíz (r) hacia todos los demás nodos del árbol. Por otra parte, el conjunto de restricciones (4.3) señala que a todos los nodos de la red les debe llegar un arco, exceptuando al nodo raíz,

El conjunto de restricciones (4.4) elimina los ciclos que puedan aparecer en el MST.

4.4 El Problema de la Ruta Más Corta

Este problema tiene que ver con la determinación de los arcos conectados en una red de transporte que constituyen, en conjunto, la distancia más corta entre un origen y un destino conocidos.

El problema de la ruta más corta puede ser modelado de la siguiente forma:

$$\text{Minimizar } Z = \sum_{(i,j) \in A} c_{i,j} x_{i,j} \quad (4.5)$$

Sujeto a:

$$\sum_{j \in N: (s,j) \in A} x_{s,j} = 1 \quad (4.6)$$

$$\sum_{i \in N: (i,t) \in A} x_{i,t} = 1 \quad (4.7)$$

$$\sum_{i \in N: (i,j) \in A} x_{i,j} - \sum_{k \in N: (j,k) \in A} x_{j,k} = 0 \quad \forall j \in N, j \neq s, t \quad (4.8)$$

$$x_{i,j} = (0,1) \quad \forall (i,j) \in A \quad (4.9)$$

Donde:

s : nodo origen.

t : nodo término.

N : conjunto de nodos de la red.

A : conjunto de arcos de la red.

$c_{i,j}$: costo de construcción (o distancia) del arco que conecta el nodo i con el nodo j .

La función objetivo (4.5) minimiza la suma de todos los costos de los arcos que están en la ruta. Las restricciones (4.6) y (4.7) aseguran que el nodo origen y el nodo destino estén presentes en el problema de la ruta más corta. El conjunto de restricciones (4.8) es un balance de flujo, donde todo el flujo que entre a un nodo, debe ser igual al flujo que sale de un nodo, exceptuando los nodos de origen y destino. Finalmente, el conjunto de restricciones (4.9) declara que todas las variables $x_{i,j}$ son binarias.

Capítulo 5: AMPL: A MATHEMATICAL PROGRAMMING LANGUAGE

En este capítulo se describe una síntesis del lenguaje AMPL, sus requisitos y potencialidades.

5.1 Introducción

La programación matemática de alto nivel involucra mucho más que sólo maximizar o minimizar una función objetivo sujeta a ciertas restricciones. Antes de que cualquier algoritmo de optimización pueda ser aplicado, se debe formular el modelo subyacente y generar las estructuras computacionales requeridas de datos, Fourer *et al.* (1990).

Existen muchas diferencias entre la forma en que los modeladores entienden un problema y en la que los algoritmos resuelven dicho problema. La “traducción” confiable desde la “forma del modelador” a la “forma del algoritmo” es frecuentemente un esfuerzo considerable.

En una aproximación tradicional a la “traducción”, el trabajo se divide entre el trabajo humano y el computacional. Primero, hay una persona quien entiende la “forma del modelador” y escribe un programa computacional que representará las estructuras de datos requeridas. Entonces el computador compila y ejecuta el programa para crear la “forma del algoritmo”. Este arreglo es muy costoso y propenso a errores. Este programa debe ser depurado por un modelador humano, el que debe revisar si la salida del algoritmo es significativa para las personas que la van a leer.

En el caso especial de la programación lineal, la mayor parte de las formas de los algoritmos es la representación de los coeficientes de las restricciones en

matrices. Un programa computacional produce una representación compacta de los coeficientes que se llaman al generador de matrices.

Muchas de las dificultades de la traducción del lenguaje del modelador al lenguaje algorítmico pueden ser circunscritas con el uso de un lenguaje de modelación para la programación matemática. Un lenguaje de modelación se diseña para expresar el lenguaje del modelador en una vía que puede servir como una entrada directa a un sistema computacional. Entonces la traducción del lenguaje algorítmico puede ser reconocida enteramente por un computador, sin una etapa intermedia de programación. Implementaciones como GAMS y MGG estuvieron en proceso en los 70's, y el desarrollo de las implementaciones se ha incrementado en los últimos años, Fourer *et al.* (1990).

5.2 El Lenguaje AMPL

AMPL es un programa computacional desarrollado en Bell Labs por R. Fourer, D.M. Gay y B.W. Kernighan en el año 1985. Este software permite que el usuario utilice un lenguaje algebraico para plantear y resolver modelos de optimización, mientras que el computador de manera interna, se comunica con el solver que corresponda.

AMPL es una interfaz entre el usuario y el solver desarrollada para ser utilizada en ambiente Unix, DOS. AMPL proporciona una interfaz sencilla que permite llevar de manera relativamente directa problemas de programación matemática al computador para poder resolverlos usando diferentes solvers (CPLEX, MINOS, etc.).

AMPL es un producto comercial, pero se puede utilizar una versión estudiantil de AMPL que está disponible en forma gratuita en Internet (www.ampl.com) para uso académico. La versión estudiantil de AMPL tiene un límite de 300 variables y 300 restricciones, mientras que la versión profesional no tiene límites en el número de

variables y restricciones, quedando limitado a la capacidad del computador donde esté instalado.

Los siguientes pasos describen el proceso para resolver un problema de optimización en AMPL:

- Plantear el modelo del problema a resolver, variables de decisión, función objetivo, restricciones.
- Generalizar el modelo, de tal forma de resumir el máximo de expresiones en un número mínimo de líneas de código.
- Implementar el modelo en lenguaje AMPL.
- Especificar los datos que definen al problema particular.
- Solucionar el problema, indicándole a AMPL el algoritmo de resolución del problema y la forma de presentación de los resultados.
- Análisis de resultados.
- Ajustar el modelo y datos si es necesario repetir el análisis.

Una de las ventajas de AMPL por sobre otros programas similares, es que la notación utilizada para definir expresiones matemáticas es muy parecida a la notación algebraica estándar.

En la Figura 5.1 se muestra el funcionamiento del lenguaje AMPL. Para comenzar, AMPL necesita un modelo de programación matemática, el que describe variables, objetivos y restricciones referidos a datos específicos. AMPL trabaja como un compilador: el modelo y la entrada son puestos en una forma intermedia la cual puede ser leída por un solver. El solver busca una solución óptima para el problema a través de la lectura del archivo intermedio producido por AMPL y aplicando el algoritmo apropiado. El solver entrega la solución como un archivo de texto, el cual puede ser visto directamente, Holmes (1992).

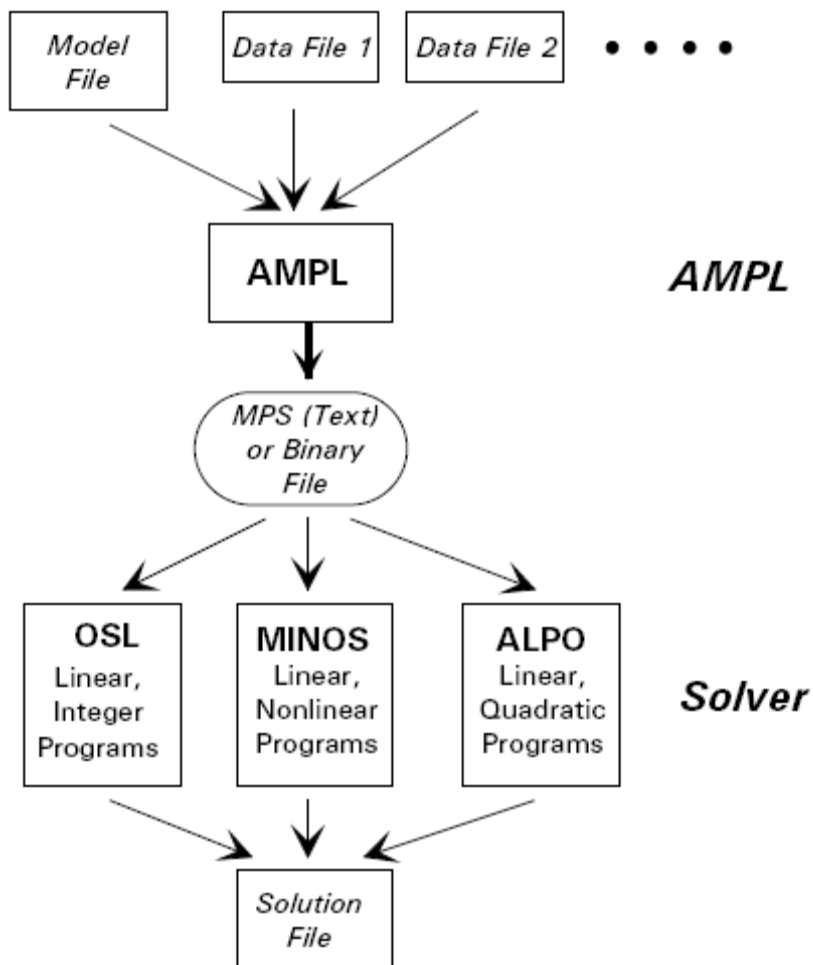
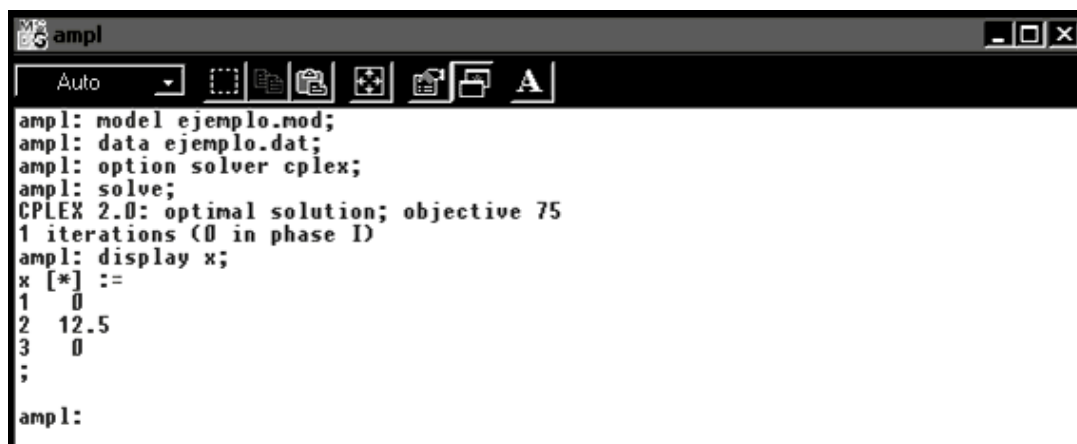


Figura 5.1: Funcionamiento de AMPL, Holmes (1992).

5.3 Entorno AMPL para MSDOS

AMPL para MSDOS, opera a través de una línea de comandos, en la cual se introducen las instrucciones línea a línea.

Desde el sistema operativo se puede acceder a este entorno llamando al programa: **Ampl.exe** y aparece la ventana de la Figura 5.2.



```

ampl: model ejemplo.mod;
ampl: data ejemplo.dat;
ampl: option solver cplex;
ampl: solve;
CPLEX 2.0: optimal solution; objective 75
1 iterations (0 in phase I)
ampl: display x;
x [*] :=
1 0
2 12.5
3 0
;
ampl:

```

Figura 5.2: Entorno de comandos de AMPL para MSDOS

Sin embargo, este entorno de comandos de AMPL no tiene facilidades para editar el modelo o los datos, o leer y escribir datos de/a un fichero fuente externo, para ello se debe apoyar en cualquier editor de código ASCII (por ejemplo el Bloc de Notas de Windows).

5.4 Resolución de Problemas en AMPL Plus

Cuando un programa de resolución está trabajando en optimizar un problema, lee un fichero de problema generado por AMPL y, cuando finaliza, escribe un fichero de solución el cual es leído por el procesador del lenguaje AMPL. Los valores solución para las variables de decisión son usadas para actualizar el resto de los valores en el problema, tales como los valores calculados de la función objetivo y de las restricciones. Todos estos valores se pueden visualizar bien con ayuda de los comandos estándar de AMPL (display en la ventana de comandos de AMPL Plus) o a través de las ventanas de visualización del modelo de AMPL Plus, Luque (2000).

Los modos más habituales de resolver problemas de optimización con AMPL Plus son:

1. Se le pide a AMPL usando el menú **Run**, que construya por un lado el fichero del modelo (submenú: **Build Model**), luego el fichero de datos (submenú: **Build Data**), y a continuación si no hubo ningún error, llamar al optimizador (submenú: **Solve Problem**).
2. Se puede crear un proyecto indicando quién es el fichero del modelo (extensión .mod) el fichero de datos (extensión .dat) y fichero de lotes (extensión .run) donde se indican algunas opciones y salidas particulares.
3. Otra forma es crear un fichero por lotes (extensión .run) que contenga todos los comandos necesarios para leer el modelo, leer los datos y otros comandos del lenguaje AMPL y ejecutarlo desde la ventana de comandos (por ejemplo: **C:\AMPL\ AMPL EJEMPLO1.RUN**).

Capítulo 6: EL MEDIAN SHORTEST PATH PROBLEM (MSPP)

En el presente capítulo se presenta, por una parte, una breve revisión bibliográfica del MSPP, y por otra parte, se presenta la formulación del MSPP utilizando dos modelos: formulación mediante programación lineal entera binaria y formulación basada en flujo multicommodity.

6.1 Revisión Bibliográfica

A partir de los años 80's, ha existido un creciente interés por el uso de técnicas multiobjetivo para el diseño de redes de transporte. Hansen (1980) y Martins (1984) formularon versiones multiobjetivo para el problema de la ruta más corta. Estos investigadores plantearon variados problemas bi-criterio, los que estaban basados sobre la distancia total de la ruta y las distancias individuales de los arcos. Estas formulaciones son interesantes y útiles en algunos casos, pero ninguna de ellas considera la accesibilidad a la ruta principal desde cualquier nodo con un objetivo implícito o explícito, Current *et al.* (1987).

La distancia total recorrida por los nodos demandantes para alcanzar el path principal fue usada como un objetivo por Halder y Majumder (1981) en un modelo para determinar la mejor separación de estaciones de una línea del metro. En su modelo, sin embargo, se asume que el path principal es dado, considerando que el diseño del path principal es la función principal del MSPP.

Los únicos modelos que consideraban *trade-offs* entre la longitud del path principal y la accesibilidad al path principal eran el problema de cobertura más corta y el problema de cobertura máxima. El criterio de costo de estas dos formulaciones es idéntico al MSPP; sin embargo, los criterios de accesibilidad son diferentes. En estas formulaciones, la accesibilidad se mide en términos de la distancia máxima cubierta.

Los autores, en Current *et al.* (1986), fueron los primeros en formular y resolver el MSPP. Estos autores diseñaron un algoritmo con el que encontraron un conjunto de soluciones no inferiores para el MSPP. El conjunto de soluciones no inferiores se ve reflejado en una curva de *trade-off* entre el costo de construcción del path principal y la accesibilidad al path principal. El algoritmo creado por estos investigadores es una buena técnica de solución del MSPP para problemas pequeños. En esos años se discutía acerca de la complejidad computacional de este tipo de problemas, y se presentaban soluciones algorítmicas para los problemas planteados.

Current y Schilling (1994), propusieron un procedimiento heurístico para resolver el MSPP generando buenas aproximaciones dentro de la frontera eficiente de resolución.

Labbé *et al.* (1998), complementan el estudio de Mesa y Boffey (1996), y estudian el MSPP bajo los criterios de minimizar costos y maximizar la accesibilidad al path principal, asignándoles una ponderación a cada criterio según su importancia relativa. Ellos proponen formulaciones de programación entera para los problemas dentro de las redes, y tratan de incluir la complejidad de todos los casos analizados.

Wang *et al.* (2002), estudian el MSPP bajo la condición de que algunas instalaciones ya están localizadas. Las instalaciones pueden ser localizadas en cualquier subconjunto de vértices. Cotas superiores e inferiores se proponen para modelos discretos y continuos.

Avella *et al.* (2005), introducen una formulación más ajustada, basada en una nueva familia de desigualdades válidas, llamadas Lifted Subtour Inequalities (Desigualdades de Ciclos Elevados). Para las Lifted Subtour Inequalities proponen un algoritmo de separación polinomial. Entonces introducen más familias de desigualdades derivadas de la investigación en relación al Problema del Vendedor Viajero Asimétrico. Esos resultados se usan para desarrollar un algoritmo Branch and

Cut que permite resolver problemas tamaño mediano en menos de 2 horas de procesamiento computacional.

Prasad y Nepal (2005) proponen un algoritmo alternativo para resolver el MSPP. El algoritmo está basado en el etiquetado de cada nodo en términos de múltiples objetivos, el que elimina ciclos, infactibilidad y paths dominados-extremos. La salida de este algoritmo es un conjunto de Pareto de paths óptimos con origen y destino predeterminados. Este algoritmo identifica el conjunto de soluciones no inferiores dejando la última palabra a los tomadores de decisión. Finalmente demuestran mediante un análisis de sensibilidad la eficiencia y las ventajas del algoritmo propuesto en términos de tiempos de ejecución computacional.

Lari *et al.* (2007) consideran dos metaheurísticas de búsqueda local: Tabú Search (TS) y Old Bachelor Acceptance (OBA). Estas investigadoras proponen un análisis comparativo de los desempeños de los dos procedimientos.

6.2 Formulación Matemática del MSPP

La formulación del MSPP considera los siguientes supuestos:

- La demanda existe sólo en los nodos y debe ser satisfecha.
- El costo de accesibilidad es calculado por la suma de todas las demandas de los nodos que no están en el path principal multiplicada por la distancia más corta de viaje hacia el nodo más cercano que se encuentre en el path principal.
- El flujo a lo largo de todos los arcos es no-capacitado.
- No existe el efecto congestión.
- Todos los costos de los arcos son no negativos.
- Las demandas y los costos son determinísticos.

La formulación matemática del MSPP puede desarrollarse desde estas 2 perspectivas:

- **Formulación mediante Programación Lineal Entera Binaria.** Esta formulación es la que se utilizó para resolver el MSPP en el presente proyecto de título.
- **Formulación mediante Flujo Multicommodity (Multibien).** Esta formulación se utilizó para comparar los resultados obtenidos con la formulación binaria del MSPP.

Para ambas formulaciones se utilizó la siguiente notación:

Parámetros:

- N : Conjunto de nodos
- $N \times N$: Conjunto de arcos tal que $i \neq j$
- A : Conjunto de arcos
- D_j : Demanda en el nodo j
- $c_{i,j}$: Costo de construcción de un arco que conecta el nodo i con el nodo j
- $T_{i,j}$: Distancia (o tiempo de viaje) por la ruta más corta desde el nodo i al nodo j
- s : Nodo Origen
- t : Nodo Destino
- Q : Subconjunto no vacío de N
- $|Q|$: Cardinalidad del conjunto Q

6.2.1 Formulación del MSPP mediante Programación Lineal Entera Binaria

Variables de Decisión

$$x_{ij} = \begin{cases} 1 & \text{si el arco } (i, j) \text{ está en el path principal} \\ 0 & \text{si no} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{si al nodo } j \text{ se le asigna el nodo } i \text{ que está en el path principal} \\ 0 & \text{si no} \end{cases}$$

El Modelo de Programación Lineal Entera

$$\text{Minimizar } Z = (Z_1, Z_2) \quad (6.1)$$

Sujeto a:

$$\sum_{j \in N: (s, j) \in A} x_{s, j} = 1 \quad (6.2)$$

$$\sum_{i \in N: (i, t) \in A} x_{i, t} = 1 \quad (6.3)$$

$$\sum_{i \in N: (i, j) \in A} x_{i, j} - \sum_{k \in N: (j, k) \in A} x_{j, k} = 0 \quad \forall j \in N : j \neq s, t \quad (6.4)$$

$$\sum_{j \in N: (i, j) \in A} x_{i, j} + \sum_{j \in N: (i, j) \in NxN} y_{i, j} = 1 \quad \forall j \in N : j \neq s, t \quad (6.5)$$

$$y_{i, j} - \sum_{h \in N: (h, i) \in A} x_{h, i} \leq 0 \quad \forall (i, j) \in NxN : i \neq s, t; j \neq s \quad (6.6)$$

$$\sum_{i \in Q, j \in Q: (i, j) \in A} x_{i, j} \leq |Q| - 1 \quad \forall Q \subseteq N : |Q| \geq 2 \quad (6.7)$$

$$x_{i, j} \in (0, 1) \quad \forall (i, j) \in A \quad (6.8)$$

$$y_{i, j} \in (0, 1) \quad \forall (i, j) \in NxN \quad (6.9)$$

Donde:

$$Z_1 = \sum_{(i, j) \in A} c_{i, j} x_{i, j} \quad (6.10)$$

$$Z_2 = \sum_{j \in N: (i, j) \in NxN} D_j T_{i, j} y_{i, j} \quad (6.11)$$

Las variables de decisión en la formulación del MSPP son $x_{i,j}$ e $y_{i,j}$. Si el arco (i,j) está sobre el path principal, que conecta al nodo origen con el nodo destino, entonces, $x_{i,j} = 1$, de otra forma, $x_{i,j} = 0$. Si el nodo j no está en el path, entonces, éste debe ser asignado a algún nodo i que está sobre el path principal. Si el nodo i es asignado al nodo j entonces, $y_{i,j} = 1$, de otra manera $y_{i,j} = 0$.

La función objetivo (6.1) representan las funciones objetivos involucradas en el problema. En las ecuaciones (6.10) y (6.11) se detallan ambas funciones objetivos, siendo Z_1 el costo total de construcción del path principal, y Z_2 el costo total de accesibilidad para llegar hacia los nodos que no están en el path principal.

Las restricciones (6.2) y (6.3), aseguran que el nodo origen y el nodo destino estén presentes en el MSPP. El conjunto de restricciones (6.4) corresponde al balance de flujo y asegura que si entra un arco a un nodo j en el MSPP, entonces un arco saldrá desde el nodo j , a menos que el nodo j sea el nodo origen o el nodo destino. El conjunto de restricciones (6.5) establece que todos los nodos deben ser cubiertos ya sea por el path principal o por una variable de asignación.

El conjunto de restricciones (6.6) señala que ningún nodo i puede realizar una asignación a un nodo j , cuando el nodo i no esté presente en el path principal.

El conjunto de restricciones (6.7) elimina los subtours indeseados que puedan aparecer en el MSPP, por consiguiente, asegura un path simple desde s a t .

Los conjuntos de restricciones (6.8) y (6.9) declaran que las variables $x_{i,j}$ e $y_{i,j}$ son binarias.

Si se consideran solamente los conjuntos de restricciones (6.2), (6.3) y (6.4) se podrá obtener la formulación del problema de la ruta más corta. Existen varios algoritmos extremadamente eficientes para resolver dicho problema. Los conjuntos de restricciones (6.5) y (6.6), sin embargo, impiden su aplicación directa al MSPP.

El carácter combinatorial que tiene este problema y las restricciones de balance de flujo influyen en la aparición de subtours con variables primarias. Por lo tanto, el conjunto de restricciones (6.7) es necesario para prevenir que los subtours aparezcan. Sin embargo, el conjunto de restricciones (6.7) puede ser extremadamente grande ya que contiene alrededor de $2^{|M|}$ restricciones para quebrar los subtours.

Debido al gran tamaño de estas restricciones y el gran número de variables binarias requeridas por el conjunto de restricciones (6.8) y (6.9) se resolvió el MSPP considerando sólo algunas restricciones del conjunto de restricciones (6.7), según comenzaron a aparecer los subtours en el proceso iterativo.

6.2.2 Formulación mediante Flujo Multicommodity del MSPP

La formulación presentada en la sección anterior requiere de un número exponencial de restricciones para eliminar los subtours. La forma usual de resolver estos modelos consiste en comenzar resolviendo utilizando un número reducido de restricciones y a medida que van apareciendo los subtours se incluyen las restricciones que los eliminan.

A continuación se presenta una formulación diferente que incorpora en el mismo modelo todas las restricciones que permiten eliminar los subtours. Cuando las soluciones (las variables) son binarias la presencia de uno o más subtours implica que se obtiene un grafo desconectado, de modo que la idea de esta formulación es precisamente obtener una solución cuyo grafo asociado esté completamente conectado.

La formulación basada en flujo multicommodity (multibien) consiste en enviar una unidad de flujo desde el nodo origen hacia cada uno de los restantes nodos de la red.

La idea es la siguiente, para cada nodo k se quiere tener una ruta que comienza en el nodo origen y termina en el nodo k . Esto asegura que siempre se tenga conectividad total.

Variables:

$$x_{ij} = \begin{cases} 1 & \text{si el arco } (i, j) \text{ está en el path principal} \\ 0 & \text{si no} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{si al nodo } j \text{ se le asigna el nodo } i \text{ que está en el path principal} \\ 0 & \text{si no} \end{cases}$$

$f_{i,j}^k$ es el flujo que pasa por el arco (i, j) en dirección al nodo k .

Formulación Matemática:

$$\text{Minimizar } Z = (Z_1, Z_2) \quad (6.12)$$

Sujeto a:

$$\sum_{j \in N: (s,j) \in A} x_{s,j} = 1 \quad (6.13)$$

$$\sum_{i \in N: (i,t) \in A} x_{i,t} = 1 \quad (6.14)$$

$$\sum_{i \in N: (i,j) \in A} x_{i,j} - \sum_{h \in N: (j,h) \in A} x_{j,h} = 0 \quad \forall j \in N : j \neq s, t \quad (6.15)$$

$$\sum_{i \in N: (i,j) \in A} x_{i,j} + \sum_{i \in N: (i,j) \in N \times N} y_{i,j} = 1 \quad \forall j \in N : j \neq s, t \quad (6.16)$$

$$\sum_{j \in N: (s,j) \in N \times N} f_{s,j}^k = 1 \quad \forall k \in N : k \neq s, t \quad (6.17)$$

$$\sum_{i \in N: (i,t) \in N \times N} f_{i,t}^k = 1 \quad \forall k \in N : k \neq s, t \quad (6.18)$$

$$\sum_{i \in N: i \neq k, t} f_{ij}^k = \sum_{h \in N: h \neq s} f_{jh}^k \quad \forall j \in N : j \neq s, t; \quad k \in N : k \neq s \quad (6.19)$$

$$f_{i,j}^k \leq x_{i,j} + y_{i,j} \quad \forall k \in N : k \neq s, t; (i, j) \in A : i \neq k \quad (6.20)$$

$$f_{i,j}^k \leq y_{i,j} \quad \forall k \in N : k \neq s, t; (i, j) \in N \times N \setminus A : i \neq k \quad (6.21)$$

$$x_{i,j} = (0,1) \quad \forall (i,j) \in A \quad (6.22)$$

$$y_{i,j} = (0,1) \quad \forall (i,j) \in NxN \quad (6.23)$$

$$f_{i,j}^k \geq 0 \quad \forall k \in N \neq s,t, (i,j) \in NxN : i \neq k \quad (6.24)$$

Donde:

$$Z_1 = \sum_{(i,j) \in A} c_{i,j} x_{i,j} \quad (6.25)$$

$$Z_2 = \sum_{j \in N : (i,j) \in NxN} D_j T_{i,j} y_{i,j} \quad (6.26)$$

Las funciones objetivos del MSPP se resumen en la expresión (6.12) y se especifican en las expresiones (6.25) y (6.26). Notar que no es necesario que la variable $f_{i,j}^k$ este presente en la función objetivo de esta formulación de flujo multicommodity.

Las restricciones (6.13) y (6.14) aseguran que exista un arco tanto en el nodo origen como en el nodo destino.

Por otra parte, el conjunto de restricciones (6.15) asegura que si entra un arco a un nodo j , entonces un arco saldrá desde el nodo j , a menos que el nodo j sea el nodo de inicio o el nodo de término.

El conjunto de restricciones (6.16) requiere que cada nodo esté dentro del path principal o bien que exista una asignación hacia un nodo que no esté dentro del path principal.

La restricción (6.17) asegura que exista una unidad de flujo que salga desde el nodo origen hacia el nodo k . Por su parte, la restricción (6.18), asegura que al nodo destino le llegue una unidad de flujo.

El conjunto de restricciones (6.19) corresponde al balance de flujos para cada una de las k rutas, en donde el flujo que entra a los arcos (i, j) en dirección al nodo k , debe ser igual al flujo de los nodos salientes de los arcos (i, j) en dirección al nodo k .

La restricción (6.20) asegura que el flujo que pasa por el arco (i, k) en dirección al nodo k , sea menor o igual, al total de las variables utilizadas para construir el path principal y las asignaciones respectivas.

Las restricción (6.21) asegura que la cantidad de variables utilizadas por los flujos que pasan por los arcos (i, j) en dirección al nodo k , sea menor a la cantidad de variables utilizadas en las asignaciones respectivas.

Finalmente, las restricciones (6.22), (6.23) y (6.24) señalan la naturaleza de las variables de decisión. Las restricciones (6.22) y (6.23) indican que tanto las variables de decisión $x_{i,j}$ e $y_{i,j}$ son binarias; la restricción (6.24) significa que la variable de decisión $f_{i,j}^k$ es no negativa y continua.

Capítulo 7: RESOLUCIÓN DEL MEDIAN SHORTEST PATH PROBLEM

En este capítulo se resuelve el MSPP con el fin de exponer el procedimiento completo de obtención de las diferentes soluciones del problema. El MSPP se resolvió mediante las dos formulaciones planteadas en el capítulo 6 para comparar la efectividad de ambas formulaciones. Para resolver el MSPP mediante la formulación binaria, se plantea un algoritmo de planos cortantes para la eliminación de subtours y la obtención de soluciones no inferiores.

7.1 Ejemplo de Aplicación del MSPP

Para mostrar el procedimiento que se propone en este trabajo para encontrar soluciones no inferiores en forma óptima del MSPP, se considera una red de 30 nodos y 108 arcos dirigidos. Posteriormente se muestran los resultados y estadísticas al aplicar el MSPP a problemas test de mayor tamaño.

7.1.1 Datos del Modelo

En la Tabla 7.1 se muestran todos los arcos de la red y sus respectivos costos $C(i, j)$. Notar que $C(i, j) = C(j, i)$ para todos los arcos (i, j) .

Arco			$C(i,j)$	Arco			$C(i,j)$	Arco			$C(i,j)$	Arco			$C(i,j)$
i	j	i		j	i	j		i	j						
1	3	12		1	2	10		1	4	9		2	4	6	
2	5	11		2	6	8		3	4	14		3	13	7	
3	14	10		4	5	16		4	12	18		4	13	17	
5	6	14		5	7	15		5	11	13		5	12	20	
6	7	3		6	8	4		7	8	5		7	10	18	
7	11	20		8	9	15		8	10	19		9	10	20	
9	26	18		9	27	19		10	11	25		10	25	16	
10	26	12		11	12	14		11	23	2		11	24	3	
11	25	13		12	13	25		12	18	21		12	23	15	
13	14	15		13	15	23		13	18	20		14	15	21	
15	16	19		15	18	13		16	17	2		16	18	23	
16	19	18		17	19	17		17	20	28		18	19	25	
18	23	24		19	20	27		19	22	25		19	23	30	
20	21	14		20	22	16		21	22	5		21	29	3	
21	30	2		22	23	13		22	24	12		22	29	4	
23	24	2		24	25	11		24	29	14		25	26	5	
25	29	12		26	27	3		26	28	5		26	29	8	
27	28	4		28	29	3		28	30	2		29	30	2	

Tabla 7.1: Costos $C(i, j)$ de los arcos de la red de 30 nodos

En la Figura 7.1 se muestra una red de 30 nodos, con los arcos que conectan los nodos y sus respectivos costos $C(i, j)$. Dado que los costos son simétricos, en la figura se muestran los arcos dirigidos en ambas direcciones, sin señalar la dirección de cada arco.

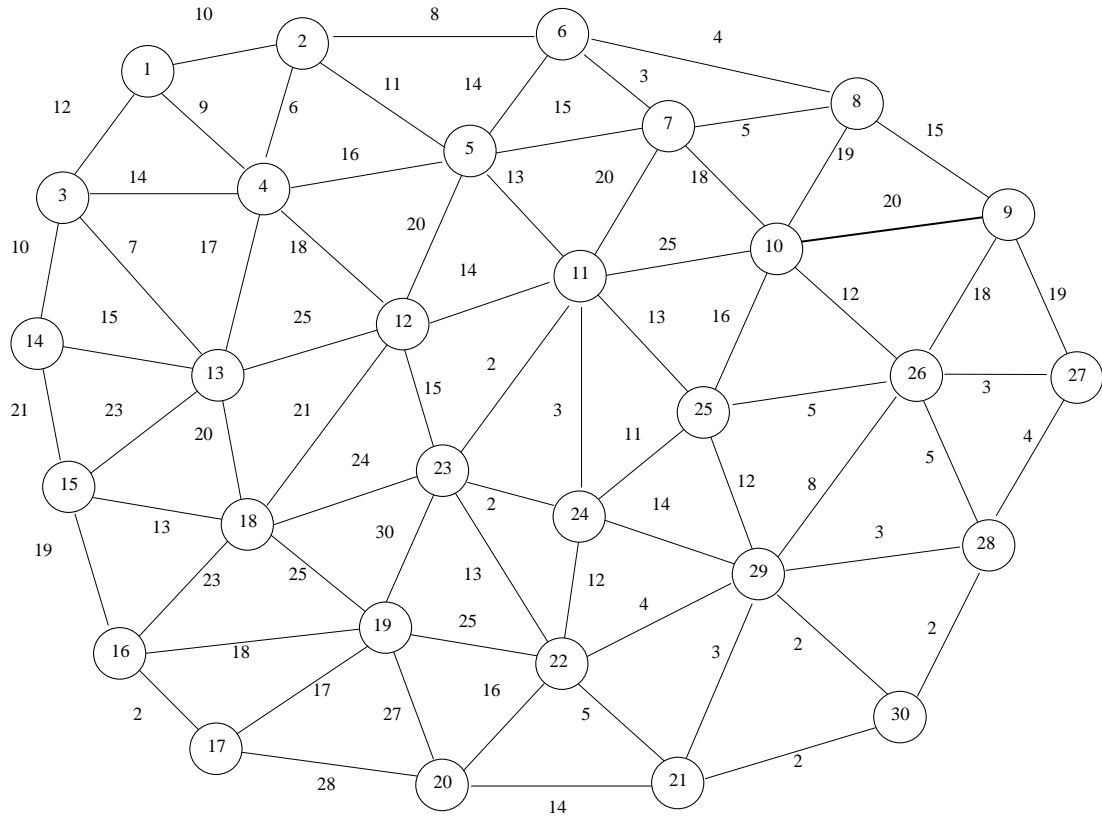


Figura 7.1: Red de 30 Nodos y 108 Arcos dirigidos

En la Tabla 7.2 se listan las demandas $D(j)$ de cada nodo. Las demandas fueron generadas aleatoriamente con el software AMPL a partir de la distribución Uniforme, con un número de “semilla” igual a 11213 (número primo), y dentro del intervalo [1000, 600000]. En el Anexo A se muestra el algoritmo que genera números aleatorios implementado en lenguaje AMPL.

Nodo	D(i)		Nodo	D(i)		Nodo	D(i)
1	491340		11	229730		21	305012
2	469047		12	38868		22	296130
3	292460		13	517553		23	484874
4	387197		14	59833		24	96947
5	184327		15	478037		25	198798
6	468441		16	275332		26	376854
7	254994		17	485852		27	339875
8	500316		18	338755		28	138695
9	145695		19	86799		29	466581
10	358888		20	32313		30	106749

Tabla 7.2: Demandas de cada nodo para la red de 30 nodos, generadas aleatoriamente.

La Tabla 7.2 muestra la matriz de distancias mínimas entre todos los pares de nodos. Esta tabla se generó con el software AMPL y se deriva de la Tabla 7.1. En el Anexo B se detalla el algoritmo de generación de la matriz de rutas más cortas.

	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
1	.	10	12	9	21	18	21	22	37	39	34	27	19	22	42	61	63	39	64	65	54	49	36	37	47	51	54	54	51	53
2	10	.	20	6	11	8	11	12	27	29	24	24	23	30	46	65	67	43	56	55	44	39	26	27	37	41	44	44	41	43
3	12	20	.	14	30	28	31	32	47	49	43	32	7	10	30	49	51	27	52	74	63	58	45	46	56	61	64	63	60	62
4	9	6	14	.	16	14	17	18	33	35	29	18	17	24	40	59	61	37	61	60	49	44	31	32	42	47	50	49	46	48
5	21	11	30	16	.	14	15	18	33	33	13	20	33	40	52	62	62	39	45	44	33	28	15	16	26	31	34	33	30	32
6	18	8	28	14	14	.	3	4	19	21	23	32	31	38	54	72	72	49	55	54	42	38	25	26	36	33	36	38	40	40
7	21	11	31	17	15	3	.	5	20	18	20	34	34	41	57	69	69	46	52	51	39	35	22	23	33	30	33	35	37	37
8	22	12	32	18	18	4	5	.	15	19	25	36	35	42	58	74	74	51	57	54	40	40	27	28	35	31	34	36	39	38
9	37	27	47	33	33	19	20	15	.	20	36	50	50	57	73	71	69	60	55	41	27	30	36	34	23	18	19	23	26	25
10	39	29	49	35	33	21	18	19	20	.	25	39	52	59	64	65	63	51	49	35	21	24	27	27	16	12	15	17	20	19
11	34	24	43	29	13	23	20	25	36	25	.	14	39	53	39	49	49	26	32	31	20	15	2	3	13	18	21	20	17	19
12	27	24	32	18	20	32	34	36	50	39	14	.	25	40	34	44	46	21	45	44	33	28	15	17	27	32	35	34	31	33
13	19	23	7	17	33	31	34	35	50	52	39	25	.	15	23	42	44	20	45	69	58	53	40	42	52	57	60	59	56	58
14	22	30	10	24	40	38	41	42	57	59	53	40	15	.	21	40	42	34	58	70	73	68	55	56	66	71	74	73	70	72
15	42	46	30	40	52	54	57	58	73	64	39	34	23	21	.	19	21	13	37	49	55	50	37	39	50	55	58	56	53	55
16	61	65	49	59	62	72	69	74	71	65	49	44	42	40	19	.	2	23	18	30	44	43	47	49	58	53	52	48	47	46
17	63	67	51	61	62	72	69	74	69	63	49	46	44	42	21	2	.	25	17	28	42	42	47	49	56	51	50	46	45	44
18	39	43	27	37	39	49	46	51	60	51	26	21	20	34	13	23	25	.	25	52	42	37	24	26	37	42	45	43	40	42
19	64	56	52	61	45	55	52	57	55	49	32	45	45	58	37	18	17	25	.	27	30	25	30	32	41	37	36	32	29	31
20	65	55	74	60	44	54	51	54	41	35	31	44	69	70	49	30	28	52	27	.	14	16	29	28	28	23	22	18	17	16
21	54	44	63	49	33	42	39	40	27	21	20	33	58	73	55	44	42	42	30	14	.	5	18	17	14	9	8	4	3	2
22	49	39	58	44	28	38	35	40	30	24	15	28	53	68	50	43	42	37	25	16	5	.	13	12	16	12	11	7	4	6
23	36	26	45	31	15	25	22	27	36	27	2	15	40	55	37	47	47	24	30	29	18	13	.	2	13	18	21	19	16	18
24	37	27	46	32	16	26	23	28	34	27	3	17	42	56	39	49	49	26	32	28	17	12	2	.	11	16	19	17	14	16
25	47	37	56	42	26	36	33	35	23	16	13	27	52	66	50	58	56	37	41	28	14	16	13	11	.	5	8	10	12	12
26	51	41	61	47	31	33	30	31	18	12	18	32	57	71	55	53	51	42	37	23	9	12	18	16	5	.	3	5	8	7
27	54	44	64	50	34	36	33	34	19	15	21	35	60	74	58	52	50	45	36	22	8	11	21	19	8	3	.	4	7	6
28	54	44	63	49	33	38	35	36	23	17	20	34	59	73	56	48	46	43	32	18	4	7	19	17	10	5	4	.	3	2
29	51	41	60	46	30	40	37	39	26	20	17	31	56	70	53	47	45	40	29	17	3	4	16	14	12	8	7	3	.	2
30	53	43	62	48	32	40	37	38	25	19	19	33	58	72	55	46	44	42	31	16	2	6	18	16	12	7	6	2	2	.

Tabla 7.3: Matriz de Distancias Mínimas para la red de 30 nodos

7.1.2 Aparición y Eliminación de Subtours en el MSPP utilizando el Método de Planos Cortantes

La formulación binaria del MSPP sin el conjunto de restricciones (6.7) provoca que se produzcan subtours en la red, los que derivan en soluciones infactibles. En este estudio se utiliza el método de planos cortantes para eliminar los subtours y encontrar soluciones no inferiores en forma óptima.

En la Figura 7.2 se muestra la solución obtenida en una iteración intermedia realizada para obtener una solución no inferior. En esta figura se observan claramente los diferentes subtours que aparecen en la red de 30 nodos:

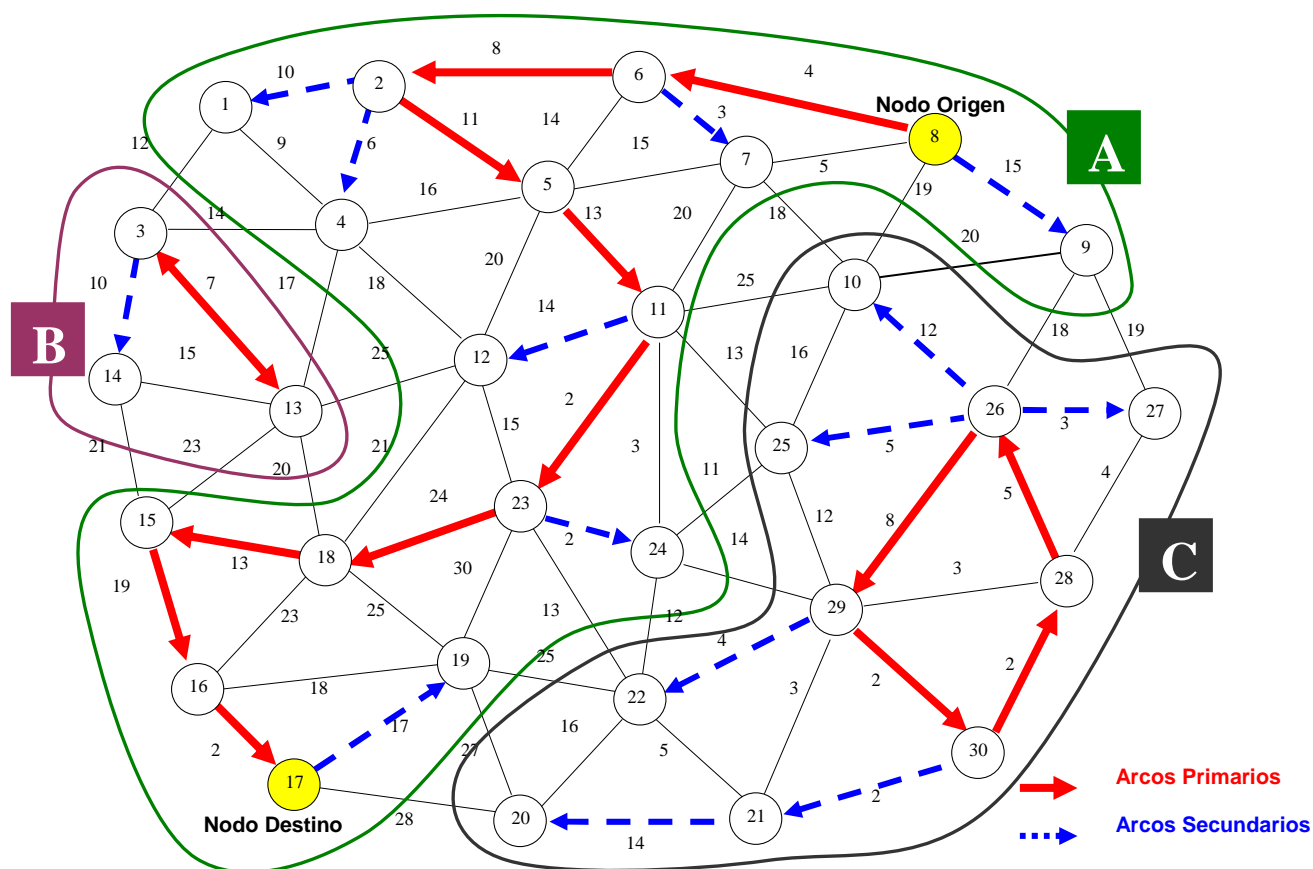


Figura 7.2: Aparición de subtours en el MSPP

Todos los arcos marcados con una flecha en la Figura 7.2, ya sea con una línea continua (color rojo) o línea segmentada (color azul), están asociadas con las variables primarias y secundarias, respectivamente, que tienen valor uno en la solución correspondiente a una iteración intermedia. Un caso particular se observa en la asignación (30,20) y (30,21). La asignación (30,20) es directa y sigue el camino más corto para llegar al nodo 20, por lo que debe pasar por el nodo 21. Sin embargo, esta asignación no significa, en ningún caso, que el nodo 21 realice una asignación al nodo 20.

Además, en la Figura 7.2 se identifican tres grandes conjuntos aislados que contienen nodos que están conectados por arcos primarios o por arcos secundarios. El conjunto A contiene los nodos que conforman el path que une al nodo origen con el nodo destino (path origen-destino) y los nodos que son asignados a este tramo. Este conjunto se considera un subtour pues, a pesar de no contener ningún ciclo, no contiene a todos los nodos de la red. Los conjuntos B y C contienen nodos que están conectados por arcos primarios y por arcos secundarios. Estos conjuntos serán llamados *Subtours Maximales*. Cada uno de estos subtours maximales posee ciclos formados por nodos primarios, los que serán denominados *Subtours Minimales*. La Figura 7.2 presenta dos subtours minimales, formados por los nodos 3 y 23, y por los nodos 26, 28, 29 y 30.

Notar que siempre es posible encontrar un conjunto que contenga al nodo origen y al nodo destino debido a las restricciones de balance de flujo (6.4) y dado que se resuelve el problema binario, por ejemplo, el conjunto A. Si este conjunto contiene todos los nodos de la red, entonces se ha encontrado la solución óptima.

En la primera etapa de eliminación de subtours, se eliminan los subtours minimales mediante el conjunto de restricciones (6.7), esto es:

$$\sum_{i \in Q, j \in Q: (i,j) \in A} x_{i,j} \leq |Q| - 1 \quad (\forall Q \subseteq N : |Q| \geq 2) \quad (6.7)$$

Con el propósito de tener conectividad en una solución del MSPP, cada conjunto aislado de nodos (un subtour maximal) induce una restricción que corta esta solución. La idea es identificar este conjunto de nodos y generar una restricción que obligue a que debe ingresarse un arco a este conjunto, primario o secundario, desde el conjunto exterior de nodos (su complemento, B^C y C^C en la Figura 7.2). Se hace notar que un subtour maximal es un conjunto de nodos que no contiene al nodo origen, al nodo destino, ningún nodo del path que los une y ningún nodo conectado por un arco secundario. Denotando por SY al conjunto de nodos aislados, la desigualdad válida que se utiliza para obtener conectividad es la siguiente:

$$\sum_{i \in SY^C, j \in SY: (i,j) \in A} x_{i,j} + \sum_{i \in SY^C, j \in SY} y_{i,j} \geq 1 \quad \forall SY \subseteq N : s, t \notin SY \quad (7.1)$$

De igual manera, el conjunto de nodos formado por el nodo origen, nodo destino, nodos sobre el path que une a estos dos nodos y todos los nodos que son conectados por un arco secundario también inducen a una restricción que corta esta solución. Ahora, la idea es identificar este conjunto de nodos y construir una restricción que obligue a que debe salir un arco, primario o secundario, desde este conjunto hacia el exterior para obtener conectividad.

Denotando por SP al conjunto de estos nodos, la restricción que se utiliza tiene la siguiente forma:

$$\sum_{i \in SP, j \in SP^C: (i,j) \in A} x_{i,j} + \sum_{i \in SP, j \in SP^C} y_{i,j} \geq 1 \quad \forall SP \subseteq N, s, t \in SP \quad (7.2)$$

Ejemplo:

- SY : Subconjunto no vacío de N . Este subconjunto está conformado por nodos asociados a los subtours minimales. En la Figura 1, existen dos conjuntos SY . $SY_2 = \{3,13,14\}$ y $SY_3 = \{10,20,21,22,25,26,27,28,29,30\}$

- SP : Subconjunto no vacío de N . Este subconjunto está conformado por los nodos del path origen-destino, y sus asignaciones respectivas. En la Figura 1 el conjunto SP contiene los siguientes nodos:
 $SP = \{1, 2, 4, 5, 6, 7, 8, 9, 11, 12, 14, 15, 16, 17, 18, 19, 23, 24\}$

Mediante el uso de las restricciones (6.7), (7.1) y (7.2) se logra romper con los subtours de la iteración de la Figura 7.2. La iteración siguiente se muestra en la Figura 7.3:

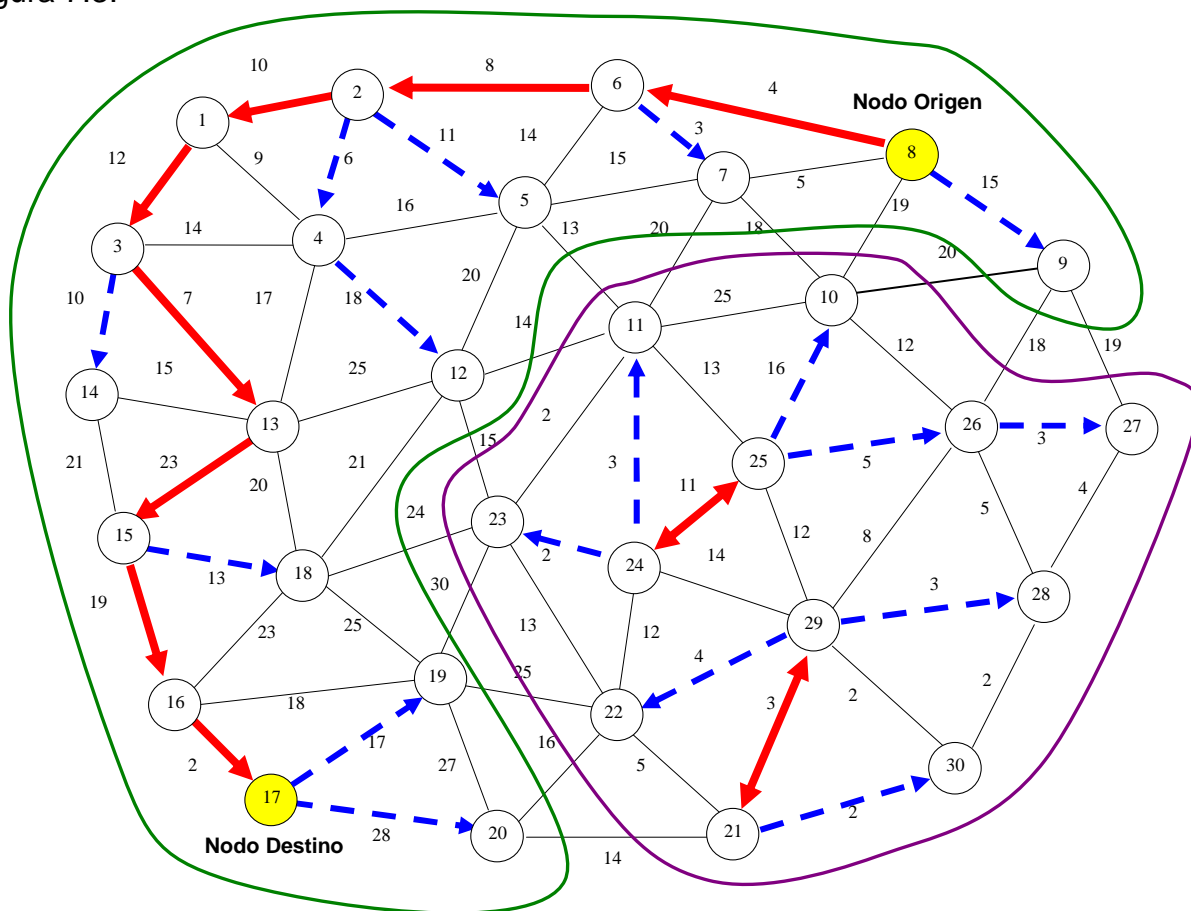


Figura 7.3: Efecto de la aplicación del método de planos cortantes

En la iteración de la Figura 7.3 se muestra el efecto de introducir en el modelo las restricciones (6.7), (7.1) y (7.2). Se observa que desde el conjunto A (subtour formado por el path origen-destino) de la Figura 7.2, salió el arco primario (1,2) hacia el conjunto B, con lo que de esa forma se logró la conectividad entre estos dos

conjuntos. También se deduce que al conjunto C fue ingresado el arco secundario (17,20) desde el conjunto A.

Los subtours minimales que se formaron en la iteración mostrada en la Figura 7.2 entre los nodos 3 y 13, 26, 28, 19 y 30, fueron eliminados con el conjunto de restricciones (6.7).

También se observa en la Figura 7.3, aparecen nuevos subtours, los que deberán ser eliminados de manera análoga al procedimiento descrito anteriormente, a través de la aplicación de restricciones que “quiebren” los subtours que aparezcan en las diferentes iteraciones. El procedimiento iterativo continuará hasta que se no aparezcan más subtours en la solución y, por lo tanto, se obtenga la solución óptima.

7.1.3 Eliminación de Arcos Secundarios

Con el objetivo de mejorar el proceso de búsqueda de soluciones para el MSPP, se presenta un procedimiento para eliminar algunos arcos secundarios que no formarán parte de ninguna solución no inferior. La eliminación de tales arcos secundarios permite reducir considerablemente el tamaño del problema a resolver y en consecuencia reducir los tiempos de proceso.

El procedimiento consiste en establecer un radio de búsqueda de nodos candidatos, con la condición de que la distancia más corta desde un nodo j a un nodo i , sea menor o igual al valor de la ruta más corta entre el nodo origen y el nodo i y al valor de la ruta más corta entre el nodo i y el nodo destino. Esto es, se elimina el arco (i,j) si se verifica la siguiente desigualdad:

$$d(i, j) \geq \min\{d(s, i), d(i, t)\} \quad (7.3)$$

Donde, $d(a,b)$ es el valor de la ruta más corta entre el nodo a y el nodo b . Luego, cuando se verifica la desigualdad el arco (i,j) se elimina pues siempre será más conveniente asignar el nodo i al nodo origen o al nodo destino.

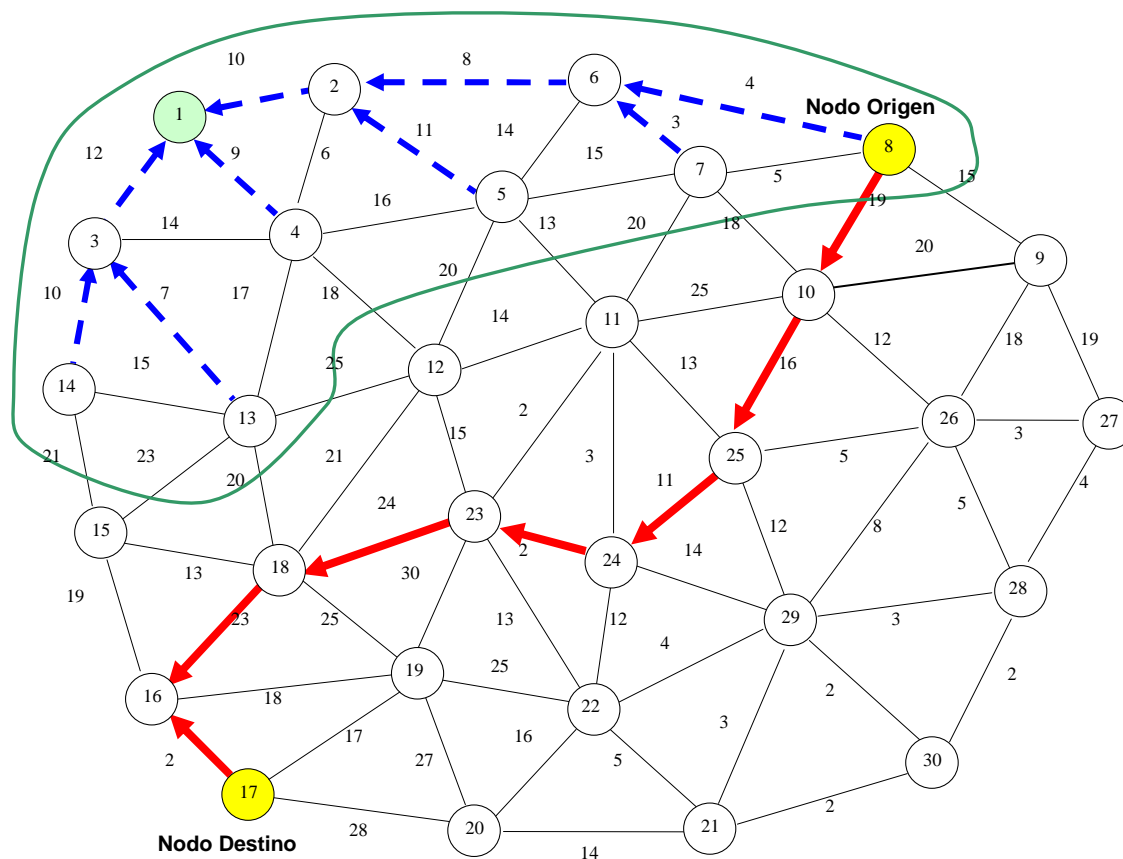


Figura 7.4: Eliminación de Arcos Secundarios para el MSPP

En la Figura 7.4 se muestran gráficamente todos los arcos secundarios candidatos a ser asignados al nodo 1. En la Tabla 7.4 se presentan explícitamente todos los arcos secundarios candidatos y sus respectivas distancias mínimas que se dirigen hacia el nodo 1:

Arco	(2,1)	(3,1)	(4,1)	(5,1)	(6,1)	(7,1)	(8,1)	(13,1)	(14,1)
RMC	10	12	9	21	18	21	22	19	22

Tabla 7.4: Arcos Secundarios Candidatos

El valor de la distancia mínima entre el nodo origen y el nodo 1 es 22, mientras que el valor de la distancia mínima entre el nodo destino y el nodo 1 es 63. Entonces el procedimiento elige el mínimo valor entre estas dos distancias mínimas (en este caso 22), y cualquier arco secundario que se dirija hacia el nodo 1, debe tener una distancia mínima menor para poder ser un arco secundario candidato, tal como se observa en la Tabla 7.4. En la Figura 7.4 el path principal pasa por el nodo origen, excluyendo todos los demás nodos del radio de arcos candidatos, por lo tanto la asignación que se realiza va desde el nodo 8 al nodo 1. Si no se hiciera este procedimiento, todos los arcos que pueden llegar el nodo 1 serían arcos secundarios candidatos para estar en la solución óptima.

El total de arcos secundarios de la red de 30 nodos es de 812 arcos, mientras que al utilizar el procedimiento de eliminación de arcos, la red quedó con un total de 278 arcos secundarios candidatos, con una reducción de un 65,5%, lo que favorece al proceso de búsqueda, y por lo tanto, hace más rápido el proceso iterativo.

7.2 Búsqueda de las Soluciones No Inferiores del MSPP

En el conjunto de soluciones no inferiores del MSPP, existen dos soluciones que son fundamentales y a la vez, son las más fáciles de encontrar: el Path Hamiltoniano (PH) y la Ruta Más Corta (RMC). Estas dos soluciones son “extremas” ya que, por una parte, el path Hamiltoniano representa la alternativa menos costosa que pasa por todos los nodos, y por otra parte, la ruta más corta es la alternativa menos costosa para construir el path principal, pero es a la vez la más costosa en términos de accesibilidad. Además, a partir de estas dos soluciones no inferiores extremas, se puede obtener un conjunto aproximado de soluciones no inferiores a través del método NISE.

A continuación se presenta la metodología utilizada para obtener el path Hamiltoniano, la ruta más corta, y cómo, a partir de estas dos soluciones no inferiores extremas, se obtuvo un conjunto de soluciones no inferiores (a través del método NISE) que van a conformar la curva de *trade-off* entre el costo total del path principal y la accesibilidad a éste path. Además, se propone una metodología para encontrar soluciones no inferiores que no fueron encontradas por el método NISE, dentro de un intervalo relevante para el tomador de decisiones.

7.2.1 Obtención del Path Hamiltoniano

El path Hamiltoniano corresponde a la solución menos costosa que pasa por todos los nodos, y es a la vez la solución más costosa desde el punto de vista de la construcción del path principal, ya que en ésta solución, no existen costos de accesibilidad. Por ejemplo, si se tratara de construir una línea del metro, y los nodos fueran ciudades, entonces la línea del metro pasaría por todas las ciudades, lo que claramente implicaría un costo muy alto para el operador del sistema. En la Figura 7.5 se presenta el path Hamiltoniano en la red de 30 nodos:

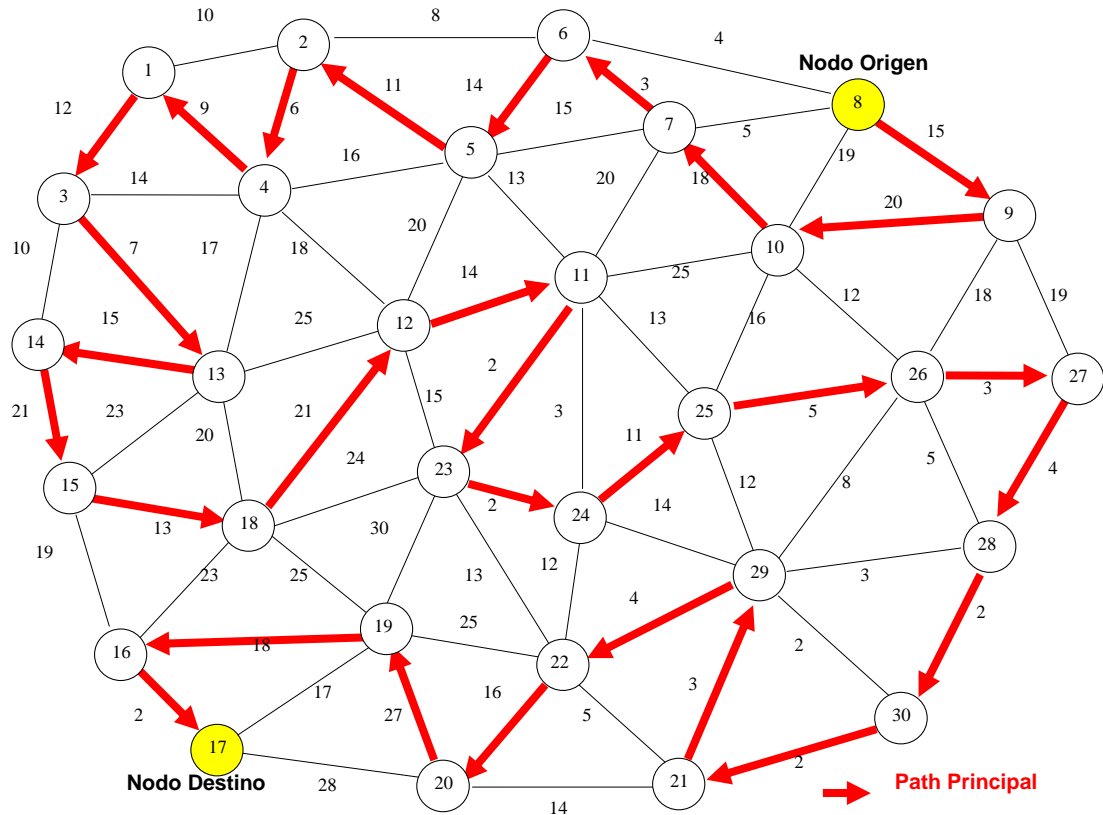


Figura 7.5: El Path Hamiltoniano para la red de 30 nodos

El PH fue generado con el método de las restricciones (capítulo 3), en donde el costo de accesibilidad fue considerado como una restricción con valor constante igual a 0, debido a que en este caso, se restringe el problema al hecho de que no existan costos de accesibilidad al path principal. Por otra parte, la función objetivo del MSPP corresponde al costo de construcción del path principal, y al ser única, se puede resolver como si fuera un problema de un solo objetivo, mediante el método Simplex. Las expresiones matemáticas utilizadas fueron las siguientes:

$$\text{Minimizar} \quad Z_1 = \sum_{(i,j) \in A} c_{i,j} x_{i,j} \quad (7.4)$$

$$\text{Sujeto a:} \quad Z_2 = \sum_{(i,j) \in N \times N} D_j T_{i,j} y_{i,j} \leq 0 \quad (7.5)$$

$$(6.2) - (6.9)$$

En la generación del PH aparecieron subtours, los que fueron eliminados utilizando el método de planos cortantes, visto en la sección 7.1.2. En total se eliminaron 19 subtours.

Al resolver el problema, el costo de accesibilidad es igual a 0 ($Z_2 = 0$), en tanto que el costo de construcción del path principal resulta ser $Z_1 = 300$. Se puede observar claramente que el path principal recorre todos los nodos de la red.

7.2.2 Obtención de la Ruta Más Corta

La Ruta Más Corta (RMC) corresponde a la opción menos costosa desde el punto de vista de la construcción del path principal, sin embargo es la opción más costosa en términos de accesibilidad. El operador que construya la línea del metro minimizará sus costos de construcción, pero las distancias que deberán recorrer los usuarios del metro serán muy grandes; en este caso los costos sociales serían muy elevados. En la Figura 7.6 se muestra la ruta más corta del MSPP para la red de 30 nodos.

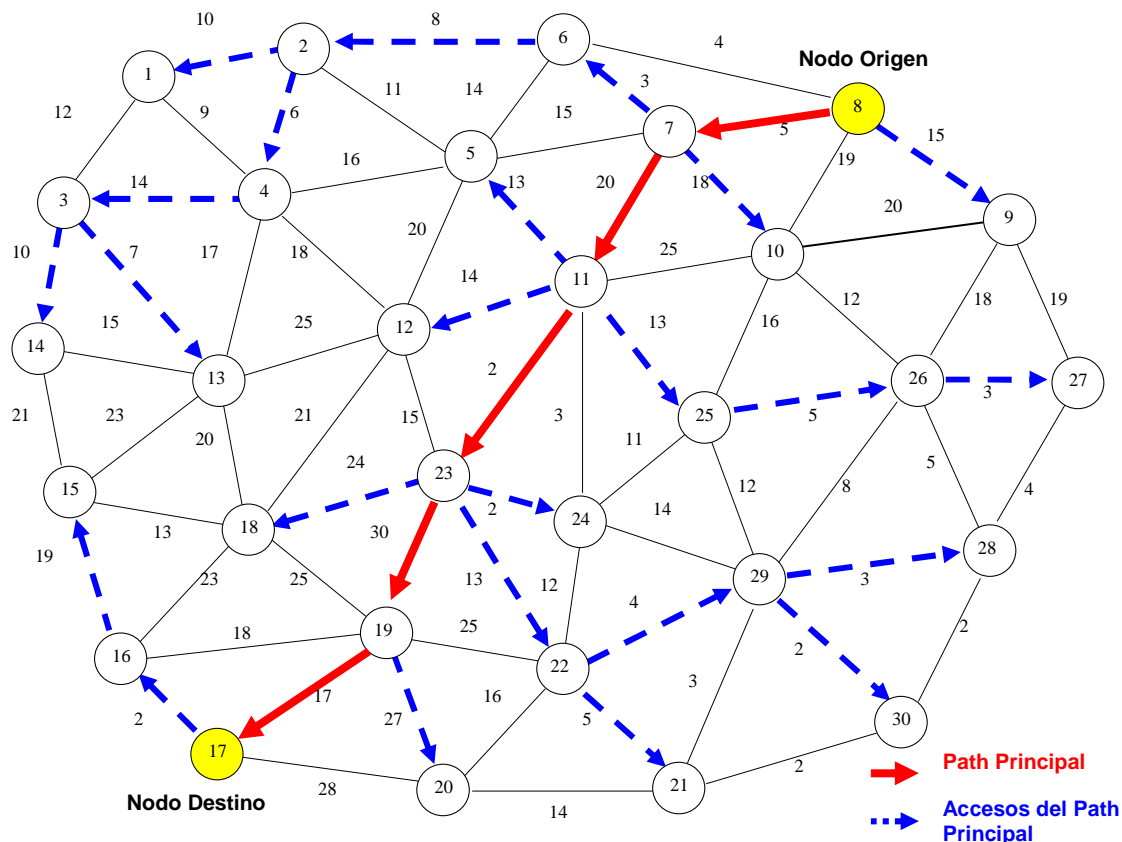


Figura 7.6: Ruta Más Corta para la red de 30 nodos

La RMC, al igual que el PH, fue obtenida con el método de las restricciones (capítulo 3), en donde el costo de construcción del path principal fue considerado como una restricción con valor constante igual a 74 (valor obtenido de la matriz de distancias mínimas de la Tabla 7.3), debido a que, en este caso, se restringe el problema al hecho de que no existan costos de accesibilidad a partir del path principal. Por otra parte, la función objetivo del MSPP corresponde al costo total de accesibilidad al path principal, y al ser única, se puede resolver como si fuera un problema de un solo objetivo, mediante el método Simplex. Las expresiones matemáticas utilizadas fueron las siguientes:

$$\text{Minimizar} \quad Z_2 = \sum_{j \in N: (i,j) \in N \times N} D_j T_{i,j} y_{i,j} \quad (7.6)$$

$$\text{Sujeto a:} \quad Z_1 = \sum_{(i,j) \in A} c_{i,j} x_{i,j} \leq 74 \quad (7.7)$$

$$(6.2) - (6.9)$$

El costo total de construcción del path principal de la ruta más corta es $Z_1 = 74$, y el costo de accesibilidad es de $Z_2 = 121820272$.

7.2.3 Obtención de las Soluciones No Inferiores y la Curva de *Trade-Off*

A partir de la obtención de los costos de construcción y accesibilidad tanto del path Hamiltoniano como de la ruta más corta, se da inicio a la búsqueda de un conjunto aproximado de soluciones no inferiores del problema. En la Figura 7.7 se grafican las dos soluciones extremas del MSPP, las que son no inferiores. El PH corresponde al punto $(Z_1, Z_2) = (300, 0)$. Por otra parte, la RMC es el punto $(Z_1, Z_2) = (74, 121820272)$.

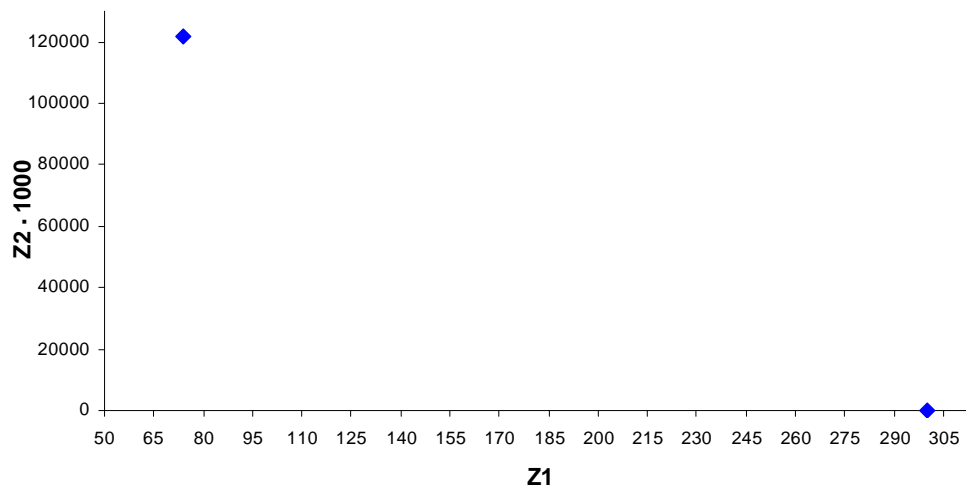


Figura 7.7: Gráfico del Path Hamiltoniano (PH) y la Ruta Más Corta (RMC)

Para obtener un conjunto aproximado de soluciones no inferiores se recurrió al método NISE, cuyos fundamentos teóricos fueron dados a conocer en el Capítulo 3.

7.2.4 Aplicación del Método NISE para obtener Soluciones No Inferiores

El método NISE maximiza cada objetivo individualmente, los que pertenecen a dos puntos en el espacio objetivo. La pendiente que conecta esos dos puntos se usa para obtener pesos para el problema ponderado.

El proceso del uso de los pesos en el problema ponderado se basa en la pendiente de la línea segmentada que conecta los dos puntos adyacentes S_i y S_{i+1} en la aproximación actual. La pendiente tiene la siguiente expresión matemática:

$$P = \frac{Z_2(S_i) - Z_2(S_{i+1})}{Z_1(S_{i+1}) - Z_1(S_i)} \quad (7.8)$$

Para el ejemplo demostrativo, se obtiene la pendiente reemplazando los valores obtenidos con el path Hamiltoniano y la ruta más corta:

$$P = \frac{121820272 - 0}{300 - 74} = 539027.75$$

En el gráfico (Figura 7.8) la línea segmentada corresponde a la pendiente entre estos dos puntos.

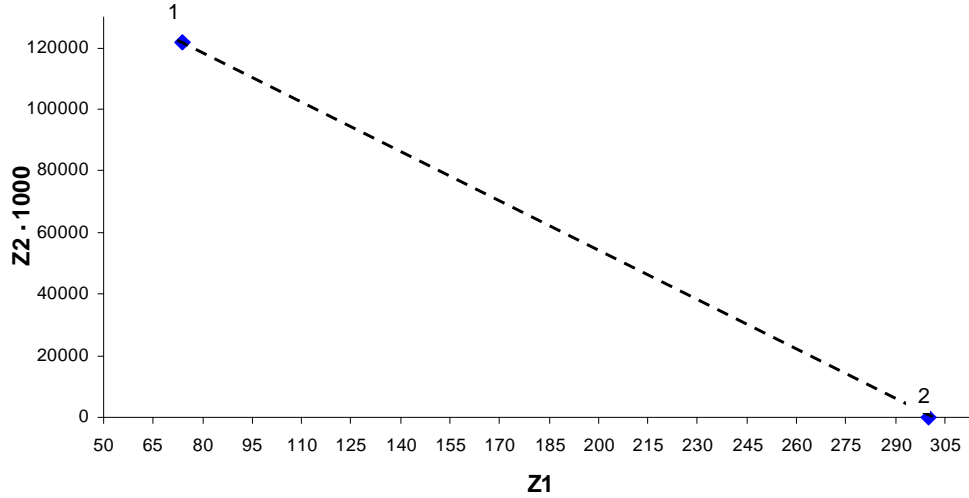


Figura 7.8: Pendiente de la recta obtenida con la solución del PH y de la RMC

Con esta pendiente, se obtiene el valor del peso que será utilizado para la generación de una solución no inferior.

Del Capítulo 3 se puede obtener la siguiente expresión matemática que servirá para la obtención de una nueva solución no inferior:

$$\text{Maximizar } Z(x_1, \dots, x_n; i; i+1) = \frac{Z_2(S_i) - Z_2(S_{i+1})}{Z_1(S_{i+1}) - Z_1(S_i)} Z_1(x_1, \dots, x_n) + Z_2(x_1, \dots, x_n) \quad (3.8)$$

Reemplazando la ecuación (3.8) en el MSPP, se obtiene la siguiente expresión matemática:

$$\text{Minimizar } Z = P \cdot \sum_{(i,j) \in A} c_{i,j} x_{i,j} + \sum_{j \in N: (i,j) \in N \times N} D_j T_{i,j} y_{i,j} \quad (7.9)$$

Por lo tanto, con esta nueva función objetivo única se puede resolver el MSPP como un problema de programación lineal con un solo objetivo, y por consiguiente, obtener una nueva solución no inferior, la que se ve graficada en la Figura 7.9:

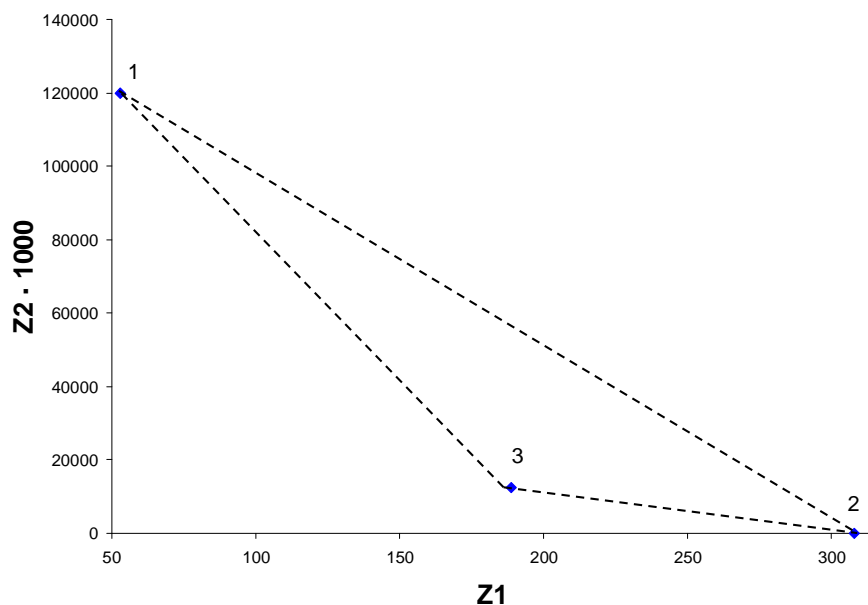


Figura 7.9: Obtención de una Solución No Inferior a partir del PH y la RMC

En la Figura 7.9 se observa que el punto $(Z_1, Z_2) = (155, 18532272)$ corresponde la solución no inferior. Idéntico procedimiento se realiza para obtener las siguientes soluciones, en el que se trazan nuevas pendientes para buscar soluciones no inferiores entre los puntos $(74, 121820272)$ y $(155, 18532272)$, y entre los puntos $(155, 18532272)$ y $(300, 0)$.

A continuación (Figura 7.10) se muestra una nueva solución no inferior, obtenida entre los puntos $(74, 121820272)$ y $(155, 18532272)$.

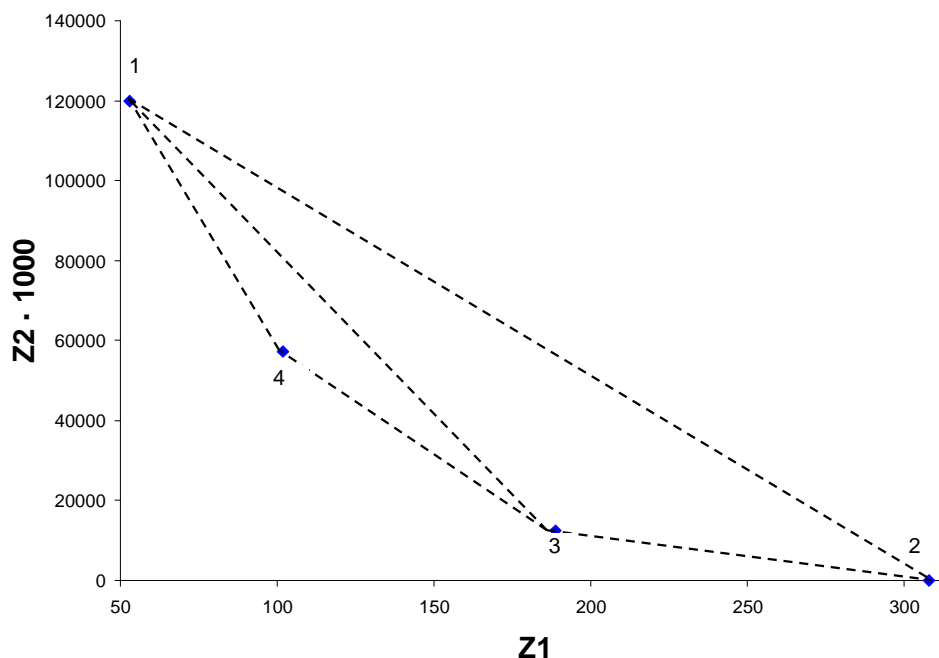


Figura 7.10: Obtención de la una nueva Solución No Inferior

En la Figura 7.10 la nueva solución no inferior se encuentra en el punto $(Z_1, Z_2) = (93, 72376375)$. De forma análoga se trazan nuevas pendientes entre la nueva solución no inferior obtenida y las anteriores, esto es entre $(74, 121820272)$ y $(93, 72376375)$, y entre los puntos $(93, 72376375)$ y $(155, 18532272)$. De esta forma se tendrán nuevas soluciones hasta que entre dos puntos no existan más soluciones, y en cuyo caso, el método NISE elegirá uno de los dos puntos como solución no inferior.

El método NISE utiliza los dos puntos extremos (path Hamiltoniano, PH y ruta más corta, RMC) para comenzar la búsqueda del conjunto aproximado de soluciones no inferiores. Posteriormente el método NISE busca los intervalos en donde podrían estar las soluciones no inferiores a través del cálculo de pendientes.

A continuación se muestran las soluciones no inferiores obtenidas al utilizar el método NISE.

7.3 Soluciones No Inferiores para la red de 30 nodos

En la Tabla 7.5 se muestran las 20 soluciones no inferiores obtenidas utilizando el método NISE:

Solución No Inferior	Costo de Construcción (Z1)	Costo de Asignación (Z2)
N°1	74	121820272
N°2	300	0
N°3	155	18532272
N°4	93	72376375
N°5	78	96540600
N°6	76	101988675
N°7	105	59813631
N°8	149	22977535
N°9	183	6249862
N°10	179	7118406
N°11	164	13845403
N°12	161	15383538
N°13	162	14865028
N°14	169	11522221
N°15	182	6443756
N°16	244	1594864
N°17	217	3070447
N°18	192	5255872
N°19	257	996534
N°20	278	452382

Tabla 7.5: Soluciones No Inferiores encontradas con el método NISE

En la Figura 7.12, el gráfico muestra la curva de *trade-off* entre el costo total de construcción del path principal (Z_1) versus el costo total de asignación (Z_2) para las 20 soluciones no inferiores de la red de 30 nodos.

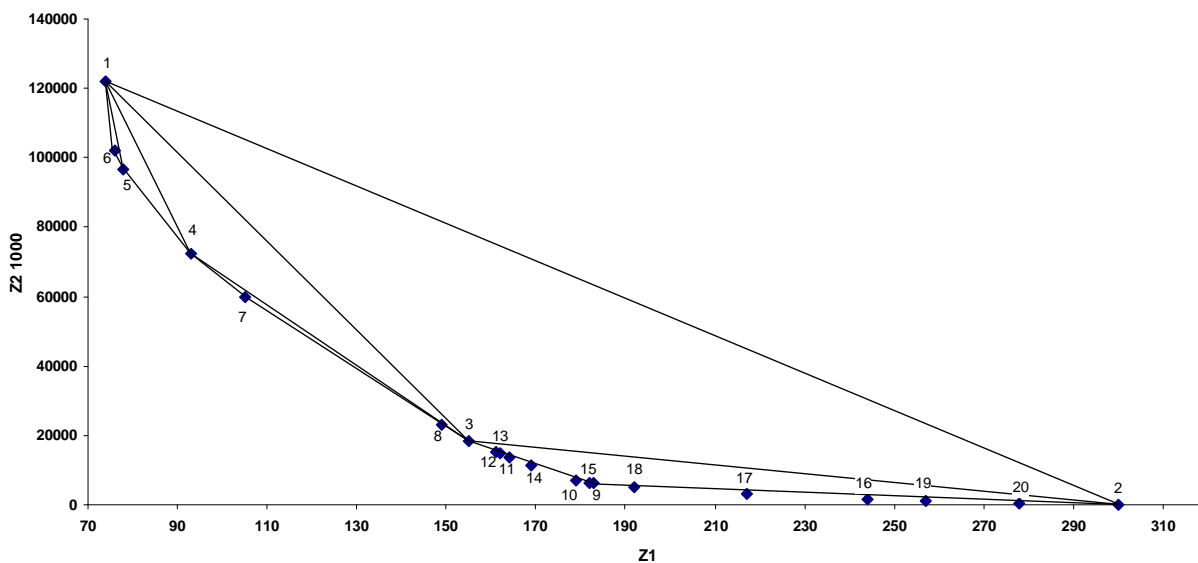


Figura 7.11: Representación Gráfica del Conjunto de Soluciones No Inferiores

De la Figura 7.11 se desprende información importante y fundamental en el proceso de decisión, ya que con esta curva de *trade-off* el tomador de decisiones tiene un abanico de posibilidades para poder elegir la mejor alternativa según su presupuesto. El tomador de decisiones es quien finalmente determina que beneficios se pueden obtener al escoger una alternativa por sobre otra.

A continuación, en las Figuras 7.12 y 7.13 se presentan dos representaciones gráficas de las soluciones no inferiores N°4 y N°7 de la red de 30 nodos respectivamente.

7.3.1 Solución No-Inferior N°4

La Figura 7.12 muestra la solución no inferior N°4 obtenida por el método NISE:

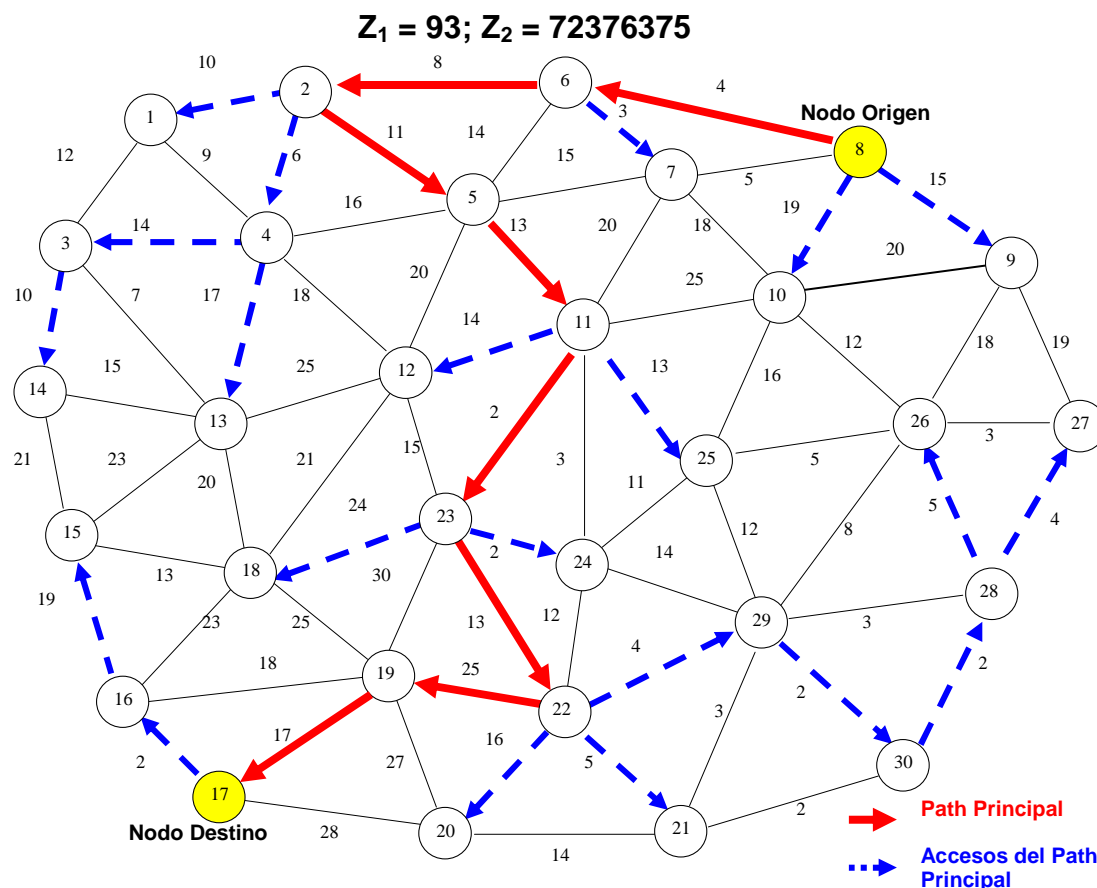


Figura 7.12: Solución óptima de la Solución No Inferior N°4

El costo de construcción del path principal en la solución no inferior N°4 es de 93, mientras que el costo de accesibilidad del path principal es de 72376375. Para encontrar esta solución, el algoritmo de planos cortantes eliminó los subtours que aparecieron en el proceso iterativo mediante 39 cortes (restricciones) y 80 iteraciones. El tiempo total de CPU de las iteraciones del algoritmo fue de 340.031 segundos.

7.3.2 Solución No Inferior N°7

A modo de ilustrar de mejor manera la obtención de soluciones no inferiores, a continuación se muestra la solución no inferior N°7 (Figura 7.13).

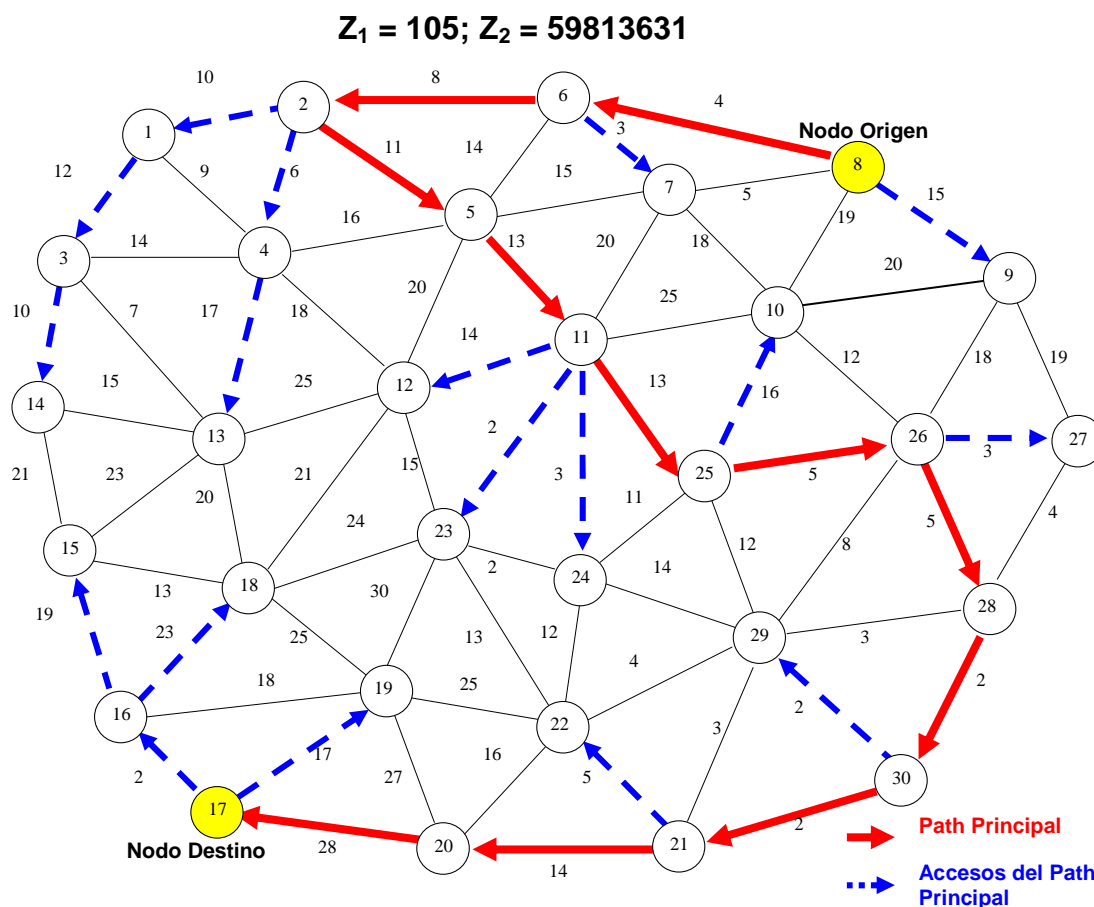


Figura 7.13: Solución No Inferior N°7

El costo de construcción del path principal en la solución no inferior N°7 es de 105, mientras que el costo de accesibilidad del path principal es de 59813631. El algoritmo de planos cortantes realizó 106 cortes (restricciones) y 138 iteraciones para obtener esta solución no inferior libre de subtours. El tiempo total de proceso de CPU de las iteraciones del algoritmo fue de 3770.83 segundos.

7.3.3 Selección de Soluciones No Inferiores

Es importante hacer notar que en el conjunto aproximado de SNI generado por el método NISE, no todas las SNI encontradas son muy relevantes. Por ejemplo, la SNI N°19, es una solución casi extrema y prácticamente no tiene costos de accesibilidad. Lo que interesa finalmente para el tomador de decisiones, es tener un conjunto aproximado de SNI en que se consideren soluciones no inferiores claves, tales como el path Hamiltoniano, la ruta más corta, la solución intermedia entre estas dos SNI, y las soluciones adyacentes a la solución intermedia.

En la Figura 7.14 se muestra un método de búsqueda de SNI, en que se toman sólo algunas SNI de las encontradas por el método NISE de la Figura 7.11.

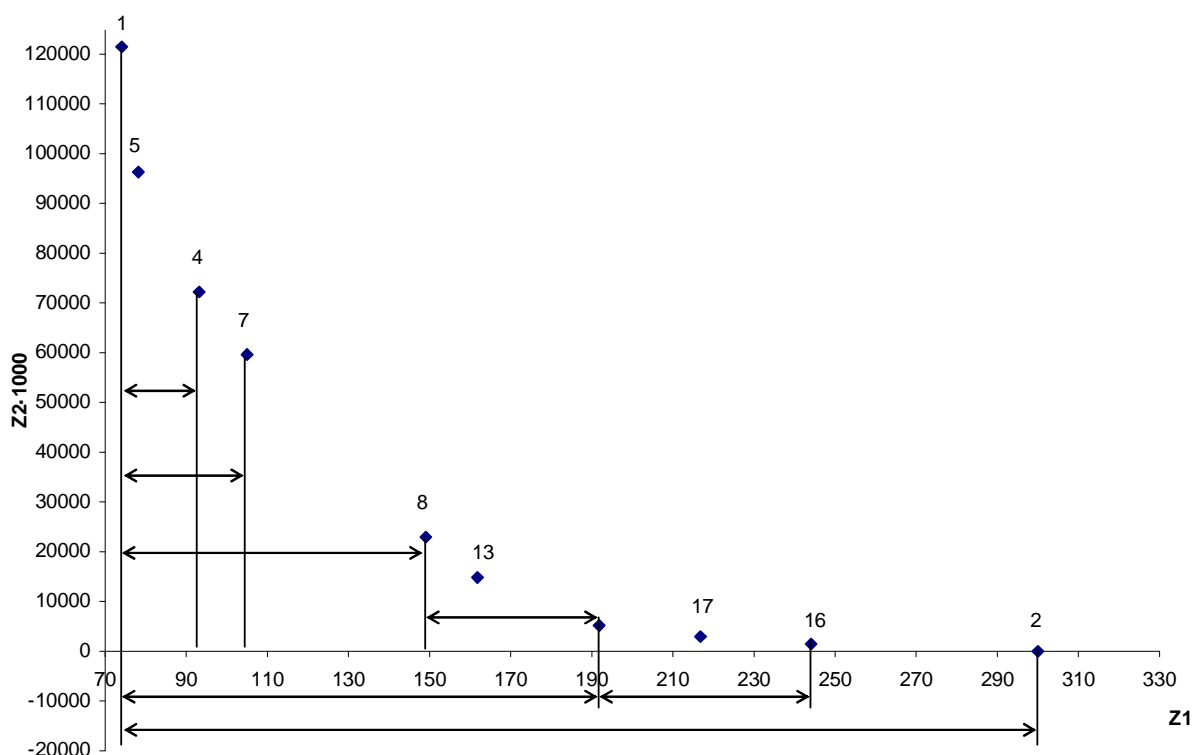


Figura 7.14: Selección de Soluciones No Inferiores para el MSPP

En este conjunto reducido de 10 SNI, primero se consideraron las dos soluciones extremas del intervalo (RMC y PH). Luego se seleccionó la SNI N°19 (192, 5255872) que es la solución intermedia entre estos dos puntos extremos. Posteriormente, se eligió la SNI N°8 (149, 22977535), que es aproximadamente la solución intermedia entre la SNI N°19 y la SNI N°2. Este procedimiento continúa hasta evaluar un cierto número de intervalos, obteniendo un número reducido de SNI.

Por otra parte, es posible que el método NISE no encuentre todas las SNI. En tal caso se puede aplicar el método de las restricciones para obtener otras SNI que se encuentren dentro del intervalo definido por dos soluciones obtenidas con el método NISE, Cohon (1978). En la Figura 7.14 se observa que entre las SNI N°7 y N°8, existe un intervalo en que el método NISE no encontró más SNI. Sin embargo, es posible encontrar otras SNI en dicho intervalo. Por ejemplo, en este rango se puede aplicar una de las siguientes restricciones (en forma independiente):

$$\left. \begin{aligned} Z_1 &= \sum_{(i,j) \in A} c_{i,j} x_{i,j} \leq 120 \\ Z_1 &= \sum_{(i,j) \in A} c_{i,j} x_{i,j} \leq 135 \\ Z_1 &= \sum_{(i,j) \in A} c_{i,j} x_{i,j} \leq 110 \end{aligned} \right\} \quad (7.10)$$

Por su parte, la función objetivo del MSPP estaría conformada solamente por los costos de accesibilidad:

$$\text{Minimizar} \quad Z_2 = \sum_{(i,j) \in N \times N} D_j T_{i,j} y_{i,j} \quad (7.11)$$

Sujeto a:

$$(6.2) - (6.9)$$

$$(7.10)$$

Utilizando las restricciones (7.10), cada una por separada, entre las SNI N°7 y N°8 se obtuvieron las tres SNI que se detallan en la tabla 7.6 y se muestran gráficamente en la Figura 7.15.

Solución No Inferior	Costo de Construcción (Z1)	Costo de Asignación (Z2)
N°7	105	59813,631
N°8	149	22977,535
N°21	110	55970434
N°22	120	48658344
N°23	135	41175493

Tabla 7.6: Búsqueda de tres SNI en un intervalo mediante el Método de las Restricciones

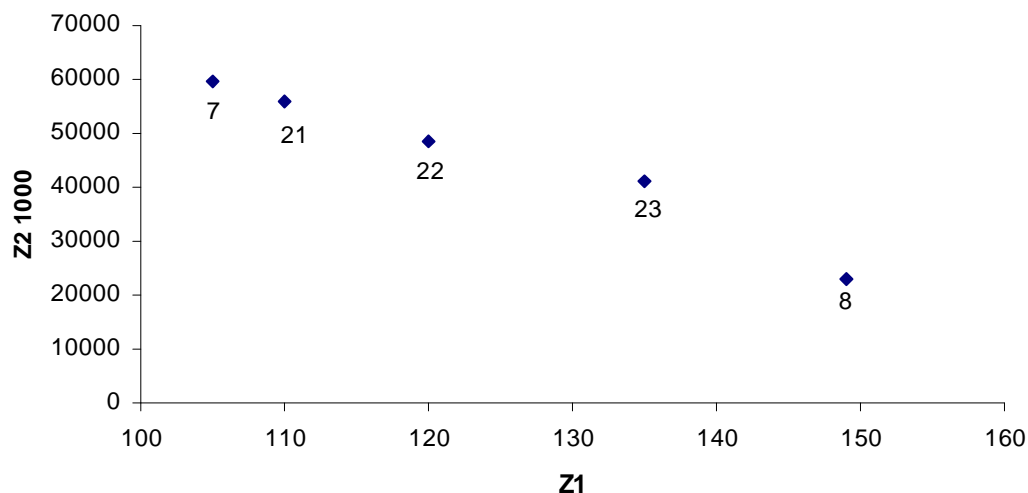


Figura 7.15: Búsqueda de tres SNI en un intervalo mediante el Método de las Restricciones

Este proceso de búsqueda puede continuar si el tomador de decisiones lo requiere. Él finalmente debe decidir cuánto va a gastar en construir el path principal, según su presupuesto.

7.4 Experiencias Computacionales para la red de 30 nodos

En esta sección se exponen los resultados de las experiencias computacionales que se realizaron con la red de 30 nodos, considerando que los nodos de origen y destino son 8 y 17 respectivamente. Primero se muestran los tiempos de CPU del MSPP resuelto mediante la formulación binaria (6.1.1), considerando con y sin eliminación de arcos secundarios. Posteriormente se exponen los tiempos de CPU del MSPP resuelto mediante la formulación basada en flujo multicommodity (6.1.2), sólo con eliminación de arcos secundarios. Finalmente se muestra una tabla comparativa de ambos tiempos de CPU

7.4.1 Experimento 1: Obtención de SNI sin reducción de arcos secundarios

En el primer experimento, se aplicaron las restricciones de eliminación de subtour (6.7), (7.1) y (7.2) las que quiebran todos los subtours que aparecen en el proceso iterativo. En este experimento no se consideró la eliminación de arcos secundarios. En la Tabla 7.7 se muestran los resultados de esta experiencia computacional:

SNI	Z1	Z2	Restricciones	Iteraciones	Subtours	TCPU
N°1	74	121820272	353	1	0	0.04688
N°2	300	0	77	6	23	0.10938
N°3	155	18532272	1073	66	720	945.84966
N°4	93	72376375	1112	80	39	495.31878
N°5	78	96540600	1112	81	0	16.036903
N°6	76	101988675	1112	82	0	2.402415
N°7	105	59813631	1268	138	156	3544.0151
N°8	149	22977535	1280	144	12	576.7357
N°9	183	6249862	1286	149	6	601.44626
N°10	179	7118406	1286	150	0	12.604881
N°11	164	13845403	1286	151	0	27.502976
N°12	161	15383538	1286	152	0	36.769436
N°13	162	14865028	1286	154	0	59.405181
N°14	169	11522221	1286	157	0	58.047972
N°15	182	6443756	1286	160	0	22.339344
N°16	244	1594864	1291	164	5	2.028013
N°17	217	3070447	1291	165	0	0.218401
N°18	192	5255872	1291	166	0	0.312002
N°19	257	996534	1291	170	0	0.717605
N°20	278	452382	1291	172	0	0.312002

Tabla 7.7: Resultados del Experimento 1

Por su parte, en la Figura 7.16 se muestran los tiempos de CPU de las SNI encontradas:

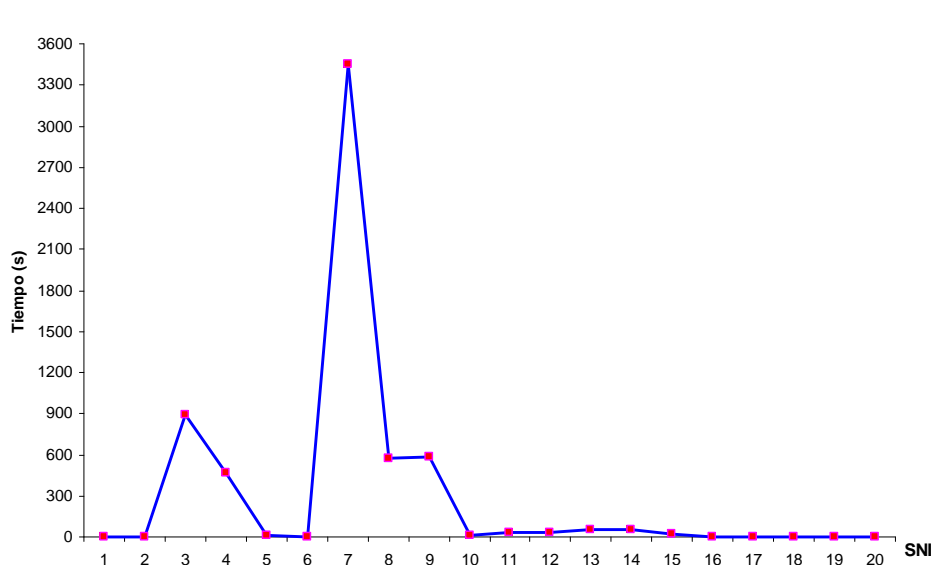


Figura 7.16: Tiempos de CPU de las SNI en el Experimento 1

El tiempo total de CPU para este experimento fue de 6402.22 segundos. Se observa que en las SNI N°3 y N°7, se registraron los mayores tiempos de CPU. Observar en la sexta columna Tabla 7.7 que en las SNI N°3 y N°7 se registró un número considerable de Subtours en el proceso iterativo. Notar que en todas las soluciones obtenidas después de la SNI N°7 se utilizan las mismas restricciones de eliminación de subtours obtenidas previamente.

7.4.2 Experimento 2: Obtención de SNI con reducción de arcos secundarios

Con el objetivo de mejorar el proceso de búsqueda de las SNI, y por ende los tiempos de CPU, se efectuó un segundo experimento, que consistió en realizar una “limpieza” de arcos secundarios que nunca van a estar presentes en ninguna solución no inferior. En la Tabla 7.8 se muestran los resultados obtenidos en el experimento 2:

SNI	Z1	Z2	Restricciones	Iteraciones	Subtours	TCPU
N°1	74	121820272	353	1	0	0.04688
N°2	300	0	77	6	23	0.10938
N°3	155	18532272	584	66	231	565.89063
N°4	93	72376375	623	80	39	340.03125
N°5	78	96540600	623	81	0	8.64063
N°6	76	101988675	623	82	0	1.6875
N°7	105	59813631	729	138	106	3770.82813
N°8	149	22977535	791	144	62	433.01563
N°9	183	6249862	797	149	6	0.125
N°10	179	7118406	797	150	0	6.0625
N°11	164	13845403	797	151	0	12.609375
N°12	161	15383538	797	152	0	15.03125
N°13	162	14865028	797	154	0	29.71875
N°14	169	11522221	797	157	0	39.34375
N°15	182	6443756	797	160	0	22.9375
N°16	244	1594864	802	164	5	1.46875
N°17	217	3070447	802	165	0	0.10938
N°18	192	5255872	802	166	0	0.15625
N°19	257	996534	802	170	0	0.453125
N°20	278	452382	802	172	0	0.1875

Tabla 7.8: Resultados del Experimento 2

En la Figura 7.17 se exponen los tiempos de CPU en función del número de las SNI encontradas:

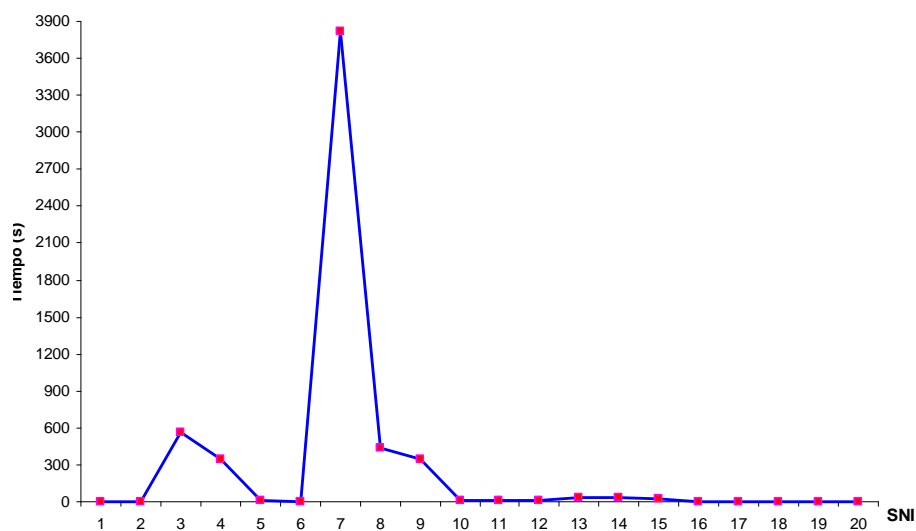


Figura 7.17: Tiempos de CPU de las SNI en el Experimento 2

En este experimento, el tiempo total de CPU para obtener las 20 soluciones no inferiores fue de 5248.45 segundos. En comparación a los resultados obtenidos con el experimento 2, se produce una disminución en los tiempos de búsqueda de las SNI aproximadamente en un 18%. Además se observa en la Tabla 7.8 que el número de restricciones de eliminación de subtours utilizadas en este experimento fue notoriamente menor en comparación con el experimento 1.

En la Tabla 7.8, al igual que en la Tabla 7.7, se observa que en todas las soluciones obtenidas después de la SNI N°8 se utilizan las mismas restricciones de determinadas previamente. Esto explica que los tiempos de CPU a partir de la SNI N°8 sean considerablemente menores.

7.4.3 Experimento 3: Obtención de SNI mediante la Formulación Basada en Flujo Multicommodity

Con el objetivo de comparar los resultados obtenidos mediante la formulación binaria del MSPP, se realizaron experimentos con la formulación mediante flujo multicommodity. En este caso se utiliza el mismo método propuesto por Cohon (1978) para encontrar las SNI. En el Anexo D, se muestra el modelo y el algoritmo de resolución del MSPP a través de la formulación mediante flujo multicommodity. En esta formulación basada en flujo multicommodity también se consideró el procedimiento de eliminación de arcos secundarios. En la Tabla 7.9 se muestran los resultados obtenidos en el experimento 3:

SNI	Z1	Z2	Restricciones	Iteraciones	TCPU
N°1	74	121820272	4724	1	0.04688
N°2	300	0	4724	2	0.10938
N°3	155	18532272	4724	1	0.484375
N°4	93	72376375	4724	2	0.37500
N°5	78	96540600	4724	3	0.296875
N°6	76	101988675	4724	4	0.203125
N°7	105	59813631	4724	8	1.375
N°8	149	22977535	4724	10	2.09375
N°9	183	6249862	4724	13	2.65625
N°10	179	7118406	4724	14	0.375
N°11	164	13845403	4724	15	0.34375
N°12	161	15383538	4724	16	0.359375
N°13	162	14865028	4724	18	0.79688
N°14	169	11522221	4724	21	1.234375
N°15	182	6443756	4724	24	1.03125
N°16	244	1594864	4724	27	1.296875
N°17	217	3070447	4724	28	0.70313
N°18	192	5255872	4724	29	0.39063
N°19	257	996534	4724	33	2.18750
N°20	278	452382	4724	35	1.53125

Tabla 7.9: Resultados del Experimento 3

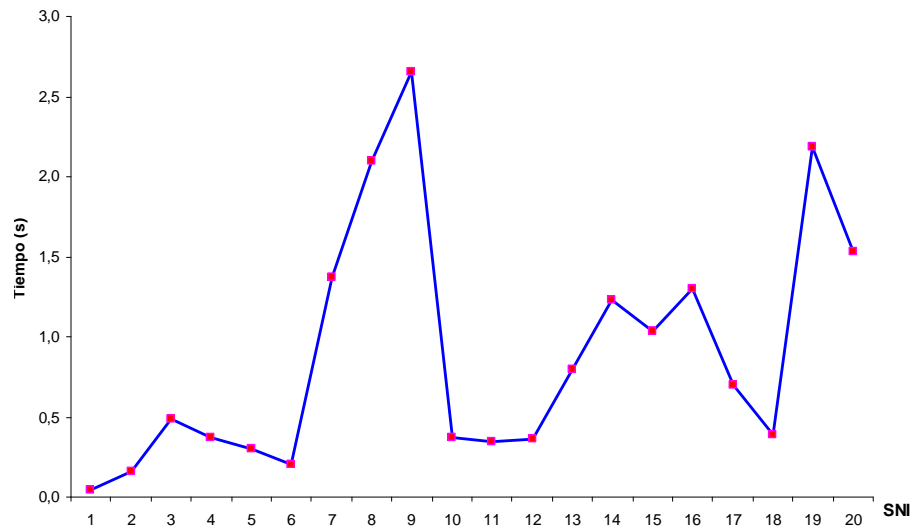


Figura 7.18: Tiempos de CPU de las SNI en el Experimento 3

Al utilizar la formulación basada en flujo multicommodity para resolver el MSPP, el tiempo total de CPU fue de 17,89 segundos. En la Figura 7.18 se visualiza que los tiempos de CPU fueron notablemente inferiores a los tiempos de los experimentos anteriores para todas las SNI.

En la cuarta columna de Tabla 7.9 se observa que existe un gran número de restricciones en esta formulación (4724 restricciones) y es el mismo para todas las SNI. Sin embargo, esto no afecta la efectividad de la formulación basada en flujo multicommodity para la red de 30 nodos.

7.4.4 Análisis de los Resultados de los Experimentos

De las Figuras 7.16 y 7.17 se observa que en las primeras dos soluciones no inferiores (path Hamiltoniano y ruta más corta) el tiempo de CPU es muy pequeño (menos de 1 segundo). Sin embargo, el tiempo de proceso para encontrar la solución no inferior N°3 es notablemente mayor, debido a que el algoritmo encuentra muchos subtours en esta solución, y por lo tanto realiza un número considerable de cortes para encontrar la solución no inferior libre de subtours. El mismo caso se da en la búsqueda de la solución inferior N°7, en donde el tiempo de CPU es bastante alto, debido al gran número de cortes e iteraciones realizadas. Los tiempos de CPU de las siguientes soluciones no inferiores son notablemente menores que el tiempo de CPU de la SNI N°7, ya que el algoritmo aplica todas las restricciones encontradas en la SNI N°7 a las siguientes SNI, y si llegara a encontrar más subtours, entonces el algoritmo realiza nuevos cortes con el fin de encontrar las SNI restantes libres de subtours.

En la Figura 7.19 se muestra una comparación de los experimentos realizados:

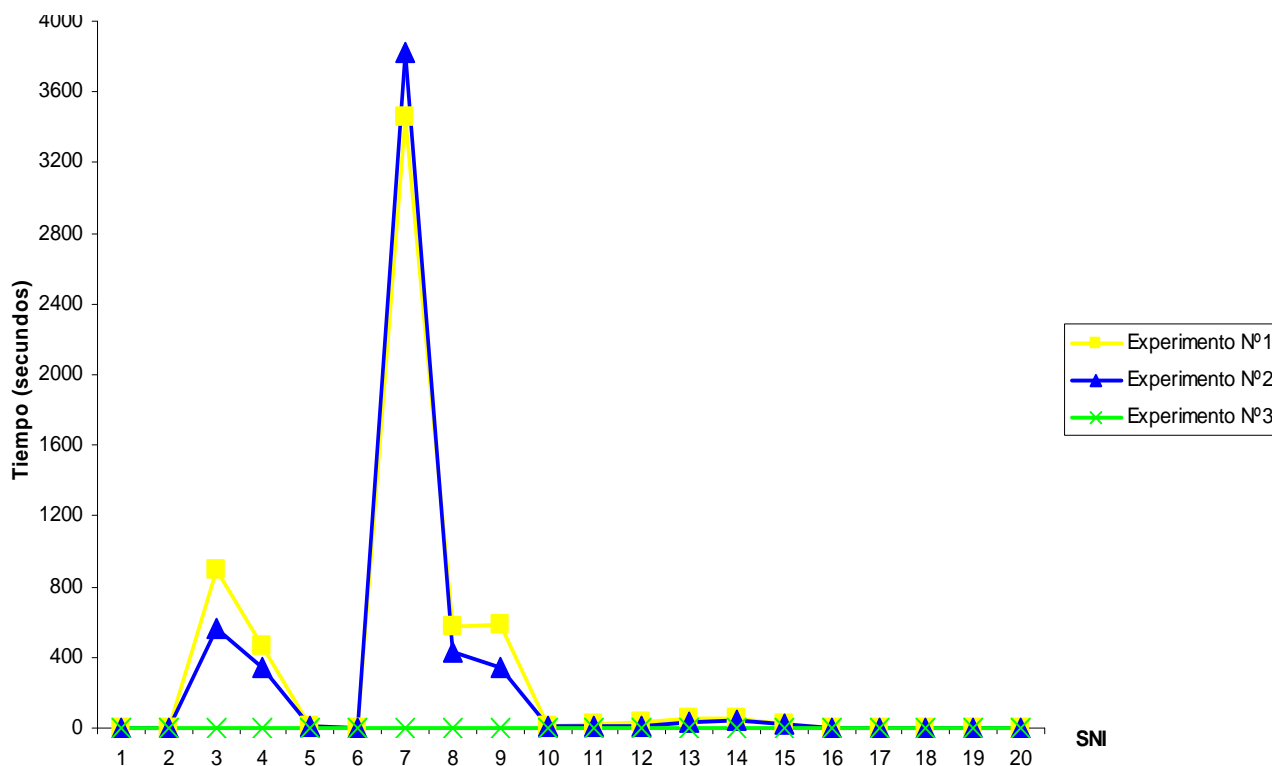


Figura 7.19: Comparación de Tiempos de CPU de los Experimentos Realizados

Se observa que los tiempos de CPU de la formulación mediante flujo multicommodity resultaron ser considerablemente menores en comparación a los obtenidos con la formulación binaria. Esto se explica principalmente a que la red analizada es pequeña, por lo que el número de restricciones que tiene la formulación basada en flujo multicommodity no afectan la efectividad en cuanto a tiempos de CPU.

7.5 Procedimiento propuesto para resolver el MSPP

Considerando todo lo anterior, en esta sección se resume el procedimiento para resolver el MSPP mediante la formulación binaria:

- Paso 1* Determinar un nodo origen y un nodo destino para el MSPP.
- Paso 2* Eliminación de arcos secundarios: buscar las rutas más cortas entre el nodo origen y cualquier otro nodo, y entre el nodo destino y cualquier otro nodo. Utilizar éstos resultados para cualquier ruta más corta a través de cualquier nodo k y cualquier arco (i, j) .
- Paso 3* Encontrar las soluciones extremas del conjunto de soluciones no inferiores (path Hamiltoniano y ruta más corta) a través del método de las restricciones. Determinar la pendiente entre estas dos soluciones no inferiores extremas.
- Paso 4* Resolver el MSPP. Determinar si la solución existen subtours en la solución encontrada.
- Paso 4.1* Si existen subtours, entonces ejecutar el método de planos cortantes para eliminar subtours maximales y minimales a través de cortes (restricciones) según vayan apareciendo. Este método realizará iteraciones hasta eliminar todos los subtours.
- Paso 4.2* Si no existen subtours en una solución, entonces la solución encontrada es óptima. En este caso ir directamente al Paso 5.

- Paso 5* Calcular nuevos intervalos de búsqueda entre la nueva solución no inferior obtenida y las anteriores. Determinar pendientes y volver al Paso 4.
- Paso 6* Finalizar cuando estén analizadas todas las pendientes entre los intervalos de soluciones no inferiores.
- Paso 7* Si se desean encontrar más soluciones no inferiores, entonces aplicar el método de las restricciones en un intervalo dado, utilizando cualquier heurística.

En el Anexo C se muestra el modelo binario y el procedimiento propuesto implementado en lenguaje AMPL.

7.6 Resultados Computacionales de los Problemas Test

Se aplicó el algoritmo propuesto a 5 Redes (problemas test). La red de 21 Nodos fue publicada en Current *et al.* (1987). Por su parte, la red de 100 nodos (pmed01) fue obtenida en la librería de Beasley (2005), mientras que las demás redes fueron generadas aleatoriamente. El MSPP fue resuelto utilizando el optimizador lineal CPLEX Versión 9.0.0 en conjunto con AMPL Versión 20021031. Todos los experimentos fueron ejecutados en un Notebook Lenovo 3000 C200 (Celeron M, 512 MB RAM).

En la resolución de la formulación binaria del MSPP, se utilizó el método de planos cortantes para eliminar los subtours. Además, se consideró la eliminación de arcos secundarios que nunca estarán presentes en la solución óptima, lo cuál redujo considerablemente los tamaños de los problemas de prueba. Para encontrar las SNI, con esta formulación, se utilizó el método NISE.

Por otra parte, en la resolución de la formulación mediante flujo multicommodity del MSPP, sólo se consideró la eliminación de arcos secundarios y la utilización del método NISE para encontrar las SNI del MSPP.

Para cada instancia, se usaron los costos originales de la red, y se generaron números aleatorios para las demandas de cada nodo. Para la redes de 21, 30 y 50 nodos se utilizaron tres pares de nodos de origen y destino elegidos aleatoriamente. Para la red de 75 y 100 nodos, se realizó el experimento para dos pares de nodos de origen y destino, debido principalmente a que el tiempo total de CPU es bastante alto.

En el caso de las redes de 30, 50, 75 y 100 nodos, se eligieron sólo 10 soluciones no inferiores del conjunto entregado por el método NISE. Esa selección se realizó de la forma propuesta en la sección 7.3.3 de este mismo capítulo.

En las tablas que se presentan a continuación, se muestra la información utilizada para cada instancia y los resultados para cada una de ellas.

En cada tabla se utilizó la siguiente notación:

- *Origen*: Nodo Origen.
- *Destino*: Nodo Destino.
- *Arcos Secundarios*: Total de arcos secundarios de la red.
- *Arcos Eliminados*: Total de arcos secundarios eliminados por el procedimiento propuesto anteriormente.
- *Arcos Secundarios Candidatos*: Total de arcos secundarios candidatos, elegidos por el procedimiento anteriormente descrito.
- *Reducción*: Porcentaje en que se reduce la cantidad de arcos secundarios.
- *Tiempo Total CPU F. Binaria (s)*: denota el tiempo total de CPU (en segundos) utilizado por la formulación binaria para encontrar el conjunto de SNI mediante el método NISE.
- *Tiempo Total CPU F. Multicommodity (s)*: denota el tiempo total de CPU (en segundos) utilizado por la formulación basada en flujo multicommodity para encontrar el conjunto de SNI mediante el método NISE.
- *Total Soluciones No Inferiores*: determina el número total de SNI generadas por el método NISE.
- *SNI*: Número de la solución no inferior obtenida por el algoritmo.
- *Z1*: Costo de construcción del path principal.
- *Z2*: Costo de accesibilidad hacia el path principal.
- *R. Bin.* : Restricciones aplicadas a una SNI en la formulación binaria.
- *It. Bin.* : Número de iteraciones hasta encontrar una SNI libre de subtours en la formulación binaria.
- *ST*: número de subtours eliminados para obtener una solución no inferior.
- *TCPU Bin*: Tiempos de CPU (en segundos) para obtener una SNI en la formulación binaria.

- *R. MC.* : Restricciones aplicadas a una SNI en la formulación mediante flujo multicommodity.
- *It. MC.*: Número de iteraciones hasta encontrar una SNI en la formulación mediante flujo multicommodity.
- *TCPU MC.*: Tiempos de CPU (en segundos) para obtener una SNI en la formulación mediante flujo multicommodity.

Cabe señalar que las SNI N°1 (*) y N°2 (*) fueron obtenidas mediante el uso de la formulación binaria. Además, estas dos primeras SNI, fueron determinadas forma independiente a las demás SNI utilizando el método de las restricciones.

7.6.1 Red de 21 Nodos

A continuación se muestran los resultados obtenidos para la red de 21 nodos y 78 arcos, variando tres orígenes y destinos escogidos al azar.

Origen	2
Destino	19
Arcos Secundarios	380
Arcos Eliminados	294
Arcos Secundarios Candidatos	86
Reducción	77.40%
Tiempo Total CPU F. Binaria (s)	1.0000
Tiempo Total CPU F. Multicommodity (s)	1.4063
Total Soluciones No Inferiores	10

Tabla 7.10 A: Resultados para la red de 21 nodos, Origen: 2; Destino: 19

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	1078	0	45	2	5	0.04688	1781	1	0.04688
2 (*)	340	84393000	186	3	0	0.03125	1781	2	0.03125
3	547	20910000	188	3	8	0.06250	1781	1	0.09375
4	376	64321000	188	4	0	0.04688	1781	2	0.06250
5	463	42099000	188	6	0	0.06250	1781	4	0.12500
6	613	15456000	195	10	7	0.23438	1781	7	0.21875
7	575	18264000	195	11	0	0.01563	1781	8	0.06250
8	883	5668000	200	15	5	0.18750	1781	11	0.35938
9	761	9726000	200	16	0	0.06250	1781	12	0.10938
10	905	5012000	200	19	0	0.25000	1781	15	0.29688

Tabla 7.10 B: Resultados para la red de 21 nodos, Origen: 2; Destino: 19

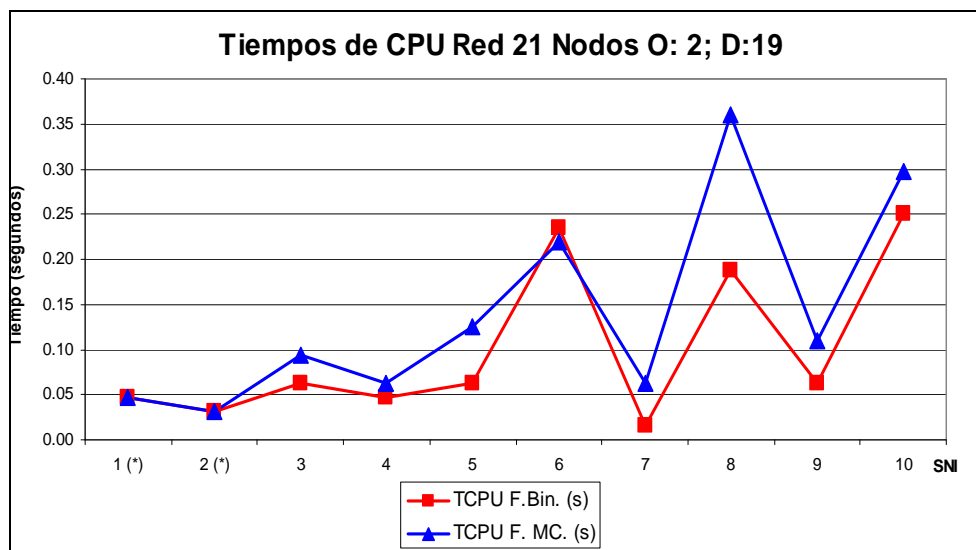


Figura 7.20: Tiempos de CPU para la red de 21 nodos, Origen: 2, Destino: 19

Origen	4
Destino	8
Arcos Secundarios	380
Arcos Eliminados	305
Arcos Secundarios Candidatos	75
Reducción	80.30%
Tiempo Total CPU F. Binaria (s)	7.3281
Tiempo Total CPU F. Multicommodity (s)	1.3125
Total Soluciones No Inferiores	11

Tabla 7.11 A: Resultados para la red de 21 nodos, Origen: 4; Destino: 8

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	1183	0	56	5	5	0.10938	1685	1	0.04688
2 (*)	77	254834000	115	6	0	0.03125	1685	2	0.03125
3	444	49408000	141	7	26	0.59375	1685	1	0.04688
4	245	143793000	177	20	36	3.37500	1685	2	0.17188
5	707	14073000	183	25	6	1.31250	1685	5	0.12500
6	627	19915000	183	26	0	0.12500	1685	6	0.06250
7	592	25010000	183	27	0	0.15625	1685	7	0.07813
8	1027	3001000	200	34	17	0.89063	1685	11	0.28125
9	827	9115000	200	35	0	0.17188	1685	12	0.06250
10	759	11923000	200	36	0	0.12500	1685	13	0.07813
11	1087	1403000	200	39	0	0.43750	1685	17	0.32813

Tabla 7.11 B: Resultados para la red de 21 nodos, Origen: 4; Destino: 8

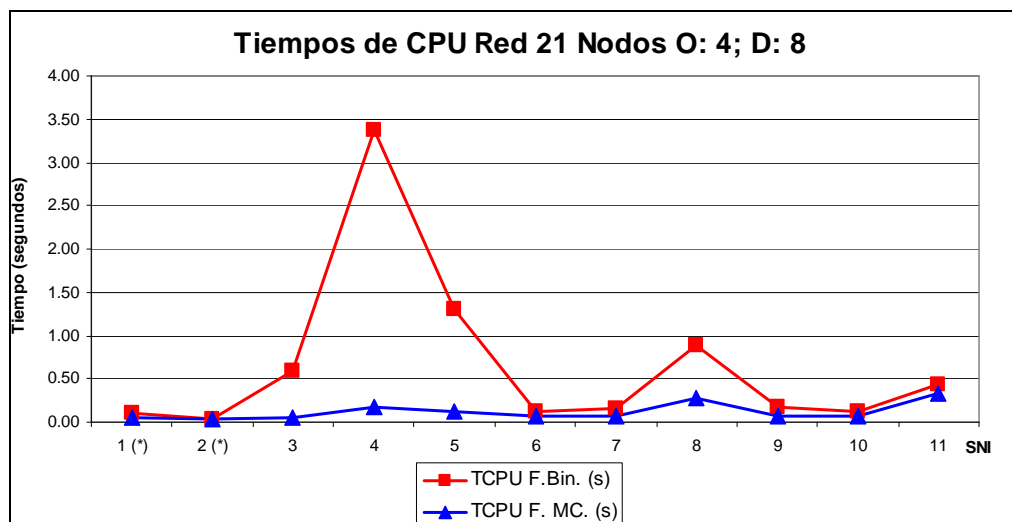


Figura 7.21: Tiempos de CPU para la red de 21 nodos, Origen: 4, Destino: 8

Origen	17
Destino	12
Arcos Secundarios	380
Arcos Eliminados	307
Arcos Secundarios Candidatos	73
Reducción	80,8%
Tiempo Total CPU F. Binaria (s)	4.1094
Tiempo Total CPU F. Multicommodity (s)	1.0625
Total Soluciones No Inferiores	9

Tabla 7.12 A: Resultados para la red de 21 nodos, Origen: 17; Destino: 12

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	1134	0	40	3	5	0.04688	40	1	0.04688
2 (*)	144	211385000	118	4	0	0.04688	118	2	0.04688
3	542	38313000	142	5	24	0.29688	1704	1	0.06250
4	535	40545000	169	15	27	2.70313	1704	2	0.07813
5	730	11676000	169	18	0	0.62500	1704	5	0.25000
6	650	17518000	169	19	0	0.07813	1704	6	0.07813
7	881	4817000	169	22	0	0.17188	1704	9	0.21875
8	770	9221000	169	23	0	0.04688	1704	10	0.09375
9	822	7071000	169	25	0	0.09375	1704	12	0.18750

Tabla 7.12 B: Resultados para la red de 21 nodos, Origen: 17; Destino: 12

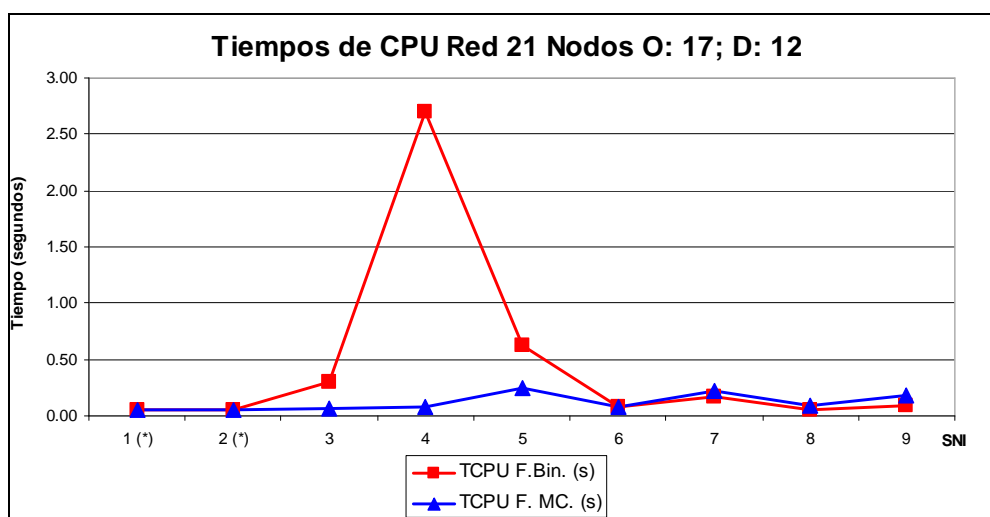


Figura 7.22: Tiempos de CPU para la red de 21 nodos, Origen: 17, Destino: 12

De la información entregada anteriormente, específicamente de las Figuras 7.20, 7.21, 7.22, se deduce que para la red de 21 nodos, los tiempos de generación de las distintas SNI de la formulación mediante flujo multicommodity son levemente mejores que los obtenidos por la formulación binaria, a pesar de que la formulación mediante flujo multicommodity tenga que considerar alrededor de 1700 restricciones.

En el gráfico siguiente (Figura 7.23) se muestra una comparación de los tiempos totales de CPU para obtener un conjunto aproximado de SNI. Se observa que, comparativamente, los resultados tanto de la formulación binaria como los resultados de la formulación basada en flujo multicommodity, no poseen diferencias significativas en cuanto a los tiempos totales de CPU.

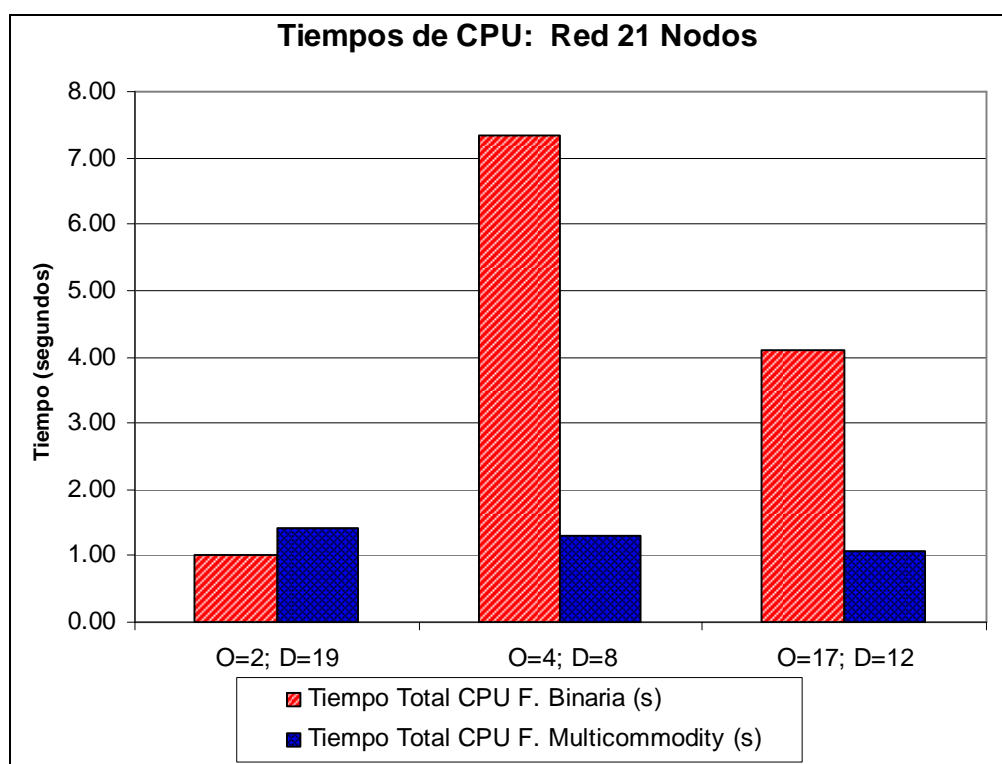


Figura 7.23: Comparación de tiempos totales de CPU para la red de 21 nodos

7.6.2 Red de 30 Nodos

En las tablas y gráficos que siguen, se muestran los resultados obtenidos para la red de 30 nodos y 142 arcos, al variar tres orígenes y destinos escogidos aleatoriamente.

Origen	8
Destino	17
Arcos Secundarios	812
Arcos Eliminados	642
Arcos Secundarios Candidatos	170
Reducción	72.70%
Tiempo Total CPU F. Binaria (s)	5632.33
Tiempo Total CPU F. Multicommodity (s)	19.64063
Total Soluciones No Inferiores	20

Tabla 7.13 A: Resultados para la red de 30 nodos, Origen: 8; Destino: 17

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	300	0	77	5	23	0.10938	4724	1	0.10938
2 (*)	74	121820272	373	6	0	0.04688	4724	2	0.04688
4	93	72376375	623	80	39	340.03125	4724	2	0.37500
5	78	96540600	623	81	0	8.64063	4724	3	0.29688
7	105	59813631	729	138	106	3770.82813	4724	8	1.37500
8	149	22977535	791	144	62	433.01563	4724	10	2.09375
13	162	14865028	797	154	0	29.71875	4724	20	0.79688
16	244	1594864	802	164	5	1.46875	4724	27	1.29688
17	217	3070447	802	165	0	0.10938	4724	28	0.70313
18	192	5255872	802	166	0	0.15625	4724	29	0.39063

Tabla 7.13 B: Resultados para la red de 30 nodos, Origen: 8; Destino: 17

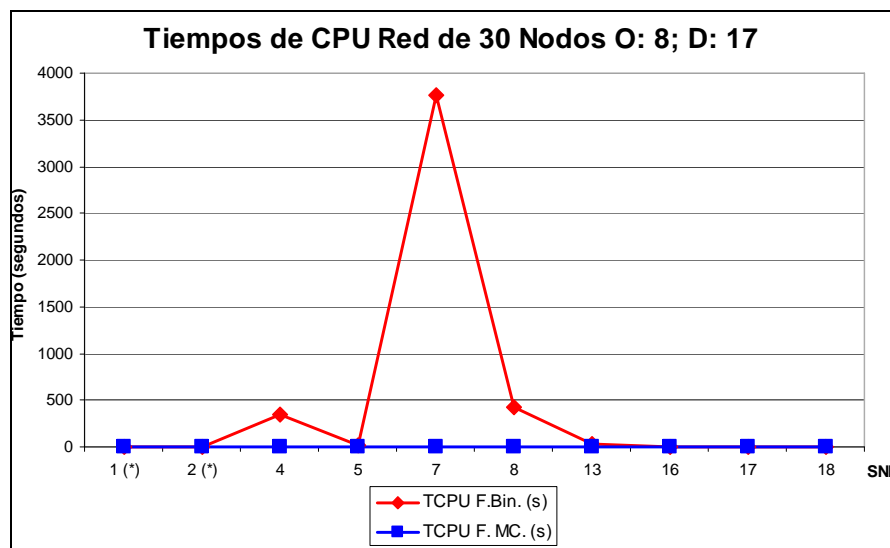


Figura 7.24: Tiempos de CPU para la red de 30 nodos, Origen: 8, Destino: 17

Origen	15
Destino	28
Arcos Secundarios	812
Arcos Eliminados	606
Arcos Secundarios Candidatos	206
Reducción	74.60%
Tiempo Total CPU F. Binaria (s)	31.875
Tiempo Total CPU F. Multicommodity (s)	62.531
Total Soluciones No Inferiores	21

Tabla 7.14A: Resultados para la red de 30 nodos, Origen: 15; Destino: 28

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	306	0	58	5	25	0.12500	58	1	0.12500
2 (*)	56	131177733	295	6	0	0.06250	295	2	0.06250
3	108	43373134	338	8	43	4.04688	4703	1	0.29688
4	90	61338321	353	14	15	3.53125	4703	2	0.28125
5	99	49888265	356	17	3	2.84375	4703	4	0.57813
7	105	45443002	356	21	0	3.65625	4703	8	0.82813
9	154	20122428	366	27	10	0.68750	4703	12	0.35938
11	147	23465235	366	29	0	1.07813	4703	14	0.32813
12	165	15527423	366	33	0	3.57813	4703	18	1.31250
16	204	7176172	381	46	15	0.37500	4703	26	1.64063

Tabla 7.14 B: Resultados para la red de 30 nodos, Origen: 15; Destino: 28

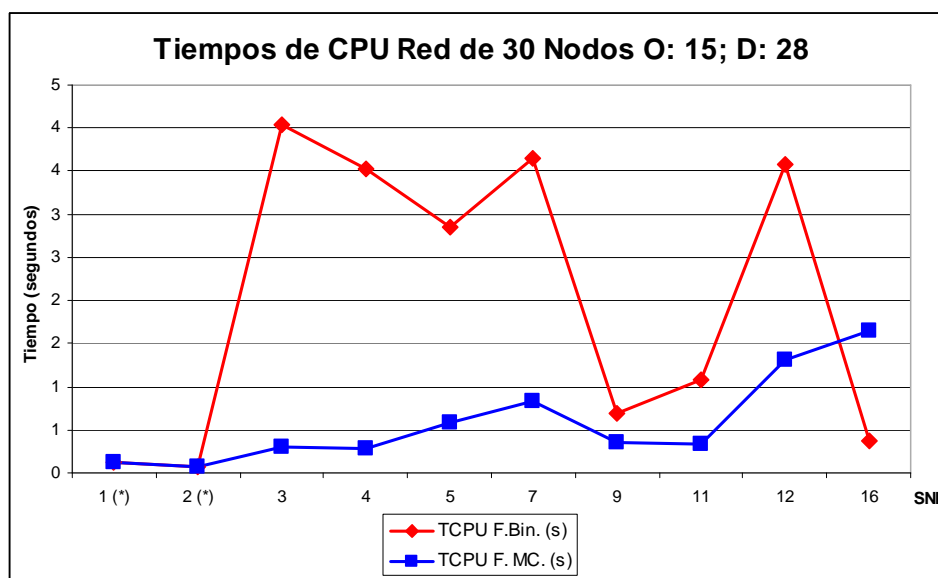


Figura 7.25: Tiempos de CPU para la red de 30 nodos, Origen: 15, Destino: 28

Origen	18
Destino	9
Arcos Secundarios	812
Arcos Eliminados	551
Arcos Secundarios Candidatos	261
Reducción	67.90%
Tiempo Total CPU F. Binaria (s)	1114.266
Tiempo Total CPU F. Multicommodity (s)	19.406
Total Soluciones No Inferiores	23

Tabla 7.15 A: Resultados para la red de 30 nodos, Origen: 18; Destino: 9

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	294	0	84	7	26	0.15625	4641	1	0.15625
2 (*)	60	131885661	322	8	0	0.09375	4641	2	0.09375
3	141	33258407	462	35	140	154.92188	4641	1	0.23438
4	117	48955574	588	78	126	625.14063	4641	2	0.34375
5	77	98541461	588	79	0	15.53125	4641	3	0.26563
7	103	66132676	591	84	3	74.42188	4641	7	0.95313
8	110	57482967	591	86	0	60.95313	4641	9	0.68750
14	170	18188318	591	97	0	5.54688	4641	20	0.29688
16	198	8019081	591	103	0	11.71875	4641	26	1.20313
20	230	2588854	597	113	6	0.140625	4641	34	0.484375

Tabla 7.15 B: Resultados para la red de 30 nodos, Origen: 18; Destino: 9

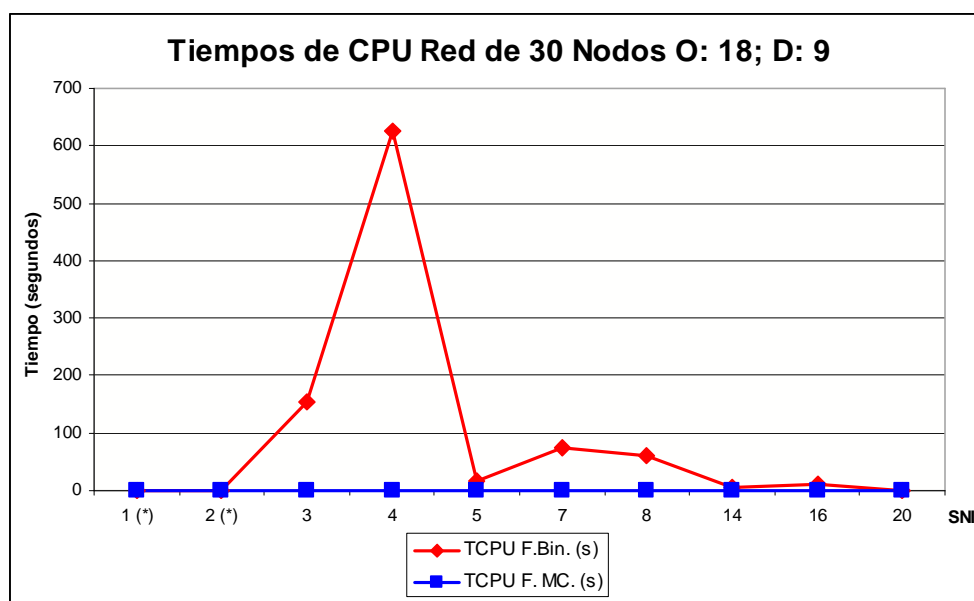


Figura 7.26: Tiempos de CPU para la red de 30 nodos, Origen: 18, Destino: 9

En las Figuras 7.24, 7.25 y 7.26, se observa que en la red de 21 nodos, los tiempos de generación de las distintas SNI de la formulación mediante flujo multicommodity son notablemente mejores que los obtenidos por la formulación binaria.

En la Figura 7.27 se muestra gráficamente la gran diferencia de tiempos totales de CPU entre una y otra formulación, sobretodo en el primer experimento (O=8, D=17). Esto muestra que la efectividad de los tiempos de CPU de la formulación basada en flujo multicommodity en redes pequeñas puede ser bastante alta, superando con creces los tiempos de la formulación binaria.

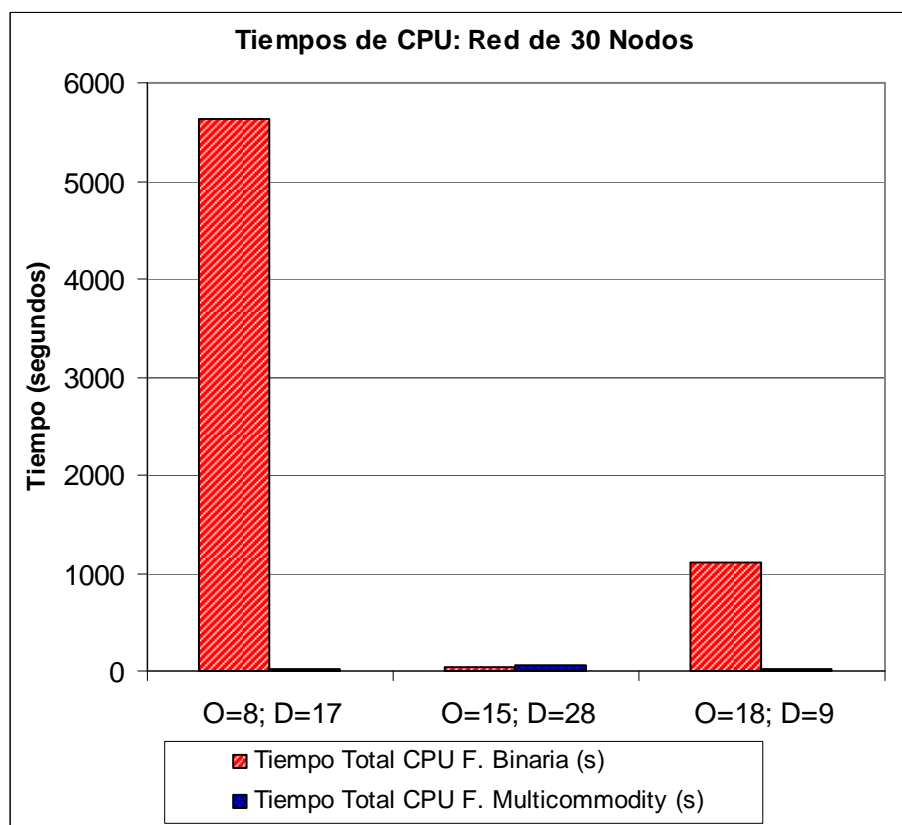


Figura 7.27: Comparación de tiempos totales de CPU para la red de 30 nodos

7.6.3 Red de 50 Nodos

A continuación se exponen los resultados obtenidos para la red de 50 nodos y 248 arcos, variando tres orígenes y destinos escogidos al azar.

Origen	3
Destino	46
Arcos Secundarios	2352
Arcos Eliminados	1710
Arcos Secundarios Candidatos	642
Reducción	72.70%
Tiempo Total CPU F. Binaria (s)	489.031
Tiempo Total CPU F. Multicommodity (s)	798.297
Total Soluciones No Inferiores	35

Tabla 7.16A: Resultados para la red de 50 nodos, Origen: 3; Destino: 46

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	14155	0	126	6	28	0.14063	13908	1	0.14063
2 (*)	2120	25437190	739	7	0	0.34375	13908	2	0.34375
3	6158	5405910	790	10	51	23.57813	13908	1	15.67188
4	3666	12400830	843	28	53	191.31250	13908	2	2.00000
8	2606	18981980	843	34	0	8.15625	13908	8	2.10938
9	3434	13630240	846	38	3	48.37500	13908	11	5.93750
11	4872	8346620	854	45	8	36.39063	13908	16	8.23438
12	4505	9202980	854	46	0	6.79688	13908	17	3.50000
14	9180	1523100	871	55	17	21.04688	13908	23	31.28125
18	7299	3285990	883	65	0	2.37500	13908	29	29.29688

Tabla 7.16B: Resultados para la red de 50 nodos, Origen: 3; Destino: 46

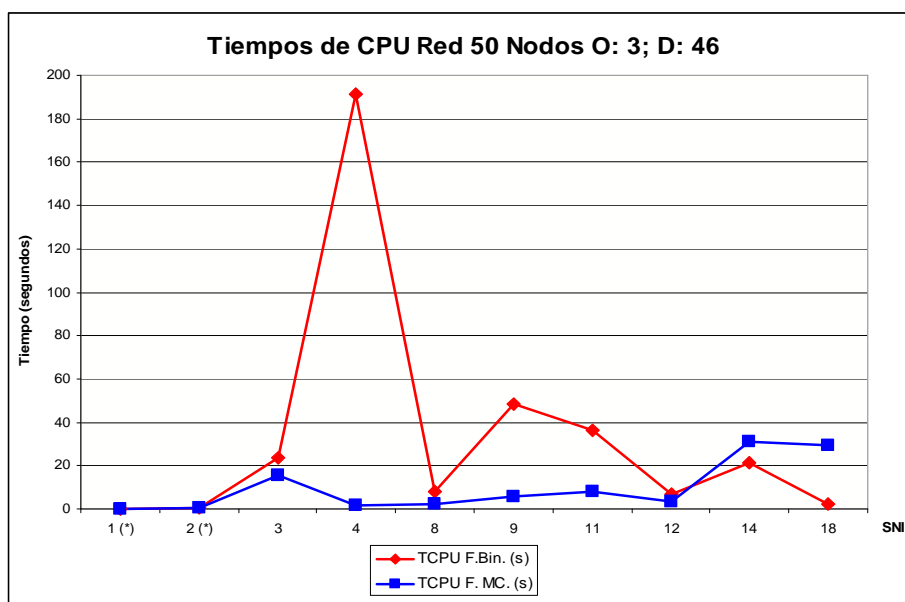


Figura 7.28: Tiempos de CPU para la red de 50 nodos, Origen: 3, Destino: 46

Origen	33
Destino	24
Arcos Secundarios	2352
Arcos Eliminados	1761
Arcos Secundarios Candidatos	591
Reducción	74.90%
Tiempo Total CPU F. Binaria (s)	1859.344
Tiempo Total CPU F. Multicommodity (s)	3662.250
Total Soluciones No Inferiores	36

Tabla 7.17A: Resultados para la red de 50 nodos, Origen: 33; Destino: 24

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	14054	0	129	6	31	0.18750	14075	1	0.18750
2 (*)	1177	34944660	710	7	0	0.26563	14075	2	0.26563
3	2051	20839970	807	24	0	92.81250	14075	3	1.10938
7	1598	27827530	810	28	3	6.85938	14075	6	2.35938
9	3377	13178230	863	50	3	172.82813	14075	11	1.84375
11	4153	10124610	866	56	3	173.35938	14075	16	16.87500
15	5982	5267320	922	73	0	5.93750	14075	23	6.71875
17	6506	4339180	922	77	0	32.29688	14075	27	17.81250
19	7301	3295460	933	84	8	65.51563	14075	31	5.09375
24	9502	1321960	955	100	0	9.98438	14075	41	71.76563

Tabla 7.17B: Resultados para la red de 50 nodos, Origen: 33; Destino: 24

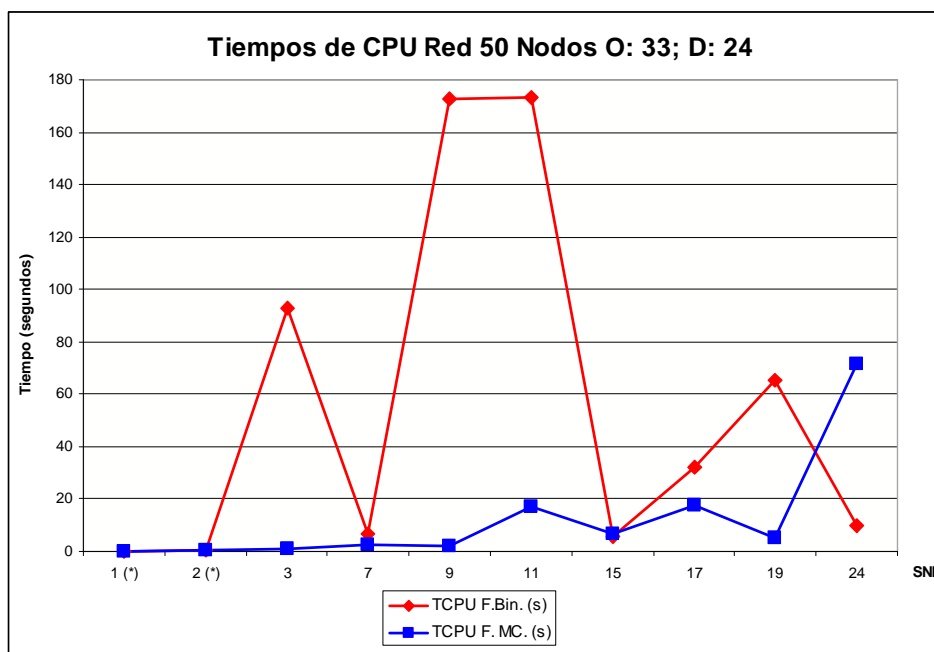


Figura 7.29: Tiempos de CPU para la red de 50 nodos, Origen: 33, Destino: 24

Origen	15
Destino	38
Arcos Secundarios	2532
Arcos Eliminados	1981
Arcos Secundarios Candidatos	551
Reducción	80.30%
Tiempo Total CPU F. Binaria (s)	265.125
Tiempo Total CPU F. Multicommodity (s)	1842.250
Total Soluciones No Inferiores	35

Tabla 7.18A: Resultados para la red de 50 nodos, Origen: 15; Destino: 38

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	14221	0	98	4	27	0.09375	13618	1	0.0938
2 (*)	1108	29831840	513	5	0	0.21875	13618	2	0.2188
3	5195	5797530	551	9	38	12.67188	13618	1	1.1094
4	2516	14201500	551	10	0	2.21875	13618	2	0.8438
5	1739	19854660	554	12	3	3.31250	13618	3	0.7969
6	3662	9687900	562	17	8	9.79688	13618	6	2.7656
7	3100	11614920	562	18	0	2.01563	13618	7	1.1094
13	6129	4322720	635	49	0	4.42188	13618	19	1.5313
17	8002	2389670	654	63	19	37.29688	13618	28	35.0625
20	10304	885140	682	75	14	7.578125	13618	34	60.8281

Tabla 7.18B: Resultados para la red de 50 nodos, Origen: 15; Destino: 38

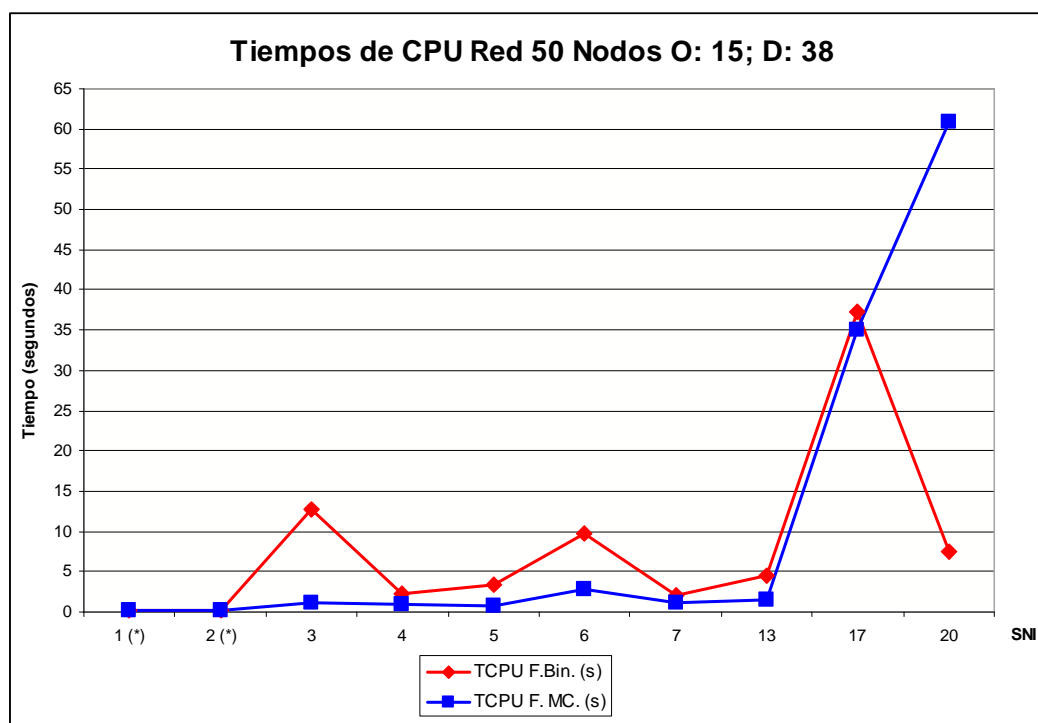


Figura 7.30: Tiempos de CPU para la red de 50 nodos, Origen: 33, Destino: 24

De los gráficos de las Figuras 7.28, 7.29 y 7.30, desprendidos de las Tablas 7.16A, 7.16B, 7.17A, 7.17B, 7.18A y 7.18B respectivamente, se puede observar que aún existe cierta superioridad en cuanto los tiempos de CPU de la formulación mediante flujo multicommodity por sobre la formulación binaria, para las mismas SNI seleccionadas. Sin embargo, las diferencias ya no son tan notorias en comparación a los resultados obtenidos de la red de 30 nodos. Es más, en algunos casos, el tiempo de CPU en obtener una SNI mediante la formulación binaria es menor que mediante la formulación basada en flujo multicommodity.

En las Tablas 7.16B, 7.17B y 7.18B se observa que la diferencia de restricciones entre una y otra formulación es alta. Sin embargo, los tiempos de la formulación mediante flujo multicommodity no se ven afectados por este factor.

En la Figura 7.31 se muestran gráficamente los tiempos totales de CPU para obtener un conjunto de SNI, ante distintas instancias. Se muestra que la formulación basada en flujo multicommodity todavía sigue siendo ser útil para esta red de 50 nodos, pero sus resultados comienzan a equipararse a los resultados de la formulación binaria.

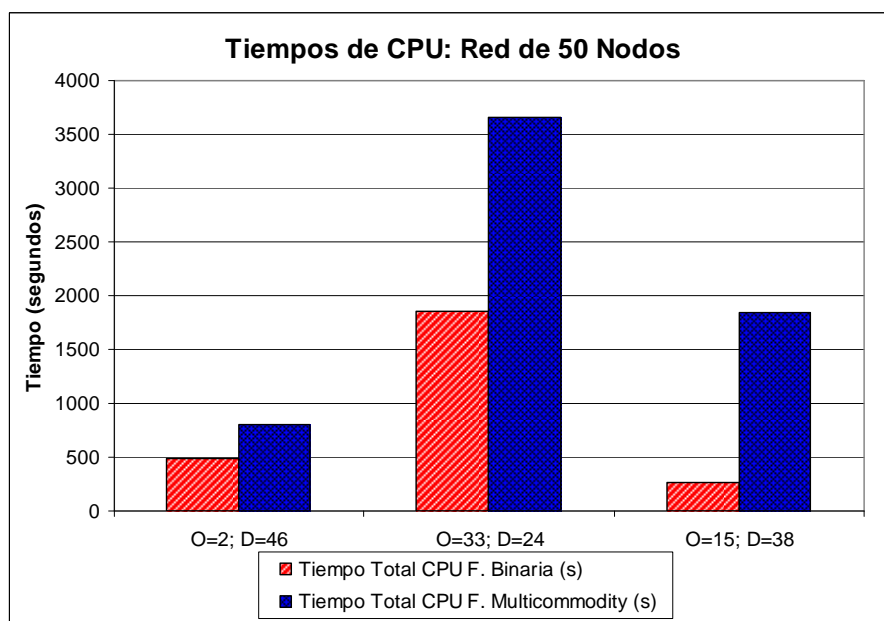


Figura 7.31: Comparación de tiempos totales de CPU para la red de 50 nodos

7.6.4 Red de 75 Nodos

En las tablas y gráficos que siguen, se muestran los resultados entregados por el algoritmo propuesto para la red de 75 nodos y 350 arcos, variando dos orígenes y destinos escogidos al azar.

Origen	4
Destino	75
Arcos Secundarios	5402
Arcos Eliminados	3682
Arcos Secundarios Candidatos	1720
Reducción	68.20%
Tiempo Total CPU F. Binaria (s)	71234.55
Tiempo Total CPU F. Multicommodity (s)	80453.485
Total Soluciones No Inferiores	38

Tabla 7.19A: Resultados para la red de 75 nodos, Origen: 4; Destino: 75

SNI	Z1	Z2	R.Bin.	lt.Bin.	ST	TCPU Bin.	R.MC.	lt.MC.	TCPU MC.
1 (*)	1206	0	205	11	57	1.29688	31457	1	1.29688
2 (*)	123	4804370	1795	12	0	0.53125	31457	2	0.53125
3	364	1079930	1905	17	110	4012.71875	31457	1	15.14063
4	176	2380530	1920	23	15	104.39063	31457	2	7.01563
7	216	1979900	1929	31	6	255.43750	31457	7	10.70313
9	244	1750120	1964	44	0	1391.76563	31457	11	13.71875
10	320	1322890	1964	47	0	2639.07813	31457	14	55.78125
14	373	1044070	2162	97	0	577.59375	31457	20	12.84375
16	458	792830	2162	102	0	2588.90625	31457	25	64.73438
17	615	471340	2180	111	18	7050.04688	31457	28	482.10938

Tabla 7.19B: Resultados para la red de 75 nodos, Origen: 4; Destino: 75

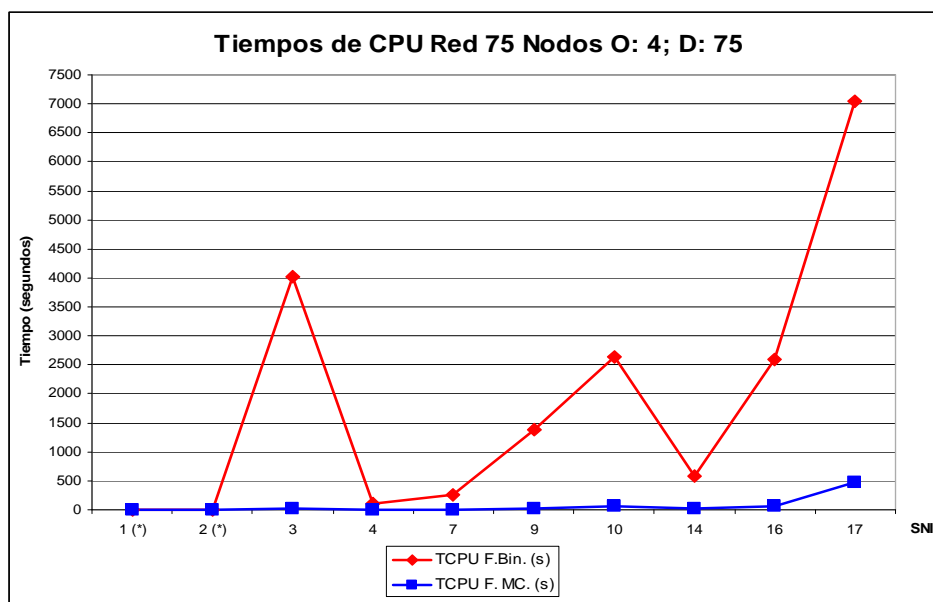


Figura 7.32: Tiempos de CPU para la red de 75 nodos, Origen: 4, Destino: 75

Origen	21
Destino	52
Arcos Secundarios	5402
Arcos Eliminados	3870
Arcos Secundarios Candidatos	1532
Reducción	71.60%
Tiempo Total CPU F. Binaria (s)	111551.48
Tiempo Total CPU F. Multicommodity (s)	98576.670
Total Soluciones No Inferiores	40

Tabla 7.20A: Resultados para la red de 75 nodos, Origen: 21; Destino: 52

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	1170	0	192	5	44	0.25000	31181	1	0.25000
2 (*)	133	5479700	1607	6	0	0.56250	31181	2	0.56250
3	381	1230680	1748	22	141	13311.62500	31181	1	171.62500
7	175	3021690	1878	68	3	514.85938	31181	7	21.54688
8	215	2621060	1892	74	14	4840.67188	31181	8	20.89063
9	290	1939160	1898	79	6	11920.79688	31181	12	197.87500
10	279	2029930	1898	80	0	1619.28125	31181	13	28.43750
15	459	882990	1952	100	0	1107.48438	31181	23	816.84375
16	413	1066230	1952	101	0	707.23438	31181	24	837.39063
19	503	720990	1952	108	0	3555.28125	31181	31	1373.51563

Tabla 7.20B: Resultados para la red de 75 nodos, Origen: 21; Destino: 52

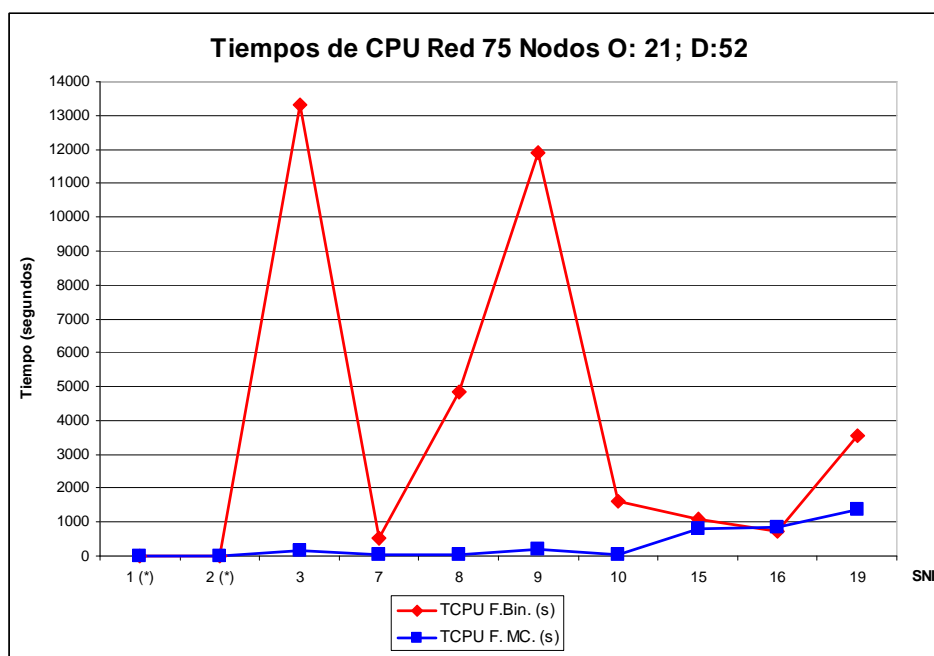


Figura 7.33: Tiempos de CPU para la red de 75 nodos, Origen: 21, Destino: 52

Para la red de 75 nodos y 350 arcos, se variaron sólo dos orígenes y destinos, debido al extenso tiempo de CPU que toma el algoritmo para encontrar soluciones no inferiores (alrededor de 30 horas en algunos casos). De los gráficos de las Figuras 7.32 y 7.33 se observa que los tiempos de obtención de las SNI seleccionadas mediante la formulación basada en flujo multicommodity aún son menores en comparación a los tiempos de la formulación binaria.

En las Tablas 7.19B y 7.20B, se visualiza que el número de restricciones de la formulación basada en flujo multicommodity es muy considerable (alrededor de 31000 restricciones) en comparación al número de restricciones utilizada por la formulación binaria (alrededor de 2000 restricciones). Este factor explica el aumento de los tiempos de CPU de la formulación basada en flujo multicommodity.

En la Figura 7.34 se muestran gráficamente los tiempos totales de CPU para obtener un conjunto aproximado de SNI, ante distintas instancias. La formulación mediante flujo multicommodity todavía sigue siendo útil para esta red de 75 nodos, aunque los resultados totales ya no son muy superiores a los de la formulación binaria.

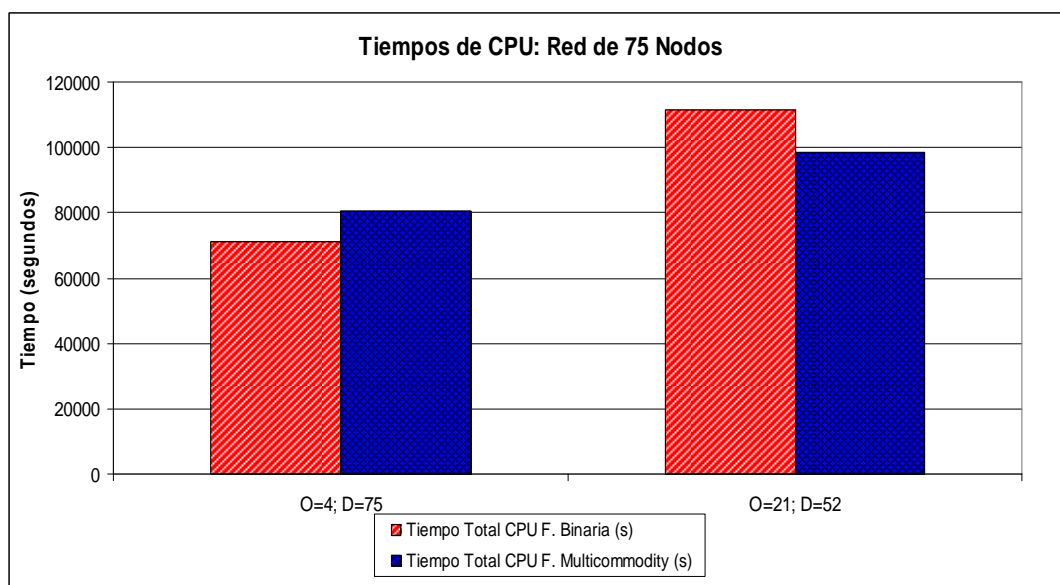


Figura 7.34: Comparación de tiempos totales de CPU para la red de 75 nodos

7.6.5 Red de 100 Nodos

A continuación se muestran los resultados obtenidos para la red de 100 nodos y 396 arcos, variando dos orígenes y destinos escogidos aleatoriamente.

Origen	40
Destino	97
Arcos Secundarios	9702
Arcos Eliminados	3990
Arcos Secundarios Candidatos	5712
Reducción	41.13%
Tiempo Total CPU F. Binaria (s)	59737.953
Tiempo Total CPU F. Multicommodity (s)	143996.843 (*)
Total Soluciones No Inferiores	36

(*) Se detuvo el proceso iterativo de la formulación flujo multicommodity en 143996.843 segundos

Tabla 7.21A: Resultados para la red de 100 nodos, Origen: 40; Destino: 97

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	4503	0	248	4	50	0.12500	52923	1	0.12500
2 (*)	299	379972038	5799	5	0	1.04688	52923	2	1.04688
3	1146	109755214	5910	8	111	2566.92188	52923	1	328.32813
7	745	161524937	5948	23	0	1463.06250	52923	8	2166.67188
8	536	221195970	5954	26	6	3983.65625	52923	9	4230.34375
9	955	133092588	5968	33	14	18303.53125	52923	12	63928.48438
10	2457	41335874	6001	39	33	3731.07813	52923	15	3699.28125
12	1380	94069909	6021	45	10	4437.46875	52923	17	5652.35938
16	1625	79011808	6024	54	0	2807.75000	-	-	-
20	1964	61972030	6033	64	0	249.85938	-	-	-

Tabla 7.21B: Resultados para la red de 100 nodos, Origen: 40; Destino: 97

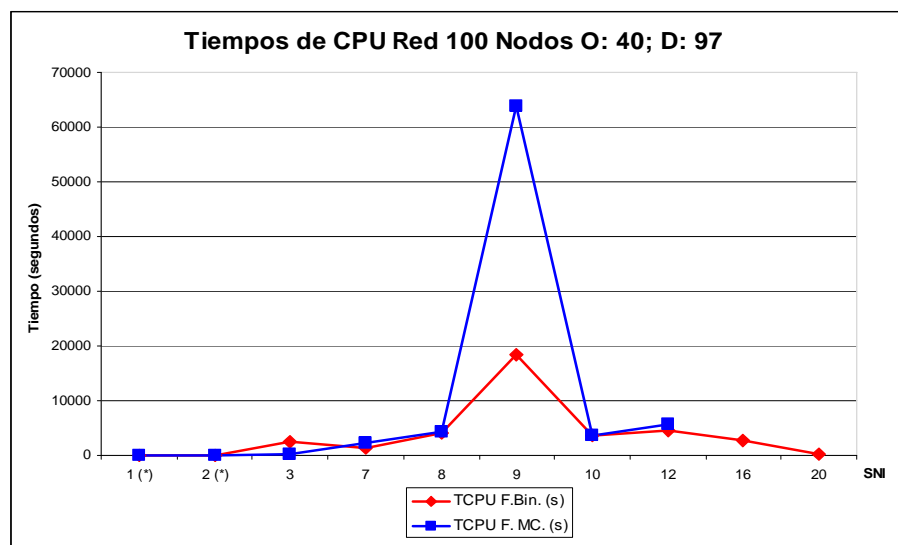


Figura 7.35: Tiempos de CPU para la red de 100 nodos, Origen: 40, Destino: 97

Origen	17
Destino	66
Arcos Secundarios	9702
Arcos Eliminados	4516
Arcos Secundarios Candidatos	5186
Reducción	46.50%
Tiempo Total CPU F. Binaria (s)	57827.453
Tiempo Total CPU F. Multicommodity (s)	160441.859 (*)
Total Soluciones No Inferiores	31

(*) Se detuvo el proceso iterativo de la formulación flujo multicommodity en 173548.432 segundos

Tabla 7.22A: Resultados para la red de 100 nodos, Origen: 17; Destino: 66

SNI	Z1	Z2	R.Bin.	It.Bin.	ST	TCPU Bin.	R.MC.	It.MC.	TCPU MC.
1 (*)	4510	0	242	8	45	1.00000	52252	1	1.00000
2 (*)	248	382754766	5286	9	0	0.87500	52252	2	0.87500
3	1103	111901924	5392	7	106	2575.00000	52252	1	526.26500
4	510	228392126	5440	18	48	16498.21875	52252	2	376.37500
6	389	272353816	5440	21	0	590.50000	52252	5	374.10938
7	716	167071263	5443	25	3	2915.53125	52252	8	981.59375
9	945	130375765	5464	36	0	2753.84375	52252	12	958.609375
13	2000	59230197	5509	49	9	205.46875	52252	20	306.37500
20	2472	41653513	5520	66	3	2678.48438	52252	34	26997.76563
22	3260	17760063	5550	76	0	7.89063	52252	38	293.57813

Tabla 7.22B: Resultados para la red de 100 nodos, Origen: 17; Destino: 66

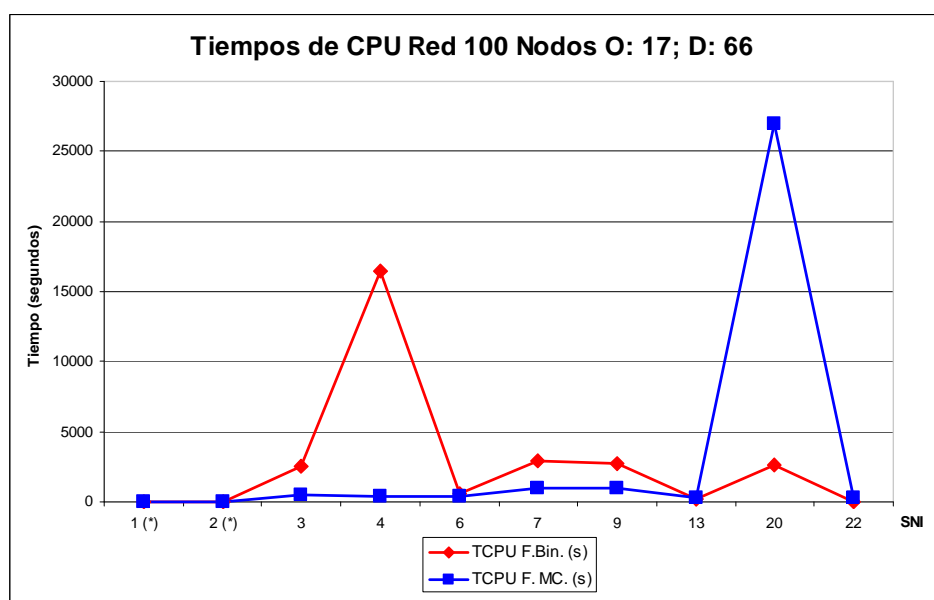


Figura 7.36: Tiempos de CPU para la red de 100 nodos, Origen: 17, Destino: 66

Para la red de 100 nodos y 390 arcos, se variaron sólo dos orígenes y destinos, al igual que con la red de 75 nodos, debido al extenso tiempo de CPU que toma el algoritmo para encontrar soluciones no inferiores (alrededor de 40 horas en algunos casos). De los gráficos de las Figuras 7.35 y 7.36 se observa que los tiempos de la formulación binaria son menores en comparación a los tiempos de la formulación mediante flujo multicommodity para obtener las SNI seleccionadas.

En la Tablas 7.21B y 7.22B se observa claramente que el número de restricciones de la formulación basada en flujo multicommodity aumentó en forma considerable, lo que implicó un aumento notable en los tiempos de obtención de las SNI. Por su parte, el número de restricciones de la formulación binaria es 10 veces menor al número de restricciones de la formulación mediante flujo multicommodity. El tiempo utilizado por la formulación binaria se explica fundamentalmente debido al proceso iterativo de búsqueda de subtours.

Los tiempos de obtención del conjunto aproximado de SNI mediante el método NISE, son bastante altos. Por una parte, la resolución del problema binario logra encontrar todo el conjunto en aproximadamente 60000 segundos. Por otra parte, se decidió detener el proceso iterativo de la formulación basada en flujo multicommodity en aproximadamente 160000 segundos. Es más, esta formulación sólo encuentra algunas SNI en estos 160000 segundos (no el conjunto completo), por lo que se muestra que la formulación basada en flujo multicommodity puede ser bastante útil para redes hasta de 75 nodos. De ahí en adelante, es recomendable utilizar la formulación binaria, y por ende, el procedimiento propuesto en el presente trabajo.

Capítulo 8: CONCLUSIONES Y FUTURAS INVESTIGACIONES

8.1 Conclusiones

Del desarrollo del tema, se obtuvieron las siguientes conclusiones:

- El MSPP es aplicable a un problema de diseño de redes en que se tenga un path conectando un nodo origen con un nodo destino (ambos predeterminados), y además, que la accesibilidad al path principal sea un objetivo a considerar.
- El procedimiento de selección de arcos secundarios candidatos, permite reducir considerablemente el tamaño del problema, lo que favorece en forma notable al proceso de búsqueda de las diferentes soluciones no inferiores. En algunos casos, el problema se llegó a reducir hasta en un 80% con esta técnica.
- El proceso de eliminación de subtours, es computacionalmente intenso, por lo que las estrategias de búsqueda y eliminación de subtours en cualquier algoritmo de resolución del MSPP deben ser muy bien pensadas y analizadas para ser ejecutadas.
- El método de planos cortantes permite eliminar la infactibilidad de las soluciones no inferiores y por lo tanto, encontrar las soluciones no inferiores del MSPP en forma óptima.
- En relación a los resultados obtenidos mediante la formulación basada en flujo multicommodity, se puede decir que esta formulación es muy recomendable para resolver el MSPP para redes pequeñas (menos de 75 nodos). Esta formulación está libre de subtours, sin embargo, posee un número considerable de restricciones, lo que provoca, en redes grandes,

considerables tiempos de CPU para las distintas iteraciones en la generación de soluciones no inferiores.

- Los resultados obtenidos mediante la formulación binaria del MSPP, permiten concluir que el algoritmo de resolución para resolver el MSPP presenta mejores resultados que los resultados obtenidos al formular el problema mediante flujo multicommodity, a partir de una red de 100 nodos.
- La generación del conjunto completo de soluciones no inferiores es computacionalmente intensa, ya sea mediante la formulación basada en flujo multicommodity o mediante la formulación binaria.
- Los analistas y tomadores de decisiones a menudo quedan satisfechos con una buena aproximación del conjunto de soluciones no inferiores. Por consiguiente, no es necesario encontrar el conjunto completo de soluciones; en algunos casos, hay sectores dentro de la curva de *trade-off* que difícilmente serán tomados en cuenta por el tomador de decisiones.
- Pueden existir intervalos en donde el método NISE no encuentre soluciones no inferiores, y sin embargo, existan soluciones no inferiores. Si el tomador de decisiones lo requiere, el analista puede realizar una búsqueda exhaustiva dentro de un intervalo dado, a través del método de las restricciones o utilizando cualquier heurística, con el fin de encontrar una mayor cantidad de soluciones no inferiores en el intervalo requerido. Todo esto ayudará a que el tomador de decisiones elija la mejor alternativa según su presupuesto y lo que esté dispuesto a gastar en la construcción del path principal, considerando el objetivo de accesibilidad del path principal. Consecuentemente, todas las alternativas generadas y expuestas en la curva de *trade-off* entre costo y accesibilidad pueden ser utilizadas por el tomador de decisiones.

8.2 Futuras Investigaciones

La complejidad computacional al utilizar un problema formulado con programación lineal entera binaria es bastante alta, sin embargo, el proceso de eliminación de subtours es relativamente sencillo desde el punto de vista de la programación e implementación del algoritmo en un software de resolución de problemas (como por ejemplo el AMPL, que fue el lenguaje utilizado en el presente trabajo).

El Median Shortest Path Problem también puede ser resuelto utilizando programación lineal entera mixta. Sin embargo, la eliminación de subtours en un problema fraccionario es mucho más compleja que en el caso binario debido a la aparición de soluciones infactibles con valores fraccionarios. No obstante, el beneficio de resolver el problema fraccionario recae en una menor complejidad computacional para resolver el MSPP, y por lo tanto, la obtención del conjunto de soluciones no inferiores mucho más rápida. Por consiguiente, se podría resolver el MSPP para redes de mayor magnitud, con tiempos de CPU considerablemente menores.

BIBLIOGRAFÍA

AMPL Tutorial [en línea] <http://www.alkires.com/OR1/amplappendix.pdf> [consulta: 10 Febrero 2008].

AVELLA, P., BOCCIA, B., SFORZA, A. 2005. A Branch-and-Cut Algorithm for the Median-Path Problem, Revista Computational Optimization and Applications, Vol. 32, pp 215-230.

BEASLEY, J.-E. 2005. OR-Library p-median – uncapacitated. [en línea] <http://people.brunel.ac.uk/~mastijb/jeb/orlib/pmedinfo.html> [consulta: 18 Marzo 2008].

COHON, J. 1978. Multiobjective Programming and Language. New York. Academic Press Inc, 333p.

CURRENT, J., REVELLE, Ch., COHON J. 1987. The Median Shortest Path Problem: A Multiobjective Approach to Analyze Cost vs. Accesibility in the Design of Transportation Networks, Revista Transportation Science, Vol. 21, No. 3: pp.188-197.

CURRENT, J, SCHILLING. 1994. The Median Tour and Maximal Covering Tour Problems. European Journal of Operational Research 73, pp. 114-126.

FERNÁNDEZ, M. 2000. Diseño e Implementación de una Interfaz para Resolver Problemas de Programación Lineal. Memoria de Ingeniero Civil Industrial. Concepción, Universidad del Bío-Bío. Facultad de Ingeniería. 109p.

FOURER, R., GAY D., KERNIGHAN W. 1990. AMPL: A Mathematical Programming Language [en línea] <http://www.ampl.com/REFS/amplmod.pdf> [consulta 18 Febrero 2008].

HALDER, D.K. , MAJUMDER A. 1981. A Method for Selecting Optimum Number of Stations for a Rapid Transit Line: An Application in Calcuta Tube Rail. Scientific Management of Transport Systems, N.K. Jaiswal (ed.), North Holland, Amsterdam, pp. 97-108.

HANSEN, P. 1980. Bicriterion Path Problems, in Lecture Notes in Economics and Mathematical Systems 177, M. Beckmann and H.P. Kunzi (eds.), Springer-Verlag, Berlín.

HOLMES, D. 1995. AMPL (A Mathematical Programming Language) at the University of Michigan, Documentation (Version 2).

JOSA, R., 2007. La Investigación Operativa [en línea] <http://www.eio.uva.es/~ricardo/io/introio.pdf> [consulta: 19 Febrero 2008].

KAMINSKY, P., RAJAN, D. 2005. Introduction to AMPL: A Tutorial. [en línea] <http://www.ieor.berkeley.edu/~ieor162/ampldoc.pdf> [consulta: 15 Febrero 2008].

LABBÉ, M., LAPORTE, G.,; RODRÍGUEZ, I. 1998. In Fleet Management and Logistics, T. Crainic, G. Laporte (eds), Kluwer, Boston, 187-204.

LARI, I., RICCA, F., SCOZZARI, A. 2007. Comparing different metaheuristic approaches for the median path problem with bounded length. European Journal of Operational Research.

LUQUE, P. 2000. Lenguaje AMPL, Versión 2a [en línea] <http://www.informatica.us.es/~calvo/ampl2a.pdf> [consulta: 20 Septiembre 2007].

MARTINS, E. 1984. On a Multi-Criteria Shortest Path Problem. European Journal of Operational Research” Vol.16, pp. 236-245.

PRASAD, K., y PARK, D. 2005. Solving the Median Shortest Path Problem in the Planning of Urban Transportation Networks Using a Vector Labeling Algorithm. Revista Transportation Planning and Technology, Vol. 28, No. 2, pp 113-133

SALLÁN, J., FONOLLOSA, J., SUÑÉ, A., FERNÁNDEZ, F. 2002. Métodos Cuantitativos en Organización Industrial I. Editions UPC, España.

TAHA, H. 1998. Investigación de Operaciones Una Introducción. Editorial Prentice-Hall, México.

WINSTON, W. 1994. Investigación de Operaciones, Grupo Editorial Iberoamérica, México.

ANEXO A: Algoritmo de Generación de Números Aleatorios

En esta sección se da a conocer el algoritmo que se utilizó para obtener números aleatorios para los problemas test que no poseían demandas $D[i]$. Para generar números aleatorios, se usó la distribución Uniforme, y una semilla (número primo). El algoritmo se implementó en el lenguaje AMPL y es el siguiente:

```
##Modelo generación números aleatorios##
```

```
set Arcos;
```

```
set Nodos;
```

```
param D {Nodos} >= 0;
```

```
##Algoritmo para generar números aleatorios##
```

```
model genera.mod;
```

```
data genera.dat;
```

```
option randseed 19937;
```

```
#semilla = 19937
```

```
display Nodos;
```

```
for {i in Nodos}{
```

```
    let D[i]:=round(Uniform(1000,100000));    #distribución uniforme para generar  
                                                #números aleatorios redondeados  
                                                #entre 1000 y 100000.
```

```
};
```

```
display D;
```

ANEXO B. Algoritmo de Generación de la Matriz de Rutas Más Cortas para el MSPP

A continuación se muestra el algoritmo utilizado para obtener la matriz de rutas más cortas para el MSPP, implementado en el lenguaje AMPL:

```
##Modelo matriz de rutas más cortas##
```

```
#Parámetros y Conjuntos#
```

```
set Nodos;  
set Arcos within {Nodos,Nodos} default {};  
param c {Arcos};  
param O;  
param D;
```

```
#Variables de Decisión#
```

```
var x {Arcos} binary;
```

```
#Función Objetivo y Restricciones#
```

```
minimize Costo: sum{(i,j) in Arcos} c[i,j]*x[i,j];  
s.t. Inicio: sum{(O,j) in Arcos} x[O,j]=1;  
s.t. Final: sum{i in Nodos: (i,D) in Arcos} x[i,D]=1;  
s.t. Balance {j in Nodos diff {O,D}}:  
    sum{(i,j) in Arcos: i!=D} x[i,j] = sum{(j,h) in Arcos: h!=O} x[j,h];
```

```

##Algoritmo para encontrar la matriz de rutas más cortas##
model matrizrmcmp.mod;
data matrizrmcmp.dat;
problem matrizrmc: x, Costo, Inicio, Final, Balance;
param Matriz {Nodos, Nodos};
for {a in Nodos, b in Nodos diff{a}: a < b} {
    let O:= a;
    let D:= b;
    solve matrizrmc;
    printf "Origen = %d Destino = %d Costo = %d\n", O, D, Costo;
    let Matriz [a,b]:= Costo;
    let Matriz [b,a]:= Costo;
    for {(i, j) in Arcos : x[i, j]!=0} {
        printf "x[%2d, %2d] = %d\n", i, j, x[i,j];
    };
};
display Matriz;

```


ANEXO C. Algoritmo de Resolución del MSPP mediante la Formulación Binaria

En este anexo se muestra el modelo y el algoritmo de que resuelve el MSPP utilizando programación lineal entera binaria, implementado en lenguaje AMPL.

```
##Modelo binario del MSPP##
```

```
#Definición de Conjuntos y Parámetros#
```

```
set Nodos;
```

```
param NS:= 1000;
```

```
set Arcos within {Nodos,Nodos} default {};
```

```
set Q{1..NS} within Nodos default {};
```

```
set SP{1..NS} within Nodos default {};
```

```
set SY{1..NS} within Nodos default {};
```

```
set MM{Nodos} within (Nodos) cross (Nodos) default {};
```

```
#MM contiene sólo los arcos candidatos
```

```
param c{(i,j) in Arcos};
```

```
param h{Nodos};
```

```
param s;
```

```
param t;
```

```
param T {Nodos, Nodos};
```

```
param KK;
```

```
param AA;
```

```
param MMS;
```

```
param P;
```

```
#Definición de Variables de Decisión#
```

```
var x{Arcos} binary;
```

```
var y{i in Nodos, j in Nodos diff {s,t}: (i,j) in MM[j]} binary;
```

#Función Objetivo#

minimize Costo: $\sum\{(i,j) \text{ in Arcos}\} P * c[i,j] * x[i,j] + \sum\{i \text{ in Nodos}, j \text{ in Nodos diff } \{s,t\}: (i,j) \text{ in MM}[j]\} h[j] * T[i,j] * y[i,j];$

#Restricciones#

s.t. Inicio: $\sum\{(s,j) \text{ in Arcos}\} x[s,j] = 1;$

s.t. Final: $\sum\{(i,t) \text{ in Arcos}\} x[i,t] = 1;$

s.t. Balance $\{j \text{ in Nodos diff } \{s,t\}\}:$

$$\sum\{(i,j) \text{ in Arcos}: i \neq t\} x[i,j] - \sum\{(j,k) \text{ in Arcos}: k \neq s\} x[j,k] = 0;$$

s.t. Asignacion $\{j \text{ in Nodos diff } \{s,t\}\}:$

$$\sum\{(i,j) \text{ in Arcos}: i \neq t \text{ and } j \neq s\} x[i,j] + \sum\{i \text{ in Nodos diff } \{j\}: (i,j) \text{ in MM}[j]\} y[i,j] = 1;$$

s.t. Prohibición $\{i \text{ in Nodos}, j \text{ in Nodos diff } \{s,t\}: (i,j) \text{ in MM}[j] \text{ and } j \neq s \text{ and } i \neq t \text{ and } i \neq s\}:$

$$y[i,j] - \sum\{(u,i) \text{ in Arcos}: u \neq t \text{ and } i \neq s\} x[u,i] \leq 0;$$

s.t. Subtour $\{g \text{ in } 1..KK\}:$

#Quiebra Subtours de cardinalidad

$$\sum\{i \text{ in } Q[g], j \text{ in } Q[g] \text{ diff } \{i\}: (i,j) \text{ in Arcos}\} x[i,j] \leq \text{card}(Q[g]) - 1;$$

s.t. Subtour2 $\{f \text{ in } 1..AA\}:$

#Genera conectividad del path

$$\sum\{i \text{ in SP}[f], j \text{ in Nodos diff SP}[f]: (i,j) \text{ in MM}[j]\} y[i,j] + \sum\{i \text{ in SP}[f], j \text{ in Nodos diff SP}[f]: (i,j) \text{ in Arcos and } i \neq t\} x[i,j] \geq 1;$$

s.t. Subtour3 $\{w \text{ in } 1..MMS\}:$

#Genera conectividad hacia los ciclos

$$\sum\{i \text{ in Nodos diff SY}[w], j \text{ in SY}[w]: (i,j) \text{ in MM}[j]\} y[i,j] + \sum\{i \text{ in Nodos diff SY}[w], j \text{ in SY}[w]: (i,j) \text{ in Arcos and } i \neq t\} x[i,j] \geq 1;$$

##Algoritmo de Resolución del MSPP binario##

#Llamada al modelo y opciones de CPLEX#

model median_path_CEAS.mod;

data pmed01.dat;

option cplex_options 'timing=1 integrality=0 mipgap=0';

option show_stats 1;

problem median_path: Costo, x, y, Inicio, Final, Balance, Asignacion, Prohibicion,
Subtour, Subtour2, Subtour3;

#Parámetros y Conjuntos#

set B within Arcos default {}; #B contiene los arcos del path principal

set A within {Nodos,Nodos} default {}; #A contiene los arcos secundarios

set FF within Arcos default {};

set ESE within Nodos default {};

set ESEY within Nodos default {};

set ESEP within Nodos default {};

set CHAO within Nodos default {};

set ENE within Nodos default {}; #ENE contiene todos los nodos de la red

set NP within Nodos default {}; #NP contiene los nodos del path principal

set Fuera within Nodos default {};

let s:=17; #nodo origen

let t:=66; #nodo destino

let KK:=0;

let AA:=0;

let MMS:=0;

param jj;

param ii;

param ff;

param TCPU; #Tiempo de CPU

let TCPU:=0;

param Iter;

let Iter:=0;

#Parámetros y conjuntos para calcular las SNI#

param N_Max:=200; #maximo de soluciones no inferiores

param XI {1..N_Max};

param YI {1..N_Max};

param XD {1..N_Max};

param YD {1..N_Max};

param SW {1..N_Max};

param SNI;

let SNI:=1;

let XI[SNI]:= 248;

let YI[SNI]:= 382754766;

let XD[SNI]:= 4510;

let YD[SNI]:= 0;

let SW[SNI]:= 1;

param ind;

param posi;

param path;

param media;

param const;

let ind:= 1;

let posi:= XI[1];

#Inicio del proceso de búsqueda de SNI libre de subtours#

repeat{

let $P := (YI[ind] - YD[ind]) / (XD[ind] - XI[ind]);$ #calcula el valor de la pendiente

let SW[ind]:=0;

#Inicio de algoritmo de eliminación de subtours

repeat{

let B:={};

let A:={};

let NP:= {};

solve median_path;

let Iter:= Iter+1;

let TCPU:= TCPU + _solve_user_time;

printf "Iteracion= %d Tiempo= %f Acumulado= %f\n", Iter, _solve_user_time, TCPU;

#Imprime tiempos de CPU e iteraciones

#Búsqueda del path principal ordenado

let FF:={};

let ii:=s;

repeat{

let {(ii,j) in Arcos: j in Nodos and $x[ii,j]=1$ }}jj:=j;

printf "x[%2d,%2d]=%f\n", ii, jj, $x[ii,j]$;

let FF:= FF union {(ii,j)};

let ii:=jj;

}while ii!=t;

printf "\n";

for {(i,j) in Arcos diff FF: $x[i,j] \neq 0$ } {

printf "x[%2d,%2d]=%f\n", i,j, $x[i,j]$;

};

#Impresión del path principal ordenado

#Imprimir las asignaciones

for {i in Nodos, j in Nodos diff {s,t}: (i,j) in MM[j] and $y[i,j] \neq 0$ } {

```

    printf "y[%2d,%2d]= %7.3f\n", i, j, y[i,j];
};

#Búsqueda de Arcos del path principal
for {(i,j) in Arcos: x[i,j]!=0} {
    let B:= B union {(i,j)};
    let NP:= NP union {i,j};
};

#Búsqueda de Arcos secundarios
for {i in Nodos, j in Nodos diff {s,t}: (i,j) in MM[j] and y[i,j]!=0} {
    let A:= A union {(i,j)};
};

#Búsqueda de los nodos del path principal
let conta:=0;
let ESE:={s};
repeat{
    let jj:=0;
    let {(i,j) in B: i in ESE and j not in ESE}jj:=j;
    if jj != 0 then {
        let ESE:= ESE union {jj};
    };
}while jj!=0 and jj!=t;

#ESE contiene los nodos del path principal
#Búsqueda de los nodos del path principal + nodos de arcos secundarios
let ESEP:=ESE;
repeat{
    let jj:=0;
    let {(i,j) in A: i in ESEP and j not in ESEP}jj:=j;
    if jj != 0 then {
        let ESEP:= ESEP union {jj};
    };
}while jj != 0;

```

#ESEP contiene los nodos del path principal + los nodos de los arcos secundarios

#SP Aplica la restricción de conectividad desde el conjunto del path a los demás

#nodos de la red

```
if card(ESEP) != card(Nodos) then {  
  let AA:=AA+1;  
  let SP[AA]:=ESEP;  
};
```

#Elimina ciclos con la restricción Subtour

```
let ENE:= NP diff ESE;
```

```
if ENE not within {} then {
```

```
  let Fuera:= {};
```

```
  for{k in ENE} {
```

```
    if k not in Fuera then {
```

```
      let CHAO:= {k};
```

```
      repeat{
```

```
        let jj:=0;
```

```
        let {(i,j) in B: i in CHAO and j not in CHAO} jj:=j;
```

```
        if jj!=0 then
```

```
          let CHAO:= CHAO union {jj};
```

```
        else {
```

```
          let ii:=0;
```

```
          let {(i,j) in B: i not in CHAO and j in CHAO} ii:=i;
```

```
          if ii!=0 then
```

```
            let CHAO:= CHAO union {ii};
```

```
        };
```

```
      }while jj!=0 or ii!=0;
```

#CHAO contiene los nodos que forman subtours minimales y despues se

#aplica la restriccion de subtour

```
let KK:=KK+1;
```

```
let Q[KK]:=CHAO;
```

```

let Fuera:= Fuera union CHAO;
    let ESEY:=CHAO;
    repeat{
        let jj:=0;
        let {(i,j) in A: i in ESEY and j not in ESEY}jj:=j;
        if jj != 0 then {
            let ESEY:= ESEY union {jj};
        };
    }while jj != 0;
#ESEY contiene los nodos capturados en CHAO más los arcos
#secundarios asociados
    if card(ESEY) != card(Nodos) then {
        let MMS:=MMS+1;
        let SY[MMS]:=ESEY;
        #SP Aplica la segunda restriccion de conectividad
        #a cada conjunto aislado le debe entrar un arco
        };
    };
};
};
};
}while ENE not within {};
#Término de la Eliminación de Subtours#

#Búsqueda de SNI en intervalos#

let const:= sum{(i,j) in Arcos}c[i,j]*x[i,j];
let path:= const;
let media:= sum{i in Nodos, j in Nodos diff {s,t}: (i,j) in MM[j]}h[j]*T[i,j]*y[i,j];
printf "\nBuscando entre (%d, %d) y (%d, %d)\n", XI[ind], YI[ind], XD[ind], YD[ind];
printf "Median = %.2f\n", sum{i in Nodos, j in Nodos diff {s,t}: (i,j) in
    MM[j]}h[j]*T[i,j]*y[i,j];

```



```
printf "Path = %.2f\n", sum{(i,j) in Arcos}c[i,j]*x[i,j];  
printf "\n";
```

```
if XI[ind] < const && const < XD[ind] then {
```

```
    let SNI:= SNI+1;  
    let XI[SNI]:= XI[ind];  
    let YI[SNI]:= YI[ind];  
    let XD[SNI]:= const;  
    let YD[SNI]:= media;  
    let SW[SNI]:= 1;
```

```
    printf "\nEntre (%d, %d) y (%d, %d)\n", XI[SNI], YI[SNI], XD[SNI], YD[SNI];
```

```
    let SNI:= SNI+1;  
    let XI[SNI]:= const;  
    let YI[SNI]:= media;  
    let XD[SNI]:= XD[ind];  
    let YD[SNI]:= YD[ind];  
    let SW[SNI]:= 1;
```

```
    printf "\nEntre (%d, %d) y (%d, %d)\n", XI[SNI], YI[SNI], XD[SNI], YD[SNI];
```

```
}
```

```
else{
```

```
    let posi:= XD[ind];
```

```
};
```

```
let ind:=1;
```

```
repeat{
```

```
    let ind:=ind+1;
```

```
    if ind > SNI then
```

```
        break;
```

```
    }while XI[ind]!=posi or SW[ind]!=1;
}while posi != XD[1];
for{i in 1..SNI}{
    printf "%2d (%4d, %9d) y (%4d, %9d) %2d\n", i, XI[i],YI[i],XD[i],YD[i], SW[i];
};
```

ANEXO D. Algoritmo de Resolución del MSPP mediante la Formulación basada en Flujo Multicommodity

En este anexo se muestra el modelo y el algoritmo de que resuelve el MSPP mediante flujo multicommodity, implementado en lenguaje AMPL.

##Modelo Multicommodity del MSPP##

#Definición de Parámetros y Conjuntos#

```
set Nodos;
set Arcos within (Nodos) cross (Nodos);
set NC within (Nodos) default {};
set AC within (Nodos) cross (Nodos) default {};
param T {Nodos,Nodos};
set MM {Nodos} within (Nodos) cross (Nodos) default {};
param h {Nodos};
param O;
param D;
param c {Arcos} >= 0;
param P;
```

#Definición de Variables de Decisión

```
var x {(i,j) in AC} binary;
var y {i in Nodos, k in Nodos diff {O,D}: (i,k) in MM[k]} binary;
var f {k in Nodos diff {O,D}, (i,j) in MM[k] union AC: i!=k and j!=O} >=0;
```

#Función Objetivo#

```
minimize Costo: sum {k in Nodos diff {O,D}, (i,k) in MM[k]} h[k] *T[i,k]*y[i,k]
               + P* sum {(i,j) in AC} c[i,j]*x[i,j];
```

#Restricciones#

subject to InicioX: $\sum\{(O,j) \text{ in } AC\}x[O,j]=1;$

subject to FinalX: $\sum\{(i,D) \text{ in } AC\}x[i,D]=1;$

subject to Continuidad{j in NC diff {O,D}}:

$$\sum\{(i,j) \text{ in } AC: i!=D\}x[i,j]=\sum\{(j,g) \text{ in } AC: g!=O\}x[j,g];$$

subject to Inicio{k in Nodos diff {O,D}}:

$$\sum\{(O,j) \text{ in } MM[k] \text{ union } AC\} f[k,O,j] = 1;$$

subject to Final {k in Nodos diff {O,D}}:

$$\sum\{(i,k) \text{ in } MM[k] \text{ union } AC\} f[k,i,k] = 1;$$

subject to Balance {k in Nodos diff{O,D}, j in Nodos diff {k,O,D}}:

$$\begin{aligned} \sum\{(i,j) \text{ in } AC: i!=k\} f[k,i,j] &= \sum\{(j,g) \text{ in } MM[k] \text{ union } AC: g!=O \text{ and } g!=D\} f[k,j,g] \\ &+ \sum\{(j,g) \text{ in } MM[k] \text{ union } AC: g=D \text{ and } (g,k) \text{ in } MM[k]\} f[k,j,g]; \end{aligned}$$

subject to Balance1 {k in Nodos diff{O,D}: (D,k) in MM[k]}:

$$\sum\{(i,D) \text{ in } AC: i!=k\} f[k,i,D] = f[k,D,k];$$

subject to Resto1{k in Nodos diff {O,D}, (i,k) in MM[k] inter AC: i!=k }:

$$f[k,i,k] \leq x[i,k]+y[i,k];$$

subject to Resto2{k in Nodos diff {O,D}, (i,j) in AC diff MM[k]: i!=k and j!=O and j!=D}:

$$f[k,i,j] \leq x[i,j];$$

subject to Resto22{k in Nodos diff {O,D}, (i,D) in AC diff MM[k]: i!=k and (D,k) in

$$MM[k]: f[k,i,D] \leq x[i,D];$$

subject to Resto3{k in Nodos diff {O,D}, (i,k) in MM[k] diff AC: i!=k }:

$$f[k,i,k] \leq y[i,k];$$

subject to Cubrir{k in Nodos diff{O,D}}:

$$\sum\{(i,k) \text{ in } AC\}x[i,k] + \sum\{(i,k) \text{ in } MM[k]\}y[i,k]=1;$$

#Algoritmo de Resolución del MSPP Mediante Flujo multicommodity#

#Llamada al Modelo y Opciones de CPLEX#

model MP-COMMODITY-CES.mod;

data pmed01.dat;

problem MP: Costo, x, y, f, Cubrir, Balance, Balance1, Inicio, Final, Resto1, Resto2, Resto22, Resto3, InicioX, FinalX, Continuidad;

option show_stats 1;

option cplex_options 'timing=1 integrality=0 mipgap=0';

#Parámetros y Conjuntos del Algoritmo#

let O:=40;

let D:=97;

param N_Max:=200;

param XI {1..N_Max};

param YI {1..N_Max};

param XD {1..N_Max};

param YD {1..N_Max};

param SW {1..N_Max};

param SNI;

let SNI:=1;

let XI[SNI]:= 299;

let YI[SNI]:= 379972038;

let XD[SNI]:= 4503;

let YD[SNI]:= 0;

let SW[SNI]:= 1;

param ii;

param jj;

param ind;

param posi;

param CC;

```

param TCPU;
let TCPU:=0;
param Iter;
let Iter:=0;
param arc_sec;
let ind := 1;
let posi:= XI[1];

#Inicio del proceso de búsqueda de SNI#
repeat {
    let P:= (YI[ind]-YD[ind])/(XD[ind]-XI[ind]);
    let SW[ind]:=0;
    let AC:= {};
    let NC:= {};
    for{(i,j) in Arcos: j!=O and i!=D}{
        let AC:= AC union {(i,j)};
        let NC:= NC union {i} union {j};
    };
    solve MP;
    let Iter:= Iter+1;
    let TCPU:= TCPU + _solve_user_time;
    printf "Iteracion= %d   Tiempo= %f   Acumulado= %f\n", Iter, _solve_user_time,
TCPU;
    let CC:=sum{(i,j) in AC} c[i,j]*x[i,j];
    printf "\nBuscando entre (%d,%d) y (%d,%d)\n",XI[ind],YI[ind],XD[ind],YD[ind];
    printf "Median= %.2f\n", sum {k in Nodos diff {O,D}, (i,j) in MM[k]} h[k] *T[i,j]*y[i,j];
    printf "Path = %.2f\n",sum{(i,j) in AC} c[i,j]*x[i,j];
    printf "\nMP.result=%s\n", MP.result;
    #Si el problema es infactible, entonces, imprime la solución, en caso contrario, no#
    if MP.result = "infeasible" then {
        let CC:=1;

```

```

}
else {
    let ii:=O;
    repeat{
        let {(ii,j) in AC: x[ii,j]!=0}jj:=j;
        printf "x[%2d,%2d]= %.2f\n", ii,jj,x[ii,jj];
        let ii:=jj;
    }while ii!=D;
    printf "\n";
    for{k in Nodos diff {O,D}, (i,k) in MM[k]: y[i,k]!=0}{
        printf "y[%2d,%2d]= %.2f\n", i,k, y[i,k];
    };
};

if XI[ind] < CC && CC < XD[ind] then {
    let SNI:= SNI+1;
    let XI[SNI]:= XI[ind];
    let YI[SNI]:= YI[ind];
    let XD[SNI]:= CC;
    let YD[SNI]:= sum {k in Nodos diff {O,D}, (i,j) in MM[k]} h[k] *T[i,j]*y[i,j];
    let SW[SNI]:=1;
    printf "\nEntre (%d,%d) y (%d,%d)\n",XI[SNI],YI[SNI],XD[SNI],YD[SNI];
    let SNI:= SNI+1;
    let XI[SNI]:= CC;
    let YI[SNI]:= sum {k in Nodos diff {O,D}, (i,j) in MM[k]} h[k] *T[i,j]*y[i,j];
    let XD[SNI]:= XD[ind];
    let YD[SNI]:= YD[ind];
    let SW[SNI]:=1;
    printf "Entre (%d,%d) y (%d,%d)\n",XI[SNI],YI[SNI],XD[SNI],YD[SNI];
}

else {
    let posi:= XD[ind];

```

```

};
let ind:=1;
repeat{
    let ind:=ind+1;
    if ind > SNI then
        break;
    }while XI[ind]!=posi or SW[ind]!=1;
}while posi != XD[1];
#Término del proceso de búsqueda de SNI e inicio de impresión de resultados#
for{i in 1..SNI}{
    printf "%2d  (%4d,%8d) y (%4d,%8d) %2d\n", i, XI[i],YI[i],XD[i],YD[i], SW[i];
};

```