TESIS

presentada el día 9 de Julio de 2010 en la

Universidad de La República, UdelaR

para obtener el título de

MAGISTER EN INGENIERÍA MATEMÁTICA

para

Darío PADULA

Instituto de Investigación : LPE - IMERL
Componentes universitarios :
UNIVERSIDAD DE LA REPÚBLICA
FACULTAD DE INGENIERÍA

Título de la tesis :

Compromiso entre Pares e ISPs en el contexto P4P: Optimización en dos niveles.

defensa realizada el 29 de Julio de 2010 por el comité de examinadores

Dr. Pablo	RODRÍGUEZ-BOCCA	Director de Tesis
Dr. Franco	ROBLEDO AMOZA	Director Académico
MSc. Ing. María	URQUHART	Presidente
MSc. Ing. Omar	VIERA	
Dr. Juan	KALEMKERIAN	
Dr. Gustavo	GUERBEROFF	
Dra. Libertad	TANSINI	

Agradecimientos

Quiero agradecer a mis tutores Dr. Franco Robledo y Dr. Pablo Rodríguez-Bocca, por la constante motivación e invaluables aportes durante el proceso de esta tesis. A los ingenieros Pablo Romero, Andrés Corez y la Lic. Elisa Bertinat no solo por sus invalorables contribuciones sino también por su gran compañerismo y dedicación. Al Dr. Eduardo Canale por su desinteresado aporte que resultó de gran utilidad para la forma de abordar el problema. A los ingenieros Matías Barrios y Claudia Rostagnol por brindarme la información necesaria para evaluar mis resultados.

Al Ing. Francois Despaux por darme una mano siempre que la necesité.

A la empresa de telecomunicaciones de la República Oriental del Uruguay (ANTEL), ya que este trabajo se realizó como parte de la actividad específica "*Sistema eficiente de distribución de video y TV en tiempo real*" en el marco del convenio entre esta y la Facultad de Ingeniería de la Universidad de la República.

Por último quiero agradecer a mi novia, familiares, amigos y compañeros de trabajo por hacerme el aguante y darme ánimo en todo momento.

Abstract

El diseño de ruteos para redes peer-to-peer (P2P) tiene muchos desafíos. Los usuarios (llamados *pares* o *peers*) exigen alta *Calidad de Experiencia* en las aplicaciones que utilizan (compartir archivos, video streaming, etc.) generando un alto tráfico en la red, el cual por lo general no contempla las limitaciones de la red de Internet subyacente. Los ISPs (Internet Service Providers) mantienen su negocio ofreciendo un buen servicio para los usuarios finales tratando de utilizar de forma eficiente sus recursos (especialmente sus enlaces internacionales que son los recursos más costosos). Una estrategia reciente que considera conjuntamente los objetivos de los pares e ISPs es la llamada $P4P$ [50] (*Proactive Provider Participation*). Esta estrategia coloca el máximo tráfico total en la red, reduciendo al mismo tiempo el porcentaje de la utilización del enlace internacional más congestionado.

En este trabajo el problema multióbjetivo P4P es resuelto de forma aproximada mediante un algoritmo FPTAS (*Fully Polynomial Time Approximation Scheme*) cuando el objetivo es compartir un sólo contenido en la red. Una metaheurística basada en $GRASP$ (*Greedy Randomized Adaptive Search Procedure*) es aplicada para resolver el problema general, cuando los pares comparten múltiples contenidos. Utilizando instancias generadas aleatoriamente se obtienen muy buenos resultados contrastando las soluciones obtenidas con cotas conocidas.
Finalmente se muestran resultados de emulaciones en una plataforma P2P real llamada GoalBit, contrastando los resultados obtenidos con la metodología P4P versus una metodología de ruteo aleatoria. Los resultados son alentadores, justificando el uso de P4P, ya que se logra reducir la utilización de los enlaces internacionales al mismo tiempo que aumenta la transferencia de flujo neto de los usuarios.

Índice

1 Introducción **11**

2 Marco Teórico **15**
2.1 Redes P2P 15
2.1.1 BitTorrent 15
2.1.2 Streaming de Video P2P 17
2.1.3 Plataforma GoalBit 17
2.2 El Modelo P4P 19
2.2.1 Introducción 19
2.2.2 Problema Multiobjetivo 22
2.3 Trabajos relacionados 27
2.3.1 Investigaciones previas de P4P 27
2.3.2 Resultados previos de implementaciones P4P 28
2.3.3 Enfoque Ono 29

3 Problemas a abordar **31**
3.1 Introducción 31
3.2 Problema General (PG) 31
3.3 Un contenido, conocimiento completo... 32
3.3.1 Variante P1: utilización fija de los enlaces ($VP1_\rho$) 33
3.3.2 Variante P1: Ruteos por caminos de largo uno ($VP1$) . . . 34
3.4 Un contenido, tráfico de fondo parcialmente... 35
3.5 Un contenido, tráfico de fondo y capacidades parcialmente conocidas (Problema 3 o P3) 36

4 Propiedades y algoritmos para los problemas de un contenido **37**
4.1 Instancias y propiedades 37
4.2 El Problema 3 es polinomial 40
4.3 Algoritmo de GOTEO para el Problema 2 (P2) 45

4.3.1 Waterfilling 45
4.3.2 Algoritmo de GOTEO 46
4.4 Algoritmo Goloso para el Problema 1 (P1) 49
4.4.1 Greedy y su desempeño en otros problemas 49
4.4.2 Algoritmo goloso simple para P4P 50
4.4.3 Contraejemplos de optimalidad para el algoritmo goloso simple 51
4.4.4 Construcción golosa saturando enlaces 52
4.4.5 Greedy con Waterfilling 55
4.5 Mejora de la solución 55
4.5.1 Algoritmo de mejora o completitud 55

5 Algoritmos basado en Ford-Fulkerson 61
5.1 Ford-Fulkerson 62
5.1.1 Algoritmo de Ford-Fulkerson 62
5.1.2 Red auxiliar 63
5.2 Variantes del Problema 1 68
5.2.1 Variante 1: Utilización fija de los enlaces de la red ($VP1_\rho$) 68
5.2.2 Variante 2: Considerando sólo caminos de largo uno ($VP1$) 69
5.3 Algoritmo para resolver el Problema (VP1) 71
5.3.1 Pasos del algoritmo 72
5.4 Algoritmo para resolver el Problema 1 (P1) 78
5.4.1 Ruteos que forman ciclos 79

6 Algoritmos para el Problema General (PG) 83
6.1 Enfoque Metaheurístico 84
6.1.1 Algoritmo Basado en Simplex para Múltiples Contenidos . 84
6.1.2 Algoritmos Genéticos 87
6.1.3 GRASP 88
6.2 GRASP: algoritmo detallado para resolver el Problema General (PG) 91
6.2.1 Fase 1 (Construcción de una solución factible) 92
6.2.2 Fase 2 (Búsqueda Local) 93

7 Resultados 97
7.1 Generación de Instancias para el Problema 1 (P1) basadas en datos reales 97
7.2 Implementación de distintos algoritmos para el Problema 1 99
7.2.1 Generalidades de los algoritmos propuestos e implementados 100
7.3 Comparación de resultados 101
7.4 Utilización de la solución en GoalBit 103

7.4.1 Emulaciones en GoalBit 105
7.5 Resultados para el Problema General 108
7.5.1 Generación de Instancias 110
7.5.2 Cota para el Problema General 111
7.5.3 Resultados para el Problema General 112
7.6 Conclusiones 119

8 Conclusiones **121**
8.1 Conclusiones y contribuciones 121
8.2 Trabajo a futuro 124

Capítulo 1

Introducción

Las redes *peer-to-peer* (P2P) son redes virtuales que se montan sobre la infraestructura de Internet. Estas imponen un paradigma que contrasta los clásicos modelos cliente-servidor, como HTTP y WWW. La esencial distinción se halla en el hecho que los pares (o nodos pertenecientes a dicha red) también juegan el rol de servidores, aliviando tareas y capacidad de procesamiento de un gran servidor central, y logrando escalabilidad. Los pares brindan entonces recursos (ancho de banda y capacidad de procesamiento) a otros pares, básicamente debido a que comparten intereses en común. Los usuarios poseen beneficios evidentes del uso de estas redes, que permiten aplicaciones de video interactivas, voz, música, tareas distribuidas y muchas otras. No obstante, el diseño de estas redes son un gran desafío desde el punto de vista de la ingeniería de tráfico.

Las ofertas crecientes de banda ancha a nivel mundial facilitan el emergente desarrollo de infraestructuras P2P, donde el ancho de banda de los pares es utilizado para la distribución de contenidos. Existen muchos ejemplos de sistemas P2P muy populares para compartir y distribuir archivos como BitTorrent [4], eMule [20], KaZaA [29], etc, por otro lado existen redes P2P asociadas al streaming (distribución de audio y video en tiempo real), algunas de estas son PPlive [38], SopCast [45], CoolStreaming [13], TVAnts [48] y GoalBit [3] entre otras. Sin embargo, la gestión de recursos para el diseño de las redes P2P debe enfrentar la variabilidad inherente de sus recursos (los pares se desconectan y conectan cuando lo desean), los requerimientos de las diversas aplicaciones que se ofrecen (que pueden incluir bajos retardos y fluctuaciones, tiempo real, altos anchos de banda), la presencia de pares "parásitos", o *free riders* [37] y el uso de enlaces internacionales, entre muchos otros. En este documento atenderemos esencialmente uno de los problemas antes enumerados: *el creciente uso de enlaces internacionales, debido al tráfico que proviene de redes P2P*.

Recientes investigaciones señalan que existen importantes segmentos de Internet seriamente congestionados, y el peso del tráfico proveniente de las redes P2P juega un factor determinante [8]. Existen varios enfoques para controlar el tráfico P2P. Políticas extremistas plantean el descarte de este tipo de tráfico, o el uso de servicios diferenciados, brindando prioridad a aplicaciones que merecen celeridad de entrega (Packet Shaper, por ejemplo [5]). Otros proponen redirigir tráficos sin instruir en el contenido de los mismos, y guiando los paquetes en base a la sustitución de direcciones de los paquetes IP (LiteLoad [27]). La estrategia que consideraremos en este documento es de planificación del protocolo de ruteo, en base a P4P (Participación Proactiva del Proveedor en redes P2P [50]), esta metodología facilita a través de portales la comunicación explícita entre aplicaciones P2P e ISPs para que ambos resulten beneficiados en sus intereses, generando ruteos más eficientes incentivando la comunicación entre pares geográficamente cercanos, logrando así, una mejor performance para el tráfico de contenidos P2P y un uso más eficiente de los recursos de la red. El problema matemático contempla dos objetivos contrastados:

1. Los pares desean maximizar su transferencia neta, sin preocuparse del recurso utilizado.

2. Los proveedores del servicio de Internet (de ahora en más los llamaremos ISPs por sus siglas en ingles Internet Networks Providers) que se sustenta de la utilización de los enlaces internacionales. De hecho, van a desear que el uso de estos enlaces debido a redes P2P sea el menor posible, o al menos, que no genere congestiones.

Se introducirán nuevos modelos matemáticos derivados del original de P4P [50], para los cuales se desarrollarán distintos algoritmos para resolverlos, generando además instancias de prueba para comparar el desempeño de los distintos algoritmos presentados en este documento. Se indicarán pautas de implementación de la metodología P4P en la plataforma real P2P para video steaming GoalBit [3] y mostraremos resultados de emulaciones en dicha plataforma los cuales justifican el uso de P4P.

Este documento se organiza en capítulos. En el Capítulo 2 se describe el funcionamiento de dos sistemas P2P de protocolos abiertos, BitTorrent (utilizado para compartir y descargar archivos) y GoalBit (video streaming), se resume la idea de P4P tal como se muestra en su trabajo original [50]. Se revisa brevemente la propuesta, introduciendo la política de control y de datos en base a portales que facilitan la comunicación entre las aplicaciones P2P y los ISPs. Concluye con una

formulación matemática del problema multiobjetivo entre aplicaciones P2P e ISPs el cual se toma aquí como modelo inicial para introducir nuevos modelos con diferentes hipótesis. Por último se comenta sobre trabajos ya desarrollados donde se aplicó P4P obteniéndose muy buenos resultados.
Es crítica la disponibilidad de conocimiento de la red, pues al montarse sobre la infraestructura de Internet la determinación de las capacidades de los enlaces (que pueden variar con el tiempo) es un problema en sí mismo. Cabe destacar que existe otra metodología llamada Ono [10] la cual comentaremos al final del Capítulo 2 que intenta cumplir con los mismos objetivos que P4P y tiene la virtud de no necesitar información exhaustiva de la red.

En el Capítulo 3 se formalizan los distintos problemas que se abordarán en este documento los cuales denotaremos indistintamente como *Problema General* o *PG*, *Problema 1* o *P1*, *Problema 2* o *P2* y el *Problema 3* o *P3*.
Además se proponen dos variantes del Problema 1 las cuales las denotaremos como $VP1_\rho$ y $VP1$. En el Problema General se admite un conocimiento completo de la red y el ruteo simultaneo de múltiples contenidos mientras que en el resto de los problemas sólo se atiende el ruteo de un contenido. En el Problema 1 se admite un conocimiento completo de la red así como sus dos variantes ($VP1_\rho$ y $VP1$) las cuales difieren de este determinando una utilización máxima para los enlaces ($VP1_\rho$) y restringiendo el ruteo de tráfico por caminos de largo 1 ($VP1$). Los Problemas P2 y P3 se centran en el diseño de tráfico saliente de un nodo destacado, que es el local (en nuestro caso Uruguay) y no admiten un conocimiento completo de la red.

En el Capítulo 4 se estudia la complejidad del P3. Se desarrolla un algoritmo que introduce una técnica llamada *waterfilling* para la utilización óptima de recursos de la red. Además se propone una metodología golosa simple de resolución del Problema 1, y se analiza su desempeño. El esfuerzo en este capítulo se centra en comprender la naturaleza de la estructura del problema a resolver. Se introducen conceptos claves para la definición posterior de una metaheurística a aplicar. Entre ellos se hallan, la diferenciación de instancias *optimistas* y *pesimistas*, y la Propiedad de Completación de una solución.

En el Capítulo 5 se formaliza la definición de un algoritmo basado en Ford-Fulkerson desarrollado para la resolución del Problema 1 y sus dos variantes. Se demuestra que el algoritmo propuesto es un $FPTAS$ (*Fully Polynomial Time Approximation Scheme*) [1] para los problemas P1, P2, P3 y VP1, lo cual nos permite aproximarnos en tiempo polinomial tanto como queramos a soluciones óptimas para cada problema. También se demuestra que el problema $VP1_\rho$ se puede resol-

ver de forma óptima en tiempo polinomial.

En el Capítulo 6 se plantea el Problema General para múltiples contenidos, el cual muestra indicios de que es un problema $NP-$ Hard, hecho que motiva a su resolución mediante un enfoque metaheurístico. En primera instancia se revisan distintas opciones e ideas, eligiendo finalmente una metaheurística basada en la generalización del algoritmo presentado en el Capítulo 5 para resolver el Problema 1, introduciendo selección de contenidos al azar. La metaheurística propuesta sigue un esquema GRASP (del inglés *Greedy Randomized Adaptive Search Procedure* [7, 21, 34, 36, 41]).

En el Capítulo 7 se muestran resultado de comparaciones de distintos algoritmos propuestos para resolver el Problema 1 frente a instancias obtenidas de datos reales. Se explica la implementación de soluciones del Problema General en una plataforma real denominada GoalBit [24] para distribuir video en tiempo real y se muestran resultados de emulaciones realizadas en esta plataforma los cuales motivan el uso de la metodología P4P. Además para probar la performance del algoritmo propuesto para resolver el Problema General, se generan instancias aleatorias y se compara el desempeño de este frente a cotas conocidas.

Por último en el Capítulo 8 se presentan las conclusiones finales de este trabajo y se propone el trabajo a futuro.

Capítulo 2

Marco Teórico

En este Capítulo describiremos brevemente las redes P2P, BitTorrent (transferencia de archivos) y GoalBit (video streaming). Se define la metodología P4P tal como se muestra en su trabajo original [50] y se dejan explícitos los objetivos contrapuestos de los pares e ISPs definiendo así el problema multiobjetivo. Además se describen algunas investigaciones previas sobre P4P y sus desempeños. Por otro lado se introduce brevemente una estrategia que persigue el mismo fin que P4P llamada Ono [10].

2.1 Redes P2P

2.1.1 BitTorrent

El protocolo BitTorrent fue desarrollado por el programador Bram Cohen en el año 2001 [4]. Tiene un fuerte sustento con la teoría de juegos y de la información. La clave de su escalabilidad es la cooperación basada en incentivos: "dar para recibir" [11]. Su popularidad también se ve beneficiada por la simplicidad del uso de su cliente, y por ser uno de los pocos sistemas P2P de protocolo abierto para la descarga de archivos.
Actualmente BitTorrent mantiene su gran popularidad, logrando una porción importante dentro del tráfico P2P de Internet [8], un informe muestra que la mitad del tráfico de Internet es de BitTorrent [6]. Para transmitir los contenidos se procede cortando los mismos en cientos o miles de piezas de igual tamaño (chunks), y cada par mediante una estrategia de cooperación debe armar ese rompecabezas a partir de lo entregado por otros pares.

La red de BitTorrent consiste de un servidor web que aloja el contenido .torrent

estático, pares que representan los usuarios de la red cuya finalidad es descargar los contenidos y una entidad o nodo central denominada tracker que contiene la lista de peers que se encuentran descargando un archivo (contenido) determinado, que es el único punto de coordinación entre los pares que comparten el mismo contenido. El traker mantiene información actualizada del estado de los pares, y hace factible la comunicación entre ellos, brindando a cada nuevo usuario una lista denominada usualmente "swarm" (enjambre o nube de pares) con los cuales conectarse e intercambiar información, de los cuales el usuario elegirá sólo algunos. Los archivos .torrent tienen un campo que permite ubicar al tracker. De esta manera se procede consultando al mismo, quien posteriormente brinda el swarm al par, por lo general esta lista se hace de forma aleatoria. Una vez que el cliente de un par inicia la descarga, comienza la consulta de piezas del contenido a sus otros vecinos. Este par puede cooperar desde que descarga tan sólo una pieza. Así es que BitTorrent aprovecha el ancho de banda de cada par lo más temprano posible, posibilitando una visión de ancho de banda global en la red, y favoreciendo la escalabilidad. En la Figura 2.1 se puede ver la arquitectura de BitTorrent.

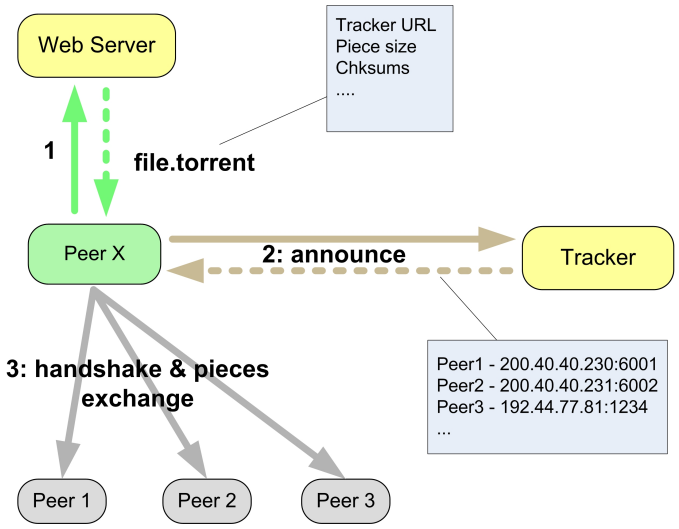


Figura 2.1: Arquitectura BitTorrent

El hecho de que el tracker genere la lista de pares (swarm) de forma aleatoria puede producir que se conecten pares que se encuentren geográficamente muy distantes generando caminos en la red muy largos. Este es uno de los motivos para el desarrollo de metodologías como P4P y Ono [10], las cuales intentan incentivar la

comunicación entre pares geográficamente cercanos.

2.1.2 Streaming de Video P2P

Hoy en día, existen redes comerciales P2P a disposición para la distribución de video streaming. Dentro de las más exitosas se encuentran PPLive [38] superando los 200000 usuarios concurrentes a un mismo canal, SopCast [45] con más de 100000 usuarios, tal como reportan sus desarrolladores, CoolStreaming [13] con más de 25000 usuarios sobre un mismo canal, PPstream [39], TVAnts [48], y TVUnetwork [49], todas de código y protocolos propietarios. Este tipo de redes presentan desafíos muy grandes para mantener la calidad del servicio, uno de estos es debido a la dinámica de los pares los cuales usualmente se conectan a la red sólo por algunos segundos [46]. A continuación describiremos algunos aspectos de la plataforma llamada GoalBit, la cual es la primera plataforma de protocolo gratuito y libre de sistemas P2P para la distribución de video streamming en tiempo real a través de Internet.

2.1.3 Plataforma GoalBit

Hoy en día la plataforma GoalBit [3] es utilizada por operadores como ANTEL en su portal de video streamming AdinetTV[1], y por los clientes finales para reproducir sus contenidos en vivo. GoalBit es una plataforma real que ofrece contenidos de video en vivo, y posee alto despliegue en Internet [3]. Su filosofía de diseño se basa en el gran éxito de BitTorrent para aplicaciones de descarga (downloading) [4, 16] aunque Goalbit no puede utilizar este mecanismo tal como lo implementa BitTorrent, ya que el contenido de GoalBit se compone de flujos continuos en vivo y no de archivos con un tamaño fijo que sólo pueden ser abiertos luego de completada su descarga. La forma como el protocolo de BitTorrent es modificado para contenido en vivo se explica en [3]. Para el caso de GoalBit, el contenido puede ser generado mediante diferentes dispositivos (tarjeta capturadora, cámara Web, otro stream $http/mms/rtp$, un archivo, etc.). El flujo es encapsulado en un *GoalBit Packetized Stream* ($GBPS$) [3] a partir del cual se generan los chunks (piezas de video) que son distribuidos mediante el *GoalBit Transport Protocol* ($GBTP$) [3]. La arquitectura de GoalBit mostrada en la Figura 2.2 está compuesta por 4 componentes diferentes, el *broadcaster*, los *super pares*, los *pares* y el *tracker*.

[1] http://adinettv.com.uy

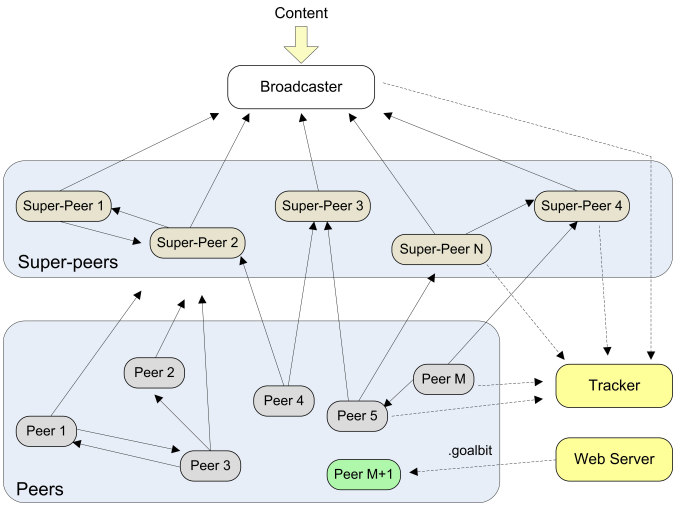


Figura 2.2: Arquitectura GoalBit

- El *broadcaster* es la pieza fundamental para el video streaming, este es quien recibe el contenido de alguna fuente (una señal análoga, una señal de streaming desde Internet, un archivo de videos almacenado, etc.), lo procesa y lo introduce en la red P2P. Si el broadcaster abandona la red el streaming automáticamente cae.

- Los *super pares* son pares altamente accesibles con altas capacidades de subida y son el nexo entre el broadcaster y los pares.

- Los *pares* son los usuarios finales del sistema. Ellos se conectan al streaming con el sólo propósito de reproducirlo, estos representan la gran mayoria de los nodos del sistema y exhiben un comportamiento muy variado en el tiempo, en particular se conectan y desconectan frecuentemente.

- El *tracker* que en el contexto de este trabajo lo llamaremos *iTracker*, mantienen un regitro de los pares conectados a cada canal (streaming). Cada vez que algún par se conecta a un canal, este se anuncia con el tracker, el cual registra al par en el conjunto de pares conectados al mismo canal (enjambre o swarm) y le devuelve al par una lista de pares candidatos para que se comunique.

2.2 El Modelo P4P

2.2.1 Introducción

Un problema básico en una arquitectura de red es cómo utilizar eficientemente los recursos para las aplicaciones de la misma (ej. compartir datos, streaming, e-mail, etc.), lo que se conoce como el *problema de control de tráfico*. La importancia de este problema se justifica en que una mala ingeniería de control de tráfico puede generar enlaces congestionados, utilización innecesaria de enlaces costosos además de un bajo rendimiento de la performance de servicio.

En las aplicaciones tradicionales punto a punto (ej. e-mail), en gran medida los ISPs se encargan del control de tráfico, calculando matrices de tráficos de gran tamaño, para determinar el ruteo. Las aplicaciones sólo definen el destino, mientras que los ISPs determinan por qué camino y cúanto tráfico mandar. En este contexto los ISPs pueden utilizar ingenierías de tráfico óptimas (ej. Multihoming[2]) para lograr un ruteo eficiente y satisfacer sus objetivos económicos.

Sin embargo, la llegada de las aplicaciones P2P (ej. streaming, intercambiar archivos, etc.) plantean grandes desafíos en el control de tráfico. A diferencia de las aplicaciones tradicionales, en muchas aplicaciones P2P, el ISP carece del conocimiento de las estrategias y de los pares que se están comunicando. Esto puede causar que el ISP genere tráfico innecesario dentro y entre los ISPs, así como saturación de enlaces que atraviesan el ISP (tráfico en la columna vertebral del ISP) y utilización innecesaria de links costosos (enlaces entre los ISPs). En segundo lugar, las aplicaciones P2P modifican su comportamiento de descarga para adaptarse a cambios en la red dificultando las estimaciones de las matrices de tráfico y por tanto generando decisiones de ruteo ineficientes. Es por estos motivos que la cooperación entre ISPs y aplicaciones P2P es crucial.

En resumen, para aplicaciones tradicionales el ISP es capaz de realizar el ruteo estimando matrices de demanda de tráfico con conocimiento completo de la red. Pero con altos niveles de tráfico P2P, el problema de control de tráfico tiene que ser resuelto conjuntamente entre éstos y las aplicaciones P2P.

Es por estos motivos que desde Julio del 2007 existe un grupo llamado *P4P Working Group*[3] (P4PWG) bajo la dirección de DCIA Member, compañías Pando

[2]Multihoming es una técnica de ruteo que divide el tráfico entre todos los puntos de acceso equitativamente, considerando el costo del tráfico que se agrega

[3]http://www.openp4p.net/front/p4pwg

Networks y Verizon Communications.
La misión de este grupo es trabajar de forma conjunta y cooperativa con los principales (ISPs), distribuidores de software (P2P) e investigadores de tecnologías para el uso de mecanismos P4P con el fin de acelerar la distribución de contenidos y optimizar la utilización de las recursos de las ISPs logrando la mejor performance posible para el usuario final.

P4P no es un protocolo[4], sino que es una estructura que permite a los ISPs optimizar el control de tráfico, principalmente debido a las redes P2P, que hoy día representa la gran mayoría del tráfico total de Internet [8]. La estructura P4P permite una mejor cooperación entre los pares y los ISPs a través de una comunicación explícita, generando una participación activa de los ISPs con los pares. Así, los pares brindarán información a los ISPs para que estos sugieran a los pares con quiénes conectarse e intercambiar información, de modo que ambos cumplan sus objetivos de la forma más eficiente. En términos generales, el objetivo de P4P es alcanzar la mejor performance de las aplicaciones, utilizando de forma justa y eficiente los recursos de la red y que ésto redunde en disminuciones de costos y mayores ingresos económicos para los ISPs.

Para que la estructura P4P tenga un buen desempeño y pueda satisfacer los objetivos de los pares y de los ISPs, esta debería contemplar los siguientes requerimientos en su diseño:

- Mejor performance P2P: usando información más exacta de la topología de la red, P4P debe ser capaz de identificar conexiones más eficientes.
- Uso eficiente de los recursos de la red: mediante la comunicación explícita entre pares y la red, P4P permite a las aplicaciones que usen la información de la topología de la red para reducir el tráfico en los enlaces costosos y así disminuir costos de los ISPs.
- Escalabilidad: P4P tiene que poder soportar un número grande de usuarios y de redes P2P en escenarios muy dinámicos.
- Robustez: Fallas en las componentes P4P podrían producir baja eficiencia, pero no que el sistema falle.
- Preservación de la privacidad: P4P debería preservar la privacidad de los ISPs cuando suministran información de la red.

[4]Los protocolos son reglas que se imponen para que el intercambio de datos funcione de forma eficiente (ej. BitTorrent [11] o eMule [20])

- Extensibilidad: Existen muchos tipos de aplicaciones P2P con diversas características (compartir archivos o streaming). P4P debería posibilitar el manejo de muchas aplicaciones con características y requerimientos diversos.

Diseño de una estructura P4P

La estructura P4P está compuesta por dos planos: el *plano de control* y el *plano de datos* .
En el plano de control, P4P introduce iTrackers los cuales son la pieza fundamental para el funcionamiento de la estructura. Estos ofician como portales para que los pares se comuniquen con los ISPs. El iTracker almacena información sobre el estado y la topología de la red, guías del proveedor o políticas y capacidades de la red. Esta información es vital para las decisiones de ruteo. Cada ISP mantiene uno o más iTrackers dependiendo del tamaño de su red.

En el plano de datos, P4P habilita a los routers a dar información precisa a los pares y así lograr un uso más eficiente de los recursos de la red. Específicamente, los routers pueden designar tasas de flujos para que el destinatario final ajuste su tasa en consecuencia. Esta componente es opcional. En este documento nos enfocaremos en el plano de control.

Plano de control de una estructura P4P

Este plano es la base de P4P y los iTrackers son la clave de esta estructura. Los iTrackers son utilizados como portales para que otros (ej. pares, appTrakers[5]) consulten por tres tipos de interfaces (info, policy y capability). Cada una de éstas presenta las siguientes características:

- La interfaz “info” permite a otros, por lo general a los pares dentro del ISP, obtener el estado y la topología de la red. Esta interfaz genera para cada dirección IP una terna ($ASID$, PID, LOC). $ASID$ es el identificador (ID) del ISP. PID es un ID que se asigna a un conjunto de nodos de la red. La asignación del PID se hace de esta manera para mantener la privacidad. Por último LOC es una coordenada virtual o geográfica del nodo.

- La interfaz “policy” permite a los pares y appTrackers obtener políticas y guías de la red. Las políticas son reglas que especifican los ISPs de cómo

[5]Los appTrakers se encargan de sugerirle a los pares con qué pares comunicarse

deben los usuarios utilizar la red, en general independientemente de las aplicaciones P2P. Las guías de la red son sugerencias específicas para que los pares utilicen los recursos de la red de forma eficiente.

- La interfaz "Capability" permite, por medio de los appTrackers, que los pares o los proveedores de contenidos soliciten capacidades a los ISPs. Un ISP puede proveer varios servicios o facilitar servidores en la red para utilizarlos como pares de modo de acelerar la distribución de contenidos P2P.

La cantidad de interfaces a implementar, la riqueza de la información transmitida y el control del acceso a las interfaces para mantener la privacidad son competencias de los ISPs.

Para entender la dinámica de la estructura P4P, mostraremos un ejemplo del modo en que son utilizadas las interfaces de los iTrackers. En la Figura 2.3 mostramos una aplicación P2P con appTracker utilizando las interfaces *info* y *policy* para solicitar la información del estado/topología y las guías/políticas de la red. En el ejemplo consideramos dos redes P2P donde cada una se expande en un ISP diferente llamados A y B. En cada ISP opera un iTracker. Los pares a y b consultan a su iTracker local a través de las interfaces *info* y *policy*. Luego se registran con el appTracker y le envían la información obtenida del iTracker. El appTracker construye la selección de pares considerando los requerimientos de la aplicación de los mismos y la información que éstos le brindan.
En la Figura 2.4, vemos un ejemplo de cómo puede ser utilizada la interfaz *capability*. El appTracker solicita al iTracker B que consulte al ISP si dispone de un servidor de alta capacidad para ayudar en la distribución de contenidos. En tal caso, el iTracker coloca el servidor en la red y le habilita la dirección al appTracker para que éste lo utilice como un par más, logrando así acelerar la distribución de contenidos P2P.

2.2.2 Problema Multiobjetivo

Mostraremos técnicas para identificar formas eficientes de comunicación entre los pares en un sistema P2P, que logran optimalidad tanto para el tráfico P2P como para las cargas en los enlaces de la red. Simplemente enfatizaremos el hecho de que los clientes P2P no tienen necesidad de conectarse con un conjunto de pares específicos. Dichos clientes podrían obtener la información de algún conjunto de pares activos con los cuales tienen buena conectividad y decidir con cuales comunicarse. Si bien este acercamiento es similar a las ingenierías de tráfico tradicionales, difiere en la forma en que se asume la demanda.

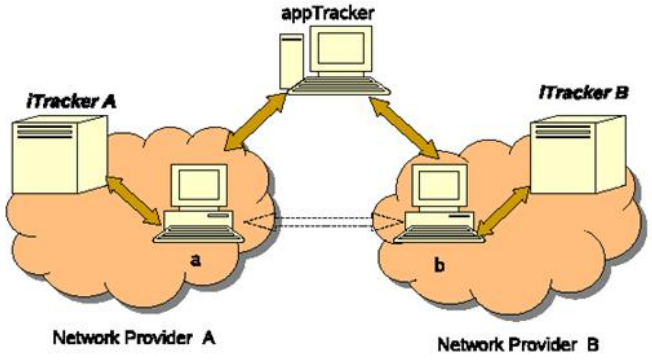


Figura 2.3: Ejemplo de redes P2P obteniendo estados/topologías y guías/políticas de la red. Figura extraída de [50]

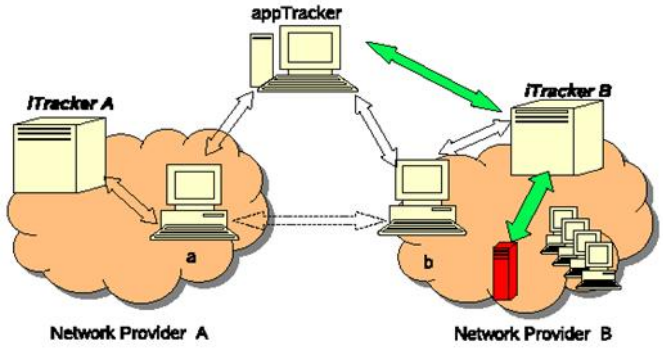


Figura 2.4: Ejemplo de redes P2P accediendo a capacidades de la red a través del iTracker. Figura extraída de [50]

Mientras las ingenierías de tráfico tradicionales asumen que cada par de $PoPs$[6] tiene asociada una demanda de tráfico fija y conocida, ésta técnica intenta cambiar la demanda del tráfico P2P para un par de $PoPs$, eligiendo eficientemente los enlaces P2P empleados. De esta forma las matrices de tráfico resultantes serán mejores desde la perspectiva de los ISPs.

[6]Los Puntos de Presencia o Points of Presence son puntos de acceso a Internet, por lo general los ISPs tienen muchos de estos distribuidos geográficamente en distintos puntos estratégicos.

Comunicación local

Esta simple técnica utiliza el conocimiento que tienen los ISPs de la topología de su red. Cuando un par se comunica con el iTracker, lo provee de las siguientes piezas de información: dirección IP, direcciones IP de otros pares en la red P2P y algún límite sobre las capacidades de los pares impuesto por los usuarios. Con esta información el iTracker puede detectar los pares que están comunicados entre sí en la red utilizando sus direcciones IP y luego sugerir pares cercanos a éstos para que se comuniquen. Esta técnica logra no sólo que las cargas en los enlaces costosos sean minimizadas, sino también que el tráfico entre pares sea más eficiente.

La desventaja de sólo generar tráfico local es que puede producir una disminución en la robustez en sistemas P2P con rotación (entrada y salida de pares en el sistema). Para solucionar esto, el iTracker podría elegir al azar un número pequeño de pares exteriores e integrarlos a los seleccionados localmente.

Modelo para tráfico no local

La técnica presentada anteriormente es aplicable para redes P2P con un número grande de usuarios (que puedan abastecerse principalmente de forma local). Esto motiva a desarrollar otra técnica para las redes P2P de menor porte pero que cuentan con una fracción importante del tráfico de la red. En estos casos, pasar por enlaces costosos del ISP es inevitable y esto necesita manejarse con cuidado. Consideraremos el siguiente escenario:
Hay K enjambres[7] en un ISP. Cada par del enjambre obtiene un único ID brindado por el appTracker y éste lo reporta al iTracker. El iTracker sigue el rastro de los pares en un enjambre dado, almacenando el número de pares en un mismo PoP y las capacidades de ancho de banda de subida y de bajada de cada par.
El iTracker mapea los $PoPs$ y les asigna $PIDs$, de esta forma construye un topología abstracta $G = (V, E)$, cuyos nodos son los $PIDs$ y las aristas son los enlaces entre los $PIDs$. Notemos que los $PIDs$ son utilizados para mantener la privacidad del ISP, y que tanto un PoP puede ser asignado a muchos $PIDs$ como muchos $PoPs$ pueden ser asignados a un solo PID. De ahora en adelante trabajaremos con el concepto de PID, suponiendo que los ISPs han definido el mapeo entre pares y $PIDs$ más conveniente.
Considerando que los mayores costos para los ISPs se producen por el uso de enlaces internacionales entonces se podría asignar un único PID a toda la red de un mismo ISP. De esta forma los enlaces inter $PIDs$ serán todos enlaces internacio-

[7] Un enjambre es un conjunto de pares que están comunicados para obtener un contenido específico

nales, siendo en estos donde los ISPs deben de aplicar mejores decisiones de ruteo de forma de lograr una utilización más eficiente de los mismos.

Los iTrackers recolectan de los ISPs la siguiente información del estatus de la red:

- b_e, la cantidad de tráfico de fondo sobre el enlace $e \in E$. Este es un tráfico perteneciente a otras aplicaciones que no provienen de redes P2P. Este tráfico no se puede redirigir.
- c_e, la capacidad del enlace $e \in E$.
- $I_e(i,j)$, es una variable que indica si el enlace $e \in E$ está en la ruta desde i a j en G, siendo i y j $PIDs$.
- Las capacidades de ancho de banda de subida y bajada de los pares dentro de los $PIDs$.

Considerando el $k-$ ésimo enjambre, podemos calcular las capacidades de subida y de bajada agregadas, $\tilde{u}_i^k$ (upload) y $\tilde{d}_i^k$ (download) sumando las capacidades de ancho de banda de los pares dentro del PID_i.
Luego, el iTracker toma en cuenta la posibilidad de comunicación local para calcular la demanda o suministro de la red para cada PID en particular. Se definen $u_i^k = \max\{\tilde{u}_i^k - \tilde{d}_i^k, 0\}$ y $d_i^k = \max\{\tilde{d}_i^k - \tilde{u}_i^k, 0\}$ como las capacidades remanentes de subida y de bajada respectivamente. Notemos que alguna de las dos es igual a cero. De esta forma las necesidades de subida o de bajada para cada PID no será totalmente contemplada internamente, por lo cual será necesario generar comunicación entre pares pertenecientes a distintos $PIDs$, generando así tráfico inter $PIDs$ y por lo tanto utilización de enlaces costosos [50].

Definición 2.2.1 *Sea la variable $t_{ij}^k \forall i,j \in$PIDs, $k = 1,\ldots,K$, la magnitud del flujo enviado desde el PID_i al PID_j asociado al $k-$ ésimo contenido la variable de decisión para atender los objetivos de los $PIDs$ e ISPs.*

El hecho de que necesariamente tenga que existir tráfico entre los distintos $PIDs$ para alcanzar sus demandas de flujo, genera una tensión entre el objetivo de las redes P2P que quieren maximizar el tráfico y el objetivo de los $ISPs$ de reducir la congestión en los enlaces costosos. Consideremos el caso general con K redes P2P, en donde el objetivo del ISP es minimizar la utilización de su MLU (utilización del enlace más exigido) eligiendo un conjunto $\{t_{ij}^k\}$ que maximice el objetivo P2P para los K enjambres [50]. Observando los objetivos de las redes P2P

y de los ISP por separado se definen los problemas de optimización $Obj_k(P2P)$ y $Obj(ISP)$ respectivamente:

Objetivo P2P para el contenido $k-$ ésimo

$$Obj_k(P2P) \begin{cases} \max \sum_i \sum_{j \neq i} t_{ij}^k & (2.1) \\ s.a. & \\ \quad \sum_{j \neq i} t_{ij}^k \leq u_i^k \quad \forall PID_i & (2.2) \\ \quad \sum_{j \neq i} t_{ji}^k \leq d_i^k \quad \forall PID_i & (2.3) \\ \quad t_{ij}^k \geq 0 \quad \forall i \neq j & \end{cases}$$

Éste problema busca maximizar el tráfico de los pares, sujeto a las restricciones de ancho de banda 2.2 y 2.3.

Objetivo ISP

$$Obj(ISP) \begin{cases} \min \max_{e \in E} \{b_e + \sum_k \sum_i \sum_{j \neq i} \frac{t_{ij}^k \times I_e(i,j)}{c_e}\} & (2.4) \\ s.a. & \\ \quad t_{ij}^k \geq 0 \quad \forall i \neq j & \end{cases}$$

Por otro lado en 2.4 se plantea el objetivo del ISP, el cuál pretende minimizar la utilización del enlace más congestionado.
Ambos objetivos se pueden expresar como un problema de optimización bi nivel que se formaliza en 2.5 y que se encuentra [50]. Esta formulación traducida al lenguaje natural significa que se quiere minimizar la utilización del enlace mas congestionado de la red, maximizando la transferencia de tráfico de los PIDs.

$$Prb(MO) \begin{cases} \min \max_{e \in E} \{b_e + \sum_k \sum_i \sum_{j \neq i} \frac{t_{ij}^k \times I_e(i,j)}{c_e}\} & \\ s.a. & \\ \quad \max \sum_i \sum_{j \neq i} t_{ij}^k, \quad \forall k \in K & \\ \quad s.a. & (2.5) \\ \quad \sum_{j \neq i} t_{ij}^k \leq u_i^k \quad \forall PID_i & \\ \quad \sum_{j \neq i} t_{ji}^k \leq d_i^k \quad \forall PID_i & \\ \quad t_{ij}^k \geq 0 \quad \forall i \neq j & \end{cases}$$

Debido a la dificultad de tener información completa de la red, en el siguiente Capítulo se introducirán 4 problemas diferentes de los cuales tres se plantean para el caso particular del ruteo de sólo un contenido ($K = 1$) variando la calidad de la información de la red. Se introduce además el Problema General para múltiples contenidos asumiendo información completa de la red.

2.3 Trabajos relacionados

2.3.1 Investigaciones previas de P4P

Existen diversas investigaciones realizadas sobre P4P [43, 50], en particular mencionaremos un trabajo realizado por un equipo de la Universidad de Yale [50] el cual fue nuestro punto de partida para definir los modelos matemáticos, y luego presentaremos algunas generalidades del trabajo presentado por la Ing. Claudia Rostagnol de la Facultad de Ingeniería UDELAR [43].

El trabajo realizado por el grupo de la Universidad de Yale plantea un modelo matemático como el introducido en 2.5. Ellos presentan dos formas para resolver el problema:

1. Como problema multiobjetivo, el cual lo resuelven en dos etapas:
   (a) Resolver el objetivo de los $PIDs$ para cada uno de los contenidos, obteniendo así un tráfico total óptimo T^k_{opt} para cada uno de los contenidos.
   (b) Luego, resolver el objetivo de los ISPs respetando el total de tráfico obtenido en la etapa anterior.
2. Reformulando el problema de tal forma que sea mono-objetivo, utilizando la función Lagrangiana.

Por más detalles sobre esta implementación referirse a [50].

El otro trabajo al que haremos referencia fue realizado por la Ing. Claudia Rostagnol [43], en éste se resolvió el mismo problema antes planteado por medio de Algoritmos Geneticos (AG) [26, 44].
Esquemáticamente la aplicación del AG fue la siguiente:

- En primer lugar se calibran los siguientes parámetros p_c (probabilidad de cruzamiento, ej. 0.8), p_m (probabilidad de mutación, ej. 0.01) y por último el número de iteraciones.

- En cada iteración se decide según p_c si se realizará un cruzamiento, de ser así, se toman dos soluciones ya existentes p_1 y p_2 y se cruzan para generar dos soluciones hijas h_1 y h_2 de la siguiente forma:

$$h_1[i,j,k] = \alpha \times p_1[i,j,k] + (1-\alpha) \times p_2[i,j,k] \, \forall i,j,k$$

$$h_2[i,j,k] = (1-\alpha) \times p_1[i,j,k] + \alpha \times p_2[i,j,k] \, \forall i,j,k$$

  Donde el parámetro $\alpha \in [0,1]$ es utilizado para realizar la combinación lineal convexa entre las dos soluciones padres.

- También en cada iteración se decide si se realizará una mutación según la probabilidad p_m, en el caso de que se realice la mutación, entonces se procede a elegir un contenido al azar k y para cada elemento de ese contenido se realiza:

$$h_1[i,j,k] = p_1[i,j,k] + \varepsilon$$

$$h_1[i,j,k] = p_1[i,j,t]; \, \forall t \neq k$$

  Donde $\varepsilon \sim \mathcal{N}(\mu, \sigma^2)$, es una variable aleatoria Normal con media μ y varianza σ^2.

- Se itera muchas veces siguiendo el esquema planteado anteriormente y se estima el *Frente de Pareto* [47] encontrando el conjunto de soluciones que no son dominadas.

2.3.2 Resultados previos de implementaciones P4P

El grupo de investigación P4P (*P4P Working Group*) se ha encargado de realizar dos test para obtener resultados de la metodología P4P utilizando información brindada por las compañias de telecomunicaciones Telefónica y Verizon, las cuales brindaron información sobre las topologías de sus redes. En ambas pruebas se determinó un contenido de 20MB para ser descargado y se crearon dos enjambres (swarms) con características similares en cuanto a su tamaño y distribución, llamando a uno "P2P Nativo" y al otro enjambre "P4P", así de esta forma poder medir las diferencias de aplicar P4P y no aplicarlo.
Cabe destacar que en ambos test se obtuvieron resultados muy alentadores. Se logró:

- Reducir significativamente el tráfico por enlaces que atraviesan el ISP (backbone trafic).

- Reducir significativamente la utilización de enlaces internacionales.

- Una gran mejoría en los tiempos de descarga de los contenidos, lo cual es muy bueno para el cliente final.

Por más información sobre los resultados referirse a [35].

2.3.3 Enfoque Ono

La idea general de Ono, al igual que P4P, es sesgar la comunicación entre pares de redes P2P incentivando a que se comuniquen aquellos que se encuentren geográficamente cerca. La diferencia entre Ono y P4P es que P4P utiliza a los ISPs para identificar distancias entre pares mientras que Ono lo hace comparando las respuestas de servidores CDN[8] (*Content Distribution Network*) los cuales brindan información sobre a qué servidores fueron derivados los pares para conseguir el contenido deseado. De esta forma se crea un mapa de ratio, el cual indica con qué frecuencia el par fue derivado a un servidor específico dentro de una ventana de tiempo. Por ejemplo si un par i el 75% de las veces fue redirigido hacia el servidor r_1 y 25% hacia el servidor r_2 en una cierta ventana de tiempo, entonces su mapa de ratio es:

$$m_i = \langle r_1 \Rightarrow 0.75, r_2 \Rightarrow 0,25 \rangle$$

De esta forma, construyendo el mapa de ratio para cada par se puede sesgar la comunicación entre ellos suponiendo que dos pares que tengan un mapa de ratio similar se encontrarán geográficamente cerca, mientras que pares que tengan mapas de ratios muy distintos se esperaría que estén muy lejos y que tengan que recorrer largos caminos para conectarse. Con esta metodología el appTracker podría construir la lista de pares (swarm) para que estos se comuniquen de forma más eficiente, sin la necesidad de un iTracker, considerando exclusivamente las distancias entre mapas de ratio reportada por cada par. El sistema mejora su precisión al considerar varias CDNs en lugar de una sola.
La ventaja de Ono frente a P4P es que no es necesario generar la comunicación explícita entre las aplicaciones P2P e ISPs, la cual es vital para la construcción de listas de pares que produzcan comunicaciones más eficientes y menos costosas. Por más detalles sobre Ono se sugiere leer [10].

[8] Son infraestructuras donde un conjunto de servidores ubicados en puntos estratégicos por todo Internet cooperan para entregar contenidos a los usuarios finales

Capítulo 3

Problemas a abordar

3.1 Introducción

En este capítulo presentaremos el Problema General para múltiples contenidos introducido en [50] y tres problemas derivados de éste, los cuales asumen el tráfico de un solo contenido ($K = 1$). Estos tres problemas varían tanto en las hipótesis de disponibilidad de información de los parámetros del modelo como en la atención a un PID destacado o todos por igual. Tener información precisa y completa sobre el tráfico de fondo y las capacidades de los enlaces es muy difícil, si no imposible. Se introduce además dos variantes del denominado Problema 1, completando un total de seis definiciones de problemas P4P. Todos los problemas que aquí se definen serán desarrollados y resueltos en los capítulos 4, 5 y 6.

Observación: 3.1.1 *Es importante destacar que todos los problemas aquí presentados asumen que a cada país se le asigna un único PID, esto simplifica la topología de la red y produce que todos los enlaces $e \in E$ serán enlaces internacionales, aunque es posible que con esta asignación existan enlaces internacionales que no estén considerados.*

3.2 Problema General (PG)

Supondremos que existen K contenidos a transferir, con $K > 0$, y que estamos frente a una red completa, de la cual tenemos conocimiento perfecto de los anchos de banda de subida y bajada para cada contenido $k \in [1, \ldots, K]$ de todos los $PIDs$. Se conocen además las matrices de capacidades y de tráfico de fondo para todos los enlaces de la red. Llamaremos PG al Problema General, el cuál se expresa matemáticamente en 3.1.

$$PG \quad \begin{cases} \min \max_{e \in E} \{b_e + \sum_k \sum_i \sum_{j \neq i} \frac{t_{ij}^k \times I_e(i,j)}{c_e}\} \\ s.a. \\ \quad \max \sum_i \sum_{j \neq i} t_{ij}^k, \quad \forall k = 1, 2, \ldots, K \\ \quad s.a. \\ \quad\quad \sum_{j \neq i} t_{ij}^k \leq u_i^k \quad \forall PID_i \\ \quad\quad \sum_{j \neq i} t_{ji}^k \leq d_i^k \quad \forall PID_i \\ \quad\quad t_{ij}^k \geq 0 \quad \forall i \neq j \\ \quad\quad \sum_k \sum_i \sum_{j \neq i} t_{i,j}^k \times I_e(i,j) \leq v_e \end{cases} \tag{3.1}$$

donde b_e y c_e representan el tráfico de fondo y capacidad del enlace e, $t_{i,j}^k$ es la magnitud de flujo que es transmitida por el nodo i y recibida por j del contenido k, $v_e = c_e(1 - b_e)$ es la capacidad que se destina exclusivamente a aplicaciones P2P (frecuentemente denominada "capacidad virtual") e $I_e(i,j)$ es la variable binaria indicatriz, que vale 1 si el tráfico t_{ij} contiene al enlace e, y 0 en caso contrario.

Como es muy difícil conseguir información completa de la red y en particular para cada uno de los contenidos a atender, se definen a continuación tres problemas que se enfocan en la distribución de sólo un contenido, $K = 1$ y asumen distintos conocimientos de capacidades y tráficos de fondo de los enlaces de la red. Se definen de tal forma que cada uno de los problemas está contenido en el anterior.

3.3 Un contenido, conocimiento completo (Problema 1 o P1)

Supondremos que existe sólo un contenido a transferir y que estamos frente a una red completa, de la cual tenemos conocimiento perfecto de los anchos de banda de subida y bajada de todos los $PIDs$. Se conocen además las matrices de capacidades y de tráfico de fondo para todos los enlaces de la red. El Problema (P1) está contenido en (PG), sólo basta considerar $K = 1$ en 3.1. De esta forma definimos matemáticamente en 3.2 el Problema 1 de optimización (P1), como:

$$P1 \quad \begin{cases} \min \max_{e \in E} \{b_e + \sum_i \sum_{j \neq i} \frac{t_{ij} \times I_e(i,j)}{c_e}\} \\ \quad s.a. \\ \quad\quad max \sum_i \sum_{j \neq i} t_{ij} \\ \quad\quad s.a. \\ \quad\quad\quad \sum_{j \neq i} t_{ij} \leq u_i \quad \forall i \in \{1, \ldots, N\} \\ \quad\quad\quad \sum_{j \neq i} t_{ji} \leq d_i \quad \forall i \in \{1, \ldots, N\} \\ \quad\quad\quad t_{ij} \geq 0 \quad \forall i \neq j \in \{1, \ldots, N\} \\ \quad\quad\quad \sum_i \sum_{j \neq i} t_{ij} \times I_e(i,j) \leq v_e \end{cases} \tag{3.2}$$

Decidimos también estudiar dos variantes del P1, en las cuales se asume conocimiento completo de la red pero se suponen distintos escenarios, los cuales pasamos a definir a continuación.

3.3.1 Variante P1: utilización fija de los enlaces ($VP1_\rho$)

En esta variante de P1, que llamaremos $VP1_\rho$, asumiremos conocimiento completo de la red y que la utilización máxima de enlaces internacionales no puede superar un umbral dado, ρ_{max}. Se sustenta en la idea que los ISPs podrían adoptar como política que los enlaces internacionales tengan una utilización máxima, por ejemplo de 90%, de esta forma definimos matemáticamente en 3.5 el Problema $VP1_\rho$:

$$VP1_\rho \begin{cases} \max \sum_i \sum_{j \neq i} t_{ij} \\ s.a. \\ \quad \sum_{j \neq i} t_{ij} \leq u_i \quad \forall PID_i \\ \quad \sum_{j \neq i} t_{ji} \leq d_i \quad \forall PID_i \\ \quad t_{ij} \geq 0 \quad \forall i \neq j \\ \quad \sum_i \sum_{j \neq i} t_{ij} \times I_e(i,j) \leq c_e(1 - b_e) \qquad (3.3) \\ \quad b_e + \sum_i \sum_{j \neq i} \frac{t_{ij} \times I_e(i,j)}{c_e} \leq \rho_{max} \qquad (3.4) \end{cases}$$

Proposición 3.3.1 *La restricción* (3.3) *es redundante, pues* (3.4) *es más restrictiva.*

Prueba. Si denotamos $T_e = \sum_i \sum_{j \neq i} t_{ij} \times I_e(i,j)$ podemos escribir (3.4) como $b_e + T_e/c_e \leq \rho_{max}$ entonces $T_e/c_e \leq \rho_{max} - b_e < 1 - b_e$, cumpliendo en particular con la restricción (3.3).

QED

Esto motiva a definir $VP1_\rho$ de la siguiente manera:

$$VP1_\rho \begin{cases} \max \sum_i \sum_{j \neq i} t_{ij} \\ s.a. \\ \quad \sum_{j \neq i} t_{ij} \leq u_i \quad \forall PID_i \\ \quad \sum_{j \neq i} t_{ji} \leq d_i \quad \forall PID_i \\ \quad t_{ij} \geq 0 \quad \forall i \neq j \\ \quad b_e + \sum_i \sum_{j \neq i} \frac{t_{ij} \times I_e(i,j)}{c_e} \leq \rho_{max} \end{cases} \tag{3.5}$$

Observemos que el Problema $VP1_\rho$ ya no es multiobjetivo, esto se debe a que fijamos la utilización máxima igual a ρ_{max}. En el Capítulo 5 se profundiza sobre este problema.

3.3.2 Variante P1: Ruteos por caminos de largo uno ($VP1$)

Para esta nueva variante del P1, que llamaremos $VP1$, también se asume conocimiento completo de la red pero se obliga a que el tráfico de flujo entre $PIDs$ se realice sólo por caminos de largo 1. Esta variante se justifica por el hecho de que al aumentar el largo de los caminos entre pares de $PIDs$ se incrementa al flujo por los enlaces de la red. En 3.6 se formaliza matemáticamente el Problema $VP1$:

$$VP1 \quad \begin{cases} \min \max_{e \in E} \{b_e + \sum_i \sum_{j \neq i} \frac{t_{ij} \times I_e(i,j)}{c_e}\} \\ \quad s.a. \\ \quad\quad max \sum_i \sum_{j \neq i} t_{ij} \\ \quad\quad s.a. \\ \quad\quad\quad \sum_{j \neq i} t_{ij} \leq u_i \quad \forall i \in \{1, \ldots, N\} \\ \quad\quad\quad \sum_{j \neq i} t_{ji} \leq d_i \quad \forall i \in \{1, \ldots, N\} \\ \quad\quad\quad t_{ij} \geq 0 \quad \forall i \neq j \in \{1, \ldots, N\} \\ \quad\quad\quad \sum_i \sum_{j \neq i} t_{ij} \times I_e(i,j) \leq v_e \\ \quad\quad\quad |Camino_{(i,j)}| = 1 \, \forall i \neq j \in \{1, \ldots, N\} / t_{(i,j)} > 0 \end{cases} \tag{3.6}$$

Como vemos en 3.6 el Problema $VP1$ es idéntico al P1 con la diferencia que al $VP1$ es más restrictivo, ya que se le agrega una restricción adicional que limita

el largo de los caminos. En el Capítulo 5 también profundizaremos sobre este problema.

A continuación definimos dos problemas que reflejan la realidad del difícil acceso a la información en cuanto al conocimiento tanto de los tráficos de fondo y de las capacidades de los enlaces.

3.4 Un contenido, tráfico de fondo parcialmente conocido (Problema 2 o P2)

En este problema supondremos que sólo podemos medir el tráfico de fondo asociado a un PID especial. En nuestro caso ese PID será Uruguay, es por esto que computamos con valores positivos sólo los tráficos de fondo asociados a los enlaces que tengan como PID origen o destino a Uruguay.
Pasar del P1 al P2 es trivial: sólo debemos definir valores de tráfico de fondo que cumplan con (3.7) y (3.8) en el P1, donde el nodo u representa a Uruguay:

$$b_{u,i} > 0,\, b_{i,u} > 0, \quad \forall i \neq u, \tag{3.7}$$

$$b_{i,j} = 0 \quad \forall i, j \neq u. \tag{3.8}$$

En términos matriciales, todas las filas y columnas tienen entrada nula en la matriz de tráfico de fondo, salvo la fila y columna correspondiente al nodo Uruguay (denotado con u). Esto deja en evidencia que el P2 es un caso particular del P1, pues este último considera una matriz de tráficos de fondo arbitraria. El problema (P2), quizás sea el menos realista ya que por lo general si se tiene conocimiento de las capacidades de los enlaces es muy probable que también se conozcan los respectivos tráficos de fondo. Esto motiva a definir el último problema, el cuál en alguna medida es el más realista en términos de información disponible.

Observación: 3.4.1 *El problema P2 requiere un tratamiento diferenciado con el nodo "especial" (Uruguay) al momento de su resolución. De hecho, las metaheurísticas aplicables al Problema 1 (algoritmos no exactos) pueden tener desempeño no satisfactorio para el Problema 2, y viceversa. Esto se fundamenta con más detalle en el Capítulo 4.*

3.5 Un contenido, tráfico de fondo y capacidades parcialmente conocidas (Problema 3 o P3)

Por un problema de acceso a la información, no es posible tener conocimiento tanto de las capacidades como de los tráficos de fondo de los enlaces, salvo del PID Uruguay.
Entonces, en el P3 se asume que los tráficos de fondo y las capacidades de las aristas que tienen como origen o destino al PID Uruguay tendrán valores positivos. Las capacidades y los tráficos de fondo de los otros enlaces los consideraremos como infinito y cero respectivamente:

$$
\begin{aligned}
&c_{u,i} > 0,\, c_{i,u} > 0, \quad \forall i \neq u,\\
&c_{i,j} = +\infty \quad \forall i, j \neq u,\\
&b_{u,i} > 0,\, b_{i,u} > 0 \quad \forall i \neq u,\\
&b_{i,j} = 0 \quad \forall i, j \neq u.
\end{aligned}
$$

El problema multi-objetivo presenta la misma forma que el P1 mostrado en (3.2). Formalizaremos en el capítulo siguiente que el compromiso entre ambos objetivos (pares e ISPs) es posible resolverlo en tiempo polinomial para el P3. Más específicamente, existe un algoritmo polinomial que permite incrementar linealmente la utilización del enlace más exigido, a cambio de un incremento lineal en la transferencia neta de los pares.

Capítulo 4

Propiedades y algoritmos para los problemas de un contenido

En el Capítulo 3 hemos introducido cuatro problemas, la única deferencia entre ellos son los parámetros de entrada, y en el caso PG la cnatidad de contenidos a transferir. También se definieron dos variantes para el Problema 1, de las cuales una asume utilización fija de los enlaces simplificando el problema y la otra restringe el ruteo de tráfico sólo por caminos de largo uno.
En este Capítulo definiremos el concepto de "Instancia" para los problemas y probaremos que el Problema 3 se puede resolver en tiempo polinomial. Por otra parte, desarrollaremos una heurística sustentada en la técnica para ruteo equitativo de flujo llamada "*waterfilling*" para resolver el Problema 2. Luego introducimos una primera aproximación para resolver el Problema 1 utilizando un algoritmo goloso (Greedy) básico, del cual veremos su desempeño frente al Problema 1 y analizamos algunas de sus desventajas, las cuales pondrán de manifiesto variantes para el Algoritmo Goloso que intentan superar estas falencias. También definiremos instancias optimistas y pesimistas para el Problema 1. Si bien en este Capítulo se introduce el concepto de algoritmos metaheurísticos, profundizaremos sobre estos en el Capítulo 6.

4.1 Instancias y propiedades

Definición 4.1.1 *Una instancia para cualquiera de los problemas definidos en el Capítulo 3 consiste en las siguientes 4 matrices de entrada:*

- *Una matriz $B \in M_{n,n}(\mathbb{R})$ que contiene en cada entrada (i,j) el tráfico de fondo $b_{(i,j)}$ correspondiente al enlace $e = (i,j)$.*

- *Una matriz $C \in M_{n,n}(\mathbb{R})$ que contiene las capacidades (i,j) $c_{(i,j)}$ correspondiente al enlace $e = (i,j)$.*
- *Un vector $U \in M_{n,1}(\mathbb{R})$ que contiene los anchos de banda de subida u_i en cada entrada. Para el Problema General $U \in M_{n,K}(\mathbb{R})$ ya que cada contenido tiene asociado un vector de capacidades de subida.*
- *Un vector $D \in M_{n,1}(\mathbb{R})$ que contiene los anchos de banda de bajada d_i en cada entrada. Para el Problema General $D \in M_{n,K}(\mathbb{R})$ ya que cada contenido tiene asociado un vector de capacidades de bajada.*

Denotemos a las instancias de entrada mediante $I = (B, C, U, D)$.

Definición 4.1.2 *El potencial de subida P_i^u para el PID_i es igual a*

$$P_i^u = \sum_{j \neq i} \min\{\frac{d_j}{n-1}, c_{i,j}(1 - b_{i,j})\} \tag{4.1}$$

Esto representa todo lo que un PID puede subir a la red en función de las capacidades de descarga de sus vecinos y de la capacidad de sus enlaces con ellos. Se observa que aparece una fracción del potencial de descarga en vista de que cada vecino va a recibir tráfico no solamente del nodo i-ésimo, sino de los $n-2$ nodos restantes.

Definición 4.1.3 *Dados $U = \{u_1, u_2, \ldots, u_n\}$, $D = \{d_1, d_2, \ldots, d_n\}$, los anchos de banda de subida y bajada para los n $PIDs$, la matriz de Capacidades de los enlaces C y la matriz B que indica el porcentaje de flujo que corresponde al tráfico de fondo para los enlaces. Decimos que una instancia $I(U, D, B, C)$ es* optimista *para el Problema 1 (P1) si el potencial de subida es mayor o igual a la capacidad de subida para todos los nodos.*
Matemáticamente:

$$u_i \leq \sum_{j \neq i} \min\{\frac{d_j}{n-1}, c_{i,j}(1 - b_{i,j})\}, \quad \forall PID_i \tag{4.2}$$

Lema 4.1.4 *Si el tráfico total $T = \sum_i \sum_{j \neq i} t_{i,j}$ es igual al mínimo entre la suma de las capacidades de subida y de bajada, entonces T es óptimo para los $PIDs$.*

Prueba. Si sumamos para todos los $PIDs$ las restricciones de ancho de banda de subida y bajada del $(P1)$ tenemos que:

$$\sum_i \sum_{j \neq i} t_{i,j} \leq \sum_i u_i \tag{4.3}$$

$$\sum_i \sum_{j \neq i} t_{j,i} \leq \sum_i d_i \tag{4.4}$$

Lo cual demuestra el Lema ya que como $T = \sum_i \sum_{j \neq i} t_{j,i}$, se deduce que $T \leq \min\{\sum_i u_i, \sum_i d_i\}$.

QED

Lema 4.1.5 *En una instancia $I(U, D, B, C)$ optimista del problema $(P1)$, existe una solución factible en la que el tráfico enviado por cada PID es igual a su capacidad de subida.*

Prueba. Se decide enviar por cada enlace un tráfico:

$$t_{i,j} = u_i \frac{\min\{\frac{d_j}{n-1}, c_{i,j}(1 - b_{i,j})\}}{\sum_{h \neq i} \min\{\frac{d_h}{n-1}, c_{i,h}(1 - b_{i,h})\}}$$

Es apreciable que el tráfico que envía cada PID resulta positivo. Además, no supera la capacidad del enlace pues por estar en condiciones optimistas:

$$t_{i,j} = u_i \frac{\min\{\frac{d_j}{n-1}, c_{i,j}(1 - b_{i,j})\}}{\sum_{h \neq i} \min\{\frac{d_h}{n-1}, c_{i,h}(1 - b_{i,h})\}} \leq$$

$$\leq \min\{\frac{d_j}{n-1}, c_{i,j}(1 - b_{i,j})\} \leq c_{i,j}(1 - b_{i,j})$$

que es el remanente del enlace.
Finalmente se observa que cada PID i envía:

$$\sum_{j \neq i} t_{i,j} = \sum_{j \neq i} u_i \frac{\min\{\frac{d_j}{n-1}, c_{i,j}(1 - b_{i,j})\}}{\sum_{h \neq i} \min\{\frac{d_h}{n-1}, c_{i,h}(1 - b_{i,h})\}} =$$

$$= u_i \frac{\sum_{j \neq i} \min\{\frac{d_j}{n-1}, c_{i,j}(1 - b_{i,j})\}}{\sum_{h \neq i} \min\{\frac{d_h}{n-1}, c_{i,h}(1 - b_{i,h})\}} = u_i$$

que es su capacidad de subida, lo cual hace que la solución sea factible.

QED

Definición 4.1.6 *Decimos que una instancia $I(U, D, B, C)$ es* pesimista *para el Problema Principal $(P1)$ si $\exists i \in \{1, 2, \ldots, n\}$, que cumpla alguna de las siguientes condiciones:*

$$\begin{cases} \sum_{j \neq i} c_{i,j}(1 - b_{i,j}) < u_i, \textit{ si } u_i \leq d_i \quad i \in PID \\ \qquad\qquad \textit{ó} \\ \sum_{j \neq i} c_{j,i}(1 - b_{j,i}) < d_i, \textit{ si } d_i \leq u_i \quad i \in PID \end{cases} \tag{4.5}$$

Observación: 4.1.7 *Cuando una instancia $I(U, D, B, C)$ es pesimista no existe forma que se alcance el objetivo de los $PIDs$ sin saturar los enlaces de los $PIDs$ que cumplen con las ecuaciones 4.5.*

4.2 El Problema 3 es polinomial

Dado que el Problema 1 es multiobjetivo, debemos definir qué se entiende por su resolución en tiempo polinomial. A partir de la definición del objetivo de los ISPs, la utilización máxima de los enlaces, ρ, es no menor que el mayor de los tráficos de fondo: $\rho \geq \max\{b_{i,j}, i, j \in \mathcal{N}\} = b_{max}$, pues sin cargar la red de tráfico se obtiene una utilización del enlace más exigido dada por b_{max}. Decimos que la resolución del Problema 3 es posible en tiempo polinomial cuando dada una utilización arbitraria $\rho_{max} \in [0, 1]$, existe un algoritmo que corre en tiempo polinomial en las entradas (anchos de banda, capacidades y matriz de tráficos de fondo) y retorna una solución factible que garantiza la transferencia máxima entre los $PIDs$, y exige a cada uno de sus enlaces que la utilización no supere el umbral dado ρ_{max}. Puesto que no es posible trabajar a menor utilización que el tráfico de fondo, podemos restringir el valor de utilización arbitrario ρ_{max} al intervalo $[b_{max}, 1]$.

Observación: 4.2.1 *Las instancias son válidas para el Problema 3 cuando la matriz C tiene valor infinito en todas sus entradas, exceptuando la fila y columna asociada al nodo destacado (Uruguay), y ceros en su diagonal. También debe cumplir que la matriz B debe ser nula, salvo por la fila y la columna asociada al nodo destacado.*

Definición 4.2.2 *Sea I una instancia arbitraria de entrada válida para el Problema 3, $\Omega(I)$ el conjunto de soluciones factibles para la instancia I con $S(A) = (\phi, t, I)$ una solución factible que es salida de un algoritmo A con entrada I, donde ϕ es la lista de caminos de tráfico y t sus respectivas magnitudes. Sea $T(S) = \sum_i \sum_{j \neq i} t_{i,j}$ la transferencia de una solución S y $b_{max} = \max_{j \neq i}\{b_{i,j}\}$.*

Decimos que es posible resolver el Problema 3 en tiempo polinomial, y lo denotamos con $P_3 \in P$, si se cumple 4.6:

$$\exists A : \forall I, \forall \rho_{max} \in [\theta, 1], T(S(A)) \geq T(S^*), \forall S^* \in \Omega(I) \tag{4.6}$$

A continuación pasamos a describir el Algoritmo A que resuelve el Problema 3 en tiempo polinomial como demostramos en el Teorema 4.2.3. En principio Uruguay es etiquetado con el nodo 1. Recibe como entradas una instancia válida y una utilización admisible $\rho_{max} \in [b_{max}, 1]$ y retorna una solución factible al Problema 3. Las Líneas 1 a 4 son de inicialización: se comienza con la solución vacía y agrega flujos y sus magnitudes posteriormente. El estudio del Algoritmo A se puede separar en dos grandes bloques (ver Figura 4.1): el primero comprendido por las Líneas 5 a 17, y el segundo de las Lineas 18 a 28. El primer bloque tiene el cometido de transferir el mayor tráfico posible entre nodos no locales (distintos del nodo 1). Básicamente se revisa lo que tiene para subir un nodo y chequea si hay capacidad de bajada disponible en el resto de los nodos no locales. Se transfiere todo lo que puede enviar en proporción con los anchos de bajada de los $PIDs$ no locales; de esta manera veremos que se asegura la transferencia máxima entre los nodos no locales ya que los enlaces entre estos tienen magnitud infinita. Por último, el segundo bloque (lineas 18 a 28) permite comunicar al nodo 1 (Uruguay) con el resto de los nodos. Se envía aquel tráfico que permite igualar la utilización del enlace más exigido. La idea de este bloque es agregar transferencia entre los $PIDs$ mejorando la solución parcial, dado que la utilización se mantiene fija igual a $\rho_{max} \in [b_{max}, 1]$.

Entradas:
Tráficos de fondo: $\{\{b_{1,i}\} \cup \{b_{i,1}\}, i = 2, \ldots, n\}$
Matriz de capacidades: C
Ancho de subida de PIDs: U
Ancho de bajada de PIDs: D
Utilización admisible: $\rho_{max} > \max_{i=2,\ldots,n}\{\{b_{1,i}\} \cup \{b_{i,1}\}\}$
Salidas:
Solución factible: $S = (\phi, t, I)$

1: $\phi \leftarrow \{\}$
2: $t \leftarrow \{\}$
3: $iterador \leftarrow 2$
4: $bajada \leftarrow \sum_{i=2}^{n} D(i)$
5: **Mientras** $iterador \leq n$ y $bajada > 0$
6: $\quad bajada \leftarrow bajada - D(iterador)$
7: $\quad$ **Para** $j = 2, \ldots, n, j \neq iterador$
8: $\quad\quad \phi \leftarrow \phi \cup (iterador, j)$
9: $\quad\quad t \leftarrow t \cup \{D(j) \times \min\{1, \frac{U(iterador)}{bajada}\}$
10: $\quad\quad D(j) \leftarrow D(j) - \frac{D(i)}{bajada} \times U(iterador)$
11: $\quad$ **Fin Para**
12: $\quad U(iterador) \leftarrow U(iterador) - D(j) \times \min\{1, \frac{U(iterador)}{bajada}\}$
13: $\quad bajada \leftarrow \sum_{i=2}^{n} D(i)$
14: $\quad iterador \leftarrow iterador + 1$
15: **Fin Mientras**
16: $D_{parcial} \leftarrow bajada$
17: $U_{parcial} \leftarrow \sum_{i=2}^{n} U(i)$
18: // Comunicación con Uruguay
19: **Para** $nodo = 2$ a n
20: $\quad \phi \leftarrow \phi \cup (1, nodo)$
21: $\quad t \leftarrow t \cup \{\min\{U(1), D_{parcial}, (\rho_{max} - b_{1,nodo}) \times C_{1,nodo}\}\}$ // Se agrega flujo sin exceder uso
22: $\quad D_{parcial} \leftarrow D_{parcial} - \{\min\{U(1), D_{parcial}, (\rho_{max} - b_{1,nodo}) \times C_{1,nodo}\}\}$
23: $\quad U(1) \leftarrow U(1) - \min\{U(1), D_{parcial}, (\rho_{max} - b_{1,nodo}) \times C_{1,nodo}\}$
24: $\quad \phi \leftarrow \phi \cup (nodo, 1)$
25: $\quad t \leftarrow t \cup \{\min\{D(1), U_{parcial}, (\rho_{max} - b_{nodo,1}) \times C_{nodo,1}\}\}$
26: $\quad U_{parcial} \leftarrow U_{parcial} - \min\{D(1), U_{parcial}, (\rho_{max} - b_{nodo,1}) \times C_{nodo,1}\}$
27: $\quad D(1) \leftarrow D(1) - \min\{D(1), U_{parcial}, (\rho_{max} - b_{nodo,1}) \times C_{nodo,1}\}$
28: **Fin Para**
29: **Retornar** $S = (\phi, t)$

Figura 4.1: **Algoritmo** $\mathcal{A}$: logra la máxima transferencia neta, a una utilización máxima dada

A continuación, se introduce un Teorema que demuestra que el Problema 3 es resoluble en tiempo polinomial según la Definición 4.2.2.

Teorema 4.2.3 $P_3 \in P$

Prueba. Sea $I = (B, C, U, D)$ una instancia válida para el Problema 3, y ρ_{max} : $b_{max} \leq \rho_{max} \leq 1$. Probaremos que la transferencia que se obtiene mediante la aplicación del Algoritmo A es la mayor dentro de todas las soluciones factibles sujeta a la restricción de utilización del enlace más congestionado, obteniendo el resultado por la Definición 4.2.2. Estudiaremos dos casos:

I. Si $\sum_{i=2}^{n} D(i) \geq \sum_{i=2}^{n} U(i)$, el Algoritmo A transfiere en el primer bloque (lineas 5 a 17) $t = \sum_{i=2}^{n} u_i$. En efecto, el primer bloque visita a todos los nodos no locales (pues la bajada siempre es mayor que el ancho de banda de subida de cada par no local). Además, en cada nodo visitado $i \in \{2, \ldots, n\}$ se transfiere a la red una magnitud igual a:

$$t_i = \sum_{j=2, j\neq i}^{n} \frac{D^{'}(j)}{\sum_{k=2,k\neq i}^{n} D^{'}(k)} U(i) = U(i), \tag{4.7}$$

donde $D^{'}(i)$ representa el ancho de bajada remanente hasta la iteración i que presenta el nodo no local i. Entonces, la suma de tráfico neto en el primer bloque de comunicación entre nodos no locales es de:

$$T_1 = \sum_{i=2}^{n} t_i = \sum_{i=2}^{n} U(i). \tag{4.8}$$

II. En caso contrario, si $\sum_{i=2}^{n} D(i) < \sum_{i=2}^{n} U(i)$ la comunicación entre nodos no locales va a permitir transferir $\sum_{i=2}^{n} D(i)$. De hecho, el Algoritmo A va a salir del ciclo *"Mientras"* (lineas 5 a 15) cuando el parámetro $bajada$ se anule, transfiriéndose tanto tráfico como el neto que pueden subir los nodos locales.

Juntando los dos casos posibles, se observa que en todos los casos los nodos no locales consiguen transferir entre ellos $T_1 = \min\{\sum_{i=2}^{n} D(i), \sum_{i=2}^{n} U(i)\}$, habiendo aprovechado que la comunicación es ideal (capacidades infinitas).
Obsérvese que en el segundo bloque del Algoritmo A (lineas 18 a 28) no se exige ninguno de sus enlaces más allá de ρ_{max}. Justamente, la utilización de sus enlaces

es de:

$$\begin{aligned}\rho_{1,i} &= b_{1,i} + \frac{t_{1,i}}{c_{1,i}} \\ &= b_{1,i} + \frac{\min\{U(1), D_{local}, (\rho_{max} - b_{1,i}) \times c_{1,i}\}}{c_{1,i}} \\ &\leq b_{1,i} + \frac{(\rho_{max} - b_{1,i}) \times c_{1,i}}{c_{1,i}} = \rho_{max}\end{aligned}$$

El último ciclo *"Para"* (lineas 19 a 28) del Algoritmo A es tal que agrega tráfico por estos enlaces hasta igualar utilización ρ_{max}, o en su defecto (ante ausencia de ancho de banda sea en el nodo local o exterior), se usa lo más posible. Si llamamos $U_{parcial}$ y $D_{parcial}$ a los respectivos ancho de banda netos remanentes de los nodos no locales luego del bloque 1, entonces en el bloque 2 se logra una transferencia neta igual a (acumulando efectos de todo el ciclo):

$$\begin{aligned}T_2 = \min\{U(1), D_{local}, \sum_{i=2,n} (\rho_{max} - b_{1,i}) \times c_{1,i}\} \\ + \min\{D(1), U_{local}, \sum_{i=2,n} (\rho_{max} - b_{i,1}) \times c_{i,1}\}\end{aligned}$$

Finalmente, este análisis permite hallar el tráfico neto logrado por el Algoritmo A, que es la suma de las contribuciones entre los bloques 1 y 2. Si $m = \min\{U, D\} = \min\{\sum_{i=2}^{n} U(i), \sum_{i=2}^{n} D(i)\}$ es el límite de transferencia entre nodos no locales, $C_{sal} = \sum_{i=2}^{n} (\rho_{max} - b_{1,i}) \times c_{1,i}$ es el límite de enlaces salientes al nodo 1 y análogamente $C_{ent} = \sum_{i=2}^{n} (\rho_{max} - b_{i,1}) \times c_{i,1}$ el límite de sus enlaces de entrada, entonces el tráfico neto obtenido con el Algoritmo A es en magnitud:

$$\begin{aligned}T &= T_1 + T_2 \\ &= m + \min\{U(1), U - m, C_{sal}\} + \min\{D(1), D - m, C_{ent}\} \qquad (4.9)\end{aligned}$$

Básicamente, podemos concebir el resultado del Algoritmo A mediante un supernodo que acumula los recursos de ancho de banda no local, y se autoabastece con transferencia m. A continuación, intenta aprovechar su remanente de subida (o bien bajada) mediante la comunicación con el nodo local. Se observa que el segundo o bien el tercer término de la ecuación (4.9) se anula: en el Caso I se anula el segundo término, mientras que en el Caso II se anula el tercero.

Toda solución factible no puede tener mejor desempeño que el Algoritmo A, si posee límite de utilización ρ_{max}. En efecto, el Algoritmo A explota todo el potencial de transferencia en el supernodo no local. Finalmente, aprovecha el nodo Uruguay a efectos de utilizar el remanente de descarga o subida del supernodo no local, sin

exceder la utilización ρ_{max}. Obsérvese que las redirecciones son aplicables entre nodos no locales, sin costo alguno, por lo que los recursos entre estos nodos son compartidos, y vale por tanto concebir un supernodo equivalente.

QED

El mecanismo propuesto para resolver P3 dio origen para formularnos la variante del Problema 1 presentada en el Capítulo 3 llamada ($VP1_\rho$), en la cual fijamos un parámetro de utilización máxima (ρ_{max}) para todos los enlaces. El procedimiento propuesto para resolver el problema $VP1_\rho$ se sustenta en el Algoritmo de Ford-Fulkerson y será introducido en el Capítulo 5.

4.3 Algoritmo de GOTEO para el Problema 2 (P2)

Observación: 4.3.1 *Como se ha dicho en el Capítulo 3, el Problema 2 es un caso particular del Problema 1. En efecto, basta con considerar las instancias del Problema 1 que anulen la matriz de tráficos de fondo en todos los enlaces que no inciden al nodo Uruguay.*

A continuación introducimos una técnica llamada *waterfilling* [9, 15, 25] diseñada para introducir flujo a la red utilizando de forma equitativa los recursos la misma, esta técnica es una pieza fundamental para el llamado algoritmo GOTEO desarrollado conjuntamente con el Ing. Pablo Romero el cual resuelve el P2.

4.3.1 Waterfilling

Es de uso frecuente esta estrategia de asignación de recursos en telecomunicaciones. La técnica de Waterfilling [9, 15, 25] es óptima a efectos de alojar potencia en canales gausianos independientes con ruido para maximizar la capacidad. Aquí se busca una analogía teniendo n enlaces independientes sobre los cuales es posible enviar tráfico.
Su nombre explica a la vez su significado: consideremos n baldes que poseen cantidades iniciales de agua l_i y capacidades $c_i, i = 1, \ldots, n$, ambas medidas en litros. A su vez deseamos echar L litros sobre los baldes que ya disponemos. ¿Cómo hacemos para que ningún balde se desborde? Es claro que si vamos a echar más agua que la suma de capacidades remanentes de todos los baldes, es imposible. Ahora bien: si L no supera la capacidad remanente de los enlaces, la mejor manera de llenarlos es tal como cae el agua si pudiera deslizase en ellos y descender por el efecto gravitatorio: llenando siempre el que posee menor cantidad de agua (baldes con igual sección). Esto es lo que se hace en $Waterfilling$: se dispone de

capacidades y se "llenan" los enlaces (baldes) según el tráfico por exceso de otro enlace (balde "congestionado"). De esta manera estamos equilibrando los baldes, y la función objetivo se uniformiza, bajo una normalización adecuada de baldes (posiblemente no todos posean igual capacidad).

4.3.2 Algoritmo de GOTEO

Presentaremos el Algoritmo de GOTEO para el Problema 2 que asegura retornar flujos factibles. El Algoritmo de GOTEO intenta maximizar el tráfico entre $PIDs$ y luego descongestionar el enlace más congestionado. El algoritmo funciona en dos bloques:

1. El primer bloque (lineas 2 a 22) tiene como cometido subir la mayor cantidad de flujo entre los $PIDs$ utilizando caminos de largo 1 aplicando la técnica "*waterfilling*" la cual logra una distribución homogénea del flujo teniendo en cuenta la saturación de los enlaces (poner más flujo en los enlaces menos congestionados y menos en los más congestionados).

2. El segundo bloque (lineas 23 a 25) es de descongestión, básicamente, detecta el enlace más congestionado (MLU) y redirige parte del tráfico que pasa por este utilizando caminos de largo 2.

En la Figura 4.2 se muestra el Pseudo-Código del Algoritmo de GOTEO el cual funciona de la siguiente manera: recibe como entradas; el número de $PIDs$, n, las matrices de capacidades y de tráfico de fondo de los enlaces, C y B respectivamente, número de iteraciones "$iter$", un parámetro llamado "uso" que determina la proporción que subirá a la red el PID distinguido (PID Uruguay etiquetado como PID 1) y por último otro parámetro llamado "$proporcion$" que indica qué proporción del flujo que pasa por el enlace más congestinado se debe de redirigir.
En el primer bloque del Algoritmo de GOTEO (lineas 2 a 22) se encuesta la capacidad de subida de cada par, por turnos (linea 3). Se analiza la receptividad de los nodos vecinos de este tráfico potencial de ser enviado, estudiando en cada turno si el que limita es la bajada de los $PIDs$ vecinos o bien las capacidades de los enlaces (linea 4), de esta forma podemos calcular cuánto es lo máximo que el PID emisor de flujo puede subir a la red, calculando el valor "$echar$", distinguiendo si el PID emisor es el destacado (Uruguay) o no (lineas 6 a 10). Si el PID emisor es Uruguay, entonces se lo penaliza dejando que suba una proporción ("uso" $\in [0, 1]$) de todo el flujo que este puede subir (recordemos que el PID Uruguay es el único que posee trafico de fondo no nulo, si le dejamos subir todo el flujo que pueda es probable que tape todos o alguno de sus enlaces) (linea 7). Ahora si el

PID emisor es distinto a Uruguay se deja que el flujo a subir sea todo los que el PID pueda (linea 9). Teniendo ya definido cuánto flujo va a subir el PID emisor es necesario determinar de que forma se introduce éste en los distintos enlaces, utilizando la técnica "*waterfilling*" identificamos un parámetro que llamamos "*Linea*" (linea 11) que indica por qué enlaces realizar la transferencia de flujo para lograr una distribución homogénea. En otras palabras, "*Linea*" nos indica cuánto es la capacidad remanente mínima que tiene que tener un enlace para ser utilizado para enviar flujo. Encuestando a cada PID vecino al PID emisor (linea 12) determinamos cuales son los enlaces aptos para hacer la transferencia (linea 13) y se le calcula el flujo a poner en ese enlace ("*Poner*") el cual es todo el flujo que se tiene que introducir en el enlace para alcanzar el valor "*Linea*" calculado con la Función "*waterfilling*", (linea 14). Por último en el bloque 1, se actualizan los tráficos T, las capacidades C y los anchos de banda U y D.
En el segundo bloque (lineas 23 a 26), se utilizan las funciones "$HallarPeorEnlace$" y "$Redirigir$" las cuales se describen a continuación:

- La función "$HallarPeorEnlace$" detecta el enlace más congestionado (MLU) luego que el primer bloque del Algoritmo de GOTEO tuvo efecto. También determina cuánto flujo tiene en "*exceso*", indicando cuánto flujo se tiene que redirigir para lograr el promedio de utilización de los enlaces de la red (puede ser otra medida)

- La función "$Redirigir$" busca caminos de largo 2 para reducir la transferencia de flujo por el enlace congestionado. Toma como entrada el enlace congestionado, el exceso y un parámetro llamado "*proporción*" que indica que porcentaje del flujo en exceso que pasa por el enlace congestionado se va a redirigir.

Como vemos en el bloque 2, primero se detecta el (MLU) y cuánto es el exceso y luego se redirige una porción de éste (siempre y cuando se pueda), este procedimiento se repite un número fijo de iteraciones.

Entradas:
Número de $PIDs$: n
Matriz de Capacidades: C
Matriz de tráficos de fondo: B
Ancho de banda de subida de los $PIDs$: U
Ancho de banda de bajada de los $PIDs$: D
Iteraciones de redirección: $iteraciones$
Velocidad de redirección: $proporcion$
Fracción de subida del nodo 1: uso
Salida:
Transferencia relativa a la capacidad de subida global: $fraccion$
Función objetivo de los ISP: $objetivo$

1: // Bloque 1: construcción de tráfico factible goloso
2: $A \leftarrow \min\{C \times (1 - B)\}$
3: **Para** $i = 1$ to n
4: $subida \leftarrow U(i)$
5: $capacidades \leftarrow \min\{A(i, :), D\}$
6: **Si** i=1 **entonces**
7: $echar \leftarrow uso \times \min\{U(i), \sum_{i=1}^{n} capacidades(i)\}$
8: **Si no**
9: $echar \leftarrow \min\{U(i), \sum_{i=1}^{n} capacidades(i)\}$
10: **Fin Si**
11: $Linea \leftarrow Waterfilling(capacidades, echar)$
12: **Para** $j = 1$ to n
13: **Si** $capacidades(i, j) > Linea$ **entonces**
14: $Poner \leftarrow capacidades(i, j) - Linea$
15: $T(i, j) \leftarrow T(i, j) + Poner$
16: $capacidades \leftarrow capacidades - Poner$
17: $D_j \leftarrow D_j - Poner; U_j \leftarrow U_j - Poner$
18: **Fin Si**
19: **Fin Para**
20: **Fin Para**
21: $transferencia \leftarrow \sum_{i=1}^{n} \sum_{j=1}^{n} T(i, j)$
22: $fraccion = \frac{transferencia}{\sum_{i=1}^{n} U(i)}$ // Bloque 2: Reducción de objetivo ISP
23: **Para** $i = 1$ to $iteraciones$
24: $(enlace, exceso) \leftarrow HallarPeorEnlace(B, C, T)$
25: $T \leftarrow Redirigir(exceso, enlace, proporcion)$
26: **Fin Para**

Figura 4.2: Algoritmo GOTEO: aplica un bloque de construcción factible golosa de tráfico, y luego redirige con Waterfilling.

4.4 Algoritmo Goloso para el Problema 1 (P1)

Para el Problema 1 (P1), en el cual se quiere maximizar el tráfico de los $PIDs$ minimizando la utilización del enlace más congestionado (3.2), surge como primera aproximación la idea de utilizar algún algoritmo goloso o greedy. Primero veremos una versión básica del algoritmo y analizaremos cuáles son sus limitaciones a la hora de resolver el Problema 1, luego se introducen mejoras al algoritmo que intentan solucionar las limitaciones del algoritmo goloso básico.

4.4.1 Greedy y su desempeño en otros problemas

Un algoritmo greedy o goloso construye una solución agregando elementos iterativamente. En cada paso agrega el próximo elemento que incrementa en menor medida cierta función objetivo. Aunque en cada iteración se hace lo mejor posible, esto no asegura que se logre construir una solución factible, ni en caso de factibilidad, una solución cuyo costo sea un óptimo global.
Existen algoritmos greedy para algunos problemas particulares que siempre encuentran la solución óptima (ej. *Kruskal* y *Prim* los cuales resuelven el problema de encontrar el árbol recubridor de costo mínimo de un grafo). Sin embargo, esto no es una regla general y en la mayoría de los problemas, no llegan a un óptimo global salvo casos excepcionales.
A continuación mostramos algunos ejemplos.

- El problema MST (Minimum Spanning Tree) consiste en hallar el subgrafo recubridor de costo mínimo de un grafo dado. El algoritmo de Kruskal, es un ejemplo de algoritmo greedy que encuentra la solución óptima al problema [14].
  El algoritmo aplicado consiste en elegir iterativamente la arista de menor costo sin que se forme un ciclo. Por tanto este problema puede resolverse en tiempo polinomial usando un algoritmo goloso, en el que en cada iteración intenta hacer lo "mejor".

- El problema TSP (Travelling Salesman Problem) consiste en elegir la mejor ruta que visite un conjunto de ciudades (nodos) dadas sin repetirlas, que se conectan a través de rutas (aristas) separadas por cierta distancia (costo). Este problema pertenece a la clase NP-hard [32] y hasta el momento no se conoce ningún algoritmo Greedy que lo resuelva en forma óptima en tiempo polinomial.

  Mostraremos con un ejemplo que un algoritmo greedy para este problema, tal que en cada nodo se elige el camino más corto para ir al siguiente sin que se cierre un ciclo antes de volver al nodo inicial, no encuentra una solución

óptima. En el grafo indicado por la Figura 4.3, partiendo del nodo A, se elige el camino más corto que es el que llega a B. Luego se escoge la arista BC, sólo queda pasar por el nodo D, entonces se elige la arista CD. Finalmente, como no se deben repetir vértices, la última arista recorrida es DA. El camino tiene un costo de 14. Sin embargo, el recorrido de costo mínimo es A-C-D-B-A, con un costo de 7.

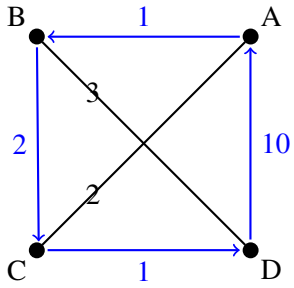


Figura 4.3: Algoritmo greedy para TSP

Dada la simplicidad de este procedimiento, es de interés analizar su desempeño. Además, en ocasiones es posible modificar las reglas en cada iteración de manera que los elementos a agregar mantengan una solución factible. De esta manera se provee una solución factible inicial que por lo general no es óptima, con la cual trabajar y modificar para disminuir su costo.

4.4.2 Algoritmo goloso simple para P4P

A continuación presentamos un algoritmo goloso en el cual cada PID sube a la red la mayor cantidad de flujo posible al resto de los $PIDs$. Este algoritmo se presenta en el Pseudo-Código mostrado en la Figura 4.4, el cual llamamos "$ConstruccionGolosa$".

El algoritmo funciona como sigue. Se toma un PID del grafo (linea 2) y se consideran todos sus vecinos de a uno (lineas 3 y 4). Se envía todo el tráfico posible a cada PID vecino en $\{1, \ldots, n\} \setminus PID_{seleccionado}$ según la capacidad de subida del $PID_{seleccionado}$, el remanente de capacidad del enlace hasta el momento, y la capacidad de descarga del vecino (linea 2 a linea 5). Se elige el tráfico a enviar acorde a la variable más restrictiva (linea 5) y se actualizan las capacidades de la red C (linea 9), los anchos de banda de subida y bajada U, D respectivamente (lineas 7 a 8) así como los flujos T (linea 6). Se procede iterativamente de forma análoga con el resto de los $PIDs$.

Entradas:
Tráfico de Fondo de los enlaces: B
Matriz de capacidades: C
Ancho de subida de $PIDs$: U
Ancho de bajada de $PIDs$: D
Salidas:
Solución factible: T

1: $T(i,j) = 0 \quad \forall i,j$
2: **Para** $i = 1$ a n
3: $Pos \leftarrow \{1,\ldots,n\} \setminus \{i\}$
4: **Para** $j = Pos$
5: $FlujoPoner \leftarrow \min\{U(i), D(j), C(i,j) \times (1 - B(i,j))\}$
6: $T(i,j) \leftarrow T(i,j) + FlujoPoner$
7: $U(j) \leftarrow U(j) - FlujoPoner$
8: $D(j) \leftarrow D(j) - FlujoPoner$
9: $C(i,j) \leftarrow C(i,j) - FlujoPoner$
10: **Fin Para**
11: **Fin Para**
12: **Retornar** T

Figura 4.4: Algoritmo "Construcción golosa"

Este algoritmo no garantiza alcanzar el máximo flujo que los $PIDs$ pueden subir a la red, lo cual no es bueno para el objetivo de los $PIDs$. Por otro lado, genera tráficos en los enlaces que son imposibles de redirigir, lo cual es un gran problema a la hora de descongestionar los enlaces más cargados y esto no es bueno para el objetivo de los ISP. Como conclusión, este algoritmo goloso puede dar resultados que no contemplen los intereses del ISP ni de los $PIDs$.

Observación: 4.4.1 *No hay razón para que el Algoritmo Goloso alcance la solución factible del Lema 4.1.5 en una instancia $I(U, D, B, C)$ optimista del problema $(P1)$. De hecho, existen ejemplos en que no lo hace.*

4.4.3 Contraejemplos de optimalidad para el algoritmo goloso simple

La filosofía del greedy presentado consiste en ser una construcción sin redirección y priorizando el objetivo de los $PIDs$, en el sentido que éstos tratarán de subir tanto flujo como les sea posible.

Es decir, al parecer no tiene sentido mandar flujo por caminos largos cuando existe un enlace directo entre dos $PIDs$, ya que al usarse menos enlaces, se saturan menos. Veremos más adelante, que en ocasiones, si los enlaces directos tienen poca capacidad, puede resultar conveniente redirigir tráfico por caminos poco congestionados de largo mayor a 1.

Por otra parte, el algoritmo goloso, a pesar de darle prioridad a los $PIDs$, tampoco logra maximizar la cantidad de flujo que estos pueden subir. Esta situación puede producirse por dos motivos:

1. por el hecho de no considerar caminos de largo mayores a uno. A modo de ejemplo, observemos la Figura 4.5. Este es un caso muy básico en el que existe un único PID capaz de transmitir, y un único PID con posibilidad de descargar. Todos los enlaces tienen igual capacidad de 1. Aunque estos dos nodos tienen la misma capacidad de dos, el enlace entre ellos no permite una transferencia de tal magnitud. En cambio, si se utilizara un camino auxiliar que utilizara el PID intermedio B, se podría duplicar el tráfico en la red.

2. por tomar malas decisiones en el ruteo. En la Figura 4.6 se muestra un ejemplo de esta situación, en la cuál tenemos sólo dos nodos que tienen capacidades de subida positivas igual a 2 (A, C) y sólo dos nodos con capacidad de bajada positivas igual a 2 (B, C). El algoritmo goloso podría elegir al nodo A para subir flujo y al nodo B para descargarlo, imposibilitando que el nodo C pueda subir flujo a la red. Si en cambio se hubiese elegido al nodo A para subir flujo y al C para descargarlo, entonces el nodo C sí podría subir flujo a la red y se alcanzaría el óptimo para los pares utilizando caminos de largo 1.

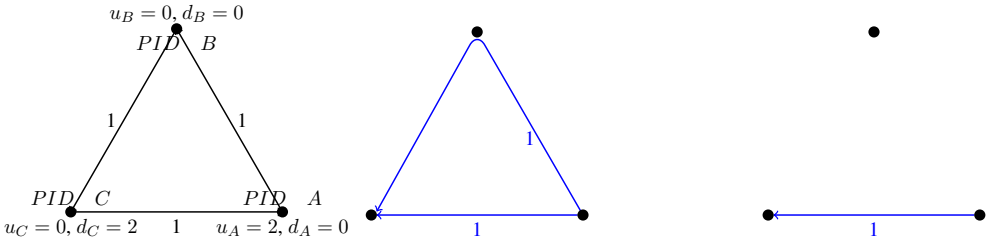
En el próximo capítulo, mostraremos un algoritmo basado en el algoritmo de Ford-Fulkerson que soluciona el problema de tomar malas decisiones de ruteo y además resuelve en tiempo polinomial el Problema 1 (P1) para el caso particular en el que sólo se envía flujo por caminos de largo 1.

Lema 4.4.2 *El Algoritmo Goloso de la Figura 4.4 no siempre logra maximizar la cantidad de flujo que pueden subir los $PIDs$.*
Prueba. *Los ejemplos mostrados en las Figuras 4.5 y 4.6 demuestran el lema.*

QED

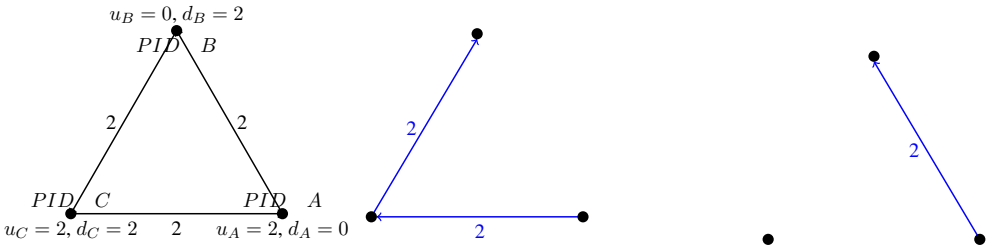
4.4.4 Construcción golosa saturando enlaces

En los simples contraejemplos propuestos para ilustrar el Lema 4.4.2, notamos que es necesario tanto alterar las características del algoritmo goloso, o agregar búsquedas locales que permitan acercarnos a mejores soluciones, estos sugieren



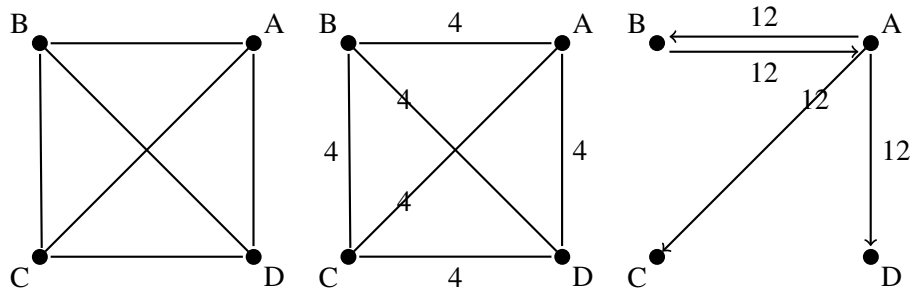
(a) Problema (b) Óptimo (c) Goloso

Figura 4.5: Ejemplo goloso con flujo no óptimo



(a) Problema (b) Óptimo (c) Goloso

Figura 4.6: Ejemplo goloso con flujo no óptimo por mal ruteo



(a) Problema (b) Óptimo (c) Goloso

Figura 4.7: Ejemplo de solución golosa

considerar caminos de mayor largo para enviar flujo.

Presentamos a continuación otro caso en el que el problema es diferente y aparece la idea de homogeneizar el flujo que circula en la red. Se considera por tanto un grafo completo de cuatro nodos como el de la Figura 4.7(a). Cada PID en el ejemplo tiene tasas de subida y de bajada $u = 40$ y $d = 12$ respectivamente. Suponemos aquí que los enlaces tienen una capacidad suficientemente grande como para no ser considerados, por ejemplo $c = 100$ para cada enlace. Obsérvese que los enlaces son bidireccionales.

Es fácil ver que la mejor manera de distribuir tráfico en estas condiciones es haciendo que cada nodo le envíe a un vecino un total de 4 unidades. Así mismo, cada nodo recibe del resto 4 unidades, llegando a su capacidad de descarga de 12.

Por otra parte, el algoritmo goloso simple tomaría un nodo al azar, por ejemplo el nodo A, e intentaría mandar la mayor cantidad de tráfico por sus enlaces. En este caso 12 unidades para cada uno de los $PIDs$ B, C y D, ya que pese a que no envía todo el tráfico disponible, se ve limitado por la capacidad de descarga de sus vecinos. A continuación, se toma otro PID al azar, por ejemplo el B (obsérvese que es independiente el nodo usado por la simetría del caso). Ahora, se aprecia que el único PID que puede aceptar tráfico es el A, ya que el resto tiene saturadas sus capacidades de descarga. Por tanto, se le envía el máximo posible que es 12. El algoritmo llega a su fin dado que ningún nodo puede aceptar más tráfico.

Como conclusión se puede notar que pese a que la cantidad de tráfico que circula por la red es en ambos casos la misma, 48 unidades, el algoritmo goloso no optimiza el objetivo del ISP que es minimizar la carga que tienen los enlaces de la red. También cabe destacar que con esta distribución de tráficos, es imposible alcanzar el óptimo redirigiendo el tráfico por caminos de mayor largo, ya que la mejor solución se obtiene cambiando las cantidades de tráfico que envía cada par.

Esto nos obliga a considerar otro tipo de algoritmo goloso.

4.4.5 Greedy con Waterfilling

Una variante del algoritmo goloso presentado inicialmente, consiste en hacer uso de las ideas sobre Waterfilling ya mencionadas en la Sección 4.3.1. De esta manera, la decisión de cuánto enviar a los vecinos se hace intentando homogeneizar la carga de los enlaces. Se introduce en el Pseudo-Código mostrado en la Figura 4.8 el algoritmo llamado "*ConstruciónGolosaWF*" el cual funciona como sigue: toma como entradas las matrices de capacidades y tráfico de fondo C y B respectivamente y los vectores de capacidades ancho de banda de subida y bajada U y D de los $PIDs$. Primero elige un PID como origen (i) para enviar flujo (linea 1), luego determinan los $PIDs$ a los que se les enviará el flujo ($PIDs_{Destino}$) (linea 3). Se determinan los parámetros a introducir en la función $waterfilling$ calculando cuánto flujo se puede subir a cada uno de los $PIDs_{Destino}$ (linea 4) y el flujo que el $PID\ i$ puede efectivamente subir a los $PIDs_{Destino}$ (linea 5). La función $waterfilling$ determina según la cantidad de flujo que puede subir el $PID\ i$ ($Poner$) y las capacidades de los $PIDs_{Destino}$ ($Capacidades$) cuál es el "tope" ($Linea$) para introducir flujo en los enlaces (linea 6). Se encuesta a cada uno de los $PIDs_{Destino}$ para determinar si estan aptos para que el $PID\ i$ les suba flujo (lineas 7 y 8). Para los $PIDs_{Destino}$ que tienen capacidad para recibir flujo, se les manda todo el flujo que se pueda sin pasarse del tope (linea 9). Por último se actualizan las matrices de tráfico (T), de capacidades (C) y los vectores de ancho de banda U y D (lineas 10 a 13)

4.5 Mejora de la solución

En esta Sección se muestra que es posible aumentar el desempeño de una solución S de algún algoritmo mediante una transformación de esta en otra denominada S', que uniformiza el uso porcentual de los enlaces internacionales.

4.5.1 Algoritmo de mejora o completitud

Sea $S = (T, \phi_{i,j})$ una solución del Problema 1, donde T es la matriz que contiene en la entrada (i, j) el tráfico neto con origen en el PID_i y destino en el PID_j, y $\phi_{(i,j)}$ es el conjunto de caminos por los cuales fluye el tráfico con terminales respectivos i y j.
Denotemos mediante $t^*_{i,j}$ al tráfico neto agregado en el enlace (i, j).

Entradas:
Tráfico de Fondo de los enlaces: B
Matriz de capacidades: C
Ancho de subida de $PIDs$: U
Ancho de bajada de $PIDs$: D
Salidas:
Solución factible: T_{fac}

1: $T(i,j) = 0 \quad \forall i,j$
2: **Para** $i = 1$ a n
3: $\quad PIDs_{Destino} \leftarrow \{1,\ldots,n\} \setminus \{i\}$
4: $\quad Capacidades \leftarrow \min\{C(i, PIDs_{Destino}), D(PIDs_{Destino})\}$
5: $\quad Poner \leftarrow \min\{U(i), \sum_{h=1}^{h=n-1} Capacidades(h)\}$
6: $\quad Linea \leftarrow waterfilling(Capacidades, -Poner)$
7: $\quad$ **Para** $j = PIDs_{Destino}$
8: $\quad\quad$ **Si** $Linea \leq Capacidades(j)$ **entonces**
9: $\quad\quad\quad FlujoPoner \leftarrow Capacidades - Linea$
10: $\quad\quad\quad T(i,j) \leftarrow T(i,j) + FlujoPoner$
11: $\quad\quad\quad U(i) \leftarrow U(i) - FlujoPoner$
12: $\quad\quad\quad D(j) \leftarrow D(j) - FlujoPoner$
13: $\quad\quad\quad C(i,j) \leftarrow C(i,j) - FlujoPoner$
14: $\quad\quad$ **Fin Si**
15: $\quad$ **Fin Para**
16: **Fin Para**
17: **Retornar** $T_{fac} \leftarrow T$

Figura 4.8: Algoritmo $ConstruccionGolosaWF$: "Construcción golosa con waterfilling"

En términos matemáticos es posible expresar este tráfico agregado mediante:

$$t^*_{i,j} = \sum_{h=1}^{n} \sum_{k=1,k\neq h}^{n} t_{h,k} I_{\phi(h,k,i,j)}, \tag{4.10}$$

donde $I_{\phi(i,j,h,k)}$ es un indicador que vale 1 sólo si el camino $\phi(i,j)$ correspondiente al tráfico $t_{i,j}$ contiene al enlace (h,k), en este mismo sentido (y cero en caso contrario).

Bajo estas condiciones, es posible evaluar el tráfico neto por cada uno de los enlaces, y consecuentemente (conociendo el tráfico de fondo y capacidades de cada

enlace) la utilización porcentual de cada enlace (el objetivo de los ISP). En particular, se observa que si agregamos tráfico en los enlaces internacionales menos congestionados es posible aumentar la transferencia neta de los pares, manteniendo el objetivo de los ISPs (siempre que no excedamos la utilización del enlace más exigido).

En el Pseudo-Código mostrado en la Figura 4.9 se presenta el algoritmo que se encarga de tal cometido. El algoritmo $CompletarSolucion$ recibe una solución del Problema 1 y aplica una normalización de utilización de enlaces, retornando otra solución que frecuentemente va a tener mayor transferencia neta entre pares, manteniendo en todos los casos el mismo uso del enlace más exigido.

Primero calcula el tráfico neto agregado entre cada uno de los enlaces de la red completa, haciendo uso de la Función $AgregarTrafico$ cumpliendo la ecuación (4.10) (lineas 1 a 5). Luego registra en la matriz F los porcentajes de utilización de todos los enlaces, $F_{i,j} = b_{i,j} + \frac{T_{i,j}}{C_{i,j}} \quad \forall i, j$, atendiendo particularmente al más congestionado (denominado (h, k)) y su correspondiente utilización máxima (variable $maximo$) (lineas 6 a 11). Las líneas 12 y 13 son de inicialización: la matriz de tráficos a la salida aún no se modificó, y los nuevos flujos es un conjunto vacío. En el último bloque (lineas 14 a 22) se encuestan todos los enlaces menos al congestionado y se calcula cual es el flujo máximo, $t^{aux}_{i,j}$, que se puede agregar al enlace (i, j) sin superar la utilización del enlace más congestionado (linea 16). Luego se sube flujo atendiendo los recursos disponibles de la red, es decir subir un flujo $t'_{(i,j)}$ para todos los nodos $i \neq j$ que cumplan que: $U_i > 0$, $D_j > 0$ y $t^{aux}_{i,j} > 0$ (linea 17). Por último se actualizan los tráficos entre los nodos y los recursos de la red (U, D y C) (lineas 18 y 19). La solución a la salida agrega entonces estos nuevos flujos, y sus correspondientes magnitudes. La solución a la entrada de esta manera es mejorada, en todos los casos que se pueda agregar flujos directos sin aumentar el uso porcentual del enlace más congestionado.

Es menester detallar las funciones que se llaman en Pseudo-Código 4.9, a saber:
$AgregarTrafico$, $ObjetivoISP$, $Maximo$, $ActualizarRecursos$ y $ParaIgualar$.

- La Función $AgregarTrafico$ simplemente aplica la Ecuación 4.10 a cada uno de los enlaces presentes en la red, con el cometido de conocer la magnitud del flujo de cada enlace.

- La Función $ObjetivoISP$ mide la utilización de cada enlaces mediante la definición del objetivo de los ISPs: $f_e = b_e + \frac{t^*_e}{c_e}$.

Entradas:
Solución: $S_{in} = (T, \phi)$
Tráfico de Fondo: B
Matriz de capacidades: C
Ancho de subida remanente de pares: U
Ancho de bajada remanente de pares: D
Salidas:
Solución mejorada: $S_{out} = (T_{out}, \phi \cup \phi^{'})$

Para $i = 1$ a n
  Para $j = 1$ a n, $j \neq i$
    $T^{*} \leftarrow AgregarTrafico(T, \phi_{i,j})$
  Fin Para
Fin Para
Para $i = 1$ a n
  Para $j = 1$ a n, $j \neq i$
    $F \leftarrow ObjetivoISP(T^{*}, B, C, i, j)$
    $(h, k, maximo) \leftarrow Maximo(F)$ // Enlace más congestionado
  Fin Para
Fin Para
$T_{out} \leftarrow T$
$\phi^{'} \leftarrow \{\}$
Para $i = 1$ a n, $i \neq h$
  Para $j = 1$ a n, $j \neq i$, $j \neq k$
    $t_{i,j}^{aux} \leftarrow ParaIgualar(F(i,j), B(i,j), C(i,j), T_{i,j}^{*}, maximo)$
    $t_{i,j}^{'} \leftarrow \min\{U(i), D(j), t_{i,j}^{aux}\}$
    $T_{out}(i,j) \leftarrow T_{out}(i,j) + t_{i,j}^{'}$
    $(U, D, C) \leftarrow ActualizarRecursos(U, D, C)$
    $\phi^{'} \leftarrow \{e_{(i,j)}\} \cup \phi^{'}$
  Fin Para
Fin Para
Retornar $S_{out} = (T_{out}, \phi \cup \phi^{'})$

Figura 4.9: Algoritmo $CompletarSolucion$: "Completación de una Solución"

- La Función $Maximo$ halla el enlace más congestionado mediante una búsqueda dentro de la matriz de utilizaciones F, como también su magnitud.

- La Función $ActualizarRecursos$ es trivial, simplemente actualiza los nue-

vos anchos de banda y capacidades.

- El objetivo de esta Función $ParaIgualar$ es calcular el tráfico que se puede agregar a la red para igualar la utilización del enlace más congestionado. Supongamos que e es un enlace (no congestionado) que posee originalmente un tráfico agregado t_e^* y respectivos tráfico de fondo y capacidad b_e y c_e. Conociendo que el enlace más exigido posee utilización f_{max} (el parámetro denominado $maximo$ en el procedimiento), entonces el tráfico adicional para igualar, t_e^{aux}, debe cumplir con la siguiente igualdad:

$$b_e + \frac{t_e^* + t_e^{aux}}{c_e} = f_{max} \tag{4.11}$$

Despejando el tráfico t_e^{aux} tenemos que:

$$t_e^{aux} = c_e(f_{max} - b_e) - t_e^* \tag{4.12}$$

Este es el tráfico que retorna la Función $ParaIgualar$. Sin embargo, puede ocurrir que el envío directo de este flujo no genere una solución factible: es preciso chequear los recursos disponibles de la red. El tráfico a adicionar se define como el máximo posible dentro de los recursos disponibles, y sin superar la utilización porcentual del enlace más exigido (flujo para igualar esta utilización dado por la ecuación (4.12)). En otras palabras, si se detecta que $t_{i,j}^{aux} > 0$ para algún par de $PIDs$ distintos i y j, es necesario chequear que el PID emisor i tenga capacidad de subida remanente ($U_i > 0$) y que el PID receptor j tenga capacidad de bajada remanente ($D_j > 0$) para que efectivamente se pueda transferir flujo entre estos. De verificarse estas condiciones entonces la magnitud del flujo a transferirse, $t'_{(i,j)}$, desde el PID i al PID j queda determinada por la Ecuación 4.13.

$$t'_{(i,j)} = \min\{U_i, D_j, t_{(i,j)}^{aux}\} \tag{4.13}$$

Obsérvese que el Algoritmo $CompletarSolucion$ no completa de manera óptima una solución factible. En efecto, supongamos que dos $PIDs$ disponen de suficiente ancho de banda como para transferir el uno al otro, pero no disponen de capacidad en sus enlaces directos. La mejor alternativa aquí es analizar la posibilidad de redirigir ese flujo potencial; no es posible en este caso enviar por el enlace directo. Una completación más eficiente de una solución debe considerar caminos no directos, teniendo presente los efectos perjudiciales inherentes a la redirección (repetición de flujo en múltiples enlaces y reduciendo capacidades).

En el siguiente Capítulo se introduce la implementación de algoritmos basados en Ford-Fulkerson que encuentran soluciones tan aproximadas a las óptimas como queramos, tanto para el Problema 1 como para las dos variantes de este ya definidas en el Capítulo 3.

Capítulo 5

Algoritmos basado en Ford-Fulkerson

En este Capítulo introducimos algoritmos basados en Ford-Fulkerson para resolver las variantes del Problema 1, $VP1_\rho$ (utilización fija de los enlaces) y $VP1$ (sólo caminos de largo uno) introducidas en el Capítulo 3. En particular el algoritmo propuesto para el $VP1_\rho$ encuentra soluciones en tiempo polinomial mientras que el propuesto para el $VP1$ es un $FPTAS$ (*Fully Polynomial Time Approximation Scheme*) [1] permitiéndonos acercarnos tanto como queramos a soluciones óptimas. Por último, con una lógica similar a la utilizada en la resolución de las variantes del Problema 1 desarrollamos un algoritmo $FPTAS$ para el Problema 1.

Recordemos que el Problema 1 (P1) es un problema multiobjetivo que consiste en encontrar un ruteo en un grafo dirigido completo $G(V, E)$ (donde V son los nodos ($PIDs$) y E son las aristas (enlaces internacionales)) que maximice el tráfico de los $PIDs$ sujeto a las restricciones de ancho de banda y capacidades de los enlaces internacionales y que además minimice la utilización del enlace internacional más congestionado (MLU *maximum link utilization*).

En el Capítulo 4 estudiamos el desempeño de un algoritmo goloso básico como candidato para resolver el Problema 1, el estudio de este algoritmo puso en evidencia sus carencias las cuales son un buen insumo para desarrollar mejores algoritmos. Los defectos más significativos del algoritmo goloso presentado en el Capítulo anterior son:

1. Genera flujos que son imposibles de redirigir que saturan los enlaces, lo cual no es deseado por los ISPs.

2. Toma malas decisiones de ruteo inhabilitando a los $PIDs$ a lograr su transferencia máxima.

3. No considera caminos de largo mayor a 1 para las transferencias de flujo, lo cual repercute tanto en el objetivo de los $PIDs$ como en el de los ISPs. Los $PIDs$ se perjudican al no lograr la máxima transferencia y los ISPs pueden verse perjudicados ya que el uso de caminos de largo mayor a 1 puede ayudar a disminuir la utilización del los enlaces, en particular del más congestionado (MLU).

A continuación presentamos dos variantes del Problema 1 las cuales ya fueron definidas en el Capítulo 3, la primer variante considera porcentajes de utilización fijos para todos los enlaces de la red ($VP1_\rho$), mientras que la segunda llamada ($VP1$) sólo considera caminos de largo uno para el tráfico de flujo. Además definiremos un algoritmo para cada una de las variantes, los cuales serán fundamentales para la construcción de un algoritmo $FPTAS$ (*Fully Polynomial Time Approximation Scheme*) para resolver el Problema 1.

5.1 Ford-Fulkerson

En esta Sección presentaremos el Algoritmo de Ford-Fulkerson el cual será utilizado para resolver todos los problemas propuestos en el presente Capítulo. Además definiremos una red de flujo auxiliar la cual se adapta a los distintos problemas planteados con el fin de poder determinar el flujo máximo aplicando el Algoritmo de Ford-Fulkerson.

5.1.1 Algoritmo de Ford-Fulkerson

El Algoritmo de Ford-Fulkerson [30] fue publicado en el año 1956 por L. R. Ford y D. R. Fulkerson. Este algoritmo permite determinar el flujo máximo entre un nodo Fuente (F) y un nodo Pozo (P), determinando el corte mínimo en una red dada $G(V, E, c)$, donde V son los nodos, E los enlaces y c las respectivas capacidades de los enlaces. El Algoritmo de Ford-Fulkerson es un procedimiento que sistemáticamente busca caminos de aumento que conecten al nodo (F) con el nodo (P), y de esta forma introducir flujo por este camino. Salvo en los casos que las capacidades toman valores irracionales, el algoritmo encuentra la solución en un número finito de iteraciones, en estos casos la complejidad del algoritmo es del orden $O(N \times |f|)$, donde $|f|$ es la magnitud del flujo. Existen otras versiones, por ejemplo, la que considera primero los caminos de aumento más cortos entre F y P,

introducida por Edmonds-Karp [19] en el 1972. En este documento utilizaremos la versión del algoritmo de Ford-Fulkerson planteada por Edmonds-Karp.

5.1.2 Red auxiliar

Todos los algoritmos presentados en este Capítulo se basan en la idea de encontrar el flujo máximo entre dos nodos que determinaremos como fuente (F) y pozo (P). Para lograr este cometido aplicaremos el algoritmo de Ford-Fulkerson en la red. El problema que surge en este caso es que hay que tener ciertas consideraciones en la topología de la red a utilizar para poder aplicar Ford-Fulkerson y que el flujo y los caminos respeten las restricciones de los distintos problemas. En otras palabras si consideramos la topología de la red original para la aplicación de los algoritmos podemos caer en ruteos que no cumplan con las restricciones del problema.
Para solucionar estas dificultades vamos a construir una red auxiliar en la cual cualquier ruteo factible en la red original sea interpretable en la red auxiliar y viceversa (es decir que existe una biyección entre ambas redes).
La idea principal es considerar una *duplicación* (splitting) de los $PIDs$ dentro de la red original, de modo de poder mapear los flujos de una red auxiliar en la red original.
Si la red dispone de los $PIDs$ $\mathcal{N} = \{1, \ldots, N\}$, una duplicación considera $PIDs$ primos a éstos $\mathcal{N}^{'} = \{1^{'}, \ldots, N^{'}\}$, donde ahora los los $PIDs \in \mathcal{N}$ se denominan $PIDs$ transmisores, mientras que a los $PIDs \in \mathcal{N}^{'}$ serán los receptores, es decir ahora cada PID de la red original tendrá asociado un PID transmisor y uno receptor. Unamos entonces los $PIDs$ transmisores en $\mathcal{N}$ a una fuente ideal F, estos enlaces son dirigidos y van desde el nodo F a los $PIDs$ en $\mathcal{N}$. La misma lógica se repite con los $PIDs$ receptores en $\mathcal{N}^{'}$ con un nodo pozo ideal P, pero ahora la dirección de estos enlaces es desde los $PIDs$ en $\mathcal{N}^{'}$ al nodo P. Luego se crean arcos que unen los $PIDs$ transmisores con todos los receptores menos con sus correspondientes $PIDs$ receptores. Por último se introducen arcos entre los $PIDs$ receptores y sus correspondientes $PIDs$ transmisores.
Esta red auxiliar es en efecto una red de flujo, la cual cumple con las siguientes propiedades:

- **Acotación**: $f(u, v) \leq c(u, v) \quad u, v \in V$, establece que el flujo a través de un enlace no puede exceder la capacidad de este.

- **Antisimetría**: $f(u, v) = -f(v, u) \quad u, v \in V$, significa que a un flujo de un vertice u a otro v le corresponde un flujo igual pero de signo contrario que va de v a u.

- **Conservación de flujo**: Si $u \neq F, P \Rightarrow \sum_{v \in V} f(u, v) = 0$, esta condición

significa que en los vertices distintos a la fuente (F) y pozo (P) no hay pérdida ni acumulación de flujo, en otras palabras, lo que llega debe ser igual a lo que sale.

La red auxiliar $R = (V, E, C)$ queda definida como:

- Nodos:

$$V = \{F, P\} \cup \mathcal{N} \cup \mathcal{N}^{'}$$

- Enlaces:

$$\begin{aligned} E = &\{(F, i),\ \forall i \in \mathcal{N}\} \cup \\ &\{(i, j^{'}),\ \forall i \neq j^{'} : i \in \mathcal{N}, j^{'} \in \mathcal{N}^{'}\} \cup \\ &\{(i^{'}, i),\ \forall i = i^{'} : i \in \mathcal{N}, i^{'} \in \mathcal{N}^{'}\} \cup \\ &\{(i^{'}, P),\ \forall i^{'} \in \mathcal{N}^{'}\} \end{aligned}$$

- Capacidades de los enlaces:

$$\begin{aligned} C = &\{c_{i,j}(\rho_{max} - b_{i,j}),\ \forall i \neq j : i \in \mathcal{N}, j \in \mathcal{N}^{'}\} \cup \\ &\{\infty; (j, i),\ \forall j = i : i \in \mathcal{N}, j \in \mathcal{N}^{'}\} \cup \\ &\{u_i; (P, i),\ \forall i : i \in \mathcal{N} \cup \\ &\{d_i; (j, F),\ \forall j : j \in \mathcal{N}^{'}\} \end{aligned}$$

Donde $c_{i,j}$ y $b_{i,j}$ es la capacidad y el tráfico de fondo del enlace $\{i, j\}$ de la red original, u_i y d_i son las capacidades de encho de banda de subida y bajada del $PID\ i$.

Ejemplo de construcción de una red auxiliar

Presentamos a continuación un ejemplo de la conversión de la red original a la red auxiliar. A modo ilustrativo, se utiliza un grafo completo con tan solo tres $PIDs$ y se asume que no existe tráfico de fondo, es decir $b_e = 0 \quad \forall e \in E$. Los pasos de la conversión son los siguientes:

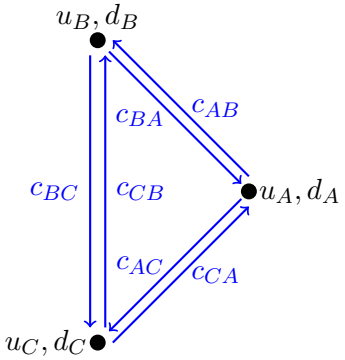


Figura 5.1: Red original

1. En primer lugar, se replican los $PIDs$ de la red (Figura 5.2), es decir que hacemos un *duplicación* de los $PIDs$, donde aquellos que se encuentran en la misma horizontal, corresponden a un mismo PID original. Llamaremos a los $PIDs$ de la izquierda como nodos transmisores, y a los de la derecha, como nodos receptores. Cada PID original tendrá su respectivo nodo transmisor y receptor en el nuevo grafo.

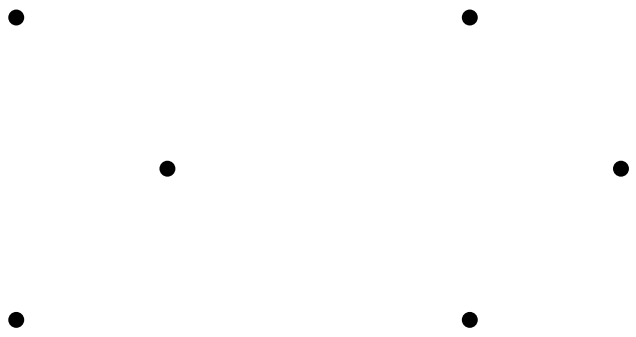


Figura 5.2: Paso 1

2. A su vez, se agregan dos nodos auxiliares, un nodo fuente F y un nodo pozo P (Figura 5.3).

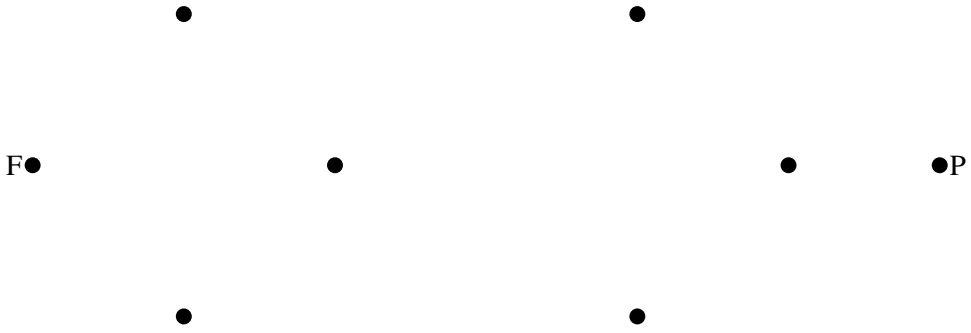


Figura 5.3: Paso 2

3. Se dibujan enlaces dirigidos entre el nodo F y cada uno de los $PIDs$ transmisores (Figura 5.4). La capacidad de dichos enlaces se fija igual a la capacidad de subida original de cada nodo. Por tanto, el tráfico máximo que saldrá de cada nodo transmisor hacia la parte derecha de la red está acotada por su capacidad del enlace con el nodo fuente.
   Por otra parte, se dibujan los enlaces dirigidos entre cada uno de los $PIDs$ receptores y el nodo P. De igual manera, las capacidades de estos enlaces se establecen como las capacidades de bajada inicial.

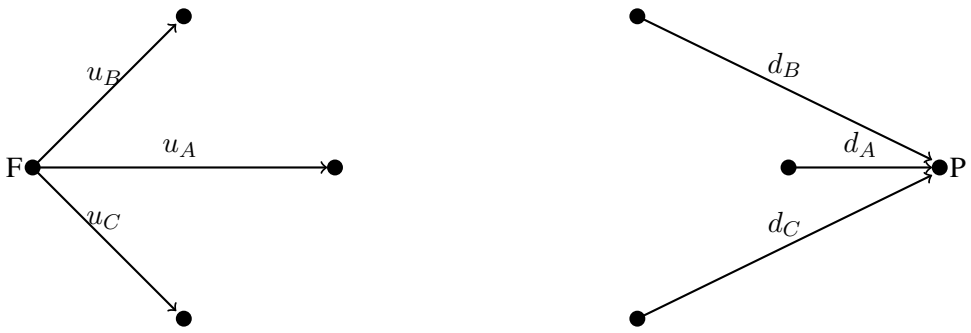


Figura 5.4: Paso 3

4. Los enlaces de los nodos transmisores con los receptores tienen las mismas capacidades que originalmente (Figura 5.5). Cabe destacar que un nodo

transmisor, solamente enviará tráfico a todos los nodos receptores, exceptuando aquel que corresponde consigo mismo.

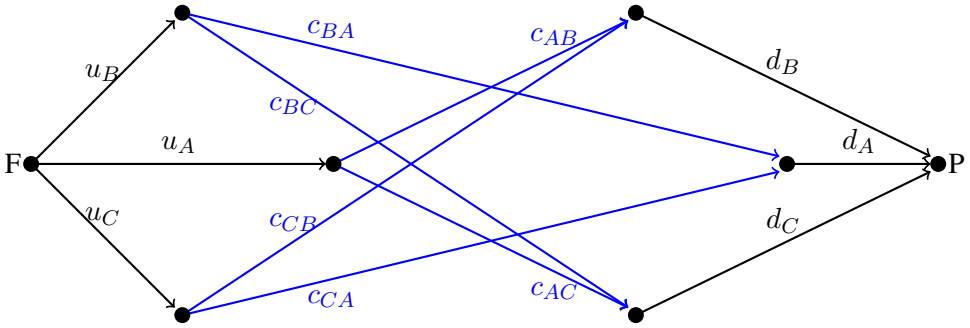


Figura 5.5: Paso 4

5. Por último se introducen los enlaces dirigidos que van desde los $PIDs$ receptores a sus correspondientes $PIDs$ transmisores (Figura 5.6), estos enlaces tendrán capacidad infinita y son los que permiten que existan caminos de largo mayor a uno para el ruteo de tráfico.

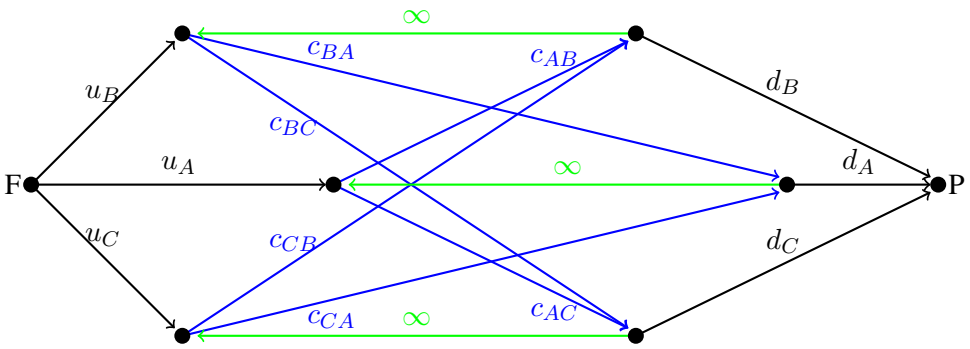


Figura 5.6: Adaptación para flujo máximo

5.2 Variantes del Problema 1

5.2.1 Variante 1: Utilización fija de los enlaces de la red ($VP1_\rho$)

Como vimos en el Capítulo 3 la variante del Problema 1, $VP1_\rho$, consiste básicamente en fijar una utilización máxima admisible, ρ_{max}, y maximizar entonces la función objetivo de transferencia neta entre los $PIDs$.

Recordando de la Sección 3.3.1 el Problema $VP1_\rho$ se plantea matemáticamente de la siguiente manera:

$$VP1_\rho \begin{cases} \max \sum_i \sum_{j \neq i} t_{ij} \\ s.a. \\ \quad \sum_{j \neq i} t_{ij} \leq u_i \quad \forall PID_i & (5.1) \\ \quad \sum_{j \neq i} t_{ji} \leq d_i \quad \forall PID_i & (5.2) \\ \quad t_{ij} \geq 0 \quad \forall i \neq j \\ \quad b_e + \sum_i \sum_{j \neq i} \frac{t_{ij} \times I_e(i,j)}{c_e} \leq \rho_{max} & (5.3) \end{cases}$$

Definición 5.2.1 *Diremos que un algoritmo logra una solución óptima el $VP1_\rho$, cuando dada una utilización máxima ρ_{max}, retorna una solución factible que logra la transferencia óptima T^*, siempre que exista.*

Es interesante destacar la utilidad práctica de esta variante: si un acuerdo entre proveedores del servicio de Internet fijan un umbral del 90% de utilización de los enlaces internacionales, se procura entonces maximizar la transferencia entre los $PIDs$, respetando con el respectivo umbral. Daremos a continuación un resultado que caracteriza la complejidad del Problema $(VP1_\rho)$:

Teorema 5.2.2 $(VP1_\rho) \in \mathcal{P}$.

Prueba. Tenemos que demostrar que podemos encontrar el flujo máximo para los $PIDs$, dada una utilización fija $\rho_{max} \in [\max_e\{b_e\}, 1]$ aplicando un algoritmo que ejecute en tiempo polinomial.
Si consideramos la red auxiliar $R(V, E, C)$ introducida en la sub Sección 5.1.2, y determinamos las capacidades de los enlaces entre $PIDs$ transmisores y receptores como $c'_{i,j} = c_{i,j} \times (\rho_{max} - b_{i,j})$, entonces aplicando el algoritmo de Ford-Fulkerson en $R(V, E, C)$ obtenemos el flujo máximo T^* el cual es óptimo para los

$PIDs$, para el porcentaje de utilización fijo ρ_{max}. Como ya vimos el algoritmo de Ford-Fulkerson es de orden polinomial, por lo cual sólo falta corroborar que la solución encontrada al aplicar dicho algoritmo verifique las restricciones que hacen referencia a las ecuaciones 5.1, 5.2 y 5.3 del Problema $VP1_\rho$, lo cual vemos a continuación:

- Por construcción de la red auxiliar, los enlaces que van desde el nodo fuente (F) a los $PIDs$ transmisores coinciden con las respectivas capacidades de ancho de banda de subida $u_i \, \forall i \in \{1 \ldots, N\}$, entonces por conservación de flujo es imposible que un nodo transmisor suba más flujo que el que efectivamente puede subir, con lo cual se verifica la restricción 5.1 de forma análoga se demuestra que también se verifica la restricción 5.2.

- La restricción 5.3 se verifica debido a que las capacidades asociadas a los enlaces que van desde los nodos transmisores a los receptores se definen con magnitudes iguales a $c'_{i,j} = c_{i,j} \times (\rho_{max} - b_{i,j}) \, \forall i \neq j$, $i \in \mathcal{N}$, $j \in \mathcal{N}'$, asegurando el cumplimiento de la restricción de utilización máxima, pues todo flujo factible en la red así construida cumple que $t_e \leq c_e(\rho_{max} - b_e)$. Consecuentemente, $b_e + t_e/c_e \leq \rho_{max}$.

QED

El Teorema 5.2.2 asegura la polinomialidad de $VP1_\rho$. Sin embargo el algoritmo de Ford-Fulkerson aplicado a la red auxiliar no siempre genera soluciones globalmente óptimas, esto se debe a que existen instancias en las cuales se producen ciclos para los ruteos de flujo por caminos de largo mayor a 1. Sobre este tema vamos a profundizar más en la Sección 5.4. Esta dificultad inspiró el planteo de una nueva variante del P1 la cual considera ruteos de flujo sólo por caminos de largo 1 evitando así la presencia de ciclos no deseados en el ruteo de tráfico.

5.2.2 Variante 2: Considerando sólo caminos de largo uno ($VP1$)

Consideremos la variante del Problema 1, $VP1$, introducida en el Capítulo 3, en la cual se busca maximizar la cantidad de flujo que los $PIDs$ pueden subir a la red utilizando sólo caminos de largo 1 sujeto a las mismas restricciones de ancho de banda y capacidades que en P1. Definimos el $VP1$ matemáticamente como:

$$VP1 \quad \begin{cases} \min \max_{e \in E} \{b_e + \sum_i \sum_{j \neq i} \frac{t_{ij} \times I_e(i,j)}{c_e}\} \\ \quad s.a. \\ \quad\quad max \sum_i \sum_{j \neq i} t_{ij} \\ \quad\quad s.a. \\ \quad\quad\quad \sum_{j \neq i} t_{ij} \leq u_i \quad \forall i \in \{1, \ldots, N\} \\ \quad\quad\quad \sum_{j \neq i} t_{ji} \leq d_i \quad \forall i \in \{1, \ldots, N\} \\ \quad\quad\quad t_{ij} \geq 0 \quad \forall i \neq j \in \{1, \ldots, N\} \\ \quad\quad\quad \sum_i \sum_{j \neq i} t_{ij} \times I_e(i,j) \leq v_e \\ \quad\quad\quad |Camino_{(i,j)}| = 1 \, \forall i \neq j \in \{1, \ldots, N\} / t_{(ij)} > 0 \end{cases} \tag{5.4}$$

Observemos que en 5.4 se introduce la restricción $|Camino_{(i,j)}| = 1$ la cual nos limita a sólo considerar caminos de tamaño 1 para todas las transferencias de flujo.
Para esta variante proponemos un algoritmo que aplica iterativamente el algoritmo de Ford-Fulkerson, de esta forma podemos obtener soluciones tan cercanas a las óptimas como queramos. Los algoritmos que poseen esta propiedad se los denomina $FPTAS$ [1].

Nuevamente, el algoritmo que desarrollaremos a continuación se aplicará sobre una red auxiliar muy similar a la definida en 5.1.2. Esta nueva red que denotaremos $G^*(V, E, C)$, se diferencia de la original sólo en que no se consideran los enlaces que unen los nodos transmisores con sus respectivos receptores. La red $G^*(V, E, C)$ queda definida como:

- Nodos: son los mismos que en la red auxiliar

$$V = \{F, P\} \cup \mathcal{N} \cup \mathcal{N}'$$

- Enlaces: se quitan los enlaces que que van desde los $PIDs$ receptores a los transmisores

$$\begin{aligned} E = &\{(F, i), \forall i \in \mathcal{N}\} \cup \\ &\{(i, j'), \forall i \neq j' : i \in \mathcal{N}, j' \in \mathcal{N}'\} \cup \\ &\{(i', P), \forall i' \in \mathcal{N}'\} \end{aligned}$$

- Capacidades de los enlaces: se quitan las capacidades definidas como infinito de los enlaces sacados

$$C = \{c_{i,j}(\rho_{max} - b_{i,j}),\ \forall i \neq j : i \in \mathcal{N}, j \in \mathcal{N}^{'}\} \cup$$
$$\{u_i; (P, i),\ \forall i : i \in \mathcal{N} \cup$$
$$\{d_i; (j, F),\ \forall j : j \in \mathcal{N}^{'}\}$$

A modo ilustrativo se muestra en la Figura 5.7 la red $G^*(V, E, C)$ para el $VP1$ para tres $PIDs$.

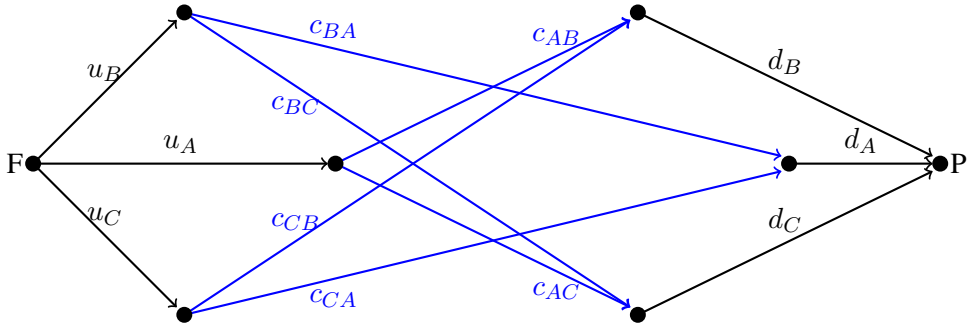


Figura 5.7: Paso 4

5.3 Algoritmo para resolver el Problema (VP1)

En la red G^* sólo se puede transferir flujo entre $PIDs$ por caminos de largo 1, esto es por construcción ya que no existen enlaces que comuniquen a los $PIDs$ receptores con sus respectivos transmisores y las direcciones de los enlaces entre PIDs solo se da en una dirección.
Cualquier camino de largo 1 generado en la red original puede implementarse en G^* y viceversa (existe una biyección para ruteos de tamaño 1), es por este motivo que aplicando Ford-Fulkerson en el grafo G^* podemos calcular el flujo máximo entre los nodos (F) y (P) el cuál coincide con el tráfico máximo que pueden alcanzar los $PIDs$ por caminos de largo 1 atendiendo las restricciones de anchos de banda y capacidades. Observemos también que a diferencia del algoritmo goloso mostrado en el Capítulo 4, éste no genera soluciones que caigan en el ejemplo mostrado en la Figura 4.6 ya que el algoritmo de Ford-Fulkerson es capaz de corregir malas decisiones de ruteo.

5.3.1 Pasos del algoritmo

El algoritmo se aplica en dos etapas:

I. Se calcula el máximo flujo que los $PIDs$ pueden subir a la red por caminos de largo 1 (F_Max), sujeto a las restricciones de anchos de banda y capacidades de los enlaces internacionales, utilizando el algoritmo de Ford-Fulkerson en G^*. Matemáticamente:

$$
P(F_Max)\begin{cases}
\max \sum_i \sum_{j\neq i} t_{ij} & \\
s.a & \\
\sum_{j\neq i} t_{ij} \leq u_i \quad \forall PID_i & (5.5)\\
\sum_{j\neq i} t_{ji} \leq d_i \quad \forall PID_i & (5.6)\\
t_{ij} \geq 0 \quad \forall i \neq j & \\
\forall e : b_e + \sum_i \sum_{j\neq i} \frac{t_{ij} \times I_e(i,j)}{c_e} \leq 1 & (5.7)\\
|Camino_{(i,j)}| = 1 \, \forall i \neq j \in \{1,\ldots,N\} / t_{(ij)} > 0 & (5.8)
\end{cases}
$$

Lema 5.3.1 *El algoritmo de Ford-Fulkerson encuentra el flujo máximo entre $PIDs$ (F_Max) para ruteos de tamaño 1.*

Prueba. Para demostrar este Lema, alcanza con probar que al aplicar el algoritmo de Ford-Fulkerson en G^* se verifican las restricciones dadas por a las ecuaciones 5.5, 5.6, 5.7 y 5.8.

Por construcción las aristas que van desde el nodo fuente (F) a los $PIDs$ transmisores coinciden con las respectivas capacidades de ancho de banda de subida $u_i \, \forall i \in \{1\ldots,N\}$, entonces por conservación de flujo es imposible que un PID transmisor suba más flujo que el que efectivamente puede subir, con lo cual se verifica la restricción 5.5 y por analogía también se verifica la restricción 5.6.

La restricción 5.7 se verifica debido a que las capacidades asociadas a los enlaces que van desde los nodos transmisores a los receptores se definen con

magnitudes iguales a las capacidades en la red original, por lo cual cualquier flujo entre pares de nodos nunca superará la capacidad del enlace.

Por último la restricción 5.8 está contemplada por construcción de G^*, como ya dijimos antes, no existen enlaces entre nodos receptores y sus correspondientes transmisores y los enlaces sólo van en un sentido (transmisor→ receptor).

QED

II. Al resolver el Problema ($P(F_Max)$) obtenemos el flujo máximo que pueden subir los $PIDs$ (F_Max). Por el Lema 5.3.1 F_Max es factible y óptimo para los $PIDs$ en esta topología (sólo caminos de tamaño 1), ahora tenemos que "acomodar" este flujo de tal forma que genere la menor utilización del enlace más congestionado (objetivo de los ISPs).
La segunda etapa del algoritmo se encarga de encontrar un ruteo que produzca una utilización mínima de todos los enlaces y en particular del enlace más congestionado (MLU), satisfaciendo el objetivo de los $PIDs$ (que la transferencia total entre los $PIDs$ sea igual a $F_Max = \sum_i \sum_{i \neq j} t_{ij}$). Para esta etapa nos proponemos resolver el Problema $P(E_min)$, buscando $\alpha_{opt} \in [\min_{e \in E}\{b_e\}, 1]$ el cual nos indica cuál es la mínima utilización de los enlaces que se puede obtener cumpliendo el objetivo de los $PIDs$ y respetando las restricciones de capacidades de enlaces y anchos de banda.

Matemáticamente el Problema $P(E_min)$ queda definido como:

$$
P(E_min)\begin{cases}
\min\limits_{\alpha\in[\min_{e\in E}\{b_e\},1]} \quad \alpha \\
s.a. \\
\quad \sum\limits_i \sum\limits_{j\neq i} t_{ij} = F_Max & (5.9) \\
\quad \sum\limits_{j\neq i} t_{ij} \leq u_i \quad \forall PID_i \\
\quad \sum\limits_{j\neq i} t_{ji} \leq d_i \quad \forall PID_i \\
\quad t_{ij} \geq 0 \quad \forall i \neq j \\
\quad \forall e \in Y = \{e : b_e \leq \alpha\} : \\
b_e + \sum\limits_i \sum\limits_{j\neq i} \dfrac{t_{ij} \times I_e(i,j)}{c_e} \leq \alpha & (5.10) \\
\quad \forall e \in \bar{Y} = \{e : \rho_e > \alpha\} : \\
\sum\limits_i \sum\limits_{j\neq i} \dfrac{t_{ij} \times I_e(i,j)}{c_e} = 0 & (5.11) \\
|Camino_{(i,j)}| = 1 \, \forall i \neq j \in \{1,\ldots,N\} / t_{(ij)} > 0 & (5.12)
\end{cases}
$$

Como se puede observar el problema planteado en $P(E_min)$ difiere del Problema 1 (P1) en tres cosas:

i. La restricción de segundo nivel $\max \sum_i \sum_{j\neq i} t_{ij}$ de (P1) cambia por 5.9, esto se justifica porque en $P(F_Max)$ calculamos el flujo máximo de los $PIDs$, F_Max, y sabemos que es la máxima transferencia posible, es por esto que aparece una igualdad.

ii. La restricción del Problema 1 que refiere a la utilización de los enlaces internacionales se la sustituye por dos restricciones nuevas 5.10 y 5.11, las cuales distinguen dos casos: (1) e pertenezca al conjunto de enlaces cuyo porcentaje de tráfico de fondo es menor o igual a α, Y, (2) e pertenezca al conjunto de enlaces cuyo porcentaje de tráfico de fondo es mayor que α, $\bar{Y}$. Estas restricciones nos aseguran que el ruteo se realizará por enlaces que tienen un porcentaje de utilización menor o igual a α.

iii. Se agrega la restricción 5.12 que indica que sólo se permiten ruteos por caminos de largo 1.

Para encontrar el valor mínimo α_{opt}, aplicamos reiteradas veces el Teorema de Bolzano [28] hasta aproximarnos tanto como queramos al valor óptimo de utilización α_{opt}:

$$\alpha_{opt} = \min_{\alpha \in [\min\{b_e\},1]} \{F_Max - F_Max(\alpha) = 0\} \tag{5.13}$$

Donde $F_Max(\alpha)$ es el flujo máximo que pueden transferir los $PIDs$ utilizando sólo enlaces que tienen una utilización menor que α y no permitiendo que estos enlaces superen el umbral α cuando la transferencia de flujo comienza.
Aplicando el algoritmo de Ford-Fulkerson en cada iteración podemos testear la condición mostrada en 5.13.
A continuación se detalla la forma de aproximarnos al valor α_{opt}.

1. Para la iteración $t = 1$ consideramos $\alpha_1 = 1$, evidentemente el valor del flujo máximo $F_Max(\alpha)$ para $\alpha = 1$ va a coincidir con F_Max, obtenido de resolver el Problema $P(F_Max)$, ya que tomando $\alpha = 1$ estamos utilizando todos los recursos disponibles de la red.

2. Para $t = 2$, $\alpha_2 = \frac{(\min_e\{b_e\}+1)}{2}$, para este valor pueden pasar dos cosas:

   - Si $\sum_i \sum_{j \neq i} t_{ij} = F_Max$, en este caso en $t = 3$, $\alpha_3 = \frac{(\min_e\{b_e\}+\alpha_2)}{2}$
   - Si $\sum_i \sum_{j \neq i} t_{ij} \neq F_Max$, en este caso en $t = 3$, $\alpha_3 = \frac{(\alpha_2+1)}{2}$

3. Se sigue iterando aplicando el Teorema de Bolzano tomando biparticiones del intervalo $[\min_{e \in E}\{b_e\}, 1]$, de esta forma para la iteración t el valor absoluto de la diferencia entre α_{opt} y α_t será menor a $\frac{(1-\min_e\{b_e\})}{2^t}$, la longitud del intervalo. Cuando se alcanza una tolerancia preestablecida, ε, el algoritmo termina. El criterio de parada se toma en función de la longitud del intervalo ya que se cumple:

$$\mid \alpha_{opt} - \alpha_t \mid < \mid \alpha_t - \alpha_{t-1} \mid < \frac{1}{2^{t-1}} < \varepsilon$$

En el Pseudo Código mostrado la Figura 5.8 se detalla cómo opera el algoritmo para encontrar el ruteo que minimiza la saturación de enlaces utilizando sólo caminos de largo 1. En este se ingresa como parámetros de entrada: la matriz de tráfico de fondo B, las capacidades de subida y bajada, U y D, la matriz de capacidades de los enlaces C, el número máximo de iteraciones $iter$ y la tolerancia tol. Se calcula el flujo máximo (F_{Max}) que se puede rutear por caminos de largo 1 utilizando el algoritmo de Ford-Fulkerson, se inicializan las variables α, $Lsup$, $Linf$ y Dif (lineas 1 a 5). Luego se empieza a iterar calculando F_α con la función max_flow, la cuál aplica Ford-Fulkerson no permitiendo la utilización de enlaces que tienen

un porcentaje de saturación mayor o igual a α y se compara F_α con F_{Max} (lineas 7 y 8). Si $F_{Max} - F_\alpha = 0$ se computa $\alpha = \alpha - \frac{(L_{sup} - L_{inf})}{2}$ para la siguiente iteración; en caso que $F_{Max} - F_\alpha > tol$ se computa $\alpha = \alpha + \frac{(L_{inf} - L_{sup})}{2}$ para la siguiente iteración (lineas 9 a 18).

Entradas: Conjunto de nodos de la red: $PIDs$
Tráfico de Fondo de los enlaces: B
Matriz de capacidades: C
Ancho de banda de subida y bajada de $PIDs$: U y D
Máximo de iteraciones: $iter$
Tolerancia: ε
Salidas: Mejor solución: α_{opt}

1: $\alpha_1 \leftarrow 1$
2: $\alpha_0 \leftarrow 0$
3: $F_{Max} \leftarrow max_flow(U, D, C, B, \alpha_1)$
4: $L_{sup} = 1$
5: $L_{inf} = \min_{e \in E}\{b_e\}$
6: $Int \leftarrow 1$
7: $i \leftarrow 1$
8: **Mientras** $(\varepsilon < Int) \cap (i < iter)$
9: $\quad F_{\alpha_i} \leftarrow max_flow(U, D, C, B, \alpha_i)$
10: $\quad Dif \leftarrow F_{Max} - F_{\alpha_i}$
11: $\quad$ **Si** $Dif == 0$ **entonces**
12: $\quad\quad L_{sup} \leftarrow \alpha_i$
13: $\quad\quad \alpha_{i+1} \leftarrow \frac{(L_{inf} + L_{sup})}{2}$
14: $\quad$ **Fin Si**
15: $\quad$ **Si** $Dif > 0$ **entonces**
16: $\quad\quad L_{inf} \leftarrow \alpha_i$
17: $\quad\quad \alpha_{i+1} \leftarrow \frac{(L_{inf} + L_{sup})}{2}$
18: $\quad$ **Fin Si**
19: $\quad Int \leftarrow \mid \alpha_{i+1} - \alpha_i \mid;\ i \leftarrow i + 1$
20: **Fin Mientras**
21: **Retornar** $\alpha_{opt} \leftarrow \alpha$

Figura 5.8: Algoritmo “MínimoEnlace”

Observación: 5.3.2 *Es importande destacar que la saturación mínima alcanzada por el Algoritmo "**MínimoEnlace**" encuentra cuál es la utilización mínima de los enlaces requerida para lograr el objetivo de los $PIDs$. Es por esto que el enlace más congestionado no siempre tendrá una utilización igual a α_{opt}, hay que tener también en cuenta el porcentaje de saturación debido al tráfico de fondo. Es por este motivo que el porcentaje de utilización del enlace más congestinado (MLU) será igual a $\rho_{max} = \max\{\alpha_{opt}, b_{max}\}$ donde $b_{max} = \max_{e \in E}\{b_e\}$.*

Definición 5.3.3 *Un algoritmo A es un **FPTAS** (*Fully Polynomial Time Approximation Scheme*) para un problema de optimización P, si dada una entrada I para P y $\varepsilon > 0$, A encuentra en tiempo polinomial en el tamaño de I y en $\frac{1}{\varepsilon}$, una solución S para I que satisface:*

$$\mid val(I) - val(S) \mid \leq \varepsilon \times val(I)$$

Donde $val(I)$ es el valor óptimo de una solución para I.

Lema 5.3.4 *Dado una tolerancia $\varepsilon > 0$, una instancia $I(U, D, C, B)$ y la menor utilización antes de empezar a transferir flujo $b_{min} = \min_e\{b_e\}$, el algoritmo "*MínimoEnlace*" es un $FPTAS$ para el problema $VP1$.*

Prueba. Como ya lo mencionamos el algoritmo "*MínimoEnlace*" funciona en dos etapas, en la primera utiliza el algoritmo de Ford-Fulkerson para calcular el flujo máximo que pueden subir los $PIDs$ por caminos de largo 1 (F_{Max}). Luego, un esquema de bipartición es tomado en el intervalo $[b_{min}, 1]$ para encontrar la mínima utilización, α_{opt}, que mantiene la transferencia máxima F_{Max}.
El número de iteraciones t debe cumplir:

$$\frac{1 - b_{min}}{2^t} < \varepsilon$$

Entonces, para el número de iteraciones t tenemos que:

$$i > log_2[1 - b_{max}] + log_2\frac{1}{\epsilon}$$

Tomando $t^* = \lceil log_2[1 - b_{max}] + log_2\frac{1}{\epsilon}\rceil$, entonces el número de iteraciones es polinomial en $\frac{1}{\varepsilon}$ ya que:

$$t^* = \lceil log_2[1 - b_{max}] + log_2\frac{1}{\epsilon}\rceil < 1 + \lceil log_2[1 - b_{max}]\rceil + \frac{1}{\epsilon}$$

Además sabemos que el algoritmo de Ford-Fulkerson (usado en cada iteración del algoritmo "*MínimoEnlace*" para determinar F_α) es polinomial con el número de nodos ($PIDs$) en la red. Como consecuencia el algoritmo "*MínimoEnlace*" es un $FPTAS$ para resolver el Problema $VP1$

QED

5.4 Algoritmo para resolver el Problema 1 (P1)

Para resolver el P1 es necesario que se pueda calcular el flujo máximo que pueden subir los $PIDs$ sin restricción en el largo de los caminos, es decir considerar una topología para aplicar Ford-Fulkerson que permita generar ruteos de largo mayor o igual a 1. Para lograr tal cometido consideremos la red G^{**} la cual coincide con la red auxiliar definida en 5.1.2. En la red G^{**} cualquier ruteo en la red original puede interpretarse en G^{**} y viceversa (existe una biyección para cualquier ruteo entre la red original y G^{**}).
A modo ilustrativo en la Figura 5.9 se muestra un ejemplo de la red G^{**} para el caso de tres $PIDs$.

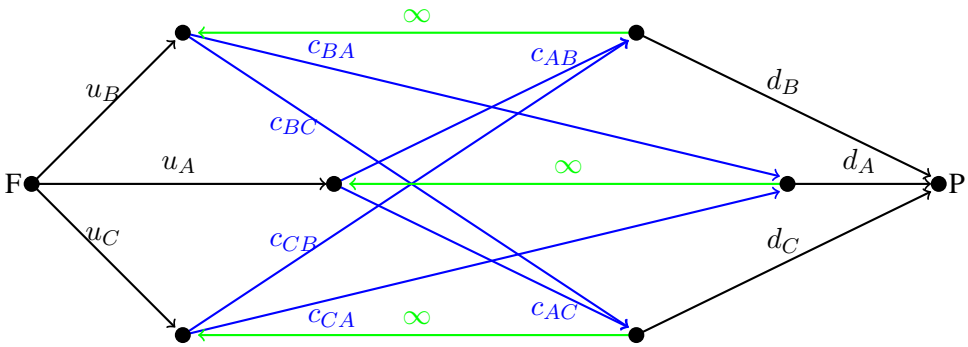


Figura 5.9: Adaptación para flujo máximo

Como existe esta biyección entre los ruteos entonces es posible encontrar el flujo máximo que pueden subir los $PIDs$ aplicando el algoritmo de Ford-Fulkerson sobre G^{**} y luego utilizando el algoritmo "*MínimoEnlace*" también sobre G^{**}, es posible encontrar un ruteo que minimice la utilización de todos los enlaces y en particular la utilización del enlace más congestionado (MLU).

Lema 5.4.1 *Dado una tolerancia $\varepsilon > 0$ y una instancia $I(U, D, C, B)$, el algoritmo "*MínimoEnlace*" es un $FPTAS$ para el problema $P1$.*
Prueba. *La demostración es análoga a la demostración del Lema 5.3.4.*

QED

5.4.1 Ruteos que forman ciclos

Un problema que debemos atender es que al aplicar Ford-Fulkerson en G^{**} es posible que se generen ruteos que se interpretan como ciclos en la red original. Este inconveniente se muestra claramente en la Figura 5.10, en la cual se ve en rojo el ruteo. La interpretación de este ejemplo en la red original es que el PID A sube flujo a la red y luego el mismo PID A descarga este flujo utilizando al PID B como auxiliar. Estas redundancias en el ruteo sólo se pueden producir cuando la transferencia de flujo se realiza por caminos de largo mayor a 1.

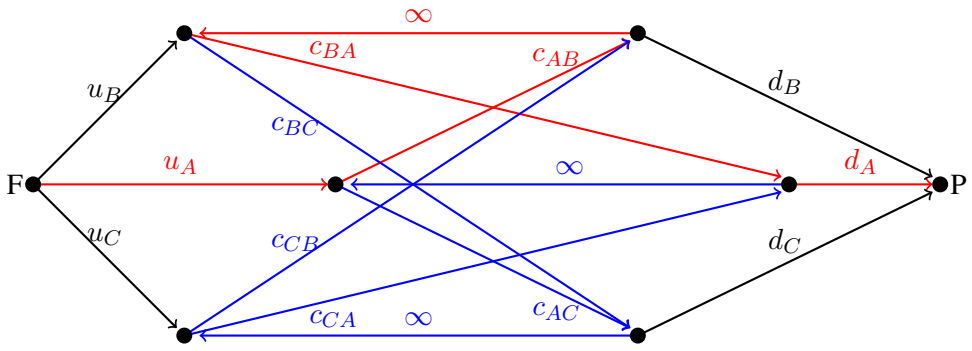


Figura 5.10: Adaptación para flujo máximo

Formas de eliminar ciclos

Veremos tres formas de eliminar los ciclos en el ruteo:

I. Suponer que el tráfico de flujo entre los pares dentro de cada PID es perfecto. En otras palabras, dentro de cada PID los pares se comunican y transfieren todo el flujo posible. Si consideramos $\tilde{u}_i = \sum_{par \in PID_i} u_{par}$ y $\tilde{d}_i = \sum_{par \in PID_i} d_{par}$ la suma de las capacidades de ancho de banda de subida y bajada para todo los pares dentro del PID_i, suponiendo tráfico perfecto dentro de cada PID, podemos decir que $u_i = \max\{\tilde{u}_i - \tilde{d}_i, 0\}$ y

$d_i = \max\{\tilde{d}_i - \tilde{u}_i, 0\}$. Observemos que $\forall PID_i$ se cumple que $u_i = 0$ o $d_i = 0$. Con el supuesto de tráfico perfecto intra PID es imposible que un PID_i se mande flujo a él mismo ya que si tiene capacidad de subir flujo, $u_i > 0$, entonces su capacidad de descarga d_i es nula, por más detalles ver [50].

II. Si trabajamos sin el supuesto de tráfico intra $PIDs$ perfecto, es decir que un PID puede tener capacidad de subida y bajada no nulas simultáneamente, entonces podemos manipular el algoritmo de Ford-Fulkerson para que éste no elija ruteos que forman ciclos. La idea es llevar el ruteo al caso anterior, es decir que si un PID i sube flujo a la red, entonces para ese ruteo en particular se computa la capacidad de bajada del PID i, $d_i = 0$, con esta metodología un mismo PID nunca podrá enviarse flujo a si mismo.

III. Luego de tener la solución con todos los ruteos y sus respectivos flujos se detectan los ruteos que generan ciclos y procedemos a redirigir estos flujos no deseados. A modo de ejemplo consideremos la situación mostrada en la Figura 5.10. Claramente tenemos que redirigir el flujo que va desde B a A, $t_{B,A}$. Supongamos que existe un flujo no nulo que va desde B a C, $t_{B,C}$, entonces podemos intercambiar estos flujos obteniendo dos ruteos distintos, $\{A, B, C\}$ y $\{B, A\}$ como se muestra en la Figura 5.11. Hay que tener en cuenta que si el flujo a redirigir $t_{B,A}$ es mayor que $t_{B,C}$ entonces es necesario encontrar otro camino para redirigir el remanente de flujo $t^*_{A,B} = t_{A,B} - t_{A,C}$ que no se pudo redirigir.

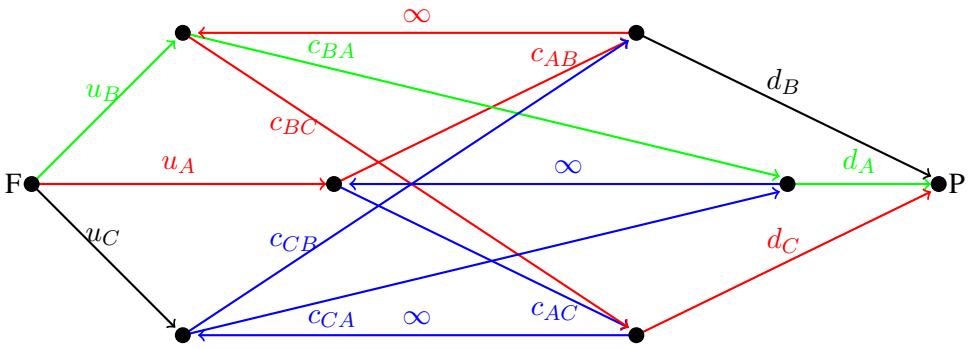


Figura 5.11: Adaptación para flujo máximo

Como vimos en este Capítulo es posible desarrollar algoritmos que encuentran en tiempo polinomial una solución tan cercana a la óptima como queramos pero

siempre suponiendo que sólo existe un contenido a transmitir, es decir $K = 1$. En el próximo Capítulo mostraremos una técnica para buscar la solución óptima del Problema General (PG) que contempla más de un contenido.

Capítulo 6

Algoritmos para el Problema General (PG)

En los capítulos anteriores estudiamos problemas para un único contenido. En la práctica las redes P2P son utilizadas para intercambiar múltiples contenidos y los ISPs se enfrentan al problema general de cómo repartir sus recursos entre todos ellos. Por tanto, un encare aislado para cada contenido no es óptimo para el Problema General.
En el Capítulo anterior demostramos que existen algoritmos que encuentran soluciones óptimas en tiempo polinomial para un solo contenido, $K = 1$, sin embargo, hasta el momento no hemos encontrado y desconocemos que exista un algoritmo que encuentre soluciones óptimas en tiempo polinomial para el Problema General, $K > 1$. Esto nos motiva a buscar soluciones en base a metaheurísticas.
En este capítulo se presentan las generalidades de la metaheurística seleccionada para resolver el Problema General (PG). Las bondades del método permiten desarrollar un algoritmo sencillo que se adapta a las características del problema retornando soluciones altamente satisfactorias.

En primera instancia se realiza una revisión de propuestas previas. El estudio posterior toma en consideración distintas opciones con el fin de desarrollar una metaheurística acorde al problema, con resultados superiores a implementaciones previas. Se discuten brevemente posibilidades de implementación de Algoritmos Genéticos y GRASP. La naturaleza del (PG) sugiere bondades de la utilización de GRASP, explotando las bondades del algoritmo "*MínimoEnlace*" introducido en el Capítulo anterior.

Finalmente se detalla la metaheurística desarrollada, basada en GRASP, en la

que se definen dos fases de mejoramiento de las soluciones (englobadas bajo el concepto de búsqueda local).

6.1 Enfoque Metaheurístico

Los primeros trabajos basados en métodos de optimización estocástica surgen en el año 1952, desarrollándose con el paso de los años nuevas ideas que dan lugar a metaheurísticas muy conocidas en la actualidad. Entre ellas algoritmos genéticos (1975) [26] y [44], recocido simulado (1983) [31], búsqueda tabú (1986) [23], colonia de hormigas (1991) [17, 18] y procedimientos de búsqueda greedy aleatorizado y adaptativo (1995), por sus siglas en inglés, GRASP (Greedy Randomized Adaptive Search Procedure) [7, 21, 34, 36, 41]. Esta última, elegida como método de resolución para el presente trabajo, es una metaheurística utilizada comúnmente para resolver problemas de optimización combinatoria de gran complejidad. Es conocido su buen desempeño en problemas NP-hard, además de su amplia aplicación en el área de las telecomunicaciones[40].

6.1.1 Algoritmo Basado en Simplex para Múltiples Contenidos

El Problema General ha sido tratado mediante diversos enfoques. Una primera propuesta calcula el objetivo de los pares mediante el Método Simplex [33], y posteriormente procura adaptar estos flujos óptimos para los pares a la red. Presenta el inconveniente de factibilidad, pues no va a ser posible adaptar el flujo deseado por los pares cuando las capacidades de los enlaces son muy reducidas. Si bien se han propuesto nuevas heurísticas que intentan solventar la factibilidad proveniente de la resolución con el método simplex (ya sea reduciendo la transferencia deseada por los pares para adaptar a las capacidades de la red, definiendo penalizaciones a los $PIDs$ más cargados y otras estrategias), éstas no presentan resultados muy satisfactorios ante instancias simples. Esto se debe básicamente a la complejidad de la definición de estrategias de penalización de $PIDs$ para capturar el problema de no adaptación de los flujos óptimos de los $PIDs$ a la red. A efectos ilustrativos explicaremos en detalle un Algoritmo basado en la atención primera al objetivo de los pares.
En el algoritmo mostrado en el Pseudo-Código mostrado en la Figura 6.1 se resuelve el problema de programación lineal correspondiente a hallar el deseo de los $PIDs$. Este es el de maximizar la transferencia neta (ecuación 6.1), sujeto a restricciones de ancho de banda de subida y bajada (ecuaciones 6.2 y 6.3) (lineas 1 a 3).

$$Obj(P2P)\begin{cases} \max \sum_{k}\sum_{i}\sum_{j\neq i} t_{ij}^{k} & (6.1) \\ s.a. & \\ \quad \forall k \sum_{j\neq i} t_{ij}^{k} \leq u_i \quad \forall PID_i & (6.2) \\ \quad \forall k \sum_{j\neq i} t_{ji}^{k} \leq d_i \quad \forall PID_i & (6.3) \\ \quad t_{ij}^{k} \geq 0 \quad \forall i \neq j & \end{cases}$$

Notemos que en el problema $Obj(P2P)$ de programación lineal no hay restricciones de capacidades de enlaces, es por este motivo que frecuentemente estos tráficos de salida no son factibles cuando se considera dichas restricciones. Es por este motivo que se procede a adaptar esta salida a la red con capacidades con un bloque de redirección de tráficos. El Algoritmo "*Simplex*" funciona como sigue: calcula el flujo máximo para cada contenido resolviendo el problema de programación lineal $ObjP2P$ (lineas 1 a 3)[1], luego se detectan enlaces que son excedidos por este flujo utilizando la función "*TestLID*" que retorna un 1 si el enlace se desborda y 0 en caso contrario (lineas 4 a 6). Por último para cada enlace desbordado se intenta redirigir el flujo que excede la capacidad del enlace por caminos de largo 2 utilizando la Función "*RedirectFlux*" cuyo Pseudo-Código se muestra en la Figura 6.2 (lineas 7 a 12).

La redirección propuesta en la Función "*RedirectFlux*" se basa en el reparto de carga en proporción con las capacidades utilizando nodos auxiliares (en otras palabras, redirigir utilizando caminos de largo 2). Su funcionamiento se detalla a continuación: el enlace desbordado lo denotamos como e_t y calculamos el flujo en exceso (linea 1), se inicializa "*totalcapacity*" en 0, es aquí donde se va a cargar cuál es el flujo potencial a ser redirigido (linea 2). Se determinan los nodos extremos, $nodo_1$ y $nodo_2$ del enlace e_t (linea 3). En las lineas (4 a 12) se procede a calcular cuanto flujo se puede redirigir a través de nodos auxiliares distintos a los nodos extremos del enlace e_t (linea 4), para esto se encuesta a todos los nodos candidatos a ser utilizados como auxiliares y se les calcula su capacidad remanente (lineas 7 y 8). Notemos que para cada nodo auxiliar vamos a tener que medir qué capacidad disponible tiene desde el nodo origen (denotado como $nodo_1$) al nodo auxiliar (i) (linea 7) y también qué capacidad disponible existe desde el nodo auxiliar (i) al

[1]Para resolver el problema de programación lineal $ObjP2P$Se utilizó la función *"solveSimplex"* de MatLab

nodo destino (denotado como $nodo_2$) (linea 8). Teniendo las capacidades disponibles asociadas al nodo auxiliar (i), verificamos que ambos enlaces ($nodo_1, i$) y ($i, nodo_2$) tengan capacidad disponible (linea 9), de ser así nos quedamos con la mínima capacidad disponible como magnitud para redirigir (linea 10) y se suma esta magnitud al total de flujo potencial a redirigir (linea 10).
Por último se redirige el flujo en exceso de forma proporcional a la capacidad disponible de cada uno de los nodos auxiliares utilizando la Función "*SendFlux*" (lineas 14 a 18).

1: **Para** $k = 1$ to K
2: $(t^*_{ij})_k \leftarrow SolveSimplex(ObjP2P)$
3: **Fin Para**
4: **Para** $t = 1$ to l
5: $boolean(t) \leftarrow TestLID(t^k_{ij}, e_t)$
6: **Fin Para**
7: **Mientras** $(sum(boolean)) \neq 0$)
8: **Para** $t = 1$ to l
9: **Si** $boolean(t) = 1$ **entonces**
10: $t^k_{e_{t1},e_{t2}} \leftarrow RedirectFlux(v_{e_t}, (t^*_{ij})_k)$
11: **Fin Si**
12: **Fin Para**
13: **Fin Mientras**

Figura 6.1: "Algoritmo Simplex"

El algoritmo basado en el Método Simplex genera frecuentemente soluciones que son imposible de adaptar en la red cuando se tienen en cuenta las capacidades de los enlaces. Básicamente luego que se consigue los flujos entre $PIDs$, estos se tienen que "acomodar" en la red real y esta tarea es muy difícil y muchas veces imposible ya que calcula transferencias entre $PIDs$ que no son factibles considerando las capacidades de los enlaces de la red. La dificultad mencionada anteriormente para el algoritmo basado en el Método Simplex nos llevan a introducir una metaheurística basada en GRASP que utiliza las bondades del algoritmo basado en Ford-Fulkerson introducido el Capítulo 5 para resolver el Problema 1.
Veremos en la Sección 6.1.3 las generalidades de la metodología GRASP y como la adaptamos para resolver el Problema General.

1: $reflux \leftarrow \sum_k \sum_i \sum_{j \neq i} t_{ij}^k \cdot I_{e_t}(i,j) - c_{e_t} \times (1 - b_{e_t})$
2: $totalcapacity \leftarrow 0$
3: $(node_1, node_2) \leftarrow e_t$
4: **Para** $i = 1$ to n, $i \neq node1$ and $i \neq node2$
5: $e_1 \leftarrow (node_1, node_i)$
6: $e_2 \leftarrow (node_i, node_2)$
7: $capacity_1 \leftarrow \sum_k \sum_i \sum_{j \neq i} t_{ij}^k \cdot I_{e_1}(i,j) - c_{e_1} \times (1 - b_{e_1})$
8: $capacity_2 \leftarrow \sum_k \sum_i \sum_{j \neq i} t_{ij}^k \cdot I_{e_2}(i,j) - c_{e_2} \times (1 - b_{e_2})$
9: **Si** $\max\{capacity_1, capacity_2\} < 0$ **entonces**
10: $capacitynode(i) = \min\{\mid capacity_1 \mid, \mid capacity_2 \mid\}$
11: $totalcapacity \leftarrow totalcapacity + capacitynode(i)$
12: **Fin Si**
13: **Fin Para**
14: **Para** $i = 1$ to n, $i \neq node1$ and $i \neq node2$
15: $e_1 \leftarrow (node_1, node_i)$
16: $e_2 \leftarrow (node_i, node_2)$
17: $i \leftarrow SendFlux(e_1, e_2, \frac{reflux * capacity}{total_{capacity}})$
18: **Fin Para**

Figura 6.2: Función "Redirección de Flujo"

6.1.2 Algoritmos Genéticos

Una alternativa considerada para resolver el problema de optimización fue la metaheurística algoritmos genéticos. Ésta se inspira en la evolución biológica, y en su aplicación se trata de que una población evolucione aplicándole acciones aleatorias de selección, cruzamiento y mutación.
Por población se designa a un conjunto de soluciones factibles y es la que se modifica en las distintas iteraciones del algoritmo con fin de aproximarse a la solución del problema. La selección es una acción en la que se eligen los individuos que serán cruzados para obtener individuos que integren la siguiente generación, dando mayor probabilidad de selección a los individuos más aptos, es decir con mejores valores de la función objetivo.
Los nuevos integrantes de la población se obtienen por medio de cruzamiento o mutación. En el primer caso se opera combinando dos integrantes de la población (padres) para generar dos descendientes. En la mutación se generan cambios aleatorios sobre algún integrante de la población dando lugar a uno nuevo. Realizados los cambios se seleccionan de la población los individuos que permanecerán en la siguiente generación. Para ver en detalle la metodología, referimos al lector a los

trabajos [26] y [44]. Una aplicación de Algoritmos Genéticos para P4P es introducido en el Trabajo realizado por la Ing. Claudia Rostagnol [43] ya mencionado en el Capítulo 2.
En la adaptación de algoritmos genéticos a este trabajo, la población es conformada por un subconjunto de soluciones factibles al Problema General (PG). En particular la población inicial se integra por un conjunto de soluciones selectas considerando aplicaciones de Greedy. Sin embargo adaptación en las instancias de cruzamiento y mutación pueden ser muy engorrosas e incluso no generar soluciones factibles. Es necesario definir una forma de cruzamiento de soluciones factibles partiendo de dos matrices de tráfico pertenecientes a distintas instancias y combinarlas para generar una nueva solución factible.
En virtud de la complejidad en la costumización al PG de adaptar esta metaheurística al (PG) y la gran simplicidad de la técnica GRASP se decide priorizar la implementación de la segunda opción sobre la primera. Si bien no se descarta que los algoritmos genéticos obtengan buenos resultados su mayor complejidad en la costumización hacen que no se los considere en una primera instancia.

6.1.3 GRASP

El método GRASP [12, 21, 22] es iterativo y consta de dos fases que se adaptan fácilmente en la aplicación: una fase constructiva y una de búsqueda local. En la primera se obtienen en forma golosa adaptativa aleatorizada soluciones factibles al problema y en la segunda se tratan de mejorar los resultados obtenidos en la fase inicial. En cada iteración se aplican ambas fases, por lo que GRASP es un método multiarranque. La mejor solución obtenida en el total de iteraciones es la solución al problema. Además el método es adaptativo dado que en la etapa de construcción se genera una solución parcial que depende de lo obtenido en los pasos previos de esa iteración. En la Figura 6.3 se muestra el Pseudo-Código del método.

En la fase inicial, mostrada en el Pseudo-Código de la Figura 6.4, se construye una solución factible tomando de manera iterativa elementos que generan soluciones parciales. El criterio de elección de los elementos a integrar durante la construcción, tal como lo sugiere el nombre de la heurística, es incorporando aquellos elementos que generen el mejor cambio en la función objetivo. Una particularidad de la forma de construcción es que en cada paso sólo se agregan elementos, y lo incorporado en iteraciones previas no se reordena sino que es inamovible.
Este procedimiento permite generar soluciones factibles que pueden ser o no óptimos locales.

Entradas:
Tamaño de la lista: $largo$
Máximo de iteraciones: $Iter$
Salidas:
Mejor solución: $LaMejorSolucion$

1: **Para** $i = 1$ a $Iter$
2: Solucion Inicial← Etapa Constructiva Greedy Aleatorizada(tamaño);
3: $SolucionBusqLocal \leftarrow BL(SolucionInicial)$;
4: **Si** $SolucionBusqLocal$ es mejor que $LaMejorSolucion$ **entonces**
5: $LaMejorSolucion = SolucionBusqLocal$
6: **Fin Si**
7: **Fin Para**
8: **Retornar** $LaMejorSolucion$

Figura 6.3: Pseudo-código GRASP

En la práctica la forma de incorporar aleatoriedad en esta fase greedy es la creación de una lista con los mejores candidatos de los que se elige uno aleatoriamente y se lo integra a la solución parcial. La ventaja del algoritmo greedy aleatorio respecto al greedy original es la posibilidad de recorrer una parte mayor del espacio de soluciones factibles. Mientras greedy, por ser determinístico asocia una única solución a cada conjunto de datos, la aleatorización permite recorrer un espectro amplio de soluciones factibles.

Por otro lado en la fase de búsqueda local, mostrada en el Pseudo-Código de la Figura 6.5, se realizan reemplazos en la solución obtenida en la fase inicial procurando mejoras de la función objetivo. Para ello se implementa un método iterativo que se aplica mientras se obtienen mejoras en la solución, buscando en cada instancia soluciones vecinas a la actual.

Entradas:
Tamaño de la lista: $largo$
Salidas:
Solución Factible: $Solucion$

1: $Solucion \leftarrow \emptyset$;
2: **Mientras** NoEsFactible(Solucion)
3: $Lista \leftarrow ListaDeCandidatos$;
4: Evaluar la función G de costo para elemento en Lista
5: $a \leftarrow$ elijo aleatoriamente un elemento de $Lista$
6: $Solucion \leftarrow Solucion \bigcup \{a\}$
7: Actualizar $Lista$
8: **Fin Mientras**
9: **Retornar** $Solucion$

Figura 6.4: Pseudo-Código Etapa Constructiva Greedy Aleatorizada

Dada una solución S se define la vecindad de S como sigue:

Definición 6.1.1 *Estructura de Vecindad*
Sea Ω un conjunto de soluciones factibles de un problema de optimización combinatoria $\max_{S\in\Omega}\{f(S)\}$. Una estructura de vecindad es una colección $\{\mathcal{N}(S)\}_{S\in\Omega}$ de subconjuntos $\mathcal{N}(S) \subseteq \Omega$ tales que:

1) $\forall S, T \in \Omega, \exists S = S_0, \ldots, S_k = T : S_i \in \mathcal{N}(S_{i-1}) \quad \forall i = 1, \ldots, k$

2) Para cada $S \in \Omega$ es posible decidir en tiempo polinomial si existe $T \in \mathcal{N}(S) : f(T) > f(S)$, y hallar esa mejor solución factible, si existe.

Se define análogamente para el caso de un problema de minimización.

La vecindad de una solución puede definirse considerando *best-improvement*, estrategia en donde se elige el vecino que genera la mayor mejora en la función, o *first-improvement* en donde se elige el primer vecino encontrado que logra una mejora.

Los resultados de la metaheurística GRASP dependen de las soluciones alcanzadas en la fase inicial, del mecanismo de búsqueda y la estructura de vecindad considerada en la segunda fase. La potencialidad de la herramienta redunda en la correcta elección de estos tres conceptos centrales y de su correcta implementación. Por más detalles de GRASP, referimos al lector a los trabajos [7, 12, 21, 22, 34].

Entradas:
Solución Factible $Semilla$
Salidas:
Mejor Solucion: $Solucion$

1: Solucion $\leftarrow$ Semilla
2: **Mientras** No Alcanzo Optimo Local
3: $V \leftarrow$ Elijo un vecino $\in Vecindad(Solucion)$;
4: **Si** V es mejor que Solucion entonces **entonces**
5: Solucion $\leftarrow V$
6: **Fin Si**
7: **Fin Mientras**
8: **Retornar** $Solucion$

Figura 6.5: Pseudo-Código Búsqueda Local"

Presentadas las generalidades del método continuamos con la adaptación al problema de interés. Para ello puntualizamos algunos aspectos respecto a las características del Problema General:

- La topología asociada al problema es un grafo completo en donde los nodos representan a los $PIDs$ y las aristas los enlaces de la red.
- Es un problema multiobjetivo en donde hay que optimizar el tráfico entre los $PIDs$ para cada uno de los K contenidos, minimizando los costos de los ISP vía reducción del tráfico en los enlaces costosos.
- El objetivo de los $PIDs$ prioriza sobre el objetivo de los ISP.
- El uso de los enlaces es una medida relativa que indica el porcentaje de ocupación de la capacidad del enlace.
- El tráfico es una medida absoluta que indica cuánto flujo se envía por los enlaces.

6.2 GRASP: algoritmo detallado para resolver el Problema General (PG)

Recordemos que la formulación matemática del problema bi-nivel "Problema General" es la siguiente:

$$PG \quad \begin{cases} \min \max_{e \in E} \{b_e + \sum_k \sum_i \sum_{j \neq i} \frac{t_{ij}^k \times I_e(i,j)}{c_e}\} \\ s.a. \\ \quad \max \sum_i \sum_{j \neq i} t_{ij}^k, \quad \forall k \in K \\ \quad s.a. \\ \quad\quad \sum_{j \neq i} t_{ij}^k \leq u_i^k \quad \forall PID_i \\ \quad\quad \sum_{j \neq i} t_{ji}^k \leq d_i^k \quad \forall PID_i \\ \quad\quad t_{ij}^k \geq 0 \quad \forall i \neq j \\ \quad\quad \sum_k \sum_i \sum_{j \neq i} t_{i,j}^k \times I_e(i,j) < v_e \end{cases} \tag{6.4}$$

La adaptación de la heurística GRASP al problema multiobjetivo PG, dedica la fase de construcción a maximizar el tráfico entre los $PIDs$ y la etapa de búsqueda local a minimizar el máximo porcentaje de uso de los enlaces. Como mostramos en el Capítulo 5 el algoritmo "*MínimoEnlace*" encuentra la solución óptima para un contenido, cuando consideramos $K > 1$ no basta con hacer el mejor ruteo para cada contenido independientemente, lo mejor que podemos lograr es hacer el "mejor" ruteo para el contenido k dado que ya ruteamos los $k-1$ contenidos anteriores.

Recordamos a continuación los parámetros del Problema General.
Dado un grafo $G = (V, E)$ completo con $|V| = n$ se definen:

- C: Matriz de capacidad de los enlaces e, $e \in E$, $C \in M_{n,n}(\mathbb{R})$
- T: Matriz de tráfico de los $PIDs$, donde $T_{i,j}$ es el tráfico entre los $PIDs$ $i, j \in V$, $T \in M_{n,n}(\mathbb{R})$
- B: Matriz de tráfico de fondo de los enlaces, $B \in M_{n,n}(\mathbb{R})$
- U^k: Vector de capacidad de subida para el contenido $k = 1, \ldots, K$ de los $PIDs$, $U^k \in \mathbb{R}^n$
- D^k: Vector de capacidad de bajada para el contenido $k = 1, \ldots, K$ de los $PIDs$, $D^k \in \mathbb{R}^n$

6.2.1 Fase 1 (Construcción de una solución factible)

Para la Fase de construcción del GRASP desarrollamos el algoritmo llamado "*RandomList*". El algoritmo selecciona aleatoriamente un contenido a la vez sin reposición en cada iteración y aplica el algoritmo "*MínimoEnlace*" para encontrar el conjunto de caminos y sus respectivas magnitudes para el contenido sorteado. Luego que se rutea el contenido sorteado se determina el conjunto de caminos y

sus magnitudes los cuales quedarán fijos a la hora de sortear el próximo contenido. Los contenidos son sorteados aleatoriamente según un vector de "criterio" p, tal que p_k es el score de prioridad del contenido k. De esta forma se priorizará los contenidos según alguna variable de interés (ej. tamaño del contenido o importancia del mismo). El algoritmo recibe como entradas: el vector de prioridad p, las matrices U^K y D^K que almacenan los anchos de banda de subida y bajada para cada PID y cada uno de los K contenidos, por último, las matrices de capacidades y tráfico de fondo C y B respectivamente.
En la matriz H se almacenará todo el flujo de cada enlace debido a tráfico P2P el cuál se va anexando en cada iteración.
El algoritmo "*RandomList*" se presenta en la Figura 6.6 el Pseudo-Código, sus pasos detallados son: se inicializan el conjunto de caminos y magnitudes P y m respectivamente como conjuntos vacíos (linea 1) y la matriz H con valores nulos (linea 2), luego mientras existan contenidos para atender y recursos en la red (enlaces no saturados) se sortea un contenido k con probabilidad p_k (linea 3) y se rutea éste utilizando el algoritmo "*MínimoEnlace*" (linea 4). Luego de tener el ruteo $Paths$ para el $k-$ ésimo contenido y sus respectivas magnitudes $magnitudes$ se introducen en los conjuntos P y m respectivamente (lineas 7 y 8). El flujo de cada enlace para cada contenido se introduce en la matriz H (linea 5). Por último determinamos $p_k = 0$ para que el contenido k no se vuelva a elegir y normalizamos el vector p siempre cumpliendo que $\sum_k p_k = 1$ (lineas 9 y 10).
Notemos que luego que se determina un ruteo para un contenido k este se ingresa en H y queda fijo e inamovible, es decir que no se altera al determinar el ruteo de contenidos posteriores. El algoritmo devuelve el conjunto de caminos orientados P para cada contenido y las correspondientes magnitudes m asociadas a cada camino en P.

6.2.2 Fase 2 (Búsqueda Local)

Luego de obtener una solución en la Fase de construcción es de interés tratar de mejorarla. Para esto introducimos una búsqueda local enfocada en minimizar el enlace más congestionado sin disminuir el tráfico total en la red (mejorando o igualando el tráfico entre los $PIDs$).
Conociendo el conjunto de caminos P y magnitudes m para cada contenido, en esta Fase la heurística se basa en minimizar el uso de los enlaces redirigiendo flujo, expresión que se resume en la Ecuación 6.5:

$$\min \max_{e \in E} \{ b_e + \sum_k \sum_i \sum_{j \neq i} \frac{t_{ij}^k \times I_e(i,j)}{c_e} \} \tag{6.5}$$

Para lograr tal cometido desarrollamos el algoritmo "*BL*" el cual se muestra

Entradas:
Matriz de anchos de banda de subida: U^k
Matriz de anchos de banda de bajada: D^k
Matriz de capacidades: C
Matriz de tráficos de fondo: B
Vector de probabilidades a priori: p
Salidas:
Best Solution: P, m

1: $P \leftarrow \emptyset$; $m \leftarrow \emptyset$
2: $H \leftarrow zeros_{n \times n}$
3: **Mientras** $\sum_i p_i > 0$ y $C \geq 0$
4: $k \leftarrow RandomAccordingWith(p)$
5: $(Path, magnitudes) \leftarrow MinimoEnlace(U^k, D^k, C, B + H, \epsilon)$
6: $H \leftarrow AddPaths(Path, magnitudes, H)$
7: $P \leftarrow P \cup Path$
8: $m \leftarrow P \cup magnitudes$
9: $p_k \leftarrow 0$
10: $Normalize(p)$
11: **Fin Mientras**

Figura 6.6: Algoritmo "RandomList"

el Pseudo-Código en la Figura 6.7. El cual detecta el enlace más congestionado y produce un ruteo distinto al original encontrado en la Fase 1 para cada contenido que pasa por el enlace saturado. El algoritmo funciona como sigue: inicializa e_{sol} con el valor 1, P_{sol} y m_{sol} con el conjunto de caminos (P) y magnitudes (m) obtenidos en la Fase 1 (algoritmo "*RandomList*") (linea 2). Luego, itera hasta que no se produzcan mejoras o se alcance un número máximo de iteraciones (linea 3 a 15). En cada iteración detecta el enlace más congestionado e_{sat} utilizando la función "*MaximoEnlace*" (linea 4). Luego toma de a un contenido que pasa por ese enlace (linea 5) y trata de encontrar un ruteo alternativo para ese contenido (linea 7) (utilizando el algoritmo "*MínimoEnlace*"), no permitiendo que se utilice el enlace e_{sat} (linea 6). Si la solución encontrada consigue una reducción de la utilización del enlace más congestionado sin perder tráfico de flujo para el contenido y además mejore soluciones anteriores (linea 8), entonces se toma este nuevo ruteo como nueva solución (lineas 9 a 12). Por último, luego que se evaluó para todos los contenidos que pasan por e_{sat} se incrementa en 1 el contador j (linea 15).
Observemos que el algoritmo va a detectar cual es el contenido que genera una

mayor reducción del enlace e_{sat} sin perder transferencia de flujo entre $PIDs$.

Entradas:
Matriz de anchos de banda de subida: U^k
Matriz de anchos de banda de bajada: D^k
Matriz de capacidades: C
Matriz de tráficos de fondo: B
Conjunto de caminos para cada contenido: P
magnitudes asociadas a los caminos en P: m
Salidas:
Mejor solución: $P_{sol}, m_{sol}, e_{sol}$

1: $j \leftarrow 0$
2: $e_{sol} \leftarrow 1$, $P_{sol} \leftarrow P$, $m_{sol} \leftarrow m$
3: **Mientras** (haya mejoras) y $j < iter$
4: $e_{sat} \leftarrow MaximoEnlace(P_{sol}, m_{sol})$
5: **Para** i in contenidos que pasan por e_{sat}
6: $C \leftarrow C_{e_{sat}} = 0$
7: $(Path, magnitudes)$ = $MinimoEnlace(U^i, D^i, C, B + (H - H_i), \epsilon)$
8: $e_i \leftarrow MaximoEnlace(Path, magnitudes)$
9: **Si** ($e_i < e_{sat}$) y ($e_i < e_{sol}$) y no disminuya el flujo del contenido k **entonces**
10: $P_{sol} \leftarrow Path$
11: $m_{sol} \leftarrow magnitudes$
12: $e_{sol} \leftarrow e_i$
13: **Fin Si**
14: **Fin Para**
15: $j \leftarrow j + 1$
16: **Fin Mientras**

Figura 6.7: Algoritmo “BL”: Búsqueda Local

Capítulo 7

Resultados

En este Capítulo mostraremos resultados obtenidos de la aplicación de distintos algoritmos propuestos para resolver el Problema 1, en particular se consideran los algoritmos GOTEO y *Goloso Simple* introducidos en el Capítulo 4, *Algoritmo Genéticos* [43] y el Algoritmo basado en Ford-Fulkerson introducido en el Capítulo 5. También se evalúa el desempeño del Algoritmo basado en GRASP introducido en el Capítulo 6 para resolver el Problema General (PG) frente a una cota conocida de éste. Se describe también cómo fueron obtenidas las instancias de prueba para el P1 y el PG así como las características de los distintos algoritmos implementados.

7.1 Generación de Instancias para el Problema 1 (P1) basadas en datos reales

En esta Sección se presenta la generación de instancias realizadas por el Ing. Matías Barrios para el Problema 1. Gracias a datos proporcionados por ANTEL fue posible obtener datos con cierta noción de realidad para el llamado P1 los cuales son de gran utilidad para ver el desempeño de los distintos algoritmos presentados. Se construyeron 12 instancias distintas considerando mediciones reales extraídas de un streaming de video de un partido de fútbol transmitido en el 2006 siguiendo la siguiente metodología:

1. De este streaming se tiene información de pares que se conectaron y desconectaron al streaming de video en una ventana de tiempo de aproximadamente 2 horas. De esta ventana se consideraron intervalos pequeños de 10 a 15 minutos aproximadamente de pares que se conectan y desconectan para ver el partido, recolectando en cada intervalo alrededor de 200 pares de los

cuales se conoce su IP y por tanto es posible realizar distintos mapeos a $PIDs$ (en nuestro caso consideramos $PIDs$ como países).

2. Para cada uno de los pares se calculó su ancho de banda de bajada utilizado para la aplicación de este streaming en particular. Entonces sumando los anchos de banda de bajada de los pares dentro de cada País se estimó el ancho de banda de bajada de cada PID de la red (Países).

3. Por lo general el ancho de banda de bajada de cada par es entre 2 y 4 veces más que el ancho de banda de subida, por lo cual se decidió tomar como estimación de los anchos de banda de subida la cuarta parte de la magnitud del ancho de bajada más un ruido blanco, como se muestra en la Ecuación 7.1:

$$u_i = \frac{d_i}{4} + \varepsilon \quad \forall i \in PIDs \tag{7.1}$$

4. La matriz de capacidades C se tomó totalmente aleatoria y simétrica, salvo para el PID Uruguay que tiene enlaces entrantes con capacidades mayores a los salientes.

5. Por último la matriz de tráfico de fondo B, se definió con valores positivos para los enlaces que tienen como origen o destino al PID Uruguay y cero para el resto. En las 12 instancias el tráfico de fondo para los enlaces que vinculan a Uruguay supera el 94% de utilización.

De esta forma se pudieron construir 12 instancias las cuales poseen un cierto grado de realidad, en el sentido que los pares están bien asignados a sus respectivos países, los anchos de banda de bajada fueron obtenidos considerando descargas reales y los anchos de banda de subida son coherentes con el conocimiento empírico que se tiene de la relación entre ancho de banda de subida y bajada. Por otro lado nada podemos decir de la coherencia de las capacidades y menos de los tráficos de fondo.
El número de $PIDs$ considerado dentro de cada instancia varía en un rango entre 17 y 75 $PIDs$.

Observación: 7.1.1 *Es importante notar que estas 12 instancias poseen características particulares que se asemejan más a instancias para el Problema 2 que para el Problema 1 ya que sólo se consideró tráfico de fondo mayor que 0 en enlaces que involucran a sólo un PID (Uruguay). De todas formas como ya vimos en el Capítulo 4 el P2 es un caso particular del P1, por lo cual estas instancias son válidas para el Problema 1. Además le da más relevancia a las comparaciones de los distintos algoritmos con el Algoritmo de "*GOTEO*" el cual fue diseñado para este tipo de instancias.*

Observación: 7.1.2 *Las 12 instancias construidas tienen la particularidad de ser "pesimistas" (ver Definición en 4.1.6). En todos los casos el PID pesimista es Uruguay debido al gran porcentaje de tráfico de fondo que tienen sus enlaces (en todas las instancias mayores a 94%) generando que la capacidad remanente para tráfico P2P quede muy limitada. En el Cuadro 7.1 se muestran algunas características de cada instancia.*

Instancias	Nro. $PIDs$	% Tráfico de Fondo
1	21	0.95
2	17	0.94
3	34	0.97
4	23	0.95
5	40	0.97
6	37	0.97
7	54	0.98
8	36	0.97
9	43	0.98
10	75	0.99
11	43	0.98
12	59	0.98

Cuadro 7.1: Número de $PIDs$ y porcentaje de tráfico de fondo de los enlaces que vinculan al PID Uruguay

Ahora bien, para los problemas que necesitan mayor información de la red como es el (PG), no contamos con información relevante. Esto nos obliga a determinar instancias total o parcialmente aleatorias.

7.2 Implementación de distintos algoritmos para el Problema 1

En esta Sección vamos a introducir las características de los algoritmos utilizados para resolver el Problema 1 considerando las 12 instancias previamente definidas. En particular tenemos resultados de 4 metodologías distintas, el *Algoritmo Goloso Simple* y el Algoritmo de GOTEO ambos introducidos en el Capítulo 4, *Algoritmos Genéticos* mencionado en el Capítulo 2 y por último el *Algoritmo basado en Ford-Fulkerson* propuesto en el Capítulo 5.

7.2.1 Generalidades de los algoritmos propuestos e implementados

Algoritmo Goloso Simple: en esta Sección se validan empíricamente las limitaciones del algoritmo detallado en la Sección 4.4. En pocas palabras el algoritmo sortea un orden de $PIDs$ para que estos suban flujo a la red. Cuando a un PID le toca su turno de transferir flujo, se mide cuánto puede subir por enlaces directos (caminos de largo 1) considerando su ancho de banda de subida y los anchos de banda de bajada de los otros $PIDs$ además de las capacidades de sus enlaces. Una vez conocida la magnitud del flujo que puede subir a la red se introduce en ésta.

Algoritmo de GOTEO: es de gran interés ver el desempeño del Algoritmo de GOTEO, introducido en la Sección 4.3.2, ya que éste fue propuesto para instancias con las características que poseen las 12 instancias construidas (sólo los enlaces asociados a un PID destacado poseen tráfico de fondo mayor que 0).

En líneas generales el algoritmo distingue entre el PID especial (Uruguay) y el resto de los $PIDs$ limitando el flujo que éste puede subir. Esta idea se sustenta en el hecho de que si el PID Uruguay sube todo el flujo que puede necesariamente taparía todos sus enlaces. Por lo cual se utiliza un parámetro de penalización llamado "*uso*" para el PID distinguido, el cual le permite subir una proporción de lo que realmente puede subir. El algoritmo encuesta a cada PID, determina cuánto flujo puede subir a la red y los sube utilizando la técnica waterfilling. Por último, con el fin de descongestionar los enlaces más saturados, se aplica un bloque de redirección, el cual detecta el enlace más congestionado y redirige parte del flujo del enlace por medio de uno a más caminos de largo dos.

Algoritmos Genéticos: esta metodología para resolver el Problema 1 se basa en la idea de la aplicación de la heurística Algoritmos Genéticos. Se parte de una población inicial (en este caso soluciones factibles del P1) y se itera el algoritmo un número considerable de veces. En cada iteración se eligen 2 soluciones de la población (llamadas padres) para determinar dos nuevas soluciones (hijos) a este proceso se lo denomina cruzamiento. Según una cierta probabilidad p_m se puede "alterar" las soluciones hijas manipulando "levemente" alguna de las nuevas soluciones hijas. La diferencia más significativa que presenta esta metodología con el resto de los algoritmos propuestos es que aquí no prima el objetivo de los $PIDs$ frente al de los ISPs, es por esto que no se intenta primero maximizar el flujo que pueden subir los $PIDs$ para luego minimizar el enlace más congestionado, sino lo que se obtiene es una estimación del "*frente de Pareto*" [47], en otras palabras, soluciones que

no estén dominadas por ninguna otra para alguna de las funciones objetivos. Los resultados mostrados en el trabajo realizado por la Ing. Claudia Rostagnol [43] no son comparables con los nuestros ya que se cuenta sólo con información de algunas métricas que indican la bondad de la estimación de los frentes de Pareto para cada instancia.

Algoritmo basado en Ford-Fulkerson: este algoritmo fue introducido en detalle en la Sección 5.4. Este resuelve el Problema 1 en dos etapas:

1. Calcula la transferencia máxima para los $PIDs$ por medio del algoritmo de Ford-Fulkerson aplicado en la red auxiliar introducida en la Sección 5.1.2.
2. Luego aplicando el Teorema de Bolzano, encuentra la utilización mínima $\alpha_{opt} \in [b_{min}, 1]$ que asegura obtener la misma transferencia alcanzada en el punto anterior. Este algoritmo es el que más se diferencia a la lógica planteada por el Algoritmo Genético ya que prioriza el objetivo de los $PIDs$ asegurándose que éstos podrán alcanzar su mejor transferencia respetando los recursos de la red. Por otro lado, se podría adaptar el algoritmo basado en Ford-Fulkerson de manera que también estime el frente de Pareto, simplemente bastaría con calcular los flujos máximos entre $PIDs$ limitando la utilización de los enlaces a valores $(\alpha_1, \ldots, \alpha_H)$, $\alpha_i \in [b_{max}, 1] \forall \alpha_h = 1, \ldots, H$. De esta forma sabremos cuánto es la máxima transferencia entre $PIDs$ para cada uno de los umbrales de utilización α_i y así con estos puntos podremos estimar el frente de Pareto.

En la siguiente Sección se presentan los resultados obtenidos por cada uno de los algoritmos anteriormente mencionados y sus comparaciones.

7.3 Comparación de resultados

En primer lugar mostraremos en el Cuadro 7.2 el desempeño del Algoritmo Goloso simple (que denotaremos como GS), GOTEO y el basado en Ford-Fulkerson (que denotaremos como FF) para las 12 instancias generadas. En este cuadro se muestran el total de $PIDs$ considerados en cada instancia (Nro. $PIDs$), el total de flujo neto transferido (Objetivo $PIDs$) alcanzado por cada uno de los algoritmos y la saturación máxima detectada en el enlace más congestionado (Objetivo ISPs) alcanzada por cada uno de los algoritmos.

		GS		GOTEO		*FF*	
Instancias	Nro. $PIDs$	Obj. $PIDs$	Obj. ISPs	Obj. $PIDs$	Obj. ISPs	Obj. $PIDs$	Obj. ISPs
1	21	13211	1.000	13101	0.9950	13211	1.000
2	17	11574	1.000	11447	0.9963	11574	1.000
3	34	30557	1.000	29960	0.9973	33556	1.000
4	23	35216	1.000	32048	0.9909	35216	1.000
5	40	72495	1.000	64409	0.9931	74148	1.000
6	37	42352	1.000	40512	0.9950	42352	1.000
7	54	69489	1.000	66583	0.9970	69489	1.000
8	36	28467	1.000	27625	0.9957	28467	1.000
9	43	34692	1.000	34692	1.000	34692	1.000
10	75	101095	1.000	101095	1.000	101880	1.000
11	43	38804	1.000	38804	1.000	38804	1.000
12	59	120360	1.000	110544	0.9955	121436	1.000

Cuadro 7.2: Resultados obtenidos para los algoritmos Golos Simple, GOTEO y el basado en Ford-Fulkerson

En el Cuadro 7.2 podemos observar que los algoritmos FF y GS saturan en su totalidad algún enlace de la red en todas las instancias, lo cual era de esperarse debido a que todas las instancias cumplen con la condición de ser "pesimistas" a causa de los altos porcentajes de tráfico de fondo que poseen los enlaces que vinculan al PID especial (Uruguay). De todas formas el Algoritmo FF logra una mayor transferencia neta en las instancias 3, 5, 10 y 12.
El Algoritmo de GOTEO logra transferencias netas menores pero saturando menos los enlaces de la red. El hecho de que el Algoritmo de GOTEO logre una menor saturación se justifica por la penalización impuesta al PID especial (Uruguay).
Los tres algoritmos igualan su transferencia neta sólo en los casos que todos saturan sus enlaces a excepción de la instancia 10 en la cual los tres algoritmos saturan enlaces pero el Algoritmo basado en Ford-Fulkerson logra una transferencia neta mayor a los otros.
Para evaluar realmente qué algoritmo es mejor decidimos ejecutar nuevamente el Algoritmo FF restringiendo la utilización de los enlaces a las mismas obtenidas por el Algoritmo de GOTEO, mostradas en el Cuadro 7.2, de esta forma podemos evaluar si efectivamente el Algoritmo FF obtiene mayores transferencias de flujo para estos niveles de utilización.
En el Cuadro 7.3 mostramos las transferencias netas logradas por cada algoritmo limitando la utilización máxima de los enlaces a la alcanzada por el Algoritmo de GOTEO. En las columnas (GS, GOTEO y FF) del Cuadro 7.3 se pueden ver las magnitudes del flujo neto alcanzado por cada algoritmo para cada una de las saturaciones máximas impuestas mostradas en la columna (Saturación GOTEO). Además en las columnas (% Inc. FF vs GOTEO y % Inc. FF vs GS) se muestran

los incrementos que se lograron en la transferencia de flujo al aplicar el Algoritmo FF en relación a los otros dos.
En efecto, por los resultados del Cuadro 7.3 podemos concluir que el Algoritmo FF logra mayores transferencias netas que el de GOTEO en 4 de las 12 instancias y logra mejores transferencias que el Algoritmo GS en 10 instancias. Recordemos que los algoritmos GS y GOTEO están diseñados para transferir flujo sólo por caminos de largo 1, esto explica las diferencias en el tráfico neto. Para las 4 instancias en las que el Algoritmo FF logró una mayor transferencia con respecto al de GOTEO fue debido a que encontró caminos de largo 2 o más para subir flujo a la red, ahora de las 10 instancias en las que el GS logró menores transferencias no podemos decir lo mismo, de hecho verificamos los ruteos para todas las instancias y se constató que sólo se generaron caminos de largo mayor a 1 para las 4 instancias en las que existen diferencias entre FF y GOTEO. Quedan en evidencia las debilidades del Algoritmo GS ya mencionadas en el Caítulo 4:

- malas decisiones de ruteo,
- la saturación de enlaces, y
- sĺo considerar caminos de largo 1.

Cabe destacar que si bien existen diferencias, éstas no son muy importantes, en el mejor de los casos el Algoritmo FF logró una transferencia del 1,2017% más que los algoritmos de GOTEO y GS. Esto no quiere decir que siempre ante cualquier instancia los rendimientos de los tres algoritmos sean similares, es de esperarse que para instancias en las cuales sea necesario generar ruteos por caminos de largo mayores que 1, el Algoritmo FF encuentre soluciones significativamente mejores que el de GOTEO, mientras que si estos ruteos no son necesarios las soluciones obtenidas por ambos serán muy parecidas o idénticas.
En particular la mayoría de las instancias aquí presentadas no necesitan ruteos por caminos mayores que 1 y en las que sí, es sólo para introducir una cantidad muy baja de flujo.

7.4 Utilización de la solución en GoalBit

Una vez obtenida una solución al problema de optimización es de gran interés su utilización en una plataforma real. A continuación se explicará la implementación de soluciones del Problema General en una plataforma real denominada GoalBit [24] la cual ya fue comentada en el Capítulo 2. En GoalBit el modelo P4P será

Instancia	GS	GOTEO	FF	Saturación GOTEO	% Inc. FF vs GOTEO	% Inc. FF vs GS
1	13096	13101	13101	0.9950	0.0000	0.0382
2	11442	11447	11447	0.9963	0.0000	0.0437
3	29949	29960	29960	0.9973	0.0000	0.0367
4	31916	32048	32048	0.9909	0.0000	0.4136
5	64423	64409	65183	0.9931	1.2017	1.1797
6	40465	40512	40512	0.9950	0.0000	0.1161
7	66549	66583	66600	0.9970	0.0255	0.0766
8	27600	27625	27625	0.9957	0.0000	0.0906
9	34692	34692	34692	1.0000	0.0000	0.0000
10	101095	101095	101880	1.0000	0.7765	0.7765
11	38804	38804	38804	1.0000	0.0000	0.0000
12	110577	110544	110855	0.9955	0.2813	0.2514

Cuadro 7.3: Desempeño de los algoritmos: Goloso simple, de GOTEO y el basado en Ford-Fulkerson ante distintas instancias fijando la utilización alcanzada por el Algoritmo de GOTEO.

ejecutado en el appTraker[1], el cual deberá de tener toda la información necesaria para implementar la heurística que encuentra la solución al Problema General (en nuestro caso será al Algoritmo GRASP propuesto para resolver el Problema General). Cada cierto tiempo el appTraker ejecutará el Algoritmo GRASP y actualizará las transferencias óptimas $t^k_{i,j}$ que este determine. Con estos valores el appTraker calculará los siguientes pesos:

$$w^k_{i,j} = \frac{t^k_{i,j}}{\sum_{j \neq i} t^k_{i,j}} \, \forall PID_i, \, \forall PID_j \; j \neq i \tag{7.2}$$

Donde $w^k_{i,j}$ representa el porcentaje de todo el tráfico saliente desde el PID_i al PID_j. Con estos pesos se intenta sesgar el tráfico entre $PIDs$ para que se obtengan las magnitudes deseadas $t^k_{i,j}$. En cada ingreso de un par a la red, el appTracker debe decidir la lista de pares que va a asignarle para converger en probabilidad al ruteo óptimo. Supongamos que un par dentro del PID_i elige a miembros de su lista con los cuales comunicarse al azar con distribución uniforme. Entonces, la manera óptima del appTracker para generar la lista de pares consiste en determinar dentro de esta una proporción $w^k_{i,j}$ de pares que pertenezcan al PID_j.

A continuación mostraremos resultados obtenidos mediante emulaciones realizadas en la red GoalBit.

[1]Para GoalBit no se implementó un iTracker, siendo el appTracker el encargado de todas las funciones

7.4.1 Emulaciones en GoalBit

Estas emulaciones se realizaron con el fin de comparar la metodología P4P para seleccionar un enjambre de pares versus una selección de pares aleatoria. Las emulaciones se hicieron antes de la creación del Algoritmo basado en Ford-Fulkerson introducido en este trabajo el cual hubiese sido nuestro mejor candidato para estas comparaciones dada su simplicidad y buenas propiedades para el tráfico de un contenido. Los resultados que mostraremos son producto de aplicar la metodología P4P basada en el Algoritmo Simplex el cual fue introducido en Sección 6.1.1. Este algoritmo opera en dos etapas:

1. Calcula el tráfico óptimo para los $PIDs$ sólo considerando las restricciones de anchos de banda y no las de capacidades de los enlaces.

2. Luego de obtenida la solución para los $PIDs$ trata de descongestionar el enlace más saturado utilizando caminos de largo 2.

La desventaja del Algoritmo Simplex radica en que no siempre la primer etapa genera una solución factible, este motivo hace que muchas veces sea muy engorroso tratar de factibilizar una solución. Incluso en muchas instancias esto puede ser imposible, por lo que se tiene que recurrir a penalizaciones en las transferencias de los $PIDs$.

Instancias para la emulación

Al igual que para las instancias del Problema 1 se utilizó información real extraída de algunos partidos de fútbol Uruguayo transmitidos en vivo por Internet. De aquí se tiene el comportamiento real de los pares (conexiones y desconexiones) durante un intervalo de tiempo del streaming para cada uno de los partidos además de su ubicación geográfica (por lo menos a nivel país). Esta información es usada como input para el emulador de GoalBit (GoalBit-emu [2]) el cual puede medir las transferencias de flujos totales en la red para el streaming en particular y también el total de flujo transferido por los enlaces internacionales que entran y salen de Uruguay. Cabe destacar que las mediciones que se consiguen son reales, es decir que los tráficos y los tiempos de espera de cada par coinciden con lo que se hubiese observado en el streaming verdadero.
En total se generaron 9 inputs para emular, estos difieren en los largos considerados de streaming, el rango de pares conectados al streaming (los pares se conectan y desconectan todo el tiempo es por esto que no se puede definir un número fijo de pares). Estas emulaciones también fueron realizadas por el Ing. Matías Barrios. En el Cuadro 7.4 se muestran las características de cada uno de los inputs: rango de pares conectados y duración del streaming en minutos.

Input	Rango pares	Tiempo
1	20-40	10
2	20-40	10
3	20-40	10
4	20-40	70
5	60-120	50
6	60-120	50
7	75-80	10
8	80-200	70
9	100-110	10

Cuadro 7.4: Características de los inputs para el emulador.

Resultados de las emulaciones

Como resultados de las emulaciones mostraremos 4 indicadores:

1. Incremento porcentual del flujo neto que pasan por enlaces internacionales salientes del PID Uruguay utilizando la metodología P4P basada en el Algoritmo Simplex en relación con flujo neto por enlaces internacionales salientes del PID Uruguay utilizando la metodología de selección de pares al azar, como se muestra en la Ecuación 7.3.

$$\Delta_{FS} = 100 \times (\frac{FS_{P4P}}{FS_{azar}} - 1) \tag{7.3}$$

   Donde FS_{P4P} y FS_{azar} son los flujos salientes por enlaces internacionales obtenidos por la metodología P4P y el azar respectivamente. Notemos que valores negativos en la Ecuación 7.3 se traduce en una menor utilización de los enlaces internacionales que entran al PID Uruguay, lo cual sería deseable de observar ya que justificaría el uso de P4P.

2. Incremento porcentual del flujo neto que pasa por enlaces internacionales entrantes al PID Uruguay utilizando la metodología P4P basada en el Algoritmo Simplex en relación con flujo neto por enlaces internacionales entrantes al PID Uruguay utilizando la metodología de selección de pares al azar, como se muestra en la Ecuación 7.4.

$$\Delta_{FE} = 100 \times (\frac{FE_{P4P}}{FE_{azar}} - 1) \tag{7.4}$$

Nuevamente se esperaría que la metodología P4P disminuya el tráfico por estos enlaces, por lo cual es deseable observar valores negativos en la Ecuación 7.4.

3. El incremento porcentual del flujo total transferido en la red que se produce al utilizar P4P versus utilizar la selección de pares al azar.

$$\Delta_{FT} = 100 \times (\frac{FT_{P4P}}{FT_{azar}} - 1) \tag{7.5}$$

Donde FT_{P4P} y FT_{azar} son los flujos totales obtenidos por las metodologías P4P y azar respectivamente. En este caso es deseable observar valores positivos de Δ_{FT}, ya que esto significa que los pares lograron una mayor transferencia de flujos.

4. La distribución de los tiempos en segundos que necesitó cada par para empezar a reproducir el streaming (Buffering time).

En el Cuadro 7.5 podemos ver que en los últimos 5 inputs se consiguieron reducciones significativas del uso de los enlaces internacionales, mostrando todas estas reducciones de al menos 38.80% en los enlaces salientes y 35.63% en los entrantes. Solo en dos de estas 5 instancias se produjo una reducción del flujo total siendo en la 6^{ta} de apenas un 1.86% y en la 7^{ma} ya más considerable se registró una reducción de un 12.17%.
Las primeras 4 instancias tienen la particularidad de tener un número muy reducido de pares, todas presentan un rango de entre 20 y 40 pares durante el streaming, quizás sea por este motivo que no evidencian mejoras en todos los indicadores. De hecho para las instancias 2 y 4 los enlaces de entrada registran un incremento bastante importante, 54.26% y 49.10% respectivamente. Además la instancia 4 presenta una disminución en el total de tráfico de un 3.52%.
De todas formas estos resultados están lejos de ser desalentadores. Si observamos los promedios de los indicadores vemos que se registró una disminución del 29.90% y 23.89% de tráfico por enlaces entrantes y salientes respectivamente y consiguiendo en promedio una transferencia de flujo neta del 1.69%.

En las figuras 7.1 y 7.2 podemos ver las distribuciones de los tiempos de espera (buffering time) de cada una de las metodologías para las instancias 5 y 9, es de esperar que la gran mayoría o todos los pares puedan comenzar a ver el video en un tiempo razonable (algunos segundos).
En la Figura 7.1 casi que no existe diferencia entre ambas distribuciones, más del 80% de los pares pudo reproducir el video en menos de 50 segundos o menos (indistintamente si se utilizo P4P o no).
En la Figura 7.2 se observa que casi el 65% de los pares bajo la metodología P4P

Instancia	Rango pares	Δ_{FS}	Δ_{FE}	Δ_{FT}
1	20-40	13.70	-59.02	2.49
2	20-40	-14.63	54.26	6.29
3	20-40	-13.05	13.35	-2.42
4	20-40	-23.22	49.10	-3.52
5	60-120	-47.57	-74.96	0.00
6	60-120	-43.33	-44.55	-1.86
7	75-80	-45.64	-59.50	-12.17
8	80-200	-38.80	-58.08	19.49
9	100-110	-56.59	-35.63	6.93
Media	70	-29.90	-23.89	1.69
Desvío	40	22.35	49.58	8.79

Cuadro 7.5: Resultados de las 9 emulaciones realizadas en la red GoalBit.

logran empezar a reproducir el video en menos de 37 segundos y casi la totalidad de los pares lo reproduce en menos de 100 segundos, mientras que utilizando la metodología de selección al azar, sólo el 43% de los pares pueden reproducir antes de los 50 segundos, luego entre 50 y 300 segundos solo lo logran un 10% y los 47% de pares restantes registran tiempos mayores a 300 segundos. Otros estudios pueden realizarse sobre medidas de calidad del streaming como por ejemplo la continuidad de la reproducción [42].

En resumen, se justifica el uso de la metodología P4P ya que se observaron decrementos en los tráficos de flujo en los enlaces internacionales que entran y salen del PID Uruguay no disminuyendo la transferencia de flujo total. Además se constató que el tiempo de espera para poder reproducir el streaming mejoró o no empeoró al aplicar P4P. Esto se traduce en un uso más eficiente de los recursos de la red y un beneficio para el usuario final.

7.5 Resultados para el Problema General

En esta Sección presentamos resultados de aplicar el Algoritmo basado en GRASP introducido en el Capítulo 6. Como ya hemos mencionado antes es muy difícil conseguir información real para evaluar los algoritmos que hemos propuesto, esto se hace aún más complicado cuando queremos evaluar la performance de la metaheurística propuesta para el Problema General. Es por esto que decidimos probar el algoritmo con instancias generadas aleatoriamente. Para evaluar la performance

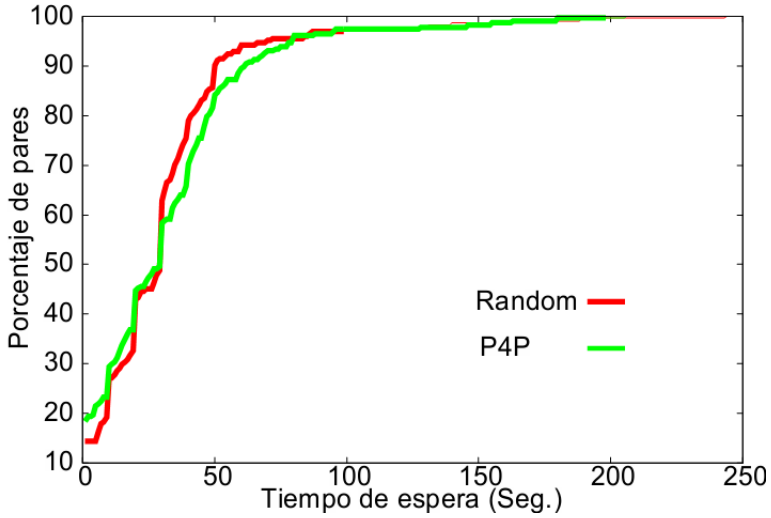


Figura 7.1: Distribución del tiempo de espera para la reproducción del video para la instancia 5.

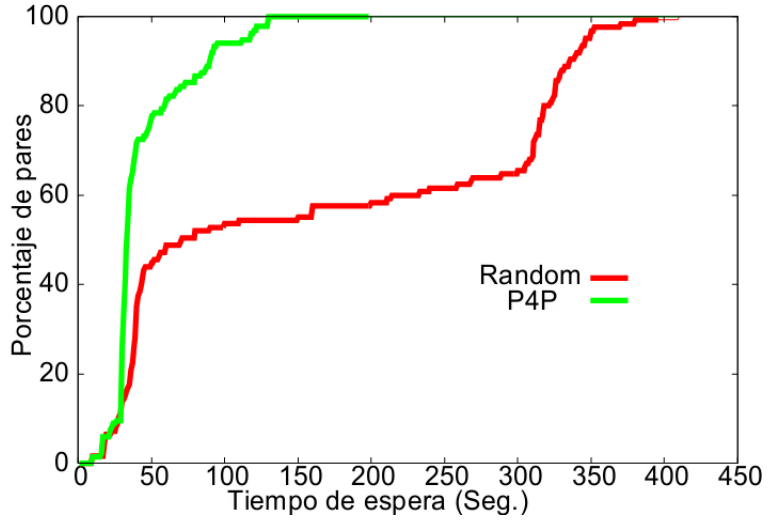


Figura 7.2: Distribución del tiempo de espera para la reproducción del video para la instancia 9.

del algoritmo GRASP compararemos las soluciones obtenidas con una cota. Esta cota será definida en la Sección 7.5.2 y la misma nos permite saber si el algoritmo logró alcanzar una solución óptima. A continuación se define cómo se generarán las instancias de prueba para el algoritmo basado en GRASP.

7.5.1 Generación de Instancias

Por falta de acceso a la información decidimos generar instancias totalmente aleatorias, sólo con el fin de ver el desempeño del Algoritmo GRASP propuesto para resolver el Problema General. La generación de instancias se realiza siguiendo los siguientes pasos:

1. Se generan los anchos de banda de subida para cada PID y cada uno de los K contenidos, simplemente se sortea para cada pareja $\{k, i\}$, $\forall i \in [1, \ldots, N]$, $\forall k \in [1, \ldots K]$ de la matriz U una uniforme [0,1] multiplicada por el número de $PIDs$ en la red (N). De esta forma determinamos los anchos de banda de subida de los $PIDs$ donde $U_{\{k,i\}} = u_i^k$, $\forall PID_i$, $\forall k \in [1, \ldots K]$.
$$U = rand(K, N) \times N$$

2. Para las capacidades de ancho de banda de bajada la metodología es muy similar a la anterior, sólo que se incluye un parámetro adicional r para reflejar el hecho de que por lo general los anchos de banda de bajada son mayores a los de subida, es por esto que siempre se define $r > 1$.
$$D = rand(K, N) \times r \times N$$
Otra opción que consideramos captura la idea de que los $PIDs$ por lo general tienen de 2 a 4 veces más ancho de banda de bajada que de subida, definiendo D como:
$$D = U \times r + \varepsilon$$
Donde ε es un ruido blanco.

3. Las capacidades de los enlaces se generan de tal forma que los $PIDs$ puedan en ocasiones subir todo el flujo que puedan y en ocasiones no. Por como fue generada la matriz de anchos de banda de subida (U) en valor esperado cada PID tendrá una magnitud de flujo para subir igual a $K \times \frac{N}{2}$. Si consideramos la matriz de capacidades C sólo generando valores aleatorios uniformes en las celdas distintas de la diagonal (la matriz C tiene valores nulos en su diagonal) esto haría que, en valor esperado, cada PID pueda subir una magnitud de flujo igual a $\frac{N-1}{2}$ por sus enlaces directos hacia los otros $PIDs$ de la red. Por lo cual se decide generar la matriz C como:
$$C = rand(K, N) \times s \times K$$
Donde s es un parámetro que regula la exigencia de los enlaces de la red. Por lo general si $s \leq 1$ es de esperarse que las instancias generadas tengan la propiedad de ser pesimistas.

4. Se asume que el tráfico de fondo para todos los enlaces es nulo.

7.5.2 Cota para el Problema General

Para tener un punto de referencia de la performance del Algoritmo GRASP para el PG se construye una cota, la cual asegura que si ésta es alcanzada entonces el algoritmo propuesto logra alcanzar una solución óptima.
La idea es muy sencilla, básicamente se reduce el Problema General al Problema 1, sumando para cada PID los anchos de banda de subida y bajada de todos los contenidos, de esta forma definimos $u_i^{'}$ y $d_i^{'}$ como:

$$u_i^{'} = \sum_k u_i^k$$

$$d_i^{'} = \sum_k d_i^k$$

De esta forma ya no existen múltiples contenidos, sino que se transformó en uno solo que mezcla todos los contenidos originales, es por esto que se logra reducir el Problema General al Problema 1.
Lo que estamos perdiendo en esta reducción es que no podemos asegurar que se cumplan las restricciones de anchos de banda pero sí se van a cumplir las restricciones de capacidades de los enlaces.

Características de la cota

Si bien la cota propuesta nos permite tener un punto de referencia de qué tan próximos estamos de las soluciones óptimas, hay que tener cuidado con la interpretación de qué consideramos como próximo. Consideremos los siguientes casos:

1. Si el Algoritmo GRASP alcanza la transferencia neta y la utilización determinada por la cota, entonces podemos asegurar que esta solución es óptima para el Problema General.

2. Si sólo se alcanza la transferencia neta lograda por la cota y no se obtiene la misma saturación de los enlaces (se obtiene una saturación mayor) entonces la mejor solución será aquella que logre alcanzar la transferencia neta definida por la cota y obtenga la mínima saturación de los enlaces.

3. Ahora, el peor de los escenarios se da cuando no se alcanza la transferencia neta lograda por la cota, en estos casos la mejor solución que podemos obtener será aquella que genere el máximo tráfico neto con la mínima utilización. Notemos que en estos casos es muy probable que la utilización obtenida por el Algoritmo GRASP sea menor que la obtenida por la cota, esto es debido

a que en encontrar la cota se introdujo más flujo a la red lo cual produce una mayor utilización de los recursos de la red.

Los tres puntos antes mencionados son de gran utilidad para determinar cuál es la mejor solución que se obtiene al aplicar el Algoritmo GRASP. Lo cual nos guía a definir como solución óptima la que cumpla la siguiente condición:

$$S(T,\rho)_{opt} = \min_{i\in[1,\ldots,iter]} \frac{\rho_i}{\frac{T_i}{T_{cota}}} \tag{7.6}$$

Donde ρ_i y T_i son la saturación mínima y la transferencia neta máxima logradas en la $i-$ ésima iteración del Algoritmo GRASP. Notemos que si tomamos la Ecuación 7.6 como función objetivo, las dos primeras situaciones mencionadas anteriormente estarán completamente contempladas, ahora la situación tres no es del todo seguro, sí es seguro que se genera un compromiso entre alcanzar el flujo máximo con la mínima saturación, pero no se descarta que existan instancias en las cuales al utilizar la Ecuación 7.6 como función objetivo se descarten soluciones que generen más flujo neto que la solución $S(T,\rho)_{opt}$.

7.5.3 Resultados para el Problema General

Se procedió a realizar distintas comparaciones utilizando las instancias generadas aleatoriamente. Las comparaciones se realizaron de la siguiente manera, se define los siguientes parámetros: $PIDs$, K, r, s, $iter$ y $veces$ los cuales describiremos a continuación:

$PIDs$: es el número de $PIDs$ considerados en la instancia.

K: es el número de contenidos considerados en la instancia.

r: es el parámetro que regula cuánto más grande se espera que sean los anchos de banda de bajada con respecto a los de subida.

s: es el encargado de regular qué tan limitadas serán las capacidades de los enlaces, es de esperarse que para valores $s \leq 1$ las instancias aquí generadas tengan la propiedad de ser pesimistas.

$iter$: es el número de veces que se ejecutan las Fases de GRASP (se toma de las $iter$ iteraciones la solución que logra el valor mínimo de la función objetivo)

$veces$: es el número de veces que se generan instancias con las mismas características, en otras palabras se repite la generación de instancias dejando fijo el número de $PIDs$, la cantidad de contenidos K, y los parámetros r y s. Esto

nos permite obtener medidas de resumen para un conjunto de instancias con características probabilísticamente iguales.

Se muestran 9 cuadros con medidas de resumen para evaluar el desempeño del Algoritmo GRASP, cada uno de estos cuadros se diferencian mayoritariamente en la elección de los parámetros r y s como se muestra en el Cuadro 7.6. Para cada uno de los 4 escenarios mostrados en cada cuadro se generaron 5 instancias (parámetro *veces*) con idénticas características, es decir estas 5 instancias se obtienen manteniendo los parámetros $PIDs$, K, r, s y $iter$ fijos.

Cuadro	Escenarios	PIDs	K	r	s	$iter$	$veces$
Cuadro 7.7	Escenario 1	15	5	2	2	10	5
	Escenario 2	15	10	2	2	10	5
	Escenario 3	30	5	2	2	10	5
	Escenario 4	30	10	2	2	10	5
Cuadro 7.8	Escenario 1	20	5	2	2	10	5
	Escenario 2	20	10	2	2	10	5
	Escenario 3	40	5	2	2	10	5
	Escenario 4	40	10	2	2	10	5
Cuadro 7.9	Escenario 1	20	5	1.6	1.9	10	5
	Escenario 2	20	10	1.6	1.9	10	5
	Escenario 3	40	5	1.6	1.9	10	5
	Escenario 4	40	10	1.6	1.9	10	5
Cuadro 7.10	Escenario 1	20	5	1.2	1.8	10	5
	Escenario 2	20	10	1.2	1.8	10	5
	Escenario 3	40	5	1.2	1.8	10	5
	Escenario 4	40	10	1.2	1.8	10	5
Cuadro 7.11	Escenario 1	20	5	1.2	2	10	5
	Escenario 2	20	10	1.2	2	10	5
	Escenario 3	40	5	1.2	2	10	5
	Escenario 4	40	10	1.2	2	10	5
Cuadro 7.12	Escenario 1	20	5	2	2	10	5
	Escenario 2	20	10	2	2	10	5
	Escenario 3	40	5	2	2	10	5
	Escenario 4	40	10	2	2	10	5
Cuadro 7.13	Escenario 1	20	5	1.5	1.7	10	5
	Escenario 2	20	10	1.5	1.7	10	5
	Escenario 3	40	5	1.5	1.7	10	5
	Escenario 4	40	10	1.5	1.7	10	5
Cuadro 7.14	Escenario 1	20	5	1.2	1.5	10	5
	Escenario 2	20	10	1.2	1.5	10	5
	Escenario 3	40	5	1.2	1.5	10	5
	Escenario 4	40	10	1.2	1.5	10	5
Cuadro 7.15	Escenario 1	20	5	2	1	10	5
	Escenario 2	20	10	2	1	10	5
	Escenario 3	40	5	2	1	10	5
	Escenario 4	40	10	2	1	10	5

Cuadro 7.6: Características de las instancias generadas

Los cuadros a continuación muestran la media y desvío de la saturación lograda por el Algoritmo GRASP para el enlace más exigido (Saturación), la media y desvío del flujo neto logrado por el GRASP (F. Max). Además para comparar las soluciones obtenidas con la cota, se calcula el cociente entre el flujo alcanzado por el GRASP y el logrado por la cota para cada una de las instancias y se muestran la media y desvío de esta medida (F_{alg}/F_c). Por último los cuadros muestran en su columna final el porcentaje de veces que la cota fue efectivamente alcanzada (% Cota). La idea aquí es ver el comportamiento del algoritmo con respecto a la variación de los parámetros r y s.
Comentaremos a continuación los resultados obtenidos para cada conjunto de instancias generadas.

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
15	5	0.9667	0.0676	559.6	11.14	1	0	100
15	10	0.7445	0.0637	1106.6	42.83	1	0	100
30	5	0.8776	0.0774	2239.3	103.90	1	0	100
30	10	0.7696	0.0700	4498.9	197.37	1	0	100

Cuadro 7.7: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 2$, $s = 2$, $iter = 10$ y $veces = 5$

En el Cuadro 7.7, podemos ver que el algoritmo GRASP logra alcanzar la cota en el 100% de las instancias generadas, logrando una saturación promedio menor a 1 en todos los escenarios. Estas instancias fueron generadas para valores de r y s iguales a 2, es de esperarse que cuanto más grande sea la magnitud r, será más sencillo alcanzar la cota. Esto se debe a la característica de la metodología para construir la cota, si las capacidades de bajada son muchos más grandes que las de subida entonces el efecto producido al confundir los contenidos en un gran contenido único (procedimiento de la cota) se diluye. Pensemos en el caso extremo donde se cumple que $d_i^k = \infty \forall\, k \in [1, \ldots, K]$, $\forall\, i \in [1, \ldots, PIDs]$, entonces cada PID sólo se vería limitado para subir flujo sólo por las capacidades de los enlaces de la red y no por el hecho de mandarle flujo a un PID específico. Lo mismo se puede observar en el Cuadro 7.8, sólo que en este se realizó para distintos valores del parámetro $PIDs$.

En el Cuadro 7.9, se limita un poco las capacidades de la red considerando el parámetro $s = 1.9$ y tomando $r = 1.6$, ahora los anchos de banda de bajada ya no se espera que sean muy superiores a los de subida, observándose que sólo una de

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
20	5	0.9169	0.0959	1012.8	67.56	1	0	100
20	10	0.8022	0.0411	2040.4	54.16	1	0	100
40	5	0.8716	0.0807	4042.4	106.95	1	0	100
40	10	0.7361	0.0854	8068.3	278.12	1	0	100

Cuadro 7.8: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 2$, $s = 2$, $iter = 10$ y $veces = 5$

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
20	5	0.8378	0.0771	957.9	19.11	0.9984	0.0036	80
20	10	0.8757	0.0868	2013.6	136.30	1	0	100
40	5	0.8384	0.0482	3860.1	241.42	1	0	100
40	10	0.8617	0.0377	7923.2	184.54	1	0	100

Cuadro 7.9: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 1.6$, $s = 1.9$, $iter = 10$ y $veces = 5$

las 5 instancias generada para $K = 5$ y $PIDs = 20$ no alcanzó la cota. Esto quizás se deba a que en las 10 veces que se iteró el algoritmo GRASP no se encontró el óptimo o quizás no era posible alcanzar la cota para esa instancia en particular.

En el Cuadro 7.10 se disminuyen más aún los parámetros r y s, tomando valores de 1.2 y 1.8 respectivamente, se observa que de las 5 instancias generadas para $K = 10$ y $PIDs = 20$ en ninguna se alcanzó la cota, pero de todas formas se logró subir en promedio un 98.63% del flujo máximo posible alcanzado por la cota. En el resto de los escenarios presentados en el Cuadro 7.10 en promedio se subió más del 99.4% del flujo máximo y en varias instancias la cota fue alcanzada.

En el Cuadro 7.11, se mantiene con respecto al Cuadro 7.10 el valor del parámetro $r = 1.2$ y se aumenta el parámetro s a 2. En todos los escenarios se alcanzó la cota al menos una vez, cabe destacar que en el escenario con valores $K = 5$ y $PIDs = 20$ fue donde la cota se alcanzó más veces (60%), pero el flujo promedio con respecto al flujo máximo fue el más bajo de todos las escenarios (97.44%) indicando que en las 2 instancias que no se alcanzó la cota las diferencias entre el flujo alcanzado por GRASP y el alcanzado por la cota fueron significativas. El hecho de que se produzcan grandes diferencias entre le flujo máximo logrado por la cota y el GRASP no significa que la solución encontrada no sea la óptima. Recordemos

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
20	5	0.8409	0.0709	973.6	54.16	0.995	0.0112	80
20	10	0.8498	0.0906	2000.0	86.83	0.9863	0.0085	0
40	5	0.9297	0.0242	4081.9	106.90	0.9972	0.0054	60
40	10	0.8428	0.0308	7932.6	86.71	0.9948	0.0050	40

Cuadro 7.10: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 1.2$, $s = 1.8$, $iter = 10$ y $veces = 5$

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
20	5	0.8232	0.0892	998.9	51.40	0.9744	0.0387	60
20	10	0.7595	0.0971	1986.6	29.48	0.9813	0.0337	20
40	5	0.8065	0.0785	3999.2	161.42	0.9871	0.017	20
40	10	0.7721	0.0574	7934.8	206.49	0.9909	0.0052	20

Cuadro 7.11: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 1.2$, $s = 2$, $iter = 10$ y $veces = 5$

que sólo podemos asegurar que se encontró una solución óptima para una instancia si la cota es alcanzada tanto en transferencia como en la saturación del enlace más congestionado, de lo contrario no podemos concluir nada.

En el Cuadro 7.12, se asume que existe correlación perfecta entre los anchos de banda de subida y bajada tomando $D = r \times U + \varepsilon$, donde ε se distribuye $\mathcal{N}(0, \frac{1}{10})$. Aquí otra vez se observa que la cota es alcanzada para el 100% de las instancias generadas, otra vez posiblemente por la elección relativamente grande de la magnitud del parámetro r, logrando diluir el efecto de colapsar contenidos.

En el Cuadro 7.13 generamos instancias con anchos de banda de bajada esperados moderadamente superiores a los de subida ($r = 1.5$), pero sí restringimos bastante las capacidades ($s = 1.7$). En este caso el algoritmo GRASP obtiene un buen desempeño en todos los escenarios, alcanzando la cota el 60% de las veces en el peor de los casos ($K = 10$ y $PIDs = 20$). Notemos que debido a que las capacidades de los enlaces se espera que sean más ajustadas por considerar el parámetro s más pequeño en comparación a los cuadros previamente mostrados, el promedio de saturación de los enlaces para cada uno de los escenarios se acerca más a la unidad, de hecho para $K = 5$ y $PIDs = 40$ se registró una saturación promedio del 98.50% y un desvío del 3.35% lo que indica que algunas de las soluciones óptimas

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
20	5	0.8339	0.0627	994.5	84.14	1	0	100
20	10	0.86248	0.0939	2018.6	113.02	1	0	100
40	5	0.8713	0.1044	3992.6	162.11	1	0	100
40	10	0.7387	0.0461	7993.8	104.61	1	0	100

Cuadro 7.12: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 2$ (asumiendo $D = U \times D + \varepsilon$), $s = 2$, $iter = 10$ y $veces = 5$

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
20	5	0.9321	0.0627	1025.9	83.26	1	0	100
20	10	0.9128	0.0847	1965.9	67.28	0.9983	0.0024	60
40	5	0.9850	0.0335	3938.7	74.22	1	0	100
40	10	0.8761	0.0735	8050.2	197.79	0.9984	0.0037	80

Cuadro 7.13: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 1.5$, $s = 1.7$, $iter = 10$ y $veces = 5$

alcanzadas saturaron al menos uno de los enlaces al 100%.

En el Cuadro 7.14 se determina el parámetro $r = 1.2$, es decir se esperaría que los anchos de banda de bajada no sean tanto más grande que los de subida, produciendo que el efecto de considerar todos los contenidos como uno sólo sea más evidente y además se considera el parámetro $s = 1.5$ produciendo que los enlaces de la red no tengan mucha holgura. De hecho el efecto de elegir estos dos parámetros con estas magnitudes se ve reflejado en dos cosas; (1) los promedios de las saturaciones de los enlaces para todos los escenarios es mayor al 95.00%, es más para los valores $K = 5$ y $PIDs = 40$ en todas las soluciones encontradas por el algoritmo GRASP se saturaron completamente todos los enlaces, (2) si bien en todos los escenarios se alcanza la cota al menos el 20% de las veces, no se registra un 100% en ninguno de éstos, esto se explica por el bajo valor considerado para el parámetro r. Sorprendentemente en el escenario que se alcanzó más veces la cota fue en el mismo que todas las soluciones saturaron totalmente algún enlace.

En el Cuadro 7.15 se determina el parámetro $s = 1$, como dijimos antes, para esta magnitud es muy probable que las intancias generadas posean la característica de ser pesimistas, lo cual se ve reflejado en la saturación promedio de los enlaces,

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
20	5	0.9907	0.0207	1008.7	31.83	0.9961	0.0043	20
20	10	0.9651	0.0482	1943.0	105.90	0.9978	0.0028	40
40	5	1.0000	0.0000	4036.8	125.42	1	0	60
40	10	0.9500	0.0314	7935.9	212.31	0.9952	0.0057	20

Cuadro 7.14: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 1.2$, $s = 1.5$, $iter = 10$ y $veces = 5$

		Saturación		F. Max		F_{alg}/F_c		
PIDs	K	Media	Desvío	Media	Desvío	Media	Desvío	% Cota
20	5	1.0000	0.0000	869.6	40.76	0.9982	0.0024	40
20	10	1.0000	0.0000	1699.9	72.49	0.9961	0.0024	0
40	5	1.0000	0.0000	3512.9	75.20	0.9971	0.0019	0
40	10	1.0000	0.0000	7233.1	160.49	0.9972	0.0019	0

Cuadro 7.15: Comparación del Algoritmo GRASP con la cota para distintas instancias, con parámetros $r = 2$, $s = 1$, $iter = 10$ y $veces = 5$

en efecto todas las soluciones para cada escenario logradas por GRASP saturan completamente algún enlace. Otra cosa a destacar es las pocas veces que la cota es alcanzada, solo para $K = 5$ y $PIDs = 20$ se logra la cota sólo en un 40% de las instancias generadas, en el resto no se alcanzó nunca, incluso para una magnitud de parámetro $r = 2$, la cual es relativamente alta y se lograban muy buenos resultados para otras instancias con capacidades más holgadas. De todas formas si miramos la proporción promedio del flujo alcanzado por GRASP en relación a la cota, vemos que en el peor de los casos GRASP logró subir un 99.72% del flujo máximo logrado por la cota. Es probable que para instancias en las cuales las capacidades de los enlaces sean limitadas se tengan que iterar más veces las distintas fases de algoritmo GRASP para alcanzar la transferencia máxima, siempre y cuando esta sea alcanzable.

7.6 Conclusiones

Resultados instancias generadas para el P1

Para ver el desempeño del Algoritmo basado en Ford-Fulkerson para resolver el P1 se consideraron instancias con datos reales extraídos de streamings de partidos de fútbol del 2006, de esta forma se pudieron estimar anchos de banda de bajada para un conjunto de $PIDs$ de la red, con éstos y utilizando información empírica de la relación que existe entre anchos de banda de subida y bajada se estimaron los anchos de banda de subida de los $PIDs$. Por otro lado se consideraron capacidades para los enlaces totalmente aleatorias y se fijaron tráficos de fondo no nulos sólo para los enlaces que vinculan al PID Uruguay.
De esta forma se obtuvieron 12 instancias cuasi reales para poder comparar los algoritmos: *Goloso simple*, GOTEO y el *basado en Ford-Fulkerson*.
El hecho de que las instancias eran todas pesimistas y la transferencia de flujo por caminos mayores a 1 era casi innecesaria, los resultados obtenidos muestran un desempeño muy similar al aplicar cualquiera de los tres algoritmos, de todas formas no sólo el Algoritmo basado en Ford-Fulkerson nunca fue superado por ninguno de los otros dos sino que en las instancias en las cuales tenía sentido utilizar caminos de largos mayores que 1, este algoritmo superó a los demás (ver Cuadro 7.3).

Resultados emulaciones en GoalBit

Nuevamente con información real extraída del streaming de varios partidos de fútbol de la Liguilla de fútbol Uruguayo del 2006, se extrajo la información del comportamiento de los pares durante el streaming en varios intervalos de tiempo de los distintos partidos. Con esta información se pudo emular en la plataforma real GoalBit el efecto de utilizar P4P para determinar la lista de pares versus utilizar una estrategia totalmente aleatoria. Los resultados fueron muy alentadores, al aplicar P4P se registró en promedio una reducción de un 29.90% de flujo saliente por enlaces internacionales del PID Uruguay y una reducción promedio de un 23.89% de flujo entrante por enlaces internacionales del PID Uruguay. Además se observó que en promedio la magnitud del flujo total transferido en la red aumentó un 1.69% (ver Cuadro 7.5), todo esto manteniendo la distribución del tiempo de espera muy similar a la lograda sin P4P (ver Figura 7.1) y en algunos casos mejorando los tiempos de espera de forma significativa (ver Figura 7.2). Recordemos que la metodología P4P aplicada en estas emulaciones se realizaron por medio del Algoritmo basado en Simplex introducido en el Capítulo 6.

Resultados para el Problema General (PG)

La falta de información real para realizar cualquier tipo de estimación de los parámetros requeridos para el PG nos obligó a tener que generar instancias de prueba totalmente aleatorias. De todas formas a modo de poder comparar el Algoritmo GRASP propuesto para resolver el PG construimos una cota tomando los K contenidos como uno solo, de esta forma llevamos el Problema PG al P1. Esta cota nos permite saber si la solución obtenida mediante GRASP es óptima si y sólo si se alcanza tanto el flujo como la saturación mínima obtenida por la cota. En el caso que no se alcancen estos valores sólo podemos medir que tan lejos estamos de la transferencia obtenida por la cota, la cuál no tiene por qué ser factible, por tanto si la cota no es alcanzada no podemos verificar si la solución obtenida por GRASP es o no una solución óptima.
Observando los cuadros mostrados en la Sección 7.5.3, notamos que cuanto más grande son los anchos de banda de bajada con respecto a los de subida el Algoritmo GRASP alcanza la cota sin problemas, también el hecho que las capacidades de los enlaces de la red sean muy limitados puede producir que se tengan que realizar más iteraciones de las fases del GRASP para que la cota sea alcanzada ya que para instancias con valores esperados de anchos de bajada relativamente "altos" la cota es alcanzada sólo en 2 instancias (ver Cuadro 7.14).

Capítulo 8

Conclusiones

8.1 Conclusiones y contribuciones

El objetivo primordial de este trabajo es proponer nuevas técnicas de ruteo que faciliten la gestión por parte de los proveedores de servicio de internet (ISPs) y además cumplan con las expectativas de los usuarios o pares de las redes de pares, o redes P2P. Hemos dado una revisión de diferentes concepciones de la ingeniería de tráfico, desarrollando en este trabajo variantes de una técnica muy prometedora que ofrece un claro compromiso entre los objetivos de pares e ISPs.

En el Capítulo 2 se introdujo el modelo matemático de P4P tal como se presenta en el artículo original [50] dejando en manifiesto cuáles son los objetivos contrapuestos desde la perspectiva de los pares e ISPs. También se introducen brevemente otros trabajos realizados y resultados obtenidos por el *P4P Working Group* utilizando información de compañías como Telefónica y Verizon. Se introduce además la metodología *ONO* la cual comparte los objetivos de P4P y tiene la ventaja de no necesitar mucha información de la red.

En el Capítulo 3 se definen formalmente todos los problemas que se abordan en este documento, los cuales llamamos:

- *Problema General* o *PG*, el cual asume conocimiento completo de la red y la transferencia simultanea de múltiples contenidos en la red.

- *Problema 1* o *P1*, se asume conocimiento completo de la red y es una simplificación del PG, ya que en el P1 sólo se considera la transferencia de un contenido en la red.

1. Variante 1 del P1 ($VP1_\rho$), aquí también como en P1 se asume conocimiento completo de la red y que sólo existe un contenido a transferir pero se fija una utilización máxima para los enlaces de la red ρ_{max}.
2. Variante 2 del P1 ($VP1$), esta variante comparte los supuestos del P1 pero se agrega una restricción más que obliga a que las transferencias se realicen sólo por caminos de largo 1.

- *Problema 2*, este problema está incluido en P1 y surge por la dificultad de contar con información completa de la red, en particular sólo se conocen los tráficos de fondo para los enlaces que vinculan a un PID especial que en nuestro caso será el PID Uruguay.

- *Problema 3*, este problema quizás sea el más realista en términos de acceso a la información ya que asume que se conocen sólo las capacidades y los tráficos de fondo de los enlaces que vinculan al PID Uruguay.

El Capítulo 4 se centra en comprender la naturaleza de la estructura del Problema 1. Mediante el estudio de diferentes instancias es posible entender algunos aspectos esenciales del problema.
En este proceso definimos dos tipos de situaciones o estados de la red que se identifican con las posibilidades que tienen los $PIDs$ de cumplir sus deseos. Para ello se introduce el concepto de instancia *optimista* y *pesimista* para designarlas. En la primera se designan los casos en los que los pares desean subir flujo inferior a las capacidades disponibles de la red, en la segunda por contraposición se designan instancias en que desean subir más de lo que las capacidades le permiten.
Para el Problema 3 se encuentra un algoritmo que lo resuelve de forma óptima en tiempo polinomial y se propone un algoritmo que llamamos *Goteo*, diseñado para resolver instancias en las cuales existe un PID con altos tráficos de fondo en sus enlaces. Este algoritmo se basa fundamentalmente en la técnica llamada *waterfilling* para generar ruteos por los enlaces de forma equitativa.
Consideramos un algoritmo Goloso simple para resolver el Problema 1 sólo con el fin de estudiar sus debilidades, las cuales tratamos de superar de distintas maneras.

La parte sustancial de este trabajo se desarrolla en el Capítulo 5, es aquí que planteamos la resolución tanto del Problema 1 como de sus dos variantes ($VP1_\rho$ y $VP1$) como un problema de corte mínimo y flujo máximo, aplicando el Algoritmo de Ford-Fulkerson en una red auxiliar construida de tal forma que existe una biyección entre las red auxiliar y la original según el problema en cuestión. El algoritmo propuesto aquí logra encontrar en tiempo polinomial la solución óptima para el $VP1_\rho$ y nos guía para desarrollar un nuevo algoritmo basado en Ford-Fulkerson que es un $FPTAS$ (*Fully Polynomial Time Approximation Scheme*) tanto para

$VP1$ como para P1. Esto nos permite aproximarnos tanto como queramos a soluciones óptimas en tiempo polinomial para ambos problemas.

En el Capítulo 6, proponemos la resolución del Problema General por medio de distintas metaheurísticas, considerando un método basado en el Algoritmo Simplex y por último GRASP. Siendo esta última la elegida para desarrollar el algoritmo para resolver el PG.
El Algoritmo basado en GRASP utiliza en su fase de construcción el mismo algoritmo basado en Ford-Fulkerson desarrollado para resolver el P1, eligiendo en cada paso un contenido para transferir. Luego de que se construyó una solución factible se aplica la fase de búsqueda local en la cual se intenta descongestionar el enlace más saturado (MLU).

En el Capítulo 7 se muestran distintos resultados:

- Mediante datos reales obtenidos de varias transmisiones en vivo de partidos de futbol de la Liguilla Uruguaya, se logró estimar anchos de banda de subida y bajada con cierto grado de veracidad. con esta metodología se generaron 12 instancias distintas para poder evaluar el desempeño de los distintos algoritmos propuestos para resolver el Problema 1. De estas comparaciones se destaca que el Algoritmo basado en Ford-Fulkerson obtuvo mejor o igual desempeño en cada una de las instancias generadas, aunque las mejoras en las soluciones no fueron muy significativas, el hecho de que no fuese superado por los otros algoritmos muestra la contundencia del algoritmo, además el tipo de instancias que aquí se generaron no son las indicadas para observar el potencial del Algoritmo de Ford-Fulkerson ya que en muy pocas de ellas se justifica el ruteo por caminos mayores a 1.

- Utilizando información de las mismas transmisiones en vivo de los partidos de la Liguilla Uruguaya del 2006, se extrajo información real del comportamiento de los pares (conexión y desconexión) conectados al streaming en varios intervalos de tiempo (ventanas). Estos datos se utilizaron como inputs para emulaciones en la plataforma real GoalBit. Los mismos inputs fueron introducidos en el emulador considerando P4P para generar la lista de pares para que se comuniquen utilizando el Algoritmo basado en Simplex versus no utilizar P4P y generar las listas de pares de forma totalmente aleatoria. El resultado de estas emulaciones justifica el uso de la metodología P4P ya que esta logra en promedio reducciones del tráfico de flujo por enlaces internacionales salientes y entrantes del PID Uruguay del 29.90% y 23.89% respectivamente. Además la transferencia neta de flujo en la red aumenta

en promedio un 1.69% y todo esto sin aumentar los tiempos de espera para empezar a reproducir el streaming.

- Por último para ver el desempeño del Algoritmo GRASP propuesto para resolver el PG, se decide por el hecho de no contar con información, generar instancias totalmente aleatorias definiendo algunos parámetros que permiten tener control en términos de valores esperados de la relación entre los anchos de banda de subida y bajada, así como de qué tan limitadas serán las capacidades de los enlaces de la red. Para poder comparar el Algoritmo basado en GRASP, se define una cota llevando el PG al P1 juntando todos los contenidos como un contenido único. Esta cota de ser alcanzada nos asegura que la solución obtenida es óptima, mientras que de no hacerlo no podemos decir nada. De todas formas la cota es muy útil para evaluar que tan lejos estamos de la transferencia neta máxima. En las distintas simulaciones que se realizaron se puede constatar que el Algoritmo GRASP alcanza sin problemas la cota cuando los anchos de banda de bajada son bastante más grandes que los de subida. Por otro lado, ante instancias con capacidades limitadas, los resultados sugieren que se tenga que iterar más veces las fases del Algoritmo GRASP para lograr soluciones óptimas.

Este trabajo se realizó como parte de la actividad específica "*Sistema eficiente de distribución de video y TV en tiempo real*" en el marco del convenio entre la Facultad de Ingeniería de la Universidad de la República y la empresa de telecomunicaciones de la República Oriental del Uruguay (ANTEL). El objetivo principal de dicha actividad era evaluar la implementación y el impacto de aplicar la metodología P4P a la distribución P2P de video en tiempo real. Es importante destacar que los buenos resultados presentados en este documento, respecto a las emulaciones de la plataforma GoalBit utilizando la metodología P4P, validan la implementación de la misma. Por otro lado, el buen desempeño del Algoritmo GRASP desarrollado en este trabajo sugiere considerar a éste como candidato para calcular los tráficos óptimos entre los $PIDs$ de la red.

8.2 Trabajo a futuro

En esta Sección presentamos algunos trabajos propuestos para el futuro:

- Para este trabajo se consideró a los $PIDs$ como países, lo cual es una forma cómoda para determinar una topología sencilla de la red. Sería interesante estudiar otro tipo de asignaciones de $PIDs$, por ejemplo conjuntos de PoP

(Points of Presence) geográficamente cercanos. Estos cambios en la topología de la red presentarían nuevos desafíos para la aplicación de los algoritmos presentados en este trabajo. Algunos de estos desafíos son:

  - Adaptar la red auxiliar para esta topoloía más desagregada de forma que se pueda seguir aplicando el Algoritmo de Ford-Fulkerson.
  - En este trabajo sólo consideramos enlaces internacionales, si se consideran a los $PIDs$ como $PoPs$[1] es necesario distinguir entre dos tipos de enlaces: los internacionales y los que atraviesan al ISP.

- Recordemos que los resultados de las emulaciones obtenidos en la Sección 7.4.1 se obtuvieron aplicando el Algoritmo Simplex, queda pendiente emular en la plataforma real GoalBit aplicando el Algoritmo GRASP para construir la lista de pares y así tener datos reales del desempeño de este.

- También sería interesante poder emular en la platoforma real GoalBit utilizando una metodología de selección de pares que siga el esquema Ono [10]. De esta forma poder comparar el desempeño de la metodología P4P versus Ono, las cuales hoy día son las grandes competidoras.

[1] Puntos de acceso a Internet

Bibliografía

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti, and M. Protasi. *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Springer-Verlag, ISBN 3-540-65431-3, 1999.

[2] A. Barrios, M. Barrios, J. Comas, and P. Perdomo. Análisis y evaluación del rendimiento de la plataforma P2P Goalbit, 2009.

[3] M. E. Bertinat, D. D. Vera, D. Padula, F. Robledo, P. Rodríguez-Bocca, P. Romero, and G. Rubino. Goalbit: The first free and open source peer-to-peer streaming network. In *Proceedings of the 5th international IFIP/ACM Latin American conference on Networking*, New York, USA, 2009. ACM.

[4] Bittorrent home page. http://www.bittorrent.org, 2007.

[5] J.-Y. L. Boudec. Some properties of variable length packet shapers. In *IEEE/ACM Transactions on Networking*, pages 329–337, 2002.

[6] Cachelogic home page. http://www.cachelogic.com, 2007.

[7] H. Cancela, F. Robledo, and G. Rubino. A GRASP algorithm with RNN based local search for designing a WAN access network. *Electronic Notes on Discrete Mathematics*, 18C:53–58, 2004.

[8] Z. Chen, H. Wang, B. Yang, and Y. Chen. A novel method of p2p hosts detection based on flexible neural tree. In *ISDA '06: Conference on Intelligent Systems Design and Applications*, pages 556–561. IEEE, 2006.

[9] R. S. Cheng and S. Verdti. Gaussian multiaccess channels with isi: Capacity region and multiuser water-filling. *IEEE Trans. Inform. Theory*, 39:773–785, 1993.

[10] D. R. Choffnes and F. E. Bustamante. A practical approach to reducing cross-isp traffic in peer-to-peer systems. In *Proceedings of ACM SIGCOMM*, 2008.

[11] B. Cohen. Incentives build robustness in bittorrent. *www.bramcohen.com*, 1:1–5, May 2003.

[12] C. Commander, C. Oliveira, P. Pardalos, and M. Resende. A GRASP heuristic for the cooperative communication problem in ad hoc networks. In *Proceedings of VI International Metaheuristics Conference (MIC 2005)*, pages 225–230, 2005.

[13] CoolStreaming home page. http://www.coolstreaming.us, 2007.

[14] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, ISBN 0262032937, 2001.

[15] T. M. Cover and J. A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, ISBN 0471062596, 1991.

[16] C. Dale and J. Liu. A measurement study of piece population in bittorrent. *IEEE Global Telecommunications Conference, GLOBECOM 2007*, pages 405–410.

[17] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[18] H. Duan, G. Ma, and S. Liu. Experimental study of the adjustable parameters in basic ant colony optimization algorithm. *IEEE Congress on Evolutionary Computation*, 1(1):149–156, 2007.

[19] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972.

[20] eMule home page. http://www.emule-project.net, 2007.

[21] T. Feo and M. Resende. Greedy randomized adaptive search procedures. In *Journal of Global Optimization*, volume 6, pages 109–133, 1995.

[22] P. Festa, P. Pardalos, L. Pitsoulis, and M. Resende. GRASP with path-relinking for the weighted MAXSAT problem. *ACM J. of Experimental Algorithmics*, 11, 2006. article 2.4: 1-16.

[23] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Boston, ISBN 978-0792399650, 1997.

[24] GoalBit - The First Free and Open Source Peer-to-Peer Streaming Network. http://goalbit.sf.net/, 2008.

[25] R. H. Gohary, Y. Huang, Z. Q. Luo, and J. S. Pang. A generalized iterative water-filling algorithm for distributed power control in the presence of a jammer. *Trans. Sig. Proc.*, 57(7), 2009.

[26] J. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press. Mi. USA, 1975.

[27] S. Horovitz and D. Dolev. LiteLoad: Content unaware routing for localizing P2P protocols. In *Proceeding of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS'08)*, pages 1–8, Miami, USA, April 2008.

[28] R. Johnsonbaugh and W. Pfaffenberger. *Foundations of Mathematical Analysis*. Dover Publications, New York, ISBN 978-0486421742, 2002.

[29] KaZaA home page. http://www.kazaa.com, 2007.

[30] W. Kocay and D. L. Krehe. *Graphs, algorithms, and optimization*. Chapman & Hall/CRC, ISBN 978-1584883968, 2004.

[31] V. Laarhoven. *Simulated Annealing: Theory and applications*. In Mathematics and its Applications, United States, E. H. L. Arts., 1995.

[32] E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, and D. B. Shmoys. *The Traveling salesman problem: a guided tour of combinatorial optimization*. John Wiley & Sons, ISBN 978-0471904137, 1985.

[33] I. Maros. *Computational Techniques of the Simplex Method (International Series in Operations Research & Management Science)*. Kluwer Academic Publishers Group, ISBN 1-4020-7332-1, 2003.

[34] M. Martínez, A. Morón, F. Robledo, P. Rodríguez-Bocca, H. Cancela, and G. Rubino. A GRASP Algorithm Using RNN for Solving Dynamics in a P2P Live Video Streaming Network. *International Conference on Hybrid Intelligent Systems*, 0:447–452, 2008.

[35] pando home page. http://www.pandonetworks.com.

[36] P. Pardalos and M. Resende. *Handbook of applied optimization*. Oxford University Press, New York, ISBN 978-0195125948, 2002.

[37] D. Peng, W. Liu, and C. Lin. Enhancing tit-for-tat strategy to cope with free-riding in unreliable p2p networks. In *Third International Conference on Internet and Web Applications and Services*, pages 336–341, Tsinghua University, Beijing, 2008. IEEE Computer Society.

[38] PPLive Home page. http://www.pplive.com, 2007.

[39] PPStream home page. http://www.ppstream.com/, 2007.

[40] M. G. C. Resende and P. M. Pardalos. *Handbook of optimization in telecommunications, Volumen 10*. Springer Science + Business Media, ISBN 978-0387306629, 2006.

[41] M. G. C. Resende and C. C. Ribeiro. *Greedy Randomized Adaptive Search Procedures*. In F. Glover and G. Kochenberger, editors, Handbook of Methaheuristics, Kluwer Academic Publishers, 2003.

[42] P. Romero. Optimización de la Estrategia de Selección de Piezas de Video en Redes P2P. Master's thesis, Universidad de la República, Facultad de Ingeniería, Montevideo, Uruguay, May 2009.

[43] C. Rostagnol. Optimización de P4P utilizando Algoritmos Genéticos para GoalBit. Technical Report RT 10-08, Instituto de Computación, Facultad de Ingeniería, Universidad de la Republica, Montevideo, Uruguay, 2009.

[44] S. N. Sivanandam and S. N. Deepa. *Introduction to Genetic Algorithms*. Springer, ISBN 978-3540731894, 2007.

[45] SopCast - Free P2P internet TV. http://www.sopcast.org, 2007.

[46] K. Sripanidkulchai, B. Maggs, and H. Zhang. An analysis of live streaming workloads on the internet. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 41–54, New York, NY, USA, 2004. ACM Press.

[47] E. G. Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, New Jersey, ISBN 978-0-470-27858-1, 2009.

[48] TVAnts home page. http://cache.tvants.com/, 2007.

[49] TVUnetworks home page. http://tvunetworks.com/, 2007.

[50] H. Xie, A. Krishnamurthy, A. Silberschatz, and Y. R. Yang. P4P: Proactive Provider Participation for P2P. Technical report, YALEU/DCS/TR-1377, Yale University, Department of Computer Science, 2007.