TESIS DEFENDIDA POR

José Domingo Colbes Sanabria

Y APROBADA POR EL SIGUIENTE COMITÉ

Dr. Carlos Alberto Brizuela Rodríguez
Codirector del Comité

Dr. José Alberto Fernández Zepeda
Codirector del Comité

Dr. Pedro Gilberto López Mariscal
Miembro del Comité

Dr. Héctor Alonso Echavarría Heras
Miembro del Comité

Dr. Hugo Homero Hidalgo Silva
Coordinador del programa de posgrado en Ciencias de la Computación

Dr. David Hilario Covarrubias Rosales
Director de Estudios de Posgrado

25 de octubre de 2011

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA

PROGRAMA DE POSGRADO EN CIENCIAS EN CIENCIAS DE LA COMPUTACIÓN

MECANISMOS ACÍCLICOS PARA PROBLEMAS DE REPARTO DE COSTOS EN JUEGOS COOPERATIVOS

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN CIENCIAS

Presenta:

JOSÉ DOMINGO COLBES SANABRIA

Ensenada, Baja California, México, octubre de 2011

RESUMEN de la tesis de **JOSÉ DOMINGO COLBES SANABRIA**, presentada como requisito parcial para la obtención del grado de MAESTRO EN CIENCIAS en CIENCIAS DE LA COMPUTACIÓN. Ensenada, Baja California, octubre de 2011.

MECANISMOS ACÍCLICOS PARA PROBLEMAS DE REPARTO DE COSTOS EN JUEGOS COOPERATIVOS

Resumen aprobado por:

Dr. Carlos Alberto Brizuela Rodríguez	Dr. José Alberto Fernández Zepeda
Codirector de Tesis	Codirector de Tesis

El análisis y diseño de algoritmos para teoría de juegos se ha convertido en un área de gran relevancia en los últimos años, ésto motivado principalmente por el auge de Internet y los problemas que naturalmente surgen del mismo. Una parte fundamental en esta área es la que trata acerca de los juegos cooperativos en problemas de reparto de costos. Existen ciertos requerimientos que deben ser cumplidos por los mecanismos diseñados, y uno de ellos consiste en encontrar una solución óptima para el problema combinatorio subyacente, considerando el conjunto de usuarios que reciben el servicio. Además de encontrar esta solución óptima, se debe decidir el monto a cobrar a cada usuario de tal modo a recuperar la inversión (equilibrio de presupuesto) y de tal modo a evitar que los usuarios se vean tentados a mentir sobre su ingreso para conseguir que se les cobre menos por el servicio (a prueba de estrategias de grupo). Un ejemplo es el problema de reparto de costos en el diseño de una red de compra-alquiler de una sola fuente (SSROB-CS). El mismo consiste en encontrar un árbol de costo mínimo donde exista un camino desde un vértice fuente a un subconjunto de vértices del grafo que representan a los usuarios, considerando que las aristas del grafo pueden ser compradas o alquiladas. Luego, se debe repartir el costo de este árbol entre los usuarios. Como el problema combinatorio subyacente pertenece a la clase NP-difícil, y si $P \neq NP$, el mismo sólo puede ser resuelto con cierto grado de aproximación. El mejor factor de aproximación del equilibrio del presupuesto para este problema es de 4.6, el cual está dado por un mecanismo de Moulin.

Recientemente se han propuesto los mecanismos acíclicos, los cuales generalizan los mecanismos de Moulin que tradicionalmente se utilizan para los problemas de reparto de costos. Un caso especial de los acíclicos lo constituyen los mecanismos incrementales, los cuales permiten utilizar algoritmos combinatorios ya existentes para transformarlos en mecanismos de reparto de costos, heredando el factor de aproximación de los mismos.

Se ha demostrado que para ciertos problemas de reparto de costos los mecanismos incrementales y los acíclicos básicos obtienen mejores resultados que los mecanismos de Moulin.

En este trabajo se analiza la aplicación del algoritmo primal-dual propuesto por Swamy y Kumar (2004) (cuyo factor de aproximación es de 4.55) en un mecanismo acíclico básico y en uno incremental. Se demuestra que este algoritmo de aproximación no puede aplicarse a estos tipos de mecanismos.

A partir de este resultado, se propone un mecanismo incremental con un factor de aproximación *esperado* de 4 en el equilibrio del presupuesto a partir del algoritmo aleatorio de Gupta *et al.* (2003). También se propone otro mecanismo incremental que utiliza el algoritmo combinatorio de Gupta *et al.* (2008), el cual es aplicado en el mecanismo de Moulin que da, a la fecha, el mejor resultado para el problema considerado. Aunque no se pudo demostrar teóricamente la factibilidad de su aplicación, resultados experimentales sobre casos de prueba generados aleatoriamente motivan a conjeturar que este algoritmo es aplicable en un mecanismo incremental para cualquier caso del problema.

Por último, se realizan experimentos para analizar el beneficio que reciben los usuarios (costo social) cuando se aplica el último mecanismo incremental propuesto y el mejor mecanismo para este problema de reparto de costos, obteniéndose resultados diferenciados. Cuando los ingresos de los jugadores son altos, no existen diferencias significativas entre ambos mecanismos; pero a medida que los mismos van decreciendo, el mecanismo de Moulin obtiene mejores resultados.

Palabras Clave: Algoritmos de aproximación, problemas de reparto de costos, mecanismos acíclicos, mecanismos incrementales, diseño de una red compra-alquiler, juegos cooperativos.

ABSTRACT of the thesis presented by **JOSÉ DOMINGO COLBES SANABRIA**, in partial fulfillment of the requirements of MASTER IN SCIENCES degree in COMPUTER SCIENCE. Ensenada, Baja California, october 2011.

ACYCLIC MECHANISMS FOR COST SHARING PROBLEMS IN COOPERATIVE GAMES

The algorithmic game theory has attracted the attention of many researchers in the last few years, largely motivated by the emergence of the Internet and the problems that naturally arise from it. An essential part of this line of research is the mechanism design for cost-sharing problems in cooperative games. There are some requirements to be fulfilled by the designed mechanisms, and one of them is to find an optimal solution to the underlying optimization problem considering the users to be served. Besides finding the optimal solution the mechanism should decide how to share the cost among the users in order to recover the service cost (budget balance). It also should motivate each player to reveal his true value, even if any group of them collude (group strategyproof). An example is the design of a mechanism for the single source rent-or-buy cost-sharing problem (SSROB-CS). This problem consists in finding a minimum-cost tree connecting a subset of vertices (representing the users) to the source, considering the fact that an edge can be rented or bought. Then, the cost-share of each user must be calculated to recover the cost of the tree. Since the underlying combinatorial problem is NP-hard, only approximations to the optimum can be guaranteed. The best up-to-date budget-balance approximation factor for this problem is 4.6, with a Moulin mechanism.

Recently, acyclic mechanisms have been proposed by Mehta *et al.* (2007). This class of mechanism strictly generalizes Moulin mechanisms that are traditionally used for cost-sharing problems. A special case of acyclic mechanisms is the incremental approach, which allows the use of approximation algorithms for the underlying combinatorial problem to turn them into cost-sharing mechanisms, preserving their approximation factors. It has been demonstrated that the basic acyclic and the incremental mechanisms outperform Moulin mechanisms for certain classes of cost-sharing problems.

In this work we analyze the application of the primal-dual algoritm proposed by Swamy and Kumar (2004) (whose approximation factor is 4.55) to both, basic acyclic and incremental mechanisms, showing that this algorithm is not suitable for these mechanisms.

From this result, we propose an incremental mechanism with an (expected) approximation factor of 4 in budget-balance from the randomized algorithm proposed by Gupta *et al.* (2003). Also, we propose another incremental mechanism using the combinatorial algorithm of Gupta *et al.* (2008), which is applied to the Moulin mechanism that gives the best up-to-date result for this cost-sharing problem. Although we could not demonstrate the feasibility of its application, experimental results on randomly

generated instances lead to the conjecture that this algorithm can be applied to an incremental mechanism for any instance of the problem.

Finally, we compare the performance of the proposed incremental mechanism with that of the best known mechanism for this problem, regarding the social cost. The results show that when the mechanism achieves a high participation, then both mechanisms perform similarly. However, when there is a low participation the Moulin mechanism outperforms the proposed approach.

Keywords: Approximation algorithms, cost sharing problems, acyclic mechanisms, incremental mechanisms, rent-or-buy problem, cooperative games.

A José, Gloria, Cristhian y Andrea

Agradecimientos

Con una profunda satisfacción personal veo hoy cristalizado uno de los sueños que tuve, junto con mi familia, desde hace ya muchos años atrás. Es por ello que haciendo un recuento de varias situaciones que se dieron en el camino y que hacen que este logro tenga un sabor aún más especial, no puedo dejar pasar la oportunidad de agradecer a las personas que me acompañaron en todo este tiempo, en especial:

A mis padres, por su amor y apoyo incondicional. A ellos les debo gran parte de lo que hoy soy, y nunca podré terminar de agradecerles por todo el esfuerzo y sacrificio que realizaron con el objetivo de darnos mayores oportunidades en la vida, y por habernos formado como personas con el ejemplo. Por animarme siempre a perseguir mis sueños, y transmitirme confianza en momentos de debilidad.

A mi hermano Cristhian, mi mejor amigo, por ser todo un ejemplo de persona para mí. Su madurez frente a las situaciones adversas, además de su apoyo en todo momento, me motivaron a dar lo mejor de mí siempre.

A mi prometida Andrea, por su paciencia, comprensión, apoyo y amor incondicional. Por estar siempre pendiente de mí a pesar de la distancia, y por darme fuerzas en los momentos más difíciles.

A mis asesores, Dr. Carlos Brizuela y Dr. Alberto Fernández, por haberme acompañado y asistido con paciencia durante todo este proceso. Al Dr. Brizuela en especial, no sólo por haberme ayudado a venir a estudiar al CICESE, sino por haber sido como un padre para mí durante este tiempo, con su ayuda y consejos.

A los miembros del comité, Dr. Gilberto López y Dr. Héctor Echavarría, por sus valiosos aportes durante el desarrollo del trabajo de tesis.

A mis compañeros de generación, del cubo de Algoritmos y a los demás compañeros de la escuela, por los momentos compartidos y la amistad formada durante estos dos años. A pesar de no estar en mi país, me sentí como en casa.

A mis amigos de Paraguay, por estar siempre acompañándome aún en la distancia, especialmente en los momentos en los que más los necesitaba.

Al Centro de Investigación Científica y de Educación Superior de Ensenada, y en especial al departamento de Computación, por haber hecho mi pasantía por la institución lo más agradable posible. A los profesores, por haber contribuido en mi formación académica.

A la entidad binacional Itaipú, por haberme brindado los recursos económicos para poder realizar mi maestría en México.

Contenido

Página

Contenido (continuación)

Contenido (continuación)

Lista de Figuras

Lista de Figuras (continuación)

Lista de Figuras (continuación)

Lista de Tablas

Capítulo I

Introducción

Un juego consiste de un conjunto de n jugadores $\{1, 2, ..., n\}$. Cada jugador i tiene su propio conjunto S_i de posibles estrategias. Para jugar, cada jugador elige una estrategia $s_{ij} \in S_i$. En función de las estrategias de los usuarios, se producen los resultados del juego (Nisan *et al.*, 2007).

La teoría de juegos es el estudio de las formas en las cuales las interacciones de las estrategias entre jugadores racionales producen resultados[1] respecto a las preferencias (o utilidades) de esos jugadores. La teoría de juegos estudia la interacción de las estrategias en ambientes tanto no-cooperativos (o competitivos) como cooperativos. Es una herramienta central para la economía y ciencias sociales, plantea preguntas desafiantes de investigación en matemáticas, y se aplica en una amplia variedad de campos, incluyendo a las ciencias de la computación, neurociencias, filosofía y biología (Zalta, 2008).

Los juegos pueden representarse enumerando todas las posibles estrategias, y las preferencias o utilidades de todos los jugadores. Generalmente se hace esto mediante una matriz de pagos (representando los posibles resultados), la cual es la representación estándar de un juego. Pero en la mayoría de los juegos, esta representación explícita es de tamaño exponencial en relación a la cantidad de jugadores (Nisan *et al.*, 2007). A veces estas descripciones se pueden evitar utilizando formas abreviadas de repre-

[1] resultados con relación a las ganancias de los jugadores, producto de las interacciones entre los mismos.

sentación tales como: juegos gráficos (grafos), juegos dispersos, juegos simétricos, juegos anónimos, juegos de efecto local, entre otros (Nisan *et al.*, 2007).

Durante los últimos años ha habido un gran crecimiento en investigación en la intersección de ciencias de la computación, teoría de juegos y teoría económica, investigación mayormente motivada por el crecimiento de Internet. Esta última transformó, informó y aceleró mercados, mientras que al mismo tiempo creó nuevas e inimaginables clases de mercados. El área de algoritmos para teoría de juegos desarrolla las ideas centrales en esta nueva área de investigación. Los algoritmos se convirtieron en el ambiente natural y plataforma por omisión para la determinación de estrategias de decisión (Nisan *et al.*, 2007).

Para los **juegos no-cooperativos**, el equilibrio de Nash constituye una solución conceptual importante. El mismo tiene diversas aplicaciones, por ejemplo: cálculo de equilibrios en juegos y mercados, aplicaciones en redes, sistemas punto-a-punto, seguridad, mercados de información, entre otros (Nisan *et al.*, 2007).

Por otro lado, el objetivo de la teoría de **juegos cooperativos** es el estudio de las formas de obtener y sostener la cooperación entre agentes dispuestos a ello. Una pregunta central en este campo es la de cómo los beneficios (o costos) de un esfuerzo comunitario se pueden dividir entre los participantes, tomando en cuenta incentivos individuales y grupales, además de otras condiciones de "justicia" (Nisan *et al.*, 2007).

I.1 Juegos Cooperativos

Formalmente, se considera un conjunto A de n **agentes** que buscan cooperar a fin de generar una ganancia. Esta ganancia generada depende de la **coalición** S de agentes que cooperan. En general, el conjunto de posibles resultados de cooperación entre

agentes en $S \subseteq A$ se indica mediante $V(S)$, donde cada resultado está dado por un vector $x \in \mathbb{R}^{|S|}$, donde x_i especifica la **utilidad** que el agente $i \in S$ obtiene en este resultado. El conjunto A de agentes y la función V definen lo que se conoce como **juego cooperativo** (conocido también como juego de coalición) con **utilidades no transferibles** (abreviado como juegos NTU). Un caso especial, conocido como juego cooperativo con **utilidades transferibles** (abreviado como juego TU), es cuando la ganancia generada por una coalición se puede dividir en forma arbitraria entre los agentes de S. En otras palabras, un juego TU se define al especificar una función $v : 2^{|A|} \mapsto \mathbb{R}$, la cual da una ganancia $v(S) \in \mathbb{R}$ generada por cada coalición S. Se supone que $v(\oslash) = 0$. El conjunto de todos los posibles resultados del juego se define como $V(S) = \{x \in \mathbb{R}^{|S|} : \sum_{i \in S} x_i \leq v(S)\}$ (Nisan *et al.*, 2007).

von Neumann y Morgenstern (1980) propusieron primero la noción de juego cooperativo. Esta noción busca abstraer todos los otros aspectos del juego excepto el aspecto combinatorio de las coaliciones que pueden formarse. Ésto es diferente en los juegos no-cooperativos, donde se enfocan en el conjunto de elecciones (jugadas) disponibles para cada agente (Nisan *et al.*, 2007).

Las ganancias generadas por las coaliciones de usuarios pueden ser tanto positivas como negativas. En el caso de que todas las ganancias sean negativas, se tiene un problema de **reparto de costos** de un servicio entre varios usuarios que lo reciben. En el caso de que las ganancias sean positivas, se tiene un problema de **reparto de ganancias**. Los problemas de reparto de costos se pueden estudiar tanto en modelos TU como NTU. El modelo TU se aplica a casos donde, por ejemplo, un proveedor de servicios realiza una inversión de costo $c(S)$ al construir una red que conecta a un conjunto de usuarios S a Internet, y quiere dividir este costo entre los usuarios que forman S. En la práctica, la función de costo c es muchas veces definida resolviendo

un problema de optimización combinatoria. La mayoría de los trabajos relacionados al reparto de costos en la literatura de algoritmos para teoría de juegos se enfocan a los juegos TU (Nisan *et al.*, 2007).

Un ejemplo de un caso en donde el modelo NTU es más aplicable, es cuando se tiene un problema de diseño de una red donde el costo incurrido por un agente i, que pertenece al conjunto S de agentes que reciben el servicio, corresponde al retraso (*delay*) que sufre este agente. Existen múltiples diseños para la red que conecta a los usuarios de S, y cada diseño corresponde a un perfil de retrasos que tendrán los agentes. El conjunto de todas las posibles soluciones para una coalición S se define como la colección de todos esos perfiles, y se denota por $C(S)$. Como los retrasos son no-transferibles, se modela este caso como un juego cooperativo NTU (Nisan *et al.*, 2007).

Algo central en la teoría de juegos cooperativos es la noción de **núcleo** (Moulin, 1995). De manera informal, el núcleo de un juego cooperativo es un resultado de cooperación entre todos los agentes donde no existe coalición en la que todos los agentes pueden beneficiarse al separarse de la coalición global. Intuitivamente, el núcleo de un juego corresponde a situaciones donde es posible sostener la cooperación entre todos los agentes de una manera económicamente estable (Gillies, 1959).

La definición formal es la siguiente: sea (A, c) un juego de reparto de costos TU. Un vector $\alpha \in \mathbb{R}^{|A|}$ (a veces llamado **asignación de costos**) está en el núcleo del juego si satisface las siguientes condiciones (Nisan *et al.*, 2007):

(i) Equilibrio presupuestario: $\sum_{j \in A} \alpha_j = c(A)$. Este concepto significa que a través de la asignación de costos a los usuarios de A, se recupera el costo de dar servicio a ese conjunto.

(ii) Propiedad de núcleo: para cada $S \subseteq A$, $\sum_{j \in S} \alpha_j \leq c(S)$. Este concepto significa

que no existe un subconjunto S donde la suma de sus contribuciones sea mayor al costo de dar servicio a ese subconjunto.

Un resultado clásico en la teoría de juegos cooperativos, conocido como el teorema de Bondareva-Shapley (Bondareva, 1963; Shapley, 1967), da una condición necesaria y suficiente para que un juego tenga un núcleo no vacío. Pero existe una dificultad con la noción de núcleo, pues en muchos problemas de reparto de costos, incluído la mayoría de los juegos de optimización combinatoria basados en problemas computacionalmente difíciles, el núcleo es vacío. Además, cuando la función de costo es difícil de calcular computacionalmente, la decisión acerca de la existencia de un núcleo vacío puede ser un problema NP-difícil (Nisan *et al.*, 2007; Jain y Vazirani, 2001a).

Por ello, se utiliza la definición de **núcleo aproximado**: un vector $\alpha \in \mathbb{R}^{|A|}$ está en el núcleo γ-aproximado (o γ-núcleo) si satisface las condiciones siguientes (Nisan *et al.*, 2007):

(i) γ-Equilibrio presupuestario: $\gamma c(A) \leq \sum_{j \in A} \alpha_j \leq c(A)$. En este caso, se recupera al menos γ del costo del servicio, y las contribuciones no superan el costo del mismo.

(ii) Propiedad de núcleo: para cada $S \subseteq A$, $\sum_{j \in S} \alpha_j \leq c(S)$. Como en el caso del núcleo, no existe un subconjunto S donde la suma de sus contribuciones sea mayor al costo de dar servicio a ese subconjunto.

El teorema de Bondareva-Shapley se puede generalizar para esta versión aproximada.

Muchos juegos cooperativos de optimización combinatoria se pueden expresar mediante un programa lineal entero (Jain y Vazirani, 2001a). Existe una fuerte conexión entre el núcleo de varios juegos de optimización combinatoria y el error de redondeo del

correspondiente programa lineal. Esta característica la observaron Deng *et al.* (1997). En los juegos de optimización combinatoria como el cubrimiento de conjunto, cubrimiento de vértices y ubicación de instalaciones, la formulación ILP es equivalente a la formulación estándar de dichos problemas. Por una propiedad de los núcleos aproximados, se tiene que el error de redondeo para la formulación ILP estándar es el valor de la aproximación del núcleo (Nisan *et al.*, 2007).

I.2 Problemas de reparto de costos

El problema de reparto de costos modela la forma de repartir, entre los usuarios, el costo en el que incurre un proveedor de servicios. El interés está en realizar un reparto de costos que motive a los usuarios a cooperar entre ellos, y de tal forma que cada usuario no pague más que el ingreso que dispone para recibir el servicio. Si se le pide pagar una cantidad mayor a su ingreso, prefiere no recibir el servicio (Pál y Tardos, 2003).

Sea A un conjunto de n usuarios o **agentes** interesados en recibir el servicio. El costo de proveerlo está dado por una función $c : 2^A \mapsto \mathbb{R}^+ \cup \{0\}$, donde $c(S)$ especifica el costo de proveer servicios a los agentes en $S \subseteq A$. Cada agente tiene un ingreso $u_i \in \mathbb{R}$ por recibir el servicio; es decir, está dispuesto a pagar a lo más u_i por recibir el servicio. El **beneficio** de un agente está dado por $u_i q_i - x_i$, donde x_i es el precio que debe pagar el agente i, y q_i es una variable que indica si el usuario recibe o no el servicio.

El ingreso u_i lo conoce únicamente el agente i. Por lo tanto, se puede ver al problema como una subasta en donde el proveedor ofrece un servicio a un determinado costo para cada usuario, y este último decide si desea el servicio teniendo en cuenta su ingreso. El

objetivo es entonces diseñar un mecanismo de subasta que incentive a los usuarios a revelar su verdadero ingreso (Nisan *et al.*, 2007).

Un **mecanismo de reparto de costos** determina cuáles usuarios recibirán el servicio y a qué precio (Jain y Vazirani, 2001a), dependiendo de la **oferta** $b_i \in \mathbb{R}$ realizada por el jugador i. Formalmente, un mecanismo de reparto de costos es una función que asocia a cada vector de ofertas $\mathbf{b}$ un conjunto $Q(\mathbf{b}) \subseteq A$ de jugadores que recibirán el servicio, y un vector $\mathbf{x}(\mathbf{b})$ de pagos. Cuando no exista ambigüedad, se puede escribir Q y $\mathbf{x}$ en lugar de $Q(\mathbf{b})$ y $\mathbf{x}(\mathbf{b})$, respectivamente. Se supone que el mecanismo cumple con las siguientes condiciones (Nisan *et al.*, 2007):

- *Transferencias no negativas (NPT)*: los pagos son no-negativos.
- *Participación voluntaria (VP)*: un usuario que no recibe el servicio, no paga; y un usuario que lo recibe, no paga más que su oferta.
- *Soberanía del consumidor (CS)*: para cada agente, existe una oferta para la cual recibirá el servicio, independientemente de las ofertas de otros usuarios.

El mecanismo es **a prueba de estrategias** si la estrategia dominante de cada agente es revelar el verdadero valor de su ingreso. Se dice que es **a prueba de estrategias de grupo** si lo mismo se mantiene para las coaliciones de agentes (Jain y Vazirani, 2001a). Esta última característica se define formalmente de la siguiente manera: sea $S \subseteq A$ una coalición de agentes, y $\mathbf{u}$, $\mathbf{u}'$ dos vectores de ofertas donde se cumple que $u_i = u'_i$ para todo $i \notin S$ (es decir, ofertan sus verdaderos ingresos). En un mecanismo a prueba de estrategias, para cada coalición S se tiene que: si la desigualdad $u'_i q'_i - x'_i \geq u_i q_i - x_i$ se cumple para cada $i \in S$, entonces se cumple con igualdad para cada $i \in S$. Es decir, no existe coalición S en donde, al ofertar los miembros el vector $\mathbf{u}'$ en lugar de $\mathbf{u}$, cada

uno reciba un beneficio mayor o igual que en el escenario verdadero, y que al menos uno tenga un beneficio estrictamente mayor (Nisan *et al.*, 2007).

Un mecanismo de reparto de costos es de **presupuesto equilibrado** si el precio total que se cobra a los agentes es igual al costo $c(S)$ en el que incurre el proveedor del servicio. Un mecanismo es **eficiente**, si maximiza el valor de los beneficios de los agentes (Jain y Vazirani, 2001a).

I.3 Algoritmos de aproximación

Muchos problemas en el campo de la teoría de juegos son **NP-difíciles**. Resulta natural entonces aplicar técnicas de algoritmos de aproximación a problemas de teorías de juegos, ya que comparten una metodología común basada en la teoría de programación lineal (Jain y Vazirani, 2001a).

Los algoritmos de aproximación son algoritmos utilizados para encontrar soluciones aproximadas para problemas de optimización, generalmente los pertenecientes a la clase NP-difícil. Para este tipo de problemas, no se conocen algoritmos que los resuelvan de manera óptima en un tiempo polinomial en el tamaño de los mismos. Los algoritmos de aproximación sacrifican optimalidad a cambio de correr en tiempo polinomial, y garantizan que la peor solución obtenida estará dentro de un factor de la solución óptima (Vazirani, 2001; Hochbaum, 1996; Ausiello *et al.*, 1999; Williamson y Shmoys, 2011).

I.4 Antecedentes

Idealmente, se busca un mecanismo de reparto de costos que sea eficiente, de presupuesto equilibrado y a prueba de estrategias de grupo. Pero un resultado clásico de teoría de juegos (Green *et al.*, 1976; Roberts, 1979) muestra que tal mecanismo no existe, incluso relajando la condición de ser a prueba de estrategias de grupo a sólo a prueba de estrategias (Jain y Vazirani, 2001a).

Teniendo en cuenta esta limitación, existen dos opciones: sacrificar el equilibrio en el presupuesto o la eficiencia. En el primer caso, se puede demostrar que si la función de costo c es decreciente y submodular (es decir, se tiene que $c(\oslash) = 0$ y para cualquier $Q_1, Q_2 \subseteq A \rightarrow c(Q_1) + c(Q_2) \geq c(Q_1 \cup Q_2) + c(Q_1 \cap Q_2))$ existe una sola forma de maximizar la eficiencia. Ésto se conoce como **mecanismo de costo marginal**. El mismo es a prueba de estrategias, pero no a prueba de estrategias de grupo (Moulin y Shenker, 2001).

En el segundo caso, un teorema fundamental de Moulin y Shenker (Moulin, 1999; Moulin y Shenker, 2001) muestra que un **método de reparto de costos monotónico cruzado** da lugar a mecanismos de reparto de costos de presupuesto equilibrado y a prueba de estrategias de grupo. Este concepto se utiliza ampliamente para tratar con problemas de reparto de costos y se lo conoce como **mecanismo de Moulin** (Moulin, 1999; Jain y Vazirani, 2001a; Pál y Tardos, 2003; Mehta *et al.*, 2007). El mecanismo de Moulin simula una subasta iterativa ascendente. En cada iteración, los precios se ofrecen simultáneamente a los jugadores restantes. Los jugadores que aceptan el precio, siguen en la subasta, los demás se remueven del conjunto. El mecanismo termina cuando todos los jugadores restantes aceptan el precio que se les ofrece en una iteración (Mehta *et al.*, 2007).

Los métodos monotónicos cruzados capturan la noción de que el agente no debe pagar más a medida que crece la cantidad de usuarios que reciben el servicio. Para definir esta propiedad, se define primero el concepto de esquema de reparto de costos: un **esquema de reparto de costos** es una función tal que para cada conjunto $S \subseteq A$, asigna un costo para cada integrante de S. Formalmente, es una función $\xi : A \times 2^A \mapsto \mathbb{R}$ tal que, para cada $S \subseteq A$ y cada $i \notin S$, $\xi(i, S) = 0$. Se dice que un esquema de reparto de costos ξ es de γ-presupuesto equilibrado si para cada conjunto $S \subseteq A$, se tiene que $\gamma c(S) \leq \sum_{i \in S} \xi(i, S) \leq c(S)$ (Nisan *et al.*, 2007).

Un método de reparto de costos es **monotónico cruzado** si para cada $J \subseteq J' \subseteq A$, se tiene que para un $i \in J$, $\xi(i, J) \geq \xi(i, J')$. La propiedad monotónica cruzada es crucial para incentivar la cooperación, pues motiva a los usuarios a invitar a otros para que se unan al servicio, pues con ello su aporte no puede aumentar (Pál y Tardos, 2003). Existe también el concepto de método **monotónico cruzado débil** si se cumple que para cada $J \subseteq J' \subseteq A$, $\sum_{i \in J} \xi(i, J) \geq \sum_{i \in J'} \xi(i, J')$ (Jain y Vazirani, 2001a).

La monotonicidad cruzada es un concepto más fuerte que el de núcleo, pues está demostrado que un esquema monotónico cruzado de γ-presupuesto equilibrado para un juego de reparto de costos (A, c), está en el núcleo del juego (Nisan *et al.*, 2007). Para cada función monotónica cruzada ξ, Moulin (1999); Moulin y Shenker (2001) dan un mecanismo para obtener el conjunto de usuarios que recibirán el servicio, y el costo para cada uno de ellos. Además, demuestran que este mecanismo es a prueba de estrategias de grupo, de γ-presupuesto equilibrado, NPT, CS y VP.

Algoritmo 1 Mecanismo de Moulin $M(\xi)$

Entrada: Conjunto de jugadores A, vector de apuestas $\mathbf{b} = (b_i)_{i \in A}$.
Salida: Conjunto de jugadores S que recibirán el servicio, vector de precios $\mathbf{p} = (p_i)_{i \in S}$.

1: Se inicializa $S = A$.
2: **si** $b_i \geq \xi(i, S)$ para cada usuario $i \in S$ **entonces**
3: **devolver** el conjunto S, la solución factible construída por ξ, y se cobra a cada jugador $i \in S$ el precio $p_i = \xi(i, S)$.
4: **si no**
5: Se elige en forma arbitraria un usuario i con $b_i < \xi(i, S)$ y se hace $S = S - \{i\}$. Se repite el paso 2.
6: **fin si**

La asignación de costos en el núcleo aproximado de un juego se puede calcular resolviendo un problema de programación lineal entera (ILP), y para muchos juegos de optimización combinatoria, este ILP es equivalente al dual de la relajación del ILP estándar del problema, y los costos repartidos corresponden a estas variables duales (Nisan *et al.*, 2007). Existe una técnica llamada **esquema primal-dual** que se usa para calcular repartos de costos que no sólo se encuentran en el núcleo aproximado del problema, sino que también satisfacen la propiedad de monotonicidad cruzada. El esquema primal-dual es una técnica estándar en el campo de algoritmos de aproximación, donde el objetivo es obtener una solución aproximada en relación a la óptima primal, y las variables duales son sólo un producto del algoritmo (Vazirani, 2001).

Existe una larga historia de conexiones entre el reparto de costos, diseño de mecanismos, y el método primal-dual (Mehta *et al.*, 2007; Jain y Vazirani, 2001a; Pál y Tardos, 2003; Devanur *et al.*, 2005; Jain y Vazirani, 2002; Parkes y Ungar, 2000). La idea del esquema primal-dual, que a menudo se utiliza para resolver problemas de minimización, es escribir el problema de optimización como un programa matemático que se puede relajar a un Programa Lineal (llamado LP primal). El dual de este LP da una cota inferior en el valor de la solución optimal del problema. El algoritmo primal-dual construye

simultáneamente una solución para el problema primal y su dual (Vazirani, 2001). Ésto se hace generalmente colocando al inicio todas las variables duales a cero, y luego se incrementan gradualmente estas variables hasta que alguna restricción del programa dual esté ajustada. Esta restricción apunta a un objeto que puede pagarse por el dual para incluirlo en la solución primal. Luego de ésto, las variables duales relacionadas a la restricción ajustada se congelan, y el algoritmo continúa incrementando las otras variables duales. El algoritmo termina cuando se obtiene una solución completa para el problema primal (Nisan *et al.*, 2007). El análisis está basado en probar que los valores de las soluciones del primal y dual construídas están cerca entre sí, y por lo tanto ambos están cerca del óptimo (Vazirani, 2001). En muchos algoritmos primal-dual, se requiere un procesamiento posterior de los resultados para bajar aún más el costo de la solución primal (Pál y Tardos, 2003; Mehta *et al.*, 2007).

I.5 Ejemplos de problemas de reparto de costos

A continuación se definen dos problemas de reparto de costos en juegos cooperativos, detallando en el Apéndice A un método de solución propuesto en la literatura (Jain y Vazirani, 2001a; Pál y Tardos, 2003) para cada uno. Además, en dicho apéndice se muestra un ejemplo de aplicación para cada método de solución. Estos dos problemas son importantes, pues el diseño de una red de compra-alquiler está relacionado a ellos.

I.5.1 Juego del árbol de Steiner

En el trabajo de Jain y Vazirani (2001a) se considera primeramente el **problema de reparto de costos del ruteo multicast**. Se tiene un conjunto U de potenciales usuarios, un grafo $G = (V, E)$ donde cada vértice representa un usuario, y las aristas

del grafo tienen asociadas un costo. Por ejemplo, considérese que un proveedor desea ofrecer un servicio a un conjunto de usuarios. Para cada conjunto $S \subseteq U$, $C(S)$ denota el costo en el que incurre el proveedor para dar servicio a los usuarios de S. Cada usuario i tiene un ingreso u_i por recibir el servicio que sólo él conoce; es decir, está dispuesto a pagar a lo más u_i. El usuario i tiene un beneficio $u_i - x_i$ por obtener el servicio, el cual se le ofrece a un precio x_i; si no lo recibe, su beneficio es 0. Se considera que cada usuario es egoísta, por lo que puede dar un falso ingreso u_i' para maximizar su beneficio.

El objetivo es determinar el conjunto de usuarios $Q \subseteq U$ que recibirán el servicio y a qué precio. Además, se debe determinar un árbol T de G, que contiene a Q y a la fuente, a través del cual se encaminará la información para dar servicio a Q.

Este problema se resuelve proponiendo un método de reparto de costos a partir del algoritmo primal-dual de Edmonds (1967) para el árbol de Steiner. El método es monotónico-cruzado, y de 2-aproximación en el equilibrio de presupuesto.

I.5.2 Juego de ubicación de instalaciones

La definición es la siguiente (Nisan *et al.*, 2007): se tiene un conjunto A de *agentes* (también conocidos como ciudades, clientes o puntos de demanda), un conjunto F de instalaciones, el costo de apertura de una instalación f_i para cada instalación $i \in F$, y una distancia d_{ij} entre cada par (i, j) de puntos en $A \cup F$ indicando el costo de conectar j con i. Se supone que las distancias provienen de un espacio métrico, es decir, son simétricos y cumplen con la desigualdad del triángulo. Para un conjunto $S \subseteq A$ de agentes, el costo de este conjunto se define como el costo mínimo de abrir un conjunto de instalaciones y conectar cada agente de S a una instalación abierta. Formalmente,

la función de costo c se define como $c(S) = \min_{F' \subseteq F}\{\sum_{i \in F'} f_i + \sum_{j \in S} \min_{i \in F'} d_{ij}\}$.

Al ser un problema de reparto de costos en un juego cooperativo, el mismo tiene muchas similitudes en cuanto a los objetivos y restricciones señalados en el juego del árbol de Steiner. Como en el juego anterior, el objetivo principal es obtener un método de reparto de costos que sea monotónico cruzado, para emplear nuevamente el mecanismo de Moulin (1999). La optimalidad para este problema consiste en obtener una solución de ubicación de instalaciones de menor costo posible para dar servicio a todos los usuarios que aún participan en el juego. Como obtener una solución óptima para el costo del servicio es **NP-difícil** (Goemans y Skutella, 2000), se utilizan métodos aproximados. Existen varios algoritmos de aproximación para el problema de ubicación de instalaciones (Mahdian *et al.*, 2006; Jain y Vazirani, 2001b; Mettu y Plaxton, 2000).

En el trabajo de Pál y Tardos (2003) se propone un método de reparto de costos a través de la variación del método primal-dual, denominado "proceso fantasma". Este proceso obtiene un factor de aproximación de 3 en el equilibrio del presupuesto, y es el mejor posible para un mecanismo de Moulin.

I.6 Motivación

Los problemas de reparto de costos tienen importancia en problemas del mundo real donde se busca distribuir, a través de ciertas condiciones, el costo de un servicio utilizado en conjunto. Los mecanismos determinan los usuarios que recibirán el servicio y el precio que deberán pagar, por lo que un buen diseño de los mismos tiene un papel importante en el proceso.

El problema de reparto de costos en el diseño de una red de compra-alquiler de una sola fuente, así como las variaciones y generalizaciones del mismo, tienen aplicación

principalmente en el diseño de redes que sean económicamente escalables. En el trabajo de Andrews y Zhang (1998) se menciona que puede aplicarse al diseño de redes telefónicas. Por lo tanto puede aplicarse también para redes de datos, como Internet.

Los mecanismos deben utilizar algoritmos que permitan obtener una solución para el conjunto de usuarios participantes, y en la mayoría de los casos estos problemas son **NP-difíciles**, por lo que se emplean los algoritmos de aproximación. Por lo tanto, es importante analizar la influencia que tienen estos algoritmos en el funcionamiento de los mecanismos que los utilizan.

Los mecanismos de Moulin se aplican tradicionalmente a los problemas de reparto de costos. Los conceptos de mecanismos acíclicos e incrementales son relativamente recientes, y mejoran los resultados obtenidos mediante mecanismos de Moulin para problemas básicos de reparto de costos. No se conocen trabajos que empleen mecanismos acíclicos ni incrementales para el problema de diseño de una red compra-alquiler de una sola fuente; por lo cual se podrían tener ventajas en relación a los que emplean el mecanismo de Moulin. Ésto permitirá a los diseñadores contar con un método más al momento de determinar el mecanismo de reparto de los costos.

Desde el punto de vista económico, idealmente se busca un mecanismo de reparto de costos que sea eficiente, de presupuesto equilibrado y a prueba de estrategias de grupo. Como es imposible obtener tal mecanismo, se utiliza el concepto de equilibrio de presupuesto aproximado. La mayoría de los trabajos se enfocan en este último, dejando de lado el análisis de eficiencia. Es interesante estudiar y analizar acerca del compromiso entre la eficiencia y el equilibrio de presupuesto aproximado, y cómo éste lo determina el tipo de mecanismo de reparto de costos utilizado.

I.7 Objetivos

I.7.1 Objetivo general

- Analizar, caracterizar y proponer algoritmos basados en mecanismos acíclicos aplicados al problema de reparto de costos en el diseño de una red de compra-alquiler de una sola fuente.

I.7.2 Objetivos específicos

- Analizar el enfoque primal-dual y sus variantes para el diseño de algoritmos de aproximación.
- Analizar los mecanismos de Moulin, los mecanismos acíclicos y los mecanismos incrementales de reparto de costos.
- Analizar la aplicación de tales mecanismos a los problemas de reparto de costos.
- Determinar cómo y bajo qué condiciones se aplican las técnicas del área de algoritmos de aproximación a los problemas de reparto de costos.
- Determinar la relación que existe entre los conceptos de eficiencia y equilibrio aproximado del presupuesto.
- Entender el problema de diseño de una red de compra-alquiler en juegos cooperativos.
- Analizar los mejores mecanismos de reparto de costos para el problema de diseño de una red de compra-alquiler de una sola fuente. Analizar sus resultados en cuanto a la eficiencia.

- Analizar los mejores algoritmos para el problema combinatorio subyacente de diseño de una red de compra-alquiler de una sola fuente.

- Determinar qué ventajas existen al aplicar los mecanismos acíclicos al problema de diseño de una red de compra-alquiler de una sola fuente.

I.8 Organización de la tesis

Esta tesis contiene seis capítulos y tres apéndices, organizados de la siguiente forma:

En el Capítulo II se define el problema del diseño de una red de compra-alquiler. Se lo formula como un problema de programación lineal entera, y se muestra la relajación del problema primal para definir el problema dual. Luego, se presenta un algoritmo primal-dual (Swamy y Kumar, 2004) que obtiene soluciones para el problema combinatorio subyacente.

En el Capítulo III se analiza la posibilidad de utilizar el algoritmo de Swamy y Kumar (2004) en un mecanismo incremental, y posteriormente en un mecanismo acíclico. Ésto se hace para mejorar el mejor resultado conocido para el problema de reparto de costos en el diseño de una red de compra-alquiler (Gupta *et al.*, 2008).

En el Capítulo IV se presentan otros algoritmos para el problema combinatorio de diseño de una red compra-alquiler. Estos algoritmos se basan principalmente en el algoritmo aleatorio propuesto por Gupta *et al.* (2003), el cual puede utilizarse en un mecanismo incremental. Otro de ellos es el utilizado en el mecanismo que da el mejor resultado para el problema de reparto de costos considerado, y se analiza su aplicación dentro de un mecanismo incremental de reparto de costos. También se presenta una proposición, que al demostrarla, garantizará la aplicación exitosa del algoritmo de (Gupta *et al.*, 2008) en un mecanismo incremental.

En el Capítulo V se presentan los experimentos que se realizan para sustentar la proposición hecha en el capítulo anterior, y se realiza una comparación de desempeño de los mecanismos de Moulin e incremental en relación al costo social.

En el Capítulo VI se hace un resumen de los puntos más importantes del trabajo, presentando las conclusiones e ideas para trabajos futuros.

En el Apéndice A se presentan dos ejemplos de problemas de reparto de costos: el árbol de Steiner y el problema de ubicación de instalaciones.

En el Apéndice B se presenta el análisis del algoritmo de Swamy y Kumar (2004) que se utiliza en los capítulos II y III.

Finalmente, en el Apéndice C se presenta un mecanismo de reparto de costos que emplea una variación del algoritmo de Swamy y Kumar (2004) para obtener un método de reparto de costos.

Capítulo II

Planteamiento del problema

En este capítulo se presenta el problema de reparto de costos a estudiar, que consiste en el diseño de una red compra-alquiler de una sola fuente. Inicialmente, se considera únicamente la optimización de la red construida, describiendo un método primal-dual para obtener soluciones aproximadas (Swamy y Kumar, 2004). Luego, se considera el problema de reparto de costos (Apéndice C), describiendo un método primal-dual que genera una función monotónica cruzada (Pál y Tardos, 2003).

II.1 Problema de ubicación de instalaciones conectadas

Como se había visto anteriormente, el problema de ubicación de instalaciones se puede describir de la siguiente manera: se tiene un grafo $G = (V, E)$, un conjunto de instalaciones $\mathcal{F} \subseteq V$, y un conjunto de clientes $\mathcal{D} \subseteq V$. Se desea determinar un subconjunto de instalaciones de $\mathcal{F}$ que se deben abrir y asignar los usuarios a las instalaciones abiertas más cercanas, de tal forma que el costo total sea mínimo.

Muchas aplicaciones prácticas requieren además que las instalaciones abiertas estén conectadas entre sí. En este caso, se desea una solución en la que los clientes se agrupen en conglomerados alrededor de las instalaciones, y que éstas estén interconectadas para permitir la comunicación entre las mismas. Este concepto se utiliza en el diseño de redes de telecomunicaciones (Andrews y Zhang, 1998; Ravi y Selman, 1999). Un

modelo común de una red de telecomunicaciones consiste de un núcleo central y nodos terminales. El núcleo consiste de un conjunto de nodos interconectados que tienen capacidad de conmutación. Cada nodo del núcleo incurre en un costo de conmutación. El diseño de la red consiste en seleccionar un subconjunto de nodos de núcleo, conectar los nodos seleccionados, y encaminar el tráfico desde los nodos terminales hasta los nodos de núcleo seleccionados. El costo de abrir una instalación corresponde al costo de conmutación del correspondiente nodo de núcleo (Swamy y Kumar, 2004).

Se requiere entonces que las instalaciones abiertas (vértices requeridos) estén conectadas entre ellas mediante un árbol de Steiner, es decir, el árbol que las conecta puede incluir nodos terminales (vértices Steiner). A este problema se le denomina juego de instalaciones conectadas (CFL, por sus siglas en inglés). Formalmente, se tiene un grafo $G = (V, E)$ con costos c_e de las aristas, un conjunto de instalaciones $\mathcal{F} \subseteq V$, y un conjunto de vértices de demanda o clientes $\mathcal{D} \subseteq V$. El cliente j tiene d_j unidades de demanda y la instalación i tiene un costo de apertura f_i, y sea $M \geq 1$ un parámetro. Una solución del CFL abre un subconjunto $F \subseteq \mathcal{F}$ de instalaciones y asigna cada demanda a una instalación abierta. Sea c_{ij} la menor distancia entre los vértices i y j en G (respecto a los costos c_e). Si el cliente j se asigna a la instalación $i(j)$, se incurre en un costo de asignación proporcional a la demanda d_j y la distancia $c_{i(j)j}$. Luego, la solución debe conectar las instalaciones abiertas mediante un árbol de Steiner T. El costo de conectar las instalaciones es simplemente el costo del árbol de Steiner T multiplicado por un factor M. En una red de telecomunicaciones, el parámetro M indica los costos más caros de conectar los nodos de núcleo mediante enlaces de ancho de banda alto. El costo total de la solución es la suma del costo de abrir las instalaciones en F, los costos de asignación de clientes, y el costo de conectar las instalaciones abiertas (Swamy y

Kumar, 2004). Más precisamente, el costo de esta solución es:

$$\sum_{i \in F} f_i + \sum_{j \in \mathcal{D}} d_j c_{i(j)j} + M \sum_{e \in T} c_e$$

El objetivo es hallar una solución de costo mínimo. Este problema atrae el interés tanto de la comunidad de investigación de operaciones (Kim *et al.*, 1996; Labbé *et al.*, 2001; Lee *et al.*, 1996) como el de la comunidad de ciencias de la computación (Gupta *et al.*, 2003; Karger y Minkoff, 2000; Swamy y Kumar, 2004).

II.2 Diseño de redes de compra-alquiler de una sola fuente (SSROB): Antecedentes

Un caso particular del CFL es cuando todos los costos de apertura son iguales a 0 y las instalaciones se pueden abrir en cualquier vértice, ésto es, $\mathcal{F} = V$. Supóngase que se sabe de antemano que una instalación se abre en una solución óptima. Entonces este problema se conoce como el problema de compra-alquiler de una sola fuente (SSROB por sus siglas en inglés).

Se tiene un vértice designado para la *fuente* s que transmite un mensaje. El objetivo es construir una red que conecta cada usuario a la fuente s a través de un camino. Cada arista e de la red se puede *comprar* a un costo Mc_e o *alquilar* a un costo c_e. Una arista comprada en la red puede dar servicio a cualquier número de caminos, mientras que en una alquilada se debe pagar un costo c_e por cada camino que utilice esa arista. Una red es factible si puede dar servicio a un camino desde cada usuario j a la fuente s. El costo de la red es la suma de los costos de todas las aristas compradas y alquiladas (Pál y Tardos, 2003).

Las aristas compradas en la red óptima de compra-alquiler deben formar un árbol

de Steiner con s como raíz, y las aristas alquiladas deben formar la conexión más corta desde cada usuario j al punto más cercano del árbol. Existen varios algoritmos de aproximación (Karger y Minkoff, 2000; Guha *et al.*, 2000; Swamy y Kumar, 2004) para este problema que explotan la estructura óptima para construir una solución en dos fases: primero se usa un algoritmo parecido al del problema de ubicación de instalaciones para juntar a los usuarios en unas pocas *ubicaciones centrales*, y alquilar un camino desde cada usuario al centro más cercano. Cuando se junta suficiente demanda en un centro, se construye un árbol de Steiner, conectando todos los centros a la fuente (Pál y Tardos, 2003).

El SSROB constituye un caso especial del problema de compra al por mayor de una sola fuente (SSBB por sus siglas en inglés). La diferencia con el SSROB es que en el SSBB se tienen K tipos de cables, cada uno con una capacidad determinada y un precio por unidad de longitud (Gupta *et al.*, 2003). El objetivo es encontrar una solución de costo mínimo que tenga suficiente capacidad en las aristas de manera a que el flujo de datos se pueda transmitir simultáneamente desde la fuente a los usuarios. Por otro lado, el problema del árbol de Steiner es un caso especial del SSROB, donde se tiene un valor de $M = 1$.

El CFL tiene relación con el problema de ubicación de instalaciones y el problema del árbol de Steiner. Sin el requerimiento de la conexión entre instalaciones, el problema se convierte sólo en la ubicación de instalaciones. Una vez determinadas las instalaciones a abrir, se puede asignar cada demanda a la instalación abierta más cercana y conectar las instalaciones abiertas mediante un árbol de Steiner (Swamy y Kumar, 2004).

Una estrategia simple que primero decide cuáles instalaciones abrir (ejecutando un algoritmo para el problema de ubicación de instalaciones), y que luego conecta las instalaciones abiertas a través de un árbol de Steiner, tiene un rendimiento pobre.

Por ejemplo, para el SSROB, ésto haría que se abra una instalación en cada punto de demanda, pero conectar todas las instalaciones puede significar un costo muy alto. Entonces existe un costo implícito para cada instalación dado por el requisito de conectividad (Swamy y Kumar, 2004). Para asegurar la viabilidad económica al abrir una instalación, se necesita agrupar suficiente demanda en la instalación. En (Guha *et al.*, 2000; Karger y Minkoff, 2000) el agrupamiento se logra resolviendo un caso del balanceo de carga en la ubicación de instalaciones (LBFL por sus siglas en inglés), donde se desea que cada instalación abierta atienda al menos M clientes. La desventaja de este método es que al reducirlo al LBFL se desperdicia información específica del problema (Swamy y Kumar, 2004).

A continuación, se describe el algoritmo propuesto por Swamy y Kumar (2004) para el SSROB. Dicho algoritmo se basa en el esquema primal-dual. La idea básica es considerar una formulación del problema como un caso de programación lineal entera y el dual de su relajación a programa lineal. El programa lineal dual se puede separar en dos partes: una parte parecida al dual del problema de ubicación de instalaciones, y la otra parecida al dual del problema del árbol de Steiner. El algoritmo tiene dos fases; la primera consiste en determinar qué instalaciones abrir, conectar las demandas a las instalaciones abiertas, y agrupar las demandas en cada instalación. Al final de esta fase se obtiene una solución primal a la ubicación de instalaciones, y una solución dual factible. En la solución primal las demandas se agrupan de tal forma que cada instalación abierta sirva a por lo menos M clientes, satisfaciendo la cota inferior de demanda. Lo anterior se hace cobrando parte del costo en el que se incurre en la porción del árbol de Steiner de la solución dual, de tal forma a aprovechar el hecho que cualquier solución al SSROB debe conectar las instalaciones que abre. El algoritmo tiene una descripción simple: cada cliente j aumenta su variable dual α_j hasta que se conecte a una

instalación donde M demandas se agrupen. Las otras variables simplemente responden a este cambio tratando de mantener la factibilidad o la holgura complementaria. La segunda fase consiste en construir un árbol de Steiner donde se conectan las instalaciones abiertas (Swamy y Kumar, 2004).

II.3 Formulación matemática del SSROB

En lo que sigue, i se usa para indicar instalaciones, j para indicar los clientes, y e para indicar las aristas de G. También se usan indistintamente los términos cliente y punto de demanda, y cada vértice j tiene d_j unidades de demanda. El valor c_e indica el costo de la arista e, y c_{ij} es el costo del camino más corto entre los vértices i y j. La constante M es el factor de multiplicación para la compra de una arista. La fuente s se considera como una instalación abierta y pertenece al árbol de Steiner construido en la solución óptima. El SSROB se puede formular naturalmente como un programa lineal entero de la siguiente manera (Swamy y Kumar, 2004):

$$\text{minimizar } C = \sum_{j \in \mathcal{D}} d_j \sum_{i \in V} c_{ij} x_{ij} + M \sum_{e \in E} c_e z_e \tag{1}$$

$$\begin{aligned}
\text{sujeto a:} \quad & \sum_{i \in V} x_{ij} \geq 1 && \forall\, j \in \mathcal{D} \\
& x_{ij} \leq y_i && \forall\, i \in V, j \in \mathcal{D} \\
& y_s = 1 && \\
& \sum_{i \in S} x_{ij} \leq \sum_{e \in \delta(S)} z_e && \forall\, S \subseteq V \backslash \{s\}, j \in \mathcal{D} \\
& x_{ij}, y_i, z_e \in \{0, 1\} &&
\end{aligned}$$

y_i indica si la instalación i está abierta o no, x_{ij} indica si el cliente j se conecta a la instalación i, y z_e indica si la arista e se incluye en el árbol de Steiner o no. Se tiene que $e \in \delta(S)$ si e tiene un extremo en S y el otro en $V \backslash S$. La primera restricción indica que cada usuario debe asignarse a al menos una instalación. La segunda indica que un usuario debe asignarse a una instalación que está abierta. La tercera indica que la instalación s está abierta. La cuarta restricción indica que debe haber un camino desde el usuario j hasta la fuente s. Relajando la restricción de que estas variables sean enteras a que $x_{ij}, y_i, z_e \geq 0$, se tiene un caso de programación lineal (LP).

Al considerar el problema SSROB, se tiene que los costos de apertura de centros son iguales a 0 y que $\mathcal{F} = V$, es decir, una instalación se puede abrir en cualquier vértice de V. Por simplicidad en la explicación, se supone que todos los d_j son iguales a 1. El programa lineal (LP) correspondiente queda como (Swamy y Kumar, 2004):

$$\min \sum_{j \in \mathcal{D}} \sum_{i \in V} c_{ij} x_{ij} + M \sum_{e \in E} c_e z_e \tag{2}$$

$$\begin{aligned} \text{sujeto a:} \quad & \sum_{i \in V} x_{ij} \geq 1 && \forall\, j \in \mathcal{D} \\ & \sum_{i \in S} x_{ij} \leq \sum_{e \in \delta(S)} z_e && \forall\, S \subseteq V \backslash \{s\}, j \in \mathcal{D} \\ & x_{ij}, z_e \geq 0 \end{aligned}$$

Las restricciones son parte de las que se tienen para el CFL. El dual del programa lineal es:

$$\max \sum_{j \in \mathcal{D}} \alpha_j \tag{3}$$

$$\text{sujeto a:} \quad \alpha_j \leq c_{ij} + \sum_{S \subseteq V \setminus \{s\}: i \in S} \theta_{S,j} \qquad \forall\, i \in V, i \neq s, j \in \mathcal{D} \tag{4}$$

$$\alpha_j \leq c_{sj} \qquad \forall\, j \in \mathcal{D} \tag{5}$$

$$\sum_{j \in \mathcal{D}} \sum_{S \subseteq V \setminus \{s\}: e \in \delta(S)} \theta_{S,j} \leq M c_e \quad \forall\, e \in E \tag{6}$$

$$\alpha_j, \theta_{S,j} \geq 0 \tag{7}$$

Intuitivamente, α_j es la contribución que realiza el cliente j para construir una solución primal factible. La restricción (4) indica que una parte de la contribución α_j se utiliza para pagar el costo de asignación del usuario j a la instalación i, mientras que lo restante se utiliza para la construcción de la parte del árbol de Steiner que une la instalación i con la fuente s. La restricción (5) indica que la contribución del usuario j no debe ser mayor que el costo del camino más corto entre él y la fuente s. La restricción (6) indica que la suma de las contribuciones de los usuarios a una arista e no debe superar el costo de compra de la misma. La restricción (7) indica que las variables duales deben ser no-negativas (Swamy y Kumar, 2004).

Antes de explicar el algoritmo propuesto por Swamy y Kumar (2004) se realiza una suposición que simplifica el problema. Se supone que una instalación se puede abrir en cualquier lugar a lo largo de una arista. Se define a los vértices y a los puntos internos de las aristas como *localidades*, y se reserva el término instalación para un vértice de V. Se puede suponer que para cualquier arista $e = (u, w)$, c_e es igual a c_{uw}, el camino más corto de u a w. La métrica c se extiende a una métrica de localidades considerando que e se compone de un número infinito de aristas de longitud infinitesimal. Así, para puntos p en e, la distancia c_{up} varía contínuamente y monotónicamente desde 0 hasta c_e al ir desde u hasta w, y $c_{wp} = c_e - c_{up}$. Para cualquier otro vértice $r \neq u, w$, se tiene que $c_{rp} = \min(c_{ru} + c_{up}, c_{rw} + c_{wp})$. Finalmente, para cualquier par de puntos p y q en

aristas $e_1 = (u, w)$ y e_2 respectivamente, se tiene que $c_{pq} = \min(c_{uq} + c_{up}, c_{wq} + c_{wp})$. También puede considerarse α_j como el radio de una esfera que tiene como centro a la demanda j.

II.3.1 Algoritmo de Swamy y Kumar (2004) para el SSROB

El razonamiento intuitivo detrás del algoritmo propuesto por Swamy y Kumar (2004) es el siguiente: supóngase que cada cliente tiene al menos M unidades de demanda. Entonces, la solución óptima ubicaría una instalación en cada uno de esos clientes y los conectaría a través de un árbol de Steiner. Entonces, el algoritmo se ejecuta en dos fases: en la primera agrupa las demandas en conglomerados de tamaño M y en la segunda se construye un árbol de Steiner juntando estos conglomerados. Inicialmente, todas las variables duales son 0.

Fase 1

Se incrementan las variables α_j para todas las demandas en esta fase. Se tiene una noción de tiempo t, inicialmente igual a 0. A medida que aumenta el tiempo, se incrementan las variables α_j a la misma razón que el tiempo. Se deben determinar localidades *tentativamente abiertas*; en $t = 0$, la fuente s está tentativamente abierta y todas las demás localidades están cerradas.

En algún instante de tiempo, se dice que una demanda j está *ajustada* con una localidad i si $\alpha_j \geq c_{ij}$. Sea S_j el conjunto de vértices con los que j está ajustada en algún instante de tiempo. Cuando se incrementa α_j, también se incrementa $\theta_{S_j,j}$ a la misma razón. Lo anterior asegura la factibilidad de la restricción (4). Por lo tanto, es suficiente describir cómo incrementar las variables duales α_j.

Las demandas pueden tener dos estados: *congeladas* o *no-congeladas*. Cuando una

demanda j se congela, se detiene el incremento de su variable dual α_j. Entonces, si la demanda j no está congelada en un tiempo t, se tiene que $\alpha_j = t$. Después de que j se congele, no se vuelve ajustada con otra localidad nueva, es decir, con una localidad que no está en S_j. Inicialmente, ninguna demanda está congelada.

Se comienza a incrementar las α_j de todas las demandas a la misma razón que el tiempo t, hasta que ocurra alguno de los siguientes eventos (si varios eventos ocurren, considerarlos en cualquier orden):

1. j se vuelve ajustada con una localidad tentativamente abierta i: entonces j se vuelve ajustada.

2. Existe una localidad cerrada i con la que al menos M demandas están ajustadas (se interceptan M o más esferas): entonces se abre tentativamente i, y se congelan todos los puntos de demanda que están ajustadas con i.

Ahora se incrementan sólo los valores de α_j de las demandas no-congeladas. Se continúa con este proceso hasta que todas las demandas se congelen. La Figura 1 muestra un ejemplo de la ejecución del algoritmo para $M = 2$, y cinco puntos de demanda. Se puede notar que aunque existen puntos continuos a lo largo de la arista, la implementación del proceso solo requiere conocer el tiempo en que ocurrirá el siguiente evento. Lo anterior se puede obtener monitoreando, para cada arista y para cada demanda j, la porción de la arista que está ajustada con j (Swamy y Kumar, 2004).

Luego se decide las localidades que se abrirán. Sea L el conjunto de localidades tentativamente abiertas. Se dice que $i, i' \in L$ son dependientes si existe una demanda j que está ajustada con ambas localidades. Se dice que un conjunto de localidades es *independiente* si ningún par de localidades de este conjunto es dependiente. Se halla un conjunto de tamaño máximo L' de ubicaciones en L de la siguiente forma: se ordenan

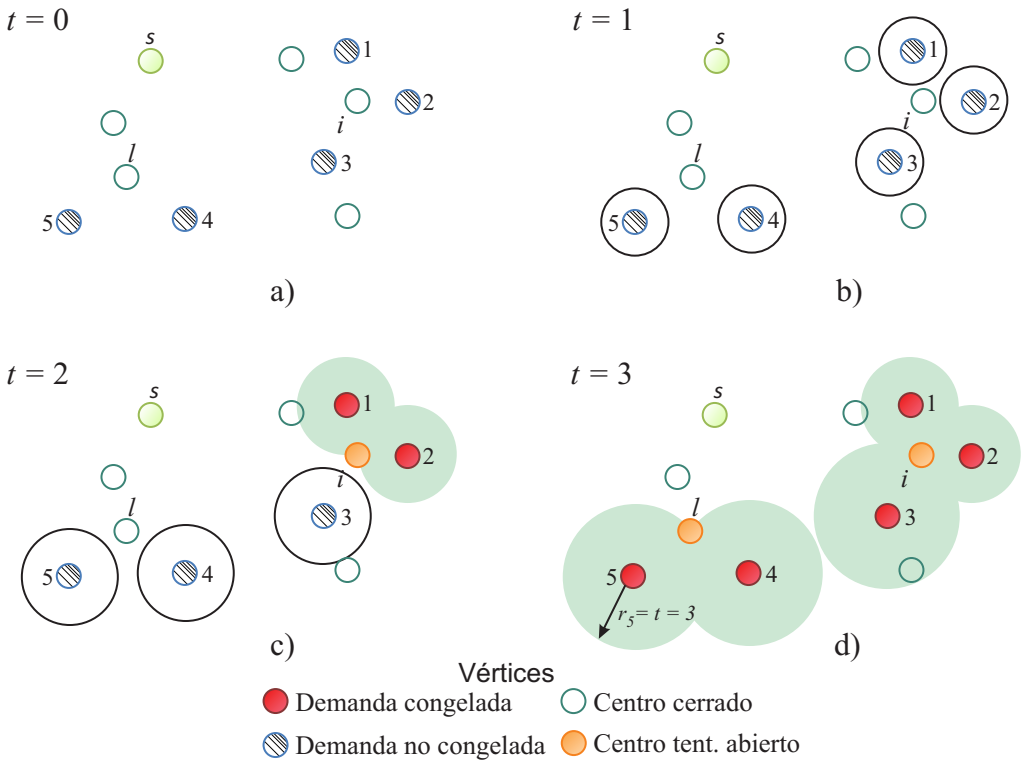


Figura 1. Un ejemplo de ejecución con $M = 2$. (a) El estado inicial, (b) $t = 1$, (c) i se vuelve ajustada con las demandas 1 y 2; i es tentativamente abierta y 1, 2 se congelan, (d) La solución final. La demanda 3 alcanza i y se congela; l se vuelve ajustada con las demandas 4 y 5, y es tentativamente abierta, ocasionando el congelamiento de 4 y 5.

las localidades en L en el orden en que tentativamente se abrieron. Se consideran las localidades en este orden y se agrega una localidad a L' si no existe una localidad dependiente que ya esté en L'; luego se abren las localidades en L'. Se puede observar que $s \in L'$, dado que s es la primera localidad tentativamente abierta.

Se asigna una demanda j a una localidad abierta de la siguiente manera: si j está ajustada con algún $i \in L'$, se asigna j a i; de otra forma, sea i la localidad en L que causó el congelamiento de j. Debe existir una localidad previamente abierta $i' \in L'$ tal que i y i' sean dependientes. Entonces, se asigna j a i'. Sea $\sigma(j)$ la localidad a la que j se asigna.

Ahora se debe construir un árbol de Steiner donde los vértices de L' son los vértices

requeridos. Primero se aumenta el grafo G de manera a incluir aristas incidentes a localidades que no sean vértices. Sean $\{i_1, ..., i_k\}$ las localidades abiertas que están en una arista $e = (u, w)$ ordenadas por distancia creciente a u, con $i_1 \neq u$, $i_k \neq w$. Se agregan las aristas $(u, i_1), (i_1, i_2), ..., (i_{k-1}, i_k), (i_k, w)$ a G (Swamy y Kumar, 2004).

Fase 2

Para una localidad $i \in L'$, sea D_i el conjunto de demandas ajustadas con i. Sea $D' = \bigcup_{i \in L'-\{s\}} D_i$. Primero, se hace $\alpha_j = 0$ para todo j. Se aumenta el valor de α_j de las demandas de D' solamente, y se simula el algoritmo primal-dual para el árbol de Steiner enraizado.

Se dice que un conjunto $S \subseteq V$ es violado si $\delta(S) = \emptyset$. Inicialmente, los conjuntos minimales violados (MVS por sus siglas en inglés) son los conjuntos de un solo elemento $\{i\}$ para $i \in L' - \{s\}$. Para un conjunto S, se define $D_S = \bigcup_{i \in S \cap L'} D_i$. El árbol T que se debe construir está vacío al comenzar. Para cada MVS S, $j \in D_S$, se aumenta α_j a una razón de $1/|D_S|$. También se aumenta $\theta_{S,j}$ a la misma razón. Lo anterior asegura que la cantidad $\sum_j \theta_{S,j}$ crece a una razón de 1 por unidad de tiempo para cualquier MVS S. Se debe notar que ésto no asegura la factibilidad de las restricciones (4) y (5).

Se dice que una arista e se vuelve ajustada si (6) se cumple con una igualdad para esta arista. Se aumentan las variables duales hasta que una arista e se vuelve ajustada. Se agrega e a T y se actualiza el MVS. Este proceso continúa hasta que no existan conjuntos violados, es decir, se tiene un solo componente (entonces s está en este componente). Ahora se consideran las aristas de T en el orden inverso al que se agregaron, y se remueven las aristas redundantes (ver Apéndice A) (Swamy y Kumar, 2004). Así se genera la solución final.

En el Apéndice B se detalla el análisis del algoritmo y muestra que el factor de

aproximación es de 5 (obteniendo el árbol de Steiner mediante el algoritmo de Prim (1957)), o de 4.55 (mediante el algoritmo de Robins y Zelikovsky (2000) para el árbol de Steiner). En el Algoritmo 2 se muestra un pseudocódigo de las dos fases consideradas.

Algoritmo 2 Algoritmo primal-dual para el SSROB

Entrada: Grafo $G = (V, E)$, demandas $D \subseteq V$, fuente s, parámetro $M > 1$.
Salida: Árbol de Steiner T, instalaciones L' y las asignaciones $\sigma(j)$.

1: {**Fase 1**}; (**Inicialización**) $L \leftarrow \{s\}$, $\mathcal{C} \leftarrow \mathcal{D}$
2: **mientras** $\mathcal{C} \neq \emptyset$ **hacer**
3: Aumentar el tiempo t y las variables duales no-congeladas a razón de t hasta que ocurra uno de los siguientes eventos:
4: **si** $j \in \mathcal{C}$ **y** $\alpha_j \geq c_{ij}$ **entonces**
5: $D_i \leftarrow D_i \cup \{j\}$
6: **fin si**
7: **si** $i \in L$ **entonces**
8: $\mathcal{C} \leftarrow \mathcal{C} - \{j\}$
9: **fin si**
10: **si** $i \notin L$ **y** $|D_i| \geq M$ **entonces**
11: $\mathcal{C} \leftarrow \mathcal{C} - D_i$
12: $L \leftarrow L \cup \{i\}$
13: **fin si**
14: **si** $j \in \mathcal{C}$ **entonces**
15: Aumentar α_j y $\theta_{S_j,j}$.
16: **fin si**
17: Se crea L' removiendo las localidades dependientes en L.
18: **para todo** $i \in L - L'$ **hacer**
19: **para todo** $j \in D_i$ **hacer**
20: $\sigma(j) \leftarrow \{i'\}$
21: **fin para**
22: **fin para**
23: **fin mientras**
24: {**Fase 2**}; (**Inicialización**) $T \leftarrow \emptyset$; $D' \leftarrow \bigcup_{i \in L' - \{s\}} D_i$; $\alpha_j \leftarrow 0$; $\theta_{S,j} \leftarrow 0$
25: Sea $\mathcal{S}$ la colección de conjuntos minimales violados.
26: **mientras** $\mathcal{S} \neq \emptyset$ **hacer**
27: **si** $\sum_{j \in \mathcal{D}} \sum_{S \subseteq V : i \in S, s \notin S} \theta_{S,j} = M c_e$ **entonces**
28: $T \leftarrow T \cup \{e\}$
29: Se actualiza $\mathcal{S}$.
30: **fin si**
31: **fin mientras**
32: Se remueven todas las aristas redundantes en T.
33: **devolver** T, L' y las asignaciones $\sigma(j)$.

II.4 Problema de reparto de costos en el diseño de una red de compra-alquiler de una sola fuente (SSROB-CS)

Hasta aquí solo se ha considerado el problema combinatorio. En la versión de reparto de costos del diseño de una red de compra-alquiler (SSROB-CS por sus siglas en inglés), se deben determinar los usuarios que recibirán el servicio, el precio para cada usuario, y una solución para el problema combinatorio considerando el conjunto de usuarios participantes. Por lo tanto, cualquier mecanismo de reparto de costos debe tener un algoritmo que resuelva el problema combinatorio subyacente. En el Apéndice C se muestra y analiza un mecanismo de Moulin para el SSROB-CS, que emplea una versión modificada del algoritmo primal-dual de Swamy y Kumar (2004).

Para el diseño de un mecanismo que cumpla con las restricciones de un problema de reparto de costos, se debe tener en cuenta los distintos tipos de mecanismos que existen en la literatura (Moulin, 1999; Mehta *et al.*, 2007; Brenner y Schäfer, 2009). En muchos problemas de reparto de costos se utilizan los mecanismos de Moulin (Mehta *et al.*, 2007). Los mismos son flexibles, intuitivos, y proveen control explícito sobre el retorno generado. También son a prueba de estrategias de grupo y se pueden usar en una gama amplia de aplicaciones. Pero este mecanismo no es suficiente pues tiene ciertos problemas, causados por las siguientes tres razones (Mehta *et al.*, 2007):

1. Existen resultados negativos (Immorlica *et al.*, 2008; Roughgarden y Sundararajan, 2006a) que muestran que en varios problemas fundamentales de reparto de costos, el mecanismo de Moulin inevitablemente sufre de un presupuesto equilibrado pobre, eficiencia económica pobre, o ambos.

2. Diseñar buenos mecanismos de Moulin es un problema no-trivial.

3. Los mecanismos de Moulin tienen aplicación primaria en problemas de *demanda binaria*, en los cuales cada usuario posee o no el servicio; y no para problemas donde se tienen niveles de servicio.

II.5 Mecanismos acíclicos

Teniendo en cuenta las limitaciones de los mecanismos de Moulin, Mehta *et al.* (2007) proponen lo que llaman **mecanismos acíclicos**, un nuevo marco de trabajo para diseñar mecanismos de reparto de costos confiables[1] y de presupuesto balanceado aproximado. Estos mecanismos generalizan los mecanismos de Moulin, retienen la mayoría de sus propiedades deseables, y resuelven los problemas que se mencionan más arriba.

De manera informal se describen las diferencias entre ambos mecanismos: los mecanismos de Moulin simulan una subasta ascendente en la que los precios se ofrecen simultáneamente a los jugadores restantes en cada iteración. Por otro lado, en una iteración de los mecanismos acíclicos, estos precios se ofertan uno a uno entre los jugadores, de acuerdo a un orden predefinido. Al momento en que un jugador rechaza el precio que se le oferta, la iteración termina inmediatamente y el jugador se remueve del resto de la subasta. Luego se comienza otra iteración con los jugadores restantes. Esta pequeña diferencia enriquece el conjunto de mecanismos confiables. La razón de ésto es que muchos métodos naturales de establecimiento de precios no dan lugar a que existan subastas ascendentes cuando los precios se ofrecen simultáneamente, pero sí lo hacen cuando algunos de los precios se suprimen por la temprana finalización de una iteración (Mehta *et al.*, 2007).

[1] A prueba de estrategias

Los mecanismos acíclicos ofrecen tres ventajas importantes en relación a los mecanismos de Moulin (Mehta *et al.*, 2007):

1. Se pueden obtener numerosos mecanismos acíclicos confiables a partir de algoritmos primales-duales clásicos. Este hecho se puede aplicar para el problema de reparto de costos en el cubrimiento de vértices, el problema de reparto de costos en el cubrimiento de conjuntos, el problema de ubicación de instalaciones, juego del árbol de Steiner, entre otros.

2. Para muchas clases importantes de problemas de reparto de costos, los mecanismos acíclicos tienen mucho mejor factor de equilibrio en el presupuesto y eficiencia en relación a los mecanismos de Moulin. Por ejemplo: el cubrimiento de vértices, ubicación de instalaciones y cubrimiento de conjuntos (Mehta *et al.*, 2007).

3. Los mecanismos acíclicos pueden extenderse a juegos de demanda general, donde se tienen parámetros múltiples en el que cada usuario se puede asignar a uno de múltiples niveles de servicios.

Para obtener estos beneficios, se realiza un sacrificio en relación a la confiabilidad (Mehta *et al.*, 2007). Los mecanismos acíclicos no son a prueba de estrategias de grupo, sino sólo a prueba de estrategias de grupo *débil* (ésto es, no existe un subconjunto de usuarios que haciendo falsas apuestas puede aumentar estrictamente los beneficios de cada miembro).

Formalmente, un mecanismo acíclico se define de la siguiente manera (Mehta *et al.*, 2007): se tiene una función de costo C y un conjunto de usuarios potenciales U, se requieren una función de reparto de costo ξ y una *función de oferta* τ. Una función de oferta especifica un tiempo no-negativo $\tau(i, S)$ para cada subconjunto $S \subseteq U$ y cada

jugador $i \in S$. Estos tiempos especifican el orden en que se ofrecen los precios a los jugadores de S; los tiempos más bajos corresponden a las ofertas más tempranas, y tiempos iguales indican ofertas simultáneas.

El mecanismo $M(\xi, \tau)$ inducido por ξ y τ es el que se describe en el Algoritmo 3 (Mehta *et al.*, 2007).

Algoritmo 3 Mecanismo acíclico $M(\xi, \tau)$

Entrada: Conjunto de jugadores U, vector de apuestas $\mathbf{b} = (b_i)_{i \in U}$.
Salida: Conjunto de jugadores S que recibirán el servicio, vector de precios $\mathbf{p} = (p_i)_{i \in S}$.

1: Se inicializa $S = U$.
2: Si $b_i \geq \xi(i, S)$ para cada usuario $i \in S$, entonces se para. Se da como salida el conjunto S, la solución factible construída por ξ, y se cobra a cada jugador $i \in S$ el precio $p_i = \xi(i, S)$.
3: Entre todos los jugadores $i \in S$ con $b_i < \xi(i, S)$, sea i^* el que tenga menor $\tau(i, S)$ (en caso de que haya más de uno, se elige arbitrariamente).
4: Se hace $S = S - \{i^*\}$ y se retorna al paso 2.

En los mecanismos de Moulin, una función de reparto de costos monotónica cruzada asegura que la secuencia de precios ofrecidos a un jugador es no-decreciente, lo cual implica que el mecanismo es confiable. Por otro lado, una función que no sea monotónica cruzada hace que las subastas iterativas no sean necesariamente ascendentes, lo cual hace que el mecanismo no sea confiable.

En cada iteración de un mecanismo acíclico, los precios se ofrecen a los usuarios de acuerdo a un orden especificado por la función τ. Si un jugador no está dispuesto a pagar el precio que se le ofrece, entonces la iteración termina inmediatamente. Tal jugador se elimina de la subasta, y la siguiente iteración inicia con los jugadores restantes. Por lo tanto, no es necesario que se ofrezca un precio a un jugador en cada iteración (Mehta *et al.*, 2007).

Al ordenar los tiempos de oferta a los jugadores restantes, se permite la construc-

ción de mecanismos confiables a partir de funciones de reparto de costos que no son monotónicas cruzadas. Intuitivamente, la terminación temprana de una iteración oculta ciertos precios a los jugadores. Si se correlacionan las iteraciones abortadas con los fallos en la monotonicidad cruzada, entonces la subasta simulada e iterativa es ascendente en el siguiente sentido: cuando se hace una oferta a un jugador, es al menos el valor ofrecido en iteraciones anteriores. Esta propiedad es suficiente para la confiabilidad. Muchos algoritmos primales-duales inducen naturalmente un método de reparto de costos que no es monotónico cruzado, pero posee precisamente este tipo de correlación (Mehta *et al.*, 2007).

Sea τ una función de oferta definida en el universo U de jugadores potenciales y ξ el método de reparto de costos. Para un subconjunto $S \subseteq U$ y un jugador $i \in S$, sean $L(i,S)$, $E(i,S)$ y $G(i,S)$ los conjuntos de jugadores de S con tiempos de oferta $\tau(\cdot,S)$ estrictamente menor, igual, o estrictamente mayor que el de i, respectivamente. La función τ es **válida** para ξ si las siguientes dos propiedades se cumplen para cada subconjunto $S \subseteq U$ y cada jugador $i \in S$ (Mehta *et al.*, 2007):

(a) $\xi(i, S\backslash T) = \xi(i,S)$ para cada subconjunto $T \subseteq G(i,S)$; i.e., al jugador i no le afecta la participación o no de los jugadores que reciban ofertas posteriores a la suya.

(b) $\xi(i, S\backslash T) \geq \xi(i,S)$ para cada subconjunto $T \subseteq G(i,S) \cup (E(i,S)\backslash\{i\})$; i.e., el concepto de monotonicidad cruzada se debe cumplir cuando se consideran los elementos de $E(i,S)\backslash\{i\}$.

Por lo tanto, el precio de un jugador i debe mantenerse fijo mientras se remueven los jugadores con tiempos de oferta mayores al de i, y sólo puede incrementarse con el borrado de jugadores con tiempos de oferta iguales al de i. La salida de un jugador con un tiempo de oferta menor no tiene restricción alguna, pues la misma hace que la

iteración termine y oculte los valores de los precios posteriores dentro de esa iteración. Tampoco existe una restricción explícita en la forma en que debe variar la función de oferta τ entre iteraciones consecutivas.

Puede verse que un método de reparto de costos es monotónico cruzado si y sólo si una función de oferta τ_0 con tiempos iguales para cada usuario es válida para dicho método.

Entonces, un mecanismo acíclico es un mecanismo $M(\xi, \tau)$ inducido por un método de reparto de costo ξ y una función de oferta τ que es válida para ξ. Las propiedades básicas de los mecanismos acíclicos son las siguientes (Mehta *et al.*, 2007):

(a) Para cada vector de apuestas $\mathbf{b}$, el mecanismo $M(\xi, \tau)$ termina en $|U|$ iteraciones.

(b) Si ξ es un algoritmo de tiempo polinomial, también lo es $M(\xi, \tau)$.

(c) Si ξ es de β-presupuesto equilibrado respecto a una función de costo C, también lo es $M(\xi, \tau)$.

(d) El mecanismo $M(\xi, \tau)$ es de participación voluntaria.

(e) El mecanismo $M(\xi, \tau)$ es a prueba de estrategias, y a prueba de estrategias de grupo *débil*.

Muchos algoritmos primales-duales diseñados para problemas de optimización se pueden transformar en métodos de reparto de costos. Lo anterior se debe a que las variables duales representan en muchos casos las contribuciones que realizan los usuarios (o conjunto de usuarios) para la construcción de la solución primal. Además, los algoritmos primales-duales tienen asociados un concepto de tiempo que puede ser útil para la definición de la función de oferta τ.

Si el precio que corresponde a un usuario es igual a la variable dual en el algoritmo primal-dual (o una fracción de la misma en el caso de variables duales que representan a conjuntos de usuarios), entonces se dice que el método de reparto de costos ξ es **canónico**. Además, si la función de oferta para un cierto usuario (o grupo de ellos) es igual o está relacionada con el tiempo en que la variable dual correspondiente a ese usuario (o grupo de los mismos) aumenta dentro del algoritmo, entonces se dice que la función de oferta es **canónica**. El mecanismo acíclico que se obtiene con un método de reparto de costos canónico y una función de oferta canónica, se conoce como mecanismo acíclico **básico**. Estos mecanismos pueden construirse con algoritmos que utilizan el esquema primal-dual o el ajuste dual (dual-fitting).

En (Mehta *et al.*, 2007) se hace la siguiente proposición: sea C una función de costo definida implícitamente por un problema combinatorio de minimización en un universo U de jugadores. Sea $\mathcal{A}$ un algoritmo primal-dual o de ajuste dual que, dado un subconjunto $S \subseteq U$, calcule una solución primal factible con un costo $C'(S)$ y una solución dual factible con un valor de al menos $C'(S)/\beta$. Entonces el método de reparto de costos canónico ξ para $\mathcal{A}$ es de β-presupuesto equilibrado.

Mehta *et al.* (2007) dan una medida alternativa al concepto de eficiencia, a la que llaman *costo social*. El costo social óptimo se define como: $\min_{S \subseteq U}[C(S) + \sum_{i \notin S} u_i]$. Para que la eficiencia sea máxima, este costo social debe ser mínimo. Con esta definición del costo social, es posible determinar los factores de aproximación de eficiencia de los mecanismos de reparto de costos.

Con ello, en (Mehta *et al.*, 2007) se muestra que los mecanismos acíclicos obtienen mejores resultados (o al menos iguales) que los mecanismos de Moulin para ciertos problemas básicos de reparto de costos (ubicación de instalaciones, cubrimiento de vértices, cubrimiento de conjuntos, árbol de Steiner). En ciertos casos superan las cotas

inferiores para el equilibrio del presupuesto aproximado y la eficiencia que se tienen para los métodos monotónicos cruzados. Todos los mecanismos diseñados en (Mehta *et al.*, 2007) se obtienen de forma genérica a partir de algoritmos primales-duales o algoritmos de ajuste dual. Los resultados son los siguientes:

- Para los problemas de reparto de costos en el cubrimiento de vértices, se muestra que con un algoritmo estándar de 2-aproximación se puede obtener un mecanismo acíclico de 2-presupuesto balanceado y de $O(\log k)$ de eficiencia, siendo k el número de jugadores. Los mejores factores de aproximación posibles con mecanismos de Moulin, tanto para el equilibrio del presupuesto como la eficiencia, son $\Omega(\sqrt[3]{k})$ y $\Omega(\sqrt{k})$, respectivamente (Immorlica *et al.*, 2008; Roughgarden y Sundararajan, 2006a).

- Para el problema de reparto de costos de ubicación de instalaciones no-métrico y el cubrimiento de conjuntos, dan un mecanismo acíclico de $O(\log k)$-presupuesto balanceado y de $O(\log k)$ de eficiencia. Los mejores factores de aproximación posibles con mecanismos de Moulin, tanto para el equilibrio del presupuesto como la eficiencia, son $\Omega(\sqrt[3]{k})$ y $\Omega(\sqrt{k})$, respectivamente (Immorlica *et al.*, 2008; Roughgarden y Sundararajan, 2006a).

- Para el problema de reparto de costos de ubicación de instalaciones métrico, se logra un mecanismo acíclico de 1.61-presupuesto balanceado y de $O(\log k)$ de eficiencia. Los mejores factores de aproximación posibles con mecanismos de Moulin, tanto para el equilibrio del presupuesto como la eficiencia, son 3 (Immorlica *et al.*, 2008; Pál y Tardos, 2003) y $\Theta(\log k)$, respectivamente (Roughgarden y Sundararajan, 2006a).

- Para el problema de reparto de costos del árbol de Steiner, se tiene que un algo-

ritmo primal-dual estándar (Agrawal *et al.*, 1995; Goemans y Williamson, 1995) induce un mecanismo acíclico con factores de aproximación que igualan los mejores resultados que se obtienen con mecanismos de Moulin, los cuales son de 2 para el presupuesto aproximado (Jain y Vazirani, 2001a; Könemann *et al.*, 2005) y $\Theta(\log^2 k)$ para la eficiencia (Roughgarden y Sundararajan, 2006b,a).

II.5.1 Ejemplo

A continuación se muestra un ejemplo de un mecanismo acíclico para el problema de reparto de costos del cubrimiento de vértices, obtenido a partir de un algoritmo primal-dual para el problema de optimización. Como se había mencionado anteriormente, para obtener un mecanismo acíclico, se debe determinar una función de oferta τ válida para el método de reparto de costos ξ (Mehta *et al.*, 2007).

En el problema de reparto de costos del cubrimiento de vértices, los jugadores del conjunto U son las aristas de un grafo no dirigido $G = (V, E)$ con peso w_v en cada vértice $v \in V$; y el costo $C(Q)$ para un subconjunto $Q \subseteq U$ de jugadores se define como el cubrimiento de vértices de peso mínimo[2] $V' \in V$ del subgrafo de G que corresponde a los jugadores de Q (Jain y Vazirani, 2001a). El mecanismo de reparto de costos debe: (i) determinar el subconjunto $Q \subseteq U$ de jugadores que recibirán el servicio (es decir, asegurar que dichos jugadores estén conectados a algún vértice de V'), (ii) obtener una solución para dar servicio a los jugadores de Q, y (iii) determinar el precio a cobrar a cada jugador de Q.

La formulación de la rejalación del programa lineal entero para el cubrimiento de

[2]Un cubrimiento de vértices de peso mínimo de un grafo no dirigido $G = (V, E)$ es un subconjunto $V' \subseteq V$ de peso mínimo (mínimo valor de $\sum_{v \in V'} w_v$), tal que para cada $(u, v) \in E$ se cumple que $u \in V'$, o $v \in V'$, o $u, v \in V'$.

vértices y la de su dual se muestran a continuación:

$$\text{minimizar} \sum_{v \in V} x_v w_v \qquad\qquad \text{maximizar} \sum_{e \in E} y_e$$

$$\text{sujeto a: } x_u + x_v \geq 1, \forall (u,v) \in E \qquad\qquad \text{sujeto a: } \sum_{v:(u,v)\in E} y_{uv} \leq w_u, \forall u \in V$$

$$x_v \geq 0, \forall v \in V \qquad\qquad y_e \geq 0, \forall e \in E$$

La variable x_v (en la formulación ILP) indica si el vértice v pertenece o no al cubrimiento. La primera restricción para el problema primal indica que cada arista del grafo debe estar cubierta. Considerando el problema dual, y_e puede interpretarse como la contribución que realiza la arista e para obtener una solución primal factible. La primera restricción para el problema dual indica que la suma de las contribuciones de las aristas incidentes a un vértice no debe exceder su peso. Se sabe que cualquier solución del problema dual es una cota inferior para el valor óptimo del problema primal de minimización (Vazirani, 2001).

En el esquema primal-dual se empieza con alguna solución primal no-factible y una solución dual factible. El proceso es iterativo, y en cada iteración se mejora la factibilidad de la solución primal y la optimalidad de la dual (manteniendo la factibilidad de la solución dual). Lo anterior se hace generalmente colocando al inicio todas las variables duales a cero, y luego se incrementan gradualmente estas variables hasta que alguna restricción del programa dual esté ajustada. Esta restricción apunta a un objeto que el dual pueda pagar para incluirse en la solución primal. Luego, las variables duales relacionadas a la restricción ajustada se congelan, y el algoritmo continúa incrementando las otras variables duales. El algoritmo termina cuando se obtiene una solución completa para el problema primal. El análisis se basa en probar que los valores de las

soluciones del primal y dual construidas están cerca entre sí, y por lo tanto ambos están cerca del óptimo (Vazirani, 2001).

El algoritmo primal-dual para el cubrimiento de vértices se muestra a continuación (Vazirani, 2001):

Algoritmo 4 Algoritmo primal-dual para el cubrimiento de vértices.

Entrada: Grafo no dirigido $G = (V, E)$ con peso w_v en cada vértice $v \in V$.

Salida: Subconjunto $V' \subseteq V$ determinado por el vector $\mathbf{x} = (x_v)_{v \in V}$, que representa un cubrimiento de vértices de G.

1: Se inicializan $\mathbf{x} = \mathbf{0}$, $\mathbf{y} = \mathbf{0}$.
2: **mientras** exista una variable dual no-congelada **hacer**
3: Se aumentan las variables duales hasta que la restricción para algún vértice v se ajuste.
4: Se congelan todas las variables duales y_e de las aristas incidentes a v. Se hace $x_v = 1$.
5: **fin mientras**
6: **devolver** el vector $\mathbf{x}$.

Se dice que un vértice u está *ajustado* si $\sum_{v:(u,v)\in E} y_{uv} = w_u$; es decir, un vértice u se agrega al cubrimiento si la suma de las contribuciones de las aristas incidentes a él iguala su peso (o costo). Este algoritmo primal-dual tiene un factor de aproximación de 2 (Vazirani, 2001). Si se toman las variables duales (al finalizar la ejecución del algoritmo) como los precios a cobrar a los usuarios, se tiene un método de reparto de costos que recupera al menos la mitad del costo de la solución construida (por la propiedad del algoritmo primal-dual descrito). Por lo tanto, el método de reparto de costos determinado de esta forma es de 2-presupuesto equilibrado.

En la Figura 2 se muestra un ejemplo de la ejecución del algoritmo, asociando un concepto de tiempo t al crecimiento de las variables duales. Inicialmente ($t = 0$), todas las variables duales son iguales a 0 (es decir, $y_e = 0$, $\forall e \in E$). Luego, se aumentan simultáneamente las variables duales hasta un tiempo $t = 1/2$, donde se tiene que v_3 está

ajustado, por lo que se lo agrega al cubrimiento, y las variables duales correspondientes a las aristas incidentes a dicho vértice se congelan. Posteriormente se vuelven a aumentar las variables duales no-congeladas hasta que en $t = 1$, v_1 se vuelve ajustado y se repite el procedimiento anterior. Finalmente, en $t = 3/2$, los vértices v_2 y v_5 se vuelven ajustados y todas las aristas están cubiertas. Por ello, se tiene que $V' = \{v_1, v_2, v_3, v_5\}$.

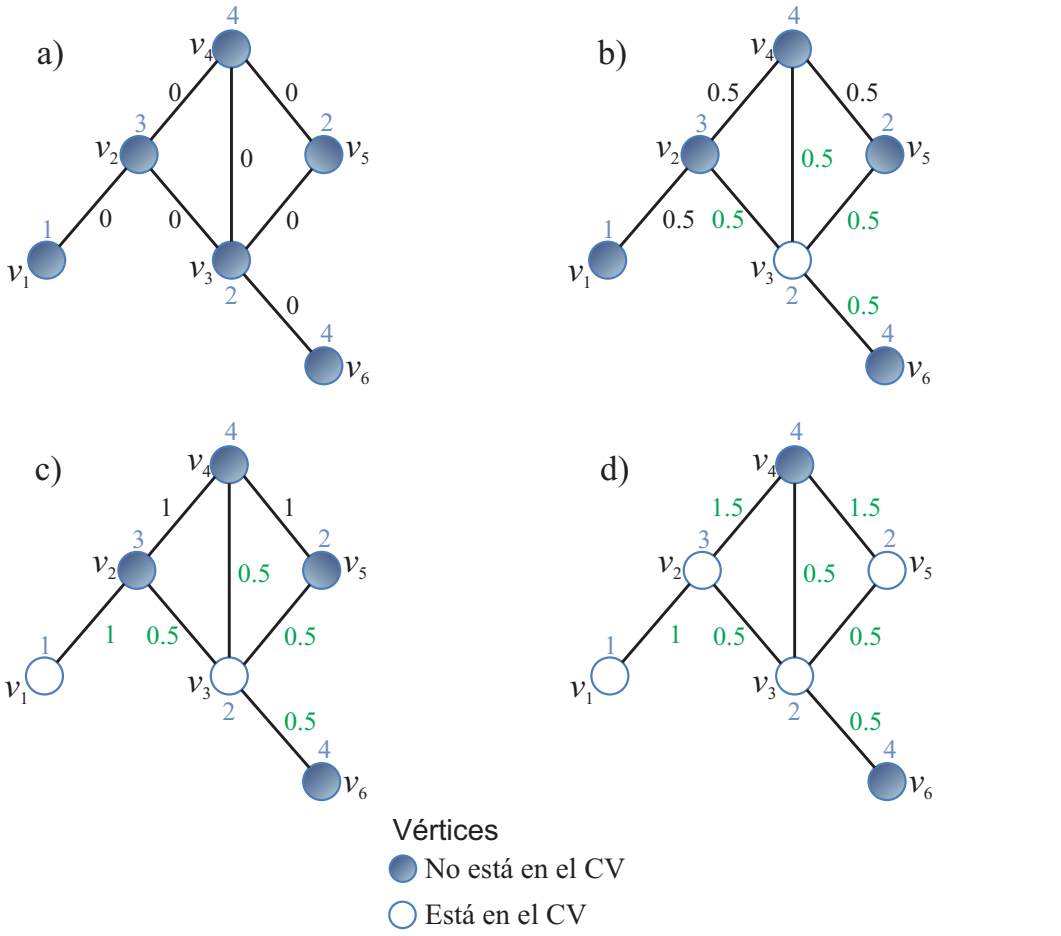


Figura 2. Ejemplo de la ejecución del algoritmo primal-dual para el cubrimiento de vértices. a) Situación inicial, todas las variables duales son iguales a 0. b) El vértice v_3 se vuelve ajustado. c) El vértice v_1 se vuelve ajustado. d) Los vértices v_2 y v_5 se vuelven ajustados. En este momento se tiene un cubrimiento de vértices ya que todas las variables duales están congeladas.

El método de reparto de costos ξ es canónico, pues simplemente asigna precios

iguales a los valores finales de las variables duales (por ejemplo, el precio ofrecido al usuario en la arista (v_2, v_4) es igual a 1.5, y así para los demás usuarios). Sea una función de oferta (también canónica) en la que $\tau(i, Q)$ es igual al tiempo en que la variable dual correspondiente al usuario $i \in Q$ se vuelve congelada durante la ejecución del algoritmo (por ejemplo, para el usuario en la arista (v_2, v_4) el tiempo de oferta es nuevamente 1.5, y así para los demás usuarios). Entonces este mecanismo primal-dual es básico (Mehta *et al.*, 2007). A continuación se demuestra que la función de oferta canónica τ es válida para el método de reparto de costos ξ, y por la definición dada en (Mehta *et al.*, 2007), el mecanismo determinado por ξ y τ es acíclico.

Teorema 1. *La función de oferta canónica τ es válida para el método de reparto de costos canónico ξ obtenido a partir del algoritmo primal-dual para el cubrimiento de vértices.*

Demostración. Sea un problema de reparto de costos del cubrimiento de vértices. Sea $R(S)$ el resultado de la ejecución del algoritmo en un caso de cubrimiento de vértices inducido por un subconjunto $S \subseteq U$ de jugadores. Se considera un subconjunto $S \subseteq U$ y un jugador $i \in S$. Sea v el vértice agregado al cubrimiento V' en el tiempo $\tau(i, S)$ en $R(S)$ al que incide i y que produce que la variable dual correspondiente a i sea congelada.

Una función es válida si cumple con dos condiciones:

1. $\xi(i, S \backslash T) = \xi(i, S)$ para cada subconjunto $T \subseteq G(i, S)$[3]. Se considera uno de tales conjuntos T. Ningún vértice agregado al cubrimiento en $R(S)$ en el tiempo

[3]Se debe recordar que: para un subconjunto $S \subseteq U$ y un jugador $i \in S$; $L(i, S)$, $E(i, S)$ y $G(i, S)$ son los conjuntos de jugadores de S con tiempos de oferta $\tau(\cdot, S)$ estrictamente menor, igual, o estrictamente mayor que el de i, respectivamente.

$\tau(i, S)$ (o antes) es un extremo de alguna arista correspondiente a T. Entonces, el hecho de eliminar jugadores de T no tiene efecto alguno en la ejecución del algoritmo primal-dual en el tiempo $\tau(i, S)$ o antes del mismo (ésto puede observarse en la Figura 2-d). Debido a que el precio final ofrecido al jugador i se determina una vez que uno de sus extremos se agrega al cubrimiento, se cumple que $\xi(i, S\backslash T) = \xi(i, S)$.

2. $\xi(i, S\backslash T) \geq \xi(i, S)$ para cada subconjunto $T \subseteq G(i, S) \cup (E(i, S)\backslash\{i\})$. Lo anterior se demuestra de forma similar. El hecho de borrar un subconjunto $T \subseteq G(i, S) \cup (E(i, S)\backslash\{i\})$ de jugadores no afecta la ejecución del algoritmo primal-dual antes del tiempo $\tau(i, S)$, por lo que el tiempo de oferta $\tau(i, S\backslash T)$ será en o después de $\tau(i, S)$. Lo anterior se debe a que los jugadores de $E(i, S)\backslash\{i\}$ pueden ser incidentes a un mismo vértice v del cubrimiento, por lo que si uno de tales jugadores se elimina, entonces el jugador i todavía no estará cubierto en el tiempo $\tau(i, S)$ (pues el costo del vértice v todavía no se pagará) y su variable dual seguirá aumentando. Entonces, $\xi(i, S) = \tau(i, S) \geq \tau(i, S\backslash T) = \xi(i, S\backslash T)$, con lo cual queda demostrado el teorema.

■

Se había señalado que un mecanismo de Moulin es un caso particular de los mecanismos acíclicos, donde los tiempos de oferta son iguales para todos los usuarios. Otro caso particular es cuando se consideran tiempos de oferta distintos para todos los usuarios. En este último caso, si además el precio que se oferta a cada jugador es el costo incremental de agregarlo al conjunto de usuarios participantes, se tiene un mecanismo incremental de reparto de costos.

II.6 Mecanismos incrementales

Brenner y Schäfer (2009) proponen un método para convertir cualquier algoritmo de ρ-aproximación para un problema de optimización combinatoria en un mecanismo acíclico de ρ-presupuesto equilibrado. Con este método, demuestran que no existe brecha entre los mejores factores de aproximación de los algoritmos de aproximación y los mecanismos de reparto de costos que sean a prueba de estrategias de grupo débil. La aplicabilidad del método se demuestra mediante el diseño de mecanismos para la calendarización y diseño de redes que mejoran los mejores resultados posibles en cuanto a equilibrio del presupuesto mediante los mecanismos de Moulin. Cabe destacar que los resultados de (Mehta *et al.*, 2007) mediante mecanismos acíclicos se centran fundamentalmente en el esquema primal-dual, mientras que los resultados de (Brenner y Schäfer, 2009) son posibles mediante cualquier tipo de algoritmos de aproximación.

En realidad, en (Brenner y Schäfer, 2009) se utilizan los mecanismos incrementales, los cuales constituyen un caso especial de los mecanismos acíclicos. Informalmente, un **mecanismo incremental** trabaja de la siguiente manera: procede mediante iteraciones; al comienzo de cada iteración, fija un orden en el conjunto de jugadores restantes. De acuerdo a este orden, pregunta a un jugador tras otro si está dispuesto a pagar el precio ofrecido por el mecanismo. Este precio es simplemente el costo incremental, es decir, el aumento en el costo total causado por la adición de este jugador al grupo actual de jugadores que recibirán el servicio. Si el jugador acepta, se le agrega al conjunto de usuarios que recibirán el servicio y nunca más se considera; si no acepta, el jugador se remueve del juego y el mecanismo continúa con la siguiente ronda. Moulin (1999) establece que si la función de costo es supermodular, esencialmente sólo los mecanismos incrementales pueden ser a prueba de estrategias de grupo y de presupuesto equilibrado.

Las contribuciones principales de Brenner y Schäfer (2009) son las siguientes:

1. *Marco para obtener mecanismos a prueba de estrategias de grupo débil.* Más específicamente, muestran cómo un algoritmo de ρ-aproximación para el problema de optimización, relacionado al problema de reparto de costos, puede convertirse en un mecanismo incremental que es de ρ-presupuesto equilibrado y prueban que este mecanismo es a prueba de estrategias de grupo débil. La construcción es simple y utilizan al algoritmo de aproximación como una caja negra. Utilizan este marco para problemas de calendarización, árbol de esparcimiento mínimo y el árbol de Steiner. Este método es más simple en comparación con los mecanismos acíclicos propuestos por Mehta *et al.* (2007), donde obtener la función de oferta para ciertos problemas es una tarea no-trivial.

2. *Método para acotar el costo social aproximado y su aplicación en problemas de calendarización.* Proporcionan condiciones y técnicas de demostración adicionales para situaciones en las que se considera la minimización del costo social. Proveen un método para facilitar la demostración de cotas superiores en el factor de aproximación del costo social de mecanismos incrementales. Esencialmente, identifican una propiedad de monotonicidad débil adicional, la cual, de ser satisfecha por el mecanismo, permite acotar el factor de aproximación del costo social.

3. *Relación con los mecanismos acíclicos.* Explican cómo en este marco propuesto, los mecanismos incrementales se pueden ver como complementarios a los mecanismos de Moulin respecto al grado de libertad que el mecanismo tiene para ordenar las propuestas de precio a los jugadores.

El resultado principal en relación al marco propuesto en (Brenner y Schäfer, 2009) es el siguiente: sea ALG un algoritmo de ρ-aproximación para un problema de optimización

$\mathcal{P}$. Entonces, existe un mecanismo de reparto de costos para $\mathcal{P}$ que es a prueba de estrategias de grupo débil y de ρ-presupuesto equilibrado.

Además del algoritmo de aproximación ALG, el ingrediente principal del marco propuesto en (Brenner y Schäfer, 2009) es una función de orden inyectiva $\tau : U \times 2^U \to \mathbb{R}^+$ que define una permutación para cada subconjunto $S \subseteq U$ ordenando los elementos de S respecto a valores crecientes de τ. Puede notarse que esta función es similar a la función de oferta definida en (Mehta *et al.*, 2007) para los mecanismos acíclicos. Para un algoritmo de aproximación ALG dado, sea $\bar{C}$ la función de costo obtenida mediante ALG para un conjunto de jugadores $S \subseteq U$. Sin pérdida de generalidad, se supone que $\bar{C}(\emptyset) = 0$. Se supone que son dados un algoritmo ALG y una función de orden τ.

El mecanismo incremental $I(\mathsf{ALG}, \tau)$ inducido por ALG y τ recibe el vector de apuestas $\mathbf{b}$ como entrada y procede de la siguiente manera (Brenner y Schäfer, 2009):

Algoritmo 5 Mecanismo incremental $I(\mathsf{ALG}, \tau)$

Entrada: Conjunto de jugadores U, vector de apuestas $\mathbf{b} = (b_i)_{i \in U}$.
Salida: Conjunto de jugadores S que recibirán el servicio, representado por el vector de asignación $\mathbf{x} = (x_i)_{i \in U}$; y el vector de precios $\mathbf{p} = (p_i)_{i \in U}$.

1: Se inicializa $A = \emptyset$, $R = U$.
2: **mientras** $A \neq R$ **hacer**
3: Entre todos los jugadores $i \in R \backslash A$, sea i^* el que tenga el menor valor $\tau(i, R)$.
4: Se calcula $p_{i^*} = \bar{C}(A \cup \{i^*\}) - \bar{C}(A)$
5: **si** $b_{i^*} \geq p_{i^*}$ **entonces**
6: Se hace $A = A \cup \{i^*\}$
7: **si no**
8: Se hace $R = R \backslash \{i^*\}$
9: **fin si**
10: **fin mientras**
11: **devolver** los vectores $\mathbf{x}$ y $\mathbf{p}$

A través de la ejecución del algoritmo, R indica el conjunto de jugadores que permanecen en ese momento en el juego, mientras que A denota el conjunto de jugadores

que se han aceptado hasta ese momento. El mecanismo empieza con el conjunto de jugadores $R = U$ e inicializa $A = \emptyset$. En cada iteración, elige el jugador i^* de $R \backslash S$ con el menor valor de τ, y calcula su costo incremental aproximado p_{i^*}, definido como el incremento en el costo aproximado $\bar{C}$ cuando el jugador i^* se agrega a A. Si el jugador i^* acepta el precio que se le ofrece, se agrega al conjunto A de jugadores que aceptan la oferta; de otra forma, se suprime de R y se retira del juego. El mecanismo continúa de esta forma hasta que eventualmente todos los jugadores restantes se aceptan. Entonces retorna el vector de asignación $\mathbf{x}$ para indicar los jugadores que se encuentran en A, y los correspondientes precios $\mathbf{p}$ (donde implícitamente se coloca $p_i = 0$ para todo $i \notin A$) (Brenner y Schäfer, 2009).

En (Brenner y Schäfer, 2009) se demuestra que este mecanismo es de ρ-aproximación debido a ALG, y que además es a prueba de estrategias de grupo débil. Luego, aplican directamente este mecanismo a problemas de calendarización y diseño de redes (árbol de esparcimiento, árbol de Steiner y TSP). En estos ejemplos, la función de orden se define implícitamente mediante el funcionamiento del algoritmo de aproximación utilizado. Para problemas más complejos, en (Brenner y Schäfer, 2009) se definen las condiciones que debe cumplir la función de orden, y se aplican estos conceptos para problemas más complejos de calendarización.

Se realiza la siguiente definición (Brenner y Schäfer, 2009): una función de orden $\tau : U \times 2^U \to \mathbb{R}^+$ es **consistente** si para todos los subconjuntos $S \subseteq T \subseteq U$, ordenados internamente por τ de la forma $S = \{i_1, i_2, ..., i_p\}$ y $T = \{j_1, j_2, ..., j_q\}$, se cumple lo siguiente: si k es mínimo con $j_k \in T \backslash S$, entonces $i_l = j_l$ para todo $l < k$. En la Figura 3 se muestra un ejemplo de consistencia.

La consistencia de τ asegura que los conjuntos ordenados $R' = R \backslash \{i_k\}$ y R coinciden en los primeros $k - 1$ jugadores; dicho de otra forma, sólo el orden de los jugadores

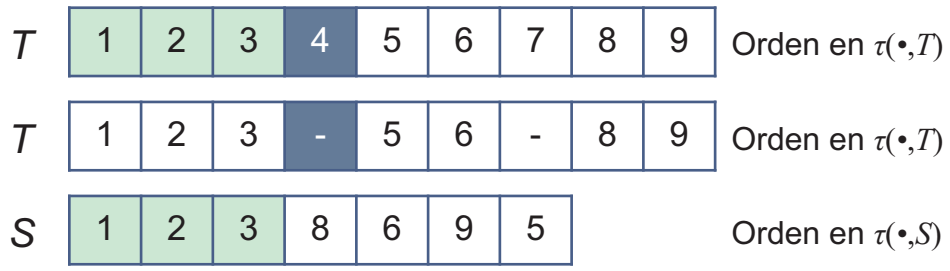


Figura 3. Ilustración de la propiedad de consistencia para dos conjuntos $S \subseteq T$.

posteriores a $\{i_k\}$ en R puede cambiar en R'. Entonces, los primeros $k-1$ jugadores corresponden al conjunto A de los jugadores que aceptaron la oferta. (Brenner y Schäfer, 2009).

Se hace también la siguiente definición: Sea ALG un algoritmo de ρ-aproximación para el problema de optimización $\mathcal{P}$. Se dice que el algoritmo ALG es τ-creciente si para cada $S \subseteq U$ y $1 \leq k \leq |S|$, se tiene que $\bar{C}(S_k) \geq \bar{C}(S_{k-1})$ (donde $S_k = \{i_1, ..., i_k\} \subseteq S$ es el conjunto de los primeros $1 \leq k \leq |S|$ elementos de S ordenados por τ).

Un punto muy importante constituye el siguiente teorema (Brenner y Schäfer, 2009):

Teorema 2. *Sea τ una función de orden consistente y sea ALG un algoritmo τ-creciente ρ-aproximado para un problema de optimización $\mathcal{P}$. Entonces, el mecanismo incremental $I(\mathsf{ALG}, \tau)$ es un mecanismo de reparto de costos a prueba de estrategias de grupo débil y ρ-presupuesto equilibrado para $\mathcal{P}$, y este mecanismo satisface la propiedad de no-transferencia positiva.*

Dado un algoritmo de aproximación ALG, el factor del equilibrio del presupuesto de $I(\mathsf{ALG}, \tau)$ es independiente de la función de orden τ usada. Pero la elección de la función de orden puede influenciar en gran medida el costo social de la solución de salida. Por lo tanto, el punto central consiste en determinar una función de orden que logre un factor de aproximación del costo social satisfactorio (Brenner y Schäfer, 2009).

II.6.1 Ejemplos

En los siguientes ejemplos se muestra que, aunque el algoritmo no sea no-decreciente, se pueden encontrar funciones de orden consistentes (implícitas en el algoritmo como en este caso) que permitan satisfacer la condición de costos no negativos.

Se tratará primero el problema de reparto de costos en el árbol de esparcimiento mínimo (MST por sus siglas en inglés), suponiendo costos no-negativos en las aristas del grafo $G = (V, E)$. Supóngase el caso dado por la Figura 4 (donde r es la raíz), y sea la función de orden τ arbitraria dada por $(a \to b \to c)$. El costo al agregar el vértice b es $C(a, b) = 4$ (si $\epsilon > 0$). Sin embargo, al agregar el vértice c, el costo es $C(a, b, c) = 3 + 3\epsilon$, teniéndose un costo incremental negativo si $\epsilon < 1/3$.

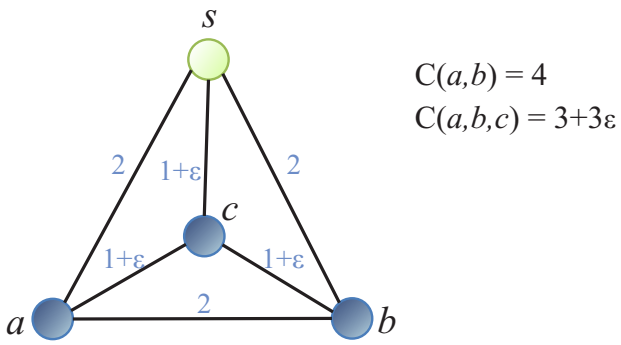


Figura 4. Caso con $0 < \epsilon < 1/3$ en el que, al agregar el vértice c luego de a y b, el costo incremental es negativo.

En el algoritmo de Prim (1957) viene implícitamente una función de orden consistente, la cual es el orden en que dicho algoritmo agrega los vértices al MST. Como los costos son no-negativos, los costos incrementales también lo serán, cumpliéndose la restricción de transferencias no-negativas. Sea $U \subseteq V$ el conjunto de jugadores potenciales. Para un subconjunto $S \subseteq U$ de vértices, sea $\tau(\cdot, S)$ el orden en que el algoritmo de Prim agrega los vértices al MST si se ejecuta en S. Se define $I^{PRIM} := I(PRIM, \tau)$

al mecanismo incremental inducido por el algoritmo de Prim y τ.

A partir del Teorema 2, se pueden enunciar los siguientes corolarios:

Corolario 1. *El mecanismo incremental I^{PRIM} inducido por el algoritmo de Prim es a prueba de estrategias de grupo débil y de presupuesto equilibrado para el problema de reparto de costos en el árbol de esparcimiento mínimo.*

Corolario 2. *El mecanismo incremental I^{PRIM} inducido por el algoritmo de Prim es a prueba de estrategias de grupo débil y de 2-presupuesto equilibrado para el problema de reparto de costos en el árbol de Steiner.*

Demostración. Se ejecuta el mecanismo de reparto de costos $M := I^{PRIM}$ sobre el conjunto de vértices requeridos. Sean $\mathcal{T}$ el MST obtenido para el conjunto salida final S^M y p_i^{PRIM} el precio para el jugador $i \in S^M$. Las salidas serán el árbol $\mathcal{T}$, y el precio $p_i = p_i^{PRIM}$ para cada jugador $i \in S^M$. La suma de estos precios es igual al costo $C(\mathcal{T})$ del MST.

Sea $\mathcal{T}^*$ el costo del árbol de Steiner óptimo teniendo como requeridos a los vértices de S^M. Se duplica cada arista de $\mathcal{T}^*$ para obtener un tour de Euler en los vértices requeridos S^M. Al recorrer este tour de Euler y acortando las distancias al saltar los vértices de Steiner, se obtiene un árbol de esparcimiento $\mathcal{T}'$ en S^M cuyo peso es a lo más $2C(\mathcal{T}^*)$. Se puede suponer que los costos de las aristas satisfacen la desigualdad del triángulo. Como $\mathcal{T}$ es un MST, se tiene que:

$$C(\mathcal{T}) \leq C(\mathcal{T}') \leq 2C(\mathcal{T}^*)$$

lo cual demuestra que es de 2-presupuesto equilibrado. ■

II.6.2 Relación entre los mecanismos acíclicos y los incrementales

Considérese la función de oferta τ de un mecanismo acíclico. Para un conjunto de jugadores $S \subseteq U$, τ divide a S en subconjuntos de jugadores con tiempos de oferta iguales $\tau(\cdot, S)$. En los mecanismos acíclicos, a los conjuntos de mayor tamaño con tiempos de oferta iguales se denominan **conglomerados**. Dependiendo del tamaño de estos conglomerados, se puede afirmar lo siguiente: (Brenner y Schäfer, 2009):

- Si todos los jugadores de S se encuentran en el mismo conglomerado, entonces se tiene un mecanismo de Moulin, y la función de reparto de costos debe ser monotónica cruzada.

- Si todos los conglomerados son de un único elemento, entonces el precio ofrecido a un jugador no se puede volver a cambiar. Los mecanismos determinados por esta función de oferta se denominan mecanismos solitarios.

Entonces, los mecanismos incrementales constituyen un tipo de mecanismos solitarios, donde se cobra a cada jugador el precio incremental de agregarlo a la solución actual. Se puede ver que las funciones de orden consistentes son válidas para los mecanismos incrementales de reparto de costos definidos en (Brenner y Schäfer, 2009). Intuitivamente, la razón de ésto es que el precio a cobrar a un jugador sólo depende del conjunto de jugadores que lo preceden en el orden determinado por τ. Como consecuencia, los mecanismos incrementales cumplen con las propiedades de los mecanismos acíclicos (Brenner y Schäfer, 2009). En la Figura 5 se ilustra la relación entre los mecanismos acíclicos, incrementales y los de Moulin.

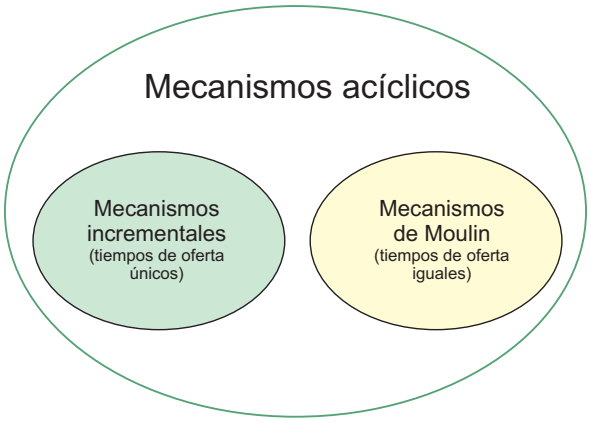


Figura 5. Mecanismos de reparto de costos.

II.7 Discusión acerca del enfoque de solución propuesto

Intuitivamente, los conceptos anteriores pueden resumirse de la siguiente manera: el objetivo es determinar un conjunto $Q \subseteq A$ de usuarios que recibirán un servicio, y repartir el costo de tal forma que los usuarios estén contentos con esa decisión. En particular se desea que ningún agente tenga deseos de cambiar de opinión posteriormente acerca de su permanencia en Q, ni incentivo para mentir acerca de su utilidad, y que ningún grupo de usuarios esté tentado a establecer su propio servicio a un costo menor a la suma de sus contribuciones (Pál y Tardos, 2003).

La mayoría de los trabajos que tratan sobre problemas de reparto de costos, se enfocan en obtener un esquema de reparto de costos monotónico cruzado de manera a aplicar el mecanismo de Moulin. Mehta *et al.* (2007) muestran que los mecanismos acíclicos se pueden aplicar para mejorar los resultados en ciertos problemas, pero no se tienen aún resultados de su aplicación para el problema de diseñar una red compra-alquiler con una fuente. Ellos presentan la aplicación del marco para mecanismos

acíclicos básicos, siendo aún incierta su aplicación en algoritmos primales-duales de más de una fase. En este caso, lo que se debería hacer es encontrar una función de oferta válida, que también puede ser complicada para ciertos problemas (Brenner y Schäfer, 2009). Además, no todos los algoritmos primales-duales pueden convertirse en mecanismos acíclicos, lo cual lo demuestran Mehta *et al.* (2007).

Existen dos parámetros importantes en los mecanismos de reparto de costos: el equilibrio del presupuesto y la eficiencia. Al utilizar un mecanismo de Moulin se prioriza el equilibrio del presupuesto, dejando de lado la eficiencia. En los trabajos más recientes (Roughgarden y Sundararajan, 2006b; Mehta *et al.*, 2007; Brenner y Schäfer, 2009) ya se analiza la eficiencia a través del costo social para los distintos mecanismos de reparto de costos. En este trabajo se pretende considerar el análisis tanto del equilibrio del presupuesto como de la eficiencia, empleando mecanismos incrementales y acíclicos.

Los mecanismos incrementales constituyen una subclase de mecanismos acíclicos. En teoría, este marco puede aplicarse a toda la variedad de algoritmos de aproximación, no limitándose a algoritmos primales-duales como se muestra en (Mehta *et al.*, 2007). En este caso, lo que debe encontrarse es una función de orden que permita obtener costos incrementales no-negativos (Brenner y Schäfer, 2009). Algo importante es que el factor del equilibrio del presupuesto es el mismo que el factor de aproximación del algoritmo utilizado para resolver el problema combinatorio subyacente, lo cual puede ser importante para mejorar los resultados conocidos en la literatura en relación al equilibrio del presupuesto. Además, para ciertos problemas de reparto de costos, los mecanismos incrementales logran mejores resultados que los de Moulin en cuanto a eficiencia (Brenner y Schäfer, 2009). En la Figura 6 se muestra un esquema acerca de la clasificación de los mecanismos y sus requerimientos.

El mejor factor de aproximación en el equilibrio del presupuesto para el SSROB-CS

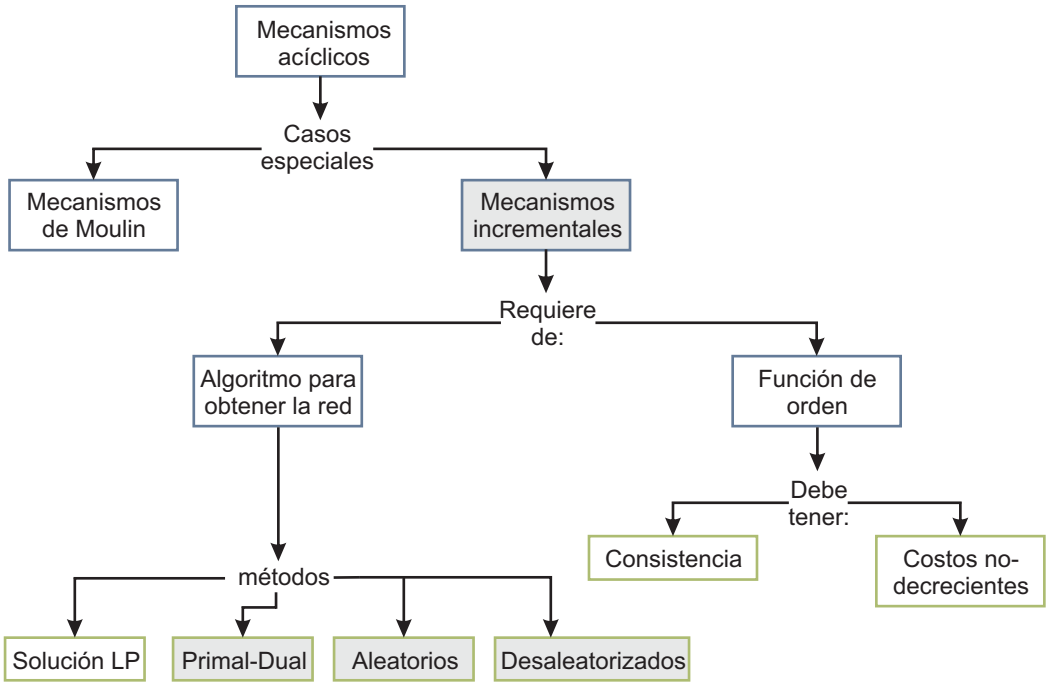


Figura 6. Esquema de mecanismos, requerimientos y algoritmos a ser considerados.

es de 4.6, mediante un mecanismo de Moulin propuesto por Gupta *et al.* (2008). Para superar este factor de aproximación, se debe analizar la aplicación a un mecanismo incremental de los algoritmos combinatorios que tengan un factor de aproximación menor o al menos igual que 4.6. Estos algoritmos son el de Swamy y Kumar (2004), el mismo algoritmo de Gupta *et al.* (2008), o bien otros que se muestran más adelante. En cuanto a un mecanismo acíclico, se considera la aplicación del algoritmo primal-dual de Swamy y Kumar (2004).

Una vez obtenido un mecanismo el mecanismo de reparto de costos, se debe analizar su eficiencia a través del costo social definido en (Mehta *et al.*, 2007).

Capítulo III

Algoritmo de Swamy-Kumar en un mecanismo incremental para el SSROB-CS

A continuación se analiza la aplicación del algoritmo primal-dual de Swamy y Kumar (2004), cuyo pseudocódigo se presenta en el Algoritmo 6, para resolver el problema combinatorio subyacente en un mecanismo incremental, y también determinar el precio para cada usuario. Se debe proponer una función de orden consistente que permita obtener costos incrementales positivos para cumplir con la restricción de no transferencias positivas, y calcular la eficiencia mediante el método propuesto por Brenner y Schäfer (2009).

Algoritmo 6 Algoritmo de Swamy y Kumar (2004)

Entrada: Grafo $G = (V, E)$, demandas $\mathcal{D} \subseteq V$, fuente s, parámetro $M > 1$.

Salida: Árbol de Steiner T, centros $F \subseteq V$ y la asignación $\sigma(j)$ para cada demanda $j \in \mathcal{D}$.

1: Aumentar las variables duales de cada $j \in \mathcal{D}$ hasta que ocurra uno de los siguientes eventos:
2: **si** j se ajusta a una localidad abierta i **entonces**
3: Se congela la variable dual de j.
4: **fin si**
5: **si** Existen M o más encuentros en una localidad i **entonces**
6: Abrir i si no es dependiente en relación a otras localidades ya abiertas. Congelar las variables duales de los usuarios que están ajustados a i.
7: **fin si**
8: Repetir los pasos anteriores hasta que todas las variables duales estén congeladas.
9: Construir el árbol de Steiner T mediante el algoritmo de Prim (1957) en el grafo de distancias más cortas.
10: Asignar cada $j \in \mathcal{D}$ al centro abierto más cercano, determinando $\sigma(j)\ \forall j \in \mathcal{D}$.
11: **devolver** T, F y $\sigma(j)\ \forall j \in \mathcal{D}$.

La metodología empleada para proponer funciones de orden es la siguiente:

- Proponer una función de orden y probarla mediante grafos generados de manera aleatoria. El modelo elegido para la generación de estos grafos es el de Bernoulli (Bollobás, 2001). Se utilizan distintas probabilidades para la adición de una arista y distintos vértices. En vez de considerar ingresos para los usuarios, se utiliza una probabilidad de participación para cada usuario.
  - Si los resultados son positivos, probar teóricamente que la propiedad se cumple para todos los casos.
  - Si se encuentra un contraejemplo, analizarlo y proponer una nueva función de orden en tenga en cuenta lo analizado.

III.1 Propuestas

A continuación se muestran las funciones de orden propuestas para el mecanismo incremental que utiliza el algoritmo de Swamy y Kumar (2004), para finalmente demostrar que existen casos para los cuales no existe una función de orden consistente que permita obtener costos incrementales positivos.

III.1.1 Funciones de orden consistentes

Primera función de orden - Menor distancia a la raíz

En esta función de orden, los usuarios están ordenados de acuerdo a su distancia a la raíz, de manera no-decreciente. Es decir, los elementos más cercanos a la raíz se agregan primero. Es una función de orden consistente "estática", es decir, que está dada desde el comienzo y no depende de la participación de los usuarios. La idea que se

tuvo para esta función de orden es que los vértices más lejanos a la raíz abrirían centros más lejanos a ella, por lo que los costos del árbol de Steiner aumentarían. Pero existen casos simples que demuestran que esta función de orden no obtiene costos incrementales positivos, como se muestra en la Figura 7.

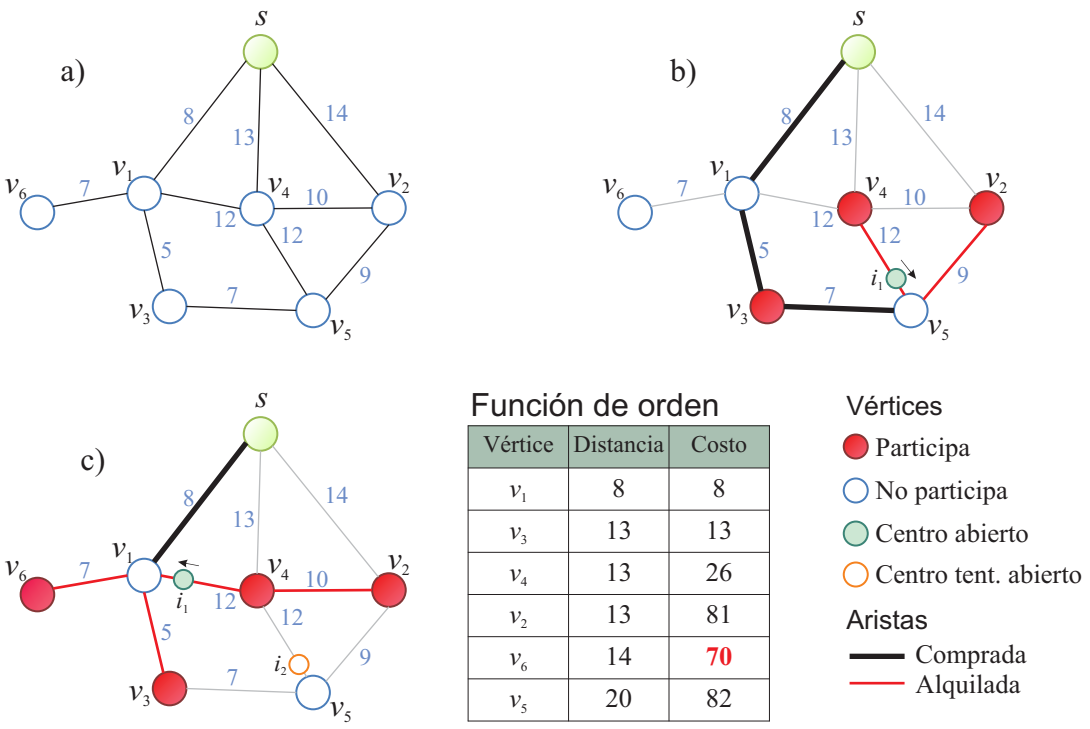


Figura 7. a) Grafo que muestra que la primera función de orden propuesta no obtiene costos no-decrecientes en las soluciones. Se considera $M = 3$. b) Solución antes de agregar a v_6. c) Solución al agregar v_6, el costo incremental es negativo.

Segunda función de orden - Menor costo incremental

En esta función de orden, se elige el usuario que represente el menor costo incremental en cada iteración. La idea es elegir el usuario en el momento en que represente una "ventaja" para el costo grupal, para que el mismo no ocasione un costo incremental negativo al agregarlo en una iteración posterior. Este método resuelve el contraejemplo anterior. Puede notarse que esta función no está dada de antemano, sino que depende

de la participación grupal; pero sigue siendo consistente. Al considerar los conjuntos $S \subset T$, se tiene que en ambos casos se seguirá el mismo orden hasta el elemento que está en T pero no en S. Ésto se debe a que los costos incrementales serán los mismos en cada iteración, por lo que se seguirán eligiendo los mismos usuarios. Pero, de igual manera al caso anterior, se muestra un contraejemplo (Figura 8) para esta función de orden.

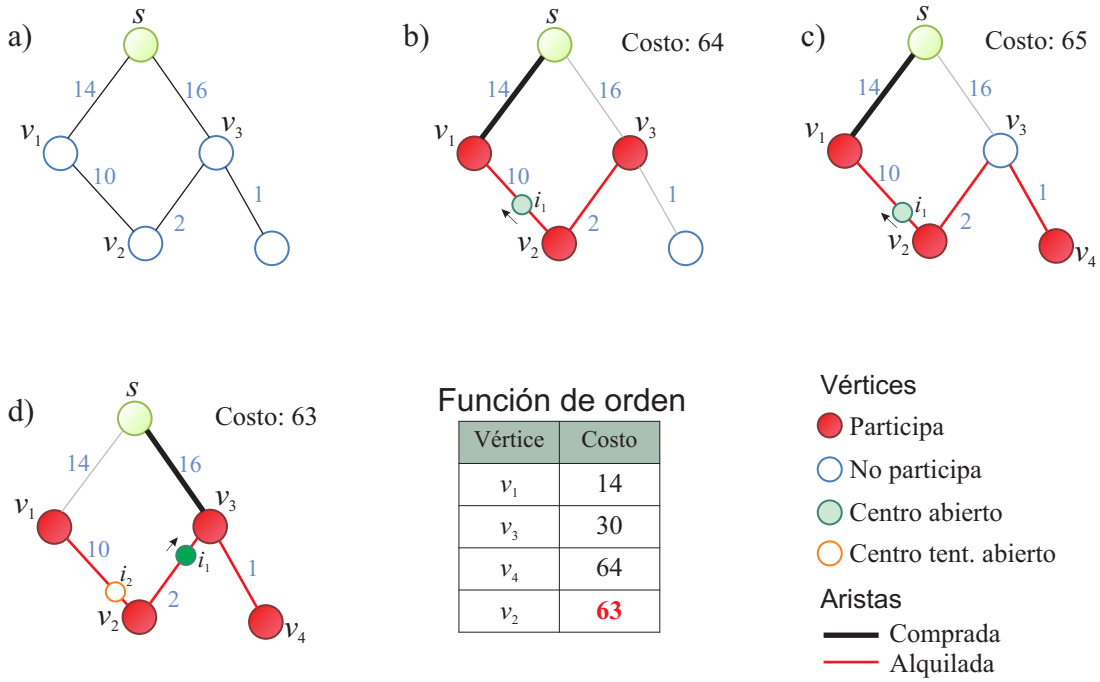


Figura 8. a) Grafo que muestra que la segunda función de orden propuesta no obtiene costos no-decrecientes en las soluciones. Se considera $M = 3$. b) Solución con v_2 en la iteración $t = 3$. c) Solución al agregar v_4, por lo cual se elige a v_2 en $t = 3$. d) Solución al considerar todos los usuarios, el costo es menor que cualquiera de las dos soluciones posibles de la iteración anterior.

Para este contraejemplo, un orden con costos incrementales no-negativos es $(v_3 \rightarrow v_2 \rightarrow v_4 \rightarrow v_1)$. El problema con este método es que se pueden tomar decisiones locales que posteriormente afectarán a los demás usuarios, lo cual ocurre al elegir los vértices v_1 y v_3 en las primeras dos iteraciones. Al elegir estos dos vértices, cualquier elección

posterior hará que el costo incremental en la última iteración sea negativo.

III.1.2 Relajaciones

Como hasta aquí no se han podido encontrar funciones de orden consistentes que permitan obtener costos incrementales positivos, entonces se relaja la restricción de consistencia para encontrar al menos una función de orden con costos de solución no-decrecientes. Además, se considera que todo el conjunto de usuarios posibles participa del juego, independientemente del precio que se les ofrece.

Tercera función de orden - Tiempo de congelamiento de la variable dual

En el trabajo de Mehta *et al.* (2007) se proponen funciones de oferta en base al tiempo de congelamiento de la variable dual. Como el algoritmo de Swamy y Kumar es primal-dual, y los mecanismos incrementales son un caso especial de los acíclicos; entonces se prueba una función de orden determinada por los tiempos de congelamiento de las variables duales. En la Figura 9 se muestra un contraejemplo para esta función de orden, básicamente a partir del contraejemplo para el MST que se muestra en (Brenner y Schäfer, 2009).

La diferencia principal entre los algoritmos vistos en (Mehta *et al.*, 2007) y el algoritmo de Swamy y Kumar (2004) es que este último consta de dos fases, y sólo en la primera se consideran las variables duales. Las decisiones que se toman en la primera fase pueden afectar la forma en que se reparten los costos, lo cual se puede ver en el ejemplo de la Figura 9.

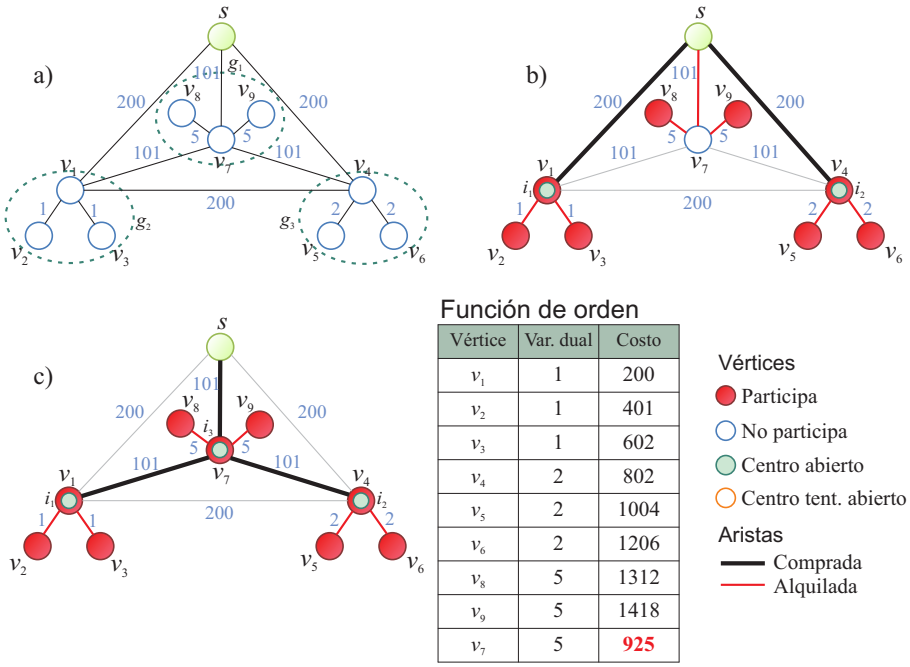


Figura 9. a) Grafo que muestra que la tercera función de orden propuesta no obtiene costos no-decrecientes en las soluciones. Se considera $M = 3$. Se muestran los grupos que tienen el mismo valor de la variable dual. b) Solución con v_9 en la iteración $t = 8$. c) Solución al agregar v_7, el costo incremental es negativo.

Cuarta función de orden - Orden de agregación de conglomerados al MST

Como en el caso anterior el problema está en el orden en que se agregan los conglomerados (representados por el centro al que están conectados en la solución global) al árbol de Steiner; por lo que en esta función de orden se considera el orden en que los centros se agregan al MST mediante el algoritmo de Prim (1957) en el grafo de distancias más cortas de los centros abiertos en la solución global (que considera a todos los potenciales usuarios). En la Figura 10 se muestra un contraejemplo para esta función de orden.

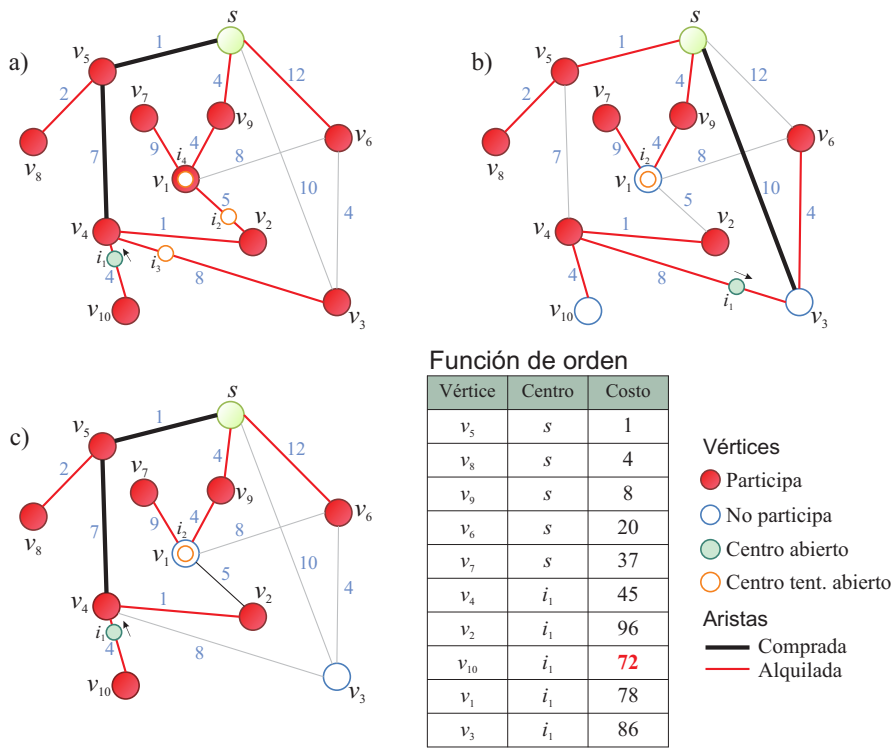


Figura 10. a) Grafo que muestra que la cuarta función de orden propuesta no obtiene costos no-decrecientes en las soluciones. Se considera $M = 3$. Se muestra la solución global. b) Solución con v_2 en la iteración $t = 8$. c) Solución al agregar v_{10}, el costo incremental es negativo.

Quinta función de orden - Orden de agregación de conglomerados al MST - vértices asignados

Se puede notar que en la función de orden anterior, los vértices inicialmente no asignados a un centro (aquellos cuyo centro tentativamente abierto es dependiente de otro) son los que crean centros abiertos en soluciones intermedias, que finalmente se cierran al considerar los vértices de un centro abierto posteriormente. Es por ello que en esta función de orden se hace una ligera modificación: se consideran primeramente los vértices asignados inicialmente a un centro en particular, hasta que se agreguen todos los conglomerados. Finalmente, se van agregando los vértices inicialmente no asignados.

En la Figura 11 se muestra que esta función de orden tiene un contraejemplo, pese

a la modificación considerada. En la solución global, la variable dual correspondiente a v_1 se congela en un valor igual a 1, mientras que en la iteración $t = 5$ se congela en un valor igual a 2. Ésto hace que la variable dual correspondiente a v_7 se congele en un tiempo menor que en $t = 4$, y no permite que se abra el centro que estaba en la iteración $t = 4$.

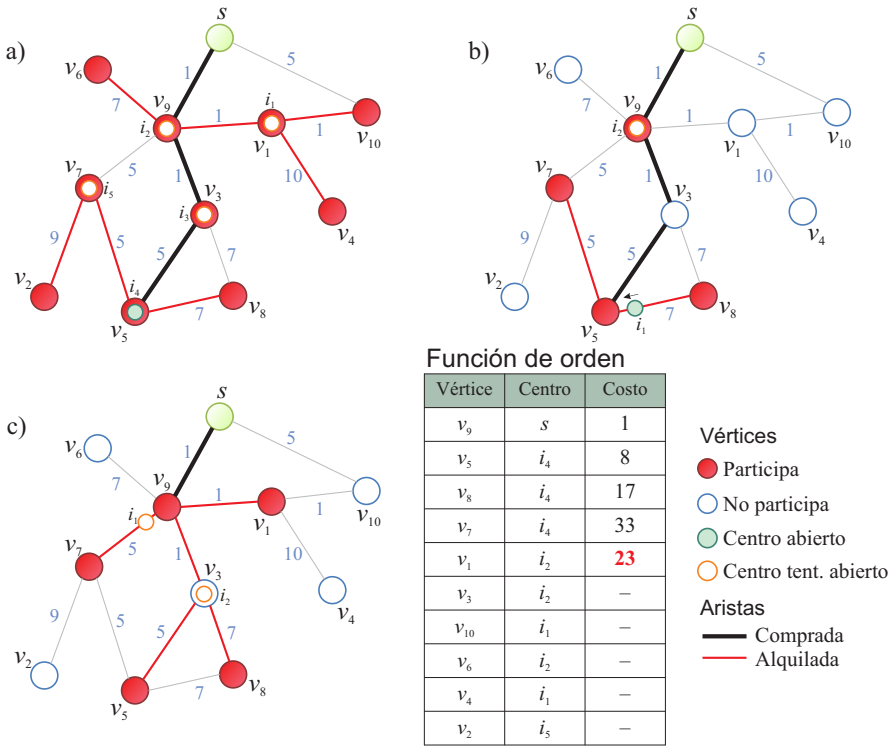


Figura 11. a) Grafo que muestra que la quinta función de orden propuesta no obtiene costos no-decrecientes en las soluciones. Se considera $M = 3$. Se muestra la solución global, con la que queda determinada la función de orden. b) Solución con v_7 en la iteración $t = 4$. c) Solución al agregar v_1, con lo que el costo incremental es negativo.

Sexta función de orden - Menor costo incremental con reordenamiento

En este caso, cada vez que se encuentra un costo incremental negativo, se vuelve a ordenar los vértices. Sea i el vértice con el que se obtiene un costo incremental negativo, y j un vértice ya agregado tal que, al sacarlo, obtiene el menor costo de solución entre

todos los vértices ya agregados. La idea intuitiva es que este vértice puede ser el causante de un costo incremental negativo, ya que puede considerarse como un mínimo local.

Entonces, ese vértice se desplaza al final, y se altera el orden en que se agregan los vértices. Este método soluciona el contraejemplo para la segunda función de orden. De todas formas, existe un contraejemplo para este método, y se muestra en la Figura 12. Como puede notarse, el criterio de ordenamiento no es el correcto.

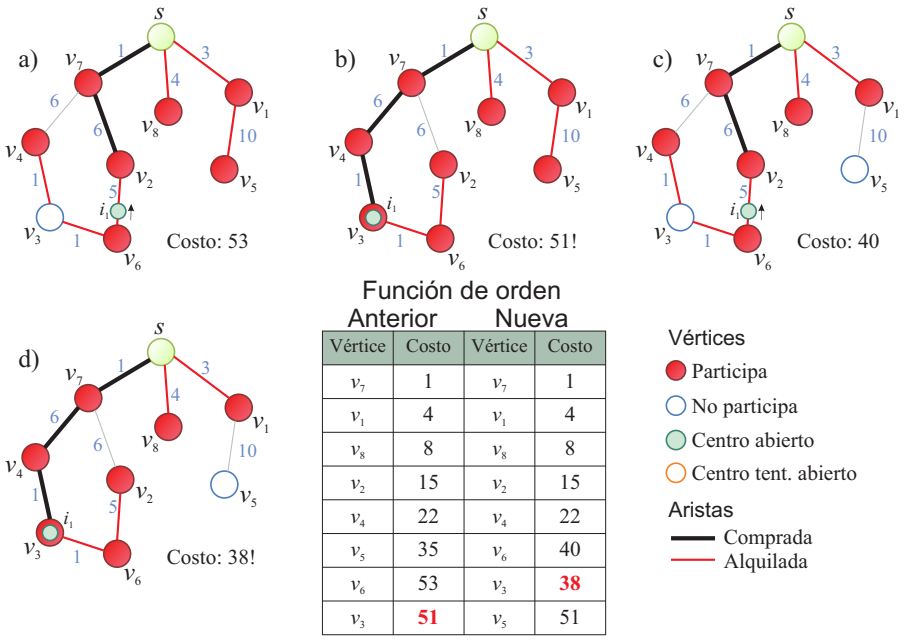


Figura 12. a) Grafo que muestra que la sexta función de orden propuesta no obtiene costos no-decrecientes en las soluciones. Se considera $M = 3$. Hasta la iteración $t = 7$, el orden está determinado por el menor costo incremental en cada iteración. b) En $t = 8$ se tiene un costo incremental negativo, y el orden de adición de vértices se altera, siendo v_5 el vértice con el que, al sacarlo, se obtiene el menor costo en la solución. c) Solución al agregar v_6, en la nueva función de orden. d) Solución al agregar v_3, se sigue obteniendo un costo incremental negativo.

En todos los contraejemplos presentados se puede encontrar una función de orden con costos incrementales crecientes. Con base a las observaciones realizadas, se pueden establecer las siguientes conjeturas:

- Los costos negativos se dan cuando aparece un nuevo centro que hace que otro de la solución anterior se vuelva dependiente a él. Este nuevo centro hace que los costos disminuyan.

- También puede ocurrir que en la nueva solución se cierre un centro de la solución anterior, sin que se abra otro nuevo.

La primera conjetura podría superarse teniendo en cuenta que en la nueva solución, las variables duales del centro de la solución anterior se congelan más tarde, mientras que las del centro "nuevo" más temprano. Pero la segunda se muestra más difícil de superar, y finalmente sirve para la obtención de un contraejemplo que demuestra que existen casos para los cuales no se puede obtener una función de orden que satisfaga la condición de costos no-negativos.

III.1.3 Existencia de la función de orden

Inicialmente se ha intentado obtener una función de orden consistente. Debido a que las propuestas no han tenido éxito en cumplir la condición de costos incrementales no-negativos, se ha relajado la condición de consistencia; considerando además la participación de todos los usuarios. Aún con ello, no se ha logrado obtener una función de orden con la propiedad deseada, por lo que a continuación se plantea la existencia de una función de orden para un caso dado, utilizando un esquema al que se llama "función de orden inversa".

La idea en este punto es demostrar que existe al menos una función de orden para cada caso, o demostrar que existe un caso para el que no se puede obtener una función de orden con costos no-decrecientes en la solución. Finalmente, se muestra un caso para el cual no existe una función de orden que permita satisfacer la condición de costos no-

negativos para un mecanismo incremental. En la Figura 13, se muestra la metodología empleada en la búsqueda de una función de orden.

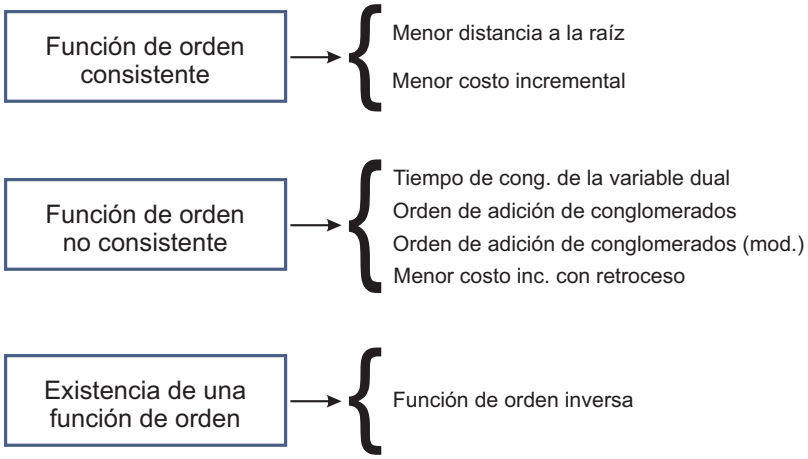


Figura 13. Metodología empleada para la búsqueda de una función de orden.

Función de orden inversa

Esta función de orden se centra principalmente en la existencia de una función de orden para un caso dado. En las funciones de orden planteadas previamente, se iniciaba con un conjunto vacío y se agregaba un jugador del conjunto U a la vez. En el método inverso propuesto:

- Inicialmente se consideran todos los jugadores del conjunto U.
- Se van quitando los jugadores, uno a la vez, siempre que el costo de la nueva solución sea menor o igual que el de la anterior. Esto se hace para obtener costos incrementales no-negativos en cada iteración.
- La función de orden es el inverso del orden de eliminación de jugadores.

Con este método podría obtenerse un contraejemplo que demuestre la inexistencia de una función de orden para el caso general; pues el mismo implicaría que existe un

subconjunto $S \subseteq U$ para el cual no existe una función de orden que permita obtener costos de solución no decrecientes.

A través del método inverso se ha podido obtener una función de orden para cada uno de los contraejemplos propuestos. Además, se han considerado grafos creados de manera aleatoria, en los que siempre se obtuvo un orden de adición de jugadores para obtener costos incrementales no-negativos.

Si bien hasta aquí puede parecer que siempre existe una función de orden para cada caso, considerando las conjeturas realizadas, se pudo obtener un caso para el que el método inverso obtiene costos incrementales negativos; es decir, se pudo obtener un contraejemplo que demuestra que existen casos para los cuales no existe una función de orden con las propiedades deseadas en el mecanismo incremental definido en (Brenner y Schäfer, 2009).

III.1.4 Grafo de contraejemplo

A continuación se presenta el grafo del contraejemplo. En el mismo se tienen siete vértices (siendo s la fuente), y se puede notar que es simétrico en cuanto a costos para los sub-árboles derecho e izquierdo. Además, se considera que $M = 3$. La idea para la creación de este grafo es el hecho de que las variables duales se congelan al intentar abrir un centro.

Como puede verse en la Figura 14, en la solución global no se abrirá centro alguno, haciendo que los centros tentativamente abiertos congelen las variables duales de tal forma que éstas no formen otros centros. Este "equilibrio" se rompe al sacar un jugador. A continuación se recuerdan conceptos importantes del algoritmo de Swamy y Kumar (2004) que ayudan a entender la demostración:

- Un usuario j se encuentra *ajustado* a un centro i si: $\alpha_j \geq c_{ij}$
- Durante la ejecución del algoritmo se aumentan las variables duales hasta que ocurra alguno de los siguientes eventos:
  - El usuario j se vuelve ajustado con una localidad i tentativamente abierta. Entonces se congela α_j.
  - Existe una localidad cerrada i con la que al menos M usuarios están ajustados. Se abre tentativamente i y los usuarios ajustados a dicha localidad se congelan.
- Luego se decide cuáles localidades abrir. Para ello, se selecciona un conjunto maximal de centros independientes. Dos centros son independientes si no existe un usuario que esté ajustado a ambos.

En el contraejemplo se cumple que al agregar al último usuario, el costo incremental siempre es negativo independientemente del orden elegido.

A partir de este resultado, se puede enunciar el siguiente teorema:

Teorema 3. *Para un mecanismo incremental definido según Brenner y Schäfer (2009),* **no existe** *una función de orden que obtenga costos incrementales no negativos utilizando el algoritmo de Swamy y Kumar (2004) para resolver el problema combinatorio subyacente en el* SSROB-CS*.*

A partir de este teorema, se tiene el siguiente corolario:

Corolario 3. *El mecanismo incremental definido en Brenner y Schäfer (2009) no puede utilizarse para resolver el* SSROB-CS*, empleando el algoritmo de Swamy y Kumar (2004) para el problema combinatorio.*

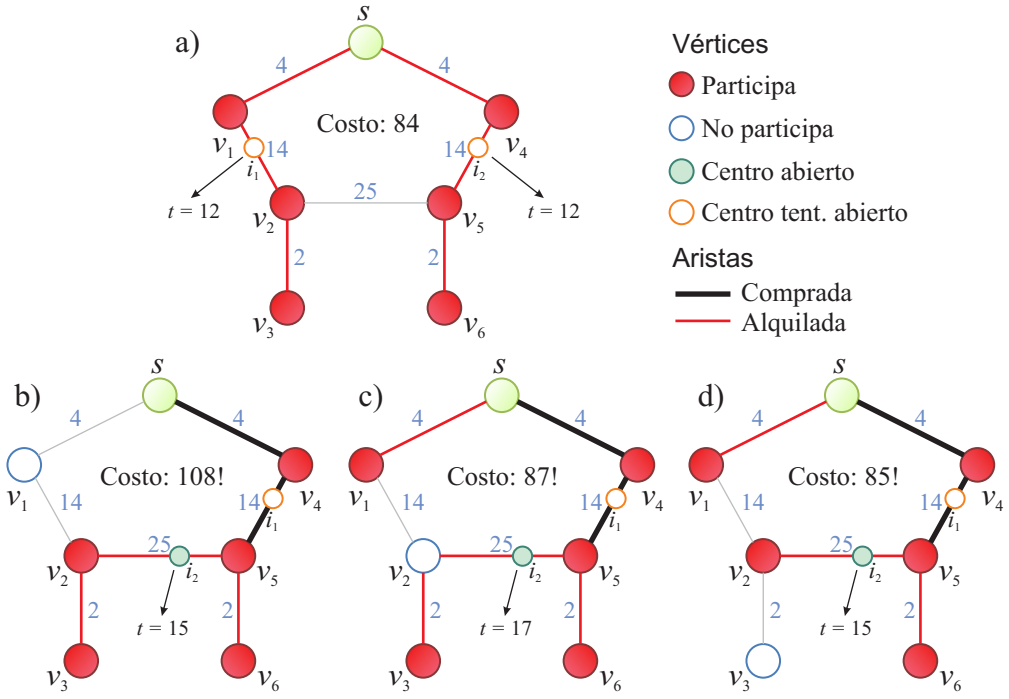


Figura 14. a) Grafo de contraejemplo. Se considera que todos los usuarios participan. b) Solución sin considerar a v_1. c) Solución sin considerar a v_2. d) Solución sin considerar a v_3. En todos los casos, el costo de la solución aumenta.

El teorema implica lo siguiente:

- No puede obtenerse una función de orden consistente que permita aproximar el costo social mediante el método propuesto en (Brenner y Schäfer, 2009), utilizando el algoritmo de Swamy y Kumar (2004) para el problema combinatorio subyacente (SSROB).

Como el algoritmo de Swamy y Kumar (2004) no puede aplicarse a un mecanismo incremental, se analiza su aplicación a un mecanismo acíclico básico. En él, se utiliza un método de reparto de costos canónico con una función de oferta canónica (Mehta *et al.*, 2007). Es decir, ambos están en función de las variables duales del algoritmo de Swamy y Kumar (2004).

III.2 Aplicación del algoritmo de Swamy-Kumar en un mecanismo acíclico para el SSROB-CS

En el Apéndice B se muestra el análisis del algoritmo de Swamy y Kumar (2004) para el SSROB considerando un algoritmo primal-dual para la Fase 2. Se demuestra que:

$$costo(OPT) \leq costo(T) \leq 3\sum_{j\in\mathcal{D}} \alpha_j^{(1)} + 2\sum_{j\in\mathcal{D}} \alpha'_j \leq 5 \cdot costo(OPT) \tag{8}$$

Donde:

- $\alpha_j^{(1)} \rightarrow$ tiempo de congelamiento de la variable dual en el agrupamiento de la Fase 1.
- $\alpha'_j = \max(\alpha_j^{(2)} - c_{\sigma(j)j}, 0) \rightarrow$ siendo $\alpha_j^{(2)}$ el tiempo de congelamiento de la variable dual en la construcción del árbol de Steiner en la Fase 2.

Diviendo por 5 la ecuación (8):

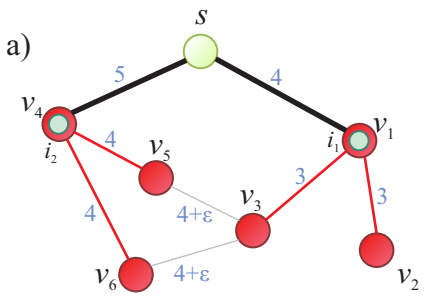
$$\frac{1}{5} \cdot costo(OPT) \leq \frac{3}{5}\sum_{j\in\mathcal{D}} \alpha_j^{(1)} + \frac{2}{5}\sum_{j\in\mathcal{D}} \alpha'_j \leq costo(OPT) \tag{9}$$

Si se utiliza la ecuación (9) como método de reparto de costos en un mecanismo acíclico básico, se asegura la recuperación de al menos 1/5 del costo de la solución óptima. En otras palabras, el factor del equilibrio del presupuesto es igual a 5.

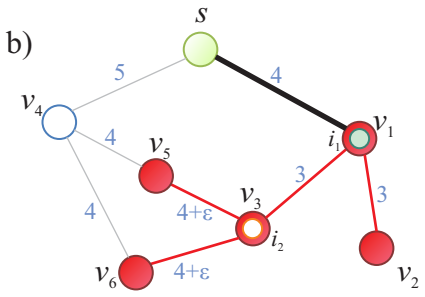
A través de un contraejemplo se demuestra que este método de reparto de costos induce a una función de oferta no-válida, por lo que el algoritmo de Swamy y Kumar (2004) no puede utilizarse en un mecanismo acíclico básico.

III.2.1 Grafo de contraejemplo

Un punto importante para la demostración es que $\alpha_j^{(2)}$ es igual a 0 si el usuario no se encuentra ajustado a algún centro abierto. En el ejemplo mostrado en la Figura 15, se considera que $M = 3$, y se tienen seis usuarios.



Vértice	$\alpha^{(1)}$	α'	Costo
v_1	3	4	3.4
v_2	3	1	2.2
v_3	3	1	2.2
v_4	4	5	4.4
v_5	4	1	2.8
v_6	4	1	2.8



Vértice	$\alpha^{(1)}$	α'	Costo
v_1	3	4	3.4
v_2	3	1	2.2
v_3	3	1	2.2
v_5	4+ε ≈ 4	**0**	**2.4**
v_6	4+ε ≈ 4	**0**	**2.4**

Figura 15. a) Grafo de contraejemplo. Se considera la iteración inicial, cuando todos los usuarios participan. b) Si el vértice v_4 no acepta el precio que se le ofrece, se tiene un nuevo reparto de costos. En este caso, los costos para los usuarios en v_5 y v_6 decrecen, por lo que la función de oferta canónica no es válida.

Recordando la definición de una función de oferta válida, en ella se debe tener que:

- El precio de un jugador i debe mantenerse fijo mientras se remueven los jugadores con tiempos de oferta mayores al de i.
- El precio sólo puede incrementarse con el borrado de jugadores con tiempos de oferta iguales al de i.

Ésto no se cumple en el ejemplo por lo siguiente: si el vértice v_4 no participa del juego, los vértices v_5 y v_6 se asignan al centro i_1. Pero al no estar ajustados a él, no colaboran en la construcción del árbol de Steiner en la segunda fase. Por lo tanto, su costo decrece y no se tiene una función de oferta válida.

Incluso si se decidiera que estos vértices contribuyan en la Fase 2, los costos de los usuarios v_1, v_2 y v_3 también decrecerían, por lo que tampoco se tendría una función de orden válida.

Ésto sucede porque las variables duales de la Fase 1 son iguales para los 5 vértices restantes, pero la suma de las variables duales en la Fase 2 decrece respecto al de la primera iteración. Por lo tanto, cualquier tipo de distribución de estas variables duales hace que el costo de al menos un usuario sea decreciente respecto al de la iteración anterior. Este contraejemplo igual sirve para el caso en que se utilice el algoritmo de Robins y Zelikovsky (2000) para la Fase 2.

Por lo tanto, se puede puede enunciar el siguiente teorema:

Teorema 4. *El algoritmo de Swamy y Kumar (2004) no puede aplicarse en un mecanismo acíclico básico definido según Mehta* et al. *(2007).*

Debido a que el algoritmo de Swamy y Kumar (2004) no puede aplicarse a un mecanismo incremental, ni a un mecanismo acíclico básico, a continuación se muestran algoritmos más recientes para el SSROB y se analiza su aplicabilidad a un mecanismo incremental para el SSROB-CS.

Capítulo IV

Algoritmos para el problema de diseño de una red de compra-alquiler

Como se muestra en el Capítulo III, el algoritmo de Swamy y Kumar (2004) no puede utilizarse para resolver el problema combinatorio en un mecanismo incremental (Brenner y Schäfer, 2009), ya que existen casos para los que no existe una función de orden con la que se obtengan costos incrementales no-negativos. Por ello, se consideran otros algoritmos propuestos en la literatura, los cuales se basan en un algoritmo aleatorio propuesto por Gupta *et al.* (2003). En estos algoritmos se supone que existe a lo más un jugador en cada vértice.

IV.1 Algoritmo aleatorio de Gupta *et al.* (2003)

Este algoritmo consiste en elegir, de forma aleatoria, los vértices de $\mathcal{D} \subseteq V$ que se toman como centros. Por lo tanto, el problema se reduce a encontrar un árbol de Steiner que conecte estos centros a la fuente, y asignar cada usuario al centro abierto más cercano. En el Algoritmo 7 se muestra el pseudocódigo del algoritmo de Gupta *et al.* (2003).

IV.1.1 Análisis del algoritmo

A continuación se revisa el análisis del Algoritmo 7, como se presenta en (Gupta *et al.*, 2003). El mismo permite realizar algunas observaciones con relación a su aplicación a un mecanismo incremental. Sean:

Algoritmo 7 Algoritmo aleatorio de Gupta *et al.* (2003).

Entrada: Grafo $G = (V, E)$, demandas $\mathcal{D} \subseteq V$, fuente s, parámetro $M > 1$.

Salida: Árbol de Steiner T, instalaciones $\mathcal{D}'$ y la asignación $\sigma(j)$ para cada demanda $j \in \mathcal{D}$.

1: Marcar cada demanda $\mathcal{D} \subseteq V$ con probabilidad de $1/M$. Sea $\mathcal{D}' \subseteq \mathcal{D}$ el conjunto de las demandas marcadas.

2: Construir un árbol de Steiner T de ρ_{ST}-aproximación, siendo $F = \mathcal{D}' \cup \{s\}$ el conjunto de vértices requeridos.

3: Para cada usuario $j \in \mathcal{D}$, obtener la asignación $\sigma(j)$, la cual está dada por el vértice de F más cercano a j.

4: **devolver** T, $\mathcal{D}'$ y $\sigma(j)$ $\forall j \in \mathcal{D}$.

- $OPT(I)$: la solución óptima para un caso dado I del problema.
- F^*: el conjunto de centros abiertos en la solución óptima.
- T^*: el árbol de Steiner de la solución óptima, formado con los vértices requeridos $F^* \cup \{s\}$.
- A^*: la asignación de usuarios a centros abiertos en la solución óptima.
- SOL: la solución entregada por el algoritmo.
- F: el conjunto de centros abiertos por el algoritmo.
- T: el árbol de Steiner de la solución dada por el algoritmo, formado con los vértices requeridos $F \cup \{s\}$.
- A: la asignación determinada por el algoritmo.

Considerando el costo esperado del árbol de Steiner en el paso 2 del Algoritmo 7, se tiene el siguiente lema:

Lema 1. *El costo esperado del árbol de Steiner T es a lo más $\rho_{ST} \cdot costo(OPT)$.*

Demostración. Es suficiente demostrar que el costo esperado del árbol de Steiner óptimo en el conjunto (aleatorio) F es a lo más $costo(OPT)$. Para lograrlo se usa el árbol T^* para construir un árbol (aleatorio) de Steiner sobre F, cuyo costo esperado es a lo más $costo(OPT)$.

Se define un árbol de Steiner T' sobre F como la unión de las aristas de T^* y las aristas del camino más corto de j al vértice i^* elemento de T^* más cercano a j $(j - i^*(j))$ para todo $j \in F \backslash \{s\} \subseteq \mathcal{D}$, donde j se asigna a $i^*(j)$ en OPT. El costo de $T^* \subseteq T'$ es determinísticamente $costo(T^*)$. Para una demanda $j \in \mathcal{D}$, el costo incurrido en comprar el camino más corto $j - i^*(j)$ es $M \cdot c_{i^*(j)j}$ con una probabilidad de $1/M$ (si $j \in \mathcal{D}$), y 0 en caso contrario.

En el peor caso, todos los caminos más cortos comprados son disjuntos, por lo que aplicando la linealidad del valor esperado se tiene que:

$$E[c(T')] = costo(T^*) + \sum_{j \in \mathcal{D}} (1/M) \cdot M \cdot c_{i^*(j)j} = costo(T^*) + costo(A^*) = costo(OPT)$$

Como para la construcción del árbol T se utiliza un algoritmo de ρ_{ST}-aproximación, y además solo se consideran los vértices de F como requeridos, se tiene que:

$$E[c(T)] \leq \rho_{ST} \cdot E[c(T')] \leq \rho_{ST} \cdot costo(OPT)$$

■

Se puede notar que en el Lema 1 se relacionan los costos de las aristas del árbol de Steiner construido con el costo óptimo de asignación. Es por esta razón que para la marcación se consideran sólo los vértices que contienen usuarios.

En el Lema 2 se acota el costo esperado de la asignación de usuarios a centros abiertos, realizado en el paso 3 del Algoritmo 7:

Lema 2. *El costo esperado de la asignación es a lo más* $2 \cdot costo(OPT)$.

Demostración. El costo esperado de las conexiones realizadas en el Paso 3 es independiente del árbol particular de Steiner construido en el Paso 2. Por lo tanto, se puede suponer para el análisis, sin pérdida de generalidad, que el árbol de Steiner del Paso 2 está dado por el árbol de esparcimiento mínimo (MST) en D (en el grafo de las distancias más cortas).

Se puede ver el Algoritmo 7, utilizando el algoritmo para el MST, en una nueva pero esencialmente equivalente manera. En vez de "lanzar monedas" de una vez para todas las demandas, el nuevo algoritmo considera las demandas de a uno en algún orden, y decide aleatoriamente para cada una según dicho orden. Dependiendo de las salidas aleatorias, la demanda: (a) se marca, se agrega a F y se conecta al árbol de Steiner pre-existente; o (b) se conecta al algún vértice previamente marcado en F.

Para decidir el orden de los vértices, se consideran dos conjuntos. Al comienzo de la iteración t, sea A_t el conjunto de los vértices previamente considerados por el algoritmo, y $B_t \subseteq A_t$ el conjunto de los vértices marcados. Inicialmente, $A_1 = B_1 = \{s\}$. En el paso t, se toma el vértice $v_t \in \mathcal{D} \backslash A_t$ más cercano a B_t, y se decide (de manera aleatoria) su inclusión a F. En el primer caso, con probabilidad de $1/M$ se define A_{t+1} y B_{t+1} añadiendo v_t a los conjuntos A_t y B_t, y se actualiza el árbol de Steiner comprando el camino más corto de v_t al vecino más cercano en B_t. En el otro caso, se hace que $A_{t+1} = A_t \cup \{v_t\}$ y $B_{t+1} = B_t$, asignando v_t al vecino más cercano en B_t.

Una observación clave es que el proceso *incremental* por el cual se construye el árbol de Steiner T sobre los centros marcados en F es el algoritmo de Prim (1957) para el MST, ejecutado sobre el grafo de distancias más cortas de los vértices en F. Entonces, este nuevo proceso aleatorio implementa de manera idéntica los dos primeros pasos del Algoritmo 7. El costo de conexión incurrido en este proceso es mayor o igual al del Algoritmo 7; ahora se completa la prueba demostrando que el costo esperado de

conexión de este nuevo algoritmo es a lo más $2 \cdot costo(OPT)$.

Sea la variable aleatoria X_t la que indique el costo de alquiler (asignando v_t al vecino más cercano en B_t) menos el costo de compra (agregando v_t a B_t y conectándolo al árbol de Steiner existente) en el paso t del algoritmo. El valor esperado de X_t, condicionado a los primeros $t-1$ "lanzamientos de moneda" tales que v_t son determinísticamente conocidos es:

$$E[X_t] = \left(1 - \frac{1}{M}\right) \cdot c_{v_t B_t} - \left(\frac{1}{M}\right) \cdot M \cdot c_{v_t B_t} \leq 0 \tag{10}$$

Esta desigualdad se cumple para cualquier salida de las primeras $t-1$ decisiones aleatorias, por lo cual se cumple incondicionalmente que:

$$E[X_t] \leq 0, \forall t \tag{11}$$

Sea $X = \sum_i X_i$ el costo de conexión menos el costo del árbol de Steiner de la solución obtenida por el algoritmo. Por la linealidad del valor esperado, se tiene que:

$$E[X] = E\left[\sum_i X_t\right] = \sum_i E[X_i] \leq 0 \tag{12}$$

Entonces, el costo de asignación esperado del algoritmo incremental es a los más el costo esperado del MST en F. Por el Lema 1, el costo esperado de este árbol es a lo más $2 \cdot costo(OPT)$; ya que para la construcción del árbol de Steiner se utilizó el algoritmo de Prim, que da un factor de 2-aproximación.

Como el costo de asignación del Algoritmo 7 no es menor al costo de asignación del algoritmo incremental, se tiene que:

$$E[costo(A)] \leq 2 \cdot costo(OPT) \tag{13}$$

■

Por último, por los Lemas 1 y 2, se obtiene el Teorema 5:

Teorema 5. *El Algoritmo 7 tiene un factor de aproximación esperado de* $(2 + \rho_{ST})$ *para el problema de diseño de una red de compra-alquiler.*

IV.1.2 Aplicación del algoritmo de Gupta *et al.* (2003) a un mecanismo incremental de reparto de costos

En el análisis de Gupta *et al.* (2003) se muestra una modificación del algoritmo propuesto por los autores. Este algoritmo tiene un factor de aproximación esperado de 4.

Como la versión modificada del algoritmo de Gupta *et al.* (2003) es incremental, y los costos de las aristas son no-negativos, entonces el costo incremental de agregar un vértice a la solución es siempre no-negativo. Por lo tanto, este algoritmo puede utilizarse en un mecanismo incremental de reparto de costos definido en (Brenner y Schäfer, 2009), con una función de orden consistente (suponiendo los mismos resultados de marcación de vértices) que está dada por el vértice $v_t \in \mathcal{D} \backslash A_t$ más cercano a B_t. En este caso, se tiene un mecanismo incremental con un factor *esperado* en el equilibrio del presupuesto de 4 (Dobzinski *et al.*, 2008). El mecanismo incremental que utiliza el algoritmo modificado es el Algoritmo 8.

Algoritmo 8 Mecanismo incremental propuesto a partir del algoritmo modificado de Gupta *et al.* (2003).

Entrada: Grafo $G = (V, E)$, demandas $Q \subseteq V$, fuente s, parámetro $M > 1$, ingresos u_j para cada $j \in Q$.

Salida: Árbol de Steiner T, jugadores que recibirán el servicio Q', instalaciones F, la asignación $\sigma(j)$ para cada demanda $j \in Q'$ y los precios x_j para cada demanda $j \in Q'$.

1: $t = 0, A_t = \{s\}, B_t = \{s\}$, $C = Q$, $Q' = Q$
2: **mientras** $C \neq \emptyset$ **hacer**
3: Elegir el vértice $v_t \in Q \backslash A_t$ más cercano a algún vértice de B_t.
4: Decidir de manera aleatoria la inclusión de v_t a F. La probabilidad de agregarlo a F es $1/M$.
5: **si** $v_t \in F$ **entonces**
6: $p = M \cdot c_{v_t B_t}$ ($c_{v_t B_t}$: costo del camino más corto de v_t al vértice más cercano en B_t)
7: **si no**
8: $p = c_{v_t B_t}$
9: **fin si**
10: **si** $p \leq u_{v_t}$ **entonces**
11: $A_t \leftarrow A_t \cup \{v_t\}$, $t \leftarrow t + 1$
12: $x_{v_t} = p$
13: **si** $v_t \in F$ **entonces**
14: $B_t \leftarrow B_t \cup \{v_t\}$
15: Comprar el camino más corto de v_t al vértice más cercano en B_t
16: **si no**
17: Alquilar el camino más corto de v_t al vértice más cercano en B_t, y hacer $\sigma(v_t)$ igual a ese vértice
18: **fin si**
19: **si no**
20: $x_{v_t} = 0$, $Q' \leftarrow Q' - \{v_t\}$
21: **si** $v_t \in F$ **entonces**
22: $F \leftarrow F - \{v_t\}$
23: **fin si**
24: **fin si**
25: $C \leftarrow C - \{v_t\}$
26: **fin mientras**
27: **devolver** T, F, $\sigma(j)$ $\forall j \in Q'$ y x_j $\forall j \in Q'$.

IV.2 Desaleatorización del algoritmo de Gupta *et al.* (2003)

Ahora se consideran dos trabajos que aplican métodos distintos para la desaleatorización del Algoritmo 7:

- Williamson y van Zuylen (2007): utilizan el método de esperanza condicional (Grimmett y Welsh, 1986) para decidir determinísticamente la apertura de los centros. Según el análisis hecho por los autores y mejorado en (Eisenbrand *et al.*, 2008), el factor de aproximación para el SSROB es de 3.28. Sin embargo, se debe resolver un programa lineal de tamaño exponencial, lo cual no es factible para tamaños arbitrarios del problema (Jung *et al.*, 2008).

- Gupta *et al.* (2008): utilizan el método de marcación t-a-t independiente (Mitzenmacher y Upfal, 2005) para reducir el espacio muestral y poder calcular los costos esperados de forma aproximada en tiempo polinomial. El factor de aproximación para el SSROB en este caso es de 4.6.

El algoritmo desaleatorizado de Gupta *et al.* (2008) se utiliza en un mecanismo de Moulin como el método de reparto de costos. Con él, los autores obtienen el mejor resultado conocido para el problema de reparto de costos en el SSROB. A continuación se dan más detalles de dicho método.

IV.2.1 Algoritmo desaleatorizado de Gupta *et al.* (2008)

Gupta *et al.* (2008) realizan un análisis distinto del Algoritmo 7. Además, también se realiza una ligera variación del algoritmo original, introduciendo un parámetro α para

modificar la probabilidad de marcar el vértice de un jugador como centro. El Algoritmo 9 presenta el pseudocódigo de la modificación del Algoritmo 7.

El parámetro α permite realizar un análisis distinto al presentado en (Gupta *et al.*, 2008). Con este análisis, el factor de aproximación β está dado por:

$$\beta \leq max \left\{ 2(1+\alpha), \frac{2(2e^{2\alpha} + e^{\alpha} - 2)}{e^{2\alpha} - 1} \right\} \tag{14}$$

Algoritmo 9 Algoritmo aleatorio modificado de Gupta *et al.* (2008).

Entrada: Grafo $G = (V, E)$, demandas $\mathcal{D} \subseteq V$, fuente s, parámetro $M > 1$, parámetro $\alpha > 0$.

Salida: Árbol de Steiner T, instalaciones $\mathcal{D}'$ y la asignación $\sigma(j)$ para cada demanda $j \in \mathcal{D}$.

1: Marcar cada demanda $\mathcal{D} \subseteq V$ con probabilidad de α/M. Sea $\mathcal{D}' \subseteq \mathcal{D}$ el conjunto de las demandas marcadas.
2: Construir un árbol de Steiner T de ρ_{ST}-aproximación, siendo $F = \mathcal{D}' \cup \{s\}$ el conjunto de vértices requeridos.
3: Para cada usuario $j \in \mathcal{D}$, obtener la asignación $\sigma(j)$, la cual está dada por el vértice de F más cercano a j.
4: **devolver** T, $\mathcal{D}'$ y $\sigma(j)$ $\forall j \in \mathcal{D}$.

Con un valor de $\alpha = 1.296$, el factor de aproximación es de 4.6. Este factor es mayor al de 3.55 de Gupta *et al.* (2003), pero permite la desaleatorización mediante la reducción del espacio muestral.

Si se utiliza una marcación t-a-t independiente (Mitzenmacher y Upfal, 2005) en lugar de una marcación totalmente independiente, Gupta *et al.* (2008) demuestran que el factor cambia muy poco. Si se considera un valor de $t = a \cdot \log(1/\epsilon)$ (donde a es una constante grande), el lado derecho de la ecuación (14) se multiplica por $(1 + \epsilon)$.

Detalles de la marcación con independencia restringida

Como Gupta *et al.* (2008) finalmente utilizan este algoritmo para el problema combinatorio en un mecanismo de Moulin, entonces consideran un tipo de construcción particular del espacio muestral de tamaño polinomial, que les permite obtener un método de reparto de costos monotónico-cruzado.

Sea $\mathbb{F}$ de tamaño $|\mathbb{F}| \geq n$ (donde n es la cantidad de vértices del grafo); el mismo debe ser lo suficientemente grande de tal forma que $\lceil \alpha|\mathbb{F}|/M \rceil / |\mathbb{F}|$ esté muy cerca de α/M. En particular, se supone que $|\mathbb{F}| \geq \Omega(M \cdot \log(n))$. Sean $\{a_1, a_2, ..., a_{|\mathbb{F}|}\}$ los elementos del campo, con los vértices de V etiquetados por los primeros n elementos $\{a_1, a_2, ..., a_n\}$. Sea $\mathcal{S}$ un subconjunto de $\mathbb{F}$ especificado de antemano, con $|\mathcal{S}| = \lceil \alpha|\mathbb{F}|/M \rceil$. Por ejemplo, puede considerarse $\mathbb{F}$ como un conjunto de tamaño p de los números naturales que van desde 0 hasta $p-1$ (donde p es un número primo). En el conjunto $\mathcal{S}$ pueden estar los primeros $|\mathcal{S}| = \lceil \alpha|\mathbb{F}|/M \rceil$ elementos de $\mathbb{F}$. Las operaciones de suma y producto en $\mathbb{F}$ son entonces la suma mod $|\mathbb{F}|$ y producto mod $|\mathbb{F}|$.

Para obtener una muestra, se genera $w = (x_0, x_1, ..., x_{t-1})$ de $\mathbb{F}^t$, de manera aleatoria y uniforme. Se define la variable aleatoria indicadora Y_i con un 1 si el vértice i es marcado como centro ($i \in F$), y 0 en caso contrario. Se hace que Y_i sea igual a 1 si $\sum_{j=0}^{t-1} x_j a_i^j (\text{mod } |\mathbb{F}|)$ pertenece a $\mathcal{S}$, y Y_i es 0 en caso contrario.

Por la definición de campo, se tiene que:

$$Pr\left[\sum_{j=0}^{t-1} x_j a_i^j (\text{mod } |\mathbb{F}|) \in F\right] = 1 \tag{15}$$

Por lo tanto, considerando el tamaño de $\mathbb{F}$, la probabilidad de que Y_i sea igual a 1 es:

$$Pr[Y_i = 1] = Pr\left[\sum_{j=0}^{t-1} x_j a_i^j (\text{mod } |\mathbb{F}|) \in S\right] = \frac{|\mathcal{S}|}{|\mathbb{F}|} \approx \frac{\alpha}{M} \tag{16}$$

Con ésto se puede notar que la probabilidad de que un vértice se elija como centro no varía. La distribución anterior genera Y_i para cada $i \in V$, pero solo se consideran $|\mathcal{D}|$ de estas variables. De acuerdo a Alon *et al.* (1986); Luby (1996), los Y_i generados de esta forma son t-a-t independientes.

Utilizando el método mostrado, se tiene que la cantidad de muestras distintas es $|\mathbb{F}|^t$. Por más que t sea un valor grande, sigue siendo una constante, por lo que el espacio muestral es de tamaño polinomial, y el costo esperado de la solución puede calcularse en tiempo polinomial. Debe existir al menos una muestra que proporcione una solución con un costo menor o igual al esperado (Mitzenmacher y Upfal, 2005), por lo que este algoritmo obtiene soluciones en tiempo polinomial.

Aplicación del algoritmo a un mecanismo de Moulin

Al igual que en el trabajo de Leonardi y Schäfer (2004), Gupta *et al.* (2008) proponen una función de reparto de los costos *esperados*. La idea intuitiva es la siguiente: cada jugador paga un costo proporcional al costo esperado de considerarlo en la ejecución del algoritmo. Para un conjunto dado de marcaciones aleatorias, los jugadores en $\mathcal{D}'$ pagarán por comprar el MST, donde sus aportes estarán dados por el esquema de reparto de costos ξ_{MST} para el árbol de Steiner propuesto por Jain y Vazirani (2001a). Todos los demás jugadores (los que se encuentren en $\mathcal{D}\backslash\mathcal{D}'$) pagarán por alquilar sus caminos más cortos a F. El reparto de costos definido para una salida aleatoria particular cubre exactamente el costo de la solución construida. El costo asignado a un jugador $j \in \mathcal{D}$ es el siguiente:

$$\xi(\mathcal{D}, j) = \frac{1}{\beta} E[M \cdot \xi_{MST}(j, F) + c_{j,F}] \tag{17}$$

donde $c_{j,F}$ es el costo del camino más corto desde j hasta el vértice más cercano a él en $F = \mathcal{D}' \cup \{s\}$, y el valor esperado está en función de todas las salidas aleatorias posibles.

Este método de reparto de costos es monotónico cruzado y tiene un factor de aproximación en el equilibrio del presupuesto de $\beta = 4.6$. Este factor es el menor conocido para el problema de reparto de costos en el SSROB.

IV.2.2 Aplicación del algoritmo a un mecanismo incremental

El autor de esta tesis propone la aplicación del algoritmo de Gupta *et al.* (2008) en un mecanismo incremental, por lo que se debe encontrar una función de orden consistente que permita obtener costos incrementales no-negativos.

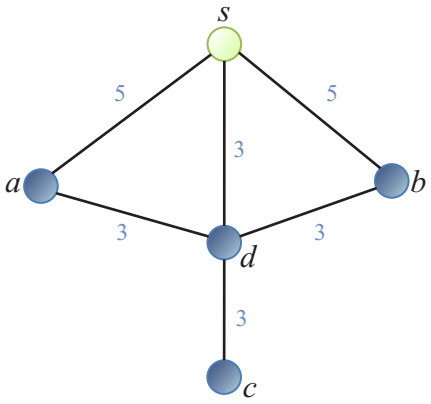
La metodología empleada por el autor de esta tesis para encontrar una función de orden fue la misma que para el algoritmo de Swamy y Kumar (2004)

- Proponer una función de orden.
- Probar que la misma obtenga costos incrementales no-negativos con casos generados aleatoriamente.
  - Si los resultados son positivos, probar teóricamente que la propiedad se cumple para todos los casos.
  - Si se encuentra un contrajemplo, analizarlo y proponer una nueva función de orden que tenga en cuenta lo analizado.

Considerando que $t = a \cdot \log(1/\epsilon)$, donde a es una constante suficientemente grande y ϵ debe ser pequeño (ya que el factor de aproximación se multiplica por $(1+\epsilon)$, entonces este algoritmo se aplica a entradas de gran tamaño. Si se considera una entrada de

tamaño $2 \leq n \leq t$, y teniendo en cuenta que $|\mathbb{F}| \geq n$; se tiene que la cantidad de combinaciones a considerar es igual a $|\mathbb{F}|^t$, mientras que considerando una independencia total de las variables aleatorias, la cantidad de combinaciones a considerar es igual a 2^n. Por lo tanto, para la primera parte de las pruebas se han considerado casos pequeños e independencia total de las variables aleatorias. Además, se considera que $\alpha = 1.296$ y $M = 3$.

A través del ejemplo de la Figura 16 se muestra que este algoritmo de aproximación no es no-decreciente. Por lo tanto, se debe encontrar una función de orden que asegure costos incrementales no-negativos.



Función de orden

Vértice	Costo
a	9.32
b	18.64
c	29.82
d	**29.04**

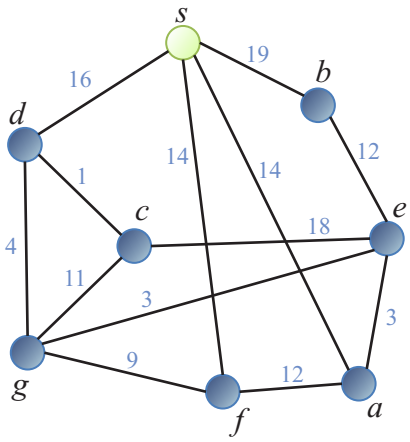
Figura 16. Grafo que muestra que el algoritmo de Gupta *et al.* (2008) no es creciente. Se muestran la función de orden y los costos de agregar esos vértices a la solución.

IV.2.3 Propuestas

A continuación se muestran las propuestas del autor de esta tesis de funciones de orden consistentes, utilizando el algoritmo de Gupta *et al.* (2008).

Primera función de orden: ordenamiento por distancias no-decrecientes a la raíz

Esta función de orden está determinada por las distancias de los usuarios a la raíz, ordenadas en forma no-decreciente. Se muestra un contraejemplo en la Figura 17 que permite realizar la siguiente observación: el costo esperado decrece cuando un vértice agregado se convierte en un "punto de encuentro"; es decir, el vértice agregado hace que los costos esperados de asignación y/o del árbol de Steiner decrezcan (como ocurre en las Figuras 16 y 17).



Función de orden

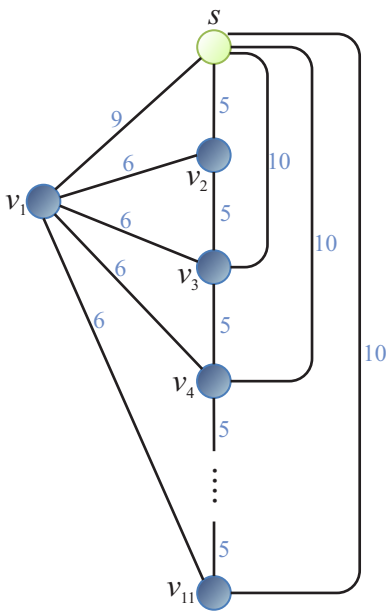
Vértice	Costo
a	26.09
b	50.09
c	72.37
d	83.87
e	89.39
f	115.84
g	**114.62**

Figura 17. Contraejemplo para la función de orden determinada por las distancias a la raíz, ordenadas en forma no-decreciente. Se muestran la función de orden y los costos de agregar esos vértices a la solución.

Lo que ocurre en este caso es que el vértice g se convierte en un punto de encuentro, que hace que decrezcan los costos esperados de asignación y del árbol de Steiner. En este grafo en particular, puede notarse que el vértice g tiene varias aristas de bajo costo, por lo que un criterio de orden podría ser el orden de adición de los vértices al MST mediante al algoritmo de Prim (1957). En este caso, el vértice g sería agregado en una iteración anterior y los costos serían no-decrecientes.

Segunda función de orden: vértice de menor costo de conexión

La idea intuitiva de esta función de orden es que siempre se agrega el vértice de menor costo de conexión al conjunto de vértices ya considerados. Pero incluso para este método pueden encontrarse contraejemplos, como el que se muestra en la Figura 18. Entonces, el hecho de considerar esta función de orden no impide que se tenga algún punto de encuentro (como el vértice v_1 del ejemplo), por más que su costo de conexión al conjunto de los vértices ya considerado sea más alto que el de los demás. Es decir, a parte de los costos de las aristas conectadas a un vértice, se debe considerar el grado del mismo.



Función de orden

Vértice	Costo
v_2	14.91
v_3	28.78
v_4	42.70
...	
v_{11}	140.12
v_1	**138.70**

Figura 18. Contraejemplo para la función de orden determinada por la adición de los vértices al MST mediante el algoritmo de Prim. Se muestran la función de orden y los costos de agregar esos vértices a la solución.

Tercera función de orden: vértice de menor costo incremental

Considerando los puntos débiles de las dos funciones anteriores, se considera una función de orden de *menor costo incremental*. En este método, en cada iteración se elige

el vértice con el menor costo esperado incremental; es decir, se consideran todos los vértices que aún no se han agregado, uno a la vez, y se elige el que obtenga el menor costo esperado de la solución. A diferencia de los métodos anteriores, esta función no se determina desde el inicio del mecanismo, sino que puede variar en relación a la participación de los usuarios en el juego. Aún así, esta función de orden es consistente.

La idea intuitiva de este método es incluir los puntos de encuentro desde el momento en que representan una "ventaja" para los costos de asignación y los del árbol de Steiner de los vértices que ya fueron agregados al conjunto de usuarios. Por ello, este método resuelve los problemas de las funciones anteriores.

En las pruebas, el autor de esta tesis no ha encontrado contraejemplos para esta función de orden, y por lo tanto se conjetura que la función de orden es válida para cualquier caso del problema. A continuación, el autor de esta tesis propone dos caminos para demostrar que la función de orden de menor costo incremental siempre obtiene costos incrementales no-negativos.

IV.2.4 Demostración de obtención de costos incrementales no-negativos con la función de orden de menor costo incremental

Suponga que se tiene una función f con la que se obtienen costos incrementales no negativos hasta una cierta iteración. Sean:

- U: conjunto de usuarios posibles.
- $E[C(S)]$: el costo esperado de dar servicio a los usuarios en $S \subseteq U$.
- Q: conjunto de usuarios agregados hasta la iteración $t-1$.

- k: usuario agregado en la iteración t.
- i: usuario agregado en la iteración $t+1$.

Se realiza la siguiente proposición:

Proposición 1. *Sea* $k = \underset{j \in U \setminus Q}{arg\ min} \Big[E[C(Q \cup \{j\})]\Big]$. *Entonces:*

$$E[C(Q \cup \{k,i\})] \geq E[C(Q \cup \{k\})]\ \forall i \in U \setminus (Q \cup \{k\})$$

Demostración. La demostración se realiza por contradicción. La idea es demostrar que no se puede dar el siguiente caso: $E[Q \cup \{k,i\}] < E[Q \cup \{k\}] \leq E[Q \cup \{i\}]$.

Suponiendo que el costo incremental es negativo al agregar el usuario i en la iteración t, se tiene que:

$$E[C(Q \cup \{k\})] > E[C(Q \cup \{k,i\})] \tag{18}$$

Por la propiedad de partición en la esperanza condicional (Grimmett y Welsh, 1986), al agregar el usuario i se tiene que:

$$E[C(Q \cup \{k,i\})] = \frac{\alpha}{M} E[C(Q \cup \{k,i\}) | i \in F] + \left(1 - \frac{\alpha}{M}\right) E[C(Q \cup \{k,i\}) | i \notin F] \tag{19}$$

Primer camino seguido

Considerando el segundo sumando del lado derecho de la ecuación (19), se cumple que:

$$E[C(Q \cup \{k,i\}) | i \notin F] = E[C(Q \cup \{k\})] + E[C(asig(i, Q \cup \{k\}))] \geq E[C(Q \cup \{k\})] \tag{20}$$

Donde $E[C(asig(i, Q \cup \{k\}))]$ es el costo esperado de asignar el usuario i a algún elemento del conjunto $Q \cup \{k\}$. Por lo tanto:

$$E[C(Q \cup \{k,i\})] \geq \frac{\alpha}{M} E[C(Q \cup \{k,i\}) | i \in F] + \left(1 - \frac{\alpha}{M}\right) E[C(Q \cup \{k\})] \tag{21}$$

Análogamente, considerando al usuario k:

$$\begin{aligned} E[C(Q \cup \{k,i\})] &= \frac{\alpha}{M} E[C(Q \cup \{k,i\}) | k \in F] + \left(1 - \frac{\alpha}{M}\right) E[C(Q \cup \{k,i\}) | k \notin F] \\ &\geq \frac{\alpha}{M} E[C(Q \cup \{k,i\}) | k \in F] + \left(1 - \frac{\alpha}{M}\right) E[C(Q \cup \{i\})] \\ &\geq \frac{\alpha}{M} E[C(Q \cup \{k,i\}) | k \in F] + \left(1 - \frac{\alpha}{M}\right) E[C(Q \cup \{k\})] \end{aligned} \tag{22}$$

Por la ecuaciones (21) y (22), la condición necesaria para que se cumpla la suposición de la ecuación (18) es que:

$$\begin{cases} E[C(Q \cup \{k,i\}) | i \in F] < E[C(Q \cup \{k\})] \\ E[C(Q \cup \{k,i\}) | k \in F] < E[C(Q \cup \{k\})] \end{cases} \tag{23}$$

Considerando nuevamente la propiedad de partición de la esperanza condicional, se tiene que:

$$\begin{aligned} E[C(Q \cup \{k,i\}) | i \in F] = \frac{\alpha}{M} & E[C(Q \cup \{k,i\}) | k, i \in F] \\ & + \left(1 - \frac{\alpha}{M}\right) E[C(Q \cup \{k,i\}) | i \in F, k \notin F] \end{aligned} \tag{24}$$

De los resultados de los casos de prueba, se pueden hacer las siguientes suposiciones:

$$\begin{cases} E[C(Q \cup \{i\}) | i \in F] \geq E[C(Q \cup \{i\})] \geq E[C(Q \cup \{k\})] \\ E[C(Q \cup \{k\}) | k \in F] \geq E[C(Q \cup \{k\})] \end{cases} \tag{25}$$

Ésto es, el costo esperado al agregar i y condicionarlo como centro es mayor que el costo esperado al condicionarlo a no ser un centro (análogamente para k).

Por lo tanto, para que se cumpla la ecuación (23), de (24) y (25) se debe tener que (análogamente para k):

$$\begin{cases} E[C(Q \cup \{k,i\}) | k, i \in F] < E[C(Q \cup \{i\}) | i \in F] \\ E[C(Q \cup \{k,i\}) | k, i \in F] < E[C(Q \cup \{k\}) | k \in F] \end{cases} \tag{26}$$

Además, para que se cumpla la ecuación (23), de (25) debe tenerse que:

$$\begin{cases} E[C(Q \cup \{k,i\})|i \in F] < E[C(Q \cup \{i\})|i \in F] \\ E[C(Q \cup \{k,i\})|k \in F] < E[C(Q \cup \{k\})|k \in F] \end{cases} \tag{27}$$

Hasta ahora no se ha encontrado algún caso donde se cumpla lo expresado en la ecuación (23), ni algún argumento que impida que la misma se cumpla. Es por ello que se pasa a otro camino para la demostración.

Segundo camino

Se hacen las mismas suposiciones que en el camino anterior:

$$\begin{cases} E[C(Q \cup \{i\})|i \in F] \geq E[C(Q \cup \{i\})] \geq E[C(Q \cup \{k\})] \\ E[C(Q \cup \{k\})|k \in F] \geq E[C(Q \cup \{k\})] \end{cases} \tag{28}$$

Ahora se consideran los vértices k e i, iniciando con un caso particular que podría considerarse como un caso límite.

Caso 1: i es una copia de k. Es decir, tienen aristas a los mismos vértices, y el costo para cada par de ellas es el mismo. Además, el costo entre las mismas es cero. Considerando dos veces la esperanza condicional, tanto para i como para k, se tiene que:

$$\begin{aligned} E[C(Q \cup \{k,i\})] = &\left(\frac{\alpha}{M}\right)^2 E[C(Q \cup \{k,i\})|i,j \in F] \\ &+ \left(\frac{\alpha}{M}\right)\left(1 - \frac{\alpha}{M}\right) E[C(Q \cup \{k,i\})|k \in F, i \notin F] \\ &+ \left(\frac{\alpha}{M}\right)\left(1 - \frac{\alpha}{M}\right) E[C(Q \cup \{k,i\})|i \in F, k \notin F] \\ &+ \left(1 - \frac{\alpha}{M}\right)^2 E[C(Q \cup \{k,i\})|i,k \notin F] \end{aligned} \tag{29}$$

Por las suposiciones hechas en (28), se tiene que:

$$E[C(Q \cup \{k,i\})|i,j \in F] = E[C(Q \cup \{k,i\})|k \in F, i \notin F] = E[C(Q \cup \{k,i\})|i \in F, k \notin F] \tag{30}$$

Por lo tanto, de las ecuaciones (29) y (30) se tiene que:

$$\begin{aligned} E[C(Q \cup \{k,i\})] = \left(1 - \left(1 - \frac{\alpha}{M}\right)^2\right) E[C(Q \cup \{k,i\}) | k \in F, i \notin F] \\ + \left(1 - \frac{\alpha}{M}\right)^2 E[C(Q \cup \{k,i\}) | i,k \notin F] \end{aligned} \tag{31}$$

Debido a que $(1 - \alpha/M) < 1$:

$$E[C(Q \cup \{k,i\})] \geq E[C(Q \cup \{k\})] \tag{32}$$

Por lo que la proposición se cumple. Lo mismo ocurre si existe un costo entre k e i.

Caso 2: Para analizar un caso general, se deben considerar los tres primeros sumandos de la ecuación (29), y analizar los costos esperados en cada caso. Esta situación se ilustra en la Figura 19. Se puede ver que se llega al mismo punto que en el primer camino.

Hasta el momento no se ha encontrado algún argumento que demuestre que la siguiente situación no puede darse:

$$E[C(Q \cup \{k,i\})] < E[C(Q \cup \{k\})] \leq E[C(Q \cup \{i\})] \tag{33}$$

La ecuación (32) indica que se puede dar el siguiente caso: al considerar por separado i y k, los costos incrementales son crecientes; pero al considerarlos juntos, el costo esperado de la solución decrece y el costo incremental es negativo. ■

Suponiendo que se cumple la Proposición 1, se tiene la siguiente conjetura:

Conjetura 1. *La función de orden que elige en cada iteración el vértice de menor costo esperado incremental, obtiene costos incrementales crecientes utilizando el algoritmo de Gupta* et al. *(2008) para resolver el problema combinatorio subyacente.*

Demostración. Sean:

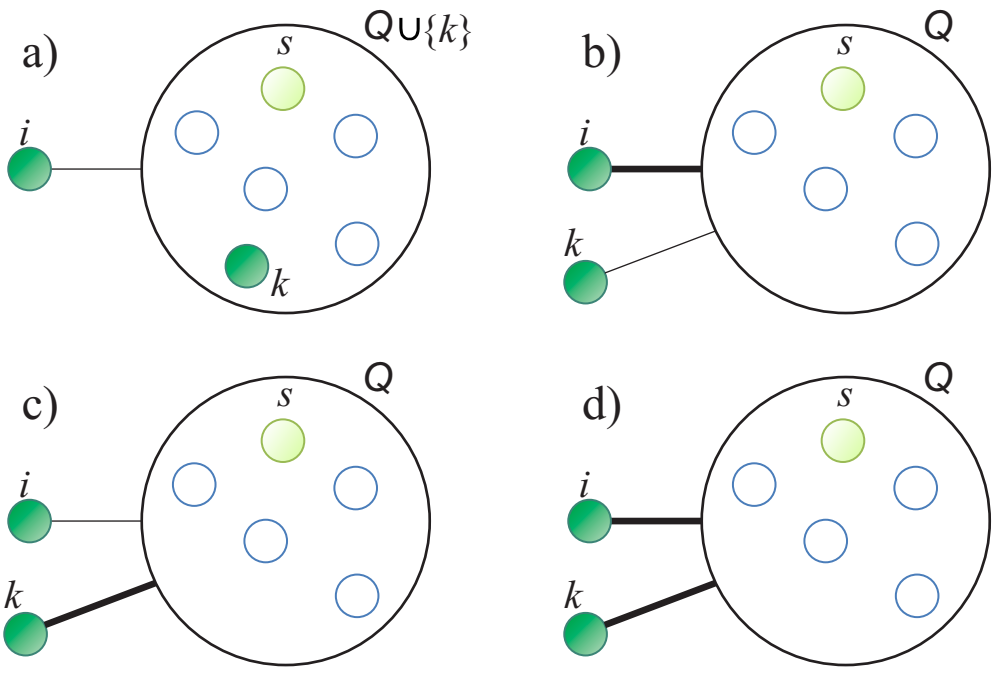


Figura 19. a) Paso inductivo en la demostración de la Proposición 1, donde se agrega el vértice i al conjunto $Q \cup \{k\}$.

- U: conjunto de usuarios posibles.
- $E[C(S)]$: el costo esperado de dar servicio a los usuarios en $S \subseteq U$.
- Q: conjuntos de usuarios agregados hasta la iteración $t-1$.
- j: usuario agregado en la iteración t.

Si se agrega el jugador j en la iteración t, para todo $0 < t \leq |U|$ debe demostrarse que:

$$E[C(Q)] \leq \arg\min_{j \in U-Q} \Big[E[C(Q \cup \{j\})]\Big]$$

La demostración se realiza por inducción.

Caso base: Al inicio, ningún vértice forma parte de Q y $E[C(Q)] = 0$. El costo

esperado al agregar a algún jugador j es:

$$E[C(\{j\})] = \frac{\alpha}{M} E[C(\{j\})|j \text{ es centro}] + \left(1 - \frac{\alpha}{M}\right) E[C(\{j\})|j \text{ no es centro}]$$
$$= \left(\alpha + 1 - \frac{\alpha}{M}\right) c_{js}$$

Como el costo depende únicamente de la distancia del usuario a la raíz, se elige el usuario más cercano a ella. Como los costos son no-negativos y $\alpha \leq M$, entonces el costo incremental es no-negativo y la proposición se cumple.

Paso inductivo: El mismo se demuestra por contradicción. La idea es suponer que en una cierta iteración se tiene un costo incremental negativo al agregar un vértice, y demostrar que ese vértice se debía haber agregado en una iteración anterior.

Sean:

- k: usuario agregado en la iteración t.
- i: usuario agregado en la iteración $t + 1$.

Suponiendo que el costo incremental es negativo al agregar el usuario i en la iteración t, se tiene que:

$$E[C(Q \cup \{k\})] > E[C(Q \cup \{k, i\})] \tag{34}$$

Pero por la Proposición 1, se debe tener que:

$$E[C(Q \cup \{i\})] \leq E[C(Q \cup \{k, i\})] \tag{35}$$

Por lo tanto, de (34) y (35):

$$E[C(Q \cup \{k\})] > E[C(Q \cup \{i\})] \tag{36}$$

Ésto último indica que el vértice k no fue el vértice con menor costo incremental en la iteración anterior, sino i; lo cual es una contradicción. ■

Capítulo V

Desarrollo de experimentos y análisis de resultados

En el Capítulo IV no se pudo demostrar que la función de menor costo incremental en cada iteración permite obtener costos incrementales positivos para todos los casos mediante el algoritmo de Gupta *et al.* (2008) para el problema combinatorio subyacente. Por ello, se prueba este mecanismo incremental de reparto de costos en casos generados aleatoriamente. Se realizarán pruebas para determinar:

- Que el costo del valor esperado, considerando variables completamente aleatorias, es no-decreciente, pese a que de este modo se tiene un tiempo de ejecución exponencial.
- Que el costo del valor esperando, considerando una marcación con independencia restringida de las variables aleatorias, es no decreciente.
- El desempeño del mecanismo incremental en cuanto al costo social, comparándolo con un mecanismo de Moulin.

V.1 Generación de casos de prueba

Los casos de prueba se generaron mediante cinco modelos conocidos en la literatura. Estos métodos determinan cuáles aristas están presentes en el grafo creado $G = (V, E)$.

V.1.1 Modelo de Bernoulli

En este caso, al considerar cada par de aristas posibles (u, v), donde $u, v \in V$, la probabilidad de que esta arista se agregue al conjunto E es p, donde $0 \leq p \leq 1$ (Bollobás, 2001). Es el método más tradicional y conceptualmente simple para la creación de grafos. Un método muy parecido es el de Erdös-Rényi (Bollobás, 2001), donde se elige aleatoriamente una arista (sin reemplazo) del conjunto de aristas posibles, hasta tener m aristas. Las probabilidades consideradas en las pruebas fueron $p = 0.2, 0.4, 0.6$ y 0.8, respectivamente.

V.1.2 Modelo de Barabasi

Este modelo utiliza el concepto de conexión preferencial. Con ello se pretende modelar el crecimiento de los sistemas del mundo real. Lo que se supone es que los vértices que se van conectando al grafo tienen una mayor tendencia a conectarse a los vértices de mayor grado ya presentes. El proceso empieza con un grafo aleatorio (puede ser el de Bernoulli) con $m_0 \geq m$ vértices. Los demás vértices se van agregando uno por uno, cada vértice se conecta a m vértices distintos ya existentes (Barabási y Albert, 1999).

Para incorporar el concepto de conexión preferencial, la probabilidad $p(v)$ de que el vértice agregado se conecte al vértice v existente, depende del grado k_v de v, por lo que $p(v) = k_v / \sum_j k_j$

Por lo tanto, los vértices con más aristas conectadas a él tienen mayor probabilidad de que el nuevo vértice agregado se conecte a él. Aunque en el artículo original hablan de m vértices distintos en cada iteración, se puede modificar el modelo para permitir que se puedan elegir los mismos vértices. Ambos se consideran en las pruebas, con valores de $p = 0.5$ para el grafo aleatorio inicial, con $m_0 = m = 2, 4, 7, 10$.

V.1.3 Modelo de Watts-Strogatz

Los grafos creados con este modelo son también conocidos como redes de "mundo pequeño". Básicamente son redes regulares reordenadas con una cierta probabilidad $0 \leq p \leq 1$ que representa la interpolación entre grafos regulares y los aleatorios. Mediante este modelo se crean grafos con vértices muy cercanos entre sí.

Inicialmente se tiene un grafo regular en un plano, dado por un anillo enlazado donde cada vértice tiene una arista a sus k vecinos más cercanos. Se selecciona un vértice y la arista que lo conecta a su vecino más cercano en el sentido de las manecillas del reloj. Con probabilidad p, se reconecta esta arista con un vértice elegido uniforme y aleatoriamente sobre el conjunto de vértices del anillo, prohibiéndose las aristas duplicadas. De otra forma, se deja la arista como está. Se repite este proceso moviéndose en el sentido de las manecillas del reloj sobre todo el anillo, considerando un vértice a la vez hasta que se consideren todos los vértices del grafo.

Luego, se consideran las aristas que conectan vértices al segundo vecino más cercano en el sentido de las manecillas del reloj. Se repite el proceso anterior de reconexión. Este proceso se realiza hasta que se consideren todas las aristas del anillo original (Watts y Strogatz, 1998). Para las casos de prueba, se consideran $k = 6, 12$, y $p = 0.25, 0.5$.

V.1.4 Modelo de evolución estocástica

En este modelo también se crea un grafo aleatorio de entrada de tamaño $m_0 \geq m$. En cada iteración se agrega un vértice más, y cada uno de ellos se conecta a m distintos vértices de manera uniforme y aleatoria (Kumar *et al.*, 2000). La diferencia principal con el modelo de Barabasi es que no existe el concepto de conexión preferencial. Para los casos de prueba, se consideraron $m = 2, 5, 8, 11$, con una probabilidad $p = 0.5$ para

la creación del grafo aleatorio inicial.

V.1.5 Detalles de los casos de prueba

Para asegurar la conectividad del grafo, se realiza una caminata aleatoria (sin repetición) de los vértices del grafo, agregando aristas si no existe conectividad entre dos vértices.

La cantidad de vértices del grafo es 23 (incluyendo a la raíz). La razón principal es que es un tamaño de entrada que permite considerar varios casos de prueba. Además, es un número primo, necesario para la creación de un campo $\mathbb{F}$ que consiste en números naturales que van desde 0 hasta 22.

Hasta aquí solo se han determinado las aristas que existen, pero no se les ha asignado peso alguno. Para ello, se consideran todas las aristas creadas, y se asignan pesos variables entre 1 y 20.

V.2 Pruebas con variables completamente independientes

Se reparten los incrementos del valor esperado de la solución según el algoritmo aleatorio de Gupta *et al.* (2003). Suponga que se añade un usuario j al conjunto Q de usuarios que recibirán el servicio. El costo ofrecido al jugador j es:

$$\xi(j) = \frac{1}{\beta}(E[c(Q \cup \{j\})] - E[c(Q)])$$

Donde β es el factor de aproximación. Si se consideran variables completamente aleatorias, el factor de aproximación es de 4.6 según Gupta *et al.* (2008).

Lo que se quiere demostrar en estas pruebas es que el costo esperado de la solución es no-decreciente. Para ello, se utilizan en total 12000 grafos generados de manera aleatoria, utilizando los modelos mostrados y todos los distintos parámetros posibles. Además, se considera una participación total de los usuarios. Para el problema se tiene que $M = 3$ y para el algoritmo, $\alpha = 1.296$.

Resultados

No se encontró caso alguno en el que se tengan costos incrementales negativos, lo cual refuerza la Proposición 1. Aún así, podría existir algún contraejemplo.

V.3 Pruebas con variables con independencia restringida

También se reparten los incrementos del valor esperado tomando como espacio muestral a $|\mathbb{F}|^t \geq n$. Suponga que se añade un usuario j al conjunto Q de usuarios que recibirán el servicio. El costo ofrecido al jugador j es:

$$\xi(j) = \frac{1}{\beta}(E[c(Q \cup \{j\})] - E[c(Q)])$$

Donde β es el factor de aproximación. Como se consideran variables con independencia restringida, el factor de aproximación es muy cercano a 4.6 según Gupta *et al.* (2008). Se utilizan nuevamente valores de $M = 3$ y $\alpha = 1.296$. Si se considera un tamaño del conjunto $\mathbb{F}$ igual a 23 (el cual es un número primo), entonces se cumple que $\lceil \alpha|\mathbb{F}|/M \rceil / |\mathbb{F}|$ está muy cerca de α/M. Además, por cuestiones prácticas (en relación al tiempo de ejecución) y tamaño del campo $\mathbb{F}$ en relación al espacio muestral inicial, se considera un valor de $t = 3$.

Resultados

Aunque se utilice un valor de $t = 3$, se obtienen costos no-decrecientes en las soluciones, por lo que los costos incrementales son no-negativos. Este resultado es más interesante que el anterior, porque igual se sigue cumpliendo la restricción de no-transferencias positivas con un espacio muestral reducido, incluso considerando un valor bajo de t.

Al considerar un valor bajo de t, el factor de aproximación en el equilibrio del presupuesto es mayor que β. El autor de esta tesis supone que ésto podría solucionarse con un t suficientemente grande, como se indica en (Gupta *et al.*, 2008), ya que el algoritmo obtendría costos crecientes independientemente del valor de t. Ésto último se debería a una propiedad del algoritmo: para cualquier elección de $w \in \mathbb{F}^t$, si el conjunto de vértices marcados en la ejecución sobre D es D' y en A es A', si $A \subseteq D$, entonces $A' \subseteq D'$. En otras palabras, los vértices considerados como centros en una iteración, seguirán siendo considerados como centros en las demás iteraciones, por lo que se tienen las mismas combinaciones de centros elegidos para las iteraciones posteriores.

El autor de esta tesis conjetura que para demostrar teóricamente la propiedad de costos de soluciones no-decrecientes con este último algoritmo, se debe demostrar la Proposición 1.

V.4 Costo social: comparación con un mecanismo de Moulin

Existen casos particulares para los cuales el mecanismo incremental obtiene un costo social muy alto ($\Omega(n)$, siendo n el número de jugadores), ya que pertenece a un tipo particular de problemas de reparto de costos con una gran submodularidad (Moulin,

1999; Brenner y Schäfer, 2009). Un mecanismo de Moulin con el algoritmo de Gupta *et al.* (2008) obtiene un factor de aproximación del costo social de $O((\log k)^2)$ (Roughgarden y Sundararajan, 2006b). Es por ello que se hace una comparación numérica entre los mecanismos de Moulin e incremental cuando ambos utilizan el algoritmo de Gupta *et al.* (2008) para resolver el problema combinatorio subyacente. Lo anterior sirve para determinar el desempeño relativo en los grafos creados aleatoriamente.

En una primera etapa, se considera que el ingreso de cada jugador está determinado por su distancia a la raíz, ya que la suma de las mismas constituye una cota superior para el óptimo del SSROB, y es lo máximo que un jugador estaría dispuesto a pagar.

En la segunda etapa, el ingreso está determinado por una variable aleatoria entre 1 y la mitad de la distancia del usuario a la raíz. Ésto se hace para forzar a los mecanismos a eliminar usuarios, y observar la forma en que cada uno toma las decisiones durante las ejecuciones respectivas.

En la última etapa, se considera un grupo limitado de grafos, y para cada uno de ellos se tiene un conjunto de vectores de ingresos. Estos ingresos son variables, y sus valores máximos se determinan de tal forma que haya distintos niveles de participación.

V.4.1 Primera etapa: ingresos determinados por la distancia del usuario a la raíz

Para esta etapa se consideraron 600 grafos por cada combinación posible de los parámetros de los modelos de grafos aleatorios a utilizar, lo que da un total de 12000 grafos ($600 \times 5 \times 4$). En cada grafo se considera como ingreso el valor de la distancia del usuario a la raíz.

Resultados

En todos los casos considerados hubo un 100% de participación de los usuarios, y como ambos mecanismos utilizan el mismo algoritmo combinatorio, los resultados en el costo social son similares. El mecanismo incremental obtiene ligeramente resultados mejores en todos los casos debido a que:

- Cuando se realiza el reparto de costos entre los usuarios en el mecanismo de Moulin, se considera la distancia entre un usuario y el centro asignado, independientemente de que existan aristas compradas en el camino.

- En el mecanismo de Moulin se reparte el costo del MST de los vértices requeridos, mientras que en el mecanismo incremental se considera el árbol de Steiner final, que tiene un costo menor o igual al del MST.

La influencia de estas diferencias ligeras entre los mecanismos se puede observar en la Tabla I. En la misma se consideran los promedios de los costos sociales de los 600 grafos para cada modelo y parámetro utilizados.

Debido a que en esta etapa todos los usuarios participan, no se puede analizar el criterio de reparto de costos de cada mecanismo, y cómo ello influye en la participación de los usuarios. Por lo tanto, en la segunda etapa se consideran ingresos variables, para analizar el comportamiento de ambos mecanismos cuando los ingresos son más bajos.

V.4.2 Segunda etapa: ingresos variables para cada usuario

Para esta etapa se considera la misma cantidad de grafos, utilizando los mismos modelos y combinaciones de parámetros. El ingreso para cada usuario varía uniformemente entre 1 y la mitad de la distancia del usuario a la raíz. En este caso, se vuelve a observar

Tabla I. Comparación del costo social promedio entre los mecanismos de reparto de costos.

Modelos	Parámetros	Incremental	Moulin
Bernoulli	$p = 0.2$	49.92	53.67
	$p = 0.4$	32.90	35.39
	$p = 0.6$	25.64	27.56
	$p = 0.8$	21.03	22.59
Barabasi	$m = 2$	56.24	60.17
(con repetición)	$m = 4$	41.80	44.86
	$m = 7$	32.23	34.66
	$m = 10$	27.81	29.92
Barabasi	$m = 2$	53.51	57.26
(sin repetición)	$m = 4$	37.92	40.76
	$m = 7$	28.46	30.57
	$m = 10$	24.33	26.12
Watts-	$k = 6$, $p = 0.25$	41.07	44.12
Strogatz	$k = 12$, $p = 0.25$	26.72	28.72
	$k = 6$, $p = 0.5$	41.47	44.55
	$k = 12$, $p = 0.5$	26.91	28.93
Evolución	$m = 2$	54.89	58.72
estocástica	$m = 5$	34.07	36.59
	$m = 8$	27.68	29.77
	$m = 11$	24.80	26.67

una similitud de resultados entre los mecanismos considerados, tanto en el costo social como en la participación, y los mismos se muestran en la Tabla II.

Resultados

Al observar la Tabla II se puede notar que existe muy poca diferencia en el costo social promedio. Lo mismo ocurre con el porcentaje de participación. Por ejemplo, para el modelo de Barabási (con repetición), la diferencia en el costo social promedio es de 1.57 a favor del mecanismo incremental; mientras que la diferencia en el porcentaje de participación es de sólo 1.1% a favor del mecanismo incremental. De los 2400 grafos creados con este modelo, el mecanismo incremental obtuvo un menor costo social en 2031 de ellos, contra 369 del mecanismo de Moulin; mientras que en 1037 grafos el

Tabla II. Comparación de desempeño entre los mecanismos de reparto de costos.

Parámetros		Bernoulli	Barabási (c.r.)	Barabási (s.r.)	Watts-Strogatz	Evolución Estocástica
Costo social prom.	(I)	32.78	38.78	35.31	34.35	35.07
	(M)	34.22	40.35	36.92	35.83	36.59
Participación	(I)	75.48%	69.29%	72.03%	73.10%	72.82%
	(M)	74.39%	68.19%	70.56%	72.06%	71.63%
Mejor resultado en CS	(I)	2077	2031	2107	2044	2069
	(M)	323	369	293	356	331
Mayor participación	(I)	1032	1037	1096	1024	1059
	(M)	566	610	519	617	559

mecanismo incremental obtuvo una mayor participación, contra 610 del mecanismo de Moulin.

En general, el mecanismo incremental obtiene mejores resultados como en la etapa anterior, principalmente por las razones mencionadas en la misma.

Puede verse que ya existen casos en los que el mecanismo de Moulin obtiene mejores resultados en cuanto a costo social y participación; lo cual hace suponer que éstos pueden acentuarse a medida que los valores de los ingresos vayan decreciendo, y es por eso que se realiza una tercera etapa de pruebas.

V.4.3 Tercera etapa: consideración de distintos niveles de participación de usuarios

En esta última etapa se considera un conjunto limitado de grafos, y lo que varía en forma aleatoria es el ingreso de cada usuario. Se consideran niveles de participación, que determinan en cierta forma la cantidad promedio de usuarios que participan del juego. Se tienen 3 niveles de participación, dependiendo del valor máximo que puede tener el ingreso de cada usuario:

- Alta: los ingresos varían aleatoriamente entre 1 y la distancia del usuario a la

raíz.

- Media: los ingresos varían aleatoriamente entre 1 y un tercio de la distancia del usuario a la raíz.

- Baja: los ingresos varían aleatoriamente entre 1 y un cuarto de la distancia del usuario a la raíz.

El objetivo principal de esta prueba es analizar el comportamiento de los mecanismos a medida que el ingreso va decreciendo, lo cual obliga a los mecanismos a eliminar usuarios del conjunto final. Se consideran 10 grafos en total, 2 grafos por cada modelo. Para cada nivel de participación se consideran 400 ejecuciones.

Resultados

Al considerar una alta participación, el mecanismo incremental obtiene una ligera ventaja en cuanto al costo social en la mayoría de los casos. Ésto se debe a la diferencia mencionada anteriormente en cuanto al funcionamiento de ambos mecanismos.

Pero esta situación cambia a medida que decrecen los ingresos de los usuarios. Puede notarse en las Figuras 20 y 21 que el mecanismo de Moulin va obteniendo mejores resultados para un nivel medio y bajo de participación.

Para algunos grafos (5, 6 y 7; ver Figura 21) con ingresos de bajo nivel de participación, el mecanismo incremental no tiene usuarios debido a que el SSROB es un problema con alta submodularidad (Moulin, 1999; Brenner y Schäfer, 2009). Ésto significa que los usuarios pueden beneficiarse con la presencia de otros usuarios (lo cual no necesariamente se da con los primeros usuarios de un mecanismo incremental), y para aprovechar esta propiedad se diseñó el algoritmo combinatorio de Gupta *et al.*

(2008). Por ello, se puede concluir que el mecanismo de Moulin tiene un mejor criterio de selección de usuarios que un mecanismo incremental de reparto de costos.

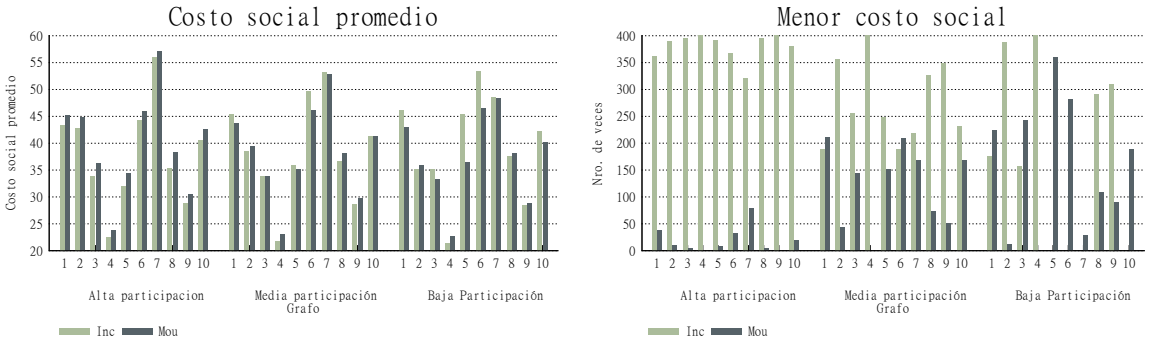


Figura 20. Izq: Gráfica que muestra el costo social promedio para cada nivel de participación. Der: Gráfica que muestra las veces que cada mecanismo obtuvo un menor costo social para cada nivel de participación

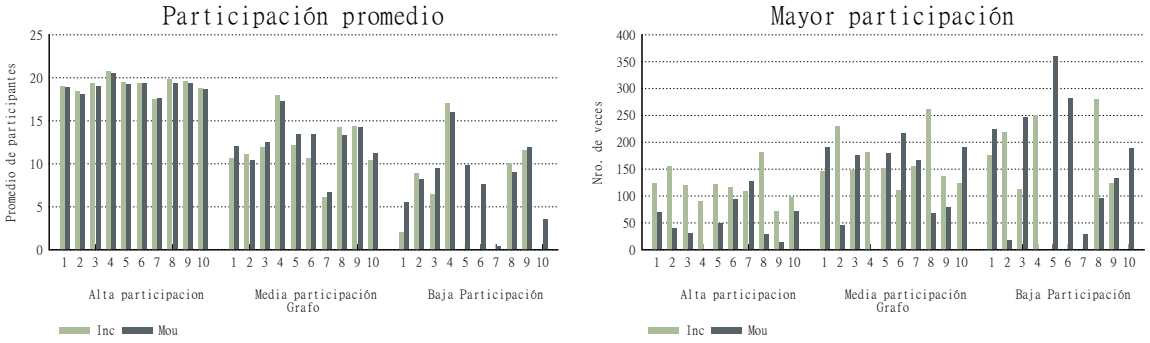


Figura 21. Izq: Gráfica que muestra la cantidad promedio de participantes para cada nivel de participación. Der: Gráfica que muestra las veces que cada mecanismo tuvo una mayor participación para cada nivel de participación

En el Capítulo VI se presenta el resumen y las conclusiones con base en el trabajo realizado así como algunas propuestas de trabajos futuros.

Capítulo VI

Conclusiones y trabajo futuro

VI.1 Resumen

Los mecanismos de reparto de costos en juegos cooperativos determinan, dado un conjunto de usuarios potenciales, los precios a ofrecerse a los usuarios y la solución para el problema combinatorio subyacente considerando al conjunto de usuarios participantes. Los mecanismos de reparto de costos necesitan de algoritmos que resuelvan el problema combinatorio subyacente, y que además determinen el funcionamiento del mecanismo considerado para un caso dado. El diseño de una red de compra-alquiler es un problema **NP-difícil**, por lo cual se utilizan algoritmos de aproximación para dar una solución al problema combinatorio para un cierto conjunto de usuarios participantes. Existen tres puntos importantes para el diseño de mecanismos de reparto de costos que interactúan entre sí: el equilibrio del presupuesto, la eficiencia y que el mecanismos sea a prueba de estrategias. Como no se puede lograr los tres aspectos a la vez, se busca una solución de compromiso entre ellas, y a partir de ésto se definen varios tipos de mecanismos.

El presente trabajo trata acerca de la búsqueda de un mecanismo de reparto de costos para el problema del diseño de una red de compra-alquiler, para mejorar los resultados obtenidos con el algoritmo propuesto por Gupta *et al.* (2008) para un mecanismo de Moulin. Este mecanismo resultante es el mejor conocido en la literatura para el SSROB-CS, con un factor de aproximación del equilibrio del presupuesto de 4.6, y eficiencia de $O(\log^2 n)$, siendo n el número de jugadores.

La idea principal era considerar la aplicación de algoritmos combinatorios ya existentes, que tengan un mejor factor de aproximación que el algoritmo de Gupta *et al.* (2008), en mecanismos de reparto de costos más recientes como los acíclicos (Mehta *et al.*, 2007) y los incrementales (Brenner y Schäfer, 2009).

Se analizó primero un mecanismo incremental, ya que en los mismos se hereda el factor de aproximación del algoritmo para el problema combinatorio subyacente, independientemente del tipo del mismo. Ésto no ocurre con los mecanismos acíclicos, dado que los mismos tienen aplicación principalmente con algoritmos de aproximación primales-duales (Mehta *et al.*, 2007) y la obtención de una función de oferta válida es un problema no-trivial (Brenner y Schäfer, 2009). Además, el mecanismo incremental cuenta con una mayor simplicidad y flexibilidad que los demás mecanismos, ya que el precio ofertado a un usuario está dado por el costo incremental de la solución.

Con un mecanismo incremental, el objetivo es encontrar una función de orden de agregación de usuarios que permita obtener costos incrementales no-negativos para satisfacer la condición de precios no-negativos.

Inicialmente se consideró el algoritmo de Swamy y Kumar (2004), el cual tiene un factor de aproximación de 4.55 para el problema combinatorio subyacente, pero se demostró que se pueden tener casos para los cuales no exista una función de orden.

Como este algoritmo es primal-dual, se consideró su aplicación en un mecanismo acíclico básico Mehta *et al.* (2007). Pero de nuevo, se demostró que existen casos para los cuales la función de oferta no es válida. Por lo tanto, el algoritmo de Swamy y Kumar (2004) no puede utilizarse en un mecanismo incremental ni en un mecanismo acíclico básico.

Por ello, se consideró la aplicación de otros algoritmos de aproximación en un mecanismo incremental. Uno de ellos es el algoritmo aleatorio de Gupta *et al.* (2003), que sí

puede aplicarse en un mecanismo incremental, obteniéndose un factor de aproximación esperado en el equilibrio del presupuesto de 4.

Además, se consideró el algoritmo propuesto por Gupta *et al.* (2008). Este algoritmo es una versión desaleatorizada del algoritmo aleatorio de Gupta *et al.* (2003), y se utiliza en un mecanismo de Moulin para el SSROB-CS. Se propuso una función de orden que agrega el jugador con menor costo incremental en cada iteración. No se han encontrado contraejemplos mediante casos de prueba generados aleatoriamente bajo distintos modelos. Se propusieron varios caminos posibles que podrían conducir a la demostración de que esta función de orden obtiene costos incrementales no-negativos para cualquier caso.

Se realizó también una comparación del costo social (parámetro relacionado con la eficiencia) que logra obtener la propuesta anterior con el que se obtiene con el mejor mecanismo de reparto de costos para el problema considerado. Se consideraron grafos de prueba generados aleatoriamente para analizar el desempeño promedio de la propuesta. Los resultados indican que no existe una diferencia significativa cuando los ingresos son altos, pero a medida que los mismos decrecen, el mecanismo de Moulin obtiene un menor costo social en la mayoría de los casos considerados.

VI.2 Conclusiones y conjeturas

A continuación se listan las principales conclusiones del presente trabajo:

- El algoritmo de Swamy y Kumar (2004) no puede utilizarse en un mecanismo incremental de reparto de costos para el SSROB-CS, al no existir una función de orden consistente para cualquier caso.

- Este algoritmo tampoco puede utilizarse en un mecanismo acíclico de reparto de costos para el SSROB-CS, ya que la función de oferta canónica no es válida.

- El algoritmo aleatorio de Gupta *et al.* (2003) puede utilizarse en un mecanismo incremental para el SSROB-CS, con una función de orden dada por el vértice más cercano a un centro abierto en cada iteración. Pero su factor de aproximación (4) es esperado, y no se encuentra muy alejado del mejor conocido para el problema (4.6). Además, conjeturamos que la eficiencia será baja, por el efecto de la submodularidad del problema tratado.

- De acuerdo a las resultados experimentales, el algoritmo de Gupta *et al.* (2008) (considerando variables aleatorias independientes) puede utilizarse en un mecanismo incremental para el SSROB-CS, con una función de orden dada por el vértice de menor costo incremental en cada iteración. Aunque no se ha demostrado teóricamente este resultado, conjeturamos que el mismo es válido para todos los casos. El punto central en la proposición es demostrar que siempre existe un orden en cada situación. Es decir, al considerar un cierto conjunto de usuarios, al sacar al menos uno de ellos, el costo esperado de la solución no aumenta.

- Un punto resaltante es que incluso al acotar el espacio muestral, tal como lo hacen Gupta *et al.* (2008), se siguen obteniendo costos incrementales no-negativos considerando la función de orden de menor costo incremental. Ésto puede deberse a una propiedad del algoritmo: en cada iteración se consideran las mismas muestras, por lo que los centros abiertos en iteraciones anteriores se siguen abriendo en la iteración actual, y los cambios solo se dan con el vértice agregado en dicha iteración (Gupta *et al.*, 2008). Para abordar una demostración de este resultado, primero se debe demostrar que la misma se cumple considerando variables

aleatorias independientes.

- En cuanto al costo social, se puede decir que al utilizar el algoritmo de Gupta *et al.* (2008) en los mecanismos incremental y de Moulin para el SSROB-CS, no existen diferencias significativas cuando los ingresos están dados por las distancias de los usuarios a la raíz. Pero a medida que los ingresos van decreciendo, se va notando un mejor desempeño del mecanismo de Moulin. Ésto se debe al efecto de submodularidad del problema, que se hace más notorio cuando los ingresos son bajos. Por lo tanto, el mecanismo de Moulin tiene un mejor criterio de reparto de costos para el SSROB-CS.

- Si bien los mecanismos incrementales tienen mayor flexibilidad y simplicidad que los mecanismos de Moulin, los mismos no tienen un buen desempeño en problemas submodulares. Lo anterior se debe a que los mecanismos de Moulin explotan el hecho de que en estos problemas los usuarios se benefician con la presencia de otros usuarios y reparten más equitativamente los costos; mientras que en un mecanismo incremental, los primeros usuarios siempre tienen un costo relativamente alto, y si no deciden participar del juego, este efecto se traslada a los demás usuarios, por lo que finalmente el costo social será bajo. Conjeturamos que ésto ocurrirá para los problemas de reparto de costos en el MST, el ruteo multicast y en la ubicación de instalaciones.

VI.3 Trabajo futuro

A partir de los resultados del presente trabajo, se presentan algunas ideas como propuestas para trabajo futuro:

- Proponer un algoritmo incremental para el problema combinatorio subyacente, de manera a utilizarlo en un mecanismo incremental de reparto de costos.

- Considerar la aplicación de los mecanismos incrementales a problemas de reparto de costos con funciones de costo supermodulares (por ejemplo, varios tipos de problemas de calendarización), ya que en funciones submodulares pueden tenerse resultados negativos.

- Aplicar los mecanismos acíclicos generales a problemas de reparto de costos, ya que en el trabajo de Mehta *et al.* (2007) solo se consideran mecanismos acíclicos básicos.

- Finalizar la demostración teórica planteada en el Capítulo IV, y demostrar que la misma se sigue cumpliendo para una marcación independiente restringida, como lo sugieren los resultados experimentales.

- Analizar si se puede aproximar el costo social cuando se tienen costos incrementales negativos, siguiendo la idea que en el caso de costos incrementales no-negativos presentan Brenner y Schäfer (2009).

- Proponer un marco de diseño de mecanismos (similar a los mecanismos incrementales), y que sean independientes del tipo de problema de reparto de costos considerado.

Referencias

Agrawal, A., Klein, P., y Ravi, R. (1995). "When trees collide: an approximation algorithm for the generalized Steiner problem on networks". *SIAM Journal on Computing*, **24**(3): 440–456.

Alon, N., Babai, L., y Itai, A. (1986). "A fast and simple randomized parallel algorithm for the maximal independent set problem". *Journal of Algorithms*, **7**: 567–583.

Andrews, M. y Zhang, L. (1998). "The Access Network Design Problem". En *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, páginas 40–49, Washington, DC, USA.

Ausiello, G., Protasi, M., Marchetti-Spaccamela, A., Gambosi, G., Crescenzi, P., y Kann, V. (1999). "*Approximation algorithms for NP-hard problems*". Springer-Verlag New York, Inc., Secaucus, NJ, USA, primera edición. 543 pp.

Barabási, A. y Albert, R. (1999). "Emergence of Scaling in Random Networks". *Science*, **286**(5439): 509–512.

Bollobás, B. (2001). "*Random Graphs*". Cambridge University Press, Cambridge, UK, segunda edición. 498 pp.

Bondareva, O. (1963). "Some applications of linear programming to cooperative games". *Problemy Kibernetiki*, **10**: 119–139.

Brenner, J. y Schäfer, G. (2009). "Cooperative Cost Sharing via Incremental Mechanisms". *Preprint 650, DFG Research Center Matheon, Berlin, Germany*.

Deng, X., Ibaraki, T., y Nagamochi, H. (1997). "Combinatorial optimization games". En *Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms*, páginas 720–729, Philadelphia, PA, USA.

Devanur, N., Mihail, M., y Vazirani, V. (2005). "Strategyproof cost-sharing mechanisms for set cover and facility location games". *Decision Support Systems*, **39**(1): 11–22.

Dobzinski, S., Mehta, A., Roughgarden, T., y Sundararajan, M. (2008). "Is Shapley cost sharing optimal?". En *Proceedings of the 1st International Symposium on Algorithmic Game Theory*, páginas 327–336, Berlin, Heidelberg.

Edmonds, J. (1967). "Optimum branchings". *J. Res. Nat. Bur. Stand.*, **71B**: 233–240.

Eisenbrand, F., Grandoni, F., Rothvoß, T., y Schäfer, G. (2008). "Approximating connected facility location problems via random facility sampling and core detouring". En *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, páginas 1174–1183, Philadelphia, PA, USA.

Feigenbaum, J., Papadimitriou, C., y Shenker, S. (2001). "Sharing the Cost of Multicast Transmissions". *Journal of Computer and System Sciences*, **63**: 21–41.

Gillies, D. (1959). "*Solutions to General Non-Zero-Sum Games*". Contributions to the theory of games, editores: A. Tucker y R. Luce. Princeton Univ. Press - Vol. 4, pág. 47-85, Princeton, NJ.

Goemans, M. y Skutella, M. (2000). "Cooperative Facility Location Games". En *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, páginas 76–85, Philadelphia, PA, USA.

Goemans, M. y Williamson, D. (1995). "A general approximation technique for constrained forest problems". *SIAM Journal on Computing*, **24**(2): 296–317.

Green, J., Kohlberg, E., y Laffont, J. (1976). "Partial equilibirium approach to the free rider problem". *Journal of Public Economics*, **6**: 375–394.

Grimmett, G. y Welsh, D. (1986). "*Probability: an introduction*". Oxford science publications. Clarendon Press, Oxford, UK, primera edición. 211 pp.

Guha, S. y Khuller, S. (1999). "Greedy Strikes Back: Improved facility location algorithms". *J. Algorithms*, **31**: 228–248.

Guha, S., Meyerson, A., y Munagala, K. (2000). "Hierarchical Placement and Network Design Problems". En *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, Vol. 31, páginas 603–612, Washington, DC, USA.

Gupta, A., Kumar, A., y Roughgarden, T. (2003). "Simpler and better approximation algorithms for network design". En *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, páginas 365–372, New York, NY, USA.

Gupta, A., Srinivasan, A., y Tardos, E. (2008). "Cost-Sharing Mechanisms for Network Design". *Algorithmica*, **50**(1): 98–119.

Hochbaum, D. (1996). "*Approximation algorithms for NP-hard problems*". PWS Publishing Co., Boston, MA, USA, primera edición. 624 pp.

Immorlica, N., Mahdian, M., y Mirrokni, V. (2008). "Limitations of cross-monotonic cost-sharing schemes". *ACM Trans. Algorithms*, **4**(2): 24:1–24:25.

Jain, K. y Vazirani, V. (2001a). "Applications of Approximation Algorithms to Cooperative Games". En *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, páginas 364–372, Hersonissos, Grecia.

Jain, K. y Vazirani, V. (2001b). "Primal-Dual Approximation Algorithms for Metric Facility Location and k-Median Problems". *Journal of the ACM*, **48**: 274–296.

Jain, K. y Vazirani, V. (2002). "Equitable cost allocations via primal-dual-type algorithms". En *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, páginas 313–321, Montreal, Quebec, Canadá.

Jung, H., Hasan, M., y Chwa, K. (2008). "Improved Primal-Dual Approximation Algorithm for the Connected Facility Location Problem". En *Proceedings of the 2nd international conference on Combinatorial Optimization and Applications*, páginas 265–277, St. John's, Canadá.

Karger, D. y Minkoff, M. (2000). "Building Steiner trees with incomplete global knowledge". En *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, páginas 613–619, Washington, DC, USA.

Kim, T., Lowe, T., Tamir, A., y Ward, J. (1996). "On the location of a tree-shaped facility". *Networks*, **28**(3): 167–175.

Könemann, J., Leonardi, S., y Schäfer, G. (2005). "A group-strategyproof mechanism for Steiner forests". En *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, páginas 612–619, Vancouver, British Columbia.

Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., y Upfal, E. (2000). "Stochastic models for the Web graph". En *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, páginas 57–65, Redondo Beach, CA, USA.

Labbé, M., Laporte, G., Rodrígues, I., y Salazar, J. (2001). "The median cycle problem". *Technical Report 2001/12, Department of Operations Research and Multicriteria Decision Aid at Université Libre de Bruxelles*.

Lee, Y., Chiu, S., y Ryan, J. (1996). "A branch and cut algorithm for a Steiner tree-star problem". *INFORMS Journal on Computing*, **8**(3): 194–201.

Leonardi, S. y Schäfer, G. (2004). "Cross-monotonic cost sharing methods for connected facility location games". *Theoretical Computer Science*, **326**(1-3): 431–442.

Luby, M. (1996). "A simple parallel algorithm for the maximal independent set problem". *SIAM Journal of Computing*, **15**(4): 1036–1053.

Mahdian, M., Ye, Y., y Zhang, J. (2006). "Approximation algorithms for metric facility location problems". *SIAM J. Comp.*, **36**(2): 411–432.

Meggido, N. (1978). "Cost allocation for Steiner Trees". *Networks*, **8**: 1–6.

Mehta, A., Roughgarden, T., y Sundararajan, M. (2007). "Beyond Moulin Mechanisms". En *Proceedings of the 8th ACM conference on Electronic commerce*, páginas 1–10, San Diego, California, USA.

Mettu, R. y Plaxton, C. (2000). "The online median problem". En *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, páginas 339–348, Redondo Beach, CA, USA.

Mitzenmacher, M. y Upfal, E. (2005). "*Probability and Computing: Randomized Algorithms and Probabilistic Analysis*". Cambridge University Press, New York, NY, USA, primera edición. 352 pp.

Moulin, H. (1995). "*Cooperative Microeconomics: A Game-Theoretic Introduction*". Princeton University Press, New Jersey, USA, primera edición. 440 pp.

Moulin, H. (1999). "Incremental cost sharing: characterization by coalition strategyproofness". *Social Choice and Welfare*, **16**: 279–320.

Moulin, H. y Shenker, S. (2001). "Strategyproof sharing of submodular costs:budget balance versus efficiency". *Economic Theory*, **18**(3): 511–533.

Nisan, N., Roughgarden, T., Tardos, E., y Vazirani, V. (2007). "*Algorithmic Game Theory*". Cambridge University Press, Cambridge, UK, primera edición. 776 pp.

Parkes, D. y Ungar, L. (2000). "Iterative combinatorial auctions: Theory and practice". En *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI)*, páginas 74–81, Austin, Texas, USA.

Pál, M. y Tardos, E. (2003). "Group Strategyproof Mechanisms via Primal-Dual Algorithms". En *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, páginas 584–593, Cambridge, UK.

Prim, R. C. (1957). "Shortest interconnection networks and some generalizations". *Bell System Technical Journal*, **36**: 1389–1401.

Ravi, R. y Selman, F. (1999). "Approximation algorithms for the traveling purchaser problem and its variants in network design". En *Proceedings of the 7th Annual European Symposium on Algorithms*, páginas 29–40, Praga, República Checa.

Roberts, K. (1979). "*The characterization of implementable choice rules*". Aggregation and Revelation of Preferences, editor: J. Laffont. North Holland Publishing Company, Holanda.

Robins, G. y Zelikovsky, A. (2000). "Improved steiner tree approximation in graphs". En *Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, páginas 770–779, San Francisco, CA, USA.

Roughgarden, T. y Sundararajan, M. (2006a). "New trade-offs in cost-sharing mechanisms". En *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, páginas 79–88, Seattle, WA, USA.

Roughgarden, T. y Sundararajan, M. (2006b). "Approximately Efficient Cost-Sharing Mechanisms". *Mimeo, Stanford University*.

Shapley, L. (1967). "On balanced sets and cores". *Naval Research Logistics Quaterly*, **14**: 453–460.

Swamy, C. y Kumar, A. (2004). "Primal-dual Algorithms for Connected Facility Location Problems". *Algorithmica*, **40**(4): 245–269.

Vazirani, V. (2001). "*Approximation Algorithms*". Springer-Verlag, Berlín, Alemania, primera edición. 380 pp.

von Neumann, J. y Morgenstern, O. (1980). "*Theory of Games and Economic Behavior*". Princeton University Press, New Jersey, USA, tercera edición. 648 pp.

Watts, D. y Strogatz, S. (1998). "Collective dynamics of small-world networks". *Letters to Nature*, **393**: 440–442.

Williamson, D. y Shmoys, D. (2011). "*The Design of Approximation Algorithms*". Cambridge University Press, Cambridge, UK, primera edición. 500 pp.

Williamson, D. y van Zuylen, A. (2007). "A simpler and better derandomization of an approximation algorithm for single source rent-or-buy". *Operations Research Letters*, **35**(6): 707–712.

Zalta, E. (2008). "*Stanford Encyclopedia of Philosophy*". Center for the Study of Language and Information, Stanford University. http://plato.stanford.edu/archives/spr2008/entries/game-theory/.

Apéndice A

Juegos cooperativos

A continuación se muestran dos problemas de reparto de costos en juegos cooperativos, detallando un método de solución propuesto en la literatura (Jain y Vazirani, 2001a; Pál y Tardos, 2003) para cada uno. Además, se muestra un ejemplo de aplicación para cada método de solución.

A.1 Juego del árbol de Steiner

En el trabajo de Jain y Vazirani (2001a) se considera primeramente el **problema de reparto de costos del ruteo multicast**. Se tiene un conjunto U de potenciales usuarios, un grafo $G = (V, E)$ donde cada vértice representa un usuario, y las aristas del grafo tienen asociadas un costo. Por ejemplo, considérese que un proveedor desea ofrecer un servicio a un conjunto de usuarios. Para cada conjunto $S \subseteq U$, $C(S)$ denota el costo en el que incurre el proveedor para dar servicio a los usuarios de S. Cada usuario i tiene un ingreso u_i', que sólo él conoce, por recibir el servicio; es decir, está dispuesto a pagar a lo más u_i'. El usuario i obtiene un beneficio $u_i' - x_i$ por obtener el servicio a un precio x_i; si no lo recibe, su beneficio es 0. Se considera que cada usuario es egoísta, por lo que puede dar un falso ingreso u_i de tal forma a maximizar su beneficio.

El objetivo es determinar el conjunto de usuarios $Q \subseteq U$ que recibirán el servicio y a qué precio. Además, se debe determinar un árbol T de G, que contiene a Q y a la fuente s, a través de la cual se encaminará la información para dar servicio a Q.

Se ha demostrado que el mecanismo de reparto de costos marginal puede ser imple-

mentado (Feigenbaum *et al.*, 2001). Aunque este modelo es ampliamente usado para el ruteo multicast, tiene la desventaja de que el sub-árbol que conecta un cierto subconjunto S de receptores puede ser arbitrariamente más costoso que el árbol más barato que une a ellos. Lo último es un árbol de Steiner óptimo que contiene a S y a la fuente. Se ha demostrado que para este juego, el núcleo es vacío (Meggido, 1978) y por lo tanto no existe un método de reparto de costo monotónico-cruzado débil (pues el núcleo está formado por funciones monotónicas cruzadas débiles). Es por ello que se utilizan los algoritmos de aproximación. Utilizando el algoritmo primal-dual de Edmonds (1967) para árboles de esparcimiento, se muestra en (Jain y Vazirani, 2001a) cómo construir una clase de métodos monotónicos-cruzados. Estos métodos están parametrizados por n mapeos, $f_i : \mathcal{R}^+ \rightarrow \mathcal{R}^+$, uno para cada usuario i (Jain y Vazirani, 2001a). Así, el proveedor de servicios puede elegir los métodos. Por ejemplo, puede usar estimaciones de la función de distribución de probabilidad de los ingresos de los usuarios.

También se propone una condición de presupuesto equilibrado de α-aproximación (Jain y Vazirani, 2001a). En el caso del ruteo multicast, se utiliza el factor de 2-aproximación del algoritmo de Steiner para obtener un método de 2-presupuesto equilibrado, y a prueba de estrategias de grupo.

Formalmente, el problema se define de la siguiente manera (Jain y Vazirani, 2001a): Sea $G = (V, E)$ un grafo no dirigido con aristas de peso c_e y un nodo raíz, el cual es la fuente del mensaje. Todos los demás nodos son usuarios, y el conjunto de ellos es U. Se supone que un mensaje puede ser duplicado en cualquier nodo sin costo. Cada arista cobra el precio c_e por transportar el mensaje, y el costo total de difundir un mensaje es el precio total cobrado por las aristas que se utilizan. Se supone también que los pesos satisfacen la desigualdad del triángulo.

Supóngase ahora que la raíz tiene un mensaje para difundir y cada usuario i ha

reportado su ingreso u_i a la raíz. El trabajo de la raíz es calcular lo siguiente:

- Un conjunto Q de usuarios seleccionados para recibir el mensaje.
- Un árbol T, que contiene a Q, para difundir el mensaje.
- Para cada usuario i, el precio x_i a ser cobrado como el costo de enviarle el mensaje.

Las restricciones del juego son las mismas que las definidas para los juegos cooperativos de reparto de costos. Solamente se añade una condición de optimalidad para el árbol T que conecta la raíz a todos los usuarios que reciben el servicio (T es el árbol de Steiner). El problema de encontrar un árbol que satisfaga esta condición es **NP-difícil**, entonces el mismo se relaja al de encontrar un árbol con un costo de a lo más dos veces el costo del óptimo. Un resultado clásico en teoría de juegos muestra que no existe un mecanismo a prueba de estrategias de grupo que sea de presupuesto equilibrado y eficiente (Green *et al.*, 1976; Roberts, 1979). También se ha demostrado que obtener un factor de aproximación constante para la condición de eficiencia (Feigenbaum *et al.*, 2001) es **NP-difícil**, por lo que ésta última es dejada de lado en (Jain y Vazirani, 2001a). Todas las demás condiciones pueden ser obtenidas mediante un método de reparto de costos monotónico-cruzado.

El objetivo del trabajo de Jain y Vazirani es determinar entonces un método de reparto de costos monotónico cruzado para este juego, para posteriormente aplicar el mecanismo de Moulin (1999).

El problema de obtener el árbol de Steiner se resuelve en forma aproximada hallando el árbol de esparcimiento mínimo (MST) de los vértices requeridos. Para ésto, se utiliza el algoritmo de Edmonds (1967) que obtiene el MST de grafos dirigidos (aunque también se puede aplicar a los no-dirigidos, transformando el grafo original $G = (V, E)$

en un grafo dirigido $G' = (V, \bar{E})$, donde las aristas de G están duplicadas) (Jain y Vazirani, 2001a). Se utiliza el esquema primal-dual para los problemas de reparto de costos, ya que las variables duales pueden verse como la contribución que realizan los usuarios para obtener una solución primal factible; o bien, tienen asociadas una noción de tiempo que permite realizar un cálculo de los costos que deben repartirse entre los usuarios.

Se hacen las siguientes definiciones (Jain y Vazirani, 2001a):

- Un conjunto $S \subset V$ es *válido* si es no-vacío y además no contiene a la raíz.
- Para cualesquiera conjuntos $S \subset V$ y $F \subset \bar{E}$ sean $\delta(S) = \{u \to v \in \bar{E} | u \in S \text{ y } v \notin S\}$, y $\delta_F(S) = \{u \to v \in F | u \in S \text{ y } v \notin S\}$.

A continuación se establece la relajación LP y su dual:

minimizar $\sum_{e \in \bar{E}} c_e x_e$	maximizar $\sum_{\text{conjunto S válido}} y_S$
sujeto a: $\sum_{e: e \in \delta(S)} x_e \geq 1$, $\forall$ conjunto válido S	sujeto a: $\sum_{S: e \in \delta(S)} y_S \leq c_e$, $e \in \bar{E}$
$x_e \geq 0$, $e \in \bar{E}$	$y_S \geq 0$, $\forall$ conjunto válido S

Se dice que la arista e *siente* al dual y_S si $y_S > 0$ y $e \in \delta(S)$. Se dice que la arista e es *ajustada* si la cantidad total de duales que siente es igual a su costo. El programa dual está tratando de maximizar la suma de las variables duales y_S sujeto a la condición de que ninguna arista siente más duales que su costo, es decir, ninguna arista está *sobre-ajustada*.

Algoritmo 10 Algoritmo de Edmonds

Entrada: Grafo $G = (V, E)$, y los costos c_e de las aristas.

Salida: Árbol de esparcimiento mínimo T de G.

1: **(Inicialización)** $F \leftarrow \emptyset$; para cada $v \in V$, $y_{\{v\}} \leftarrow 0$.

2: **(Incremento de aristas)** Cuando exista un conjunto insatisfecho:
Hallar todos los conjuntos activos respecto a F. Para cada uno de tales conjuntos S, aumentar su variable dual y_S hasta que alguna arista e se vuelva ajustada; $F \leftarrow F \cup \{e\}$. Para cada nuevo conjunto activo R, hacer: $y_R \leftarrow 0$.

3: Sea e_1, e_2, ..., e_l la lista ordenada de aristas de F.

4: **(Borrado inverso)** Para $j = l$ hasta 1:
Si no existen conjuntos insatisfechos respecto a $F - \{e_j\}$, entonces $F \leftarrow F - \{e_j\}$

Jain y Vazirani (2001a) utilizan el algoritmo de Edmonds, el cual está basado en el esquema primal-dual. Empezando con las soluciones primales y duales triviales, iterativamente mejora la factibilidad del primal y la optimalidad del dual. Cuando una arista se vuelve ajustada, se incluye en una lista ordenada F. En cualquier iteración, se dice que un conjunto $S \subset V$ es *insastifecho* si es válido y si $\delta_F(S) = \emptyset$. Se dice que cualquier conjunto mínimo insatisfecho es un conjunto *activo* (Jain y Vazirani, 2001a).

Para probar propiedades del algoritmo, se asocia una noción de *tiempo* con el mismo. En una unidad de tiempo, el algoritmo crece los duales en una unidad. En una iteración, el algoritmo halla todos los conjuntos activos, y aumenta sus variables duales hasta que una nueva arista, e, se vuelve ajustada. Con ésto la iteración termina, y la arista e es agregada a la lista F. El algoritmo continúa hasta que no haya conjuntos insatisfechos. Luego, se realiza un procedimiento de *borrado inverso* (Edmonds, 1967). Las aristas que quedan en F forman un branching dirigido a la raíz (que constituye el MST para el grafo no dirigido).

Debido a que el algoritmo aumenta variables duales solo para conjuntos fuertemente conectados, y realiza un borrado inverso al final, es posible mostrar que cada conjunto válido S tal que $y_S > 0$, debe satisfacer $|\delta_F(S)| = 1$. Ésto, unido al hecho de que son

tomadas sólo las aristas ajustadas, llevan a mostrar que el costo del MST determinado es precisamente igual al costo dual total:

$$\sum_{e \in \bar{E}} c_e x_e = \sum_{\text{conjunto S válido}} y_S$$

Este punto demuestra que el MST determinado es óptimo. Como consecuencia, se tiene que el LP siempre tiene una solución entera óptima. Ésto ayuda a demostrar que el método de reparto de costos es de presupuesto equilibrado (Jain y Vazirani, 2001a). Entonces, sea Q el conjunto de usuarios que recibirán el servicio. Son dadas las funciones $f_i : \mathcal{R}^+ \to \mathcal{R}^+$, una para cada usuario i. Se usa el algoritmo de Edmonds para hallar un árbol de esparcimiento mínimo que contenga a Q y a la raíz.

Luego se define un método de reparto de costos ξ. El reparto de costos para el usuario $i \in Q$ se define como sigue: sea T el menor tiempo en el que existe un camino que consiste de aristas ajustadas de i a la raíz. En un tiempo $t \leq T$, sea $S(t)$ el conjunto de vértices alcanzables desde i usando aristas ajustadas. En cada tiempo $t < T$, el algoritmo aumenta la variable dual $y_{S(t)}$ a razón de una unidad. Sea:

$$F(t) = \sum_{j \in S(t)} f_j(t)$$

El reparto de costos para el usuario i será (Jain y Vazirani, 2001a):

$$\xi(i, Q) = \int_0^T \frac{f_i(t)}{F(t)} dt$$

Puede notarse que las funciones $f_i : \mathcal{R}^+ \to \mathcal{R}^+$ determinan el grado de contribución que debe realizar un usuario en relación al grupo en que se encuentra. En el caso que se requiera repartir equitativamente el costo del crecimiento de la variable dual, se deben tener funciones idénticas para todos los usuarios.

A.1.1 Ejemplo de aplicación del método

A continuación se muestra un ejemplo de la ejecución del método de reparto de costos propuesto por Jain y Vazirani (2001a). Considérese el grafo G de la Figura 22. La fuente es el vértice r, y se tienen tres usuarios $\{a, b, c\}$. Las aristas tienen asociadas un costo. Inicialmente, todos los subconjuntos que no incluyen a la fuente son insatisfechos. Por lo tanto en $t = 0$, los subconjuntos mínimos son: $\{a\}$, $\{b\}$ y $\{c\}$. En $t = 2$, las aristas $e4$ y $e_4^{'}$ se vuelven ajustadas, ésto se indica en la Figura 23. Los valores $y_{\{a\}} = y_{\{c\}} = 2$ ya no pueden aumentarse.

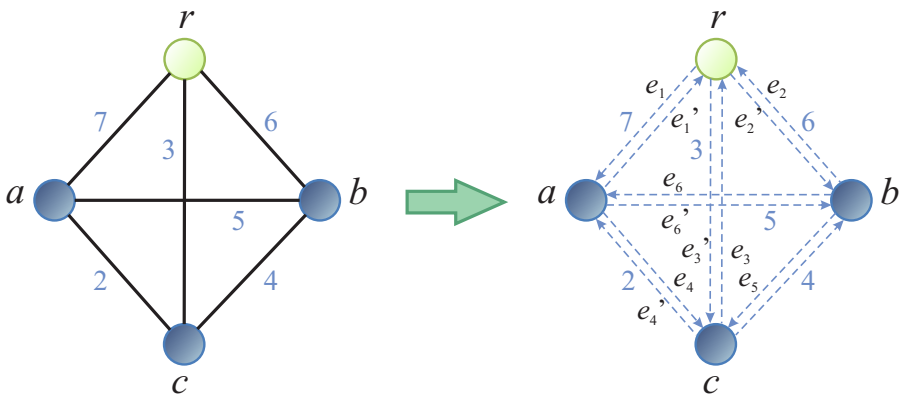


Figura 22. Presentación del ejemplo. La fuente es el vértice r, y se tienen tres usuarios $\{a, b, c\}$. Se transforma el grafo no dirigido G en uno dirigido G', duplicando las aristas de G.

En $t = 2$, los conjuntos activos pasan a ser $\{b\}$ y $\{a, c\}$, por lo que se aumentan sus variables duales. En $t = 3$, la arista e_3 se vuelve ajustada. El valor $y_{\{a,c\}} = 1$ ya no puede aumentarse. En ese instante, el único conjunto activo es $\{b\}$, por lo que su variable dual sigue aumentando hasta $t = 4$, donde la arista e_5 se vuelve ajustada y se tiene que $y_{\{b\}} = 4$. El conjunto de aristas seleccionadas queda como $F = \{e_4, e_4^{'}, e_3, e_5\}$. A través del borrado inverso, se elimina $e_4^{'}$, por lo que el conjunto de aristas del MST resulta: $F = \{e_4, e_3, e_5\}$.

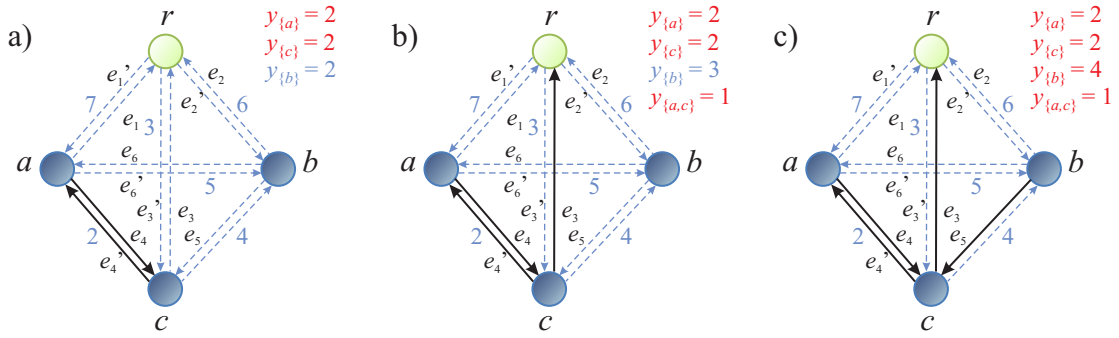


Figura 23. Ejecución del algoritmo de Edmonds: a) En $t = 2$ se seleccionan las aristas e_4 y e_4'. b) En $t = 3$ se selecciona la arista e_3. c) En $t = 4$ se selecciona la aristas e_5, y al eliminar e_4' se obtiene el MST de G.

Considerando funciones f_i idénticas para cada usuario, así como los tiempos en los que cada uno se conecta a la fuente, se tienen los siguientes valores para el reparto de costos:

$$\xi(Q, a) = \int_0^2 1dt + \int_2^3 \frac{1}{2}dt = 2.5.$$
$$\xi(Q, b) = \int_0^4 1dt = 4$$
$$\xi(Q, c) = \int_0^2 1dt + \int_2^3 \frac{1}{2}dt = 2.5$$

Como el costo del MST es igual a 9, se puede comprobar que la suma de los precios calculados mediante el método es igual al costo del MST.

Mediante un teorema (Jain y Vazirani, 2001a), se demuestra que este método de reparto de costos ξ es monotónico cruzado. Si este método se emplea en un mecanismo de Moulin, se cumple con NPT, VP, CS, es de presupuesto equilibrado y a prueba de estrategias de grupo. Sin embargo, la condición de presupuesto equilibrado dado más arriba es difícil de conseguir (Jain y Vazirani, 2001a). Es por ello que se consideran los métodos de γ-presupuesto equilibrado. Jain y Vazirani (2001a) demuestran que el mecanismo de Moulin cumple con NPT, VP, CS, es de γ-presupuesto equilibrado y a

prueba de estrategias de grupo. Debido a que el factor de aproximación del algoritmo que calcula el árbol T para servir a Q (árbol de Steiner) es de $\gamma = 2$, el mecanismo propuesto es de 2-presupuesto equilibrado.

A.2 Juego de ubicación de instalaciones

La definición es la siguiente (Nisan *et al.*, 2007): se tiene un conjunto A de *agentes* (también conocidos como ciudades, clientes o puntos de demanda), un conjunto F de instalaciones, el costo de apertura de una instalación f_i para cada instalación $i \in F$, y una distancia d_{ij} entre cada par (i, j) de puntos en $A \cup F$ indicando el costo de conectar j con i. Se supone que las distancias provienen de un espacio métrico, es decir, son simétricos y cumplen con la desigualdad del triángulo. Para un conjunto $S \subseteq A$ de agentes, el costo de este conjunto es definido como el costo mínimo de abrir un conjunto de instalaciones y conectar cada agente de S a una instalación abierta. Formalmente, la función de costo c es definida por $c(S) = \min_{F' \subseteq F}\{\sum_{i \in F'} f_i + \sum_{j \in S} \min_{i \in F'} d_{ij}\}$

Al ser un problema de reparto de costos en un juego cooperativo, el mismo tiene muchas similitudes en cuanto a los objetivos y restricciones señalados en el juego del árbol de Steiner. Como en el juego anterior, el objetivo principal es obtener un método de reparto de costos que sea monotónico cruzado, de manera a emplear nuevamente el mecanismo de Moulin (1999). La optimalidad para este problema consiste en obtener una solución de ubicación de instalaciones de menor costo posible para dar servicio a todos los usuarios que aún participan del juego. Como obtener una solución óptima para el costo del servicio es **NP-difícil** (Goemans y Skutella, 2000), se utilizan métodos aproximados. Existen varios algoritmos de aproximación para el problema de ubicación de instalaciones (Mahdian *et al.*, 2006; Jain y Vazirani, 2001b; Mettu y Plaxton, 2000;

Guha y Khuller, 1999).

A continuación se establece la relajación LP y su dual, donde las variables x_i y y_{ij} son las variables relajadas de las que indican si una instalación i está abierta, y si el agente j se conecta a la instalación i, respectivamente:

$$\text{minimizar} \sum_{i \in F} f_i x_i + \sum_{i \in F} \sum_{j \in A} d_{ij} y_{ij} \qquad \text{maximizar} \sum_{j \in A} \alpha_j$$

$$\text{sujeto a:} \sum_{i \in F} y_{ij} \geq 1, \forall j \in A \qquad \text{sujeto a: } \beta_{ij} \geq \alpha_j - d_{ij}, \forall i \in F, j \in A$$

$$x_i \geq y_{ij}, \forall i \in F, j \in A \qquad \sum_{j \in A} \beta_{ij} \leq f_i, \forall i \in F$$

$$x_i, y_{ij} \geq 0, \forall i \in F, j \in A \qquad \alpha_j, \beta_{ij} \geq 0, \forall i \in F, j \in A$$

Se puede notar, sin pérdida de generalidad, que en cualquier solución factible del problema dual, $\beta_{ij} = \max(0, \alpha_j - d_{ij})$. Por lo tanto, para especificar una solución dual solo se necesita dar $\boldsymbol{\alpha}$. Se puede ver que el problema dual captura las restricciones del núcleo del problema de ubicación de instalaciones (Goemans y Skutella, 2000). Para cualquier solución factible $\boldsymbol{\alpha}, \boldsymbol{\beta}$ del problema dual, se satisface la condición de núcleo del problema de ubicación de instalaciones (Nisan *et al.*, 2007). Por lo tanto, el error de redondeo da el mejor factor del presupuesto equilibrado para una asignación de costos que satisface las condiciones del núcleo. Los mejores resultados en el campo de algoritmos de aproximación muestran que el intervalo, en el peor caso, está entre 1/1.52 (Mahdian *et al.*, 2006) y 1/1.463 (Guha y Khuller, 1999).

Al diseñar un algoritmo primal-dual para el juego de ubicación de instalaciones, se puede ver a la cantidad $\beta_{ij} = \max(0, \alpha_j - d_{ij})$ como la contribución del agente j a la

instalación i, y decir que una instalación i es ajustada con respecto a la solución dual $\boldsymbol{\alpha}$ si la contribución total que recibe i en $\boldsymbol{\alpha}$ iguala a su costo de apertura, es decir, si $\sum_{j \in A} \max(0, \alpha_j - d_{ij}) = f_i$ (Nisan *et al.*, 2007).

Siguiendo el paradigma general del esquema primal-dual (igual que para el árbol de Steiner), se considera el siguiente algoritmo para calcular el reparto de costos en el juego de ubicación de instalaciones: se inicializan las variables de reparto de costos α_j a cero y gradualmente se incrementan hasta que ocurra alguno de los dos eventos: una instalación i se vuelve ajustada, en este caso, se abre dicha instalación y los repartos de costos α_j de todos los agentes j con contribución positiva a i son *congelados* (es decir, no se vuelven a incrementar). En el otro caso, si para un agente j y una instalación i que ya está abierta, se tiene que $\alpha_j = d_{ij}$, entonces el reparto de costo del agente j es congelado. Este proceso continúa hasta que todas las variables de reparto de costos α_j estén congeladas (Nisan *et al.*, 2007).

Este método de reparto de costos no es monotónico cruzado (Pál y Tardos, 2003): al añadir nuevos usuarios cerca de otro usuario ya existente i, hace más fácil que el agente i sea satisfecho, y una vez que i deja de aumentar su variable dual (que es su contribución al costo), los demás agentes restantes pueden verse afectados negativamente.

La idea que permite obtener repartos de costos monotónicos cruzados es tener una variable de reparto *fantasma* para cada usuario, y fue establecida por Pál y Tardos (2003). Aunque en el algoritmo original se obliga a detener el incremento de la contribución del usuario de manera a no sobrecargar algún conjunto de usuarios, se sigue aumentando la variable de contribución fantasma hasta que todos los usuarios estén satisfechos. Aplicando esta técnica al juego de ubicación de instalaciones, la contribución extra de los fantasmas hace que se abran más instalaciones y que los usuarios se conecten más rápido que en el algoritmo original. Esta idea asegura trivialmente la monotonici-

dad cruzada, pues al añadir más usuarios (y sus fantasmas) puede solo ocasionar que se abran más instalaciones, y que cada usuario sea satisfecho antes. El desafío principal es mostrar que incluso aunque la mayoría de las instalaciones solo son virtualmente pagadas por los fantasmas, los costos repartidos aún pagan lo suficiente para construir una solución factible (Pál y Tardos, 2003).

Algoritmo 11 Algoritmo de Pál y Tardos

Entrada: Grafo $G = (\{A \cup F\}, E)$, donde F es el conjunto de instalaciones y A el conjunto de usuarios, los costos de apertura de las instalaciones y de las aristas de G.

Salida: Conjunto $F' \subseteq F$ de instalaciones a ser abiertas, asignación de usuarios a las instalaciones abiertas y determinación de la contribución de cada usuario.

1: **(Inicialización)** Para cada $j \subseteq A$, $\alpha_j^{'} \leftarrow 0$.

2: **(Incremento de variables fantasmas)** Cuando exista un usuario no satisfecho: Aumentar uniformemente $\alpha_j^{'}$ para todo usuario $j \subseteq A$. Si una instalación p se vuelva ajustada, hacer $t(p) \leftarrow \alpha_j^{'}$. Si no existe una instalación q abierta tal que $d_{pq} \leq 2t(p)$, entonces p es abierta.

3: **(Determinación de la contribución)** Calcular la contribución α_j de cada usuario j, donde $\alpha_j = \min\left(\min_{p:j \in S_p} t(p), \min_{p:j \notin S_p} d_{jp}\right)$

Para resolver el problema, Pál y Tardos utilizan el algoritmo de Mettu y Plaxton (2000). El mismo puede ser interpretado como un algoritmo primal-dual, aunque no lo es (Pál y Tardos, 2003). Se hace crecer una "esfera" uniforme para cada usuario, cuyo radio es el valor de la variable dual fantasma. Una vez que un usuario "alcanza " una instalación, empieza a llenarla. Sea $t(p)$ el tiempo en el que la instalación p se vuelve ajustada durante la ejecución del algoritmo. Para cada instalación p, sea S_p el conjunto de clientes vecinos tales que si los mismos contribuyen el mismo costo $\alpha = t(p)$ pueden pagar por la conexión a p y también por su costo de apertura. O bien, puede definirse como el conjunto de usuarios que contribuyen a "llenar" p, es decir, los que están a una distancia menor que $t(p)$ de p. No se puede cobrar el valor de α a cada usuario

del conjunto, pues los mismos pueden tener otras instalaciones más cercanas a las que prefieren contribuir, pero la idea es hacer que los usuarios de S_p paguen lo suficiente de tal forma a que si se abre p, se pueda recuperar una fracción constante del costo (Pál y Tardos, 2003).

Entonces, se aumenta la variable dual α_j de cada usuario j hasta que la variable fantasma de j alcance alguna instalación satisfecha, o alguna instalación que haya alcanzado se llene. Más formalmente:

$$\alpha_j = \min\left(\min_{p:j\in S_p} t(p), \min_{p:j\notin S_p} d_{jp}\right)$$

Debido a que la cantidad de instalaciones abiertas es mayor que en el algoritmo original, se puede aplicar un post-procesamiento para determinar cuáles instalaciones quedan abiertas, y así mejorar la calidad de la solución. Para cada instalación p se decide si abrirla o no en el momento en que p se vuelve ajustada. La instalación p queda abierta si y sólo si no existe una instalación ya abierta q de tal forma que $d_{pq} \leq 2t(p)$. Ésto hará que las instalaciones abiertas estén lo suficientemente lejanas entre sí. Luego, se asigna cada usuario a la instalación más cercana (Pál y Tardos, 2003).

Debido a que con ésto se logra obtener una función monotónica cruzada, con el mecanismo de Moulin se consigue que el mecanismo de reparto de costos sea de NPT, VP, CS y a prueba de estrategias de grupo. Además, se demuestra que se recupera al menos 1/3 del costo, por lo que el mecanismo es de 1/3-presupuesto equilibrado (Pál y Tardos, 2003). Se ha demostrado que este factor es el mejor posible para esquemas de reparto de costos monotónicos cruzados (Nisan *et al.*, 2007).

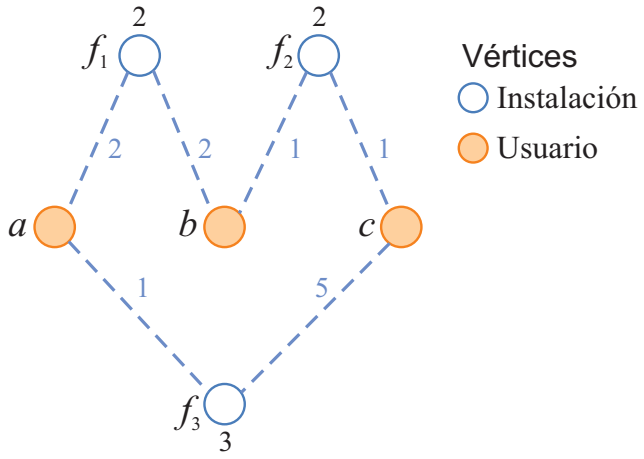


Figura 24. Presentación del ejemplo. En el grafo G se tienen tres usuarios $\{a, b, c\}$ al igual que tres instalaciones posibles $\{f_1, f_2, f_3\}$. Las aristas representan los costos de acceso a las instalaciones. Los vértices que representan las instalaciones tienen indicados los respectivos costos de apertura.

A.2.1 Ejemplo de aplicación del método

A continuación se muestra un ejemplo de la ejecución del método de reparto de costos propuesto por Pál y Tardos (2003). Considérese el grafo G de la Figura 24. En el grafo G se tienen tres usuarios $\{a, b, c\}$, al igual que tres instalaciones posibles $\{f_1, f_2, f_3\}$. Las aristas representan los costos de acceso a las instalaciones. Los vértices que representan las instalaciones tienen indicados los respectivos costos de apertura. Al igual que en el juego de Steiner, asociamos una noción de tiempo a las variables duales. Inicialmente, en $t = 0$, todos los usuarios son insatisfechos, por lo que se aumentan las variables duales.

En $t = 2$, la instalación f_2 es satisfecha por lo que $t_{f_2} = 2$, y es abierta por ser la primera. Siguiendo el paradigma general en un algoritmo primal-dual, las variables duales correspondientes a los usuarios b y c deberían congelarse. Pero en el paradigma de las variables duales fantasmas, las mismas continúan creciendo de manera a contribuir para la apertura de otros centros. Puede verse que a partir de $t = 2$, el usuario b

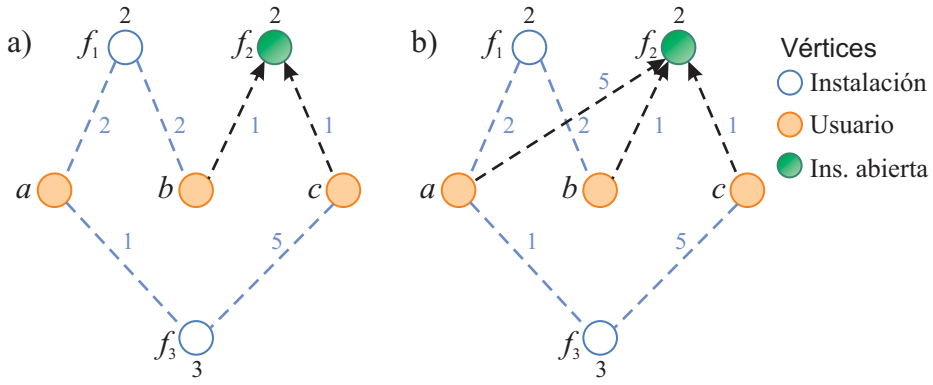


Figura 25. Ejecución del algoritmo de Pál y Tardos: a) En $t = 2$ se llena la instalación f_2 y la misma es abierta, y se asignan los usuarios b y c a f_2. b) En $t = 3$ se llena la instalación f_1 con la ayuda de la contribución fantasma del usuario b. Pero esta instalación no es abierta pues $d_{f_1 f_2} = 3 \leq 6 = 2t(f_1)$, y se asigna el usuario a a f_2.

contribuye para la apertura de la instalación f_1.

En $t = 3$ la instalación f_1 es satisfecha por lo que $t_{f_1} = 3$. Esta instalación no es abierta pues se cumple que $d_{f_1 f_2} = 3 \leq 6 = 2t(f_1)$. Por lo tanto, el usuario a es asignado a la instalación f_2 para ser satisfecho.

Las contribuciones de cada usuario son: $\alpha_a = 3$, $\alpha_b = 2$ y $\alpha_c = 2$. El costo de la solución obtenida es de 9, mientras que la suma de las contribuciones es solo 7. Por lo tanto, generalmente no se recupera el costo total, pero con este algoritmo se asegura la recuperación de al menos 1/3 de la inversión realizada.

A.2.2 Análisis del Algoritmo

Mettu y Plaxton (2000) muestran que el costo de la solución que obtiene su algoritmo es a lo más 3 veces el costo óptimo $c^*(J)$. Ellos no consideran a su algoritmo como primal-dual, y su análisis no relaciona el costo de su solución con el dual de la formulación LP del problema. Pál y Tardos (2003) proporcionan una garantía más fuerte: el reparto de costos corresponde a una solución dual factible, y recuperan al menos 1/3 de la solución

construida. El siguiente lema demuestra que el reparto de costos es competitivo (Pál y Tardos, 2003).

Lema 3. *Cada solución al problema de ubicación de instalaciones tiene un costo de al menos* $\sum_{j \in A} \alpha_j$.

Demostración. La suma de los costos calculados para los usuarios no excede el costo de cualquier solución SOL, pues el conjunto S_p de clientes asignados a una instalación p en una solución óptima OPT nunca paga más que el costo de dar servicio a S_p y abrir p, o sea $\sum_{j \in A: j \in S_p} \alpha_j \leq f_p + \sum_{j \in A: j \in S_p} d_{jp}$. Si se considera el conjunto F^* de centros abiertos en OPT, se tiene que:

$$\begin{aligned} \sum_{j \in A} \alpha_j &= \sum_{p \in F^*} \sum_{j \in A: j \in S_p} \alpha_j \\ &\leq \sum_{p \in F^*} f_p + \sum_{p \in F^*} \sum_{j \in A: j \in S_p} d_{jp} \\ &= costo(OPT) \leq costo(SOL) \end{aligned}$$

■

A continuación se relaciona el reparto de costos al costo de la solución de Mettu y Plaxton (2000). Recuérdese que para cada instalación p se define el conjunto S_p de clientes más cercanos que pagan completamente por la instalación (si no tenían otras opciones).

Lema 4. *Si las instalaciones p y q están abiertas, entonces S_p y S_q son disjuntos.*

Demostración. Considérense dos instalaciones abiertas p y q, suponiendo sin pérdida de generalidad que p se abre después de q. Como p no fue cerrado, debe estar lo suficientemente alejado de la instalación q. Ésto es debido a que según el algoritmo

se cumple que: $d_{pq} \geq 2t(p) \geq t(p) + t(q)$. Por lo tanto, S_p y S_q son disjuntos (Pál y Tardos, 2003). ■

Ésto significa que cada usuario contribuye a llenar a lo más una instalación abierta. Por motivos de análisis, se asignan todos los usuarios del conjunto S_p a la instalación p, para cada instalación abierta p. Cada usuario que no pertenece al conjunto S_p de cualquier instalación abierta p se asigna a la instalación abierta más cercana a él.

Se afirma que los usuarios en el conjunto S_p de cada instalación abierta p pagan lo suficiente para cubrir $1/3$ del costo de p así como $1/3$ de sus costos de conexión. Por definición de S_p, $f_p = \sum_{j \in S_p}(t(p) - d_{jp})$. Después de manipulaciones algebraicas, se obtiene que el costo de apertura f_p más el costo de conexión $\sum_{j \in S_p} d_{jp}$ es igual a $|S_p|t(p)$. Para probar la afirmación es suficiente mostrar que el precio correspondiente a cada usuario es al menos $t(p)/3$ (Pál y Tardos, 2003).

Lema 5. *Sea p una instalación abierta, y sea $j \in S_p$ un usuario. Entonces $3\alpha_j \geq t(p)$.*

Demostración. Considérese un usuario $j \in S_p$. Por contradicción, supóngase que $\alpha_j < t(p)/3$. Sea q la primera instalación satisfecha que j alcanza, es decir $d_{jq} \leq \alpha_j$ y $t(q) \leq \alpha_j$.

Si q está abierta, se tiene que $d_{pq} \leq t(p) + \alpha_j < 2t(p)$. Ésto es una contradicción debido al funcionamiento del algoritmo.

Si q no está abierta, existe una instalación q' tal que $d_{qq'} \leq 2t(q) \leq 2\alpha_j$ (véase la Figura 26). Se debe notar que $p \neq q'$, pues $t(q') \leq \alpha_j < t(p)$. Por la desigualdad del

triángulo y por las suposiciones iniciales se tiene que:

$$\begin{aligned} d_{pq'} &\leq d_{pj} + d_{jq} + d_{qq'} \\ &\leq t(p) + \alpha_j + 2t(q) \\ &\leq t(p) + 3\alpha_j \\ &< 2t(p) \end{aligned}$$

Debido al funcionamiento del algoritmo (en cuanto a la condición de apertura de centros), se tiene nuevamente una contradicción, con lo cual se prueba el lema (Pál y Tardos, 2003). ■

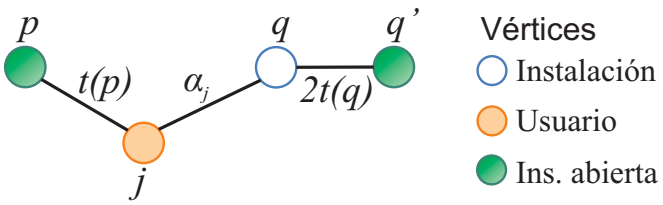


Figura 26. Si $\alpha_j < t(p)/3$, entonces $d_{pq'} < 2t(p)$.

De manera análoga se puede probar que el precio a cobrar a los usuarios que no pertenecen al conjunto S_p para cualquier instalación abierta p es al menos 1/3 de sus costos de conexión (no interesa su contribución a los centros) (Pál y Tardos, 2003).

Lema 6. *Para cada usuario j que no pertenece al conjunto S_p para cualquier instalación abierta p, existe una instalación abierta q tal que $3\alpha_j \geq d_{jq}$.*

Demostración. Sea r una instalación llena a la que contribuyó el usuario j, por lo que se tiene que $\alpha_j \geq t(r)$ y $\alpha_j \geq d_{jr}$. Como r no es abierta en el tiempo $t(r)$ por el algoritmo, debe existir una instalación abierta q tal que $d_{qr} \leq 2t(r)$. Entonces, por la desigualdad del triángulo se cumple que $d_{jq} \leq d_{jr} + d_{rq} \leq \alpha_j + 2t(r) \leq 3\alpha_j$. ■

Debido a los Lemas 5 y 6, se tiene que todos los usuarios contribuyen al menos 1/3 del costo de asignación a las instalaciones abiertas, por lo que se puede establecer el siguiente Lema (Pál y Tardos, 2003):

Lema 7. *El costo de la solución construida por el algoritmo es a lo más* $3\sum_{j\in A}\alpha_j$.

Finalmente, considerando los lemas 3 y 7 se puede enunciar el siguiente teorema (Pál y Tardos, 2003):

Teorema 6. *El algoritmo propuesto por Pál y Tardos es un método de reparto de costos monotónico cruzado, competitivo, y de* 3*-presupuesto equilibrado para el problema de ubicación de instalaciones.*

Apéndice B

Análisis del algoritmo de Swamy-Kumar (2004)

La formulación del problema dual es la siguiente:

$$\max \sum_{j \in \mathcal{D}} \alpha_j \tag{37}$$

$$\text{sujeto a:} \quad \alpha_j \leq c_{ij} + \sum_{S \subseteq V \setminus \{s\} : i \in S} \theta_{S,j} \qquad \forall\, i \in V, i \neq s, j \in \mathcal{D} \tag{38}$$

$$\alpha_j \leq c_{sj} \qquad \forall\, j \in \mathcal{D} \tag{39}$$

$$\sum_{j \in \mathcal{D}} \sum_{S \subseteq V \setminus \{s\} : e \in \delta(S)} \theta_{S,j} \leq M c_e \quad \forall\, e \in E \tag{40}$$

$$\alpha_j, \theta_{S,j} \geq 0 \tag{41}$$

Sean $(\alpha^{(1)}, \theta^{(1)})$, $(\alpha^{(2)}, \theta^{(2)})$ los valores de las variables duales al final de las fases 1 y 2, respectivamente.

Lema 8. *La solución dual* $(\alpha^{(1)}, \theta^{(1)})$ *es factible.*

Demostración. La restricción (38) es satisfecha, pues una vez que j se ajusta con i, α_j y $\sum_{S \subseteq V : i \in S, s \notin S} \theta_{S,j}$ aumentan a la misma razón. La restricción (39) también es satisfecha, pues si j se ajusta con s, entonces la variable dual correspondiente a j se congela debido a que s es una localidad tentativamente abierta.

Considere una arista $e = (u, w)$. Sea $l(j)$ la contribución de j al lado izquierdo de la restricción (40) para esta arista, es decir, $l(j) = \sum_{S : e \in \delta(S), s \notin S} \theta^{(1)}_{S,j}$. Supóngase que

$c_{ju} \leq c_{jw}$. Entonces, j se vuelve ajustada con u antes que con w. Considérese un punto p de la arista (u, w) a una distancia x de u. Si p fue el último punto en esta arista con el cual j se volvió ajustada (antes de que se congele), entonces $l(j) \leq x$. Ésto último puede ser explicado de la siguiente forma: antes que la esfera correspondiente a j alcance a u, j no contribuye para la sumatoria de $l(j)$, pues e (que está fijo) no pertenece a $\delta(S_j)$ ya que la esfera de j no alcanza a u en ese momento. Sean $p_1, p_2, ..., p_k$ los puntos (en el orden en que se ajustan a j) entre u y p a los cuales j se ajusta antes de alcanzar a p. Cuando la esfera de j alcanza a u, ésta comienza a contribuir con $l(j)$ de la siguiente manera: para cada punto p_i entre u y p, la contribución máxima de la variable $\theta_{S,j}$ es la distancia entre p_i y p_{i+1}. Después de ajustarse con p_{i+1}, $\theta_{S,j}$ contribuye como máximo la distancia entre p_{i+1} y p_{i+2}, y así sucesivamente. Entonces, $l(j)$ tiene como valor máximo la distancia entre u y p.

Se define $f(j, x)$ como 1 si j está ajustada con p y j no se congeló en el tiempo en el que se volvió ajustada con p, de otra forma $f(j, x)$ es 0. Entonces, se puede escribir que: $l(j) = \int_0^{c_e} f(j, x)dx$. Intercambiando la sumatoria y la integral en (40), se tiene que:

$$\sum_{j \in \mathcal{D}} \sum_{S \subseteq V : e \in \delta(S), s \notin S} \theta_{S,j}^{(1)} \leq \sum_{j \in \mathcal{D}} \int_0^{c_e} f(j, x)dx = \int_0^{c_e} \sum_{j \in \mathcal{D}} f(j, x)dx$$

Por otro lado, para cada x se cumple que: $\sum_{j \in \mathcal{D}} f(j, x) \leq M$. De otra forma, se tendrían más de M demandas que están ajustadas con un punto tal que ninguna de estas demandas esté congelada, lo cual sería una contradicción. Entonces, $\int_0^{c_e} \sum_{j \in \mathcal{D}} f(j, x)dx$ es a lo más Mc_e, lo cual prueba el lema (Swamy y Kumar, 2004). ■

Lema 9. *Al final de la Fase 1, el costo de asignación de cada demanda j es a lo más $3\alpha_j^{(1)}$.*

Demostración. Claramente se cumple si j está ajustada con una localidad en L'. De

otra forma, sea j una demanda asignada a una localidad i. Sea i' la instalación tentativamente abierta que hizo que j se congele. Se debe cumplir que i e i' son dependientes (por el funcionamiento del algoritmo). Entonces existe una demanda k que está ajustada tanto con i como con i'. Sea $t_{i'}$ el tiempo en que i' fue tentativamente abierta (se define t_i similarmente), por lo que se cumple que $\alpha_j^{(1)} \geq t_{i'}$. Además, como i e i' son dependientes, e i fue abierta, entonces $t_i \leq t_{i'}$.

Por la desigualdad del triángulo: $c_{ij} \leq c_{ik} + c_{ki'} + c_{i'j} \leq 2\alpha_k^{(1)} + \alpha_j^{(1)}$. En el instante de tiempo $t = \alpha_k^{(1)}$, k está ajustada tanto con i como con i'. Supóngase que se vuelve ajustada primero con i (el otro caso es similar). Si i está tentativamente abierta, entonces k se congela y no se ajustará con i'. Por lo tanto, se tiene que $\alpha_k^{(1)}$ se congela luego de que k esté ajustada con i e i'. Se afirma que $\alpha_k^{(1)} \leq t_{i'}$, considerando que a partir de que k se vuelve ajustada con i e i' se pueden dar tres casos:

1. k se congela debido a otra instalación tentativamente abierta antes de que i e i' lo hagan. Este caso resulta una contradicción a la suposición inicial de que i está abierta.

2. k se congela debido a que i se vuelve posiblemente abierta, entonces $\alpha_k^{(1)} = t_i \leq t_{i'}$ y la afirmación se cumple.

3. k se congela debido a que i' se vuelve posiblemente abierta. Debido a que $t_i \leq t_{i'}$, se cumple que en este caso ambas localidades son tentativamente abiertas al mismo tiempo, por lo que $\alpha_k^{(1)} = t_i = t_{i'}$ y la afirmación se cumple.

Entonces, considerando que $\alpha_j^{(1)} \geq t_{i'} \geq \alpha_k^{(1)}$ y que $c_{ij} \leq 2\alpha_k^{(1)} + \alpha_j^{(1)}$, finalmente se tiene que $c_{ij} \leq 3\alpha_j^{(1)}$, por lo que el lema se cumple (Swamy y Kumar, 2004). ■

Lema 10. *Si i es una localidad abierta y $j \in D_i$, $c_{\sigma(j)j} \leq \alpha_j^{(1)}$.*

Demostración. Como $i \in L'$ entonces $\sigma(j) = i$. Por la definición de D_i, j está ajustada con i. Por lo tanto, $\alpha_j^{(1)} \geq c_{ij} = c_{\sigma(j)j}$. ■

Ahora se considera la Fase 2 y el siguiente lema establece una cota superior para el costo del árbol T. Se debe recordar que: $D' = \bigcup_{i \in L'-\{s\}} D_i$.

Lema 11. *costo*$(T) \leq 2 \sum_{j \in D'} \alpha_j^{(2)}$.

Demostración. En cualquier instante de tiempo de la Fase 2, se define la variable $\theta_S = \sum_{j \in \mathcal{D}} \theta_{S,j}$, donde S es el conjunto minimal violado. Se ha observado anteriormente que θ_S crece a una razón de 1. Entonces, la Fase 2 simula el algoritmo primal-dual para el problema del árbol de Steiner enraizado con s como raíz. Por lo tanto, el costo del árbol está acotado por un valor de $2 \sum_S \theta_S$ (Agrawal *et al.*, 1995), donde la suma es sobre todos los subconjuntos de vértices S. De acuerdo a la segunda fase del algoritmo, α_j crece a la misma razón que $\theta_{S,j}$, por lo que $\sum_S \theta_S = \sum_{j \in D'} \alpha_j^{(2)}$ y el lema se cumple. ■

Lema 12. *Considere una demanda j y un centro abierto i. Si $i \neq s$, entonces $\alpha_j^{(2)} \leq c_{\sigma(j)j} + c_{ij} + \sum_{S \subseteq V : e \in \delta(S), s \notin S} \theta_{S,j}^{(2)}$. Además, $\alpha_j^{(2)} \leq c_{\sigma(j)j} + c_{sj}$.*

Demostración. Si $j \notin D'$, $\alpha_j^{(2)} = \theta_{S,j}^{(2)} = 0$ y las desigualdades se cumplen. Entonces se considera una demanda $j \in D'$ y una instalación $i \neq s$. Durante la ejecución de la Fase 2, sea S_t el componente al cual j contribuye en el tiempo t. Considérese el menor tiempo t' para el cual $i \in S_t$. Luego de este tiempo, ambos lados de la desigualdad (38) aumentan a la misma razón, entonces sólo se necesita acotar el aumento de α_j antes del tiempo t' (Swamy y Kumar, 2004).

Sea $l = \sigma(j)$. Debido a que se hace crecer α_j, se debe cumplir que $j \in D_l$ y por lo tanto $c_{jl} \leq \alpha_j^{(1)}$. Se afirma que $t' \leq M c_{li}$. Ésto es cierto pues S_t siempre contiene a l, y en el tiempo $t = M c_{li}$ todas las aristas a lo largo del camino más corto entre l e i

se han vuelto ajustadas. Debido a que α_j aumenta a una razón de a lo más $1/M$, el aumento en α_j en el tiempo t' es a lo más $\frac{Mc_{li}}{M} = c_{li} \leq c_{lj} + c_{ij}$. Esto prueba la primera desigualdad. La segunda se prueba de manera análoga (Swamy y Kumar, 2004). ■

Si se define una solución para el problema dual $(\alpha', \theta^{(2)})$, donde $\alpha' = \max(\alpha_j^{(2)} - c_{\sigma(j)j}, 0)$, se tiene el siguiente lema:

Lema 13. *La solución dual* $(\alpha', \theta^{(2)})$ *es factible.*

Demostración. Por el Lema 12, se tiene que esta solución cumple con las restricciones (38) y (39). Además, los valores de las variables $\theta^{(2)}$ satisfacen la restricción (40). Por ello, la solución $(\alpha', \theta^{(2)})$ es factible. ■

Ahora puede probarse el teorema principal del trabajo presentado por (Swamy y Kumar, 2004). Sea OPT la solución óptima y SOL la solución entregada por el algoritmo.

Teorema 7. *El algoritmo obtiene una solución con un costo de a lo más* $5 \cdot costo(OPT)$.

Demostración. Según la definición de ${\alpha'}_j$, se tiene que $\alpha_j^{(2)} \leq {\alpha'}_j + c_{\sigma(j)j}$. Por el Lema 11, el costo del árbol de Steiner T está acotado por:

$$costo(T) \leq 2 \sum_{j \in \mathcal{D}} {\alpha'}_j + 2 \sum_{j \in D'} c_{\sigma(j)j} \leq 2 \cdot costo(OPT) + 2 \sum_{j \in D'} \alpha_j^{(1)}$$

Por los Lemas 9 y 10, se tiene que el costo de asignación de j es a lo más $\alpha_j^{(1)}$ si $j \in D'$, y a lo más $3\alpha_j^{(1)}$ si $j \notin D'$. Si se denomina A a la asignación de los usuarios a

las localidades abiertas, se tiene que (Swamy y Kumar, 2004):

$$\begin{aligned} costo(SOL) &= costo(T) + costo(A) \\ &\leq 2 \cdot costo(OPT) + 2\sum_{j \in D'} \alpha_j^{(1)} + \sum_{j \in D'} \alpha_j^{(1)} + 3\sum_{j \notin D'} \alpha_j^{(1)} \\ &= 2 \cdot costo(OPT) + 3\sum_{j \in D'} \alpha_j^{(1)} + 3\sum_{j \notin D'} \alpha_j^{(1)} \\ &= 2 \cdot costo(OPT) + 3\sum_{j \in \mathcal{D}} \alpha_j^{(1)} \\ &\leq 5 \cdot costo(OPT) \end{aligned}$$

■

En la Fase 2 se puede usar cualquier algoritmo de ρ_{ST}-aproximación para construir un árbol de Steiner en las localidades abiertas y obtener así un factor de aproximación de $3 + \rho_{ST}$, suponiendo que $\rho_{ST} \leq 2$. Por lo tanto se tiene el siguiente teorema:

Teorema 8. *Existe un algoritmo de* 4.55*-aproximación para el problema de compra alquiler de una sola fuente.*

Demostración. Sean A^* y T^* la asignación y el árbol de Steiner respectivamente de la solución óptima OPT. El árbol en OPT puede aumentarse para contener a las localidades abiertas por el algoritmo. Ésto se puede conseguir conectando cada $l \in L'$ al árbol T^* a través del camino más corto $l - j - i^*(j)$ para $j \in D_l$, donde $i^*(j)$ denota la instalación a la que j es asignada en la solución óptima (véase la Figura 27). Considerando que $|D_l| \geq M$, se tiene que para cada $l \in L'$, el costo de agregar aristas para conectar l a T^* es:

$$M(\text{costo del camino más corto } l - j - i^*(j) \text{ para } j \in D_l) \leq \sum_{j \in D_l} (c_{i^*(j)j} + c_{lj})$$

Extendiendo la sumatoria a todas las localidades $l \in L'$, el costo del árbol construido es a lo más:

$$\begin{aligned} costo(T^*) + \sum_{l \in L'} \sum_{j \in D_l} c_{i^*(j)j} + \sum_{l \in L'} \sum_{j \in D_l} c_{lj} &= costo(T^*) + \sum_{j \in D'} c_{i^*(j)j} + \sum_{j \in D'} c_{\sigma(j)j} \\ &\leq costo(T^*) + \sum_{j \in \mathcal{D}} c_{i^*(j)j} + \sum_{j \in D'} c_{\sigma(j)j} \\ &\leq costo(T^*) + costo(A^*) + \sum_{j \in D'} c_{\sigma(j)j} \end{aligned}$$

Como no se puede encontrar el árbol óptimo eficientemente, se utiliza un algoritmo de ρ_{ST}-aproximación. Por lo cual, utilizando los resultados de los Lemas 10 y 11, el costo de la solución construida por el algoritmo está limitado por (Swamy y Kumar, 2004):

$$\begin{aligned} costo(SOL) &= costo(A) + costo(T) \\ &\leq \sum_{j \in \mathcal{D}} c_{\sigma(j)j} + \rho_{ST}(costo(T^*) + costo(A^*) + \sum_{j \in D'} c_{\sigma(j)j}) \\ &= \sum_{j \notin D'} c_{\sigma(j)j} + \sum_{j \in D'} c_{\sigma(j)j} + \rho_{ST} \sum_{j \in D'} c_{\sigma(j)j} + \rho_{ST}(costo(T^*) + costo(A^*)) \\ &\leq 3 \sum_{j \notin D'} \alpha_j^{(1)} + (1 + \rho_{ST}) \sum_{j \in D'} \alpha_j^{(1)} + \rho_{ST}(costo(T^*) + costo(A^*)) \\ &\leq 3 \sum_{j \notin D'} \alpha_j^{(1)} + 3 \sum_{j \in D'} \alpha_j^{(1)} + \rho_{ST}(costo(T^*) + costo(A^*)) \\ &= 3 \sum_{j \in \mathcal{D}} \alpha_j^{(1)} + \rho_{ST} \cdot costo(OPT) \\ &\leq 3 \cdot costo(OPT) + \rho_{ST} \cdot costo(OPT) = (3 + \rho_{ST}) \cdot costo(OPT) \end{aligned}$$

Usando un algoritmo con $\rho_{ST} = 1.55$ (Robins y Zelikovsky, 2000), el teorema queda demostrado. ■

Se había hecho la suposición de que una localidad podía ser abierta en cualquier punto de una arista, o sea, en ubicaciones que no son vértices. Ésto puede hacer que

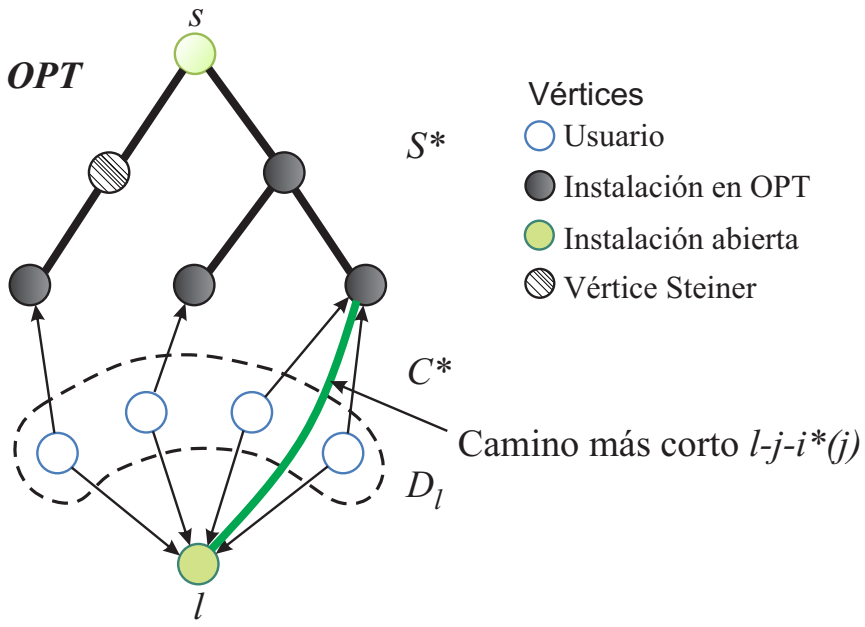


Figura 27. Extendiendo el árbol de Steiner T^* de OPT a las localidades abiertas por el algoritmo.

la solución dada por el algoritmo no sea factible de acuerdo a la formulación hecha para este problema. El siguiente teorema muestra que la modificación de la solución obtenida por el algoritmo, de tal forma a volverla factible, no empeora el resultado de dicha solución.

Teorema 9. *Una solución dada por el algoritmo puede convertirse a otra en la que se tengan sólo vértices como centros, sin aumentar el costo de la primera solución.*

Demostración. Sea $e = (u, w)$ una arista y supóngase que se abren localidades en puntos internos de e. Sea D_e un conjunto de demandas que son asignadas a tales localidades. Sea $D_u \subseteq D_e$ el conjunto de demandas que alcanzan su localidad asignada en e a través de u, es decir: $c_{\sigma(j)j} = c_{uj} + c_{\sigma(j)u}$ para $j \in D_u$, similarmente se define D_w. El árbol de Steiner T debe contener al menos a uno de u o w. Si ambos $u, v \in T$, se asignan clientes en D_u a u y clientes de D_w a w sin aumentar el costo. Supóngase que $u \in T$, $w \notin T$ (el caso inverso es similar). Se asignan todas las demandas en D_u a u. En el caso de que

$|D_w| < M$, se asignan clientes en D_w a u y se remueven aristas en T que están sobre e. En este caso el costo disminuye, pues el costo de la porción de arista removida de T está multiplicada por M. Pero si $|D_w| \geq M$, se reasignan todos los clientes de D_w a w y se agrega la arista e a T. En este caso el costo también disminuye, porque la cantidad de clientes es mayor o igual a M. Se puede notar que T sigue siendo un árbol de Steiner sobre las localidades abiertas, y que el costo total solo puede decrecer. Entonces, se pueden cambiar todas las localidades abiertas a los vértices de G sin aumentar el costo total (Swamy y Kumar, 2004). ■

Apéndice C

Mecanismo de Moulin para el SSROB-CS

En el Capítulo II sólo se consideró la optimización de la solución construida para dar servicio a un conjunto de usuarios. A partir de ahora se considerará la forma de repartir el costo de la solución construida entre los usuarios. Pál y Tardos (2003) proponen un método de reparto de costos monotónico cruzado que recupera al menos el 1/15 del costo de la solución construida, y utilizan el esquema primal-dual para ello.

Las aristas compradas en la red óptima de compra-alquiler deben formar un árbol de Steiner con s como raíz, y las aristas alquiladas deben formar la conexión más corta desde cada usuario j al punto más cercano del árbol. Existen varios algoritmos de aproximación (Karger y Minkoff, 2000; Guha *et al.*, 2000; Swamy y Kumar, 2004) para este problema que explotan la estructura óptima para construir una solución en dos fases: primero se usa un algoritmo parecido al del problema de ubicación de instalaciones para juntar a los usuarios en unos pocos *centros*, y alquilar un camino desde cada usuario al centro más cercano. Cuando se junta suficiente demanda en un centro, se construye un árbol de Steiner, conectando todos los centros a la fuente (Pál y Tardos, 2003).

Puede aplicarse el esquema de reparto de costos para el problema de ubicación de instalaciones, de manera a repartir los costos de alquiler, y usar el método de Jain y Vazirani (2001a) para compartir el costo de las aristas compradas entre los centros. Pero el algoritmo que junta a los usuarios para formar los centros, debe tomar decisiones acerca de elegir en dónde se ubicarán los centros y cómo asignar los usuarios a ellos. Estas decisiones (necesariamente influenciadas por la adición de nuevos usuarios)

pueden tener efectos arbitrarios sobre a qué centro se asigna un usuario en particular y el número de otros usuarios con los que comparte el centro. Esto puede hacer que el reparto de costos no sea monotónico cruzado (Pál y Tardos, 2003).

Pál y Tardos (2003) resuelven este problema mediante un proceso que evita la toma de decisiones arbitrarias. Como en el caso del juego de ubicación de instalaciones, hacen crecer una *esfera* (en alusión al hecho de incrementar gradualmente la variable dual) alrededor de cada usuario. Debido a que no existe un costo de apertura (a diferencia del juego de ubicación de instalaciones), se abre un centro p que ha sido alcanzado por M o más esferas. Un algoritmo primal-dual clásico (Swamy y Kumar, 2004), congelaría a todos los usuarios que han sido conectados al centro, y dejarían de incrementar los radios de las esferas.

Para asegurar la monotonicidad cruzada, se usa el mismo "truco" que en el problema de ubicación de instalaciones, es decir, una esfera fantasma continúa creciendo, incluso luego de que los usuarios son conectados. Se debe notar que debido a que las esferas fantasmas siguen contribuyendo a abrir nuevos centros, se abren más de los necesarios. Esto da lugar a un nuevo problema: cada usuario puede estar conectado a múltiples centros, y el costo de construir un árbol de Steiner para todos los centros puede ser arbitrariamente alto (Pál y Tardos, 2003).

Pál y Tardos proponen tratar el problema de dos formas diferentes. Cuando se calcula el reparto de costos, se pretende construir un árbol de Steiner sobre todos los centros. Se distribuye el costo de cada centro y se trata de cobrarlo equitativamente a todos los usuarios del centro. Debido a que cada usuario puede estar conectado a múltiples centros, el usuario elige a cuál centro contribuir (se supone que desea contribuir al centro que requiera la menor contribución) (Pál y Tardos, 2003). Debido a que la mayoría de los centros no son contribuídos por suficientes usuarios, no se puede

esperar que se recupere una fracción constante del costo del árbol de Steiner en todos los centros. Por ello, solo se elige un subconjunto de centros, y se asigna cada usuario a sólo un centro.

La parte difícil es relacionar ambos procesos: uno que pretende construir el árbol de Steiner sobre todos los centros y calcular el reparto de costos, y el proceso primal-dual estándar que se usa para construir un árbol real sobre un subconjunto de centros; y mostrar cómo los costos repartidos que crecen en el primer proceso pagan por la solución construída por el segundo (Pál y Tardos, 2003).

Se realizan las mismas suposiciones que en (Swamy y Kumar, 2004). Las aristas pueden considerarse como segmentos de longitud igual a su costo, se pueden abrir localidades a lo largo de las aristas y las demandas son unitarias (Pál y Tardos, 2003).

C.1 El proceso fantasma

El proceso fantasma considerado es similar al proceso propuesto por Pál y Tardos (2003) para el problema de ubicación de instalaciones. Por ello, cada usuario j tiene un fantasma, que es una esfera $B(j,t)$ con j como centro y radio igual al tiempo t. La diferencia está en que en vez de que los fantasmas contribuyan para abrir instalaciones, ahora se buscan localidades que son potenciales puntos de congregación de clientes. Buenas candidatas son las localidades que han sido alcanzadas por M o más esferas fantasmas. Sea $\mathcal{C}$ el conjunto (variable en el tiempo) de tales localidades. Nótese que en cualquier tiempo, el conjunto $\mathcal{C}$ puede ser representado por una colección de vértices, de aristas y segmentos de aristas (Pál y Tardos, 2003).

Un *componente conectado* de $\mathcal{C}$ es cualquier conjunto maximal C de $\mathcal{C}$ tal que cualesquiera dos localidades $p_1, p_2 \in C$ están conectadas por un camino que se encuentra

completamente dentro de C (Pál y Tardos, 2003).

Se considera el comportamiento del conjunto $\mathcal{C}$ en el tiempo. En el comienzo, $\mathcal{C}$ está vacío y no existen componentes. A medida que aumenta el tiempo, comienzan a aparecer los componentes. Cada componente empieza como una sola localidad p que ha sido alcanzada por algún conjunto S de clientes, donde $|S| \geq M$. Cuando aumenta el tiempo, el conjunto de localidades alcanzadas por los usuarios de S localmente se ve como una esfera con centro en p que crece uniformemente (nótese que el conjunto $\cap_{j \in S} B(j,t)$ de localidades alcanzadas por todos los usuarios de S puede verse como una unión de varias esferas, así como el mismo conjunto S puede iniciar múltiples componentes de $\mathcal{C}$). A medida que crecen estas esferas, eventualmente dos o más de ellas se tocarán, uniéndose en un solo componente. En general, pares de componentes se conectarán y unirán. Entonces en cualquier tiempo, cada componente C de $\mathcal{C}$ puede ser expresado como una unión de esferas uniformemente crecientes de diferentes radios, donde cada una de ellas comenzó como un componente independiente en una simple localidad p. Nótese que, debido a que las esferas fantasmas crecen indefinidamente, eventualmente el conjunto $\mathcal{C}$ consistirá de un solo componente (Pál y Tardos, 2003).

Existe una correspondencia entre la forma en que crece $\mathcal{C}$ y el algoritmo primal-dual estándar para el árbol de Steiner (Agrawal *et al.*, 1995; Goemans y Williamson, 1995). El algoritmo mantiene una lista de componentes. Inicialmente, cada vértice terminal está en un componente separado. El algoritmo hace crecer uniformemente una esfera alrededor de cada vértice, y cada vez que las esferas de dos vértices se tocan, se construye la arista que une ambos vértices, uniendo los dos componentes. La única diferencia es que mientras que en el algoritmo estándar se empieza con un conjunto fijo de componentes, en el proceso fantasma los componentes pueden ser creados en puntos arbitrarios en el tiempo. Se hará uso de esta conexión para demostrar más adelante

que los precios determinados pagan una fracción constante del costo de la solución construida (Pál y Tardos, 2003).

C.2 Reparto de costos

Para especificar cómo los precios son calculados, se necesitan ciertas definiciones. Se dice que un usuario j está *conectado* a un componente C de $\mathcal{C}$ en un tiempo t, si $B(j,t) \cap C \neq \emptyset$. El usuario está *satisfecho* si está conectado a un componente que contiene a la fuente s. Para cada usuario j, sea $t(j)$ el tiempo en que el primer usuario se vuelve conectado a un componente y sea $t'(j)$ el tiempo en que j es satisfecho. Se debe hacer que cada usuario pague por los costos de alquiler asociados a él. Ésto sugiere que se cobre a cada usuario una cantidad proporcional al tiempo en que está desconectado. Además, cada usuario conectado a un componente C debe contribuir una fracción equitativa del costo del crecimiento de dicho componente. Si un usuario está conectado a múltiples componentes, se permitirá que contribuya al componente donde su contribución es menor (Pál y Tardos, 2003).

Para cada componente C de $\mathcal{C}$, sea $J(C)$ el conjunto de usuarios conectados a C. Para un usuario conectado j, sea $a_j(t)$ el tamaño máximo $J(C)$ entre todos los componentes C a los que j está conectado en el tiempo t. Se define una función f_j para cada usuario j de la siguiente manera: $f_j(t) = 1$ para $0 \leq t \leq t(j)$, $f_j(t) = M/a_j(t)$ para $t(j) \leq t \leq t'(j)$, y $f_j(t) = 0$ para $t > t'(j)$. Se definen dos versiones del reparto de costos α_j y α'_j para cada usuario j. El reparto de costos final, determinado más adelante, es una suma ponderada de estos valores definidos, y equilibrados para optimizar la fracción de costo recuperado. Los valores α_j y α'_j son definidos de la siguiente manera

(Pál y Tardos, 2003):

$$\alpha_j = \int_0^{t'(j)} f_j(t)dt \qquad \text{y} \qquad \alpha'_j = \int_0^{t(j)} f_j(t)dt = t(j)$$

Se puede determinar entonces la primera de las tres propiedades deseables para el método de reparto de costos:

Teorema 10. *Los costos compartidos α_j y α'_j son funciones monotónicas cruzadas en el conjunto $\mathcal{D}$.*

Demostración. Al añadir más usuarios, sólo se puede hacer que el conjunto $\mathcal{C}$ sea más grande, y entonces cada usuario se conecta antes. Además, cada componente de $\mathcal{C}$ solo puede incrementarse al añadir más usuarios, así como se incrementa el número de usuarios conectados a ese componente, entonces los costos compartidos crecen más lentamente (Pál y Tardos, 2003). ■

El algoritmo propuesto por Pál y Tardos para obtener los tiempos $t(j)$ y $t'(j)$ para cada $j \in \mathcal{D}$ se muestra en el Algoritmo 12. Sea $B(C,t)$ el radio del componente C en el tiempo t. Luego de obtener estos tiempos, se calculan los valores de α_j y α'_j según las fórmulas vistas anteriormente.

C.3 Construyendo una solución

El objetivo ahora es demostrar que los costos compartidos α y α' pueden recuperar una fracción constante del costo de la red que da servicio a los usuarios de $\mathcal{D}$. Recuérdese que cada nuevo componente de $\mathcal{C}$ empieza como un simple punto p, llamado *centro*. Para una localidad p, sea $t(p)$ el tiempo en que p aparece en el conjunto $\mathcal{C}$. Nótese que $\mathcal{C}$ es sólo la unión de las esferas $B(p, t - t(p))$, con radio $t - t(p)$ alrededor de cada centro p (Pál y Tardos, 2003).

Estos centros son usados como puntos de congregación. Aunque, de manera a poder pagar el costo del árbol de Steiner en los centros, se debe seleccionar solo un subconjunto de centros para abrirlos, y asignar a cada usuario para que contribuya con a lo más un centro abierto. Ésto se hace de manera similar a lo hecho en el juego de ubicación de instalaciones. Para cada centro p, se decide su apertura en el momento en que es creado. Se abre p si en el tiempo $t(p)$ no existe un centro abierto dentro de un radio de $2t(p)$ de p. Como en el caso del juego de ubicación de instalaciones, ésto asegura que ningún jugador j puede estar en el conjunto $S_p = \{j \in \mathcal{D} \mid c_{jp} \leq t(p)\}$ de usuarios que contribuyen a la apertura de p para dos centros abiertos distintos. El siguiente resultado se obtuvo en el juego de ubicación de instalaciones (Pál y Tardos, 2003).

Lema 14. *Sea p un centro abierto, y sea $j \in S_p$. Entonces $3{\alpha'}_j \geq t(p)$. Para un usuario j que no pertenece a ningún S_p, existe un centro fantasma abierto p tal que $c_{jp} \leq 3{\alpha'}_j$.*

La red que se construye es entonces la siguiente: se compra un árbol de Steiner que conecta centros abiertos a la fuente s, y se alquila un camino desde cada usuario al centro abierto más cercano. El Lema 14 muestra que los costos compartidos α' pueden pagar 1/3 de los costos alquilados de la red (Pál y Tardos, 2003). La siguiente sección muestra cómo los valores de α pueden pagar una fracción constante del árbol que se construye.

Algoritmo 12 Cálculo de los tiempos de conexión y satisfacción de los usuarios para el método de reparto de costos propuesto por Pál y Tardos para el SSROB-CS.

Entrada: Grafo $G = (V, E)$, demandas $\mathcal{D} \subseteq V$, fuente s, parámetro $M > 1$.
Salida: Tiempos $t(j)$ y $t'(j)$ para cada $j \in \mathcal{D}$.

1: $\mathcal{C} \leftarrow \emptyset$, $t \leftarrow 0$
2: $t(j) \leftarrow 0$, $t'(j) \leftarrow 0$ para todo $j \in \mathcal{D}$
3: **mientras** $\exists j \in \mathcal{D}$ no conectado, o $\mathcal{C}$ no tiene un único componente conectado a la raíz **hacer**
4: Sea $S_p = \{j \in \mathcal{D} | t \geq c_{j,p}\}$ para alguna instalación p.
5: Se aumenta t (También $B(j,t)$ $\forall j \in \mathcal{D}$, y $B(C,t)$ $\forall C \in \mathcal{C}$; a la misma razón de t) hasta que:
6: (a) $|S_p| \geq M$, ó
7: (b) $B(C,t) \cap B(C',t) = \emptyset$
8: **si** (a) **entonces**
9: $j \in S_p$ queda conectado a p.
10: **si** $t(j) = 0$ **entonces**
11: $t(j) \leftarrow t$; $\mathcal{C} \leftarrow \mathcal{C} \cup \{p\}$; $B(\{p\}, t) \leftarrow 0$;
12: **fin si**
13: **fin si**
14: **si** (b) **entonces**
15: Sea P el conjunto de aristas que forman el camino más corto entre C y C'.
16: Se unen C y C' a través de P, dando lugar a un único componente C'' en $\mathcal{C}$.
17: **si** $s \in C''$ **entonces**
18: **para todo** $j \in \mathcal{D}$ conectado a una instalación $p \in C''$ **hacer**
19: **si** $t'(j) = 0$ **entonces**
20: $t'(j) \leftarrow t$
21: **fin si**
22: **fin para**
23: **fin si**
24: **fin si**
25: **fin mientras**
26: **devolver** $t(j)$ y $t'(j)$ para cada $j \in \mathcal{D}$

C.4 Análisis del algoritmo

Para motivos de análisis, supóngase que el algoritmo primal-dual de Agrawal *et al.* (1995) es usado para construir el árbol de Steiner en los centros abiertos. El algoritmo empieza con cada centro en un componente separado, y hace crecer uniformemente una

esfera alrededor de cada centro. Cada vez que dos esferas se tocan, se verifica si las mismas están en el mismo componente. En el caso que no estén, se compra el camino más corto entre los centros de las esferas, fusionando ambos componentes en uno solo. Además, el centro p de la primera esfera que alcanza la fuente s puede comprar el camino más corto entre p y s (Pál y Tardos, 2003).

Se puede considerar que el algoritmo incurre en un costo de M por unidad de tiempo por hacer crecer cada componente que no contiene al vértice fuente s (hacer crecer el componente que contiene a s es gratis). Ésto es, si $a(t)$ denota el número de componentes (que no contienen a la fuente) en el tiempo t, el costo total de hacer crecer los componentes es $M \int_0^\infty a(t)dt$, donde la integral es sobre toda la ejecución del algoritmo. Por argumentos estándares, el costo del árbol construido es a lo más dos veces el costo de crecimiento (Pál y Tardos, 2003).

Para un centro abierto p, sea $S_p(t)$ el conjunto de usuarios que están a una distancia t de p. Nótese que $S_p = S_p(t(p))$. Para cada componente C' se define un conjunto (variable en el tiempo) $Contrib(C')$ de usuarios que serán responsables de mantener un flujo estable de contribución de al menos $M/3$ por unidad de tiempo de los incrementos de tiempo $f_j(t)$ de los repartos de costos. Formalmente, se tiene que:

$$Contrib(C') = \bigcup_{p \in C'} S_p(max\{t, t(p)\})$$

Los conjuntos contribuyentes son disjuntos, como se demuestra en el siguiente lema. Ésto es, ningún usuario contribuye a más de un componente en cualquier instante de tiempo.

Lema 15. *En cualquier tiempo t, los conjuntos contribuyentes de dos componentes distintos C'_1 y C'_2 son disjuntos.*

Demostración. Supóngase que $Contrib(C'_1) \cap Contrib(C'_2) \neq \emptyset$. Entonces existen centros $p \in C'_1$, $q \in C'_2$ tal que $S_p(max\{t, t(p)\}) \cap S_q(max\{t, t(q)\}) \neq \emptyset$. Sin pérdida de generalidad, se supone que $t(p) \geq t(q)$. Si $t \geq t(p) \geq t(q)$, por la desigualdad del triángulo se tiene que $c_{pq} \leq 2t$ y así p y q deben pertenecer al mismo componente, por lo que se tiene una contradicción. Si $t \leq t(p)$, entonces $c_{pq} \leq t + t(p) \leq 2t(p)$, lo cual significa que no se hubiese abierto p, teniéndose una contradicción (Pál y Tardos, 2003). ■

Considere un usuario j que pertenece a un conjunto contribuyente de un componente de Steiner C'. En el proceso fantasma, este usuario puede estar conectado a varios componentes de $\mathcal{C}$, y algunos de estos componentes pueden tener un número mayor de usuarios conectados a ellos. Entonces, aunque el componente C' sea "pequeño" (es decir, no tiene muchos contribuyentes), puede suceder que la mayoría de sus contribuyentes tienen una conexión a un componente "grande" de $\mathcal{C}$, por lo que sus contribuciones $f_j(t)$ no son suficientes para dar soporte al crecimieno del componente C' en el tiempo t. Pero lo que se demostrará en el Lema 16 es que luego, en un tiempo $3t$, el componente C' habrá crecido lo suficiente para cubrir el área que los componentes grandes originales ocupaban en el tiempo t y conectar a todos los usuarios que estaban conectados a componentes grandes en el tiempo t. Estos usuarios pueden pagar juntos lo suficiente para el crecimiento de C' en el tiempo $3t$ con sus razones de crecimiento $f_j(t)$ en el tiempo t. Antes de demostrar el Lema 16, se enunciarán ciertas afirmaciones correspondientes a lo demostrado para el juego de ubicación de instalaciones (Pál y Tardos, 2003).

Nota 1. *Para una localidad $p \in \mathcal{C}$, existe un centro abierto q (posiblemente $q = p$) tal que $c_{pq} \leq 2t(p)$.*

Nota 2. *Si los centros p y q están abiertos, entonces S_p y S_q son disjuntos.*

Lema 16. *Supóngase que en el tiempo t, los usuarios i y j están conectados al mismo componente C del conjunto $\mathcal{C}$. Entonces en el tiempo $3t$, i y j pertenecen al conjunto contribuyente del mismo componente de Steiner C'.*

Demostración. Sean i y j usuarios conectados al componente C de $\mathcal{C}$ en el tiempo t. Nótese que C es una unión de esferas alrededor de centros (cerrados o abiertos). El usuario i está conectado a C, entonces debe existir un centro q con una esfera $B(q, t - t(q))$ que intersecta a la esfera fantasma de i. Similarmente existe un centro q' y una esfera $B(q', t - t(q'))$ que intersecta a la esfera fantasma de j. Como los centros q y q' pertenecen al mismo componente, debe existir una secuencia de centros $q = q_1, q_2, ..., q_k = q'$, con esferas $B_l = B(q_l, t - t(q_l))$ tales que las esferas de dos centros consecutivos se intersectan.

Por la Nota 2, existe una secuencia de (no necesariamente de elementos todos distintos) centros abiertos $p_1, p_2, ..., p_k$, tales que para cada l, $c_{p_l,q_l} \leq 2t(q_l)$. Véase la Figura 28 como ejemplo. Es fácil verificar que la distancia $c_{ip_1} \leq 2t(q_1) + t \leq 3t$, y análogamente $c_{p_kj} \leq 3t$. La distancia $c_{p_lp_{l+1}}$ puede ser acotada por $4t$ de la siguiente manera:

$$c_{p_lp_{l+1}} \leq c_{p_lq_l} + c_{q_lq_{l+1}} + c_{q_{l+1}p_{l+1}} \leq 2t(q_l) + (t - t(q_l)) + (t - t(q_{l+1})) + 2t(q_{l+1}) \leq 4t$$

Entonces cada par de esferas consecutivas $B(p_l, 3t)$ deben intersectarse, probando que p_1 y p_k están en el mismo componente de Steiner en el tiempo $3t$. Por definición de *Contrib*, tanto i como j deben estar en el mismo conjunto contribuyente de este componente de Steiner (Pál y Tardos, 2003). ■

Ahora se puede probar el siguiente lema:

Lema 17. *Sea j un usuario tal que en el tiempo t, $j \in Contrib(C')$ para algún componente de Steiner C'. Entonces se tiene que $f_j(t/3) \geq M/|Contrib(C')|$.*

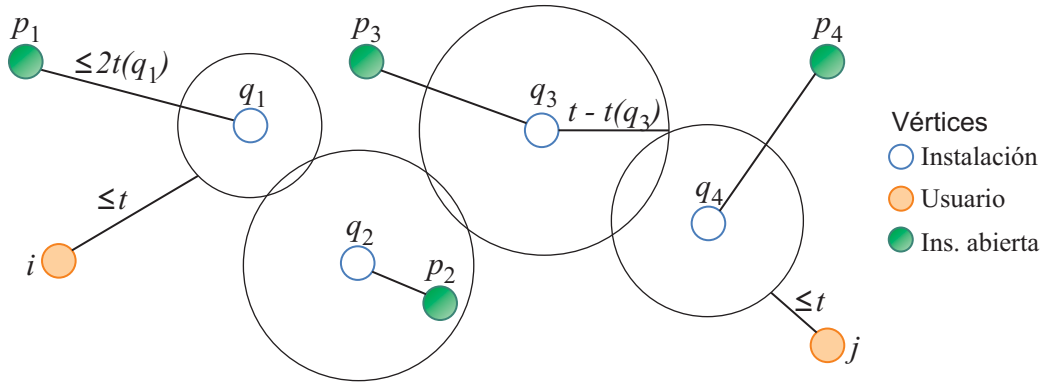


Figura 28. Clientes i y j conectados a un componente C en el tiempo t.

Demostración. Nótese que debido a que $|S_p| \geq M$ para cualquier centro abierto p, $|Contrib(C')|$ es siempre al menos M, y entonces $M/|Contrib(C')| \leq 1$. Si j no está conectada en el tiempo $t/3$, entonces $f_j(t/3) = 1$ y la desigualdad se cumple.

Si j está conectada en el tiempo $t/3$, $f_j(t/3) = M/|J(C)|$ para algún componente C de $\mathcal{C}$. Por el Lema 16, $Contrib(C')$ en el tiempo t contiene todos los usuarios que contribuían con C en el tiempo $t/3$ (Pál y Tardos, 2003). ■

Usando el Lema 17, se puede acotar el costo del árbol construido.

Lema 18. *El costo del árbol construido es a lo más* $6\sum_j \alpha_j$.

Demostración. En cualquier tiempo t, los usuarios pueden obtener un rendimiento de $Ma(t)$ de sus contribuciones en el tiempo $t/3$. Se afirma que $Ma(t) \leq \sum_{j\in\mathcal{D}} f_j(t/3)$. Para comprobarlo, sea k el número de componentes de Steiner en un instante t. Los

componentes son enumerados como $C_1, C_2, ..., C_k$. Por los Lemas 15 y 17 se tiene que:

$$\begin{aligned}\sum_{j\in\mathcal{D}} f_j(t/3) &= \sum_{j\in C_1} f_j(t/3) + ... + \sum_{j\in C_k} f_j(t/3) \\ &\geq \sum_{j\in C_1} \frac{M}{|Contrib(C_1)|} + ... + \sum_{j\in C_k} \frac{M}{|Contrib(C_k)|} \\ &= \frac{M|Contrib(C_1)|}{|Contrib(C_1)|} + \frac{M|Contrib(C_k)|}{|Contrib(C_k)|} \\ &= M + ... + M \\ &= Mk = Ma(t)\end{aligned}$$

Por lo tanto:

$$\int_0^\infty Ma(t) \leq \int_0^\infty \sum_{j\in\mathcal{D}} f_j(t/3)dt = \sum_{j\in\mathcal{D}} \int_0^\infty f_j(t/3)dt = \sum_{j\in\mathcal{D}} 3\alpha_j = 3\sum_{j\in\mathcal{D}} \alpha_j$$

Debido a que $M\int_0^\infty a(t)$ puede pagar al menos la mitad del árbol construido, los costos de compra de la solución no exceden $6\sum_{j\in\mathcal{D}} \alpha_j$ (Pál y Tardos, 2003). ■

De los Lemas 14 y 18 se puede enunciar el siguiente teorema, que representa una cota superior para el costo de la solución construida (Pál y Tardos, 2003):

Teorema 11. *El costo de la solución construida es a lo más* $\sum_{j\in\mathcal{D}}(3\alpha'_j + 6\alpha_j)$.

Ahora se necesita una cota inferior para el costo de la solución construida, por lo que se demuestra el siguiente teorema:

Teorema 12. *Cada solución factible para un caso del* SSROB *tiene un costo de al menos*

$max(\frac{1}{2}\sum_{j\in\mathcal{D}} \alpha_j, \sum_{j\in\mathcal{D}} \alpha'_j)$.

Demostración. Considérese una solución arbitraria SOL para el caso del SSROB. Se demostrará que su costo debe ser mayor o igual a $\sum_{j\in\mathcal{D}} \alpha'_j$ y $\frac{1}{2}\sum_{j\in\mathcal{D}} \alpha_j$.

Primero se considera la cota de $\sum_{j\in\mathcal{D}} \alpha'_j$. Se considera una esfera abierta de radio α'_j alrededor de cada usuario j. Nótese que por construcción, ninguna localidad es cubierta por más de M esferas y la fuente s queda afuera de todas las esferas. La solución SOL debe proveer un camino p_j desde cada usuario j a la fuente s. Cada camino p_j debe ir desde el centro de la esfera j hasta su exterior, entonces debe viajar al menos una distancia α'_j dentro de la esfera. Se cobrará a cada camino p_j la distancia que viaja dentro de la esfera correspondiente a j. Se afirma que la suma de los cobros a todos los caminos es menor que el costo de SOL. Ésto es debido a que cada segmento de arista alquilada fue cobrado a lo más una vez, mientras que cada segmento comprado fue cobrado a lo más M veces. Teniendo en cuenta que la suma de los radios de las esferas es menor o igual a la suma de los cobros, se tiene que $\sum_{j\in\mathcal{D}} \alpha'_j \leq costo(SOL)$ (Pál y Tardos, 2003).

Ahora se considera la cota de $\frac{1}{2}\sum_{j\in\mathcal{D}} \alpha_j$. Supóngase que se tienen distancias $d_{j,e}$ de aristas para cada usuario j con las siguientes propiedades:

1. Para cada arista e y cada usuario j, $d_{j,e} \leq c_e$.
2. Para cada arista e, $\sum_{j\in\mathcal{D}} d_{j,e} \leq 2Mc_e$.
3. El camino más corto desde cada usuario j a la fuente s en la métrica d_j tiene una longitud de al menos α_j.

Nuevamente, considérese una solución arbitraria SOL. Debido a que cada usuario debe tener un camino a s por la propiedad (3), se tiene:

$$\alpha_j \leq \sum_{e \text{ comprada}} d_{j,e} + \sum_{e \text{ alquilada para j}} d_{j,e}$$

Sumando estas desigualdades para todos los usuarios j de $\mathcal{D}$, se obtiene:

$$\sum_{j\in\mathcal{D}} \alpha_j \leq \sum_{e \text{ comprada}} \sum_{j\in\mathcal{D}} d_{j,e} + \sum_{j\in\mathcal{D}} \sum_{e \text{ alquilada para j}} d_{j,e}$$

El costo de la solución arbitraria puede escribirse como:

$$cost(SOL) = \sum_{e \text{ comprada}} Mc_e + \sum_{j\in\mathcal{D}} \sum_{e \text{ alquilada para j}} c_e$$

De las propiedades (1) y (2) se tiene que:

$$\sum_{e \text{ comprada}} \sum_{j\in\mathcal{D}} d_{j,e} \leq 2 \sum_{e \text{ comprada}} Mc_e \quad \text{y} \quad \sum_{j\in\mathcal{D}} \sum_{e \text{ alquilada para j}} d_{j,e} \leq \sum_{j\in\mathcal{D}} \sum_{e \text{ alquilada para j}} c_e$$

Combinando estas desigualdades se tiene que:

$$\sum_{j\in\mathcal{D}} \alpha_j \leq 2 \cdot costo(SOL)$$

Ahora queda demostrar que tal $d_{j,e}$ existe. Para un usuario j, sea $S_j(t)$ el conjunto de localidades *alcanzables* por j. Una localidad p es alcanzable por j, si está dentro de la esfera fantasma de j (es decir, $c_{jp} \leq t$), o si está en un componente al que j está conectado. Una arista es *cruzada* por el conjunto S_j, si tiene un extremo dentro de S_j y otro fuera. Se comienza con todos los $d_{j,e} = 0$. En cualquier tiempo t, se incrementa la longitud $d_{j,e}$ de cada arista e que es cruzada por S_j a razón de $f_j(t)$ (Pál y Tardos, 2003).

Debido a que en cada instante de tiempo se incrementan las longitudes de todas las aristas en algún corte $j - s$ a la razón $f_j(t)$, la longitud del camino más corto $j - s$ en la métrica d_j será igual a $\int_0^\infty f_j(t)dt = \alpha_j$. Con ésto se prueba la propiedad (3). La vecindad de cada S_j se mueve a una velocidad mayor o igual a 1 (puede saltar cuando los componentes se fusionan), entonces ninguna arista puede acumular un $d_{j,e}$ que es mayor que la longitud original c_e. Con ésto se prueba la propiedad (1).

Finalmente, considérese la propiedad (2). Cada arista e puede ser cruzada por dos tipos de vecindades, correspondientes a las esferas fantasmas de usuarios y las esferas fantasmas de componentes. Cada localidad en e puede ser cruzada por a lo más M esferas fantasmas de usuarios, contribuyendo a lo más una unidad de longitud por unidad de tiempo, y a lo más una esfera fantasma de componente, contribuyendo M unidades de longitud por unidad de tiempo. Entonces, $\sum_{j\in\mathcal{D}} d_{j,e} \leq Mc_e + Mc_e = 2Mc_e$ (Pál y Tardos, 2003). ■

Combinando los Teoremas 10, 11 y 12; finalmente se tiene lo siguiente:

Teorema 13. *La función de reparto de costos* $\xi(\mathcal{D}, j) = 1/5\alpha'_j + 2/5\alpha_j$ *es monotónica cruzada, competitiva y recupera al menos una fracción de* $1/15$ *del costo de la solución construida.*

Demostración. Sea SOL la solución construida y OPT la óptima para un caso dado. El teorema indica que:

$$\frac{1}{15} costo(SOL) \leq \sum_{j\in\mathcal{D}} \left(\frac{1}{5}\alpha'_j + \frac{2}{5}\alpha_j \right) \leq costo(OPT)$$

La primera desigualdad es válida, ya que multiplicando ambos miembros por 15 se tiene que:

$$costo(SOL) \leq \sum_{j\in\mathcal{D}} \left(\frac{15}{5}\alpha'_j + \frac{30}{5}\alpha_j \right) = \sum_{j\in\mathcal{D}} (3\alpha'_j + 6\alpha_j)$$

lo cual es válido según el Teorema 11.

Para la segunda desigualdad, se debe probar que:

$$\max\left(\frac{1}{2}\sum_{j\in\mathcal{D}} \alpha_j, \sum_{j\in\mathcal{D}} \alpha'_j \right) \geq \sum_{j\in\mathcal{D}} \left(\frac{15}{5}\alpha'_j + \frac{30}{5}\alpha_j \right)$$

Por lo tanto, se tienen dos casos:

(i) Si $\max(\frac{1}{2}\sum_{j\in\mathcal{D}}\alpha_j, \sum_{j\in\mathcal{D}}\alpha'_j) = \frac{1}{2}\sum_{j\in\mathcal{D}}\alpha_j$, entonces:

$$\frac{1}{2}\sum_{j\in\mathcal{D}}\alpha_j = \frac{4}{10}\sum_{j\in\mathcal{D}}\alpha_j + \frac{1}{10}\sum_{j\in\mathcal{D}}\alpha_j \geq \frac{4}{10}\sum_{j\in\mathcal{D}}\alpha_j + \frac{2}{10}\sum_{j\in\mathcal{D}}\alpha'_j = \frac{2}{5}\sum_{j\in\mathcal{D}}\alpha_j + \frac{1}{5}\sum_{j\in\mathcal{D}}\alpha'_j$$

La última desigualdad es válida pues en este caso: $\frac{1}{2}\sum_{j\in\mathcal{D}}\alpha_j \geq \sum_{j\in\mathcal{D}}\alpha'_j$.

(i) Si $\max(\frac{1}{2}\sum_{j\in\mathcal{D}}\alpha_j, \sum_{j\in\mathcal{D}}\alpha'_j) = \sum_{j\in\mathcal{D}}\alpha'_j$, entonces:

$$\sum_{j\in\mathcal{D}}\alpha'_j = \frac{4}{5}\sum_{j\in\mathcal{D}}\alpha'_j + \frac{1}{5}\sum_{j\in\mathcal{D}}\alpha'_j \geq \frac{2}{5}\sum_{j\in\mathcal{D}}\alpha_j + \frac{1}{5}\sum_{j\in\mathcal{D}}\alpha'_j$$

La última desigualdad es válida pues en este caso: $\frac{1}{2}\sum_{j\in\mathcal{D}}\alpha_j \leq \sum_{j\in\mathcal{D}}\alpha'_j$.

Por lo tanto, por el Teorema 12, finalmente se tiene que:

$$costo(OPT) \geq \sum_{j\in\mathcal{D}}\left(\frac{15}{5}\alpha'_j + \frac{30}{5}\alpha_j\right)$$

con lo cual queda demostrado el teorema. ■