Tesis de Licenciatura en Ciencias Matemáticas

Programación Matemática y Subastas por Combinaciones

Alexis Jawtuschenko
alexis.jawtuschenko@gmail.com
LU: 062/04
Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Matemática
Director: Dr. Guillermo A. Durán
Co-Director: Dr. Javier Marenco

Diciembre de 2010

Agradecimientos

Quiero agradecer a mi director, el Dr. Guillermo Durán, por haberme permitido integrar tan interesante proyecto, y por haber dirigido mi tesis, consecuencia de tal trabajo. Por sus indicaciones y ayuda que hicieron posible esta tesis. Es un excelente director y ha sido muy grato trabajar con él.

A mi co-director, el Dr. Javier Marenco le agradezco toda su ayuda y contención y todo su tiempo, en el que me proporcionó valiosos comentarios que mejoraron esta tesis.

A la Dra. Flavia Bonomo por su apoyo y colaboración. A ellos tres les agradezco permitirme formar parte de su grupo.

Agradezco especialmente también a los profesores Dr. Gabriel Acosta y Dr. Juan Pablo Pinasco por evaluar e interesarse en esta tesis.

Quiero agradecer a mi amigo Luciano Perez por las invaluables charlas de Microeconomía y Teoría de Juegos que me ayudaron a entender de qué estaba hablando. Por darme ánimo para el negocio aún cuando el panorama se vea *borroso*, y por proponer y sostener la idea de la consideración de lo no homogéneo aún cuando el contexto y lo inmediato luzcan *muy homogéneos*. A él también agradezco el haber compartido interminables horas de estudio, que se hicieron más amenas con ayuda del humor.

Agradezco a Francisco Cayol, a quien prácticamente podría considerar un amigo, por sus incansables elogios y por fomentar y aportar a mi desarrollo profesional.

A mi amigo Teo, por su amistad y por acompañar en momentos difíciles y a él junto con Verónica y Yanina, por los largos fines de semana de estudio que con su compañía se hicieron muy agradables y devinieron en mucho aprendizaje, inclusive luego de soportar largos viajes en medios de transporte público. Agradezco al resto de mis amigos de la facultad, que afortunadamente son muchos.

A Victoria. A Dieguito "Diegoogle" Gawinowich por ayudarme a ser mejor programador. A Theo, Joaquín, Eugenia y Gala. A Mamá, Ignacio y Nicolás. A mi abuela Vala. A Sabrina, por todo.

A mi abuela Tin quien nunca perdió la visión, de las cosas importantes.
Por todas aquellas bonitas tardes que no se irán de mis recuerdos, que incluían anécdotas pero sobre todo sistemáticos reconocimientos, elogios, apoyo y congratulación.

A papá.

Resumen

En esta tesis se estudian el problema de la determinación de los ganadores y el problema de la asignación óptima, en términos de la maximización del ingreso, en una subasta Multiunitaria de Procuración Pública. En primer lugar hacemos un resumen de la teoría microeconómica elemental de subastas y de subastas por Combinaciones en tanto mecanismos. Aquí incluimos una definición de las clases más frecuentes de subastas y citamos los trabajos más importantes en el tema, y hacemos foco en la intrínseca interdisciplinariedad de este tema, en el que intervienen la Microeconomía, la Computación Científica, la Programación Matemática y la Investigación Operativa.

Citamos dos ejemplos de licitaciones de procuración, uno de los cuales destacamos por tratarse de un mecanismo implementado en otro país emergente que resultó en una mejora substancial del bien procurado con un aumento de precio sensiblemente menor que el pronosticado en principio.

En el contexto de la Computación Científica, redactamos las generalidades de la teoría de Complejidad Computacional y establecemos la terminología, definiciones y propiedades necesarias para luego poder discutir los dos problemas de Optimización Combinatoria asociados a una Subasta por Combinaciones, el Problema de la Determinación del Ganador y el problema de la Optimización del Ingreso. Estos problemas son formalmente definidos para después demostrar que pertenecen a una clase de problemas computacionales muy duros de tratar en el caso general.

Respecto de la aplicación que nos ocupa al final del trabajo, nos proponemos modelizar y resolver una licitación pública en la que el Gobierno de la Ciudad de Buenos Aires se procura conexiones de acceso a la red *Internet* por parte de ciertos proveedores de este servicio que operan actualmente en la Ciudad. Proponemos un modelo y un diseño originales que explotan ciertas características distintivas de la instancia que trabajamos. Para la resolución implementamos una instancia de Programación Lineal Entera (ILP, en inglés, *Integer Linear Programming*) que obtiene la solución óptima en fracciones de segundos. El modelo diseñado fue utilizado por el Gobierno de la Ciudad en la licitación realizada a fines de 2008.

Abstract

In this graduation thesis we study the winner determination problem and the optimal assignment problem, in terms of the maximization of the revenue, in a Multi-unit Public Procurement Auction. First we make a survey of the elements of the economic theory of auctions and combinational auctions as mechanisms. Here we include a definition of the canonical classes of auctions and we cite the most important works in this area. We focus in the condition of this topic of being inherently interdisciplinary, in which intervene Economics, Computer Science, Mathematical Programming and Operations Research.

We survey two examples of procurement auctions, one of which is an example of a mechanism implemented in an emergent country that resulted in a substantial improvement of the goods procured with an increase of the price which is very lower than those suggested by the price trends in the first place.

In the context of Computer Science, we develop the generalities of the Computational Complexity theory and we state the terminology, definitions and properties needed to the discussion of the two Combinatorial Optimization problems involved in a Combinational Auction, the Winner Determination Problem and the Revenue Optimization problem. These problems are formally defined and later demonstrated to belong to a class of problems that are very hard to treat in the general case.

Finally, we consider a real-world application, where we intend to model and solve a public auction in which the Government of the City of Buenos Aires seeks a procuration of Internet connections from certain private providers. We propose an original model and an original design that exploit certain distinctive characteristics of the instance we've worked in. For the resolution we developed an Integer Linear Programm finding the solution within a second. The designed model was used by the Government of the City of Buenos Aires in the tender carried out in 2008.

Índice general

1. Subastas - Subastas por Combinaciones **3**
1.1. Definiciones y notaciones básicas. Hipótesis 12
1.1.1. Posturas y Eficiencia 14
1.1.2. Complejidad y Consideraciones Algorítmicas 15
1.1.3. El Problema de la Procuración 16
1.2. Resumen de la tesis 16

2. Aplicaciones exitosas **18**
2.1. Subasta de rutas de Omnibus de Londres 18
2.2. Una subasta por combinaciones para licitar contratos de servicios gastronómicos para las escuelas en Chile 19

3. Dos Problemas Computacionales en Subastas **22**
3.1. Complejidad computacional 22
3.1.1. Tamaños de instancias y entradas de algoritmos . . . 22
3.1.2. Función de complejidad de un algoritmo 23
3.1.3. Clasificación de algoritmos 24
3.1.4. Complejidad de un problema 25
3.2. Problemas NP-Completos 26
3.3. El problema de la Optimización del Ingreso 30
3.3.1. Formulación del problema 30
3.4. El problema de la Determinación del Ganador 33
3.4.1. Formulación del problema 33
3.4.2. Observaciones ulteriores 35

4. Diseño e Implementación del Caso de Estudio **37**
4.1. Contexto y definición del Problema 37
4.1.1. Propuesta de la Agencia de Sistemas de Información 45
4.1.2. Nuestra primera propuesta 45
4.1.3. Modelo definitivo implementado 48
4.2. Modelo de programación lineal entera 50
4.2.1. Parámetros 51
4.2.2. Variables 51

4.2.3. Función objetivo y restricciones 51
4.2.4. Encontrando múltiples óptimos 53
4.3. Pujas y resultados computacionales 55

5. Conclusiones **62**

A. Algunos resultados sobre subastas **64**

B. Código Zimpl **73**

C. Código C ++ **77**

Capítulo 1

Subastas - Subastas por Combinaciones

Los tratamientos teóricos de las subastas usualmente analizan situaciones en las cuales hay un bien en particular a ser comprado o vendido y la pregunta es qué formato de subasta utilizar, en función de los resultados deseados. El diseño y la implementación de subastas para su aplicación en el mundo real, sin embargo, son problemas necesariamente mucho más abarcativos ([2], [34], [36]).

El caso paradigmático es la subasta de procuración de bienes. Por ejemplo, cuando un gobierno decide comprar un bien o adquirir un servicio en forma global, debe decidir los tamaños de las distintas regiones a ser beneficiadas, qué garantías de desempeño requerir y a qué niveles, cómo medir y eventualmente premiar la calidad, si tomar en cuenta las diferencias en *performances* pasadas de proveedores y en caso afirmativo cómo hacerlo, etc. Se observa entonces la necesidad de resolver problemas adicionales surgidos de la práctica ([37]).

A menudo estas decisiones de *presentación* o *empaquetado*, que establecen qué será vendido o comprado en la subasta y cómo serán comparadas las pujas, se encuentran entre las decisiones más críticas que hace el subastador. Es de esperar que, en el caso de una procuración gubernamental, si el gobierno decide especificar que las pujas ofreciendo un servicio deben cubrir grandes porciones de territorio, entonces los pequeños proveedores serían efectivamente excluidos e impedidos de pujar. Al mismo tiempo, los grandes proveedores, presentando economías de escala al servir a un gran mercado, podrán dar señales de una reducción de costos al disminuir sus pujas (notar que en este caso una puja consiste, en la práctica, en pedir un precio a cambio de un determinado servicio). Entonces, dados estos efectos de compensación entre proveedores de distinta envergadura, sería importante establecer las áreas de provisión de servicio para no cercenar desde el comienzo las posibilidades de participantes significativos.

Quien tenga un objeto para vender, a menudo desea venderlo al mayor precio que pueda obtener. En ocasiones existe un mercado establecido que se encarga del problema de la determinación de precios, fuera del alcance de quien pone un bien en venta. Pero bajo ciertas condiciones, la situación puede reducirse a una que en microeconomía se conoce como problema *principal - agentes*. Un ejemplo de esto es una subasta, en la que el subastador es el *principal* y los participantes son los *agentes*. Decimos entonces que *el mecanismo elegido por el principal para asignar bienes es una subasta* ([24], [31]). El término *licitación*, que también usamos, refiere a un caso particular de subasta, en el que hay un marco legal público y que mayormente se implementa en el contexto gubernamental para la procuración de algún bien o servicio.

La teoría de subastas se encuentra entre los temas más influyentes estudiados en economía durante los últimos cuarenta años. Las subastas formulan y responden ciertas preguntas fundamentales de esta ciencia, como lo son "¿quién debería obtener los bienes y a qué precios?" Al responder estas cuestiones, las subastas proveen el micro-fundamento de los mercados, es decir explican el resultado agregado en función de los comportamientos individuales, donde por resultado agregado se entiende la asignación final de los recursos de la economía: la lista de agentes, los recursos iniciales de cada uno y las características de los agentes (valuaciones, utilidades, preferencias). De hecho muchos mercados modernos estan organizados como subastas ([27]).

Veamos los tipos de subasta simple comúnmente considerados ([6], [23], [30], [41]). En el apéndice A presentamos y demostramos resultados importantes de la teoría de subastas que en particular vinculan a todos ellos.

Subastas de primer precio, a sobre cerrado Este es el formato estandarizado de subastas para licitaciones gubernamentales. Cada comprador potencial determina y envía su puja de manera privada. El subastador se encarga de vender el objeto a quien haya hecho la puja más alta, y se lo vende exactamente al precio que este último haya comunicado (el subastador necesita eventualmente definir alguna noción de ruptura de empates; se suele asumir que se sortea el ganador de manera aleatoria de entre aquellos que hayan presentado la puja más alta).

Subastas Inglesas Un substador invita a hacer pujas (orales). Las pujas continúan hasta que nadie esté dispuesto a continuar subiendo el precio. Si nadie interrumpe con una nueva puja, entonces el objeto es asignado a quien haya hecho la última (mayor) puja.

Subastas Holandesas El subastador comienza anunciando un precio alto. Este precio es entonces decrementado gradualmente (puede ser automáticamente, con un reloj) hasta que un postor llama a la detención. El primer participante en hacer el llamado adquiere el bien al presente precio en el momento de su intervención.

Subastas Todos-Pagan Las pujas se efectúan como en una subasta inglesa, siendo el mayor postor quien se lleva el bien, pero *todos* pagan su puja más alta, incluyendo a todos aquellos que no son ganadores de ningún bien.

Subastas de Vickrey Las subastas de segundo precio fueron por primera vez propuestas y analizadas por William Vickrey en 1961, año en que escribió el artículo de teoría de juegos pionero en subastas. Hoy en día constituye el paradigma económico predominante para pensar las subastas. Vickrey sostenía la necesidad del uso de subastas diseñadas especialmente cuando se trata de bienes públicos importantes (Vickrey [47], [48]). En una subasta de Vickrey el bien es vendido al postor más alto, pero al precio de la puja perdedora más alta. Este será el *segundo* precio más alto, a menos que hubiera un empate para el primer lugar, en cuyo caso el ganador sería designado aleatoriamente de entre los mejores postores. Se asume que las pujas son enviadas simultáneamente por todos los postores usando el mecanismo de *sobre cerrado* y por esto se identifica una subasta Vickrey con una subasta de segundo precio a sobre cerrado.

Ver en el apéndice A las definiciones de *beneficio* (*payoff*) y *estrategia dominante*.

Teorema 1.1. *(Vickrey 1961) En una subasta de segundo precio, para todo jugador j es una estrategia dominante el pujar su verdadera valuación v_j. Es decir, si b_j es la puja de j, entonces sea como pujen los otros jugadores, es óptimo para j establecer $b_j = v_j$. Más aun, la subasta de segundo precio es eficiente.*

Demostración. Ver el Teorema A.1 en el apéndice A. □

Teorema 1.2. *(Vickrey 1961) La Subasta de Segundo Precio y la Subasta Inglesa resultan en un* payoff *equivalente para todos los jugadores.*

Demostración. Ver el Teorema A.6 en el apéndice A. □

Teorema 1.3. *(Vickrey 1961) Cuando las valuaciones $\{v_i\}$ son tomadas aleatoriamente de manera independiente de una* misma *distribución con función de distribución acumulada F y con soporte en $[0, 1]$ (esto es, $F_1 = ... = F_N$, hipótesis que llamaremos* de simetría*) entonces la subasta* high bid *(primer precio a sobre cerrado), la subasta de segundo precio, la subasta inglesa, la subasta holandesa y la subasta* todos pagan *son todas equivalentes en términos del* payoff*.*

Demostración. Ver el Teorema A.7 en el apéndice A. □

A continuación probaremos que, bajo no muy restrictivas hipótesis, la subasta de segundo precio es la *única* subasta eficiente tal que el pago que efectúa un postor ganador no depende de su puja, a menos de una expresión sumada al pago de cada jugador j, que *no* depende de b_j, la puja de j.

Teorema 1.4. *Supongamos que para todo j, v_j pueda tomar cualquier valor en $[0, 1]$. Entonces una subasta es eficiente y pujar verazmente es débilmente dominante si y sólo si valen (a) el postor más alto gana y (b) para todo jugador j el pago p_j satisface*

$$p_j = \begin{cases} \max_{i \neq j} b_i + t_j(b_{-j}), & \textit{si } j \textit{ gana;} \\ t_j(b_{-j}), & \textit{si } j \textit{ pierde.} \end{cases}$$

para cierta función t_j, donde b_{-j} es el vector de pujas con b_j excluida.

Demostración. Ver el Teorema A.3 en el apéndice A. □

La regla por la cual los ganadores pagan un segundo precio, o un primer precio perdedor, puede parecer contraproducente a primera vista, en términos de la maximización del ingreso del subastador, puesto que el participante ganador ha expresado una volutad de pagar un precio mayor. Sin embargo, no hay que perder de vista que esta idea afecta drásticamente las motivaciones y los comportamientos de cualquier postor. Los postores adaptan su comportamiento a las reglas de juego determinadas por el subastador: cuán alto pujen los participantes depende de qué tipo de subasta es usado. En este caso, debería resultar intuitivo que los jugadores en una subasta de segundo precio presentarán pujas estrictamente mayores que las que presentarían en una de primer precio, pues en caso de ganar sólo tendrían que pagar algún precio menor que el que comunicaron. Como fundamento formal a lo que acabamos de decir, en su artículo de 1961 Vickrey probó un teorema de neutralidad del ingreso, que afirma que bajo ciertas hipótesis la subasta de segundo precio resulta en el mismo ingreso promedio para el subastador que aquel en el que resultaría una subasta de primer precio a sobre cerrado.

Teorema 1.5. *Asumimos las hipótesis del Teorema 1.3 y suponemos que*

$$\forall v \; J'(v) > 0, \tag{1.1}$$

donde

$$J(v) = v - \frac{1 - F(v)}{F'(v)}.$$

Entonces cualquiera de las subastas del Teorema 1.3 maximiza el ingreso esperado del subastador, siempre que el subastador defina un precio de reserva v^* *tal que* $J(v^*) = 0$.

Demostración. Ver el Teorema A.8 en el apéndice A. □

La indagación/investigación original de Vickrey trató tanto acerca de subastas de un sólo ítem como de subastas de múltiples ítems idénticos, proveyendo un mecanismo en el cual es una estrategia dominante para los postores el reportar sus valuaciones verazmente y en el que los resultados son eficientes[1]. Para un sólo ítem, al mecanismo se lo menciona como la subasta de segundo precio a sobre cerrado, o simplemente como la subasta Vickrey.

En presencia de las reglas de la subasta de Vickrey, un postor ganador no podría nunca cambiar el precio que paga ni hacer que este último se vea afectado, así que no hay incentivos para que ningún postor tergiverse su valuación. Desde la perspectiva del postor, el monto de su puja determina solamente si gana, y sólo comunicando su valuación verdadera puede estar seguro de, en caso de ganar, pagar a lo sumo el precio que estaba dispuesto a pagar.

En el tratamiento original de Vickrey del caso de ítems múltiples de un bien homogéneo, que puede estar disponible tanto en cantidades continuas como discretas, se asume que cada postor tiene valores marginales monótonos no crecientes para el bien. Por ejemplo, como pasa en la siguiente asignación definida de las cantidades de ítems ganados en los precios pagados (como suma de precios unitarios):

$$cantidad\ de\ bienes \xrightarrow{p} precios$$

$$n \xmapsto{p} pn$$

$$n + k \xmapsto{p} pn + (p - \delta)k$$

donde δ es un número no negativo y estrictamente menor que p, definido por supuesto por cada participante en función de n y de k.

Los postores envían simultáneamente pujas cerradas que constituyen *curvas de demanda*[2]. El subastador combina las curvas de demanda individuales de manera usual para determinar una curva de demanda agregada

[1] Se dice que la implementación de un mecanismo es *eficiente* cuando a cada bien se lo lleva quien debería llevárselo, es decir, quien realmente haya valuado cada bien de manera maximal. Si la implementación del mecanismo asigna el bien i al postor j al precio p, existiendo un resultado posible que mantenga la misma asignación de todos los bienes salvo para i, asignado en este caso a un postor j' distinto de j que haga una valuación $p' > p$, entoces este último resultado devendría en un ingreso mayor para el subastador, y la implementación primera no sería eficiente

[2] Aquí cometemos un abuso de terminología. Es usual reservar las palabras "oferta" (o curva o función de oferta) y "demanda" para funciones bien definidas del precio de mercado. En el caso de una subasta, sea para vender o para comprar (procuración) no existe un precio definido hasta que no se hagan efectivas las pujas, y en este caso sólo para el adjudicado quedará bien definido el precio.

y un precio de equilibrio para una cantidad fija de unidades. Cada postor gana la cantidad demandada de bienes si esta se realiza en el precio y la asignación de equilibrio. Sin embargo, en vez de pagar el precio con que pujó, un postor ganador paga el *costo de oportunidad* por las unidades ganadas.

Ver en el apéndice A una formalización del mecanismo VCG.

Teorema 1.6. *En el mecanismo VCG, pujar verazmente (ie. $b_j := v_j$) es una estrategia dominante para cada jugador j.*

Demostración. Ver el Teorema A.2 en el apéndice A. □

El mecanismo puede ser usado como un mecanismo para vender (una subasta clásica) o bien como un mecanismo para comprar (una subasta "reversa"). Descripta como una subasta clásica, los compradores pagan con un descuento en relación al precio de equilibrio. Descripta como una subasta reversa, los vendedores reciben primas en comparación con el precio de equilibrio. Como fue mencionado más arriba, una de las principales virtudes del mecanismo es que las conclusiones no son sensibles a las asunciones acerca de qué puedan saber los postores acerca de las valuaciones y estrategias de los demás. No obstante, una razón obvia para el desuso del mecanismo de Vickrey en aplicaciones de gran escala con diversos ítems es la misma que para otras subastas en combinaciones o por paquetes: la complejidad en todos los aspectos de su implementación ([7], [33], [43], [44]).

En su trabajo, Vickrey mostró con un ejemplo de modelo lo que luego se probaría en forma general como el *teorema de equivalencia de ingresos*, que establece que dados distintos mecanismos de subasta, si estos definen la misma asignación de bienes, entonces producen el mismo ingreso al subastador (ver las secciones 3.3.1 y 3.4.1 para las definiciones de asignación de bienes e ingreso del subastador).

Decisiones que definen una subasta Las razones por las cuales usar subastas como mecanismo para asignar bienes son usualmente las siguientes:

- Las subastas son rápidas.
- Son difíciles de corromper.
- Ayudan a revelar las preferencias, a obtener información acerca de las valuaciones de los compradores ([8]).

Los economistas destacan la tercera de estas razones, que se da porque los participantes están forzados a poner su dinero en los bienes por los que expresaron interés.

En primer lugar, quien decide instanciar una subasta ha rechazado ya un número de alternativas importantes, por ejemplo, la de publicar un precio y esperar por alguien que compre el bien a ese precio. Decidirse por una subasta significa decidirse por un proceso relativamente formal para vender o adquirir bienes.

Las siguientes son cuestiones importantes para resolver al momento de diseñar e implementar una subasta.

¿Qué se vende? Quizás lo primero que hay que decidir al optar por hacer una subasta sea qué precisamente es lo que se pone en venta (o se adquiere). En el caso de la subasta de licencias tal vez también haya que decidir qué usos deberían autorizarse para cada licencia. En este caso un participante ganador aún tiene que probar que está legalmente calificado y los competidores pueden objetar y retrasar el otorgamiento de una licencia. Por ejemplo, en una subasta de bancarrota el subastador puede necesitar decidir si vender una compañía entera como una unidad o subastar el conjunto de todos los bienes individuales que la conforman en forma simultánea, qué garantías ofrecer concernientes a las condiciones físicas de sus bienes, qué términos y condiciones de financiamiento requerir, cuánto tiempo conceder a los compradores para alcanzar los requisitos de aprobación para la adquisición de los bienes.

¿Quién deberá participar? Una segunda decisión que debe hacer alguien que organiza una subasta es quién puede presentar pujas. ¿A cualquiera le está permitido pujar? Si no, ¿qué certificaciones, cualidades, serán requeridas?

Cómo serán comparadas las pujas en pos de seleccionar al ganador. Si la calidad del participante es importante, esta tiene que ser definida, medida de alguna manera y probablemente restringida.

Cómo será puesto el precio. Si ha de haber una transacción luego de la determinación de los ganadores, cómo se definirá el precio de los bienes de antemano. Una opción es que sea el monto de la puja ganadora, como en el caso de las subastas *standard* a sobre cerrado, mientras que otra opción es que sea el monto de la mejor puja perdedora, como en las subastas de segundo precio (*Vickrey Auctions*), como vimos anteriormente.

¿Qué información se revelará y cuándo? Finalmente, ¿qué información se revelará después de la subasta? Para la discusión de que se ocupa el presente párrafo definimos el siguiente término ([13]).

Definición 1.1. *Una colusión es un pacto ilícito en daño de un tercero.*

En una secuencia de subastas involucrando a los mismos postores (subastas a más de una ronda), revelar regularmente las pujas perdedoras al término de cada ronda puede ayudar a los jugadores a hacer colusión para las rondas siguientes, por ejemplo, turnándose. En contraposición a esto, mantener las pujas en secreto por tiempo indeterminado puede ayudar

justamente a tapar alguna posible colusión. De manera que el asunto involucra no sólo determinar qué información revelar sino también cuándo hacerlo.

Modos de pensar las subastas Los teóricos de los juegos construyen modelos analíticos de subastas, usualmente de subastas simples aisladas. En estos modelos se calculan equilibrios de Nash, en el caso de subastas con valores privados: un equilibrio de Nash en un modelo de subastas es un conjunto de estrategias para cada jugador tal que ningún jugador tiene nada que ganar mediante cambios *unilaterales* de sus estrategias. La importancia de calcular estos equilibrios se debe precisamente a que las subastas de valores privados se modelan como juegos *no cooperativos*. En una subasta que no sea de Vickrey, además, no necesariamente existen estrategias dominantes.

Definición 1.2. *Decimos que una subasta es de pagos balanceados si*

$$\sum_{j=1}^{N} p_j = 0,$$

donde p_j es el pago que realiza el jugador j.

Teorema 1.7. *Supongamos que $\{v_j\}$, las valuaciones de cada jugador j, sean aleatorias e independientes cada una de una distribución con función de distribución acumulada F_j y con soporte en $[0, 1]$. Entonces existe una subasta eficiente y de pagos balanceados en la que pujar verazmente constituye un equilibrio Bayesiano.*

Demostración. Ver el Teorema A.4 en el apéndice A. □

Teorema 1.8. *Bajo las hipótesis del Teorema 1.7, dadas dos subastas tales que, en equilibrio Bayesiano, (a) para todo j y v_j, la probabilidad de ganar para un jugador j con valuación v_j es la misma en ambas subastas, y (b) para todo j, el monto que paga el jugador j con valuación 0 es la misma en ambas subastas, entonces, para todo j y v_j, el* pago esperado de equilibrio *para el jugador j con valuación v_j es el* mismo *en ambas subastas.*

Demostración. Ver el Teorema A.5 en el apéndice A. □

Como las subastas involucran incertidumbre y los postores habitualmente tienen información privada, estas estrategias no son las pujas propiamente dichas, sino funciones de la información privada del postor, que a menudo es su estimación del valor de los ítems (o del costo, en el caso de subastas reversas para adquirir bienes).

La investigación operativa/*management science* presenta modelos orientados a la decisión. A diferencia de los teóricos de los juegos, los investigadores operativos no se sienten obligados a limitar sus modelos analíticos

a unos en los cuales todos los jugadores se comporten exactamente de manera óptima. Se asume que los agentes usan "reglas del pulgar" simples pero plausibles, en lugar de ser optimizadores perfectamente racionales.

Subastando múltiples ítems La literatura de teoría de juegos comenzó a hacer foco en subastas de múltiples ítems luego de 1994, cuando los teóricos de los juegos se vieron involucrados en el diseño de las subastas de la Comisión Federal de Comunicaciones en Estados Unidos (Federal Communications Commission -FCC- auctions, Klemperer, 1999, [17], [23]).

La subasta de múltiples ítems implica problemas adicionales de los que ocuparse. Por ejemplo, se tiene que decidir si los ítems se subastarán simultáneamente o secuencialmente (o en una secuencia de subastas simultáneas de menor número [40]).

Si los ítems son subastados simultáneamente, también hay que decidir si se permitirá a los postores pujar por combinaciones de ítems. En caso afirmativo, además, ¿será permitida cualquier combinación? En el caso de no permitirse todas las combinaciones posibles, el subconjunto de combinaciones permitidas también debe estar bien definido.

Si los ítems en subastas son idénticos, la aceptación de *curvas de demanda* como pujas, o lo que en nuestro caso serán tramos tarifarios, forma parte del diseño, v. g. "100 ítems a un precio de hasta $1000 y 20 más a $990".

Las subastas por combinaciones son aquellas subastas en las que los postores efectivamente pueden establecer pujas por combinaciones de ítems, llamados "paquetes", en vez de sólo hacerlo por ítems individuales y su estudio es inherentemente interdisciplinario. En primer lugar las *subastas* son un tópico importante de la microeconomía. En segundo lugar, la posibilidad de hacer pujas por paquetes lleva el trabajo al área de la investigación operativa, y hace especialmente necesarias las técnicas de la optimización combinatoria y la programación matemática; de hecho los investigadores operativos también fueron contribuyentes activos a los primeros trabajos en subastas (Friedman 1956 [19] y Rothkopf 1969 [39]) y la mayor parte del trabajo temprano en subastas apareció por primera vez en revistas de investigación operativa. Por último, la computación científica interviene ocupándose de la expresividad de los distintos lenguajes de pujas y de los aspectos algorítmicos y de complejidad computacional del problema de optimización combinatoria. Ha habido un aumento significativo de investigación acerca de la determinación de los ganadores en subastas por combinaciones (Sandholm, 2002). El estudio de las subastas por combinaciones está por esto situado en la intersección de la economía con la investigación operativa y la computación científica.

Las subastas por combinaciones pueden ser estudiadas en un amplio rango de entornos de subastas, determinados por características importantes como la cantidad de vendedores y compradores, la cantidad de bie-

nes puestos en venta, las preferencias de las partes, y el tipo de información privada que cada participante tenga acerca de las preferencias de los demás. Se puede escribir como observación que la mayoría de los entornos de subasta tienen tanto elementos de valores comunes como privados.

La ventaja de las Subastas por Combinaciones (CAs) es que los participantes pueden expresar sus preferencias más plenamente. Esto es particularmente importante cuando los bienes son *Complementos*. Los mercados combinatorios en general, en los que las pujas pueden ser expresadas en paquetes de ítems pueden ser mecanismos de coordinación deseables económicamente en sistemas *multiagente* donde los ítems son complementos y al mismo tiempo substitutos.

El componente clave es que cada postor pueda expresar de manera más completa su interés y pujar por paquetes de ítems. Recientemente, variadas industrias han utilizado subastas por combinaciones. Por ejemplo, han sido usadas para *industrial procurement* (determinación de precios y cantidades en un mercado de bienes industriales), y han sido propuestas para la venta de *slots* de arribo y partida de aeropuertos como así también para la asignación de epectro radial para servicios de comunicación inalámbrica ([15], [17], [38]). En cada caso, la motivación preponderante para usar subastas por combinaciones es la presencia de complementariedades entre los bienes, es decir que la suma de las utilidades de dos bienes por separado es menor que la utilidad del conjunto de ambos bienes: $u(\{j_1\}) + u(\{j_2\}) < u(\{j_1, j_2\})$, que difieren de un postor a otro ([5]).

Se asume que el objetivo del subastador es la optimización de los ingresos que devengan de la subasta. Si esto es así, el trabajo de optimización a ser llevado a cabo una vez que una lista de posturas es enviada puede ser formulado a menudo como un problema de programación entera. Si se permiten pujas por cualquier combinación de ítems, entonces este cómputo puede volverse inmanejable, al menos en instancias del "peor caso" ([20]).

1.1. Definiciones y notaciones básicas. Hipótesis

Subasta Multiunitaria - Multiunit Auction Subasta por combinaciones en la que se ponen a la venta conjuntos de ítems idénticos. Los postores expresan la cantidad deseada de cada tipo de ítem.

Subasta Inversa - Reverse Auction Subasta para comprar bienes o servicios. Hay un comprador (subastador) y muchos vendedores competidores. Esto es lo "inverso" de la subasta habitual, para vender bienes o servicios en la que hay un vendedor y muchos compradores competidores. Este tipo de subastas también es llamado subasta de procuración, pues son subastas programadas para procurarse bienes o servicios.

Notaremos por $\mathcal{S} = \{1, ..., m\} \subset \mathbb{N}$ al conjunto de ítems en subasta y por $\mathcal{J} = \{1, ..., n\} \subset \mathbb{N}$ al conjunto de postores (equivalentemente, jugadores,

participantes). Observar que el conjunto $\mathcal{J}$ no incluye al subastador, quien pone los ítems en venta.

Notamos también por $\mathfrak{P}(\mathcal{S})$ al conjunto de partes de $\mathcal{S}$.

El participante no tiene la necesidad de saber nada acerca del uso que hagan los demás con los bienes ganados, y se espera que aplique sólo lo que él sabe para determinar su valuación privada de esos ítems.

Definición 1.3. *Una Valuación es una función $v : \mathfrak{P}(\mathcal{S}) \longrightarrow \mathbb{R}_{\geq 0}$ que asigna un valor real $v(S)$ a cada subconjunto de ítems S.*

Observar que $v(S) \geq 0$ para todo paquete $S \subset \mathcal{S}$. Dado un jugador $i \in \mathcal{J}$, la valuación que depende de i es notada v_i. Notamos por $\mathcal{V}$ al conjunto de todas las valuaciones de los jugadores.

Se dice que la subasta es de *valores privados* cuando cada postor (o participante) tiene una valuación por cada paquete de ítems, y estas valuaciones no dependen de la información privada de los otros postores. Cada participante conoce sus valores pero no necesariamente los valores de los otros participantes. En el caso de estudio del Capítulo 4 asumiremos la hipótesis de valores privados.

La siguiente hipótesis asumida es llamada *libre disposición*: $v_i(T) \geq v_i(S)$ para todo $T \supseteq S$. Equivalentemente, se puede considerar la siguiente definición (Cramton, Shoham y Steinberg 2006).

Definición 1.4. *Se dice de una función de valuación v que satisface la condición de libre disposición si $v(S \cup T) \geq v(S)$ para todas las combinaciones S y T.*

En particular estamos diciendo que disponer de un ítem de una combinación no puede incrementar el valor de la combinación, siempre y cuando se entienda, en este contexto, "libre" como "de costo nulo" y "disposición" como "desechar" o "eliminar" (i.e. dejar de consumir o utilizar) ciertas cantidades de los bienes. Es decir, disponer de un ítem tiene costo nulo.

Observar que sin esta última asunción, **nunca** se podría llegar a un equilibrio. Veamos este ejemplo:

Sean dados dos jugadores i y j, y sus respectivas funciones de valuación v_i y v_j, una cierta cantidad de un bien de clase a y una cierta cantidad otra de un bien de clase b. Notemos por el par (n, m) al paquete conformado por n unidades de a más m unidades de b. Supongamos que las valuaciones de los bienes unitarios cumplan

$$v_i(1, 0) > v_i(1, 1)$$

y

$$v_j(1, 0) \leq v_j(1, 1)$$

En una asignación en la que resulten ganadores tanto i como j, cada uno de un paquete $(1,1)$, es decir de una unidad de a más una unidad de b cada uno, el jugador j tendría *incentivo* a ofrecer al jugador i el paquete $(1,0)$ a cambio del paquete $(1,1)$ del jugador i. O sea j daría su unidad de a a cambio de una unidad de a más una unidad de b, con lo cual obtendría una unidad más de b al precio de $\{a,b\}$.

También asumimos la hipótesis de *normalización*: $v_i(\emptyset) = 0$.

Dados un paquete $S \subseteq \mathcal{S}$ y un número real $p_i \geq 0$ definimos la *utilidad del paquete S al precio* p_i para el postor i como $u_i(S, p_i) = v_i(S) - p_i$.

Para juegos con información privada se define un *mecanismo de revelación directa* como una aplicación de la información privada del jugador en los resultados de utilidad. Más en general, un *mecanismo* es una forma de describir y determinar las posibles acciones de cada agente. Es una aplicación de las acciones de los agentes en el conjunto de resultados. Constituye una especificación para una secuencia de acciones contingentes efectuadas por uno o más agentes.

El problema de la Asignación Eficiente por Combinaciones es:

$$\max_{S=(S_1,\ldots,S_n)} \sum_{i \in \mathcal{J}} v_i(S_i)$$

sujeto a:

$$S_i \cap S_j = \emptyset, \quad \forall i, j.$$

Definición 1.5. *Los ítems son complementos cuando un conjunto de ítems tiene mayor utilidad que la suma de las utilidades de sus ítems individuales.*

También para el subastador tienen un valor agregado las subastas por combinaciones. Permitir a los participantes expresar de manera más completa sus preferencias tiene a menudo como consecuencias un mejoramiento en la eficiencia económica (asignar los bienes a quienes realmente los valoraron más) y un mayor ingreso de la subasta.

Definición 1.6. *Decimos que un par de bienes son substitutos cuando el aumento del precio de uno de ellos no resulta en una disminución de la demanda por el otro.*

1.1.1. Posturas y Eficiencia

A continuación nos ocuparemos del problema de decidir cuál debe ser el lenguaje de pujas o posturas (*bidding language*). Las diferentes opciones varían en expresividad y en simplicidad. Una puja en una subasta es una expresión de la preferencia de un participante por los diversos resultados o asignaciones ([32]). La manera más directa de capturar o registrar tales preferencias es haciendo que cada participante asigne un valor monetario a

cada posible asignación de bienes. Este método, si bien permitiría expresar de manera completa todas las posibles posturas, carece de simplicidad. En efecto, supongamos que el conjunto de participantes es de cardinal n y que el conjunto de bienes en subastsa es de cardinal m. Sabemos que el conjunto de funciones (o asignaciones)

$$\{1, ..., n\}^{\{1,...,m\}} := \{\text{funciones } u, u : \{1, ..., m\} \longrightarrow \{1, ..., n\}\}$$

tiene cardinal n^m, que es exactamente el tamaño de la puja que estaríamos obligando a cada jugador a presentar. Si asumimos inclusive la ausencia de consideraciones externas, de manera que cada participante se preocupe solamente por los bienes que él mismo recibe, la complejidad decrece a 2^m, que es el cardinal del conjunto de partes del conjunto de ítems, pero que aún es impráctica para todos lo m naturales salvo para pequeños valores.

Entre los lenguajes de puja que mencionaremos se encuentran el lenguaje OR ("o aditiva"), que permite al participante presentar posturas por paquetes de manera *no exclusiva* pero puede capturar solamente (todas) las valuaciones superaditivas. En contraste, el lenguaje XOR ("o exclusiva"), que permite al jugador pujas *exclusivas* por paquetes, permite capturar todas las valuaciones, pero requiere de expresiones exponencialmente más largas que el lenguaje OR. Sin embargo, pedirle a un participante que comunique una valuación exhaustiva es a menudo innecesario, porque muchas partes de tal valuación pueden ser irrelevantes para computar la asignación de bienes.

1.1.2. Complejidad y Consideraciones Algorítmicas

Una vez fijado el lenguaje de pujas, la cuestión sigue siendo cómo computar la asignación de bienes dado un conjunto de pujas. Este problema es llamado el *problema de la determinación del ganador* (WDP, *winner determination problem*). El problema es el siguiente: dado un conjunto de pujas en una subasta por combinaciones, encontrar una asignación de bienes a postores, incluyendo la posibilidad de que el subastador retenga algunos ítems, de manera que se maximicen los ingresos del subastador. El problema, que es naturalmente representado como un problema de programación lineal entera, es intrínsecamente complejo. Específicamente, es un problema NP-Completo, con lo que es improbable que exista un algoritmo de tiempo polinomial que compute la asignación óptima. Peor aún, el problema es no uniformememte aproximable, en el siguiente sentido: casi con certeza no existe un algoritmo de tiempo polinomial y una constante d tales que, para todos los inputs, el algoritmo produzca una respuesta que esté a lo sumo a $1/d$ de la respuesta óptima. Todos estos resultados se ampliarán en el Capítulo 3.

1.1.3. El Problema de la Procuración

El Problema de la Procuración es el problema de la asignación por combinaciones en el que los ítems son adquiridos *de* los agentes, y el objetivo es *minimizar* el costo total de tal adquisición, sujeta a restricciones dadas por una cantidad minimal de bienes que se necesita procurar.

Denotando el costo de procurar del jugador i el paquete $S \subset \mathcal{S}$ por $c_i(S)$, el costo total de una asignación $j \in \mathcal{J}^{\mathcal{S}}$ es

$$\sum_{i \in \mathcal{J}} c_i(j^{-1}(i))$$

donde $j^{-1}(i)$ es la preimagen, o imagen inversa, del jugador i por la asignación j. El análisis de eficiencia exacta en el problema de la procuración es exactamente el mismo que en el problema de "venta" de bienes, mencionado anteriomente.

En este caso se asume como hipótesis que los costos sean *subaditivos*.

Luego de todo lo expuesto hasta ahora acerca de la topología de las subastas podemos observar que el subtipo de subasta que nos ocupará en este trabajo será una subasta de **procuración reversa multiunitaria de primer precio, a sobre cerrado, de una sola ronda**.

El objetivo es que el principal obtenga, de cada agente, el número de bienes que realice la igualdad entre la demanda y la oferta totales (habitual en las versiones a sobre cerrado de las *multiunit auctions* reversas).

1.2. Resumen de la tesis

Primer capítulo En el primer capítulo describimos suscintamente la teoría microeconómica de diseño de mecanismos de subastas, enumeramos clases conocidas de subastas e identificamos el contexto teórico y la subclase de subasta en la que se encuentra la implementación que describimos en el Capítulo 4.

Segundo capítulo En este capítulo se describen brevemente dos aplicaciones exitosas a problemas reales de las subastas por combinaciones. Se trata de dos casos de procuración pública de bienes mediante una subasta reversa, situación similar a la que nos ocupará en el Capítulo 4.

El primer caso comentado es uno de los primeros casos importantes de los que se tiene registro en la literatura especializada acerca de subastas de procuración ([11]). El segundo caso es considerado de mucha importancia porque constituye un caso exitoso de aplicación en un país en desarrollo, con claros beneficios materiales para el *principal*, es decir en este caso, el Estado Nacional ([14], [16]).

Tercer capítulo Aquí estudiamos los conceptos básicos de la complejidad computacional como tópico fundamental de la computación científica que se ocupa de la tratabilidad de problemas computacionales, sean o no de optimización combinatoria.

Más adelante formulamos los problemas computacionales asociados a una subasta por combinaciones, en términos de programación lineal entera, lo que facilita el uso de los conceptos a que se hace referencia en el párrafo anterior para dejar en evidencia la aparente intratabilidad de estos problemas.

Cuarto capítulo Aquí es donde hacemos nuestro aporte al problema de la determinación de los ganadores en una subasta de procuración por combinaciones.

Describimos el problema en su contexto fáctico y luego las propuestas que ofrecimos en la discusión con el principal, la Agencia de Sistemas de Información del Gobierno de la Ciudad Autónoma de Buenos Aires.

Finalmente describimos la propuesta aceptada para trabajar en el problema y nuestro diseño e implementación originales mediante programación lineal entera.

Quinto capítulo En este capítulo exponemos las conclusiones del trabajo realizado.

Capítulo 2

Aplicaciones exitosas

2.1. Subasta de rutas de Omnibus de Londres

El mercado de ómnibus de Londres provee uno de los primeros ejemplos de uso de una clase de subasta por combinaciones en el contexto de la procuración pública (ver [11] y [22]). Este mercado comprende alrededor de 800 rutas a través de un área de 1630 kilómetros cuadrados, usadas por más de 3,5 millones de pasajeros por día. Antes de la etapa de desregulación los servicios de ómnibus en el Gran Londres eran provistos por la empresa estatal *London Buses Limited*. El acta de Transporte Regional de Londres de 1984 reorganizó el sector y designó a *London Regional Transport* (LRT, empresa de transporte estatizada en 1933) como la autoridad responsable de la procuración y provisión de servicios públicos de transporte en el Gran Londres. El acta también instrumentaba un sistema de franquicias permitiendo a LRT invitar a operadores privados a presentar variantes de proyectos para llevar a cabo los servicios de ómnibus.

A los efectos de aumentar la competencia LRT implementó una división separada de "tendering" independiente de su división operacional. La introducción de subastas de rutas fue gradual hasta llegar al formato de subasta adoptado definitivamente, que es una variante de una subasta por combinaciones de primer precio. En una subasta por combinaciones de primer precio típica los postores envían pujas simultáneamente sobre cualquier ítem individual, o bien sobre cualquier paquete de ítems, y el subastador resuelve el problema de la determinación del ganador mediante la determinación de la mejor asignación de ítems a postores (en los casos de procuración la asignación óptima es la que minimiza el costo total), y los ganadores reciben el monto con el que pujaron por los bienes que ganaron.

Así como en una subasta por combinaciones estándar, en la subasta LRT los postores pueden pujar por cualquier número de rutas o paquetes de rutas. No hay restriccón en el número de pujas a establecer. La característica distintiva de la subasta LRT es que cada puja es un compromiso firme a

brindar los recursos necesarios para el servicio pero de manera no exclusiva, en el sentido de que dos pujas por rutas diferentes definen implícitamente una puja por el paquete formado por esas rutas. Una importante consecuencia de esta regla es que los participantes no tienen permitido pujar más alto por un paquete que la suma de las pujas sobre cualquier partición. La motivación original para imponer esta regla fue el hecho de que el mercado estaba caracterizado principalmente por economías de escala (y por el alcance de cada postor), y que al permitir a los jugadores expresar todo esto, LRT vería reducidos sus costos y mejoraría la eficiencia económica.

La subasta de rutas de ómnibus de Londres devino en un incremento de la calidad del servicio y en un decremento de los costos para el gobierno de Londres.

2.2. Una subasta por combinaciones para licitar contratos de servicios gastronómicos para las escuelas en Chile

El sistema educativo chileno está usando actualmente modelización matemática para asignar contratos y licencias de servicios gastronómicos para las escuelas públicas mediante una subasta por combinaciones reversa de ronda simple y a sobre cerrado (ver [16]).

El país está dividido en *Unidades Territoriales* (UT's). La Junta Nacional de Auxilio Escolar y Becas (JUNAEB, un organismo del Estado Chileno) implementa una subasta por año en un tercio de estas unidades territoriales, otorgando contratos que duran tres años. Cada puja especifica la cobertura de un conjunto de unidades territoriales y cada firma puede presentar tantas pujas como desee, siendo estas aceptadas o rechazadas de manera íntegra. El objetivo es dar la mayor calidad posible de servicio a todas las UT's, dentro del presupuesto. El modelo le dio transparencia y objetividad al proceso completo, generando competencia entre las distintas empresas. También permitió a las compañías construir pujas territoriales con flexibilidad para incluir sus economías de escala (dado que JUNAEB permite pujar por paquetes de UT's, las empresas pueden tomar ventaja de las economías de escala que se presentan por ofrecer un gran número de servicios). Este modelo trajo consigo una mayor complejidad pero aún así se pudo encontrar una solución óptima.

Para evitar la concentración excesiva, que podría hacer vulnerable al programa, se pusieron cotas superiores a los números de UT's a ser asignadas a una firma. Cada firma tiene una cota distinta, dependiente de su capacidad financiera y operativa. Asimismo se consideraron sólo las pujas por encima de un precio predeterminado para eliminar las pujas irrealmente bajas (notar que esta también es una subasta de *procuración*).

En esta aplicación se trabajó también un problema que se menciona en la literatura como la *maldición del ganador* (ver [14]). A partir de 2004 la industria de empresas de alimentos comenzó a exhibir un preocupante nivel de quiebras. Ciertas bancarrotas se dieron en empresas participantes de la licitación después de haber ganado unidades territoriales.

Estas quiebras provocaron grandes pérdidas para el Estado Chileno por dos motivos. En primer lugar, reponer el servicio resultó ser mucho más costoso que los contratos originales, principalmente porque licitar pocas UT's es un proceso menos competitivo y tienden a presentarse las compañías que están operando en las vecindades de las UT's que quedan desabastecidas. Además necesariamente los contratos que se licitan son de más corta duración y dificultan sensiblemente la amortización de las inversiones. En segundo lugar, se provoca perjuicios a los alumnos, que se ven afectados por la interrupción de los servicios de alimentación. Muchos de estos jóvenes provienen de hogares modestos y por lo tanto el programa de JUNAEB representa una parte importante de sus dietas. Toda la política pública de alimentación escolar aparece cuestionada por estas quiebras.

Un factor importante que provocaba las quiebras era el desconocimiento, por parte de algunas empresas, sobre la realidad operacional de ciertas UT's que presentan condiciones desmejoradas cuando se las compara con la mayoría de la UT's del país. El punto central es que algunas empresas no evaluaban en forma adecuada UT's que son de *atractivo* singularmente bajo para operar, proponiendo muy buenos precios que les permitieran ganar ciertas unidades pero que sin embargo no les permitían cumplir con los contratos. Otras empresas conocían mejor la realidad operacional y no cometían este error. El principal motivo para explicar esta situación de asimetría de información entre postores radica en el factor geográfico.

Se advierte entonces la conveniencia de homogeneizar lo más posible las características de la UT's, luego de medir apropiadamente su *atractivo*, a fin de minimizar los riesgos de quiebra, problema que constituye la principal motivación de [14].

Al igual que en el caso del Capítulo 4, para encontrar la asignación óptima para cada escenario, se formuló el problema como un modelo de programación lineal con variables binarias. Se definió una variable binaria por cada puja, en donde la decisión es aceptar la puja o rechazarla.

El uso de un modelo matemático por parte de JUNAEB trajo consigo mejoras capitales al proceso de otorgamiento de los contratos de servicios gastronómicos para las escuelas. Comparando la subasta hecha en 1999 (ya con el nuevo mecanismo) con aquél proceso al que reemplazó, usado en 1995, sigue que se dieron progresos substanciales en la calidad nutricional de los alimentos tanto como en la infraestructura de los servicios y en las condiciones de trabajo de los empleados de las firmas participantes. Todas

estas mejoras aumentaron el *costo* total en un 24 por ciento (notar que el bien otorgado por las firmas ahora es otro). Pero, si bien las tendencias de precio predijeron un incremento de al menos el 22 por ciento para obtener más alta calidad, el precio promedio de las porciones de comida creció sólo el 0,76 por ciento, gracias al nuevo mecanismo implementado. Este ahorro sumó unos 40 millones de dólares al año, equivalente al monto necesario para alimentar a 300.000 alumnos durante un año.

Capítulo 3

Dos Problemas Computacionales en Subastas

En este capítulo se estudian el Problema de la Determinación de los Ganadores y el Problema de la Asignación Optima, dos problemas computacionales centrales en subastas por combinaciones.

Antes de detallar los dos problemas mencionados desarrollamos ciertos conceptos de la teoría de complejidad computacional necesarios para su estudio.

3.1. Complejidad computacional

En esta sección daremos las primeras definiciones y propiedades de los problemas y algoritmos para poder comenzar a discutir y formalizar la noción de tratabilidad de un problema, en el sentido de intentar responder si puede ser resuelto algorítmicamente en tiempo razonable.

3.1.1. Tamaños de instancias y entradas de algoritmos

Como veremos en este capítulo, mediremos la complejidad de un algoritmo con una función del tamaño de la entrada de ese algoritmo. Surge entonces la primera pregunta acerca de qué es el tamaño de una entrada.

En problemas de optimización combinatoria la entrada es un objeto combinatorio, como pueden serlo una familia de conjuntos finitos de enteros, o un grafo. Para comunicar esta entrada a una máquina para la solución del problema, independientemente del modelo de cómputo utilizado, debemos codificarla de alguna manera, es decir *representarla* como una secuencia de símbolos sobre algún alfabeto fijo, como el $\{0, 1\}$.

Una vez elegido el alfabeto, y decidido que la entrada de un algoritmo sea representada como una secuencia (cadena de caracteres) de símbolos, podemos escribir la siguiente definición.

Definición 3.1. *Dado un alfabeto Σ el tamaño de la entrada de un algoritmo es la* longitud *de la secuencia de símbolos en Σ que la representa, es decir, el número de símbolos en ella.*

3.1.2. Función de complejidad de un algoritmo

La medida del rendimiento de un algoritmo más ampliamente aceptada es el tiempo que consume antes de producir la respuesta final a un problema. En el análisis de algoritmos expresamos los requerimientos de tiempo en términos del número de operaciones elementales necesarias para su ejecución. Para esto asumimos que cada operación elemental requiere *tiempo unitario*.

Para clasificar el comportamiento de un algoritmo independientemente de cualquier entrada particular tomamos el conjunto de todas las entradas de un tamaño dado, y definimos la complejidad del algoritmo para ese tamaño como el comportamiento de *peor caso* del algoritmo en esas entradas. Es decir que, si denotamos $|E|$ al tamaño de la entrada E y $N_{\mathcal{A}}(E)$ al número de operaciones elementales que realiza el algoritmo $\mathcal{A}$ para producir la respuesta con la entrada E, la función de complejidad del algoritmo $\mathcal{A}$ es la siguiente asignación:

$$\varphi_{\mathcal{A}} : \mathbb{N} \longrightarrow \mathbb{N}$$

$$\varphi_{\mathcal{A}}(n) = \max_{|E|=n} N_{\mathcal{A}}(E)$$

Al estudiar la complejidad de un algoritmo estaremos interesados solamente en el comportamiento del algoritmo con entradas de tamaño grande, pues son aquellas entradas las que determinarán cuándo un algoritmo es aplicable. Las constantes multiplicativas en funciones de complejidad no son consideradas pues no afectan a la razón de crecimiento de éstas: las diferencias entre un algoritmo de complejidad $2n^3$ y uno de complejidad n^3 pueden volverse irrelevantes mediante un cambio tecnológico que duplique la velocidad de una máquina. Por otro lado, los términos de crecimiento más lento eventualmente serán eclipsados por los términos de crecimiento más rápido, siendo n suficientemente grande, como pasa con el término $\log(n)$ en la cota $n^2 + \log(n)$. Recordar que $\frac{\log(n)}{n^2}$ puede hacerse tan cercano a cero como se quiera con tal de tomar n suficientemente grande, lo que implica que, asintóticamente, $n^2 + \log(n)$ se comporta igual que n^2.

Por todo esto es que estamos interesados en la *razón de crecimiento* de la complejidad de un algoritmo. Y es para estudiar las razones de crecimiento de las funciones que usaremos el siguiente formalismo.

Definición 3.2. *Dadas dos funciones $f : \mathbb{N} \longrightarrow \mathbb{R}^+$ y $g : \mathbb{N} \longrightarrow \mathbb{R}^+$ escribimos*

$$f(n) = O(g(n))$$

o bien

$$f(n) \in O(g(n))$$

si existe una constante $c > 0$ y un natural n_0 tales que para todo $n > n_0$ $f(n) \leq cg(n)$.

3.1.3. Clasificación de algoritmos

La utilidad práctica de un algoritmo estará determinada por la razón de crecimiento de su mejor cota de tiempo conocida. La cuestión es, entonces, qué tasas de crecimiento consideraremos como soluciones aceptables a problemas computacionales.

Existe un consenso general según el cual un algoritmo es una solución útil en sentido práctico a un problema computacional sólo si su función de complejidad crece acotada por algún polinomio.

Definición 3.3. *Decimos que un algoritmo $\mathcal{A}$ es eficiente si existe un polinomio $p \in \mathbb{R}[x]$ tal que $\varphi_{\mathcal{A}}(n) = O(p(n))$, donde $\varphi_{\mathcal{A}}$ es la complejidad de $\mathcal{A}$.*

Alternativamente llamaremos *de tiempo polinomial* a los algoritmos eficientes.

Los algoritmos para los cuales la complejidad asintótica no sea un polinomio en sí mismo pero esté acotada por algún polinomio también califican para estar en esta clase. Son ejemplos de esto $n^{2,5}$ y $n\log(n)$.

Para entender la importancia de la clase de los algoritmos polinomiales, consideraremos la clase de los algoritmos restantes, aquellos que sobrepasan cualquier cota polinomial, y a los que nos referiremos, por convención, con el término de *algoritmos exponenciales* o *no eficientes*, aún cuando la mejor cota para su función de complejidad no sea exactamente una función exponencial.

Definición 3.4. *Decimos que un algoritmo es de complejidad exponencial si no es de complejidad polinomial, es decir, si no es eficiente.*

Ejemplos de razones de crecimiento exponencial son 2^n, (o bien k^n, para cualquier $k > 1$), $n!$, 2^{n^2}, n^n, $n^{\log(n)}$. Observar que existen funciones que crecen más rápido que cualquier polinomio pero más lento que 2^{n^ϵ} para todo $\epsilon > 0$, como lo es nuestro último caso citado $n^{\log(n)}$. A estas tasas de crecimiento se las suele llamar también *subexponenciales*.

Los algoritmos eficientes toman más ventaja de los avances tecnológicos. Cada vez que un punto de ruptura tecnológica aumenta diez veces la velocidad de las computadoras, el tamaño de la instancia más grande resoluble por un algoritmo polinomial dentro de una hora se multiplicará por una constante con valor entre 1 y 10. En contraste, un algoritmo exponencial experimentará tan sólo un aumento *aditivo* en el tamaño de la instancia que pueda resolver dentro de una cantidad fija de tiempo. Finalmente,

destacamos que los algoritmos eficientes tienen interesantes propiedades de *clausura*: los algoritmos polinomiales pueden ser combinados para resolver casos especiales del mismo problema; un algoritmo polinomial puede invocar a otro algoritmo polinomial como una *subrutina* y el algoritmo resultante seguirá siendo eficiente.

3.1.4. Complejidad de un problema

Definimos informalmente la complejidad de un problema para su clasificación.

Definición 3.5. *La complejidad de un problema es la mejor de las complejidades de los algoritmos que resuelven el problema.*

La siguiente definición ayuda a materializar la noción acerca de cuándo un problema de computación se considera satisfactoriamente solucionado.

Definición 3.6. *Decimos que un problema está bien resuelto si existe un algoritmo eficiente que lo resuelve.*

Veamos algunos importantes ejemplos de problemas computacionales bien resueltos.

El Problema del Camino Dirigido Mínimo (*Shortest-Path Problem*) es el siguiente. Dado un grafo dirigido $G = (V, E)$ y un *peso* o *costo* $c_j \geq 0$ asociado con cada arco $e_j \in E$, una instancia del *Problema del Camino Dirigido Mínimo* es el problema de encontrar un camino dirigido desde un nodo fuente distinguido (s) a un otro nodo distinguido (la terminal t), con costo total mínimo.

Este problema es un caso de problema bien resuelto. Con el algoritmo de Dijkstra podemos encontrar su solución en tiempo $O(n^2)$, en donde n es el cardinal de V, el conjunto de vértices del grafo.

Dado un arreglo de n enteros, el problema de ordenar el arreglo, por ejemplo de manera creciente, es otro problema bien resuelto, y su complejidad es $O(n \log(n))$. Esta complejidad se realiza por ejemplo en el algoritmo *Merge Sort* que pertenece al paradigma *Divide and Conquer*, y tiene complejidad de peor caso del orden de $n \log(n)$.

Naturalmente, dado cualquier problema computacional, por ejemplo uno de Optimización Combinatoria, podemos preguntarnos si puede ser bien resuelto. Es decir, si la complejidad intrínseca del problema admite algoritmos eficientes para su resolución.

Veamos el siguiente problema llamado *máximo clique*.

Definición 3.7. *Dado un grafo $G = (V, E)$ el problema del máximo clique es el de encontrar el subconjunto $C \subseteq V$ de cardinalidad máxima que cumpla que para cualesquiera $u, v \in C$ distintos, el arco $[u, v]$ pertenece a E (notar que los conjuntos finitos tienen máximo).*

Es decir que lo que se busca es un subgrafo de G tal que todos sus vértices estén conectados entre sí, y que tenga la mayor cantidad de vértices posible con esta propiedad.

Cualquiera que haya hecho el intento de resolver eficientemente este problema y se haya encontrado con que ninguno de los algoritmos exactos encontrados es substancialmente distinto a una enumeración exhaustiva de todos los candidatos a solución, podría preguntarse: "¿existe tal cosa buscada, un algoritmo eficiente que resuelve el *máximo clique*?". Una de las dos respuestas a esta pregunta afirmaría que el problema es *inherentemente intratable*, es decir que el problema tiene una complejidad intrínseca tal que todo algoritmo que lo resuelva necesariamente será de tiempo exponencial. Desafortunadamente, probar intratabilidad puede ser tan arduo como encontrar algoritmos eficientes, sea lo que corresponda en cada caso.

Para no caer en ninguna de estas situaciones extremas, la teoría de NP-Completitud provee muchas técnicas directas para probar que un problema dado es simplemente tan duro de tratar como un gran número de otros problemas reconocidos ampliamente como dificultosos y que han desconcertado a los expertos por años. Aún cuando no se sepa todavía si la NP-Completitud de un problema implique su intratabilidad inherente, el conocimiento acerca de su condición de NP-Completo brinda información valiosa acerca de qué lineas de trabajo tienen el potencial de ser más productivas. La búsqueda de un algoritmo exacto eficiente debería ser considerada de baja prioridad y, por ejemplo, sería más apropiado concentrarse en desarrollar algoritmos polinomiales que resuelvan casos particulares del problema general, o bien, tratándose del caso general, en desarrollar algoritmos que encuentren la solución rápidamente en muchas instancias a pesar de no garantizar que lo haga de esa manera para las instancias de peor caso.

El problema del máximo clique en un grafo es un problema para el cual no se conoce aún un algoritmo eficiente y para el cual tampoco se ha podido probar la intratabilidad. Constituye un ejemplo importante de aquellos problemas que discutiremos en la siguiente sección, los problemas NP-Completos ([10], [35]).

3.2. Problemas NP-Completos

Ituitivamente, un problema de decisión es un problema computacional en el que las respuestas posibles son *Sí* y *No*, o bien, 0 y 1.

Definición 3.8. *Un problema de decisión es un par $(\mathcal{I}, \psi)$ donde $\mathcal{I}$ es el conjunto de instancias del problema y ψ es una función*

$$\psi : \mathcal{I} \longrightarrow \{0, 1\}$$

La función ψ en la definición anterior es tal que para $I \in \mathcal{I}$, $\psi(I)$ vale 0 si y sólo si la instancia I es afirmativa. Lo más importante aquí, que es para lo cual aparece la mayor dificultad, es justamente, dado un problema de decisión, computar su función ψ.

Intimamente relacionados con los problemas de decisión, se definen los *problemas de optimización*. Definimos una *instancia*, en este caso, como un par (F, c), en donde F es un conjunto cualquiera finito (el conjunto de puntos *factibles*) y $c : F \longrightarrow \mathbb{R}$ es la aplicación *costo*. Entonces ahora definimos un problema de optimización como sigue.

Definición 3.9. *Un problema de optimización es un conjunto de instancias* $\{I_j = (F, c)_j\}_{j \in J}$*, con algún conjunto de índices* J*.*

El objetivo aquí es, dada una instancia (F, c), hallar un y perteneciente a F tal que para todos los x en F valga $F(y) \leq F(x)$.

Existe otra versión de un problema de optimización que es particularmente importante en el estudio de la complejidad del problema por ser la más cercana al prototipo de problemas computacionales estudiados tradicionalmente por la teoría de la computación y que se llama la *versión de decisión* del problema. La versión de decisión es de hecho una pregunta que puede ser respondida por *Sí* o por *No* y tiene la siguiente forma.

> Dada una instancia de un problema de optimización *y un entero* L, ¿existe una solución factible $f \in F$ tal que $c(F) \leq L$?

La versión de decisión de un problema de optimización es, en sí misma, un problema de decisión.

Estamos interesados en clasificar los problemas de decisión de acuerdo a su complejidad. Notaremos por P a la clase de problemas de decisión que pueden ser resueltos por un algoritmo de tiempo polinomial. En otras palabras, P es la clase de problemas bien resueltos, aquellos para los cuales, dada una instancia de cada uno, tenemos una manera eficiente de afirmar si la respuesta es *Sí* o *No*.

Informalmente podemos definir NP en términos de lo que deberíamos llamar un *algoritmo no determinístico*. Un tal algoritmo está compuesto por dos etapas separadas, siendo la primera una etapa de *conjetura* (*guessing stage*) y la segunda una etapa de *verificación* (*checking stage*). Dada una instancia I de un problema, la primera etapa meramente "conjetura" alguna estructura S. Entonces proveemos ambas estructuras I y S como *inputs* para la etapa de verificación, que procede a computar de manera determinística normal, terminando eventualmente con respuesta *Sí*, eventualmente con respuesta *No*, o bien sin siquiera terminar y permanecer computando continuamente (los dos últimos casos no necesitan ser distinguidos).

Para un problema Π denotamos por D_Π a la clase de todas sus instancias y por Y_Π a la clase de sus instancias *afirmativas*. Un algoritmo no determinístico "resuelve" un problema de decisión Π si valen las siguientes dos propiedades:

1. Si I pertenece a Y_Π entonces existe una cadena de caracteres S que, conjeturada para el input I, producirá, en la etapa de verificación, la respuesta *Sí* para I y S.

2. Si I no pertenece a Y_Π entonces no existe ninguna estructura S que, conjeturada para el input I, produzca, en la etapa de verificación, la respuesta *Sí* para I y S.

Se dice que un algoritmo no determinístico que resuelve un problema de decisión Π opera en tiempo polinomial si existe un polinomio p tal que para toda instancia en Y_Π existe alguna conjetura S que produciría, en la etapa de verificación, la respuesta *Sí* para I y S en tiempo $O(p(|I|))$. Notar que esto tiene el efecto de imponer una cota polinomial también para el tamaño de la estructura S conjeturada, pues sólo puede usarse una cantidad de tiempo acotada polinomialmente para examinar esa conjetura. Es esta noción de *verificabilidad en tiempo polinomial* la que la clase NP intenta capturar. Notemos que verificabilidad en tiempo polinomial no implica, por supuesto, resolubilidad determinística en tiempo polinomial.

La clase NP se define como la clase de todos los problemas de decisión Π que puedan ser resueltos por algoritmos no determinísticos de tiempo polinomial. Aquí el término "resolver" es sólo usado como una formalidad; un "algoritmo no determinístico de tiempo polinomial" es meramente un dispositivo definicional para capturar la noción de verificabilidad eficiente, en lugar de ser un método realista en un modelo de cómputo para resolver problemas de decisión.

Convenimos la siguiente notación para formalizar la definición de NP: sea Σ un alfabeto finito fijo y sea $\$$ un símbolo distinguido en Σ. Este símbolo distinguido marca el fin de la entrada del algoritmo y el comienzo del certificado.

Dada una cadena de símbolos de Σ, x, entonces su longitud es denotada por $|x|$.

Definición 3.10. *Decimos que un problema de decisión* Π *pertenece a la clase NP si existe un polinomio* $p(n)$ *y un algoritmo* $\mathcal{A}$ *(el* algoritmo de verificación de certificado*) tal que la siguiente afirmación es verdadera:*

> *La cadena* x *es una instancia afirmativa del problema* Π *si y sólo si existe una cadena de símbolos en* Σ*, el certificado* $c(x)$*, tal que* $|c(x)| \leq p(|x|)$*, y con la propiedad de que* $\mathcal{A}$*, provisto del* input $x\$c(x)$*, alcanza la respuesta* Sí *en tiempo* $O(p(|x|))$*.*

Definición 3.11. *Una* transformación polinomial *de un problema de decisión* Π_1 *a un problema de decisión* Π_2 *es una función* $f : D_{\Pi_1} \longrightarrow D_{\Pi_2}$ *que satisface las dos condiciones:*

1. f *es computable por un algoritmo polinomial y*
2. $\forall\, I \in D_{\Pi_1}, I \in Y_{\Pi_1} \iff f(I) \in Y_{\Pi_2}$.

Si existe una transformación polinomial de Π_1 a Π_2 escribimos $\Pi_1 \propto \Pi_2$, que se lee "Π_1 se transforma en Π_2".

La importancia de las transformaciones polinomiales se desprende del siguiente lema.

Lema 3.1. *Si* $\Pi_1 \propto \Pi_2$ *entonces* $\Pi_2 \in P$ *implica* $\Pi_1 \in P$.

Definición 3.12. *Un problema de decisión* Π *es NP-Completo si valen las dos siguientes condiciones:*

1. $\Pi \in NP$ *y*
2. $\forall\, \Pi' \in NP\ \Pi' \propto \Pi$.

El Lema 3.1 nos hace identificar a los problemas NP-Completos como los problemas más duros de NP en el siguiente sentido.

- Ningún problema NP-Completo conocido puede ser resuelto por ningún algoritmo eficiente conocido.
- Si hubiera un algoritmo polinomial para un problema NP-Completo cualquiera entonces habría algoritmos polinomiales para *todos* los problemas NP-Completos.

Otra consecuencia importante del Lema 3.1 es que los problemas NP-Completos tienen la propiedad de estar en P si y sólo si P = NP. Decidir la veracidad de la relación P = NP constituye uno de los problemas abiertos más importantes de la actualidad.

Si bien hasta aquí discutimos la complejidad de problemas pertenecientes a la clase NP, con conceptos similares se puede probar que ciertos problemas fuera de NP también son *duros* de tratar. cualquier problema de decisión Π, miembro de NP o no, al cual podamos transformar un problema NP-Completo, tendrá la propiedad de no poder ser resuelto en tiempo polinomial a menos que P = NP. Diremos que un tal problema es NP-Hard, puesto que es, en algún sentido tan duro (*hard*) como los problemas NP-Completos.

Definición 3.13. *Decimos que un problema de optimización* Π *es NP-Hard si existe un problema NP-Completo* Π' *tal que* $\Pi' \propto \Pi$.

3.3. El problema de la Optimización del Ingreso

La característica fundamental que hace más atractivas a las subastas por combinaciones es, como ya dijimos, la posibilidad que tienen los postores de expresar preferencias complejas sobre colecciones de ítems que exhiben la condición de ser complementos algunos y substitutos otros. Es esta generalidad del conjunto de pujas lo que hace extremadamente difícil a los participantes el proveer el *input* para una subasta por combinaciones. En efecto, cada postor debe enviar su *función de valuación* definida sobre el espacio de todos los paquetes de ítems, lo que, dados m bienes en venta, hace que cada postor deba definir pujas sobre $2^m - 1$ paquetes.

3.3.1. Formulación del problema

El problema de la optimización del ingreso (*revenue*) del subastador puede ser formulado como sigue. Denotamos por $\mathcal{S}$ al conjunto de todos los bienes individuales siendo subastados. Asumimos que tenemos m ítems, es decir, $|\mathcal{S}| = m$. Cualquier subconjunto $C \subseteq \mathcal{S}$ representa una combinación de bienes. Las reglas de cada subasta pueden especificar cuáles $C \subseteq \mathcal{S}$ son **combinaciones permitidas**, esto es, aquellas combinaciones de bienes por las cuales los jugadores tienen permitido enviar pujas. La familia de todas las combinaciones permitidas es denotada por $\mathcal{P}$. Inmediatamente se tiene $|\mathcal{P}| \leq 2^m$.

En cualquier resultado obtenido en una subasta en donde se ponen en juego distintos bienes simultáneamente, las combinaciones ganadoras deben ser disjuntas, pues ningún bien puede ser vendido más de una vez. Formalmente, denotemos por $\Omega_{\mathcal{P}}$ al conjunto de **resultados** factibles:

$$\Omega_{\mathcal{P}} := \{\mathcal{W} \subseteq \mathcal{P} \,:\, C, C' \in \mathcal{W} \Rightarrow C \cap C' = \emptyset\}.$$

Sea $b(C)$ la mayor de las pujas enviadas para la combinación C. Si no hay pujas por C entonces definimos $b(C) = 0$. Definimos ahora la siguiente asignación de la familia de resultados factibles $\mathcal{W}$ en los números reales:

$$rev \,:\, \Omega_{\mathcal{P}} \longrightarrow \mathbb{R}$$

dada por

$$rev(\mathcal{W}) := \sum_{C \in \mathcal{W}} b(C)$$

En otras palabras $rev(\mathcal{W})$ -del inglés *revenue*- es el ingreso que recogería el subastador si los bienes fueran vendidos a los jugadores que enviaron las propuestas maximales para los paquetes (combinaciones) en $\mathcal{W}$. Lo llamamos el *ingreso*, para el subastador, del resultado factible $\mathcal{W}$. Notar la diferencia entre calcular el ingreso respecto de calcular la ganancia.

En esta sección el foco estará puesto en el problema del subastador, notado (REV$_{\mathcal{P}}$), consistente en determinar un resultado factible que optimice el ingreso, al que notaremos $\mathcal{W}_{\text{opt}}$.

Definición 3.14. ***[Problema de la Optimización del Ingreso,*** **(REV$_{\mathcal{P}}$).]** *El problema de la optimización del ingreso es el problema de determinar $\mathcal{W}_{opt} \in \Omega_{\mathcal{P}}$ tal que*

$$rev(\mathcal{W}_{opt}) = max\{rev(\mathcal{W}) \,:\, \mathcal{W} \in \Omega_{\mathcal{P}}\}.$$

Llamamos a un tal $\mathcal{W}_{\text{opt}}$ un resultado óptimo, y llamamos, a cualquier combinación $C \in \mathcal{W}_{\text{opt}}$, una **combinación ganadora** de $\mathcal{W}_{\text{opt}}$. Notemos que no es requerimiento que todo bien sea vendido.

El problema de encontrar un resultado óptimo, (REV$_{\mathcal{P}}$), puede ser formulado como un problema de programación lineal entera como sigue ([46]):

queremos

$$\max \sum_{C \in \mathcal{P}} b(C) x_C$$

sujeto a las restricciones

$$(\forall i \in \mathcal{S}) \sum_{C | i \in C} x_C \leq 1$$

$$(\forall C \in \mathcal{P}) \;\; x_C \in \{0,\, 1\}$$

El problema que acabamos de formular se denomina SET PACKING en un hipergrafo $\mathcal{P}$ con pesos $b(C)$ para todo $C \in \mathcal{P}$, un muy conocido problema de optimización combinatoria NP-Completo (ver Garey & Johnson, 1979 [18] y Karp, 1972 [21]). Por supuesto que con esta observación queda demostrado que el problema de la Optimización del Ingreso también es NP-Completo. Sin embargo, existen numerosos casos especiales para los cuales el problema es manejable computacionalmente. Por ejmplo, siempre que la matriz M de (1) sea *totalmente unimodular* (se dice de una matriz de enteros que es totalmente unimodular si el determinante de cualquier submatriz cuadrada es 0, 1 o -1), el problema puede ser resuelto en tiempo polinomial (ver [12], [42] y Schrijver, 1986 [46]).

Este problema puede ser escrito de una manera más compacta en la forma

$$\max \{\mathbf{b}^T \mathbf{x} \,:\, M\mathbf{x} \leq \mathbf{1},\; \mathbf{x} \in \{0,\, 1\}^{|\mathcal{P}|}\}, \qquad (1)$$

donde

$$M \in \{0,\, 1\}^{m \,\times\, |\mathcal{P}|}$$

$$M = (m_{i,C})$$

$$m_{i,C} = 1 \iff i \in C$$

mientras que

$$\mathbf{b} = (b(C))_{C \in \mathcal{P}}$$

y **1** es el vector de unos de $\{0,\ 1\}^m$ (recordar que $m = |\mathcal{S}|$).

La siguiente observación muestra que la superaditividad en las combinaciones tiene una consecuencia inmediata en la determinación de pujas ganadoras. Una combinación C^* nunca será una combinación ganadora si alguno de los bienes en C^* pudiera ser vendido por un precio total más grande:

Observación 3.1. *Sea $\mathcal{W}_{opt}$ una asignación óptima (resultado óptimo) y sea $C^* \in \mathcal{W}_{opt}$. Sean C_1, C_2, ... , C_k combinaciones permitidas disjuntas dos a dos: C_1, C_2, ... , $C_k \in \mathcal{P}$ y $C_i \cap C_j = \emptyset$ para todos $1 \leq i < j \leq k$ tales que $\cup_i C_i \subseteq C^*$. Entonces vale*

$$b(C^*) \geq \sum_{i=1}^{k} b(C_i).$$

Demostración. Definimos $\mathcal{W}' := (\mathcal{W}_{\text{opt}} \setminus \{C^*\}) \cup \{C_1, C_2, ..., C_k\}$. Notar que las hipótesis implican que $\mathcal{W}'$ es una asignación factible. Ahora

$$\sum_{C \in \mathcal{W}'} b(C) = \sum_{C \in \mathcal{W}_{\text{opt}}} b(C) - b(C^*) + b(C_1) + b(C_2) + ... + b(C_k)$$

Por la definición de la función rev y observando que $\mathcal{W}_{\text{opt}}$ es una asignación óptima

$$\sum_{C \in \mathcal{W}_{\text{opt}}} b(C) = rev(\mathcal{W}_{\text{opt}}) \geq rev(\mathcal{W}') = \sum_{C \in \mathcal{W}'} b(C)$$

Entonces

$$\sum_{C \in \mathcal{W}_{\text{opt}}} b(C) - b(C^*) + b(C_1) + ... + b(C_k) \leq$$

$$\leq \sum_{C \in \mathcal{W}_{\text{opt}}} b(C),$$

con lo que, cancelando, queda

$$b(C_1) + ... + b(C_k) \leq b(C^*).$$

□

3.4. El problema de la Determinación del Ganador

El otro problema fundamental de optimización combinatoria presente en una subasta por combinaciones es el *problema de la determinación del ganador*. A diferencia del problema anterior, que aparentemente sólo tiene un interés teórico (y con importantes consecuencias de esta índole), este problema tiene una importancia fáctica grande y es realmente en él donde se vuelcan todos los esfuerzos computacionales "del lado del subastador".

Sin dejar que confunda la similitud entre las desigualdades de un problema y otro, la diferencia drástica se observa en el hecho de que es este problema el que contesta la cuestión acerca de qué participante debe llevarse qué bien y por qué.

El problema de la determinación del ganador (en inglés WDP — *Winner Determination Problem*) es el siguiente: dado un conjunto de pujas (*bids*) en una subasta por combinaciones, encontrar una asignación de ítems a postores (el subastador puede conservar algunos de los ítems) que maximiza los ingresos del subastador (notar que no hablamos de *ganancia* del subastador, que implicaría considerar, por ejemplo, los costos de llevar a cabo la subasta). Las pujas son expresiones en un lenguaje determinado, mediante las cuales los jugadores reportan sus valuaciones por subconjuntos de ítems. El ingreso del subastador es optimizado eligiendo una asignación que maximice la suma, sobre todos los postores, de las valuaciones de los postores hechas por el subconjunto de ítems que reciben. No discutiremos aquí lo concerniente a la veracidad de las valuaciones, en el sentido de que los postores pueden no hacer valuaciones reales por motivos estratégicos. Tampoco consideraremos el hecho de que en algunos mecanismos de subasta el precio a pagar por los postores ganadores difiera de la propia postura (Vickrey, 1961), implicando que el valor de la asignación óptima no sea igual al ingreso percibido por el subastador.

3.4.1. Formulación del problema

Como hasta ahora, usamos la siguiente notación. El conjunto de postores es notado por $\mathcal{J} = \{1, ..., n\}$, el conjunto de ítems por $\mathcal{S} = \{1, ..., m\}$. Un paquete S es un conjunto de ítems: $S \subseteq \mathcal{S}$. Para un paquete S y un jugador i notamos por $v_i(S)$ la puja que el jugador i hace por el paquete S, esto es, el precio maximal que i anuncia estar dispuesto a pagar por S.

Definición 3.15. *Una asignación de ítems es un conjunto de variables binarias* $(x_i(S) \in \{0, 1\} | i \in \mathcal{J}, S \subseteq \mathcal{S})$.

Decimos que una asignación $(x_i(S) \,|\, i \in \mathcal{J}, S \subseteq \mathcal{S})$ es *factible* si no asigna ningún ítem más de una vez,

$$\sum_{i \in \mathcal{J}} \sum_{S \subseteq \mathcal{S}, S \ni j} x_i(S) \leq 1 \quad \text{para todo} \quad j \in \mathcal{S} \tag{3.1}$$

y si asigna a lo sumo un subconjunto de ítems a cada postor

$$\sum_{S \subseteq \mathcal{S}} x_i(S) \leq 1 \quad \text{para todo} \quad i \in \mathcal{J}. \tag{3.2}$$

Definición 3.16. ***[Problema de la Determinación del Ganador, WDP.]*** *Dadas pujas v_i, $i = 1, ..., n$, el* problema de la determinación del ganador *es el problema de computar*

$$x = \arg\max \sum_{i \in \mathcal{J}, S \subseteq \mathcal{S}} v_i(S) x_i(S) \tag{3.3}$$

sujeto a las restricciones 3.1 y 3.2.

Observación 3.2. *(Buena definición de la restricción 3.2) Dada una asignación óptima, podemos suponer que $x_i(S) = 1$ si y sólo si el postor i obtiene la combinación S como paquete* maximal*, es decir que si $x_i(S) = 1$ entonces para todo $T \subsetneq S$ vale $x_i(T) = 0$.*

Demostración. Recordemos las siguientes asunciones que habíamos postulado: para cualquier postor i suponemos

- $\forall\, S, T \subseteq \mathcal{S}\ v_i(S \cup T) \geq v_i(S) + v_i(T)$
- $\forall\, S \subseteq \mathcal{S}, |S| \geq 2 \Rightarrow \forall\, s \in S\ v_i(S) > v_i(\{s\})$,

que llamamos *complementariedad* y *libre disposición*. Supongamos una asignación óptima $x^* = (x_i^*(S) \,|\, i \in \mathcal{J}, S \subseteq \mathcal{S})$ y paquetes A, B y C tales que

$$C = A \dot{\cup} B$$

y tales que A y B hayan sido asignados al jugador i_0:

$$x_{i_0}^*(A) = x_{i_0}^*(B) = 1$$

$$x_{i_0}^*(C) = 0.$$

Ahora definamos una nueva asignación $\tilde{x} = (\tilde{x}_i(S) \,|\, i \in \mathcal{J}, S \subseteq \mathcal{S})$ igual a la anterior, salvo por

$$\tilde{x}_{i_0}(A) = 0$$

$$\tilde{x}_{i_0}(B) = 0$$

$$\tilde{x}_{i_0}(C) = 1.$$

Dado que habíamos supuesto $v_i(A \cup B) \geq v_i(A) + v_i(B)$, si f es nuestra función objetivo definida por (3.3),

$$f(\tilde{x}) = \sum_{i \in \mathcal{J}, S \subseteq \mathcal{S}} v_i(S) \tilde{x}_i(S) =$$

$$= \sum_{i \neq i_o} \sum_{S \subseteq \mathcal{S}} v_i(S)\tilde{x}_i(S) + \sum_{S \subseteq \mathcal{S}} v_{i_0}(S)\tilde{x}_{i_0}(S) =$$

$$= \sum_{i \neq i_o} \sum_{S \subseteq \mathcal{S}} v_i(S)\tilde{x}_i(S) + \sum_{S \neq A,B,C} v_{i_0}(S)\tilde{x}_{i_0}(S) + v_{i_0}(C) =$$

$$= \sum_{i \neq i_o} \sum_{S \subseteq \mathcal{S}} v_i(S)x_i^*(S) + \sum_{S \neq A,B,C} v_{i_0}(S)x_{i_0}^*(S) + v_{i_0}(C) \geq$$

$$\geq \sum_{i \neq i_o} \sum_{S \subseteq \mathcal{S}} v_i(S)x_i^*(S) + \sum_{S \neq A,B,C} v_{i_0}(S)x_{i_0}^*(S) + v_{i_0}(A) + v_{i_0}(B) =$$

$$= \sum_{i \in \mathcal{J}} \sum_{S \subseteq \mathcal{S}} v_i(S)x_i^*(S) = f(x^*).$$

Entonces la asignación $\tilde{x}$ también es óptima, con lo cual podemos suponer que el óptimo del problema es uno de paquetes maximales. □

WDP es un problema de optimización combinatoria más general que (REV$_{\mathcal{P}}$). En el problema del ingreso la función objetivo considera sólo la máxima puja hecha por cada combinación permitida de bienes. En cambio en el presente WDP se consideran todas las combinaciones factibles de asignación de bienes a postores, sin necesidad de que el ganador de algún paquete haya hecho la mejor puja por él. En otras palabras, en WDP si por el paquete S sólo pujaron los jugadores i y j y las valuaciones respectivas cumplen la relación $v_i(S) \geq v_j(S)$, esto no implica que el jugador i se gane el paquete S. En conclusión, como habíamos visto que (REV$_{\mathcal{P}}$) es un SET PACKING y WDP es aún más general, resulta que WDP es al menos NP-Hard (ver Garey & Johnson, 1979 [18] y [45]).

3.4.2. Observaciones ulteriores

En el caso en que todas las combinaciones posibles de bienes son permitidas, si el número de combinaciones para las cuales se envían ofertas crece exponencialmente en n entoces el problema de la asignación óptima podría resolverse en tiempo polinomial en el tamaño del input. Por supuesto que en esta situación, común en muchos problemas de optimización combinatoria, si el tamaño del input creciera exponencialmente en n, tan sólo el problema de registrar todos los datos del input no sería computacionalmente manejable ([3]).

No hay manera de asegurar que el proceso de subastas simultáneas sea manejable para valores grandes de n si no se imponen restricciones al conjunto de combinaciones permitidas, no sólo porque el problema sea NP-completo, sino también porque el tamaño de los datos de entrada en sí mismo podría volverse inmanejable.

En algunos mercados de subastas se ponen a la venta simultáneamente una cantidad considerable de bienes. A menudo para un jugador el valor

de un bien depende de qué otros bienes él gana. En ese tipo de situaciones, el permitir el envío de ofertas por combinaciones de ítems puede ofrecer un modo de incrementar la eficiencia de la asignación de bienes, pero puede al mismo tiempo agrandar las dificultades computacionales ([9]).

Capítulo 4

Diseño e Implementación del Caso de Estudio

4.1. Contexto y definición del Problema

Actualmente en la Ciudad de Buenos Aires existe una decena de empresas proveedoras de conexiones de internet con distintas tecnologías. Las opciones son conexión por pulso telefónico o *dial-up* (actualmente casi en desuso), conexión a través de una línea telefónica digitalizada o *ADSL*, el cable coaxial de cobre, la fibra óptica (considerada la mejor de las tecnologías por cable) y por último la conexión inalámbrica o *Wi-fi*. Las empresas presentes cubren, todas juntas, toda la Ciudad. Las tecnologías enumeradas más arriba ofrecen un servicio cuya velocidad de transferencia se encuentra entre los 512 kilobits por segundo y los 6 megabits por segundo aproximadamente, y cualquiera de ellas se puede adquirir para uso particular por alrededor de $ 150 mensuales.

Nuestro problema consistió en implementar en el año 2008 una licitación para proveer conexiones de internet a 709 escuelas y otras instituciones públicas de la Ciudad de Buenos Aires. En este caso se adjudicará a ciertas empresas de telecomunicaciones que operan en la Ciudad la provisión del servicio a algún subconjunto de escuelas. Se trata de una subasta reversa pues el principal, el Gobierno de la Ciudad de Buenos Aires, se procura bienes comprados a los agentes.

También se la puede pensar como una subasta de *venta* afirmando que el Estado de la Ciudad vende a las empresas participantes un activo (la deuda que la Ciudad contrae con las empresas que proveerán los servicios de internet), y que las empresas pagan ese activo mediante descuentos, buenos precios por volumen grande de unidades, servicios de alta calidad.

Los ítems en subasta serán entonces 709 escuelas y los participantes o postores serán el conjunto { Cablevisión, Telefónica, Iplan, Metrotel, Telecom, Telmex } (ver Figuras 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, y 4.7, en las que se

Figura 4.1: Cablevisión

muestran las áreas de cobertura estimadas de cada empresa, información de la que disponíamos antes de diseñar el mecanismo de licitación).

Como primera observación afirmamos que en este tipo de problemas de aplicación no sirven las soluciones heurísticas ([4]). Estamos obligados a encontrar un óptimo pues al definir un ganador necesariamente se definen perdedores, y no quedaría claro cómo justificar qué participantes son perdedores como resultado de un método informal o heurístico.

En las siguientes subsecciones discutimos dos propuestas iniciales para el diseño de la licitación, una por la Agencia de Sistemas de Información y otra por nosotros, y finalmente exponemos el modelo acordado para la implementación. Este trabajo previo entra en el marco de la definición y diseño del mecanismo, que como dijimos más arriba es una parte difícil e ineludible del trabajo. No queda claro antes de la licitación qué sistemas son más convenientes, y quien debe poner en marcha la licitación necesita una imagen clara de cada propuesta, sobre todo de sus aspectos técnicos.

Figura 4.2: Telecom

Figura 4.3: Telefónica de Argentina

Figura 4.4: Telmex

Figura 4.5: Iplan

Figura 4.6: Metrotel

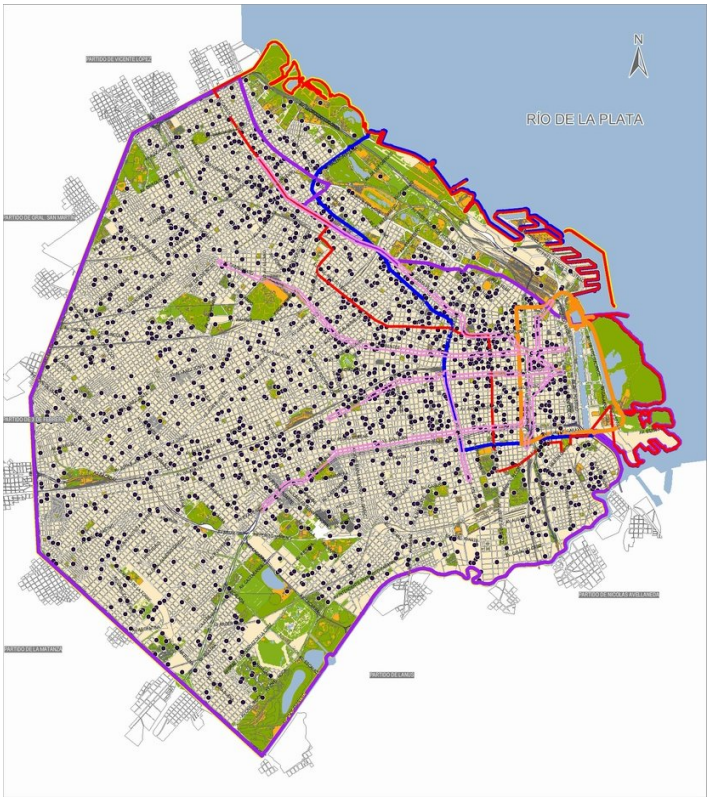

Figura 4.7: Todas las empresas

4.1.1. Propuesta de la Agencia de Sistemas de Información

La primera intención de la Agencia de Sistemas de Información era licitar las escuelas de manera tal que las pujas se hagan por institución. Es decir, cada participante propone 709 precios unitarios, uno por cada bien en subasta, y se realizan 709 subastas al mejor postor.

Algunas desventajas de esta propuesta son:

- En pimer lugar se pierde la posibilidad de tener **economías de escala**, es decir un descuento por paquetes grandes de escuelas asignadas a una misma empresa que tenga grandes coberturas.
- En segundo lugar este procedimiento es muy propenso a la **colusión**. Supongamos por ejemplo que una zona de tamaño considerable estuviera cubierta sólo por dos participantes (en nuestro caso las empresas Cablevisión y Telefónica son los únicos participantes presentes en una superficie de la ciudad mayor al 50 por ciento). El riesgo es, entonces, que, sabiendo que no tienen competencia en esas escuelas, podrían dividirse la región y poner un precio alto cada uno a su parte sabiendo que serían necesarios ganadores (ver Figura 4.7).
- Otra desventaja de este modelo es la posible distorsión de precios, que se volverían excesivamente altos donde hay poca competencia.
- Se hace muy engorroso para cada participante el presentar sus pujas. La Agencia de Sistemas de Información hubiera entregado a cada empresa una lista con los 709 nombres de las instituciones y sus respectivas direcciones postales, lo que hubiese conllevado a cada empresa un gasto significativo de tiempo y de recurso humano para la tarea de revisar cada dirección en el mapa y entoces determinar si la escuela se encuentra bajo el área de cobertura y si se tiene interés por ganar ese ítem.
- Hay zonas de cobertura en donde se intersecan más de dos empresas. Esto hubiese dado lugar a pujas interesantes por paquetes en una licitación por combinaciones.

4.1.2. Nuestra primera propuesta

La superficie de la Ciudad de Buenos Aires está dividida en distritos escolares (ver Figura 4.8). Se definió una partición en 11 regiones de la ciudad teniendo en cuenta tanto la partición existente en principio en distritos escolares como el área de cobertura de cada participante, esto último para evitar aumentos de precio en regiones cubiertas parcialmente por los participantes (ver Figura 4.9). Notar que es factible que una empresa preste su

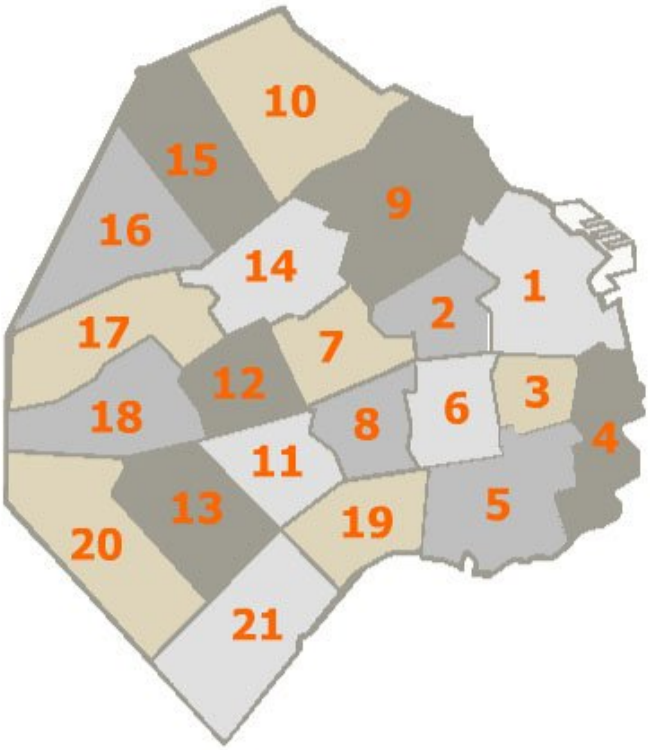

Figura 4.8: Distritos escolares

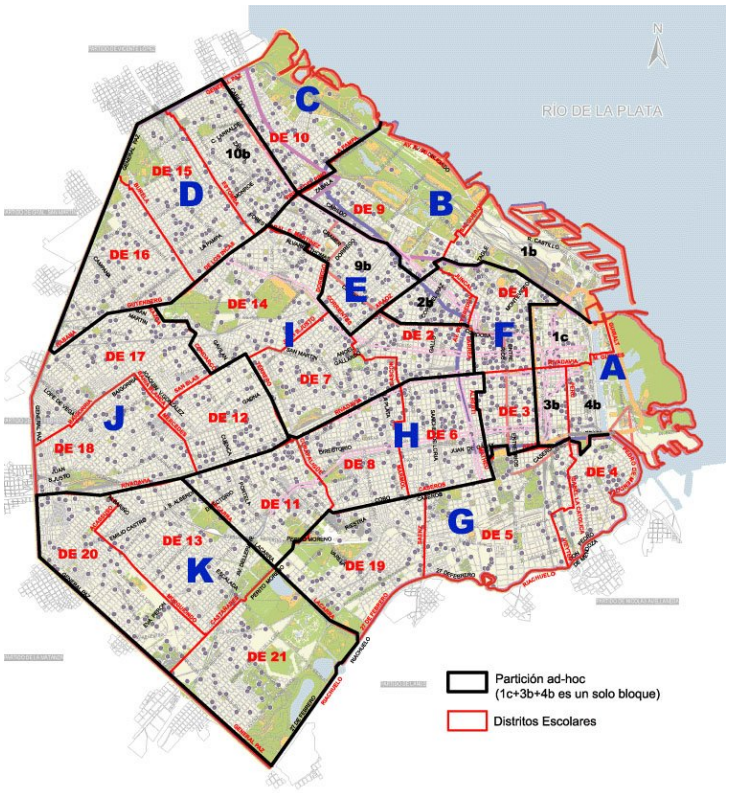

Figura 4.9: Partición de regiones en subasta

servicio a una escuela que no se encuentre en su área de cobertura, solamente que a un precio mayor.

Cada región contenía distritos escolares enteros, salvo en pocos casos en los que se dividieron algunos y se definieron *sub-distritos*.

Se propuso entonces una subasta por combinaciones en donde el conjunto de los ítems en subasta eran las 11 regiones diseñadas. Por supuesto que la idea era permitir pujar en combinaciones de regiones, bajo la condición de que los paquetes por los cuales se expresaran intenciones debían ser subconjuntos conexos (en el sentido de abstraer a la Ciudad de Buenos Aires como un subconjunto de $\mathbb{R}^2$). Esta condición se fijó para evitar que ciertos participantes pudieran forzar que se incluya en un mismo paquete regiones distantes del norte (en donde hay mucha competencia) y del sur de la Ciudad; veamos el siguiente ejemplo. Supongamos dos empresas con la condición de ser las únicas presentes en la mitad sur de la ciudad. Supongamos que forman cada una un paquete disconexo con una región en el extremo sur y otra región en el extremo norte. Entonces podrían ponerle precio alto al paquete y seguir siendo ganadoras por ausencia de competi-

dores en la zona sur. El resultado es que se les concedería, a un precio alto, regiones del extremo norte, en donde hay más competencia y en donde serían factibles mejores pujas, es decir precios más bajos.

Una desventaja de este modelo radica en la rigidez de la división previa en distritos escolares y consiste en que perjudica claramente a algunas empresas. Por ejemplo la empresa Metrotel, que opera en principio una manzana alrededor de las líneas de subterráneo, puede pujar por la intersección entre las líneas de subte y algún distrito. Como las regiones propuestas son indivisibles, Metrotel estaría obligada a ofrecer un precio alto por aquellas escuelas lejanas a la red de subterráneo que se encuentren en alguna región de su interés. Una empresa no está imposibilitada de ofrecer su servicio en un punto que no esté contenido en su cobertura *a priori*; simplemente sucede que el costo de brindar una conexión a un tal cliente sería excesivamente mayor que aquel para las escuelas presentes en su área de trabajo actual. Todo esto la llevaría a una derrota segura contra una puja de una empresa con cobertura transversal que pueda ofrecer un precio razonable por un paquete mayor en el sentido de la contención.

También cabe mencionar que la información del área de cobertura de que se disponía era una estimación de la Agencia, con la correspondiente probabilidad de error, y que este modelo tampoco disminuía sensiblemente la probabilidad de colusión.

4.1.3. Modelo definitivo implementado

En esta subsección se presenta el modelo definitivo, que es el que fue aceptado finalmente por la Agencia de Sistemas de Información, y con el cual se llamó a licitación a fines del año 2008.

Cada empresa presenta un único precio unitario de abono mensual, un listado de las escuelas por las que participa y los descuentos por cantidad de ítems. Más formalmente, cada participante expresa sus preferencias definiendo dos tablas de parámetros y una constante. La constante que define cada participante es denotada p y equivale al *precio único por una conexión de internet para una escuela cualquiera por mes*. Se supone que este es el precio que cobraría cada empresa si la Agencia contratara el servicio para una escuela en particular. En la primera tabla se marcan aquellas escuelas en las que el participante está interesado. El Cuadro 4.1 muestra un ejemplo de la información que hasta aquí debe presentar cada empresa. Las variables α_i son binarias, es decir que pueden valer sólo 0 o 1 y se interpreta que la empresa hace una propuesta (expresa un interés) por la escuela i si y sólo si α_i vale 1. Ahora definimos *tramo tarifario* como un elemento cualquiera del conjunto de intervalos de enteros $T :=$ {[0, 19], [20, 39], [40, 59], [60, 79], [80, 99], [100, 149], [150, 199], [200, 249], [250, 299], [300, 399], [400, 499], [500, 599], [600, 699], [700, 709]}.

En la segunda tabla, entonces, cada participante j especifica qué por-

Número de Escuela	Preferencia
1	α_1
2	α_2
3	α_3
...	...
...	...
...	...
...	...
707	α_{707}
708	α_{708}
709	α_{709}

Cuadro 4.1: Tabla de preferencias

Tramo Tarifario	Descuento
[0, 19]	d_1
[20, 39]	d_2
[40, 59]	d_3
...	...
$[a_k, b_k]$	d_k
...	...
[500, 599]	d_{12}
[600, 699]	d_{13}
[700, 709]	d_{14}

Cuadro 4.2: Descuentos por volumen

centaje de descuento aplica a p en función del tramo tarifario en que se encuentre la cantidad de escuelas que gane. Es decir, para cada participante se tiene que si la cantidad de escuelas que gana es un número del tramo tarifario $[a_k, b_k] \in T$, entoces el precio unitario que cobrará ese participante será $\tilde{p} = p\,(1 - d_k^j/100)$, en donde d_k^j es un número entre 0 y 100 y es precisamente el descuento que hace el participante j por ser adjudicatario de una cantidad de escuelas entre a_k y b_k (ver Cuadro 4.2).

Este modelo representa una mejora respecto de los otros propuestos ya que dificulta fuertemente la colusión, permite explotar economías de escala, no quita a participantes de menor cobertura la posibilidad de ser ganadores por la ausencia, finalmente, de paquetes de distritos rígidos que no necesariamente coincidan con la forma del área de cobertura de ciertas empresas. Como veremos más adelante, además, permitió ahorros significativos a la hora de hacer las adjudicaciones para contratar los servicios.

Dadas las pujas de todas las empresas definimos *regiones*, ahora a posteriori, en función de las intersecciones de las distintas preferencias. Si N es la cantidad de participantes, para cada j entre 1 y N denotamos por P_j al conjunto de escuelas en la puja de la empresa j. Una región será una intersección maximal de ítems, en el siguiente sentido: decimos que un conjunto de escuelas $R = \{e_i : i \in I\}$ es es una región si existen índices $n_1, ..., n_k$ tales que

i $P_{n_1} \cap ... \cap P_{n_k} = R$ y
ii $\forall n \neq n_1, ..., n_k \quad P_n \cap R = \emptyset$

Observemos que ahora las regiones pueden dividirse en dos o más ganadores. Nuestro modelo determina, para cada región, cuántos ganadores hay en ella y cuántas escuelas de ella se asigna a cada uno.

El objetivo es minimizar, sobre todas las combinaciones posibles de empresa-tramo tarifario, la suma del costo unitario, con el descuento correspondiente al tramo, multiplicado por la cantidad de escuelas que gana la empresa en ese tramo.

Se puede demostrar que esta es una manera válida de plantear la minimización del costo total por las conexiones sobre todas las posibles asignaciones de escuelas a empresas con la condición de que se sujete el problema a las siguientes restricciones:

1. Dada una región, la suma de la cantidad de escuelas de esa región que recibe cada empresa debe ser exactamente igual a la cantidad de escuelas que la conforman (restricción para cubrir toda la demanda de conexiones de internet).

2. Cada empresa cobrará un único precio unitario por escuela, que resulta de aplicar el descuento correspondiente al tramo tarifario en que se encuentre la cantidad total de escuelas que gane.

3. Una empresa no podrá ganar escuelas por las que no haya expresado preferencia.

En la siguiente sección veremos el modelo de programación lineal entera diseñado para encontrar la solución óptima de este problema con tiempos de ejecución eficientes.

4.2. Modelo de programación lineal entera

Aquí estudiaremos en detalle los parámetros, variables, las restricciones y la función objetivo usados en nuestro diseño.

4.2.1. Parámetros

Definimos

$R = \{R_1, R_2, ..., R_k\}$, las regiones definidas como intersecciones de escuelas.

$E = \{E_1, E_2, E_3, E_4, E_5, E_6\}$, las empresas participantes.

$T = \{[0, 19], [20, 39], ..., [500, 599], [600, 699], [700, 709]\}$, los tramos tarifarios, todos intervalos de enteros.

Para todo $1 \leq r \leq k$ definimos $esc(R_r)$ como la cantidad de escuelas en la región R_r.

Para todo $(i, t) \in E \times T$ definimos $c(i, t)$ como el costo unitario de una conexión de la empresa i aplicando el descuento del tramo t (costos por escuela por tramo).

Para cada $(i, j) \in E \times R$ definimos el parámetro $pres(i, j)$, $pres(i, j) = 1$ si y sólo si el participante i emitió una puja por las escuelas de la región j (presencia de empresa por área), y $pres(i, j) \in \{0, 1\}$.

Finalmente notamos, para cada $t \in T$, $\text{mín}(t)$ y $\text{máx}(t)$ al mínimo y máximo de cada tramo respectivamente.

4.2.2. Variables

Para cada $(i, j) \in E \times R$ se define $x_{ij} \in \mathbb{Z}_{\geq 0}$ como la cantidad de escuelas ganadas por la empresa i en la región j.

Dado $(i, t) \in E \times T$ definimos una variable binaria y_{it}, donde $y_{it} = 1$ si y sólo si la empresa i gana una cantidad de escuelas en el tramo t, $y_{it} \in \{0, 1\}$.

También para cada $(i, t) \in E \times T$ definimos $z_{it} \in \mathbb{Z}_{\geq 0}$ tal que z_{it} vale la cantidad de escuelas que gana la empresa i en el tramo t. Es decir que en nuestro caso vale que si la empresa i_0 gana una cantidad total de escuelas d_0 y d_0 pertenece al intervalo t_0, entonces

$$z_{i_0 t_0} = d_0$$

y

$$\forall t \neq t_0 \quad z_{it} = 0.$$

Las variables z_{it} se introducen para modelar apropiadamente los precios según la cantidad de escuelas asignadas a cada empresa, como una cuestión técnica del modelado por programación lineal entera.

4.2.3. Función objetivo y restricciones

El problema de programación lineal entera que queremos resolver es

$$\text{mín} \sum_{(i,t) \in E \times T} c(i, t)\, z_{it}$$

sujeto a las siguientes restricciones:

i $\forall j \in R \sum_{i \in E} x_{ij} = esc(j)$

ii $\forall i \in E \sum_{t \in T} y_{it} = 1$

iii $\forall (i,t) \in E \times T \sum_{j \in R} x_{ij} \geq \text{mín}(t) - 709\,(1 - y_{it})$

iv $\forall (i,t) \in E \times T \sum_{j \in R} x_{ij} \leq \text{máx}(t) + 709\,(1 - y_{it})$

v $\forall (i,t) \in E \times T\ z_{it} \geq \sum_{j \in R} x_{ij} - 709\,(1 - y_{it})$

vi $\forall (i,j) \in E \times R$ tal que $pres(i,j) = 0\ x_{ij} = 0$.

vii $\forall (i,j) \in E \times R\ x_{ij} \in \mathbb{Z}_{\geq 0}$.

viii $\forall (i,t) \in E \times T\ z_{it} \in \mathbb{Z}_{\geq 0}$.

ix $\forall (i,t) \in E \times T\ y_{it} \in \{0,1\}$.

Las restricciones se explican como sigue:

i Dada una región la suma de las escuelas en ella que se asignan a cada empresa debe igualar la cantidad total de escuelas en esa región.

ii Como las variables y_{it} son binarias, la condición sobre la suma de las variables es equivalente a imponer que haya un par (i_0, t_0) tal que $y_{i_0 t_0} = 1$ y para todos los demás $y_{it} = 0$. Entonces esta restricción dice que a cada empresa corresponde un único tramo tarifario, aquel en el que se encuentre la cantidad total de escuelas que gane.

iii Para cada elección posible (i,t) de emprea y tramo tarifario, si la empresa gana una cantidad de escuelas en ese tramo tarifario entonces la correspondiente variable y_{it} valdrá 1 y la desigualdad dice que la cantidad de escuelas ganada por la empresa i será mayor o igual que el mínimo del tramo t.

Para todos los otros tramos, aquellos en los que no cae la empresa i, la correspondiente variable y_{it} valdrá 0 y entonces la cantidad $\text{mín}(t) - 709\,(1 - y_{it})$ será menor o igual a 0.

iv Análogamente, para cada elección posible (i,t) de empresa y tramo tarifario, si la empresa gana una cantidad de escuelas en ese tramo tarifario entonces la correspondiente variable y_{it} valdrá 1 y la desigualdad dice que la cantidad de escuelas ganada por la empresa i será menor o igual que el máximo del tramo t.

Para todos los otros tramos, aquellos en los que no cae la empresa i, la correspondiente variable y_{it} valdrá 0 y entonces la cantidad $\text{mín}(t) - 709\,(1 - y_{it})$ será mayor o igual a 709, que es el total de instituciones en la licitación.

v Con estas desigualdades establecemos cierta dependencia entre las variables z_{it} y las variables x_{ij} e y_{it}. Para cada elección de empresa y tramo tarifario (i, t), si la empresa i cae en el tramo t entonces $y_{it} = 1$ y la desigualdad expresa que la cantidad de escuelas que gana la empresa i en el tramo t debe ser mayor o igual a la cantidad total de escuelas ganadas por i, lo que es correcto por la unicidad del tramo al que pertenece el número de ítems ganado por cada participante.

En los otros casos, y_{it} valdrá 0 y lo que dice la desigualdad es que la correspondiente z_{it} deberá ser mayor o igual a un número que es menor o igual a cero.

vi Por último esta restricción impone sencillamente que si la empresa i no expresó preferencia por ninguna escuela de la región j entonces no debe ganar ninguna escuela de esa región.

Finalmente queremos comentar el siguiente corte *por igualdad* agregado al modelo.

$$\forall i \in E \quad \sum_{t \in T} z_{it} = \sum_{j \in R} x_{ij}$$

En esta fórmula se cuenta el cardinal de un mismo conjunto de dos maneras distintas. Supongamos i fijo, una empresa en E. Del lado izquierdo tenemos la suma, sobre todos los tramos tarifarios t, de las escuelas que gana i en el tramo t. Según lo que dijimos anteriormente, en esta suma serán todos los términos iguales a cero, salvo a lo sumo uno que será igual a la cantidad de escuelas ganadas. Del lado derecho tenemos la suma, sobre todas las regiones j, de la cantidad de ítems que gana la empresa i en cada región j, lo que equivale nuevamente al total ganado por i.

Con esto queda clara la validez de la igualdad. Por supuesto que esta igualdad estaba ya implícita en las restricciones iniciales, pero este corte acelera significativamente la búsqueda del óptimo. En las pruebas realizadas en escenarios con datos ficticios, el modelo sin el corte agregado encontraba el óptimo en un tiempo comprendido entre dos y tres minutos, mientras que luego de agregar estas restricciones el programa resolvía el problema en décimas de segundo. Esto sucede pues por ser una igualdad permite despejar una variable en función de las demás, con la consecuencia de que la dimensión del espacio en el que se encuentren los puntos factibles de la relajación lineal asociada al modelo disminuye en 1.

4.2.4. Encontrando múltiples óptimos

Dada una solución óptima, en esta sección mostramos cómo se puede modificar el modelo para prohibir esta solución, con el objetivo de determinar si existen óptimos alternativos. Es de especial interés la búsqueda de

múltiples óptimos en un problema de aplicación como este, pues al definir ganadores necesariamente se definen perdedores.

Supongamos que encontramos un óptimo x^* y que sus coeficientes son los números x^*_{ij}, es decir que la asignación ganadora es:

$$\forall\,(i,j) \in E \times R \quad (x^*)_{ij} = x^*_{ij}$$

Ahora usaremos las coordenadas de x^* como parámetros de la siguiente manera. Quisiéramos forzar, en el programa, que al menos una coordenada de x sea distinta que la respectiva coordenada de x^*, y así, si el algoritmo llega a un nodo óptimo con igual valor en el funcional que para x^*, tendríamos que este último no sería el único óptimo de nuestro problema original.

Usando notación convencional de lógica proposicional, lo que queremos agregar al programa es la siguiente disyunción:

$$\Big(\bigvee_{(i,j)\,:\,x^*_{ij}>0} x_{ij} \le x^*_{ij} - 1\Big) \quad \vee \quad \Big(\bigvee_{(ij)\,\in\,E\times R} x_{ij} \ge x^*_{ij} + 1\Big)$$

Esto es lo mismo que escribir:

$$\Big(\bigvee_{(i,j)\,:\,x^*_{ij}>0} 709 - x_{ij} \ge 709 - (x^*_{ij} - 1)\Big)$$

$$\vee$$

$$\Big(\bigvee_{(ij)\,\in\,E\times R} x_{ij} \ge x^*_{ij} + 1\Big)$$

Pero el problema es agregar esta información en forma de desigualdades lineales. Para esto introducimos dos matrices de variables binarias

$$v,\, w \;\in\; \{0,1\}^{E\times R}$$

y agregamos las siguientes restricciones al programa lineal entero.

$$\forall (i,j) \;\in\; E \times R, \quad x^*_{ij} > 0: \quad 709 - x_{ij} \ge (709 - (x^*_{ij} - 1))\, w_{ij}$$

$$\forall (i,j) \;\in\; E \times R: \quad x_{ij} \ge (x^*_{ij} + 1)\, v_{ij}$$

$$\sum_{(i,j)\,:\,x^*_{ij}>0} w_{ij} \;+\; \sum_{E\times R} v_{ij} = 1$$

Estas restricciones agregadas aseguran que para una variable se cumpla la condición buscada y para todas las demás variables queden restricciones que son triviales dado que están todavía sujetas a las restricciones iniciales. Entonces por un lado podrían darse dos o más de las condiciones buscadas,

y por el otro no se pierde ninguna solución que no sea el x^* que queremos prohibir.

Ahora supongamos que restringimos n óptimos x_1^*, x_2^*, ..., x_n^* cada uno con su correspondiente conjunto de variables binarias w^1, w^2, ..., w^n y v^1, v^2, ..., v^n, entonces para cada lista de restricciones, una por cada óptimo, tenemos definido un conjunto $\{x_j^*\}^c$. Las restricciones de todos los óptimos, juntas, definen el conjunto

$$\{x_1^*\}^c \cap \{x_2^*\}^c \cap ... \cap \{x_n^*\}^c$$

que por las leyes de de Morgan es igual a

$$\{x_1^*, x_2^*, ..., x_n^*\}^c.$$

Entonces si hay algún punto factible que queda fuera de $\{x_1^*, x_2^*, ..., x_n^*\}^c$, necesariamente tiene que ser alguno de los óptimos que no queremos.

4.3. Pujas y resultados computacionales

Para implementar el modelo de programación lineal entera usamos el lenguaje **Zimpl**, de Thorsten Koch ([25], [26]) y lo resolvimos usando el paquete **Scip** ([1]). Con este *software* fue posible encontrar la solución óptima en menos de un segundo en distintos problemas de prueba como así también en el problema real luego de recibir las propuestas de las empresas.

El código del programa lineal en Zimpl puede verse en el anexo. Para ejecutar el programa fue usado un microprocesador AMD Athlon 64 X2 Dual Core 4400+ de 2,31 gigahertz en una computadora con un gigabyte de memoria RAM.

Si bien nuestro diseño contemplaba la presencia de las seis empresas enumeradas anteriormente, en la implementación para la licitación de conexiones de internet se recibieron pujas de sólo cuatro empresas proveedoras. En los Gráficos 4.10, 4.11, 4.12, 4.13 se incluye el número de escuelas por las que expresaron preferencia las empresas y se esbozan las zonas de la Ciudad en las que se encuentran.

En la Figura 4.14 se detallan las regiones que quedaron determinadas como intersección de preferencias con sus respectivas cantidades de escuelas.

Las empresas Telefónica y Telecom ofrecieron conexiones con cable de cobre mientras que Cabelvisión y Telmex ofrecieron servicios por fibra óptica.

Para incentivar el uso de mejores tecnologías se ponderaron los precios de Cablevisión y Telmex con el coeficiente 0,9 (se consideró sólo el 90 por ciento del precio que pedían estas empresas a cambio de una conexión por

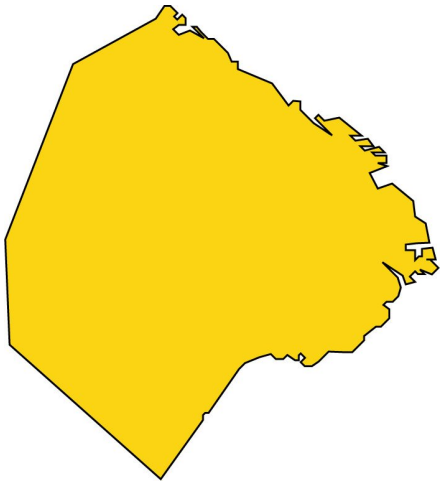

Figura 4.10: Cablevisión, 709 escuelas

fibra óptica). Esta ponderación estaba debidamente explicitada en los pliegos de la licitación y sólo se usó para determinar a los ganadores.

El primer resultado que se presenta es el de la licitación que incluye a las cuatro empresas mencionadas anteriormente. Posteriormente surgió una impugnación por parte de la Agencia de Sistemas de Información para la empresa Cablevisión y se corrió nuevamente el programa lineal para este nuevo contexto con tres empresas.

En el primer caso la empresa Cablevisión presenta un precio unitario alto e inverosímil, con la idea de hacer uso por un lado de su monopolio en la zona Sur (monopolio no real, sino referido al universo de las empresas que participaron de la licitación) y por otro lado de su condición de única empresa que cubre toda la superficie de la Ciudad. Otro posible factor presente en la puja de Cablevisión tal vez sea que quienes definieron la valuación sobre los tramos tarifarios hayan considerado que del lado del subastador sólo se evaluaría el número del porcentaje que se aplicaría como descuento a cada tramo. Y justamente el descuento que ofreció Cablevisión para el tramo tarifario $[700, 709]$ fue cercano al 80 %, alcanzando así un precio levemente superior que el mínimo de los precios ofrecidos sobre todos los tramos tarifarios y sobre todas las empresas.

Los resultados numéricos son los siguientes.

Escenario 1: Las cuatro empresas presentes.

- Cablevisión gana las **709** escuelas
- Costo de esta asignación: **$ 665.992,06**
- Costo promedio por escuela: **$ 939,34**

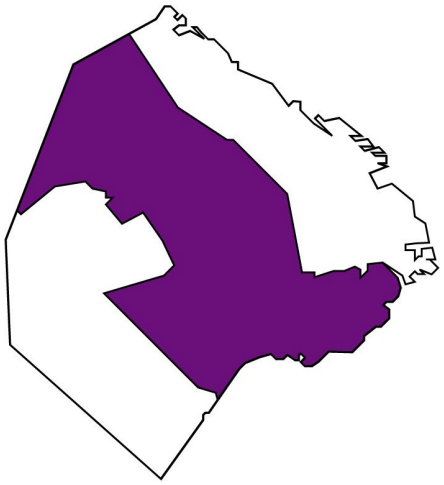

Figura 4.11: Telefónica, 348 escuelas

Ver Figura 4.15.

En el segundo caso, sin la empresa Cablevisión, se quitó la restricción de cobertura total de la Ciudad porque evidentemente quedaría un problema de Programación Lineal Entera sin puntos factibles.

El resultado es el siguiente.

Escenario 2: Sólo Telefónica, Telecom y Telmex.

- Se asignan **461** de las 709 escuelas.
- Asignación óptima:
  - 348 escuelas asignadas a Telefónica.
  - 99 escuelas asignadas a Telecom.
  - 14 escuelas asignadas a Telmex.
- Costo de esta asignación: **$ 382.691,85**
- Costo promedio por escuela: **$ 830,13**

Ver Figura 4.16.

A continuación hacemos un análisis del Escenario 1. Tomando el precio final de $ 665.992,06 se puede calcular el beneficio de usar nuestro diseño conjeturando el probable comportamiento de cada una de las empresas en una licitación hecha según la idea original de la Agencia de Sistemas de Información, de a una escuela por vez. Además hay que observar que si bien

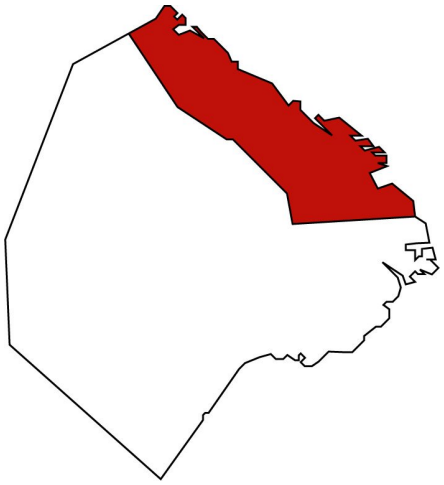

Figura 4.12: Telecom, 99 escuelas

todas las escuelas fueron adjudicadas a una sola empresa (por la restricción de cobertura), la asignación se hizo a un precio menor del que hubiera puesto esta empresa ganadora en los otros diseños.

Suponiendo que Cablevisión ofrece su peor precio ($ 4.696,7) por las escuelas en las que no tenía competencia y que en el resto de las escuelas cada uno pone su mejor precio, el costo de subastar las escuelas de a una habría sido **$ 1.539.330,21**. Esta conjetura es sostenible pues dado que Cablevisión hizo uso de la ausencia de competencia en gran parte de la Ciudad para ganar el total de los ítems a un precio más alto que el mejor ofrecido sobre todas las empresas y todos lo tramos tarifarios, bien podría haber hecho uso de esta ventaja cobrando muy caras las escuelas en las que no tiene competencia.

Inmediatamente observamos un ahorro de $ 1.539.330,21 - $ 665.992,06 = **$ 873.338,15** que es un 31 % más alto que el precio pagado por la asignación completa de conexiones,

$$\$\ 665.992{,}06 = \$\ 873.338{,}15 - \textbf{\$ 207.346,09}.$$

Ahora suponemos en forma muy optimista que en los renglones en que sólo puede proponer un precio Cablevisión, esta empresa no pone su peor precio. Tampoco sería muy ajustado a la realidad pensar en que este participante no haga ninguna especulación con el hecho de tener exclusividad en el 35 % de las escuelas, es decir, no podemos pensar que fuera a proponer su mejor precio por todo este paquete de ítems. Entonces suponemos que propone se **segundo** mejor precio, que es el 40 % de su precio más alto. Además continuamos bajo la hipótesis de que para el resto de los ítems cada uno pone su **mejor** precio. En este caso habríamos llegado a un costo

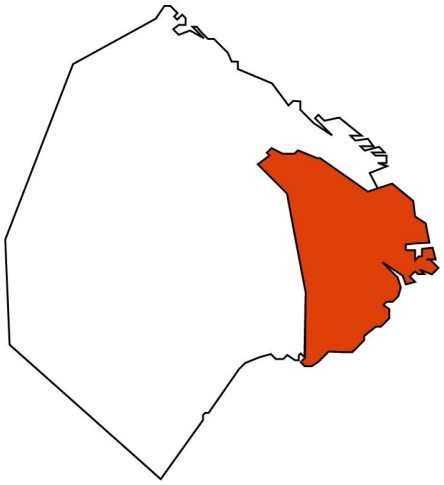

Figura 4.13: Telmex, 97 escuelas

de **$ 840.461,25**.

Inmediatamente calculamos un ahorro de $ 840.461,25 - $ 665.992,06 = **$ 174.469,19**, equivalente al monto necesario para pagar **186** conexiones a nuestro precio óptimo encontrado en el primer escenario, número de escuelas que representa más del 26 % del total en disputa.

Continuamos ahora en forma aún más optimista. Bajo la hipótesis de que en todos los renglones con competencia exista algún participante que ofrece el **mejor precio registrado** en toda la licitación ($ 785,29), y que nuevamente Cablevisión ofrezca su **segundo mejor** precio en donde tiene exclusividad, el costo sería **$ 827.931,33**.

En este caso el ahorro suma $ 827.931,33 - $ 665.992,06 = **$ 161.939,27**, equivalente al monto necesario para conectar a más del 24 % de las escuelas.

Finalmente la impugnación fue desestimada por lo que el Escenario 1 fue el aplicado.

La aplicación fue exitosa pues el diseño permitió plantear correctamente la idea que se tenía para la licitación y además porque el modelo permitió resolver el problema computacional rápidamente cuando llegaron los pliegos con las propuestas de las empresas y adicionalmente trajo consigo un ahorro significativo para el Estado con respecto al modelo de licitación planteado originalmente.

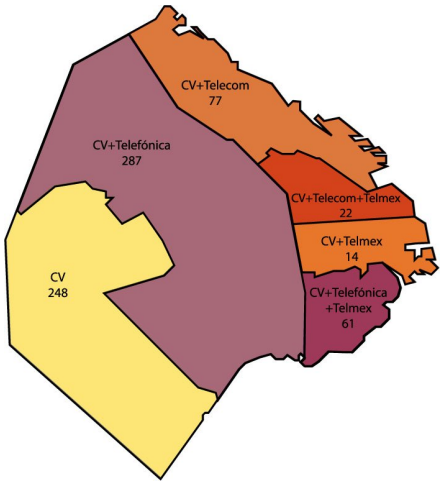

Figura 4.14: Regiones determinadas por las preferencias

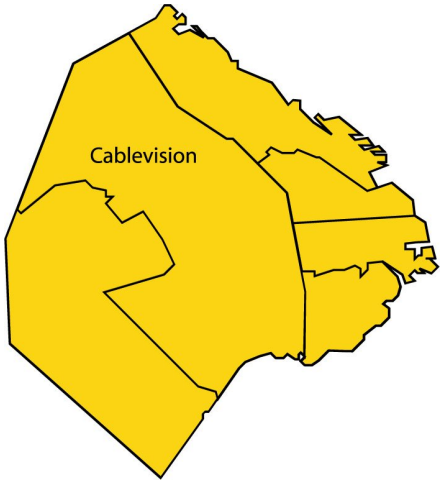

Figura 4.15: Asignación ganadora del Escenario 1

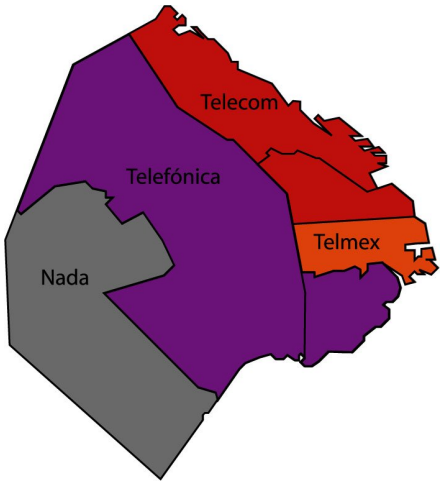

Figura 4.16: Asignación ganadora del Escenario 2

Capítulo 5

Conclusiones

En este trabajo hicimos una implementación de un modelo de programación lineal entera original y diseñado por nosotros para resolver el problema de una licitación de procuración pública de bienes por parte del Estado.

Propusimos más de un modelo de mecanismo al principal de la licitación, en este caso la Agencia de Sistemas de Información del Gobierno de la Ciudad de Buenos Aires, y después de discutirlos llegamos al diseño que se presenta en este trabajo.

Entre las razones por las cuales no se implementan muy frecuentemente las Subastas por Combinaciones en el caso general, se puede destacar el hecho de que no sólo es la determinación de los ganadores un problema muy duro de computar para el subastador, sino que también para el jugador lo es la mera definición de las valuaciones, para establecer su propuesta, cuando el número de ítems en subasta es grande y los ítems son diversos, y además no se puede presuponer que un participante tenga conocimientos de programación matemática ni de diseño de mecanismos.

Como se vió en la Sección 4.3 hubo ahorros muy significativos gracias al uso de nuestro diseño, aun sin disponer de la posibilidad de definir todos los aspectos del mecanismo implementado.

Como características distintivas de nuestro mecanismo de licitación citamos el *precio único* y el *descuento por volumen*. Por estas características distintivas y por otras de la instancia particular, afirmamos que nuestro diseño podría usarse en cualquier licitación en la cual se presenten descuentos sobre un precio unitario por cantidad de ítems, y otras cualidades destacables como la posibilidad de economías de escala por parte de participantes de mayor envergadura y la presencia de *áreas de cobertura* de ítems con intersección no vacía, pudiéndose aprovechar esto último para que la competencia sobre un mismo paquete de ítems haga bajar su precio en una licitación de procuración. Con todo esto queda pendiente la pregunta interesante acerca de en qué otra licitación podría usarse nuestro diseño.

También en la Sección 4.3 comentamos que nos servimos del lenguaje Zimpl (ver [25]), de Thorsten Koch, para el cómputo de la asignación ganadora. Este lenguaje se cuenta entre las mejores opciones de software libre para estos propósitos, la resolución de programas lineales. Y junto con nuestro diseño inteligente de los cortes incluidos en las restricciones pudimos llegar a la solución óptima en tiempos significativamente cortos.

Apéndice A

Algunos resultados sobre subastas

Definición A.1. *Dado un jugador j su función de valuación se denota v_j y, para un subconjunto de bienes S, definimos $v_j(S)$ como lo máximo que j está dispuesto a pagar por S. Si j gana S y paga p por él, entonces decimos que el beneficio neto que recibe j (*net payoff*) es $v_j(S) - p$.*

Definición A.2. *En un juego se dice que una estrategia X es dominante para un jugador j si, sin importar la estrategia que elijan los otros jugadores, la estrategia X resulta en un pago óptimo para j.*

Teorema A.1. *(Vickrey 1961) En una subasta de segundo precio, para todo jugador j es una estrategia dominante el pujar su verdadera valuación v_j. Es decir, si b_j es la puja de j, entonces sea como pujen los otros jugadores, es óptimo para j establecer $b_j = v_j$. Más aun, la subasta de segundo precio es eficiente.*

Demostración. Supongamos que un jugador j establece $b_j < v_j$. El resultado se ve afectado para j sólo si la máxima puja b establecida por los otros jugadores cumple

$$b_j < b < v_j.$$

En este caso el jugador j pierde (con un beneficio neto nulo, pues no tiene el objeto ni paga nada). En cambio, ganaría si pujara $b_j = v_j$. En este último caso, bajo la hipótesis del segundo precio y suponiendo todavía que la mejor puja sobre todos los otros jugadores es $b < v_j$, el *payoff* neto para j resulta $v_j - b > 0$.

Si ahora j establece $b_j > v_j$ entonces, si la mejor puja enviada por otro jugador es b y cumple

$$v_j < b_j < b,$$

entonces j pierde, y recibe un beneficio neto nulo. Si en cambio $b < b_j$, entonces j gana pero recibe un beneficio neto de $v_j - b$, que es negativo.

Finalmente, y este es el mejor caso para j, si $b < b_j$ pero b_j se define como igual a v_j, entonces j gana el bien y finaliza con un *payoff* neto de $b_j - b > 0$.

Concluimos que definir la puja como igual a la valuación (*truthful bidding*) es una estrategia dominante para cualquier jugador.

Habiendo probado que es óptimo para los participantes el pujar verazmente, entonces dado que el ganador es aquel que envía una puja maximal, resulta que la subasta de segundo precio es eficiente. □

La generalización del mecanismo de Vickrey para una subasta multiunitaria hecha por Clarke y Grooves se describe formalmente como sigue. Sea $\bar{x}$ un vector de cantidades enteras de bienes en subasta (hay $\bar{x}_k$ unidades del bien k). Dado un jugador $j \in \{1, ..., N\}$, $v_j(x^j)$ es la valuación que hace j del vector no negativo (coordenada a coordenada) x^j, y denotamos por $\hat{v}_j$ a la función de valuación que j reporta al subastador.

El subastador computa una asignación óptima

$$x^* = \text{argmax}_{x^1 \ldots x^N} \sum_{j=1}^{N} \hat{v}_j(x^j)$$

sujeta a

$$\sum_{j=1}^{N} x^j \leq \bar{x}$$

donde la restricción es una desigualdad coordenada a coordenada (x^* es una secuencia de N vectores).

El precio pagado por un jugador j se define como

$$p_j = \alpha_j - \sum_{m \neq j} \hat{v}_m(x^*_m)$$

$$\alpha_j = \text{máx} \left\{ \sum_{m \neq j} \hat{v}_m(x_m) \ : \ \sum_{m \neq j} x_m \leq \bar{x} \right\}.$$

Observar que α_j sólo depende de lo que reportan (a lo sumo) todos menos j.

Por ejemplo, supongamos dos ítems en venta, A y B, y dos postores $j = 1, 2$. Cada uno envía como pujas las valuaciones $\hat{v}_j(A)$ por A, $\hat{v}_j(B)$ por B y $\hat{v}_j(AB)$ por ambos. Supongamos también

$$\hat{v}_1(AB) \geq \hat{v}_2(AB)$$

y

$$\hat{v}_1(A) + \hat{v}_2(B) \geq \hat{v}_1(B) + \hat{v}_2(A)$$

(por simetría, este caso es general). Para ilustrar el mecanismo, ahora sí, pongamos

$$\hat{v}_1(AB) < \hat{v}_1(A) + \hat{v}_2(B).$$

El resultado es que el jugador 1 gana A y el 2 gana B. El jugador 1 paga $\hat{v}_2(AB) - \hat{v}_2(B)$ y el jugador 2 paga $\hat{v}_1(AB) - \hat{v}_1(A)$.

Ahora estamos en condiciones de probar que en la extensión VCG, *truthful bidding* es también una estrategia dominante.

Teorema A.2. *En el mecanismo VCG, pujar verazmente (ie. $b_j := v_j$) es una estrategia dominante para cada jugador j.*

Demostración. Consideremos un jugador j y una colección de valuaciones $\{\hat{v}_m\}_{m\neq j}$ reportadas por los jugadores salvo j. Si j comunica una valuación cualquiera $\hat{v}_j$ entonces denotamos $\hat{x}^j$ al correspondiente vector asignación y por $\hat{p}_j$ al pago que efectúa j. Cuando j puja verazmente denotamos x^{j*} al vector asignación y por p_j^* al pago.

Supongamos que j reporta una valuación $\hat{v}_j$. Si su valuación verdadera es v_j, resulta que su beneficio es

$$v_j(\hat{x}^j) - \hat{p}_j =$$

$$= v_j(\hat{x}^j) + \sum_{m\neq j} \hat{v}_m(\hat{x}^m) - \alpha_j$$

$$\leq \text{máx}\{v_j(x^j) + \sum_{m\neq j} \hat{v}_m(x^m) \ : \ \sum_{m=1}^{N} x^m \leq \bar{x}\} - \alpha_j$$

(se supone que una valuación veraz es mayor o igual que una falsa). Pero la última expresión es igual a

$$v_j(x^{j*}) - p_j^*,$$

con lo que, nuevamente, establecer como puja la valuación verdadera es óptimo para j. □

La clave de la propiedad de tener estrategia dominante que verifica la subasta de segundo precio es el hecho de que el pago que efectúa un postor ganador no depende de su puja. A continuación probaremos que, bajo no muy restrictivas hipótesis, la subasta de segundo precio es la *única* subasta eficiente con esta propiedad, a menos de una expresión sumada al pago de cada jugador j, que *no* depende de b_j.

Teorema A.3. *Supongamos que para todo j, v_j pueda tomar cualquier valor en $[0, 1]$. Entonces una subasta es eficiente y pujar verazmente es débilmente dominante si y sólo si valen (a) el postor más alto gana y (b) para todo jugador j el pago p_j satisface*

$$p_j = \begin{cases} \text{máx}_{i\neq j}\, b_i + t_j(b_{-j}), & \textit{si } j \textit{ gana;} \\ t_j(b_{-j}), & \textit{si } j \textit{ pierde.} \end{cases}$$

para cierta función t_j, donde b_{-j} es el vector de pujas con b_j excluida.

Demostración. Tomemos una subasta eficiente en la que pujar verazmente sea dominante. Entonces el postor más alto gana y tenemos (a).

Sea ahora $t_j^L(b_j, b_{-j})$ el pago de j si es que perdió y las pujas fueron (b_j, b_{-j}) (pueden tener que pagar aún los perdedores). Si dadas $b_j^{'}$, $b_j^{''} \leq \text{máx}_{i \neq j}\, b_i$ resulta

$$t_j^L(b_j^{'}, b_{-j}) > t_j^L(b_j^{''}, b_{-j})$$

entonces j resulta mejor pujando $b_j^{''}$ (pues pagaría menos) aun cuando $v_j = b_j{'}$. Esto contradice la propiedad de existencia de estrategia dominante. Luego, podemos escribir $t_j^L(b_j, b_{-j})$ como

$$t_j^L(b_j, b_{-j}) = t_j(b_j). \tag{A.1}$$

Análogamente para el pago de j si gana escribimos (usamos L por *lose* y W por *win*)

$$t_j^W(b_j, b_{-j}) = \hat{t}_j(b_j). \tag{A.2}$$

Ahora, si $v_j = \text{máx}_{i \neq j}\, b_i$, si j gana o pierde ambas cosas son eficientes, entonces como *truthful bidding* es dominante, el jugador j debe ser indiferente a ambos (recordemos que un jugador tiene una valuacion v_j del bien que compraría, y paga un precio p y su *payoff* es $v_j - p$, con lo que, si $v_j = p$, j es indiferente a comprar el bien o no pues en ningún caso ganaría nada). De A.1 y A.2 igualamos el *payoff*:

$$\max_{i \neq j} b_i - \hat{t}_j(b_{-j}) = -t_j(b_{-j}).$$

Es decir

$$\hat{t}_j(b_{-j}) = \max_{i \neq j} b_i + t_j(b_{-j}),$$

con lo que tenemos (b). Recíprocamente, si valen (a) y (b) la subasta es eficiente y del Teorema A.1 sigue que *truthful bidding* es dominante. □

Definición A.3. *Decimos que una subasta es de pagos balanceados si*

$$\sum_{j=1}^{N} p_j = 0.$$

Teorema A.4. *Supongamos que $\{v_j\}$ sean aleatorias e independientes cada una de una distribución con función de distribución acumulada F_j y con soporte en $[0, 1]$. Entonces existe una subasta eficiente y de pagos balanceados en la que pujar verazmente constituye un equilibrio Bayesiano.*

Demostración. Para la demostración asumimos $N = 2$. En una subasta en la que el postor más alto gana y el postor 1 apuesta b_1 y paga $P_1(b_1)$, este último elegirá b_1 para maximizar

$$\int_0^{b_1} v_1 dF_2(x) - P_1(b_1),$$

si es que el jugador 2 puja con veracidad. Derivando e igualando a cero nos queda

$$b_1 F_2^{'}(b_1) = P_1^{'}(b_1).$$

Entonces si definimos

$$P_1(b_1) = \int_0^{b_1} x F_2^{'}(x) dx,$$

la mejor respuesta que puede hacer 1 a 2 es pujar verazmente, pues la condición buscada vale para $b_1 = v_1$, y también lo hace la condición $F_2^{'}(v_1) > 0$. Similarmente, la veracidad es la mejor respuesta para 2 si su función de pago es

$$P_2(b_2) = \int_0^{b_2} x F_1^{'}(x) dx.$$

Finalmente definimos como funciones de pago respectivas a

$$p_1(b_1, b_2) = P_1(b_1) - P_2(b_2)$$

$$p_2(b_1, b_2) = P_2(b_2) - P_1(b_1).$$

Ahora es inmediato que $\sum p_j = 0$ y que la veracidad sigue siendo un equilibrio (esto último porque restar $P_2(b_2)$ al pago de 1 no afecta sus incentivos y similarmente para 2). □

Teorema A.5. *Bajo las hipótesis del Teorema A.4, dadas dos subastas tales que, en equilibrio Bayesiano, (a) para todo j y v_j, la probabilidad de ganar para un jugador j con valuación v_j es la misma en ambas subastas, y (b) para todo j, el monto que paga el jugador j con valuación 0 es la misma en ambas subastas, entonces, para todo j y v_j, el* pago esperado de equilibrio *para el jugador j con valuación v_j es el* mismo *en ambas subastas.*

Demostración. Tomemos una de las dos subastas y sea $(b_1(v_1), ..., b_n(v_n))$ un vector de pujas de equilibrio Bayesiano hechas por los jugadores cuando las valuaciones son $(v_1, ..., v_n)$. El jugador j tiene la decisión de comportarse como si su valuación fuese $\hat{v}_j$ cuando en realidad es v_j. Queda instanciado el problema de optimización

$$\max_{\hat{v}_j} \left[G_j(\hat{v}_j) v_j - P_j(\hat{v}_j) \right], \tag{A.3}$$

donde

$$G_j(\hat{v}_j) = \text{probabilidad de que gane el jugador } j \tag{A.4}$$

y

$$P_j(\hat{v}_j) = \text{pago esperado del jugador } j \tag{A.5}$$

siempre que j puje $b_j(\hat{v_j})$ y cada uno de los otros jugadores i puje de acuerdo a $b_i(\cdot)$, la función de puja de equilibrio. Por definición de equilibrio, la $\hat{v}_j$ que maximiza A.3 es $\hat{v}_j = v_j$, y entonces obtenemos la condición

$$P_j^{'}(v_j) = G_j^{'}(v_j)v_j \text{ para todo } j. \tag{A.6}$$

Integrando por partes en A.6 queda

$$P_j(v_j) = v_j G_j(v_j) - \int_0^{v_j} G_j(x)dx + k_j, \tag{A.7}$$

donde k_j es una constante de integración. De A.7 se observa que el pago esperado de j es k_j si $v_j = 0$. Por la hipótesis (b) esto es verdadero también para la otra subasta. Más aun, por la hipótesis (a), la probabilidad que tiene j de ganar en la otra subasta es $G_j(v_j)$ para toda v_j. Por esto, de A.7 se tiene que el pago esperado del jugador j es $P_j(v_j)$ y su beneficio neto esperado es $G_j(v_j)v_j - P_j(v_j)$ en ambas subastas. □

Notemos que en una subasta de segundo precio, por el Teorema A.1, los jugadores pujan con veracidad. De manera que la probabilidad que tiene un jugador de ganar es simplemente la probabilidad de que todos los otros jugadores tengan valuaciones menores. Por definición, la probabilidad de que un jugador i tenga una valuación no mayor que j es $F_i(v_j)$. Por independencia, entonces, en una subasta de segundo precio nos queda

$$G_j(v_j) = \prod_{i \neq j} F_i(v_j). \tag{A.8}$$

Más aun, en una tal subasta vale

$$P_j(0) = 0. \tag{A.9}$$

Pero A.8 y A.9 también valen para la Subasta Inglesa. Para ver esto basta notar que en la inglesa, un postor seguirá pujando más alto hasta que el presente precio alcance su valuación, y entoces el jugador de más alta valuación gana, con lo que vale A.8. Hemos demostrado el

Teorema A.6. *(Vickrey 1961) La Subasta de Segundo Precio y la Subasta Inglesa resultan en un* payoff *equivalente para todos los jugadores.*

Antes de extender el resultado de equivalencia definimos un tipo de subasta llamada *high bid*.

Definición A.4. *La subasta* high bid *es el mecanismo de subasta en el cual*

1. *los postores establecen pujas como números reales no negativos, a sobre cerrado,*
2. *el ganador es el jugador que haya enviado la puja más alta y*
3. *el ganador paga el monto de su puja.*

Teorema A.7. *(Vickrey 1961) Cuando las valuaciones $\{v_i\}$ son tomadas aleatoriamente de manera independiente de una* misma *distribución con función de distribución acumulada F y con soporte en $[0, 1]$ (esto es, $F_1 = ... = F_N$, hipótesis que llamaremos* de simetría*) entonces la subasta* high bid*, la subasta de segundo precio, la subasta inglesa, la subasta holandesa y la subasta* todos pagan *son todas equivalentes en términos del* payoff*.*

Demostración. El mecanismo de la subasta holandesa es formalmente el mismo que el de la *high-bid auction*, pues el precio que un postor está de acuerdo en pagar en la holandesa es el mismo que la puja que establecería en la *high-bid*.

Consideremos un equilibrio simétrico $b(\cdot)$ de la subasta *high-bid*, esto es, $b(v)$ es la puja que haría cualquier postor con valuación v (existe un equilibrio simétrico por la hipótesis de simetría acerca de los jugadores). Se puede probar que $b(\cdot)$ debe ser no decreciente. Supongamos que no es estrictamente creciente, es decir supongamos que existe un intervalo $[v^*, v^{**}]$ tal que para todo $v \in [v^*, v^{**}]$, $b(v) = b^*$. Tenemos $v^* - b^* \geq 0$ porque gracias al intervalo en el que $b(\cdot)$ vale constantemente b^*, una puja de valor b^* gana con probabilidad positiva (y por esto si $v^* - b^* < 0$, el jugador tendría un beneficio negativo). Entonces llegamos a

$$v^{**} - b^* > 0. \tag{A.10}$$

Pero si un jugador con precio de reserva v^{**} puja b^*, empata en la puja más alta con probabilidad positiva. Entonces si aumenta levemente su puja, aumenta con un salto su probabilidad de ganar (pues los empates ahora tendrán probabilidad 0), lo que vale la pena hacer en vista de A.10. Concluimos que $b(\cdot)$ debe ser *estrictamente* creciente, lo que significa que el jugador de más alta valuación siempre gana. Luego, el Teorema A.5 implica que la subasta *high-bid* es equivalente a la subasta de segundo precio. El mismo argumento se puede aplicar a la subasta *todos pagan*. □

El siguiente importante teorema establece la interesante propiedad de que cualquiera de las subastas mencionadas en el Teorema A.7, modificadas poniendo un precio de reserva (es decir, que el subastador no esté dispuesto a vender ningún bien por menos de un valor v^*), puede ser utilizada para maximizar el ingreso del subastador.

Teorema A.8. *Asumimos las hipótesis del Teorema A.7 y suponemos que*

$$\forall v \; J^{'}(v) > 0, \tag{A.11}$$

donde

$$J(v) = v - \frac{1 - F(v)}{F^{'}(v)}.$$

Entonces cualquiera de las subastas del Teorema A.7 maximiza el ingreso esperado del subastador, siempre que el subastador defina un precio de reserva v^* *tal que* $J(v^*) = 0$.

Demostración. Dada la *simetría* a priori de los jugadores, la igualdad A.7 implica que, para toda subasta simétrica, el ingreso esperado del subastador es N veces el ingreso que recogería de cada uno (recordar que las distribuciones acumuladas ahora son todas la misma F), es decir

$$N \int_0^1 \left[vG(v) - \int_0^v G(x)dx + k \right] dF(v). \tag{A.12}$$

Por la fórmula de integración por partes esta expresión es igual a

$$N \int_0^1 \left[\int_0^v G^{'}(x)xdx \right] dF(v) + k.$$

Ahora continuamos el cálculo de la integral, que es igual a

$$\int_0^1 \left[\int_0^v G^{'}(x)xdx \right] F^{'}(v)dv =$$

$$F(1) \int_0^1 G^{'}(x)xdx - \int_0^1 G^{'}(v)vF(v)dv =$$

$$\int_0^1 G^{'}(v)vdv - F(1)G(1) + \int_0^1 \left[G(v)F^{'}(v)v + F(v)G(v) \right] dv =$$

$$\int_0^1 G^{'}(v)vdv - 1 + \int_0^1 \left[G(v)F^{'}(v)v + F(v)G(v) \right] dv =$$

$$G(1) - 1 - \int_0^1 G(v)dv + \int_0^1 \left[G(v)F^{'}(v)v + F(v)G(v) \right] dv =$$

$$\int_0^1 \left[G(v)F^{'}(v)v - G(v)(1 - F(v)) \right] dv.$$

Por otro lado, usando la definición de J resulta que

$$\int_0^1 J(v)G(v)dF(v) =$$

$$\int_0^1 \left(v - \frac{1 - F(v)}{F^{'}(v)} \right) G(v) F^{'}(v) dv =$$
$$\int_0^1 \left[G(v) F^{'}(v) v - G(v)(1 - F(v)) \right] dv.$$

Llegamos a que la expresión A.12 puede ser reescrita como

$$N \int_0^1 J(v) G(v) dF(v) + k. \tag{A.13}$$

Ya hemos dicho que se puede probar que $G(v)$ debe satisfacer $G^{'}(v) \geq 0$ y

$$0 \leq G(v) \leq 1. \tag{A.14}$$

En Matthews (1984) se demuestra que G también satisface

$$\forall v \int_v^1 \left[(F(x))^{N-1} - G(x) \right] dF(x) \geq 0. \tag{A.15}$$

Consideramos ahora el problema de maximizar A.13 sujeto a A.14 y A.15. Notar de A.3 y A.7 que $k \leq 0$, de otra manera un postor con valuación cero tendría un beneficio esperado negativo, que es imposible, pues siempre se tiene la opción de no participar. Luego la opción de k maximizante es $k = 0$, es decir

$$P(0) = \text{pago realizado por un jugador con valuación cero} =$$
$$0 - \int_{\{0\}} G(x) dx + 0 = 0. \tag{A.16}$$

De las expresiones A.11, A.13, A.14 y A.15, la elección óptima de $G(v)$ es

$$G(v) = \begin{cases} (F(v))^{N-1}, & \text{si } v > v^*; \\ 0, & \text{si } v \leq v^*, \end{cases} \tag{A.17}$$

donde $J(v^*) = 0$. Pero todas las subastas del Teorema A.7, modificadas por un precio de reserva con valor v^* satisfacen A.16 y A.17. Satisfacen A.16 porque el pago realizado por un jugador con valuación cero será nulo, y satisfacen A.17 pues, por una lado, en cualquier subasta la probabilidad de ganar es 0 si la valuación es no mayor que el precio de reserva y, por otro lado, dadas las hipótesis de simetría e independencia, la probabilidad de ganar teniendo una valuación mayor que el precio de reserva es la probabilidad de que los $N - 1$ restantes jugadores tengan una valuación menor, que para cada jugador j es exactamente

$$\prod_{i \neq j} F(v) = (F(v))^{N-1}.$$

De esta manera todas las subastas del Teorema A.7 son solución del problema de maximización del ingreso esperado del subastador. □

Apéndice B

Código Zimpl

Parámetros

```
set R := {
  "A",
  "B",
  "C",
  "D",
  "E",
  "F"
};

set C := {
  "cv",
  "ta",
  "tc",
  "tm"
};

set T := {
  "0 - 19",
  "20 - 39",
  "40 - 59",
  "60 - 79",
  "80 - 99",
  "100 - 149",
  "150 - 199",
  "200 - 299",
  "300 - 399",
  "400 - 499",
  "500 - 599",
```

```
29    "600 - 699",
30    "700 +"
31  };
32
33  param escuelas[R] :=  <"A"> 248,
34                        <"B"> 14,
35                        <"C"> 77,
36                        <"D"> 22,
37                        <"E"> 287,
38                        <"F"> 61;
39
40  param costo[T*C]:=|"cv","ta","tc","tm"|
41        |   "0 - 19"| c11, c12, c13, c14|
42        |  "20 - 39"| c21, c22, c23, c24|
43        |  "40 - 59"| c31, c32, c33, c34|
44        |  "60 - 79"| c41, c42, c43, c44|
45        |  "80 - 99"| c51, c52, c53, c54|
46        |"100 - 149"| c61, c62, c63, c64|
47        |"150 - 199"| c71, c72, c73, c74|
48        |"200 - 299"| c81, c82, c83, c84|
49        |"300 - 399"| c91, c92, c93, c94|
50        |"400 - 499"|c101,c102,c103,c104|
51        |"500 - 599"|c111,c112,c113,c114|
52        |"600 - 699"|c121,c122,c123,c124|
53        |"700 +"    |c131,c132,c133,c134|;
54
55  param presencia[R*C]:=|"cv","ta","tc","tm"|
56                        |"A"|   1,   0,   0,   0|
57                        |"B"|   1,   0,   0,   1|
58                        |"C"|   1,   0,   1,   0|
59                        |"D"|   1,   0,   1,   1|
60                        |"E"|   1,   1,   0,   0|
61                        |"F"|   1,   1,   0,   1|;
62
63  param minimo[T] :=  <"0 - 19"> 0,
64                      <"20 - 39"> 20,
65                      <"40 - 59"> 40,
66                      <"60 - 79"> 60,
67                      <"80 - 99"> 80,
68                      <"100 - 149"> 100,
69                      <"150 - 199"> 150,
70                      <"200 - 299"> 200,
71                      <"300 - 399"> 300,
72                      <"400 - 499"> 400,
```

```
 73                       <"500 - 599"> 500,
 74                       <"600 - 699"> 600,
 75                       <"700 +"> 700;
 76
 77  param maximo_tr[T] := <"0 - 19"> 19,
 78                          <"20 - 39"> 39,
 79                          <"40 - 59"> 59,
 80                          <"60 - 79"> 79,
 81                          <"80 - 99"> 99,
 82                          <"100 - 149"> 149,
 83                          <"150 - 199"> 199,
 84                          <"200 - 299"> 299,
 85                          <"300 - 399"> 399,
 86                          <"400 - 499"> 499,
 87                          <"500 - 599"> 599,
 88                          <"600 - 699"> 699,
 89                          <"700 +"> 709;
 90
```

Variables

```
 91  var x[R*C] integer >= 0;
 92
 93  var y[C*T] binary;
 94
 95  var z[C*T] real >= 0;
 96
```

Función Objetivo

```
 97  minimize fobj:
 98     sum <i,t> in C*T: (costo[t,i] * z[i,t]);
```

Restricciones

```
 99  subto r1: forall <j> in R do
100  (sum <i> in C: x[j,i]) == escuelas[j];
101
102  subto r2: forall <i,t> in C*T do
```

```
103  (sum <j> in R: x[j,i]) >=
104     minimo[t] - 709 * (1 - y[i,t]);
105
106  subto r3: forall <i,t> in C*T do
107  (sum <j> in R: x[j,i]) <=
108     maximo_tr[t] + 709 * (1 - y[i,t]);
109
110  subto r4: forall <i> in C do
111  (sum <t> in T: y[i,t]) == 1;
112
113  subto r5: forall <i,t> in C*T do
114  z[i,t] >=
115     (sum <j> in R: x[j,i]) - 709 * (1 - y[i,t]);
116
117  subto r6: forall <i,j> in C*R
118  with presencia[j,i] == 0 do x[j,i] == 0;
119
120  subto r7: forall <i> in C do
121  (sum <t> in T: z[i,t]) == (sum <j> in R: x[j,i]);
```

Apéndice C

Código C ++

listaInt.h

```
#ifndef _LISTAINT_H_
#define _LISTAINT_H_

struct NodoBi;
#include <iostream>
#include <fstream>
using namespace std;

class listaInt
{
    public:

    int tam()const;
    int iesimo(int i)const;
    listaInt();
    void ag_adelante(int x);
    void ag_atras(int x);
    void sin_primero();
    void sin_ultimo();
    void agregarle(const listaInt& l2);
    listaInt copiar()const;
    bool iguales(const listaInt& l2);
    void borrar_Lista();
    listaInt& operator=(const listaInt& l);
    listaInt(const listaInt& l);
    ~listaInt();
    friend ostream&
        operator<<(ostream& f,const listaInt& l);
    friend istream&
```

```
        operator>>(istream& is, listaInt& l);

    private:

    int cantidad;
    NodoBi* primero;
    NodoBi* ultimo;

};

#endif
```

diccionario.h

```
#ifndef _DICCIONARIO_H_
#define _DICCIONARIO_H_

#include "listaInt.h"
#include <iostream>
#include <fstream>

struct NodoABBD;
typedef int clave;
typedef int valor;
using namespace std;

class diccionario
{
    public:

    diccionario();
    bool contieneclave(clave k)const;
    valor buscarvalor(clave k)const;
    void definir(clave k, valor v);
    void indefinir(clave k);
    void unircon(const diccionario& d2);
    void intersecarcon( const diccionario& d2);
    int cantclaves()const;
    listaInt claves()const;
    listaInt valores()const;
    bool iguales(diccionario& d2)const;
    void borrar_diccionario();
    diccionario& operator=(const diccionario& d);
    diccionario(const diccionario& d);
```

```
    ~diccionario();
    friend ostream&
        operator<<(ostream& f,const diccionario& d);
    friend istream&
        operator>>(istream& is, diccionario& d);

    private:

    int cantidad;
    NodoABBD* arbol;

    bool contieneclave_aux
        (const NodoABBD* raiz,clave k)const;
    valor buscarvalor_aux
        (const NodoABBD* raiz,clave k)const;
    NodoABBD* definir_aux
        (NodoABBD* raiz,clave k, valor v);
    NodoABBD* indefinir_aux(NodoABBD* raiz,clave k);
    NodoABBD* unircon_aux
        (NodoABBD* raiz1, const NodoABBD* raiz2);
    NodoABBD* intersecarcon_aux
        (NodoABBD* raiz, const NodoABBD* raiz2);
    void claves_aux
        (NodoABBD* raiz,listaInt& _claves)const;
    void valores_aux
        (NodoABBD* raiz,listaInt& _valores)const;
    NodoABBD* borrar_diccionario_aux(NodoABBD* raiz);

};

#endif
```

listaInt.cpp

```
#include "listaInt.h"
#include <cstdlib>
#include <iostream>
#include <fstream>
using namespace std;


struct NodoBi
{
    int valor;
```

```
    NodoBi* siguiente;
    NodoBi* anterior;
};

int listaInt::tam()const
{
    return this->cantidad;
}

int listaInt::iesimo(int i)const
{
    NodoBi* _iterador_ = this->primero;
    for(int j=0; j<i; j++)
    {
        _iterador_ = _iterador_->siguiente;
    }
    return _iterador_->valor;
}

listaInt::listaInt()
{
    this->cantidad = 0;
    this->primero = NULL;
    this->ultimo = NULL;
}

void listaInt::ag_adelante(int x)
{
     NodoBi* _n = new NodoBi;
     _n->valor = x;
     _n->anterior = NULL;
     _n->siguiente = this->primero;
     if(this->cantidad == 0)
     {
          this->ultimo = _n;
     }
     else
     {
          (this->primero)->anterior = _n;
     }
     this->primero = _n;
     (this->cantidad)++;
}
```

```
void listaInt::ag_atras(int x)
{
    NodoBi* _n = new NodoBi;
    _n->valor = x;
    _n->siguiente = NULL;
    _n->anterior = this->ultimo;
    if(this->cantidad == 0)
    {
        this->primero = _n;
    }
    else
    {
        (this->ultimo)->siguiente = _n;
    }
    this->ultimo = _n;
    (this->cantidad)++;
}

void listaInt::sin_primero()
{
    if(this->cantidad == 1)
    {
        this->primero = NULL;
        this->ultimo = NULL;
        this->cantidad = 0;
    }
    else
    {
        NodoBi* segundo = this->primero->siguiente;
        delete this->primero;
        this->primero = segundo;
        (this->cantidad)--;
    }
}

void listaInt::sin_ultimo()
{
     if(this->cantidad == 1)
    {
        this->primero = NULL;
        this->ultimo = NULL;
        this->cantidad = 0;
    }
    else
```

```
    {
        NodoBi* anteultimo = this->ultimo->anterior;
        delete this->ultimo;
        this->ultimo = anteultimo;
        (this->cantidad)--;
    }
}

void listaInt::agregarle(const listaInt& l2)
{
    NodoBi* iterador_ = l2.primero;
    while(iterador_ != NULL)
    {
        this->ag_atras(iterador_->valor);
        iterador_ = iterador_->siguiente;
    }
}

bool listaInt::iguales(const listaInt& l2)
{
    if(this->tam() == l2.tam())
    {
        NodoBi* _this_ = this->primero;
        NodoBi* _l2_ = l2.primero;
        for(int j=0; j<this->tam(); j++)
        {
            if(_this_->valor != _l2_->valor)
            {
                return false;
            }
            _this_ = _this_->siguiente;
            _l2_ = _l2_->siguiente;
        }
        return true;
    }
    return false;
}

void listaInt::borrar_Lista()
{
    NodoBi* l = this -> primero;
    while(l != NULL)
    {
        NodoBi* sig = l->siguiente;
```

```
        delete l;
        l = sig;
    }
}

listaInt& listaInt::operator=(const listaInt& l)
{
    (*this).borrar_Lista();

NodoBi* indicador = l.primero;

while(indicador != NULL)
{
(*this).ag_atras(indicador -> valor);
indicador = indicador -> siguiente;
}

return *this;
}

listaInt::listaInt(const listaInt& l)
{
    this->cantidad = 0;
    this->primero = NULL;
    this->ultimo = NULL;
    *this = l;
}

listaInt::~listaInt()
{
    (*this).borrar_Lista();
}

ostream& operator<<(ostream& f,const listaInt& l)
{
    NodoBi * indicador = l.primero;

    if(indicador != NULL)
    {
        f << "["<<indicador->valor;
        indicador = indicador->siguiente;
    }

    while( indicador!=NULL )
```

```
    {
        f <<","<< indicador->valor;
        indicador=indicador->siguiente;
    }
    f<<"]";

    return f;
}

istream& operator>>(istream& is, listaInt& l)
{
    l.~listaInt();
    is.get();
    char c=is.peek();
    while( c != ']')
    {
        int valor = 0;
        is >> valor;
        l.ag_atras(valor);
        c=is.peek();
        if(c == ',')
        {
            is.get();
        }

    }
    return is;
}
```

diccionario.cpp

```
#include "listaInt.h"
#include "diccionario.h"
#include <cstdlib>


struct NodoABBD
{
    int clave;
    int valor;
    NodoABBD* derecho;
    NodoABBD* izquierdo;
};
```

```
diccionario::diccionario()
{
    this->cantidad=0;
    this->arbol=NULL;
}
bool diccionario::contieneclave_aux
    (const NodoABBD* raiz,clave k)const
{
    if (raiz==NULL)
    {
        return false;
    }
    else if((raiz->clave) == k)
    {
        return true;
    }
    else if((raiz->clave) < k)
    {
        return this->
                contieneclave_aux(raiz->derecho,k);
    }
    else
    {
        return this->
                contieneclave_aux(raiz->izquierdo,k);
    }
}

bool diccionario::contieneclave(clave k)const
{
    return this->contieneclave_aux(this->arbol ,k);
}

valor diccionario::buscarvalor_aux
            (const NodoABBD* raiz,clave k)const
{
     if(raiz!=NULL)
     {
        if ((raiz->clave) == k)
        {
            return raiz->valor;
        }
        else
        {
```

```
            if((raiz->clave) < k)
            {
                return buscarvalor_aux
                            (raiz->derecho,k);
            }
            else
            {
                return buscarvalor_aux
                            (raiz->izquierdo,k);
            }
        }
     }
     return 0;
}

valor diccionario::buscarvalor(clave k)const
{
   return buscarvalor_aux(this->arbol,k);
}

NodoABBD* diccionario::definir_aux
                (NodoABBD* raiz,clave k, valor v)
{
    if(raiz == NULL)
    {
        raiz=new NodoABBD ;
        raiz->clave = k;
        raiz->valor = v;
        raiz->izquierdo = NULL;
        raiz->derecho = NULL;
        (this->cantidad)++;
    }
    else if((raiz->clave)< k)
    {
       raiz->derecho=definir_aux(raiz->derecho,k,v);
    }
    else if((raiz->clave) > k)
    {
        raiz->izquierdo=definir_aux
                            (raiz->izquierdo,k,v);
    }
    else
    {
        raiz->valor=v;
```

```
    }
    return raiz;
}

void diccionario::definir(clave k, valor v)
{
   this->arbol=definir_aux(this->arbol,k,v);
   return;
}

NodoABBD* diccionario::indefinir_aux
                        (NodoABBD* raiz,clave k)
{
    if(raiz!=NULL)
    {
        if(contieneclave_aux(raiz,k) == 1)
        {
            if(raiz->clave < k)
            {
               raiz->derecho=indefinir_aux
                                (raiz->derecho,k);
            }
            else if(raiz->clave > k)
            {
                raiz->izquierdo=indefinir_aux
                                (raiz->izquierdo,k);
            }
            else
            {
                if(raiz->derecho==NULL)
                {
                    NodoABBD* nodo_a_extraer = raiz;
                    raiz = raiz->izquierdo;
                    nodo_a_extraer->izquierdo = NULL;
                    delete nodo_a_extraer;
                    (this->cantidad)--;
                }

                else
                {
                    NodoABBD* nodominimo=
                                    raiz->derecho;
                    while(nodominimo->
                                    izquierdo!=NULL)
```

```
                    {
                        nodominimo=nodominimo->
                                            izquierdo;
                    }
                    raiz->clave=nodominimo->clave;
                    raiz->valor=nodominimo->valor;
                    raiz->derecho=indefinir_aux
                       (raiz->derecho,
                                nodominimo->clave);
                }
            }
        }
    }

    return raiz;
}

void diccionario::indefinir(clave k)
{
    this->arbol=indefinir_aux(this->arbol,k);
}

NodoABBD* diccionario::unircon_aux
            (NodoABBD* raiz1, const NodoABBD* raiz2)
{
    if(raiz2 != NULL)
    {
        if(contieneclave_aux(raiz1,raiz2->clave) == 0)
        {
            raiz1=definir_aux
                   (raiz1,raiz2->clave,raiz2->valor);
        }

        raiz1=unircon_aux(raiz1,raiz2->derecho);
        raiz1=unircon_aux(raiz1,raiz2->izquierdo);
    }
    return raiz1;
}

void diccionario::unircon(const diccionario& d2)
{
    this->arbol=unircon_aux(this->arbol,d2.arbol);
}
```

```
int diccionario::cantclaves()const
{
    return this->cantidad;
}

NodoABBD* diccionario::intersecarcon_aux
            (NodoABBD* raiz1, const NodoABBD* raiz2)
{
    if(raiz1 != NULL && raiz2 != NULL)
    {
       if(contieneclave_aux(raiz2,raiz1->clave)==0)
       {
           raiz1=indefinir_aux(raiz1,raiz1->clave);
           raiz1=intersecarcon_aux(raiz1,raiz2);
       }
       else
       {
           raiz1->derecho=intersecarcon_aux
                           (raiz1->derecho,raiz2);
           raiz1->izquierdo=intersecarcon_aux
                          (raiz1->izquierdo,raiz2);
       }
    }
    return raiz1;
}

void diccionario::intersecarcon
                           (const diccionario& d2)
{
    this->arbol=intersecarcon_aux
                           (this->arbol, d2.arbol);
}

void diccionario::claves_aux
            (NodoABBD* raiz, listaInt& _claves)const
{

    if(raiz!=NULL)
    {
        claves_aux(raiz->izquierdo,_claves);
        _claves.ag_atras(raiz->clave);
        claves_aux(raiz->derecho,_claves);
        return;
    }
```

```
}

listaInt diccionario::claves()const
{
    listaInt lista_claves;
    claves_aux(this->arbol,lista_claves);
    return lista_claves;
}

void diccionario::valores_aux
            (NodoABBD* raiz, listaInt& _valores)const
{
    if(raiz!=NULL)
    {
        valores_aux(raiz->izquierdo,_valores);
        _valores.ag_atras(raiz->valor);
        valores_aux(raiz->derecho,_valores);
        return;
    }
}

listaInt diccionario::valores()const
{
    listaInt lista_valores;
    valores_aux(this->arbol,lista_valores);
    return lista_valores;
}

bool diccionario::iguales(diccionario& d)const
{
    return(this->cantidad==d.cantidad &&
        this->claves().iguales(d.claves()) &&
        this->valores().iguales(d.valores()));
}

NodoABBD* diccionario::borrar_diccionario_aux
                                    (NodoABBD* raiz)
{
    while (raiz != NULL)
    {
        raiz = indefinir_aux(raiz, raiz->clave);
    }
    return raiz;
```

```
}

void diccionario::borrar_diccionario()
{
    this->arbol = borrar_diccionario_aux
                                    (this->arbol);
    return;
}

NodoABBD* copiar_arbol(const NodoABBD* raiz)
{
    if(raiz == NULL)
    {
        return NULL;
    }
    else
    {
        NodoABBD* copia = new NodoABBD;
        copia->clave = raiz->clave;
        copia->valor = raiz->valor;
        copia->derecho = copiar_arbol(raiz->derecho);
        copia->izquierdo = copiar_arbol
                                    (raiz->izquierdo);
        return copia;
    }
}

diccionario& diccionario::operator=
                                (const diccionario& d)
{
    (*this).borrar_diccionario();
    this->arbol = copiar_arbol(d.arbol);
    this->cantidad = d.cantidad;
    return *this;
}

diccionario::diccionario(const diccionario& d)
{
    this->cantidad = 0;
    this->arbol = NULL;
    *this = d;
}

diccionario::~diccionario()
```

```
{
    (*this).borrar_diccionario();
}

ostream& operator<<(ostream& f,const diccionario& d)
{
    listaInt l_claves=d.claves();
    listaInt l_valores=d.valores();
    if(d.cantidad != 0)
    {
        f<<"[("<< l_claves.iesimo(0)<<","
                    <<l_valores.iesimo(0)<<")";
    }
    for( int i=1;i<d.cantidad;i++)
    {
        f<<";("<< l_claves.iesimo(i)<<","
                    <<l_valores.iesimo(i)<<")";
    }
    f<<']';
    return f;
}

istream& operator>>(istream& is, diccionario& d)
{
    d.~diccionario();
    is.get();
    char c = is.peek();
    while( c != ']')
    {
        clave x = 0;
        is >> x;
        valor y = 0;
        is >> y;
        is.get();
        d.definir(x,y);
        c = is.peek();
        if(c == ',')
        {
            is.get();
        }
    }
    return is;
}
```

main.cpp

```
#include <iostream>
#include "listaInt.h"
#include "diccionario.h"
#include <fstream>
#include <vector>
#include <string>
#include <math.h>

using namespace std;

int main()
{
    diccionario cv_c;
    cv_c.definir(0,0);
    ifstream h("cv_c.txt");
    h>>cv_c;

    diccionario ta_c_2;
    ta_c_2.definir(0,0);
    ifstream k("ta_c_2.txt");
    k>>ta_c_2;

    diccionario tc_c_2_a;
    tc_c_2_a.definir(0,0);
    ifstream m("tc_c_2_a.txt");
    m>>tc_c_2_a;

    diccionario tmx_f_2;
    tmx_f_2.definir(0,0);
    ifstream q("tmx_f_2.txt");
    q>>tmx_f_2;

    cout<<endl<<"Prueba ta_c_2"<<endl<<ta_c_2;
    cout<<endl<<"Prueba tc_c_2_a"<<endl<<tc_c_2_a;
    cout<<endl<<"Prueba tmx_f_2"<<endl<<tmx_f_2;

    diccionario reg;
    for (int i = 1; i < 710; i++)
    {
      int etiqueta_i = 8*(cv_c.contieneclave(i)) +
      4*(ta_c_2.contieneclave(i)) +
      2*(tc_c_2_a.contieneclave(i)) +
```

```
  1*(tmx_f_2.contieneclave(i));
    reg.definir(i, etiqueta_i);
}

vector < pair<int , int > > A;
int long_ = (reg.claves()).tam();
for(int i = 0; i < long_; i++)
{
    A.push_back(make_pair((reg.valores()).
        iesimo(i),(reg.claves()).iesimo(i)));
}
sort(A.begin(),A.end());
ofstream regiones;
regiones.open("regiones.txt");
int _count_ = 1;
listaInt _cantidades_;
vector < string > S;
S.push_back("ta_c_2");
S.push_back("tc_c_2_a");
S.push_back("tmx_f_2");
string reg_ ("");
int temp;
for (int i = 0; i < long_; i++)
{
    regiones<<A[i].first<<", ";
    regiones<<A[i].second<<endl;
    if (A[i].first == A[i + 1].first)
    {
        _count_ ++;
    }
    else
    {
        _cantidades_.ag_atras(_count_);
        temp = A[i].first;
        int pot_ = 2;
        while ( pot_ >= 0 )
        {
            if ( temp >= pow (2, pot_) )
            {
                reg_ += S[2 - pot_];
                reg_ += ", ";
                temp = temp - pow (2, pot_);
            }
            pot_ --;
```

```
            }
            regiones<<endl<<"Total "<<A[i].first
            <<" = "<<reg_<<": "<<_count_<<endl<<endl;
            _count_ = 1;
            reg_ = "";
        }
    }
    int _total_escuelas_ = 0;
    int tam_cantidades = _cantidades_.tam();
    for (int n = 0; n < tam_cantidades; n ++)
    {
        _total_escuelas_ += _cantidades_.iesimo(n);
    }
    regiones<<endl<<"Total escuelas: "<<
                                    _total_escuelas_;
    regiones<<endl<<"long_ = "<<long_;
    regiones.close();
    return 0;
}
```

Bibliografía

[1] Achterberg, T. 2007. *Constraint Integer Programming. Ph.D. Thesis.* Technische Universität Berlin.

[2] An, N., Elmaghraby, W., Keskinocak, P. 2005. *Bidding strategies and their impact on revenues in combinatorial auctions.* Journal of Revenue and Pricing Management (3,4).

[3] Andersson, A., Tenhunen, M., Ygge, F. 2000. *Integer Programming for Combinatorial Auction Winner Determination.* Proceedings of the Fourth International Conference on Multi-Agent Systems (39-46).

[4] Archer, A., Papadimitriou, C., Talwar, K., Tardos, E. 2003. *An Approximate Truthful Mechanism for Combinatorial Auctions with Single Parameter Agents.* Proceedings of the 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2003), (205-214).

[5] Bartal, Y., Gonen, R., Nisan, N. 2003. *Incentive Compatible Multi Unit Combinatorial Auctions.* M. Tennenholtz, ed., Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge (72-87).

[6] Binmore, K. 2007. *Playing for Real. A text on Game Theory.* Oxford University Press.

[7] Bikhchandani, S., de Vries, S., Vohra, R., Schummer, J. 2002. *Linear Programming and Vickrey Auctions.* Brenda Dietrich and Rakesh V. Vohra, eds., Mathematics of the Internet: E-Auction and Markets, New York: Springer Verlag, (75-115).

[8] Boutilier, C. 2002. *A POMPD Formulation of Preference Elicitation Problems.* National Conference on Artificial Intelligence, (239-246).

[9] Cantillon, E., Pesendorfer, M. 2006. *Combination Bidding in Multi-Unit Auctions.* CEPR discussion paper 6083.

[10] Cook, S. 1971. *The Complexity of Theorem Proving Procedures.* Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, (151-158).

[11] Cramton, P., Shoham, Y., Steinberg R. 2006. *Combinatorial Auctions.* The MIT Press, Massachusetts Institute of Technology.

[12] de Vries, S., Vohra, R., V. 2003. *Combinatorial Auctions: A Survey.* INFORMS Journal on Computing. Vol. 15, No. 3.

[13] *Diccionario de la Lengua Española de la Real Academia.* Real Academia Espanola, 2002.

[14] Durán, G., Epstein, R., Martínez, C., Zamorano, G. 2010. *Quantitative Methods for a New Configuration of Territorial Units in a Chilean Government Agency Tender Process.* Interfaces (en prensa).

[15] Elmaghraby, W., Keskinocak, P. 2002. *Combinatorial Auctions in Procurement.* Technical Report, School of Industrial and Systems Engineering, Georgia Institute of Technology.

[16] Epstein, R., Henríquez, L., Catalán, J., Weintraub, G., Martínez, C. 2002. *A Combinational Auction Improves School Meals in Chile.* Interfaces (32,6).

[17] FCC. 2002. *The Federal Communications Commission Public Notice DA02-260*.

[18] Garey, M. R., Johnson, D. S. 1979. *Computers and Intractability. A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, New York.

[19] Friedman, L. 1956. *A Competitive Bidding Strategy.* Operations Research. 4 (104-112).

[20] Gonen, R., Lehmann, D. 2001. *Linear Programming Helps Solving Large Multi-unit Combinatorial Auctions.* Technical Report TR-2001-8, Leibniz Center for Research in Computer Science.

[21] Karp, R. M. 1972. *Reducibility Among Combinatorial Problems.* R. E. Miller, J. W. Thatcher eds., Complexity of Computer Computations, Plenum Press. New York (85-103).

[22] Kennedy, D., Glaister, S., Travers, T. 1995. *London Bus Tendering, Greater London Group.* Report, London School of Economics.

[23] Klemperer, P. 1999. *Auction Theory: A Guide to the Literature.* Journal of Economic Surveys (13).

[24] Klemperer, P. 2000. *The Economic Theory of Auctions.* Cheltenham, UK: Edward Elgar.

[25] Koch, T. 2004. *Rapid Mathematical Programming.* Technische Universität Berlin.

[26] Koch, T. 2007. *Zimpl User Guide (Zuse Institute Mathematical Programming Language).* Zuse-Institut Berlin.

[27] Mas-Colell, A., Whinston, M., Green, J. 1995. *Microeconomic Theory.* Oxford University Press.

[28] Maskin, E. 2004. *The Unity of Auction Theory: Milgrom's Masterclass.* Journal of Economic Literature. Vol. 42, No. 4 (1102-1115).

[29] Matthews, S. 1984. *On the Implementability of Reduced Form Auctions.* Econometrica 52 (1519-1522).

[30] Milgrom, P. 2004. *Putting Auction Theory to Work.* Cambridge: Cambridge University Press.

[31] Milgrom, P., Weber, R. 1982. *A Theory of Auctions and Competitive Bidding.* Econometrica, 50 (1089-1122).

[32] Nisan, N. 2000. *Bidding and Allocation in Combinatorial Auctions.* Proceedings of the 2nd ACM Conference on Electronic Commerce (1-12).

[33] Nisan, N., Ronen, A. 2000. *Computationally Feasible VCG Mechanisms.* Proceedings of the 2nd ACM Conference on Electronic Commerce (EC 00), (242-252).

[34] Nisan, N., Ronen, A. 2001. *Algorithmic Mechanism Design.* Games and Economic Behavior, 35 (166-196).

[35] Papadimitriou, Ch., Steiglitz, K. 1982. *Combinatorial Optimization. Algorithms and Complexity* Dover Publications, Inc. New York.

[36] Pekeč, A., Rothkopf, M. 2003. *Combinatorial Auction Design.* Management Science (49,11).

[37] Pekeč, A. 2001. *Tradeoffs in Combinatorial Auction Design.* 2001 Combinatorial Bidding Conference, Wye River Conference Center, Queenstown, MD, October 27.

[38] Rassenti, S., Smith, V., Bulfin, R. 1982. *A Combinatorial Auction Mechanism for Airport Time Slot Allocation.* Bell Journal of Economics, 13 (402-417).

[39] Rothkopf, M. 1969. *A Model of Rational Competitive Bidding.* Management Science, 15 (362-372).

[40] Rothkopf, M., Dougherty, E., Rose, M. 1986. *Comment on Multi-Object Auctions: Sequential vs. Simultaneous Sales* Management Science (32).

[41] Rothkopf, M., Park, S. 2001. *An Elementary Introduction to Auctions.* Rutcor Research Report.

[42] Rothkopf, M. H., Pekeč, A., Harstad, R. M. 1998. *Computationally Manageable Combinational Auctions.* Management Science (44,8).

[43] Rothkopf, M. H., Teisberg, T. J., Kahn, E. P. 1990. *Why Are Vickrey Auctions Rare?.* Journal of Political Economy (98, 94-109).

[44] Sandholm, T. 2000. *Issues in Computational Vickrey Auctions.* International Journal of Electronic Commerce (4, 107-129).

[45] Sandholm, T., Suri, S., Gilpin, A., Levine, D. 2002. *Winner Determination in Combinatorial Auction Generalizations.* Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS).

[46] Schrijver, A. 1986. *Theory of Linear and Integer Programming*. John Wiley & Sons, Chichester.

[47] Vickrey, W. 1961. *Counterspeculation, Auctions, and Competitive Sealed Tenders.* Journal of Finance (16, 8-37).

[48] Vickrey, W. 1962. *Auction and Bidding Games.* Recent Advances in Game Theory, The Princeton University Conference (15-27).