Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Matemática

Cálculo de Pacing Equilibriums para Mercados de Subastas vía Programación Lineal Entera

Tesis de Licenciatura en Ciencias Matemáticas

Gustavo Martín Hurovich

Director: Dr. Juan José Miranda-Bront

Buenos Aires, Marzo 2020

Cálculo de Pacing Equilibriums para Mercados de Subastas vía Programación Lineal Entera

En el mundo de la publicidad online, un problema fundamental es el de decidir qué anuncio mostrarle a cada usuario. Actualmente, esta asignación se realiza a través de subastas, en las cuales se generan apuestas en nombre de los anunciantes. De resultar ganador en alguna de ellas, su anuncio será el que efectivamente se le presente al usuario. Uno de los mayores inconvenientes que tiene este esquema es que si un anunciante excede su presupuesto, pierde la oportunidad de participar en subastas subsecuentes. Es por eso que surge el *budget management*, un conjunto de diversas estrategias para regular el gasto de cada uno de los participantes, permitiéndole ganar aquellas oportunidades que le proveen mayor utilidad. Dentro de esta categoría se encuentra el *pacing multiplicativo*, el cual consiste en, para cada anunciante, reducir uniformemente sus apuestas originales.

Dada esa metodología, es posible pensar al mercado como un *juego*, en el que cada participante tiene como estrategias posibles el elegir por qué factor reducir sus apuestas, y recibe utilidad por cada ítem que logra ganar. En estos juegos se puede definir un tipo de equilibrio, al que se denomina *pacing equilibrium*. Aunque encontrar un equilibrio que maximice *revenue* o *social welfare* es un problema NP-Completo, se puede formular un modelo de Programación Lineal Entera Mixta que sirva para encontrar dichos equilibrios o uno arbitrario.

En este trabajo analizamos propiedades teóricas de dicho modelo, así como características de los equilibrios resultantes. Presentamos la definición de instancias ajustadas para aquellas en las cuales los presupuestos son, en algún sentido, reducidos, y mostramos empíricamente que son más difíciles de resolver para este modelo. A su vez, presentamos una serie de desigualdades válidas que refuerzan el modelo y tienen un impacto positivo, permitiendo resolver instancias de tamaños más grandes.

Palabras claves: Publicidad Online, Budget Management, Pacing Equilibrium, Programación Lineal Entera, Teoría de Juegos Algorítmica.

PACING EQUILIBRIUMS FOR AUCTION MARKETS VIA INTEGER LINEAR PROGRAMMING

In the context of online advertising, a fundamental problem involves deciding which ad to show each user. Currently, this allocation is made through auctions, on which bids are generated on behalf of advertisers. In case of winning in any of them, their ad will be the one efectively shown to the user. One of the biggest obstacles of this scheme is that if an advertiser exceeds its budget, they lose the oportunity to participate in subsequent auctions. That is why budget management emerges, a set of various strategies to adjust the pace of budget consumption over time of each of the participants, allowing them to win those opportunities that might provide them higher utility. Within this category is the multiplicative pacing, which consists in, for each advertiser, reduce uniformly its original bids.

Given this methodology, it is possible to model the market as a game, where players are allowed to choose a factor to scale their bids, and receive utility for each item they win. In this games we can define a specific type of equilibrium, which is called pacing equilibrium. Even though finding a revenue-maximizing or social welfare-maximizing pacing equilibrium is an NP-Complete problem, it is possible to formulate an equivalent mixed integer linear programming model to compute these equilibriums

In this thesis, we analyze theoretical properties of such model, as well as several characteristics of the resulting equilibria. We present the definition of budget-tight instances for the ones on which budgets are, in some way, small, and we show empirically that tightness of the budgets has a direct impact on the computation time to solve the instance. Moreover, we present a set of valid inequalities that reinforce the model and have a positive impact, allowing it to solve larger instances.

Keywords: Online Advertising, Budget Management, Pacing Equilibrium, Integer Linear Programming, Algorithmic Game Theory.

AGRADECIMIENTOS

Me gustaría agradecer a toda la gente que me acompañó en este largo camino. En particular, quiero mencionar a algunes.

A mis compañeros de cursada, y a los que de tanta cursada se me hicieron amigues. Al Colo y a Juli, con quienes nos reímos tanto que hasta nos olvidamos de estudiar. A Murguis, Rouli y Gasti, que de tantos momentos compartidos nos volvimos indistinguibles. A Espi, Nacho y Fosqui, compañeros de viajes y aventuras. A Laion, Agu y Eicho. A Guty, Flor, Gordo, Tomo, Fran, Pierna, Chanchu. A todos con quienes disfruté compartiendo pasantías, clases, cursadas, TPs, recreos, partidos de metegol y ping-pong.

A todos los docentes con los que me formé. A Mati, Facu, Vero, Dani, Juanjo, quienes me contagiaron su amor por la docencia, y de quienes aprendí mil y un estrategias y maneras de enseñar.

A mis amigos de la vida, quienes me vieron disfrutar la matemática desde siempre. A Juli y Tina, a Milton, a Berto, Mati, Zequi y Tom. Con todo el cariño que les tengo les agradezco por acompañarme desde hace tantos años.

A Fran y Willy, por ser jurados de esta tesis y aportarme comentarios y devoluciones.

A mis directores, Juanjo e Isa. Increíbles guías en este camino, siempre dispuestos a sentarse a pensar conmigo y trabajar el problema. Lo que aprendí de ustedes en estos meses no tiene nombre. A Gonza, por su apoyo y ayuda para realizar este trabajo.

A mis viejos, que me dieron la oportunidad de estudiar en esta universidad. A mis hermanas, que me acompañan y cuidan desde pequeños. A toda mi familia.

A Camu, por ser una Compañera con C mayúscula. Por tu contención, cariño y cuidado. Por bajarme a tierra cuando hace falta, y dejarme volar cuando lo necesito. Por tu aguante todas las noches de nervios antes de un examen, por tu confianza constante. Por todo, gracias.

Índice general

1. INTRODUCCIÓN

1.1 Motivación

Hoy en día, las publicidades online juegan un rol fundamental en las compañías de internet. Para los anunciantes, es una manera de publicitar sus productos, ideas y proyectos, mientras que para las páginas web es una oportunidad de financiarse sin trasladar el costo a los usuarios. Si bien la relación entre ambos puede ser directa, eso tiene un problema de escala: un anunciante no puede acordar con todos los espacios publicitarios, y, similarmente, el dueño de uno de esos espacios probablemente tampoco tenga los recursos para ofrecerlo a todos los posibles anunciantes. Es así como surgen las plataformas de publicidad online, como intermediarias entre ambas partes.

Vale aclarar que muchas veces, los roles de plataforma y proveedores de espacios para publicidad están combinados. Es el caso de motores de búsqueda, como Google o Bing y redes sociales como Facebook o Twitter, quienes pueden hacer uso de sus abundantes recursos para ofrecer una buena experiencia a anunciantes y usuarios sin necesidad de recurrir a un intermediario externo. Así, cuando un usuario realiza una búsqueda online, es el mismo buscador quien decide qué publicidad mostrarle.

Al agregar a esta plataforma como intermediaria, uno podría argumentar que incluso cuando esos dos roles están desdoblados, la página web que muestra la publicidad ya no juega un papel tan importante, pues delegó su poder de decisión a la plataforma centralizada. Por este motivo, a lo largo de este trabajo vamos a omitir ese rol, y simplemente hablar de anunciantes, audiencia, y la plataforma que es el nexo entre ellos.

Uno de los problemas fundamentales que las plataformas tienen que resolver es encontrar una correspondencia entre anunciantes y audiencia, para decidir qué publicidad mostrar en cada espacio, o a cada usuario. Una opción *naive* podría ser realizar la asignación al azar: cuando una página es cargada y hace un pedido por una publicidad, asignarle cualquiera de las disponibles. Sin embargo, esto resulta una muy mala decisión, pues los anunciantes terminan invirtiendo dinero en publicitarle a un público que muy probablemente no esté interesado, y la audiencia recibe anuncios que no le son de interés. Y desde el lado de la plataforma, si lo que ofrece es un precio por click, y a los usuarios no les interesan las publicidades que reciben, no va a obtener ganancia alguna. En definitiva, todos pierden.

Eso introduce una noción muy interesante. Cada publicidad puede ser muy buena o muy mala, según a quién se la muestre. Así, para una pequeña emprendedora porteña que vende muebles artesanales, anunciar su página a un joven que vive en la ciudad de Buenos Aires vale mucho más que mostrársela a otro que vive a miles de kilómetros de distancia, cuyo interés no va a poder consolidarse en una venta.

Similarmente, cada anunciante podría valorar distinto las diferentes interacciones de un usuario con el anuncio. De esta manera, a una marca establecida en el mercado le podrá alcanzar que cierto subconjunto de la población *observe* su publicidad (lo que se llama una *impresión*), mientras que nuestra emprendedora puede encontrar más valioso que los usuarios efectivamente hagan un *click* y sean redirigidos a su sitio.

¿Qué pasaría si consideráramos un sistema en el que solamente existe una oportunidad para mostrar un anuncio? En este caso, podríamos realizar una subasta: todos los anunciantes ofertan en secreto lo que pagarían por mostrar el suyo; el que ofertó más alto gana, y paga lo que ofertó. Este mecanismo se llama *subasta a sobre cerrado de primer precio*. Si bien parece a priori una opción razonable, tiene un problema fundamental, que es que los anunciantes tienen un incentivo natural para ofertar menos de lo que realmente pagarían. En Vickrey [16] se propone el método de *subasta a sobre cerrado de segundo precio* (o subasta Vickrey), en el cual todos los participantes ofertan en secreto, y el ítem es entregado al que realizó la oferta más alta (tal como en la versión de primer precio), pero el precio es fijado por la segunda apuesta más alta. La elección de este método, lejos de arbitraria, es fundamentada en el hecho de que es poco manipulable, es decir que la estrategia óptima de cada uno de los participantes es ofertar lo que realmente valora el ítem a subastar. Más aún, esta propiedad se conserva incluso si algunos de los participantes conocen las ofertas del resto.

Una característica importante de la subasta Vickrey es que, en cierta manera, hace a la subasta más justa. La oportunidad la va a ganar aquel que pueda obtener un mayor beneficio de ella, sin importar el conocimiento previo que tenga sobre el resto de los participantes. Esto permite que nuevos jugadores puedan incorporarse sin estar en desventaja (por ejemplo, por no conocer cómo suelen ser las apuestas de los demás), e impide manipulaciones por parte de los jugadores con mayor disponibilidad de recursos.

En el contexto de subastas online existe una enorme cantidad de oportunidades para repartir. ¿Cómo podemos extender esta subasta para que permita distribuir todas ellas? Si pensamos que cada oportunidad (o cierto grupo de oportunidades) de mostrarle un anuncio a un usuario es un ítem u objeto por el que los anunciantes compiten, podemos pensar que estamos en una situación en la que hay un conjunto de participantes o jugadores (los anunciantes) queriendo obtener algunos de esos ítems. Cada jugador cuenta con cierto *presupuesto* y tiene una *valuación* para cada objeto, y quiere recibir un conjunto de ítems que maximicen su ganancia. El organizador, por su parte, quiere realizar esa asignación, pero no es claro qué quiere priorizar. Una opción es tratar de maximizar lo recaudado, aunque eso puede dejar disconformes a los participantes, pues eso resultará en menos ganancia para ellos. Otra opción es maximizar la ganancia promedio de ellos, pero esto podría resultar en un reparto desigual del mismo, que perjudique a muchos pequeños anunciantes. En definitiva, una buena asignación de los objetos tiene que ser justa (no priorizar a ningún anunciante por sobre el resto), atractiva para los anunciantes (es decir, que incentive a que ellos utilicen esta plataforma y no otra), útil para la plataforma (que le proporcione una recaudación razonable), y, como mencionamos anteriormente, poco manipulable (que la estrategia óptima de todos sea ser fieles a sus valuaciones reales).

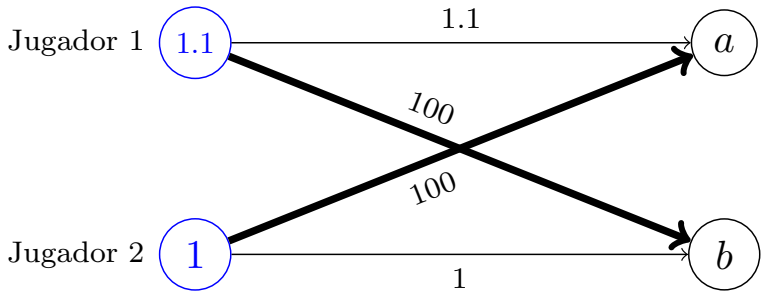


Fig. 1: Representamos al mercado como un grafo bipartito, con los jugadores a izquierda y los ítems a derecha. Aquí, el jugador 1 tiene un presupuesto de $B_1 = 1.1$, y una valuación de 1.1 por el ítem a y 100 por el ítem b, mientras que el jugador 2 tiene como presupuesto $B_2 = 1$ y valuaciones de 100 por a y 1 por b

Dado que el precio por cada *impresión* o *click* resulta muy bajo, y se tiene una cantidad muy grande de ellos, diremos que un ítem es un conjunto de oportunidades. Idealmente el grupo es homogéneo, siendo todas del mismo tipo y con destinatarios "parecidos". En ese sentido, podemos considerar que los ítems son *divisibles*, y en caso de algún empate podremos elegir qué proporción del conjunto (es decir, cuántas de esas oportunidades) asignar a cada uno de los ganadores.

Una solución inicial adoptada en la práctica fue la de realizar sucesivas subastas para obtener la asignación de bienes a anunciantes. Por cada objeto se realiza una subasta de sobre cerrado de segundo precio, utilizando la valuación de cada anunciante por ese objeto como su apuesta. Mientras tenga presupuesto restante, el anunciante seguirá participando en subastas subsiguientes.

Sin embargo, esto trae varios problemas, la mayor parte consecuencia de uno común: para un anunciante con presupuesto limitado, ganar objetos en las primeras subastas puede dejarlo sin presupuesto para las siguientes (y por consiguiente perderse de obtener mayor ganancia).

Consideremos el ejemplo de la Figura 1, con dos jugadores y dos ítems. Si se subasta primero el ítem a, recibiremos apuestas respectivas de 1.1 y 1 (el jugador 2 apuesta lo máximo que puede pagar, que es todo su presupuesto), por lo que el jugador 1 gana la totalidad del ítem a un precio de 1. Luego, al subastar b, el jugador 1 solo puede apostar lo que le quedó de dinero, que es 0.1, por lo que el jugador 2 ganará dicho ítem, pagando 0.1. Aquí, los jugadores reciben una ganancia de 0.1 y 0.9 respectivamente. Es fácil ver que esta situación es muy lejana a la ideal, pues cada jugador ganó el ítem que menos quería. De haberse realizado las subastas en el orden opuesto, la asignación habría respetado esta preferencia para ambos. Lo mismo habría sucedido si el jugador 1 hubiera *mentido* con su presupuesto real, diciéndonos que no era 1.1 sino 0.9, o si hubiera manifestado menor interés por el ítem a.

Respecto a este ejemplo, vale aclarar que es verosímil asumir que un jugador puede tener valuaciones más altas que su presupuesto. Por un lado, porque muchas veces los jugadores son conservadores respecto al presupuesto que quieren gastar, pero eso no sig-

nifica que no puedan obtener una ganancia grande en caso de obtener un ítem. Por otro, porque es fácil imaginar una situación en la que se subastaron varios ítems con anterioridad a estos dos, y que estos presupuestos son los que les sobraron al finalizar las mismas, cuando originalmente eran más altos. Ahora bien, un jugador no puede apostar más que su presupuesto en ningún caso, sin importar qué valuación tenga por ese ítem; se estaría arriesgando a gastar más que lo que tiene, y esto está prohibido.

Podemos mencionar entonces dos desventajas claras de este sistema:

- Cambiar el orden en que se realizan las subastas puede generar grandes diferencias en las ganancias de los anunciantes (como vimos en el ejemplo de la Figura 1), pudiendo la elección del orden perjudicar o beneficiar a algunos por sobre el resto de manera arbitraria.
- Permite *manipulaciones*. Un anunciante podría decir que tiene menos presupuesto que el que realmente tiene, y potencialmente obtener mayor beneficio. También podría hacerse pasar por varios anunciantes con las mismas valuaciones pero el presupuesto distribuido, con el mismo fin.

Uno podría argumentar que en dicho ejemplo, el Jugador 2 perdió en buena ley, pues en definitiva contaba con un presupuesto menor y por ende no le correspondía elegir qué ítem quería. Pero para el otro participante, la situación es realmente desalentadora. Por manifestar algo de interés por el objeto a, se perdió de obtener el ítem que realmente quería. Como mencionamos antes, el problema fue que **se quedó sin presupuesto demasiado rápido**.

Si bien una decisión puede ser dejar que cada participante decida cómo utilizar su presupuesto, las plataformas (en su afán de simplificar la experiencia de uso) usualmente ofrecen la posibilidad de administrar el presupuesto de los anunciantes. En [3] se analizan diversas soluciones a este problema, algunas regulando en qué subastas participa cada jugador, y otras permitiéndoles participar en todas pero reduciendo las apuestas que realizan. Dentro de este segundo grupo, está la opción de *shading* o *pacing multiplicativo* de las apuestas. Este consiste en, para cada jugador i, elegir un factor $0 < \alpha_i \leq 1$ por el cual multiplicar las apuestas; un α_i cercano a cero se corresponderá con un enfoque conservador, probablemente ganando menos ítems y resultando en dinero de sobra al final del día, mientras que valores cercanos a 1 con la estrategia que mencionamos de apostar agresivamente hasta quedarse sin presupuesto (y potencialmente perdiendo la oportunidad de participar en las últimas subastas).

Para cada posible elección de α de cada participante, tenemos una asignación resultante de ítems. Estudiaremos solo aquellas que constituyan un equilibrio, es decir aquellas elecciones de dicho parámetro tales que, dada la elección de los demás, individualmente no hay incentivo para cambiar el propio. En Conitzer et al. [8] analizan este tipo de situaciones, a las que llaman *pacing equilibriums* (PE), y veremos en detalle en el Capítulo 2.

Si bien la existencia de un PE siempre está garantizada, no resulta claro cómo hacer

para efectivamente encontrar uno. Más aún, en general dicho equilibrio no es único; si bien sucede con poca frecuencia, pueden haber escenarios con múltiples (y muy diversos) equilibrios, incluso con grandes diferencias en las métricas usuales (como son la recaudación total de dinero, y el bienestar social de los participantes). Encontrar un equilibrio arbitrario no es sencillo, y en Conitzer et al. [8] se demuestra que encontrar el equilibrio que maximiza esas métricas es un problema NP-Completo. Los autores proponen un Modelo de Programación Lineal Entera Mixta (PLEM, ver Capítulo 3) cuyas soluciones se corresponden con equilibrios óptimos. Si bien la formulación es correcta, su resolución mediante paquetes tradicionales se limita a instancias pequeñas con tan solo unos pocos anunciantes. En concreto, analizan instancias de entre 2 y 10 participantes, y cuando hay en juego solamente 8 objetos, son capaces de resolver menos de la mitad de las instancias que consideraron.

El objetivo de este trabajo es estudiar el problema de calcular efectivamente un *pacing equilibrium* (PE), entendiendo sus propiedades y comportamiento en distintos contextos de subastas así como también trabajar sobre el modelo de PLEM propuesto en Conitzer et al. [8] para desarrollar un algoritmo capaz de abordar instancias de mayor dificultad que las consideradas hasta el momento.

1.2 Trabajos Relacionados

El problema de asignación de espacios publicitarios online es muy estudiado en la actualidad. En Choi et al. [5] se presenta una extensa revisión de literatura al respecto. Una de sus áreas de investigación es la de administración de presupuestos, buscando estrategias para evitar que el dinero de los participantes se agote prematuramente. En este sentido podemos hacer referencia a los trabajos de Karande et al. [12] y Charles et al. [4]. Existen dos grandes grupos de propuestas al respecto: las que se enfocan en selección de subastas, que limitan el conjunto de ítems por los que oferta cada anunciante, y las de modificación de apuestas, que buscan estrategias para disminuir los precios ofertados. En Balseiro et al. [3] se comparan algunas de esas estrategias, en particular la que denominan *bid shading*.

Conitzer et al. [8] consideran esta última estrategia y definen al mercado como un *pacing game*, un juego en el cual los anunciantes son los jugadores y sus estrategias posibles son el factor por el cual reducir sus apuestas. En este caso, los ítems son repartidos mediante subastas de sobre cerrado de segundo precio, propuestas en Vickrey [16]. Definen los *pacing equilibriums* (PE) como los equilibrios para dichos juegos, y formulan un PLEM cuyos puntos factibles se corresponden con los PE. En un trabajo posterior, Conitzer et al. [7] analizan los *first-price pacing equilibriums*, similares a los anteriores pero considerando subastas de primer precio. Para este problema muestran que se puede computar sus soluciones de manera eficiente, siendo posible modelarlo como un problema de optimización convexa y utilizando el algoritmo de Eisemberg y Gale [10]. En Balseiro et al. [1], por su parte, se introduce otra noción distinta de equilibrios, a la vez que métodos para encontrarlos.

En Balseiro y Gur [2] se consideran distintas técnicas para encontrar equilibrios, don-

de las estrategias de los jugadores se adaptan dinámicamente. Una de ellas es utilizada por Conitzer et al. [8], donde se muestra que los datos iniciales juegan un papel muy importante en la calidad de sus resultados. Por otro lado, en Kroer et al. [14] se proponen esquemas para solución de instancias de tamaños reales, mediante la resolución de una versión compactada del problema y luego un *lifting* para obtener asignaciones en la instancia original.

1.3 Estructura de la tesis

En el Capítulo 2 introducimos formalmente el problema, e incluimos ejemplos que ilustran escenarios posibles. Mencionamos algunas propiedades de los PE y la complejidad de encontrarlos. En el Capítulo 3 presentamos los Modelos de Programación Lineal y Programación Lineal Entera Mixta y los algoritmos usados en la práctica para resolverlos, mientras que en el Capítulo 4 efectivamente modelamos el problema de encontrar un PE como un PLEM.

Luego, en el Capítulo 5, analizamos características del problema en función de los datos de entrada. Mencionamos los mecanismos de generación de instancias para evaluar el modelo, y las clasificamos de acuerdo a las propiedades de sus PE, evaluando cómo afectan a la performance de los algoritmos de resolución. Mostramos ejemplos particulares que nos dieron una intuición sobre la dificultad de cada tipo de instancia, y corroboramos dichas intuiciones experimentalmente.

En el Capítulo 6 proponemos diversas mejoras al modelo. En las secciones 6.1 y 6.2 planteamos la incorporación al modelo de algunas familias de desigualdades válidas. En la Sección 6.3 investigamos agregar nuevas variables que permitan explotar algunas situaciones presentes en los equilibrios, como por ejemplo empates, y en 6.4 consideramos distintas familias de cortes que sirven de cota superior para la cantidad de ganadores que puede haber. En 6.5 proponemos una familia de desigualdades válidas que relacionan los factores α_i de los jugadores según si logran o no ganar algún ítem. En 6.6, utilizamos información de las valuaciones para deducir y fijar los valores que tienen que tomar algunas variables. En 6.7 mencionamos el trabajo de guiar al algoritmo de resolución en su búsqueda de soluciones factibles. Consideramos elección de parámetros del mismo, estrategias de prioridad de variables sobre las cuales hacer *branching* y la inclusión de heurísticas de Feasibility Pump.

Finalmente, en el Capítulo 7 evaluamos cómo impactan todas ellas en la performance del *solver* al buscar un PE, y en el Capítulo 8 presentamos nuestras conclusiones y líneas de investigación a futuro.

2. PACING EQUILIBRIUMS

2.1 Preliminares

A lo largo de este trabajo vamos a considerar un sistema con un conjunto $N = \{1 \dots n\}$ de *bidders* o anunciantes y un conjunto $M = \{1 \dots m\}$ de ítems divisibles a asignar entre ellos. Para cada par $(i, j) \in N \times M$ definimos $v_{ij} \in \mathbb{R}_+$ a la valuación del *bidder* i por el ítem j. Desde la perspectiva de dicho *bidder*, conseguir ese ítem por un precio p_j menor a v_{ij} le generará una ganancia de $v_{ij} - p_j$, es decir igual a la diferencia entre su valuación por él y el precio pagado.

Diremos que $B_i > 0$ es el presupuesto del *bidder* i. Esta es la cantidad máxima de dinero a la que se compromete pagar dicho anunciante en total, y por ende no debemos permitir que nuestra solución implique que alguno consuma más que ese presupuesto.

Vamos a distribuir cada producto mediante una subasta de sobre cerrado de segundo precio. Como vimos, dada una única subasta, la estrategia óptima de cada participante es apostar la valuación real que tiene del ítem; no puede conseguir mayor ganancia realizando una apuesta distinta. En caso de haber un empate en las apuestas ganadoras por un ítem, el precio estará determinado por la apuesta más alta. Además, en ese caso, cada anunciante ganador se llevará una proporción de ese producto a determinar por el organizador.

El objetivo es calcular, para cada anunciante, un *multiplicador de pacing* $\alpha_i \in [0, 1]$ que va a reescalar uniformemente todas las apuestas que vaya a realizar. De este modo, la *apuesta reescalada* del participante i por el ítem j será $\alpha_i v_{ij}$. Uno puede entonces analizar este sistema como un juego, donde los anunciantes son los jugadores y sus estrategias son la elección de cada $\alpha_i \in [0, 1]$. Es un juego simultáneo, pues todos eligen su estrategia sin conocer la del resto, y no es de suma cero (recordar el ejemplo de la Figura 1). En este juego, podemos interpretar a las subastas sucediendo todas en simultáneo. Será responsabilidad de cada jugador el elegir un multiplicador de pacing lo suficientemente pequeño como para no exceder su presupuesto.

Como en todo juego, es de interés encontrar equilibrios, es decir aquellas situaciones en las que ningún jugador puede obtener mayor beneficio modificando individualmente su estrategia. Vamos a considerar entonces la siguiente definición.

Definición 1. *Definimos un pacing equilibrium (PE) como una asignación de:*

- *Multiplicadores de pacing* $\alpha_i \in [0, 1]$ *para cada jugador,*
- *Precios* $p_j \in \mathbb{R}_+$ *para cada ítem*
- *Una fracción* $x_{ij} \in [0, 1]$ *del ítem* j *asignada al jugador* i

de manera tal que se satisfagan simultáneamente las siguientes condiciones:

1. *Para cada ítem j, $\sum_{i \in N} x_{ij} = 1$.*
   Es decir, cada ítem se reparte íntegramente entre los jugadores.
2. *$x_{ij} > 0$ únicamente si $\alpha_i v_{ij} = \max\{\alpha_k v_{kj} : k \in N\}$.*
   O sea, solo recibe parte del ítem aquel (o aquellos empatados) con la apuesta más alta.
3. *Si $x_{ij} > 0$, entonces $p_j = \max\{\alpha_k v_{kj} : k \in N \setminus \{i\}\}$. El precio lo fija la segunda apuesta más alta (que, como dijimos, puede coincidir con la apuesta más alta).*
4. *$\sum_{j \in M} s_{ij} \leq B_i$, donde $s_{ij} := x_{ij} p_j$. Acá, s_{ij} es el gasto del jugador i por el ítem j, y la condición expresa que el gasto total de un jugador no puede exceder su presupuesto.*
5. *Si la desigualdad en 4. es estricta para algún i, entonces $\alpha_i = 1$.*

Formalmente, notamos un PE como $E := (\{\alpha_i\}_{i \in N}, \{p_j\}_{j \in M}, \{x_{ij}\}_{i \in N, j \in M})$

Es importante notar que un PE requiere tanto un vector de multiplicadores de pacing como una asignación de los ítems que respete los presupuestos y el precio fijado por la segunda apuesta más alta.

Una pregunta que surge naturalmente es si dicha situación es efectivamente un equilibrio. Es decir, si dada esa asignación, la estrategia de todo jugador es óptima como respuesta a las demás. ¿Será posible que, dada una situación como la mencionada, uno de los jugadores reciba mayor beneficio si cambia su estrategia? Si esto sucediera, no sería correcto referirnos a esta situación como equilibrio. En efecto, vale el siguiente resultado:

Proposición 1 (Conitzer, et al. [8])**.** *Dado un PE, el perfil de estrategias $\alpha = (\alpha_1, \ldots \alpha_n)$ constituye un equilibrio. Es decir, para cada jugador i, su multiplicador de pacing α_i es óptimo como respuesta a los demás.*

2.2 Ejemplos

En esta sección presentamos algunos ejemplos que ilustran diversos juegos, situaciones particulares, y sus respectivos equilibrios.

Ejemplo 1. *Consideremos el sistema de la Figura 2. Aquí el jugador 2 nunca va a agotar todo su presupuesto, por lo que su multiplicador de pacing será $\alpha_2 = 1$. Fijado ese multiplicador, el precio del ítem a no va a ser superior a $\alpha_2 v_{2a} = 1$, por lo que el primer jugador tampoco va a gastar todo su presupuesto. De haber un equilibrio de pacing, tendría que respetar entonces $\alpha_1 = \alpha_2 = 1$, y en ese caso el jugador 1 va a apostar 4 por el ítem a, y ganarlo a un precio de 1 (lo que le genera una ganancia de $4 - 1 = 3$).*

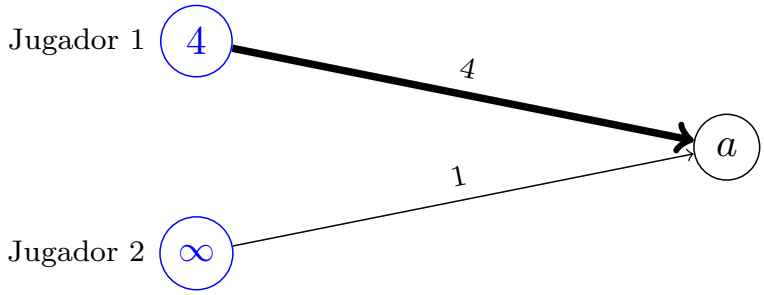


Fig. 2: Ejemplo 1

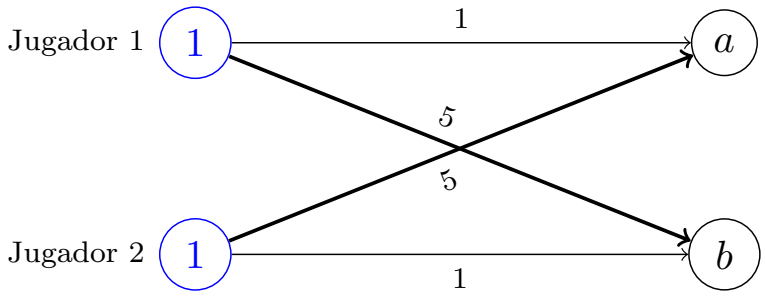
¿Qué habría pasado si el presupuesto del jugador 1 hubiera sido de 0.99? Como antes, $\alpha_2 = 1$, pero ahora tiene que cumplirse $\alpha_1 \leq \frac{1}{4}$, pues si fuera mayor, el jugador 1 ganaría la totalidad del ítem a y tendría que pagar 1, ¡y eso es más de lo que se comprometió a gastar! Entonces, como $\alpha_1 < 1$, la condición 5 implica que el jugador 1 tiene que gastar todo su presupuesto, por lo que tampoco puede apostar menos que el jugador 2 por el único ítem en disputa (ya que en tal caso gastaría un total de 0). Por lo tanto, necesariamente tiene que tener $\alpha_1 = \frac{1}{4}$ y así se constituye el único equilibrio posible. Ambas apuestas resultan iguales a 1, por lo que ese también será el precio del ítem. Como el jugador 1 tiene que gastar todo su presupuesto, la plataforma le asigna 0.99 del ítem y lo restante al jugador 2.

Ejemplo 2. *Volvamos ahora el ejemplo de la Figura 1, que causaba problemas con las subastas secuenciales. ¿Cómo sería un equilibrio para este sistema? Proponemos los siguientes multiplicadores de pacing:* $\alpha_1 = 1$, $\alpha_2 = \frac{1.1}{100} = 0.011$. *Con ellos, los precios de los ítems resultan, respectivamente,* 1.1 *(que coincide con ambas apuestas reescaladas, pues hubo un empate) y* 0.011 *(fijado por la apuesta del jugador* 2*, quien perdió). Así, el ítem 2 se asigna totalmente al jugador* 1*, y el ítem 1 se reparte, teniendo en cuenta que el jugador* 2 *tiene que gastar todo su presupuesto. Como el precio resultó* 1.1*, el jugador* 2 *gastará* 1 *y el otro los restantes* 0.1*.*

Como observación, ahora la recaudación de la plataforma fue $1.1 + 0.011 = 1.111$*, contra los* 1.1 *que habría recaudado con el método de subastas secuenciales.*

Se puede ver que esta situación es claramente mucho más beneficiosa que la original: el jugador con más presupuesto ganó el ítem que más quería (e incluso parte del otro), y el otro participante igualmente pudo hacerse con gran parte del que él valoraba más. Es la situación ideal que mencionábamos antes, con ambos anunciantes recibiendo mayor rédito (y no a costa de la plataforma, pues ésta incluso terminó con mayor recaudación).

Ejemplo 3 (Puede haber múltiples equilibrios)**.** *Consideremos la siguiente situación:*



Aquí podemos chequear fácilmente que se cumplen las condiciones de un equilibrio si elegimos $\alpha_1 = \frac{1}{5}$ y $\alpha_2 = 1$, con el ítem a para el jugador 2 por un precio de $\frac{1}{5}$, y el b (que resultó con un precio de 1) asignado en su totalidad al jugador 1 para que agote su presupuesto. Ahora bien, como la situación es simétrica, uno podría invertir los roles y obtener otro equilibrio, análogo pero distinto.

Vale la pena mencionar que, si bien por simplicidad mostramos este ejemplo, existen situaciones con múltiples equilibrios en las que no hay simetría en los jugadores.

2.3 Propiedades

Al analizar la Definición 1 de un PE, una de las condiciones parece arbitraria, o al menos que no fue discutida en profundidad. Nos referimos a la condición 5, que fija un valor para el multiplicador de pacing de los jugadores i que cumplen $\sum_{j \in M} s_{ij} < B_i$. Lejos de arbitraria, hay dos grandes motivos para considerarla.

- En primer lugar, resulta razonable desde el punto de vista de los jugadores que sus apuestas no sean reescaladas si no resultaron con problemas de presupuesto.
- Además, de no agregar esta condición surgen gran cantidad de equilibrios que perjudican a la plataforma injustamente.

  Por ejemplo, dado un equilibrio $E = (\{\alpha_i\}_{i \in N}, \{p_j\}_{j \in M}, \{x_{ij}\}_{i \in N, j \in M})$, podemos encontrar para cada $\lambda \in [0, 1)$ otro $E_\lambda = (\{\lambda\alpha_i\}_{i \in N}, \{\lambda p_j\}_{j \in M}, \{x_{ij}\}_{i \in N, j \in M})$ que también resulte factible, pero con el precio de cada ítem disminuido por ese factor.

Más aún, para todo juego existe un equilibrio trivial, tomando $\alpha = (0, \ldots, 0)$, que, si bien cumple las primeras cuatro condiciones, evidentemente no resulta de ninguna utilidad. Informalmente, la condición 5 sirve como contención del *pacing*, evitando que los jugadores reescalen demasiado sus apuestas (y situaciones como el equilibrio trivial mencionado).

Una pregunta que surge naturalmente a esta altura es si, a pesar de no haber unicidad, podemos encontrar hipótesis que garanticen existencia de al menos un *pacing equilibrium*.

Es claro que sí si nos olvidamos de la última condición, pero ya vimos que dichas soluciones pueden ser poco razonables. Se puede ver que la respuesta a este interrogante es positiva, y podemos enunciar el siguiente teorema.

Teorema 2 (Existencia de PE, Conitzer et al. [8])**.** *Siempre existe un Pacing Equilibrium.*

2.4 Complejidad de encontrar un Pacing Equilibrium

Sabemos que tenemos garantizada la existencia de un equilibrio. ¿Cuán complejo será encontrar uno? ¿Y encontrar aquel que sea óptimo respecto a cierta métrica? Para ello, introducimos las métricas que vamos a analizar.

Definición 2. *Definimos el* ***revenue*** *de una solución al juego como la cantidad de dinero recaudada por la plataforma,* $z_{rev} := \sum_{i \in N} \sum_{j \in M} s_{ij}$.

Observación 1. *En un equilibrio,* $z_{rev} = \sum_{j \in M} p_j$.

Así como se analiza el *revenue* como beneficio a la plataforma, estaría bueno tener una métrica que considere el beneficio obtenido por los anunciantes.

Definición 3. *El* ***social welfare*** *se define como la suma de las valuaciones ganadoras, es decir* $z_{sw} := \sum_{i \in N} \sum_{j \in M} x_{ij} v_{ij}$.

Si bien esta última suena razonable, tiene el problema de no ser robusta, siendo sensible a los cambios de escala. Por ejemplo, duplicar todas las valuaciones no modifica el resultado de las subastas, pero genera un incremento artificial en el *social welfare*. Es por ello que se introduce una medida del bienestar social que es efectivamente robusta frente a este tipo de cambios.

Definición 4. *El* ***paced welfare*** *de un equilibrio es* $z_{pw} := \sum_{i \in N} \sum_{j \in M} x_{ij} \alpha_i v_{ij}$. *Se corresponde con la suma de las valuaciones reescaladas ganadoras.*

Dada la robustez de esta última métrica respecto a la anterior, nos referiremos a ella simplemente como *welfare*, ignorando la que no toma en cuenta el reescalado de las apuestas.

Volviendo entonces a la pregunta original, ¿cuán difícil será encontrar un equilibrio que maximice el *revenue* o el *welfare*? El siguiente resultado responde esta pregunta.

Teorema 3 (Conitzer et al. [8])**.** *Los problemas de encontrar un PE que maximice revenue, social welfare o welfare son NP-Completo.*

Vale destacar que la complejidad del problema de factibilidad (es decir, el de encontrar algún equilibrio) aún permanece como una pregunta abierta. No podemos afirmar que sea un problema sencillo. En el Capítulo 4 se plantea un modelo de programación lineal entera mixta (PLEM) cuyos puntos factibles coinciden con los equilibrios buscados. Como mencionamos, en este trabajo nos enfocamos en analizar y reforzar dicho modelo.

3. PROGRAMACIÓN LINEAL ENTERA

3.1 Programación Lineal

La Programación Lineal (PL) permite expresar modelos para encontrar una valuación en $\mathbb{R}_+$ para un conjunto de variables x, de manera tal que se maximice (o minimice) una función objetivo $z : \mathbb{R}^n \to \mathbb{R}$ y que se cumpla un conjunto de desigualdades lineales $Ax \leq b$ ($A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$). Todo problema de PL puede escribirse de la siguiente forma:

$$\begin{array}{ll} \textbf{max} & z(x) = cx \\ \textbf{sujeto a} & \\ & Ax \leq b \\ & x_i \geq 0 \quad \forall i = 1 \dots n \end{array}$$

A su vez, esto tiene una interpretación geométrica que se basa en entender al grupo de restricciones $Ax \leq b$ como hiperplanos que definen a un poliedro en $\mathbb{R}^n_+$. A modo de ejemplo, en la Figura 3 se grafica el siguiente Modelo PL:

$$\begin{array}{ll} \textbf{max} & z(x,y) = x + y \\ \textbf{sujeto a} & \\ & \frac{1}{3}x - y \leq -2 \\ & 2x - y \leq 5{,}5 \\ & \frac{1}{2}x + y \geq 2 \\ & 2x - y \geq 0{,}5 \\ & x, y \in \mathbb{R}_+ \end{array} \tag{3.1}$$

Existen algoritmos que encuentran una solución óptima para cualquier Modelo PL en tiempo polinomial (e.g. ver Karmarkar [13]). Además, existen otros algoritmos que si bien no son polinomiales funcionan bien en la práctica como es el caso de Simplex, algoritmo propuesto por Dantzig [6,9]. Hoy en día existe una variedad de paquetes que implementan estos algoritmos de manera eficiente tales como CPLEX, AMPL, Gurobi, etc.

3.2 Programación Lineal Entera Mixta

Si bien PL permite modelar una gran variedad de problemas, hay muchos que no pueden ser representados utilizando solamente variables continuas, y necesitan restringir el dominio de algunas de las variables a los enteros ($\mathbb{Z}$). Estos problemas se pueden modelar

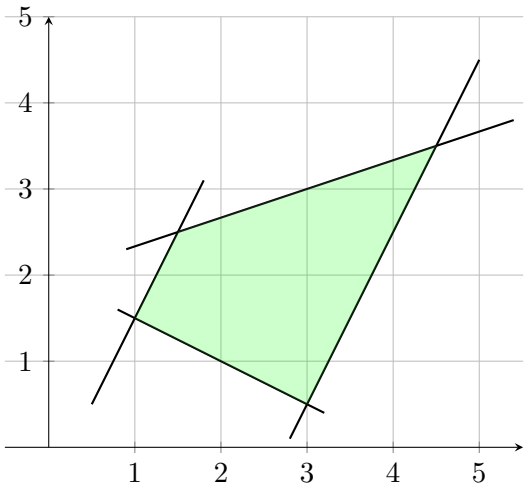


Fig. 3: Ejemplo de poliedro descripto por sistema de ecuaciones PL 3.1.

mediante los llamados Modelos de Programación Lineal Entera (PLEM). Un PLEM tiene la siguiente estructura:

$$\begin{array}{lll} \textbf{max} & z(x) = cx & \\ \textbf{sujeto a} & & \\ & Ax \leq b & \\ & x_i \geq 0 \quad & \forall i \in C \cup I \\ & x_i \in \mathbb{Z} \quad & \forall i \in I \end{array} \tag{3.2}$$

Donde $I \dot{\cup} C = \{1, \ldots, n\}$, siendo I el conjunto de las variables que solo pueden tomar valores enteros, y C aquellas que pueden tomar cualquier valor real. Si llamamos $P = \{x \in \mathbb{R}^n \mid Ax \leq b, x \geq 0\}$ al poliedro que describe el modelo, el conjunto de soluciones factibles es $S = \{x \in P \mid x_i \in \mathbb{Z}\ \forall i \in I\}$. Además, decimos que $\bar{x} = \text{máx}\{z(x) \mid x \in P\}$ es la relajación lineal de P. Y como $S \subseteq P$, entonces se puede afirmar que la solución óptima x^* de S cumple que $z(x^*) \leq z(\bar{x})$, es decir, la relajación del PLEM es una cota superior de la solución óptima dado que estamos considerando un problema de maximización. En un problema de minimización vale, de manera análoga, que $z(\bar{x})$ es una cota inferior para la solución óptima de S, es decir $z(x^*) \geq z(\bar{x})$.

Desde el punto de vista geométrico, se llama $conv(S)$ a la cápsula convexa de S, o sea, el poliedro más chico que contiene a todos los puntos de S. Si $conv(S)$ se puede representar con un conjunto polinomial de restricciones, entonces se puede hallar la solución óptima utilizando cualquier algoritmo que resuelva PL. En la Figura 4 se puede ver la representación gráfica del PLEM de ejemplo que se presenta a continuación, donde P está

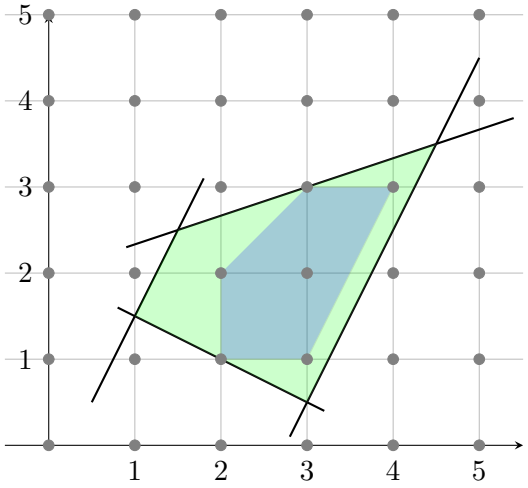


Fig. 4: Ejemplo de poliedro descripto por sistema de ecuaciones PLEM 3.2.

de color verde y $conv(S)$ de color azul:

$$
\begin{array}{ll}
\textbf{max} & z(x,y) = x + y \\
\textbf{sujeto a} & \\
& \frac{1}{3}x - y \leq -2 \\
& 2x - y \leq 5{,}5 \\
& \frac{1}{2}x + y \geq 2 \\
& 2x - y \geq 0{,}5 \\
& x, y \in \mathbb{Z}
\end{array}
$$

El problema de resolver un PLEM general pertenece a la clase de complejidad NP-Hard, ya que varios problemas que también están en NP-Hard se pueden modelar como PLEM. Como mencionamos anteriormente, se puede probar que si $P = conv(S)$ y se tiene una cantidad polinomial de restricciones para representar a P, entonces se puede utilizar cualquier algoritmo de PL para encontrar la solución óptima de S. Sin embargo, no se sabe si encontrar la cápsula convexa de un conjunto cualquiera S es polinomial, o si la cantidad de restricciones para representarla lo es. A continuación, se presentan las técnicas más frecuentes para resolver PLEM.

3.3 Algoritmos para PLEMs

3.3.1 Branch&Bound (B&B)

La primera estrategia para resolver PLEM se basa en una utilización de los algoritmos de PL. Cuando se aplica uno de estos algoritmos sobre un PLEM, la estrategia más común es relajar las restricciones que restringen el dominio de las variables a los enteros. Es por esto que la solución óptima que arroja el algoritmo puede contener alguna variable entera tomando un valor fraccionario, lo cual no es factible en el problema original. En estos casos

lo que se hace es aplicar la táctica Divide & Conquer y se divide el problema en dos o más subproblemas, de manera tal que la solución de la relajación no sea solución de ninguno de ellos pero a su vez no se pierda ninguna solución entera.

Comúnmente, para dividir el problema en subproblemas, se utiliza el denominado *branching* por Variable, que consiste en tomar una variable entera que tenga un valor fraccionario x_i^* en la solución de la relajación y crear dos subproblemas: uno agregando la condición $x_i \leq \lfloor x_i^* \rfloor$ y el otro agregando $x_i \geq \lceil x_i^* \rceil$. El criterio para elegir cual variable x_i tomar puede variar según cómo se va generando el árbol de ejecución, entre los más comunes están: aquella con parte fraccionaria más cercana a 0.5, más cercana a 0, y más cercana a 1.

Por otro lado, para evitar explorar ramas del árbol que no lleven a una solución óptima, se aplica el denominado *bounding*, que consiste en aplicar podas sobre el árbol basadas en cotas del funcional. Generalmente, durante la ejecución del algoritmo se mantiene una solución x_{UB} (x_{LB}) que es la mejor encontrada hasta el momento, y la cuál sirve de cota superior (inferior), si el problema es de minimización (maximización), la cual permite estando en un nodo dejar de explorarlo si la relajación vale $z(x^*) > z(x_{UB})$ ($z(x^*) < z(x_{LB})$). Esta cota también suele llamarse *cota primal*.

Otro de los parámetros que se puede ajustar es cómo se recorre el árbol de nodos que se va generando. Este puede ser recorrido de las maneras más convencionales como Depth First Search (DFS), lo cuál es particularmente conveniente para conseguir Cotas Primales de manera veloz y para no utilizar tanta memoria, o Best Bound First (BBF) que utiliza más memoria pero pretende minimizar el número de nodos explorados al elegir siempre el nodo con mejor *cota dual*, o sea, valor de la relajación.

3.3.2 Planos de Corte

Otra forma de abordar problemas con variables enteras o binarias, es mediante el agregado de *planos de corte*. Un plano de corte es una restricción que se cumple para toda solución en S, pero no se cumple para la solución óptima fraccionaria de la relajación del problema. Un esquema general que representa a los algoritmos de planos de corte se define en los siguientes pasos:

1. Resolver la relajación. Sea x^* el óptimo.
2. Si $x^* \in S$, terminar.
3. Encontrar $a'x \leq b'$ una restricción válida para todo $x \in S$ pero violada por x^* y agregarla a la formulación.
4. Volver al Paso 1.

Mediante esta técnica se busca ir acercándose a representar con restricciones lineales a $conv(S)$ ya que como se mencionó anteriormente, si se obtiene un PL que describa $conv(S)$ entonces va a tener una solución óptima que pertenece a S.

Una de las desventajas de esta técnica es la dificultad de encontrar planos de corte que se ajusten a la cápsula convexa de S, ya sea porque es computacionalmente ineficiente o por la dificultad teórica que esto implica. Es por esto que esta técnica no suele utilizarse por sí misma sino en conjunción con Branch&Bound.

3.3.3 Branch&Cut (B&C)

La técnica Branch&Cut es una combinación de las mencionadas previamente B&B y Planos de Corte. Este algoritmo surge a partir de la necesidad de reducir la enumeración de nodos al utilizar B&B. Para lograr esto, la idea es aplicar una cierta cantidad de iteraciones de planos de corte en los nodos del árbol para así lograr ajustar su cota dual y por lo tanto poder aplicar Bounding de manera más frecuente.

Esta técnica es particularmente interesante en la etapa de experimentación porque su eficiencia depende de un compromiso entre la cantidad de iteraciones de planos de corte y la cantidad de nodos que se enumeran. Esto debe ser ajustado ya que las iteraciones de plano de corte pueden ser muy costosas y a partir de cierto número la enumeración resulta más eficiente.

Uno de los parámetros a ajustar es la cantidad de iteraciones de planos de corte que se llevan a cabo en cada uno de los nodos y otro es la cantidad total de desigualdades que se agregan al modelo en cada iteración. Una estrategia que a veces se utiliza es realizar más iteraciones de plano de corte cuanto más cerca del nodo raíz se está. Esto en general ofrece un buen compromiso entre tiempo invertido en la separación de desigualdades y enumeración de nodos. En particular, cuando solamente se utilizan iteraciones de planos de corte en el nodo raíz, el algoritmo se conoce como Cut&Branch (C&B).

4. CÁLCULO DE PACING EQUILIBRIUMS VÍA PLEM

4.1 Modelo Original

En la Sección 2.4, dijimos que el problema de encontrar un PE no es sencillo. En la práctica, además de demostrar la existencia, suele ser importante tener algún procedimiento que permita calcular un PE. Como mencionamos, es posible formular un modelo de programación lineal entera mixta que permita representar nuestro problema; los equilibrios de pacing se corresponderán con puntos en la región factible, y, en caso de que haya múltiples equilibrios, uno puede elegir qué función objetivo quiere optimizar (si *revenue*, *social welfare*, *paced welfare* o alguna otra).

Vamos a analizar el modelo propuesto en Conitzer et al. [8]. En primer lugar, definimos las variables continuas

- $\alpha_i \in [0, 1]$: Multiplicador de *pacing* del jugador i.
- $s_{ij} \geq 0$: La cantidad efectivamente gastada por el jugador i en el ítem j.
- $p_j \geq 0$: El precio resultante del ítem j.
- $h_j \geq 0$: La apuesta más alta por el ítem j

y las variables enteras

- $d_{ij} \in \{0, 1\}$: Toma valor 1 si y solo si el jugador i ganó parte del ítem j.
- $y_i \in \{0, 1\}$: Toma valor 1 si y solo si el jugador i gastó todo su presupuesto.
- $w_{ij} \in \{0, 1\}$: Toma valor 1 si y solo si el jugador i es el "ganador"del ítem j.
- $r_{ij} \in \{0, 1\}$: Toma valor 1 si y solo si el jugador i resultó segundo por el ítem j.

Expandimos brevemente la explicación respecto a la semántica de las variables en el modelo resultante. Las variables d_{ij} serán 1 para todo jugador cuya apuesta sea la más alta (posiblemente empatada) por el ítem j. Podría suceder que en el reparto, la proporción que le corresponda a ese jugador sea 0, pero aún así lo consideramos dentro de los ganadores. Por otra parte, w_{ij} valdrá 1 para exactamente uno de esos jugadores por cada ítem; este será el proclamado "ganador" pero solo a fines de determinar el valor de la apuesta más alta, h_j. Similarmente, r_{ij} se usará para determinar el precio, eligiendo a alguno de los potencialmente varios jugadores con la segunda apuesta más alta. Algo importante de

mencionar a esta altura es que esto induce simetrías: van a existir soluciones *distintas* pero esencialmente equivalentes; esto lo vamos a estudiar en mayor profundidad en la Sección 6.2.

Por último, recordamos los datos de entrada del problema, $B_i > 0$ el presupuesto del jugador i y $v_{ij} \geq 0$ la valuación del jugador i por el ítem j, y definimos $\overline{v}_j$ en función de ellos:

$$\overline{v}_j := \max_{i \in N} v_{ij}$$

Con estas variables podemos formular las ecuaciones que definen un PE. Como mencionamos antes, en el modelo armado los equilibrios se corresponden con puntos factibles del PLEM, representados por los puntos dentro del siguiente poliedro.

$$\sum_{j \in M} s_{ij} \leq B_i \qquad (\forall i \in N) \tag{4.1}$$

$$\sum_{j \in M} s_{ij} \geq y_i B_i \qquad (\forall i \in N) \tag{4.2}$$

$$\alpha_i \geq 1 - y_i \qquad (\forall i \in N) \tag{4.3}$$

$$\sum_{i \in N} s_{ij} = p_j \qquad (\forall j \in M) \tag{4.4}$$

$$s_{ij} \leq B_i d_{ij} \qquad (\forall i \in N, \forall j \in M) \tag{4.5}$$

$$h_j \geq \alpha_i v_{ij} \qquad (\forall i \in N, \forall j \in M) \tag{4.6}$$

$$h_j \leq \alpha_i v_{ij} + (1 - d_{ij})\overline{v}_j \qquad (\forall i \in N, \forall j \in M) \tag{4.7}$$

$$w_{ij} \leq d_{ij} \qquad (\forall i \in N, \forall j \in M) \tag{4.8}$$

$$p_j \geq \alpha_i v_{ij} - w_{ij} v_{ij} \qquad (\forall i \in N, \forall j \in M) \tag{4.9}$$

$$p_j \leq \alpha_i v_{ij} + (1 - r_{ij})\overline{v}_j \qquad (\forall i \in N, \forall j \in M) \tag{4.10}$$

$$\sum_{i \in N} w_{ij} = 1 \qquad (\forall j \in M) \tag{4.11}$$

$$\sum_{i \in N} r_{ij} = 1 \qquad (\forall j \in M) \tag{4.12}$$

$$r_{ij} + w_{ij} \leq 1 \qquad (\forall i \in N, \forall j \in M) \tag{4.13}$$

$$\alpha_i \in [0, 1] \qquad (\forall i \in N) \tag{4.14}$$

$$s_{ij} \geq 0 \qquad (\forall i \in N, \forall j \in M) \tag{4.15}$$

$$p_j \geq 0 \qquad (\forall j \in M) \tag{4.16}$$

$$h_j \geq 0 \qquad (\forall j \in M) \tag{4.17}$$

$$d_{ij} \in \{0, 1\} \qquad (\forall i \in N, \forall j \in M) \tag{4.18}$$

$$y_i \in \{0, 1\} \qquad (\forall i \in N) \tag{4.19}$$

$$w_{ij} \in \{0, 1\} \qquad (\forall i \in N, \forall j \in M) \tag{4.20}$$

$$r_{ij} \in \{0, 1\} \qquad (\forall i \in N, \forall j \in M) \tag{4.21}$$

La condición (4.1) garantiza que nadie gaste más que su presupuesto, mientras que la (4.2) refleja lo pedido por las variables y_i, formalizando qué significa que alguien gaste todo su presupuesto. La restricción (4.3) refleja la precondición de que si a alguien le sobra presupuesto, entonces no puede haber reescalado sus apuestas. Mediante (4.4) garantizamos que todo producto se asignó en su totalidad, y con (4.5) que solo los ganadores puedan obtener parte del mismo. La apuesta más alta es fijada por (4.6) y (4.7), mientras que (4.9) y (4.10) fijan el precio como la segunda. Finalmente, (4.8), (4.11), (4.12) y (4.13) dicen que hay exactamente un ganador y un segundo por cada producto, que son distintos, y que el ganador tiene que ser alguno de los que gana parte del ítem.

Es importante notar que, al corresponderse los pacing equilibriums con los puntos en la región factible, no importa cuál es la función a optimizar. Uno podría elegir una función constante, la cual garantiza que todo punto factible es óptimo, o alguna que priorice cierto tipo de equilibrios por sobre otros. En este sentido, una opción es encontrar el que maximiza (o minimiza) z_{rev}, z_{pw} u otros. Incluir una función objetivo podría ayudar computacionalmente a encontrar un equilibrio, pero potencialmente perjudicar la prueba de optimalidad. En todos los casos $z_{\_}$ refiere al valor óptimo de la función objetivo, y llamaremos $z_{\_}^{LP}$ al valor óptimo de la relajación lineal del modelo (4.1)–(4.21), es decir, relajando las condiciones de integralidad.

4.2 Primeros interrogantes

Al comenzar a analizar el problema notamos que, efectivamente, es difícil encontrar soluciones para este modelo, aún para tamaños pequeños. Una primera explicación que le encontramos es que el modelo posee una cantidad muy reducida de puntos en la región factible, lo cual puede afectar a la performance de los paquetes de resolución, si no son capaces de identificar algún punto factible. Cuando el problema incluya una función objetivo no constante, el aporte de un algoritmo de B&B se va a ver perjudicado si no puede encontrar soluciones factibles que le permitan eliminar por optimalidad algunas ramas del árbol de exploración del espacio de soluciones.

En ese mismo sentido, ¿en qué contextos habrá multiplicidad de equilibrios? Es claro que si existe un único equilibrio, puede darse la situación que el paquete de resolución lo encuentre, pero no sea capaz de probar dentro del tiempo límite que es óptimo respecto a alguna de las métricas.

Por otro lado, está la situación de las simetrías. Dado un empate en el primer lugar por un ítem, tenemos varias combinaciones de las variables w_{ij} y r_{ij} que inducen puntos factibles distintos, pero que representan el mismo equilibrio. Y esto dispara otro interrogante adicional: ¿sucederán, en la práctica, dichos empates? Si la respuesta es que en general no, ¿cuán particular tiene que ser una instancia para que suceda alguno? Y si la respuesta es que sí, entonces, ¿con qué frecuencia?

A priori, uno podría intuir que en un contexto real es posible que varios anunciantes ofrezcan la misma valuación por un ítem. Las valuaciones no son el valor *real* que tiene un espacio publicitario para el participante, sino una estimación que el mismo realiza sobre este. A modo de ejemplo, dos anunciantes cuyas valuaciones reales de un ítem son \$0.987 y \$1.004 respectivamente, podrían ofrecer la misma valuación de \$1, tanto por desconocimiento del valor real, como por naturaleza del comportamiento humano, que tiende a preferir los números enteros. Y lo mismo podría pasar en el otro eje: un anunciante podría proveer valuaciones iguales para diversos ítems. Dada esta situación, la intuición nos guía a pensar que los empates pueden ser factibles. Sin embargo, esto no resultó ser un factor determinante en la aparición de empates. En instancias con valuaciones aleatorias (provenientes de distribuciones continuas e independientes) pudimos observar que los empates resultaron sumamente frecuentes. En la Sección 6.3 analizamos los empates en detalle, y proveeremos intuiciones sobre por qué suceden.

Y siguiendo esa línea, ¿qué instancias será sensato analizar? En Conitzer et al. [8] utilizan algunas familias de instancias aleatorias generadas ad-hoc. ¿Serán representativas de la realidad? ¿Afectará la elección de dichas distribuciones a la dificultad para encontrar un equilibrio? En el próximo capítulo, intentamos responder esta última pregunta.

5. ANÁLISIS DEL MODELO SEGÚN EL TIPO DE INSTANCIA

5.1 Instancias

Inicialmente, para analizar la performance del modelo, utilizamos instancias artificiales creadas siguiendo la metodología propuesta en Conitzer et al. [8]. En concreto, analizamos dos tipos de instancias: las llamadas *Completas*, donde todos los *bidders* tienen valuación positiva para todos los objetos, y las *Sampled*, en las cuales solo están interesados en un subconjunto de los mismos.

Para armar las *Completas*, cada valuación v_{ij} se elige de manera independiente e idénticamente distribuida como una variable aleatoria Unif[0,1]. Luego, el presupuesto B_i para el *bidder* i se muestrea de una Unif $\left[0, \frac{1}{n}\sum_{j=1}^{m} v_{ij}\right]$. Si bien una elección más natural podría haber sido tomar como extremo superior el promedio de las valuaciones (es decir, dividir por m en vez de por n), decidimos respetar la elección original de los autores para poder comparar más fielmente los resultados.

Para las instancias *Sampled*, el procedimiento es el siguiente: primero elegimos para cada uno de los *bidders* un subconjunto $V_i \subseteq M$ de ítems por los que va a estar interesado, y luego procedemos como en el caso anterior pero únicamente sobre ese subconjunto. A los efectos de este trabajo, decidimos para cada $j \in M$ su pertenencia o no a cada V_i con probabilidad 0.5 y de manera independiente de los demás. De esta manera, la cantidad esperada de valuaciones positivas en una instancia *Sampled* será $\frac{nm}{2}$.

Lo primero que podemos observar es que, dada esta mecánica de generación de instancias, los presupuestos pueden resultar muy pequeños respecto a las valuaciones, por cómo fueron elegidos los extremos de la distribución de la cual surgen. Además, algo parecido puede suceder con las valuaciones, pudiendo ser algunas arbitrariamente cercanas a cero. Es importante aclarar que, sin embargo, no podemos considerar que si un jugador tiene una valuación muy chica por un ítem, entonces no lo gana: podría pasar que, en un equilibrio, los restantes jugadores resulten con un multiplicador de pacing muy pequeño en relación al de éste, y que termine efectivamente ganándolo. Si este fuera el caso, al restringirle la posibilidad de ganarlo estaríamos eliminando un (y potencialmente el único) equilibrio.

Por otro lado, sería interesante entender si el mecanismo para generar las valuaciones y los presupuestos de una instancia afecta a la dificultad para encontrar un equilibrio. Si alguno de los anunciantes tiene un presupuesto muy pequeño, entonces probablemente no sea capaz de conseguir ningún ítem. Y si, por el contrario, fuera muy abultado, entonces probablemente le sobre presupuesto al final del día. ¿Será esto de utilidad para encontrar un equilibrio más rápidamente? ¿Hará falta introducir esta deducción de manera explícita en el modelo?

5.2 Propiedades

Intuitivamente, un *bidder* cuyo presupuesto es menor que una fracción de la suma de sus valuaciones no va a poder ganar todos aquellos ítems por los que tiene interés. Más aún, en las instancias consideradas observamos que los presupuestos usualmente no alcanzaban para siquiera una fracción de los ítems. Esto se ve reflejado en que en el equilibrio encontrado, todos terminaron disminuyendo sus presupuestos (es decir, con un multiplicador de pacing menor a 1). Esto motivó la siguiente definición:

Definición 5. *Decimos que una instancia tiene* ***presupuestos ajustados*** *si admite un equilibrio tal que* $y_i = 1 \ \forall i \in N$*. En ese caso, decimos que ese equilibrio es* ***ajustado****.*

Notemos que decir que una instancia tiene presupuestos ajustados nos habla de *uno* de sus equilibrios. Nada nos dice que no pueda tener otro equilibrio donde alguno de los *bidders* tenga un presupuesto holgado.

Algo interesante es analizar el problema de encontrar un equilibrio que maximice *revenue* con estas instancias. En primer lugar, cada *bidder* está restringido a no pagar más que el presupuesto al que se comprometió, esto es $\sum_{j \in M} s_{ij} \leq B_i$. Por lo tanto, también vale que $z_{rev} = \sum_{i \in N} \sum_{j \in M} s_{ij} \leq \sum_{i \in N} B_i$. Es decir que el *revenue* de un equilibrio no puede ser mayor que la suma de presupuestos. Para aquellos equilibrios ajustados (cuando los haya), este máximo se alcanza.

Observación 2. *Si una instancia tiene presupuestos ajustados, cualquier equilibrio ajustado maximiza revenue.*

Esta observación, a priori inocente, tiene implicaciones en la performance al intentar resolver el modelo. Esto es, en parte, consecuencia del siguiente resultado:

Proposición 4. *Si una instancia tiene presupuestos ajustados,* $z_{rev} = z_{rev}^{LP}$

Demostración. Tomemos una instancia con presupuestos ajustados. Por definición, ésta tiene un equilibrio ajustado, que como mencionamos maximiza *revenue*. Como en este equilibrio se cumple $y_i = 1 \ \forall i \in N$, las restricciones (4.1) y (4.2) implican que en dicho equilibrio, $z_{rev} = \sum_{i \in N} B_i$. Pero además, todo punto en la región factible de la relajación lineal tiene que cumplir (4.1), con lo cual $z_{rev}^{LP} \leq \sum_{i \in N} B_i$. Como siempre vale $z_{rev} \leq z_{rev}^{LP}$, podemos concluir que $z_{rev} = z_{rev}^{LP}$. □

5.3 Generación de instancias holgadas.

Hemos mencionado que las instancias estudiadas resultaron ser, en su gran mayoría, de presupuestos ajustados. Nos gustaría poder generar instancias que también incluyan jugadores con presupuestos holgados. Una opción es agregar un conjunto de jugadores con un presupuesto más grande que la suma de sus valuaciones. Sin embargo, esto tiene

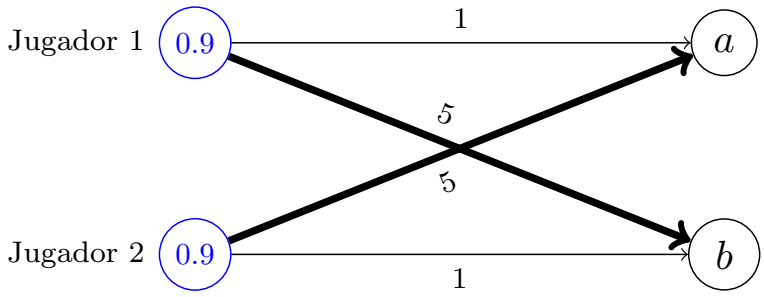
un problema que es que sus presupuestos son *demasiado* holgados, no dando lugar a una situación más real donde el dinero sobra porque el anunciante no pudo comprar todo lo que quería (y, aún sin *pacing* en sus apuestas, perdió algunos de los ítems).

Decidimos entonces tomar una de las instancias que generamos, e incrementarle el presupuesto de manera uniforme a todos. Introdujimos un nuevo parámetro $b > 0$ a nuestro problema, que será el factor por el cual vamos a multiplicar los presupuestos de **todos** los anunciantes. De este modo, no estamos priorizando ninguno de los anunciantes por sobre el resto, y podemos garantizar que si ese parámetro es lo suficientemente grande, todos los anunciantes van a resultar eventualmente holgados en los equilibrios. Llevado al extremo, si b es tan grande que los presupuestos de todos los jugadores superan a la suma de sus valuaciones, entonces en un equilibrio todos van a resultar holgados. Más aún, en este caso el equilibrio es evidente: todos los α_i valen 1, los precios están determinados por las apuestas, y los ítems pueden distribuirse de cualquier manera en caso de un empate.

Algo que observamos al reescalar los presupuestos de esta manera, utilizando parámetros $b_1 < b_2 < \dots$ es que una vez que un jugador pasa a ser holgado, no vuelve a resultar ajustado para un multiplicador mayor. En otras palabras, pareciera que el incrementar los presupuestos de todos los jugadores solamente puede generar nuevos jugadores holgados. Esto parece razonable, pues estamos incrementando el presupuesto de jugadores que ya les sobraba dinero, ¿por qué habrían de estar ajustados ahora?

Sin embargo, esto no es necesariamente cierto. Veamos el siguiente ejemplo:

Ejemplo 4. *Se tienen los siguientes jugadores, ítems y valuaciones:*



Se puede ver que se tiene un equilibrio con $\alpha_1 = 1$ *y* $\alpha_2 = \frac{1}{5}$*. Con ellos, el jugador 1 gana la totalidad del ítem b por* $\frac{1}{5}$*, y ambos empatan por el ítem a, que termina con un precio de* 1*. Como el jugador 2 resultó con pacing, tiene que gastar todo su presupuesto, por lo que se le asigna una proporción suficiente de dicho ítem.*

Ahora bien, ¿qué pasa si incrementamos los presupuestos, de 0.9 a 1? Volvimos a la situación del ejemplo 3, en el cual uno de sus equilibrios tiene al jugador 1 ajustado y al 2 holgado, ¡al revés que acá!

Por lo tanto, aunque empíricamente no lo observamos para ninguna de las instancias generadas, es posible que al incrementar los presupuestos alguno o algunos de los jugadores

pasen de ser holgados a ajustados.

5.4 Instancias Ajustadas

Comentamos que las instancias generadas resultaron casi en su totalidad con todos los jugadores con presupuestos ajustados. Consideremos un escenario donde una instancia no la podemos resolver, pero al duplicar los presupuestos *sí*, y encontramos un equilibrio ajustado $E = (\alpha, \{p_j\}_{j\in M}, \{x_{ij}\}_{i\in N, j\in M})$. Es fácil ver que $E_{\frac{1}{2}} = \left(\frac{\alpha}{2}, \{\frac{p_j}{2}\}_{j\in M}, \{x_{ij}\}_{i\in N, j\in M}\right)$ es un equilibrio para la instancia con los presupuestos originales. En efecto, todos apuestan la mitad por todos los ítems, por lo que los precios son la mitad y los ganadores son los mismos, y el reparto sigue siendo válido pues

$$2B_i = \sum_{j\in M} s_{ij} = \sum_{j\in M} x_{ij}p_j = 2\sum_{j\in M} x_{ij}\frac{p_j}{2}$$
$$\implies B_i = \sum_{j\in M} x_{ij}\frac{p_j}{2}$$

Lo que evidencia que en $E_{\frac{1}{2}}$ no solo no se exceden los presupuestos, sino que también se alcanzan, de modo que la condición 5 se cumple.

Así como recién duplicamos los presupuestos, también podríamos haberlos multiplicado por cualquier $\lambda > 0$. Sin embargo, si ese λ es tan grande que hace que aparezcan jugadores holgados, entonces ya no podremos deducir un equilibrio equivalente en la instancia original. Si tratáramos de hacer lo mismo, estaríamos disminuyendo el α_i de los holgados, forzándolos a gastar todo su presupuesto (lo cual no podemos garantizar, y en efecto probablemente no suceda).

Es importante mencionar es que este es un análisis que podemos hacer *a posteriori*: no tenemos manera de saber, dada una instancia, cuántos jugadores van a resultar ajustados en el equilibrio que encontremos.

5.5 Cantidad de jugadores ajustados al reescalar presupuesto

Es de interés entonces analizar cómo varía la cantidad de jugadores con presupuestos ajustados según el parámetro b por el cuál se reescala el presupuesto. Buscamos entender cómo es la relación (al menos para las instancias generadas) entre la magnitud de los presupuestos respecto a las valuaciones, y la cantidad de jugadores que resultan ajustados en el equilibrio encontrado. Para ello, generamos una instancia *Completa* de $n = 8$ jugadores y $m = 8$ ítems, y la intentamos resolver para valores de b entre 1 y 16 a intervalos de 0.5. Es una posibilidad que, para ciertos valores de b, las instancias sean muy complicadas para el software de resolución y no logremos obtener un equilibrio; no sabemos de antemano cuáles vamos a poder resolver y cuáles no.

Para resolver esta y el resto de las instancias, implementamos el modelo en el lenguaje de programación C++, utilizando CPLEX 12.9 [15] como paquete de resolución PL y PLEM con todos sus parámetros por defecto. Se estableció un tiempo límite por instancia de 10 minutos, y una función objetivo constante, de modo que termina si encuentra un equilibrio.

Veamos los resultados para esta instancia:

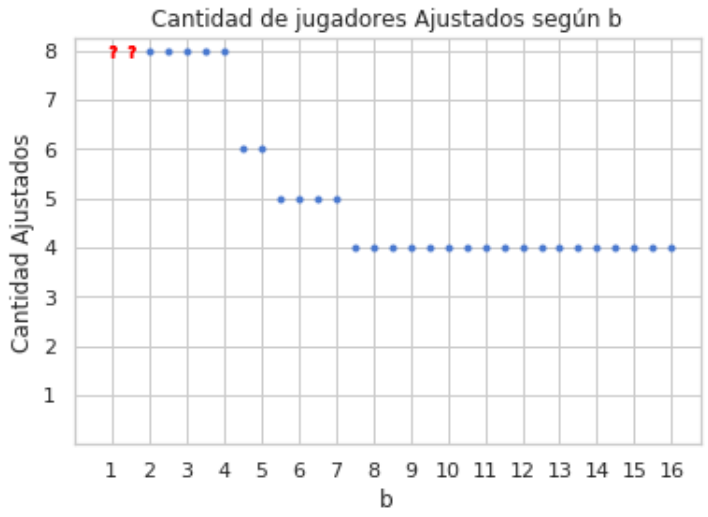


Fig. 5: Cantidad de jugadores ajustados al multiplicar los presupuestos por b

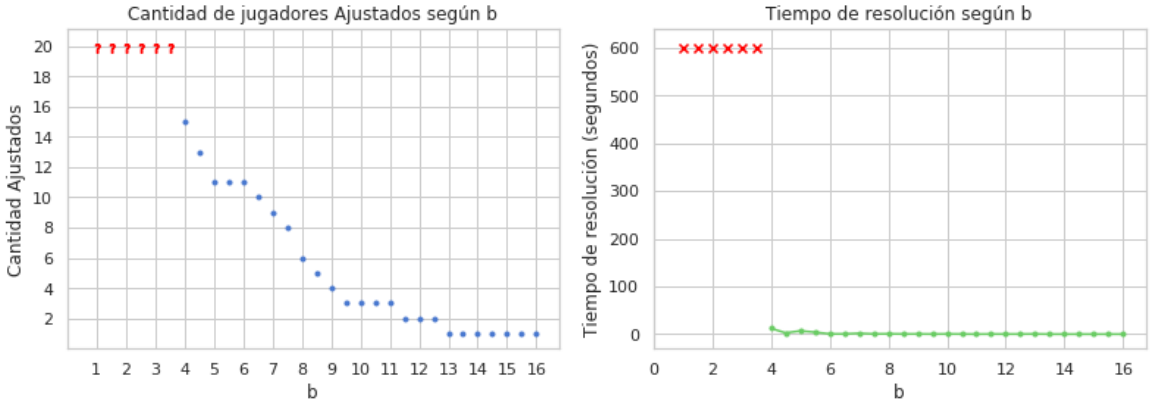
En la Figura 5 podemos ver los resultados. En azul la cantidad de jugadores que resultaron ajustados en los equilibrios encontrados, y en rojo los valores de b para los cuales no encontramos ningún equilibrio en el tiempo dado. Podemos ver que efectivamente la cantidad de jugadores ajustados decrece al aumentar los presupuestos, tal como esperábamos. Sucedió, además, que para los valores más altos de b que consideramos (hasta 16), siguió habiendo un 50 % de los jugadores ajustados.

Algo que notamos al observar los tiempos de ejecución es que la hipótesis de que el valor de b afectara la performance de resolución resultó cierta. En efecto, si mostramos los tiempos de resolución en función de dicho parámetro, se puede observar una clara tendencia:

Fig. 6: Tiempos de resolución de una instancia con $n = m = 8$, en función del parámetro b. En rojo, aquellos valores de b para los cuales no se encontró solución dentro del tiempo límite de 10 minutos.

Para la instancia original (es decir, con multiplicador $b = 1$) y con $b = 1.5$, el programa no logró encontrar ningún equilibrio dentro del tiempo límite de 10 minutos. Para todos los valores de b mayores a 2 fue posible encontrar una solución. Más aún, se puede observar una clara tendencia decreciente en los tiempos de resolución al aumentar el valor de dicho parámetro, siendo posible la resolución de manera casi inmediata para valores altos. Naturalmente, esto abre nuevas posibilidades: si aumentar el presupuesto de los jugadores provoca que las instancias sean resolubles en tiempos tanto más bajos, ¿podremos resolver, para ciertos b, instancias con mayor cantidad de jugadores y/o ítems?

Para responder esta pregunta, repetimos el mismo experimento, ahora para una instancia *Completa* con $n = m = 20$. Es importante notar que este es un tamaño bastante mayor que los que pueden resolver en el trabajo original.



(a) Cantidad de jugadores ajustados. *(b)* Tiempo hasta encontrar un equilibrio.

Fig. 7: Para la instancia de $n = m = 20$ analizada, cantidad de jugadores ajustados, y tiempo hasta encontrar un equilibrio, según el valor de b. En rojo, aquellos valores de b para los cuales no se encontró solución dentro del tiempo límite de 10 minutos.

Podemos observar que, a pesar de que no podemos resolver la instancia original (es decir, cuando $b = 1$), efectivamente a partir de $b = 4$ no solo podemos resolverla, sino que además en un tiempo mucho más pequeño, inferior a 10 segundos. Y a partir de $b = 6$, todas fueron resueltas en un tiempo menor a un segundo.

Para los presupuestos correspondientes al primer b para el cual pudimos encontrar un PE, resultaron 15 jugadores ajustados, un 75 % del total de participantes; al aumentar el b este número decrece, hasta llegar a un solo jugador ajustado cuando $b \geq 13$. Es importante mencionar que, para los valores de b entre 1 y 3.5, al no haber encontrado un PE, no podemos saber cuántos jugadores resultan ajustados en él.

5.6 Dificultad según cantidad de jugadores con presupuestos Holgados

¿Se mantendrá esta tendencia para tamaños más grandes, y consistentemente para varias instancias? Para intentar responder esto, planteamos un nuevo experimento. Vamos a generar instancias de mayor cantidad de jugadores e ítems, con igual cantidad de ambos. Para cada tamaño $n = m \in \{16, 18, \ldots, 30, 34, \ldots, 50\}$ generamos 5 instancias, y tomamos $b \in \{1, 2, 3, 4, 6, 8, 10, 12, 16\}$ como multiplicadores de presupuestos. Elegimos, tal como antes, una función objetivo constante. El objetivo es analizar cómo impacta el valor de b en el tiempo de resolución, y en particular si las instancias pueden ser resueltas dentro del tiempo límite. Mostramos, para cada combinación de n y b, cuántas de las 5 instancias pudimos resolver, y, por otro lado, el tiempo promedio en segundos hasta encontrar una solución.

La Figura 8 tiene distintos mensajes relevantes. En primer lugar, observando cada tamaño individualmente, vemos que la dificultad cae drásticamente al aumentar el parámetro b. No solamente en cantidad de instancias resueltas, sino también en los tiempos de

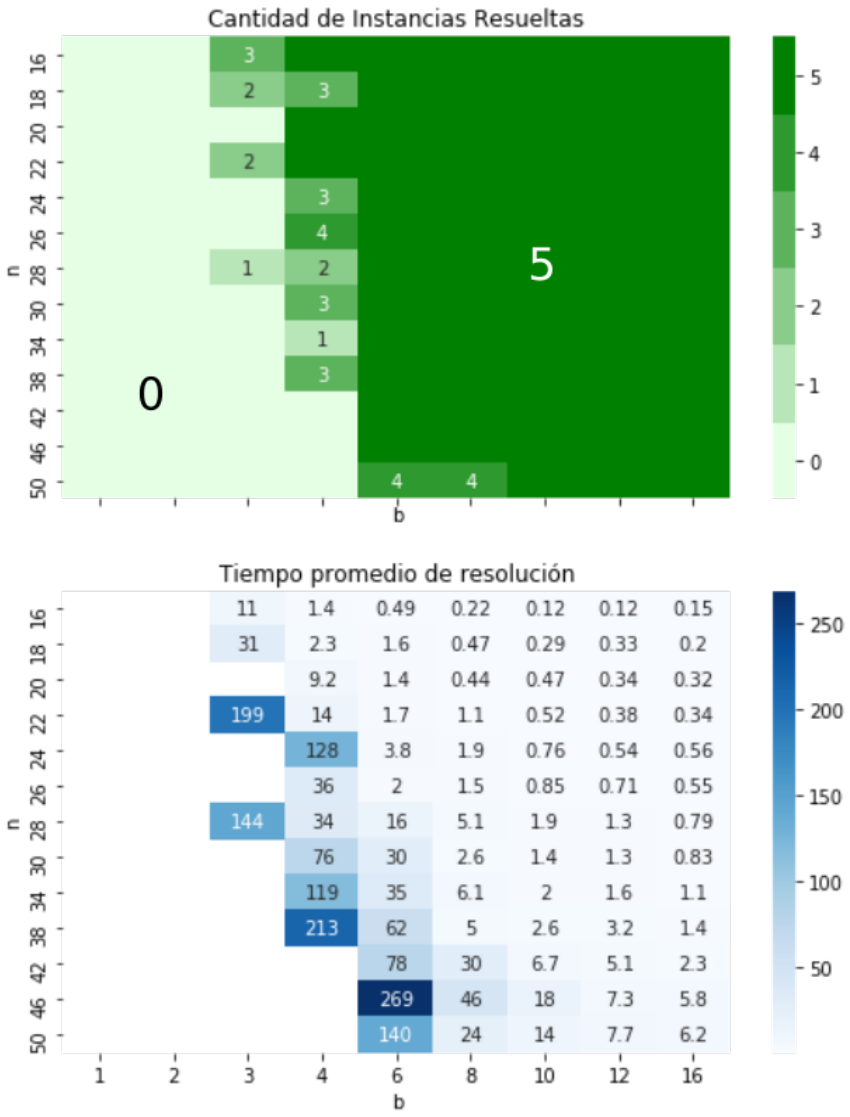


Fig. 8: Cantidad de instancias resueltas, y tiempo promedio de resolución en segundos.

resolución.

Por otro lado, también se evidencia la creciente dificultad al incrementar el n. Incluso para aquellos valores de b para los cuales pudimos resolver todas las instancias, aumentar el tamaño de las mismas perjudicó los tiempos de resolución. Sin embargo, se evidencia que para valores altos de b, la dificultad respecto al n aumenta a un ritmo menor.

Por último, hay que hacer énfasis en los tamaños de las instancias que pudimos resolver. Para poner en perspectiva, el modelo logra resolver las instancias de 5.1 de hasta alrededor de 10 anunciantes e ítems, y solamente un porcentaje de ellas. Al aumentar tan solo un poco los presupuestos, hemos logrado resolver algunas de hasta 50 (y pareciera ser que nos podríamos extender incluso más). Es decir que aumentar el b permite resolver instancias de un tamaño considerablemente más grande.

5.7 Cantidad de jugadores ajustados al reescalar presupuesto (instancias grandes)

El siguiente paso es analizar más en detalle cómo cambia la cantidad de jugadores ajustados al incrementar los presupuestos. Antes lo mostramos para una sola instancia, y de tamaño pequeño, ahora nos gustaría ver si dicha tendencia generaliza. Más aún, para las instancias más grandes que resolvemos, con 46 y 50 anunciantes e ítems, cuántos jugadores necesitamos que hayan sido holgados para lograr ser capaces de resolverlas? Y en esa misma línea, ¿será que la dificultad de una instancia está dada por la cantidad de jugadores ajustados? ¿O por la cantidad de holgados? También podría surgir del porcentaje que resulte de unos y otros.

Para entender esto, mostramos en la Figura 9 el porcentaje promedio de jugadores ajustados en las instancias que pudimos resolver.

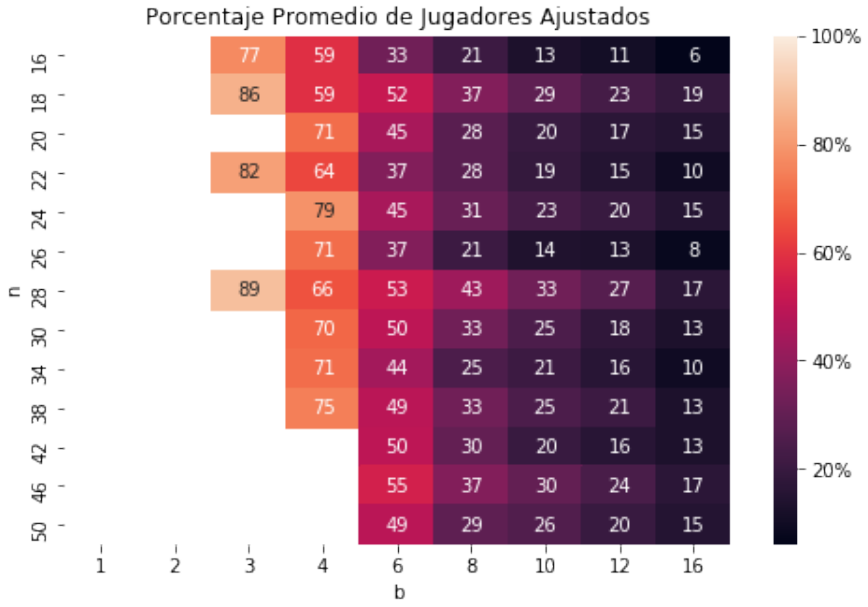


Fig. 9: Porcentaje de jugadores ajustados, promediado entre las instancias resueltas.

Para dejar más claro el gráfico, veamos por ejemplo las instancias de $n = m = 50$, con un factor $b = 6$. Resolvimos 4 de las 5 instancias, quienes resultaron con 30, 20, 22 y 25 jugadores ajustados respectivamente. En promedio fueron 24.25 jugadores, lo que representa un 48.5 % de los 50 que había en esa instancia.

Podemos observar que, a medida que aumenta el tamaño, somos capaces de resolver instancias con menores porcentajes de jugadores ajustados. Sin embargo, algo interesante es que la cantidad no decrece tan sustancialmente. Para los tres tamaños más grandes podemos resolver con $b = 6$ instancias de hasta entre 25 y 30 anunciantes ajustados, mientras que cuando esa es la totalidad de jugadores y no reescalamos los presupuestos, no llegamos a encontrar ningún equilibrio. En otras palabras, pareciera ser que no es la cantidad de ajustados la que determina la dificultad para nuestro modelo, sino que es una

combinación entre el tamaño de la misma y la relación entre ajustados y holgados.

6. ESTUDIO DEL MODELO

El objetivo de nuestro trabajo es mejorar el tiempo de resolución del PLEM propuesto en el Capítulo 4. Sabemos que hay instancias más *difíciles* que otras, que potencialmente no lleguemos a resolver en un tiempo razonable, pero nos gustaría mejorar el porcentaje de instancias para las que obtenemos algún equilibrio. Asimismo, para las instancias que podamos resolver, nos gustaría hacerlo en el menor tiempo posible. En base a lo analizado en el capítulo anterior, vamos a restringirnos en un principio a las instancias ajustadas generadas en 5.1 (que resultaron ser las más difíciles), y luego ver si los resultados generalizan al aumentar los presupuestos.

Para abordar este problema, tomamos un enfoque en dos etapas. La primera parte del trabajo consiste en buscar modificaciones teóricas al modelo original, que refuercen la formulación. Luego, analizar cuáles de esas modificaciones efectivamente mejoran la eficiencia de resolución.

6.1 Desigualdad precio-apuesta más alta

Una primera familia de desigualdades válidas que se puede considerar es la de relación entre la apuesta más alta por un ítem y su precio. Es claro que al corresponderse con la primera y la segunda apuesta más alta, tiene que haber un orden entre ambas en un equilibrio (y dicho sea de paso, en cualquier asignación). El modelo garantiza esto en todas las soluciones factibles enteras, pero no necesariamente en las fraccionarias contenidas en la relajación lineal del problema. Podemos entonces agregar las siguientes desigualdades al modelo.

$$p_j \leq h_j \qquad (\forall j \in M) \tag{22}$$

Sin querer ser redundante, es importante justificar que dichas desigualdades son válidas para este modelo. Si no lo fueran, podrían estar dejando afuera de la región factible a alguna solución, incluso tornando al problema en infactible. En efecto, las variables h_j se corresponden con la apuesta más alta por el ítem j, y las variables p_j con su precio (que debería coincidir con la segunda apuesta más alta en nuestro modelo). Dado que una asignación factible tiene un orden natural entre ambas variables, podemos afirmar que es correcto establecer este orden explícitamente en nuestro modelo.

En términos prácticos, las desigualdades (22) refuerzan el poliedro de la relajación lineal. Al analizar la relajación lineal en el nodo raíz en un conjunto acotado de instancias,

encontramos que la solución fraccionaria no cumple la mayoría de las mismas, por lo que decidimos analizar qué ocurre al agregarlas. Notamos inmediatamente el efecto que producen, reduciendo drásticamente los tiempos de resolución.

6.2 Rompimiento de simetrías en ganadores

Como mencionamos anteriormente, el modelo contiene simetrías. Esto significa que puede haber muchas soluciones distintas pero equivalentes, en el sentido que representan el mismo PE. Las restricciones (4.8) y (4.11) explicitan que hay exactamente un "ganador declarado" (marcado con $w_{ij} = 1$) dentro de los potencialmente varios ganadores (aquellos con $d_{ij} = 1$). La decisión de quién de ellos es no tiene ningún efecto en el equilibrio resultante.

A modo de ejemplo, supongamos que en un equilibrio los jugadores 1 y 2 empatan por el ítem 5, el que se reparte de alguna manera entre ellos. Claramente ambos resultaron ganadores, por lo que $d_{15} = d_{25} = 1$. Una solución podría tener como declarado ganador a cualquiera de los dos. Es decir, cualquiera de las siguientes dos opciones es válida:

(a) $w_{15} = r_{25} = 1$ y $w_{25} = r_{15} = 0$

(b) $w_{15} = r_{25} = 0$ y $w_{25} = r_{15} = 1$

Claramente declarar ganador a uno o a otro no afecta el equilibrio, y nos gustaría poder considerarlos como uno solo. Más aún, la existencia de múltiples soluciones factibles a nuestro PLEM que sean equivalentes puede afectar la performance del *solver*, pues el árbol de posibilidades que se explora podría resultar mucho más grande que lo que debería ser.

Podemos entonces tomar alguna decisión arbitraria sobre quiénes tienen permitido ser el "ganador declarado" y el "declarado segundo" cuando haya un empate por el primer lugar. Si bien esta decisión elimina puntos de la región factible, esto no es un problema siempre y cuando quede, para cada uno de los puntos eliminados, otro equivalente dentro de la región factible. La decisión que tomamos es que, ante un empate en las apuestas más altas por un ítem, sea declarado ganador aquel con el índice más alto, y declarado segundo el que tenga el índice más pequeño. Para probar que eliminar todos aquellos puntos factibles que no respetan esta condición no es un problema, introducimos las siguientes definiciones.

Definición 6. *Dado un punto x^* en la región factible de (4.1)–(4.21), decimos que tiene* ***ganadores ordenados*** *si para todo ítem j se cumple que*

$$w_{ij} = 1 \;\Leftrightarrow\; i = \max\{\, k \in N \mid d_{kj} = 1 \,\}$$

Es decir, si para cada ítem el "ganador declarado" es aquel con el índice más grande dentro de los que realizaron la apuesta más alta por él.

Definición 7. *Dado un punto x^* en la región factible de (4.1)–(4.21), decimos que tiene* ***segundos ordenados*** *si para todo ítem j para el cual hay un empate en las apuestas más altas se cumple que*

$$r_{ij} = 1 \;\Leftrightarrow\; i = \min\{\, k \in N \mid d_{kj} = 1 \,\}$$

Esta última definición es similar a la 6, pero con la salvedad de que solo se refiere a aquellos ítems en que hubo un empate. Es claro que si no hubo empate por el primer lugar, el "declarado segundo" no puede tener su correspondiente variable $d_{ij} = 1$

Con estas definiciones, podemos enunciar el siguiente resultado, que garantiza que podemos romper de esta manera (arbitraria) las simetrías, sin modificar la factibilidad (ni el valor del óptimo cuando haya una función objetivo no constante) del problema.

Lema 5. *Dado un punto x^* en la región factible de (4.1)–(4.21), existe otro punto $\overline{x^*}$ también factible, que representa el mismo PE y posee ganadores ordenados y segundos ordenados.*

El rompimiento de simetrías puede hacerse de múltiples maneras. Exploramos dos alternativas distintas, que se detallan en las siguientes secciones.

6.2.1 Primera opción

La primera propuesta se concentra en los declarados ganadores, forzando a que los puntos factibles tengan ganadores ordenados.

Proposición 6. *La siguiente familia de desigualdades elimina simetrías para nuestro problema*

$$2w_{ij} - 1 \leq d_{ij} - \frac{1}{n}\sum_{i'>i} d_{i'j} \qquad (\forall i \in N, \forall j \in M) \qquad \text{(SB1)}$$

Demostración. Dada una solución factible de (4.1)–(4.21), por el Lema 5 existe otra equivalente con ganadores ordenados y segundos ordenados. Por lo tanto, alcanza con probar que es válida para este último.

En este punto factible, dado un $w_{ij} = 0$, el lado izquierdo de (SB1) vale -1. Para concluir que la restricción vale, alcanza con observar que al lado derecho lo podemos acotar por

$$-\frac{1}{n}\sum_{i'>i} d_{i'j} \geq -\frac{1}{n}\sum_{i'>i} 1 = -\frac{n-i}{n} \geq -1$$

Ahora, dado un $w_{ij} = 1$, la restricción (4.8) fuerza a que $d_{ij} = 1$. Como nuestro punto factible tiene ganadores ordenados, debe suceder que $d_{i'j} = 0\ \forall i' \in \{i+1, \ldots, n\}$. De este modo, la restricción queda $1 \leq 1$, lo cual es válido. □

Esta familia de desigualdades, si bien es válida, presenta dos particularidades negativas. En primer lugar, solo remueve las simetrías en las variables w_{ij}, no quita las potenciales simetrías en las r_{ij} (por ejemplo en el caso de un triple empate por la apuesta más alta). En segundo lugar, y a priori más importante, es que al ser desigualdades de tipo Big-M pueden generar un efecto negativo en el modelo, como no ser capaces de recortar el árbol y solo ralentizar las relajaciones lineales.

6.2.2 Segunda Opción

La segunda alternativa que analizamos utiliza un enfoque parecido. Busca forzar a que la región factible se limite a puntos con ganadores ordenados, pero además pudimos agregar restricciones similares para restringir el poliedro a solo aquellos puntos factibles con segundos ordenados.

Proposición 7. *La siguientes familia de desigualdades son correctas para nuestro problema.*

$$w_{ij} \leq 1 - d_{i'j} \qquad (\forall i \in N, \forall i' \in N_{>i}, \forall j \in M) \qquad \text{(SB2)}$$

$$r_{ij} \leq 2 - d_{ij} - d_{i'j} \qquad (\forall i \in N, \forall i' \in N_{<i}, \forall j \in M) \qquad \text{(SB3)}$$

Demostración. Como antes, basta probar que son válidas para puntos con ganadores y segundos ordenados. Podemos observar que, en ese caso, si un $w_{ij} = 1$, entonces para ningún $i' > i$ vale $d_{i'j} = 1$, por lo que se cumple (SB2). Similarmente, si un $r_{ij} = 1$, entonces o bien no es ganador (en cuyo caso $d_{ij} = 0$), o es aquel de menor índice (con lo cual $d_{i'j} = 0 \ \forall i' < i$), En ambos casos, queda implicado que (SB3) se cumple. □

Un aspecto positivo es que pudimos eliminar parcialmente las simetrías en r_{ij}. Es decir, en caso de un múltiple empate por el primer lugar entre tres o más *bidders*, tanto el ganador como el segundo quedarán unívocamente determinados a partir de sus índices. Vale la pena notar, sin embargo, que no considera el caso en que haya un único ganador y múltiples empatados por el segundo lugar.

Por otro lado, elimina las Big-M que aparecieron en la primera opción, con lo cual computacionalmente uno esperaría que funcionen mejor.

Algo a notar y tener en cuenta respecto a estas restricciones es que su cantidad es del orden de $\mathcal{O}(n^2 m)$, lo cual es más grande que el tamaño del modelo. Por este motivo decidimos agregarlas como planos de corte. La decisión de en qué nodos intentar agregar estos cortes, y de qué manera, será abordada en detalle en el Capítulo 7.

Por último, es lógico que si los empates no se dan naturalmente, estas desigualdades carezcan de sentido, pues no hay ninguna simetría para romper. Esto nos motiva a analizar sus causas y frecuencia en detalle, y tener al menos una intuición de por qué suceden.

6.3 Empates

6.3.1 Intuición sobre los empates

Como mencionamos anteriormente, la intuición respecto a la existencia de empates no es clara a priori. Menos aún, por qué en el modelo original se lo tiene en cuenta explícitamente, por ejemplo tomando recaudos para considerar a los ítems como divisibles. Si observamos los ejemplos 2 y 3, dichos empates suceden en todos sus equilibrios por al menos un ítem.

Pero más interesante es el Ejemplo 1, en el cual no se da un empate por el único ítem. Podemos observar que, cuando modificamos el presupuesto del jugador 1, el equilibrio cambió sustancialmente, y *efectivamente se generó un empate*. ¿Qué sucedió? El jugador 1 se vio forzado a incluir *pacing* en su apuesta. Es claro que en la situación original, al no tener ningún jugador que apostar menos que su valuación, no tendría por qué suceder un empate, pues la única manera de que suceda es que los dos jugadores tengan exactamente la misma valuación por el ítem. Sin embargo, al forzar a un jugador a reescalar sus apuestas, se vio forzado a elegir un multiplicador de $\alpha_1 = 0.25$. No podía ser *mayor*, pues en ese caso ganaría la totalidad del ítem, excediéndose de su presupuesto. Pero tampoco podía ser *menor*: al tener multiplicador α_1 menor estricto que 1, la condición 5 de la que hablamos fuerza a que tenga que gastar todo su presupuesto, por lo que tiene que al menos empatar por ese ítem.

Y esta es la clave: los jugadores que resultan sin *pacing* (es decir, los que tienen $\alpha_i = 1$) tienen sus apuestas fijas, las cuales resultan con empates o no según la distribución de sus valuaciones. Son los que efectivamente escalan sus apuestas (los que tienen $\alpha_i < 1$) los que podrían generar los empates, pues están forzados a gastar todo su presupuesto. Si uno de ellos no empatara, tendría que suceder que los precios de los ítems que ganó sumen *exactamente su presupuesto*, pues los gana íntegros. Si bien uno puede construir un caso patológico donde eso suceda (ver Ejemplo 6 en el Apéndice), es claro que no es la norma; para gastar todo su presupuesto uno debería tener que ganar ítems de manera parcial (lo que únicamente sucede si empata por ellos).

6.3.2 Efectos en el PE

Por un lado, un empate por el ítem j implicará que su precio coincida con la apuesta más alta, pues al ser el precio fijado por la segunda apuesta y coincidir ésta con la primera, ambos tienen que ser iguales. Por lo tanto tendremos $p_j = h_j$.

Por otro lado, sean $i, i' \in N$, $i \neq i'$ dos jugadores que empatan por el ítem j. Como ambos realizaron la apuesta más alta por ese ítem, tenemos que $\alpha_i v_{ij} = \alpha_{i'} v_{i'j} > 0$. Luego, dado que las valuaciones están fijas, si notamos $k = \frac{v_{i'j}}{v_{ij}}$ podemos afirmar que

$$\alpha_i = k\alpha_{i'}$$

6.3.3 Modelado

Con el fin de explotar la información respecto a los empates, agregamos un nuevo conjunto de variables a nuestro modelo. Definimos la variable binaria $e_j \in \{0, 1\}$ que toma valor 1 si se ha producido un empate en el primer puesto por el ítem $j \in M$ y 0 en caso contrario.

Notemos que dado un equilibrio, es muy fácil detectar un empate por el ítem j: sucede si y solo si para ese j hay al menos dos $d_{ij} = 1$. Incorporamos entonces al modelo las restricciones que indican que las variables e_j representan efectivamente esta situación:

$$e_j \leq \sum_{i \in N} d_{ij} - 1 \tag{E1}$$

$$e_j \geq \frac{1}{n-1} \left(\sum_{i \in N} d_{ij} - 1 \right) \tag{E2}$$

La restricción (E1) fuerza a que, si hay un único ganador del ítem j, entonces $e_j = 0$, como corresponde. Notar que, dado que $e_j \geq 0$, también estamos forzando a que $\sum_{i \in N} d_{ij} \geq 1$, pero no es un inconveniente pues esto ya estaba implicado por las restricciones (4.8) y (4.11) del modelo. La (E2), por su parte, fija una cota inferior positiva (y menor o igual a 1) para e_j cuando hay más de un ganador, efectivamente forzando a que valga 1.

Una vez que garantizamos que las variables e_j toman el valor correcto, podemos utilizar la información que nos proporcionan, y las ventajas que mencionamos en la Sección 6.3.2, agregando dos nuevas desigualdades:

Proposición 8. *Dados $i \in I$, $j \in J$, las siguientes desigualdades son válidas para el modelo.*

$$h_j - p_j \leq \overline{v}_j (1 - e_j) \tag{E3}$$

$$h_j - \alpha_i v_{ij} \leq \overline{v}_j (2 - d_{ij} - e_j) \tag{E4}$$

Demostración. Recordamos, en primer lugar, que $\overline{v}_j := \text{máx}_{i \in N} v_{ij}$. La desigualdad (E3) fuerza a que en caso de un empate por un ítem, su precio sea igual a la apuesta más alta, pues ya vimos que en un equilibrio vale que $p_j \leq h_j$ (22). Como también queremos que la desigualdad sea válida cuando *no* hay empate, agregamos un factor lo suficientemente grande como para que, si $e_j = 0$, entonces la restricción no imponga ninguna condición sobre h_j y p_j. En efecto, en ese caso la desigualdad resulta $h_j - p_j \leq \overline{v_j}$. Esta desigualdad vale, pues como $p_j \geq 0$ y para el i ganador $\alpha_i \leq 1$, entonces

$$h_j - p_j \leq h_j = \alpha_i v_{ij} \leq v_{ij} \leq \overline{v_j}$$

La (E4), en tanto, obliga a que en caso de un empate, la apuesta más alta sea igual a la de los jugadores que empatan (y por lo tanto, resulten iguales entre ellas también).

Para ello, primero identificamos aquellos jugadores que empatan por dicho ítem, siendo los i que cumplen $d_{ij} = e_j = 1$, y para ellos esta desigualdad fuerza a que $h_j \leq \alpha_i v_{ij}$, que junto a (4.6) nos dan la igualdad. Igual que antes, agregamos un factor para que si no hay empate, la restricción se cumpla trivialmente, pues $h_j - \alpha_i v_{ij} \leq h_j \leq \overline{v_j}$. □

Naturalmente, uno podría discutir el hecho de que, al agregar variables, estamos convirtiendo a nuestro problema en uno más complejo. Sin embargo, esto no es necesariamente cierto para los PLEMs, pues agregar variables junto a un conjunto de restricciones resulta en un modelo de mayor dimensión pero potencialmente más ajustado.

6.4 Restricciones sobre la cantidad de ganadores

A lo largo de la última sección analizamos la presencia de empates en un equilibrio. Vimos que un empate entre dos jugadores i e i' por el ítem j fija una relación entre dos de las variables α_i y $\alpha_{i'}$, dada por los datos de entrada v_{ij} y $v_{i'j}$. ¿Podría pasar entonces que dos jugadores empaten por dos ítems distintos?

6.4.1 Escenario simple: dos jugadores y dos ítems

En un escenario natural como los que mencionamos, podemos suponer que no hay una dependencia entre las valuaciones de distintos jugadores por un objeto. Entonces, ¿qué implica que dos jugadores i e i' ganen los mismos dos ítems j y j'? Para cada ítem, tienen que haber realizado ambos la misma apuesta:

$$\alpha_i v_{ij} = \alpha_{i'} v_{i'j}$$
$$\alpha_i v_{ij'} = \alpha_{i'} v_{i'j'}$$

Si notamos que como ganaron un ítem tienen que tener valuación positiva por él, y reordenamos, queda:

$$\alpha_i = \frac{v_{i'j}}{v_{ij}} \alpha_{i'}$$
$$\alpha_i = \frac{v_{i'j'}}{v_{ij'}} \alpha_{i'}$$

Luego, se tiene que cumplir que

$$\frac{v_{i'j}}{v_{ij}} = \frac{v_{i'j'}}{v_{ij'}} \tag{6.1}$$

Si esa relación entre las valuaciones (que recordemos, son datos de entrada) no se cumple, entonces no podrá suceder que ambos jugadores ganen los dos ítems.

Proposición 9. *Dados $i, i' \in N$, $j, j' \in M$, si la submatriz de valuaciones $\begin{pmatrix} v_{ij} & v_{ij'} \\ v_{i'j} & v_{i'j'} \end{pmatrix}$ es inversible, entonces en un equilibrio de pacing no puede suceder que los jugadores i e i' ambos resulten ganadores de ambos ítems j y j'.*

Demostración. Vimos que, si ambos jugadores ganan ambos ítems, entonces tiene que cumplirse (6.1). Despejando, esto implica que

$$v_{ij}v_{i'j'} - v_{i'j}v_{ij'} = 0$$

por lo cual la submatriz del enunciado no es inversible. □

Corolario 10. *Dados $i, i' \in N$, $j, j' \in M$, si la submatriz de valuaciones $\begin{pmatrix} v_{ij} & v_{ij'} \\ v_{i'j} & v_{i'j'} \end{pmatrix}$ es inversible, entonces la siguiente desigualdad es válida:*

$$d_{ij} + d_{i'j} + d_{ij'} + d_{i'j'} \leq 3 \tag{6.2}$$

Podemos utilizar este hecho para restringirlo explícitamente en el modelo. Primero definimos

$$D \coloneqq \left\{ (i, i', j, j') \in N^2 \times M^2 \;\middle|\; \det \begin{pmatrix} v_{ij} & v_{ij'} \\ v_{i'j} & v_{i'j'} \end{pmatrix} \neq 0 \right\}$$

Dada una tupla (i, i', j, j') en D, agregamos la desigualdad del Corolario al modelo.

En el caso en que nuestro grafo de valuaciones sea completo (es decir, que todos los jugadores tengan valuaciones positivas por todo ítem), y toda submatriz de 2×2 de valuaciones sea inversible, entonces estaríamos agregando $\binom{n}{2}\binom{m}{2} = \frac{n(n-1)}{2}\frac{m(m-1)}{2} = \mathcal{O}(n^2m^2)$ desigualdades. Experimentos preliminares nos permitieron observar que agregarlas al modelo directamente perjudica los tiempos de ejecución, por lo cual las incorporamos como cortes, tal como hicimos con las desigualdades (SB2)-(SB3).

6.4.2 Generalización a múltiples jugadores e ítems

Naturalmente, nos gustaría poder generalizar esta idea. Vamos a suponer, como antes, que no hay una relación entre las valuaciones, para razonar liberados de ese condicional

constante. Intuitivamente, si i_1 empata con i_2 por un ítem j_1, e i_2 empata con i_3 por otro ítem j_2, entonces hay una relación fijada entre α_{i_1}, α_{i_2} y α_{i_3}; conociendo uno ya conocemos los otros dos. Luego, no podría pasar que i_3 e i_1 empaten por un tercer ítem.

Es decir que (siempre y cuando las valuaciones cumplan alguna precondición) no puede darse una cadena cíclica de empates de un subconjunto de los jugadores por un subconjunto de los ítems. Esta es la situación que queremos modelar.

Para trazar un paralelo con el caso sencillo y ver cómo estamos intentando generalizarlo, decimos que de las 9 posibilidades de ganadores (cada uno de los tres jugadores por cada uno de los tres ítems), a lo sumo pueden cumplirse cinco en simultáneo. Este *cinco* surge de que, como mencionamos, cada empate fija una relación entre dos de los α_i. Y los empates se relacionan con ganadores "de más" por cada ítem. Teniendo tres jugadores, a lo sumo podemos tener dos ganadores extra.

Uno estaría tentado, entonces, a extender esta idea a subconjuntos de jugadores e ítems arbitrarios. Dados un conjunto $I \subseteq N$ de jugadores, y otro $J \subseteq M$ de ítems, queremos poder acotar la cantidad de ganadores que puede haber. Como antes, cada ítem lo puede ganar al menos uno de los jugadores sin problema, pero no puede haber más de $|I| - 1$ ganadores extra.

Propuesta: Dados $I \subseteq N$, $J \subseteq M$, proponemos la siguiente desigualdad:

$$\sum_{i \in I} \sum_{j \in J} d_{ij} \leq |I| + |J| - 1 \tag{P}$$

Ahora bien, ¿será cierto que esta nueva familia extiende al caso simple de dos jugadores y dos ítems? En otras palabras, ¿habrá algún escenario donde estas nuevas restricciones capturen situaciones que en el caso simple habrían sido pasadas por alto?

Consideremos el siguiente ejemplo:

Ejemplo 5. *Se tienen 3 ítems, y 3 jugadores con presupuestos de* 2*, y la siguiente matriz de valuaciones:*

$$V = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Observando que todos los jugadores tienen presupuestos suficientes para ganar todo lo que quieren, todo equilibrio resulta con $\alpha_1 = \alpha_2 = \alpha_3 = 1$ (y con ellos, cualquier asignación de los ítems es válida). Con esos multiplicadores, observamos que hay empate por cada uno de los ítems, entre los dos interesados en él, por lo que en total hay 6 ganadores. Y esto contradice a la desigualdad propuesta, pues

$$\sum_{i \in I} \sum_{j \in J} d_{ij} = 6 > 5 = 3 + 3 - 1$$

Para entender por qué sucede esto, tenemos que observar la matriz de valuaciones. Lo importante no es que hay varias valuaciones iguales, sino que las relaciones entre los α que fija cada empate son redundantes. Lo mismo podría suceder con la siguiente matriz de valuaciones:

$$V = \begin{pmatrix} 0 & 40 & 400 \\ 2 & 0 & 200 \\ 1 & 10 & 0 \end{pmatrix}$$

Para estas valuaciones, si los presupuestos alcanzan, también se pueden dar empates por todos los ítems. Dados los empates por los ítems 1 y 2, si conocemos el multiplicador de pacing del jugador 3, entonces ya fijamos también los de los demás, $\alpha_1 = \frac{\alpha_3}{4}$ y $\alpha_2 = \frac{\alpha_3}{2}$. Si la relación entre las valuaciones de los jugadores por el ítem 3 no contradicen la relación entre sus α, entonces dicho tercer empate puede suceder. Y efectivamente eso es lo que pasa en estos ejemplos.

En ambos ejemplos podemos observar que, tal como en el caso simple, hay relaciones entre las valuaciones que invalidan las desigualdades propuestas. Para ese caso, pudimos exhibir una condición suficiente que nos permitió garantizar que la desigualdad es válida, lo que motiva a pensar que podría existir una condición similar para la generalización. Que la submatriz de valuaciones de los jugadores e ítems involucrados sea inversible (o rango completo cuando $|I| \neq |J|$) ya no nos alcanza. Y tampoco la idea de pedir que toda submatriz de 2×2 de la matriz de valuaciones sea inversible; basta ver cualquiera de los ejemplos presentados.

La condición que encontramos es la siguiente:

Proposición 11 (Condición suficiente)**.** *Sean I y J dos conjuntos de jugadores e ítems respectivamente. Si para todo $r \in \{1, \dots, \text{mín}\{|I|, |J|\}\}$, y todos $I' = (i_1, \dots i_r) \in I^r$ y $J' = (j_1, \dots j_r) \in J^r$ tales que $v_{i_k j_k} \neq 0 \ \forall k \in \{1, \dots, r\}$ se cumple (tomando $i_{r+1} = i_1$) que*

$$\prod_{k=1}^{r} \frac{v_{i_{k+1}, j_k}}{v_{i_k, j_k}} \neq 1$$

Entonces la desigualdad (P) es válida para I y J.

El argumento de por qué la condición es correcta es bastante intuitivo. La desigualdad propuesta limita la cantidad de ganadores, y es cierta salvo que haya, tal como en los ejemplos que mencionamos, una cadena circular de empates. En ese caso, cada empate fija una relación entre los multiplicadores de pacing de los jugadores que intervienen, que depende de sus valuaciones por ese ítem. Como la cadena es circular, el producto de dichas relaciones tiene que ser 1 para no generar una contradicción.

Validar esta condición no es un problema fácil, pues potencialmente hay que chequear todas las permutaciones de todos los posibles subconjuntos de I y de J. Es por esto que decidimos no incluirlas en nuestro modelo.

6.5 Orden entre los α_i.

Ahora bien, no necesitamos dos jugadores que empaten por un ítem para tener una relación entre sus α_i; si uno de ellos lo ganó, entonces tiene que haber realizado la apuesta más alta, lo que impone una cota superior sobre el α correspondiente a los demás.

Esto es, si $d_{ij} = 1$, entonces $\alpha_i v_{ij} \geq \alpha_{i'} v_{i'j}$ para cualquier otro jugador i'. Podemos entonces plantear la siguiente desigualdad, para aquellos jugadores i' con valuación positiva por ese ítem.

Proposición 12. *Sean i, $i' \in N$, $j \in M$ tales que $v_{i'j} > 0$. Entonces, la siguiente desigualdad es válida para (4.1)–(4.21)*

$$\alpha_{i'} - \frac{v_{ij}}{v_{i'j}} \alpha_i \leq 1 - d_{ij} \tag{6.3}$$

Demostración. Veamos que es válida. En primer lugar, si $d_{ij} = 1$, estamos en la situación recién mencionada, el jugador i ganó el ítem j, por lo cual $\alpha_i v_{ij} \geq \alpha_{i'} v_{i'j}$ para todo $i' \neq i$. Luego, como $v_{i'j} > 0$, vale que $\alpha_{i'} \leq \frac{v_{ij}}{v_{i'j}} \alpha_i$ y podemos deducir que $\alpha_{i'} - \frac{v_{ij}}{v_{i'j}} \alpha_i \leq 0 = 1 - d_{ij}$. Por otro lado, si $d_{ij} = 0$, como $\alpha_{i'} \leq 1$ y $\frac{v_{ij}}{v_{i'j}} \alpha_i \geq 0$, la desigualdad vale trivialmente. □

¿Cuántas desigualdades estamos introduciendo al modelo? Suponiendo que la matriz de valuaciones es completa, se tienen en total $\binom{n}{2} m = \frac{n(n-1)}{2} m = \mathcal{O}(n^2 m)$ desigualdades. Esta cantidad puede resultar muy grande para nuestro modelo, por lo que en la Sección 7.3 vamos a analizar si es preferible agregarlas al modelo directamente o como planos de corte.

6.6 Preprocesamiento

Al introducir las instancias a analizar en la Sección 5.1, mencionamos que una posibilidad es que los jugadores solamente estén interesados en un subconjunto de los ítems. Esto se corresponde con lo que sucede en la realidad, donde los anunciantes no están interesados por todas las oportunidades que ofrece la plataforma.

Dado que las valuaciones son, desde nuestra perspectiva, un dato de entrada, podemos utilizarlas para deducir variables y relaciones entre ellas.

En principio, si un ítem no tiene interesados, podemos removerlo de nuestro sistema. Luego, es claro que si un jugador tiene una valuación nula por un ítem, no lo va a ganar pues su apuesta por él va a ser cero. Por lo tanto, para todos aquellos $i \in N, j \in M$ tales que $v_{ij} = 0$, podemos fijar que $d_{ij} = w_{ij} = s_{ij} = 0$.

Más adelante analizaremos si estas modificaciones impactan en los tiempos de resolución del problema.

6.7 Fine-Tuning Cplex

Como mencionamos, utilizamos como base para resolver nuestras instancias un *solver* de propósito general. A lo largo de este capítulo, estuvimos analizando diversas modificaciones al modelo que, esperamos, refuercen la formulación y permitan encontrar soluciones más rápido. Pero algo que también se puede hacer es ajustar los parámetros del *solver*, utilizando nuestro conocimiento específico de este problema en particular. Para nuestros experimentos utilizamos ILOG CPLEX 12.9, por lo que nuestros parámetros son los que este software nos provee.

En primer lugar, existe el parámetro MIP Emphasis switch. Mediante éste, podemos indicarle si queremos que priorice encontrar más rápidamente soluciones factibles (a costa de mayor tiempo probando optimalidad), o si priorizar el tiempo hasta la prueba de optimalidad, sin importar el tiempo que demore hasta empezar a encontrar soluciones factibles. En su valor por defecto, toma una actitud balanceada en este aspecto.

Nuestro modelo, por naturaleza, tiene muy pocos puntos factibles. De hecho, si logramos eliminar todas las simetrías, hay un punto en la región factible por cada equilibrio. En Conitzer et al. [8] se muestra que la multiplicidad de equilibrios no es frecuente, algo que nosotros también pudimos observar en la práctica. Por lo tanto, en la gran mayoría de las instancias que analizamos, vamos a tener un único punto factible, por lo que cualquier solución factible va a ser óptima. Esto motivó a indicar mediante este parámetro que se ponga el énfasis en factibilidad.

En segundo lugar, CPLEX por defecto decide según el caso si aplicar alguna o algunas de sus diversas heurísticas para la generación de soluciones factibles. Dado que, como dijimos, nuestro problema se caracteriza por la dificultad de encontrar soluciones factibles, naturalmente surge el interés de utilizar una heurística de *Feasibility Pump*, tal como es descrita en Fischetti, et al. [11]. Afortunadamente dicho software incluye la posibilidad de ejecutar una variante de esta heurística, por lo que probamos pedirle explícitamente que la introduzca en su búsqueda de soluciones.

Por último, nos gustaría poder influir en el proceso de decisiones del *solver* durante la generación del árbol de enumeración. Una de las decisiones con mayor peso que toma es, en cada nodo del árbol de B&B, qué variable entera va a utilizar para hacer *branching*. En nuestro modelo, tenemos cuatro familias de variables enteras, todas ellas binarias: y_i, d_{ij}, w_{ij} y r_{ij}.

Es importante notar que el algoritmo de B&B es sensible al fijado de prioridad en la selección de la variable de *branching*. Si una de las familias tiene mayor prioridad que todo el resto, entonces lo que sucede es que siempre que haya alguna de ellas con valor fraccionario en el óptimo de la relajación lineal, van a ser las elegidas para hacer

branching (por más que haya otras que parecieran ser mejores opciones). Por lo tanto, elegimos agrupar varias de las familias al asignarles prioridades.

El conjunto de posibles asignaciones de prioridades es demasiado grande como para explorarlo en su totalidad, por lo que vamos a restringirnos a un pequeño grupo. Notamos que, dada la asignación de ganadores (esto es, las variables d_{ij}), los *declarados ganadores* (w_{ij}) quedan determinados, así como los *declarados segundos* para todos aquellos ítems por los que hubo empates. Esto motivó a plantear el siguiente orden de prioridad:

Orden de prioridad 1

- **Prioridad Alta**: y_i, d_{ij}
- **Prioridad Baja**: w_{ij}, r_{ij}

Para fortalecer la idea de que las prioridades pueden tener un impacto en nuestro modelo, vamos a probar también el orden inverso.

Orden de prioridad 2

- **Prioridad Alta**: w_{ij}, r_{ij}
- **Prioridad Baja**: y_i, d_{ij}

Por otro lado, no hay que olvidar que en la Sección 6.3 introdujimos la familia de variables binarias e_j, que podemos agregar o no al modelo. Decidimos, para ambos órdenes, incluirlas en el grupo de las d_{ij}.

7. EXPERIMENTACIÓN Y RESULTADOS

En este capítulo resumiremos los experimentos realizados, con sus respectivos resultados, análisis, y discusión. Tal como en el Capítulo 5, el modelo y sus modificaciones fueron implementados en el lenguaje de programación C++11, utilizando CPLEX 12.9 como paquete de resolución PL y PLEM para resolver la relajación del nodo raíz y desarrollar algoritmos de B&C. Los experimentos se realizaron en una computadora con CPU Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz y 32GB de memoria, con un tiempo límite de 10 minutos para la resolución de una instancia.

En primer lugar, analizamos el efecto de las modificaciones presentadas en el Capítulo 6, sobre instancias generadas como en la Sección 5.1, *Completas* y *Sampled*. Vamos a mirar tanto el porcentaje de instancias resueltas dentro del tiempo límite como el tiempo promedio que requirió llegar a una solución. En todos los casos, miramos tres funciones objetivo distintas:

- Función constante.
- z_{rev}
- z_{pw}

En segundo lugar, vamos a analizar estas modificaciones y su efecto sobre instancias holgadas.

7.1 Desigualdades precio-apuesta más alta

En primera instancia, queremos analizar cuánto afecta a la performance del modelo agregar la familia de desigualdades válidas propuesta en 6.1. Mencionamos que, al mirar los óptimos de las relajaciones lineales, encontramos que muchas de estas desigualdades cortan a los óptimos, por lo que puede tener sentido agregarlas de manera que el algoritmo trabaje sobre un poliedro más ajustado.

Para analizar su efecto, elegimos utilizar instancias de los mismos tamaños que reportan en el trabajo original. Generamos 5 instancias *Completas* y 5 *Sampled* para cada combinación de $n \in \{2, 4, 6, 8, 10\}$ y $m \in \{4, 6, 8, 10, 11, 12, 14\}$. Corrimos todas las instancias con y sin la desigualdad (22), y comparamos para cada m y n la cantidad de instancias resueltas, y el tiempo que fue necesario hasta obtener una solución para cada una. Cuando la función objetivo no es constante, mostramos el tiempo empleado hasta encontrar una solución óptima.

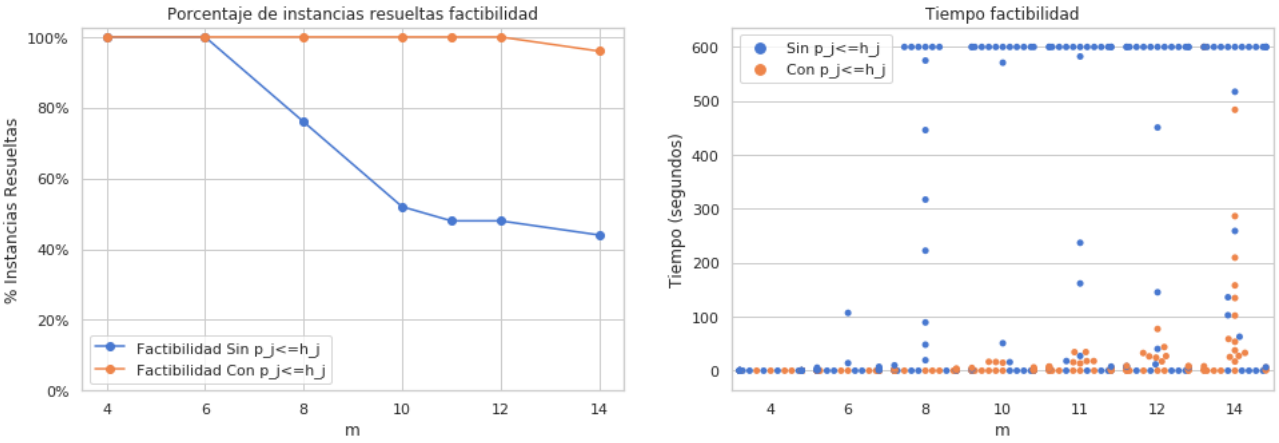


Fig. 10: Instancias *Completas*, con y sin la desigualdad (22). A la izquierda, porcentaje de instancias resueltas dentro del tiempo límite, en función del m. A la derecha, el tiempo de cada instancia hasta encontrar una solución.

Instancias *Completas*

En la Figura 10 comparamos los resultados sobre las instancias *Completas*, con función objetivo constante. En primer lugar, podemos observar que para el modelo original pudimos replicar la tendencia decreciente al aumentar el m reportada en Conitzer et al. [8], obteniendo porcentajes similares. Pero al agregar la desigualdad (22), la performance mejora notoriamente, y podemos resolver casi la totalidad de las instancias. Para el tamaño más grande analizado, $m = 14$, el modelo original solo resuelve el 44 % de las instancias, mientras que con la nueva restricción no solo pudimos resolver el 96 % (todas salvo una), sino que además la mayoría pudo ser resuelta dentro del primer minuto.

Si analizamos ahora las mismas instancias, pero maximizando z_{rev} y z_{pw}, vemos que sucede algo parecido (ver figuras 11 y 12). La diferencia entre incorporar o no la desigualdad al modelo es realmente notable, tanto en porcentaje de instancias resueltas como en los tiempos de resolución. Algo que podemos mencionar es la variabilidad de performance en las corridas sin la desigualdad: cuando maximizamos z_{rev} se resuelve la misma cantidad de instancias con $m = 10$ que con $m = 8$, y en tiempos similares. Maximizando z_{pw} sucede algo más extremo, que es que resolvemos más instancias con $m = 12$ que con $m = 11$.

Por otro lado, agregar una función objetivo tiene una consecuencia, que es que la ejecución no necesariamente finaliza al obtener una solución factible, pues puede haber una solución con un mejor funcional. Y aunque no la haya, hay que *probar* que el equilibrio encontrado es óptimo. En este sentido, algo que observamos al maximizar z_{rev} es que todas las instancias resultaron ajustadas. Y por la Proposición 4, en dichas instancias el óptimo de la relajación lineal coincide con el valor de la función objetivo en todo equilibrio ajustado. Luego, encontrar un equilibrio ajustado viene con prueba de optimalidad, pues su funcional coincide con la cota dual; no hubo instancias con solución factible y no óptima para esta métrica.

Algo distinto sucedió al maximizar z_{pw}: si bien ninguna instancia tuvo múltiples equi-

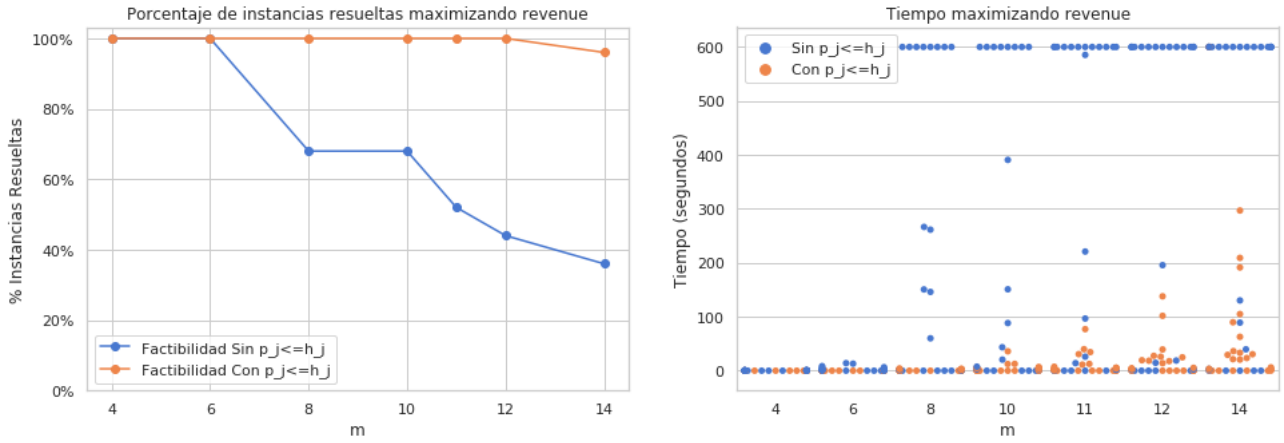


Fig. 11: Ídem Figura 10 maximizando *revenue*

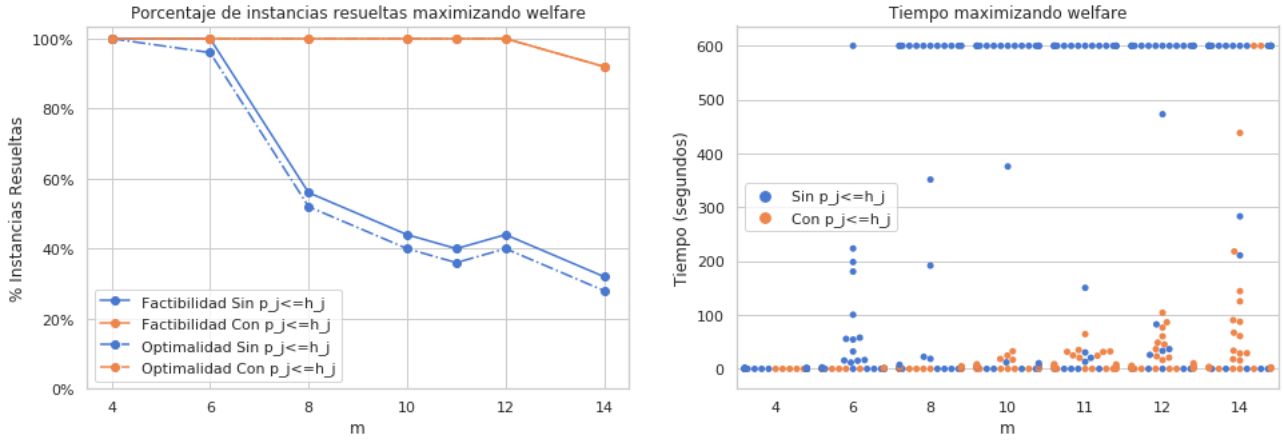


Fig. 12: Ídem Figura 10 maximizando *welfare*. A la derecha, tiempos hasta optimalidad.

librios con funcionales distintos, en algunas no se logró cerrar el árbol de exploración antes de agotar el tiempo disponible, por lo que hay un pequeño *gap* entre la cantidad de instancias resueltas a factibilidad y a optimalidad.

Por último, es interesante observar que, incluso para los tamaños más grandes que consideramos, la mayor parte de las instancias para las que se encontró un equilibrio fueron resueltas en tan solo unos pocos segundos. Esto da cuenta de la variabilidad de la dificultad para encontrar un equilibrio dentro de instancias del mismo tamaño.

Instancias *Sampled*

Podemos observar en las figuras 13, 14 y 15 que la diferencia sigue siendo sustancial para este tipo de instancias. Para las tres funciones objetivo, logramos resolver a optimalidad la totalidad de las instancias cuando agregamos las desigualdades (22), y en tiempos sorprendentemente pequeños. Sin esa desigualdad, los porcentajes resultaron parecidos a los reportados en Conitzer et al. [8], aunque se evidencia mayor variabilidad en ellos.

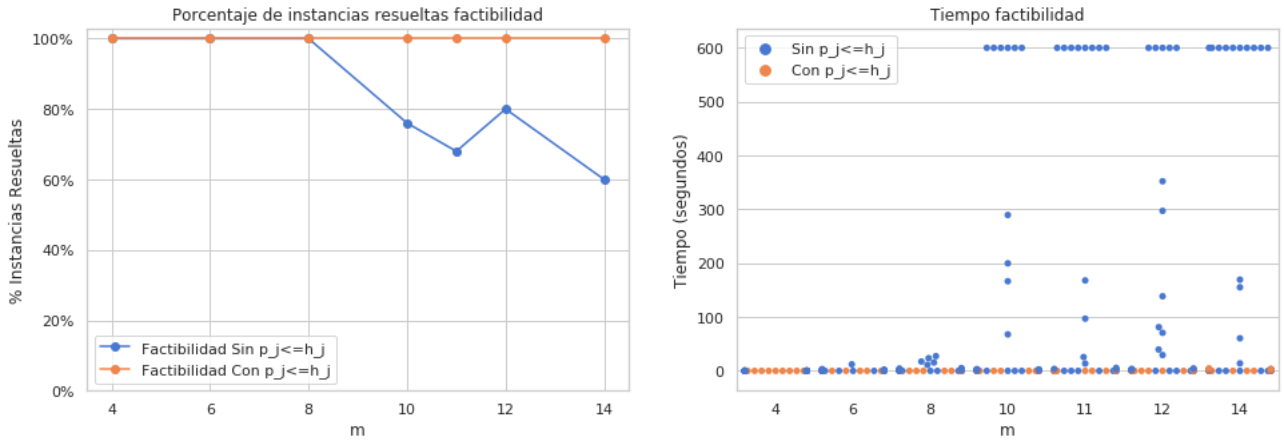


Fig. 13: Instancias *Sampled*, función objetivo constante.

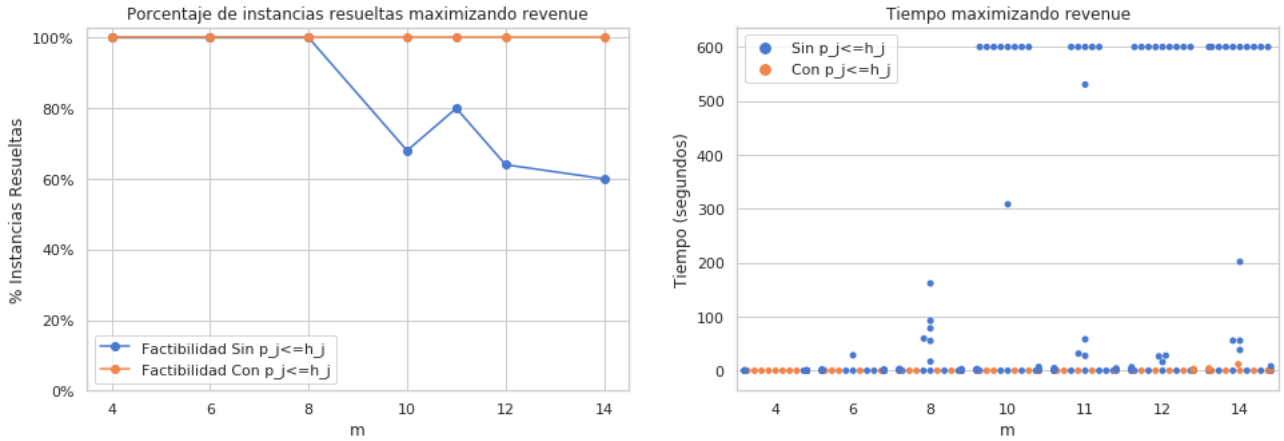


Fig. 14: Instancias *Sampled*, maximizando *revenue*.

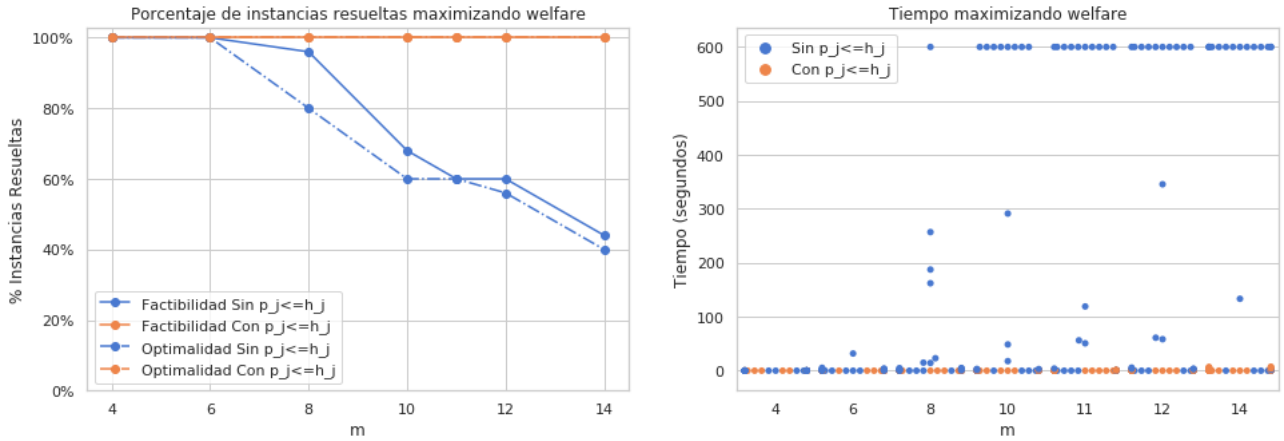


Fig. 15: Instancias *Sampled*, maximizando *welfare*. A la derecha, tiempos hasta optimalidad.

Podemos destacar situaciones particulares, como que las instancias de tamaño $m = 11$ resultaron más difíciles que las de $m = 10$ y $m = 12$ al utilizar una función objetivo constante, mientras que ocurre lo opuesto cuando maximizamos z_{rev}.

En la Figura 15 podemos observar con más detalle algo que mencionamos que sucedió en las instancias *Completas*. Si miramos $m = 14$, se puede ver que todas las instancias o bien fueron resueltas en unos segundos, o bien no se les encontró solución óptima.

Para ver los resultados en función del n, referimos al lector a la Sección B.1 del Apéndice. En general las tendencias resultaron parecidas al mirarlas respecto a n o a m, por lo que de acá en adelante mostraremos únicamente los resultados en función del m.

Tanto para las instancias *Completas* como para las *Sampled*, el resultado es claro: agregar la desigualdad (22) genera un impacto muy positivo en la capacidad del algoritmo para encontrar soluciones. Por ese motivo, de aquí en adelante vamos a trabajar con instancias de tamaños más grandes, para poder entender el impacto del resto de las modificaciones, y si permiten resolver este nuevo grupo de instancias.

Generamos 5 instancias *Completas* para cada combinación de $n, m \in \{8, 10, 12, 14, 16\}$, y 5 *Sampled* para cada $n \in \{10, 12, 14, 16, 18\}$ y $m \in \{12, 14, 16, 18, 20\}$, y las utilizamos para analizar el resto de las contribuciones del Capítulo 6.

7.2 Preprocesamiento para instancias *Sampled*

En primer lugar, analizamos el impacto de los preprocesamientos introducidos en la Sección 6.6. Para ello, vamos a restringirnos a las instancias *Sampled*, pues únicamente en ellas se modifica el modelo, fijando el valor de algunas de las variables. Vamos a realizar un experimento similar al de la sección anterior, comparando la resolución con y sin el preprocesamiento. Para comparar los tiempos, tomamos el tiempo promedio de resolución de las instancias de cada m, considerando a los de las instancias que no pudimos resolver como 600 segundos (es decir, el límite de tiempo).

En la Figura 16 podemos observar la comparación entre agregar o no el preprocesamiento, cuando la función objetivo es constante. Podemos ver que cuando $m = 18$ resultó de utilidad, pero con $m = 20$ pudimos resolver una instancia menos. Algo interesante es que, a pesar de eso, los tiempos promedios de resolución fueron consistentemente mejores para todos los m.

Algo similar sucede al considerar z_{rev}, en la Figura 17 podemos ver un patrón similar al caso anterior. En el caso de z_{pw} sucede algo distinto; el preprocesamiento permite resolver más instancias, y el tiempo promedio se mantiene similar.

En definitiva, si bien la diferencia no es sustancial, el balance es positivo, por lo que decidimos incorporar el preprocesamiento al modelo de aquí en adelante.

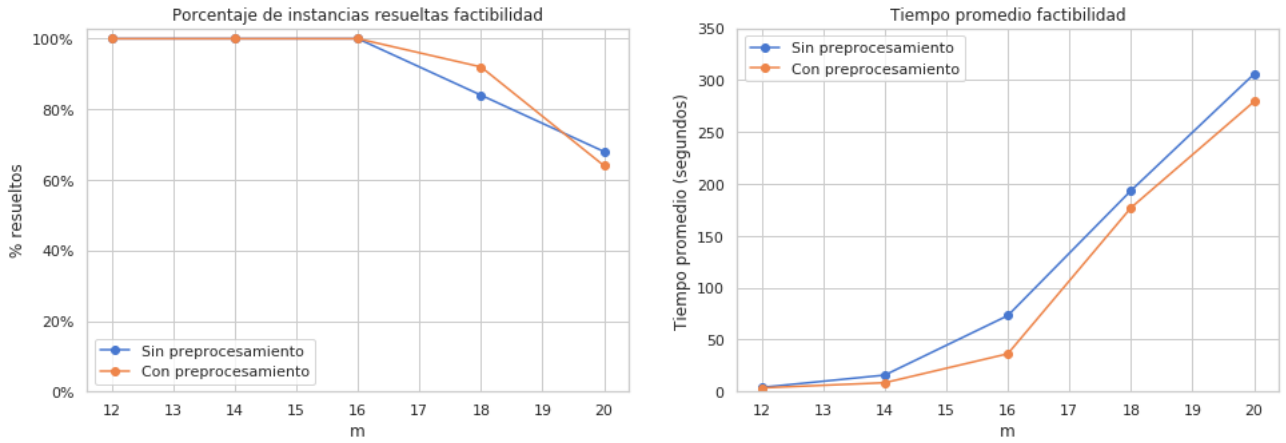


Fig. 16: Función objetivo constante. A la izquierda, porcentaje de instancias resueltas, en función del m. A la derecha, tiempo promedio de resolución, considerando 600 segundos para aquellas que no se pudo resolver.

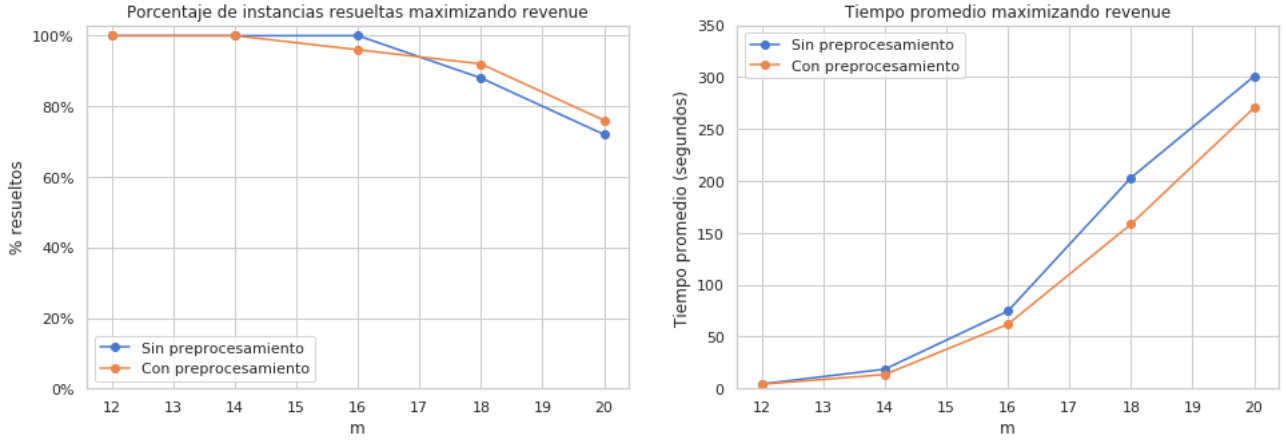


Fig. 17: Maximizando *revenue*

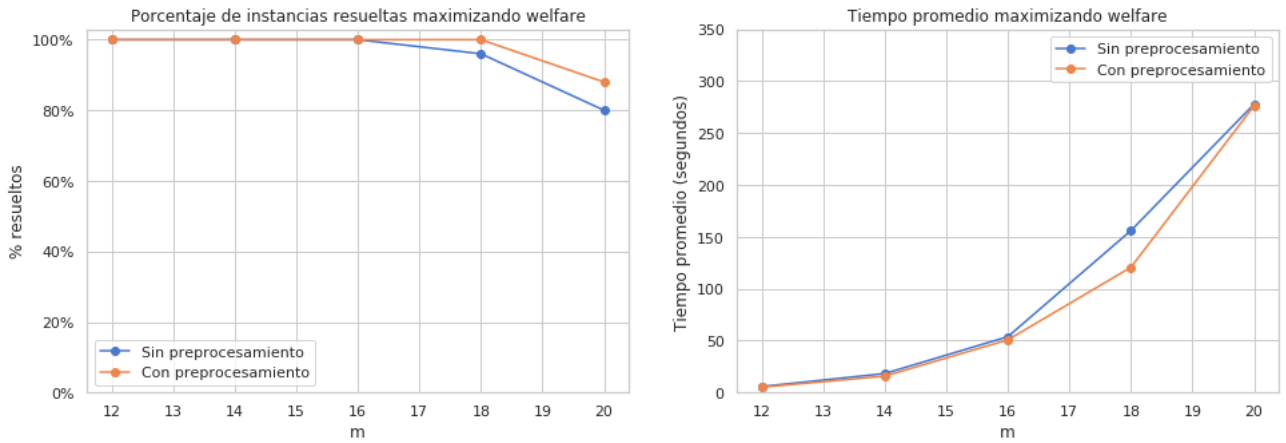


Fig. 18: Maximizando *welfare*

7.3 Motivación para usar cortes

Al introducir varias de las familias de desigualdades, mencionamos que, de ser una cantidad muy grande, podrían ser perjudiciales para el modelo. En cada nodo del árbol de enumeración tenemos que resolver un PL, por lo cual si el modelo es muy grande, esta tarea se dificulta. Vamos a probar, entonces, agregar o no las desigualdades (6.3) al modelo directamente, y evaluar cuántas instancias podemos resolver. Recordamos que nuestro modelo incluye las desigualdades (22) y el preprocesamiento explicado en la Sección 6.6.

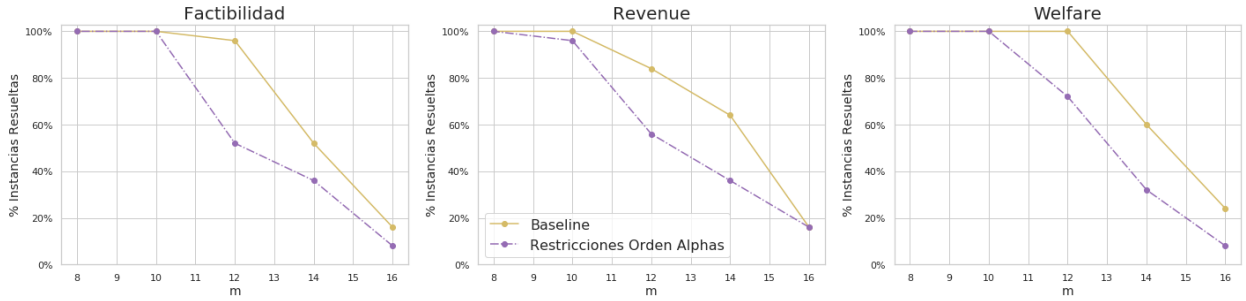


Fig. 19: Instancias *Completas*, con y sin las desigualdades (6.3) de Orden entre los α agregadas al modelo.

Como se puede observar claramente en la Figura 19, agregar las restricciones perjudicó al modelo para las tres funciones objetivo. La diferencia es sustancial, pudiendo resolver en algunos casos menos de la mitad de las instancias que utilizando el modelo base. Podemos explicar esto señalando que, al agregar esta cantidad de restricciones al modelo, cada una de las relajaciones lineales requiere más tiempo para ser resuelta. Esto se ve reforzado cuando miramos la cantidad de nodos explorados en los árboles de exploración; salvo en aquellas instancias que pudimos resolver en unos pocos segundos, agregar estas desigualdades redujo significativamente la cantidad de nodos que se pudo explorar. Lo mismo se observó en las instancias *Sampled*.

Esto no nos dice que agregar dichas desigualdades como planos de corte resulte en una mejora, pero evidencia el efecto negativo en la performance al incorporarlas directamente al modelo. Algo similar sucede con la segunda variante de Rompimiento de Simetrías y con las cotas para la cantidad de ganadores, donde al agregar (SB2)-(SB3) y (6.2) respectivamente se observa un efecto semejante.

7.4 Prioridades de *branching*

Evaluamos a continuación los dos órdenes de prioridad de *branching* desarrollados en la Sección 6.7.

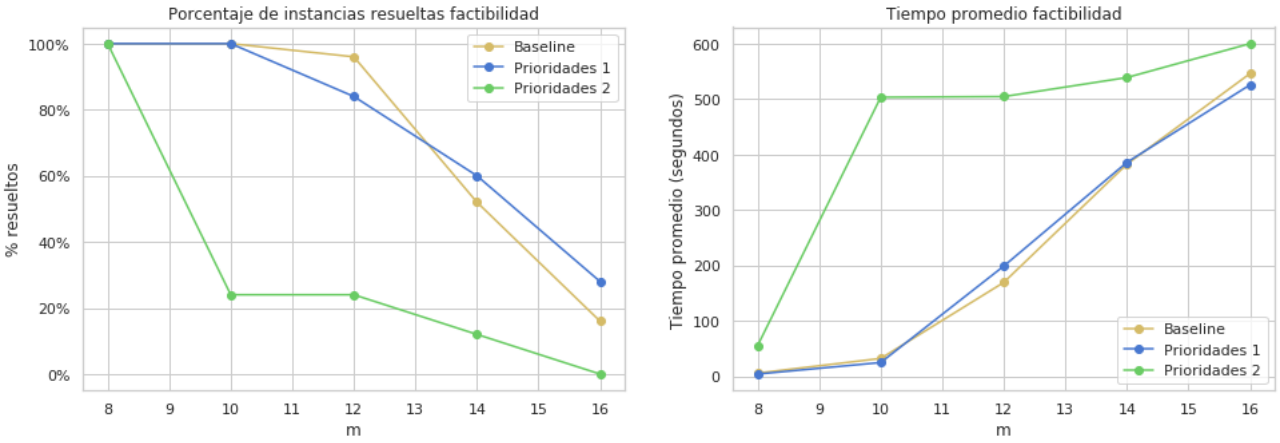


Fig. 20: Instancias *Completas*, función objetivo constante. A la izquierda, porcentaje de instancias resueltas. A la derecha, tiempo promedio de resolución, considerando 600 segundos para aquellas que no se pudo resolver.

Hay dos cosas a comentar respecto a la Figura 20. En primer lugar, se evidencia que el orden de prioridades 2 es realmente perjudicial para la habilidad del *solver* para encontrar puntos factibles. La cantidad de instancias que se pudieron resolver en ese caso es sustancialmente menor que si no se hubieran fijado prioridades de *branching*, llegando a no poder resolverse ninguna con $m = 16$. En segundo lugar, podemos afirmar que el otro orden de prioridades resultó *razonable*. Si bien no hay evidencia clara de que agregar el orden represente una mejora en la resolución, es posible que en combinación con otras de las desigualdades sí logre incrementar la cantidad de instancias resueltas.

Para las instancias *Sampled*, así como con las otras dos funciones objetivo, los resultados fueron similares, y pueden observarse en la sección B.2 del Apéndice.

7.5 Empates y Rompimiento de Simetrías

El siguiente experimento evalúa agregar el resto de las desigualdades al modelo. Probamos la primera variante de Rompimiento de Simetrías, y las variables de Empates con sus restricciones.

En la Figura 21 vemos el porcentaje de instancias resueltas según m, para las instancias *Completas* y *Sampled*, y las tres funciones objetivo. Podemos ver que modelar los empates resultó una mejora con una función objetivo constante, y, para las instancias *Completas*, también cuando maximizamos z_{pw}. Las desigualdades de Rompimiento de Simetrías no mostraron mejoras en los tiempos, ni marcaron una diferencia positiva respecto al modelo base.

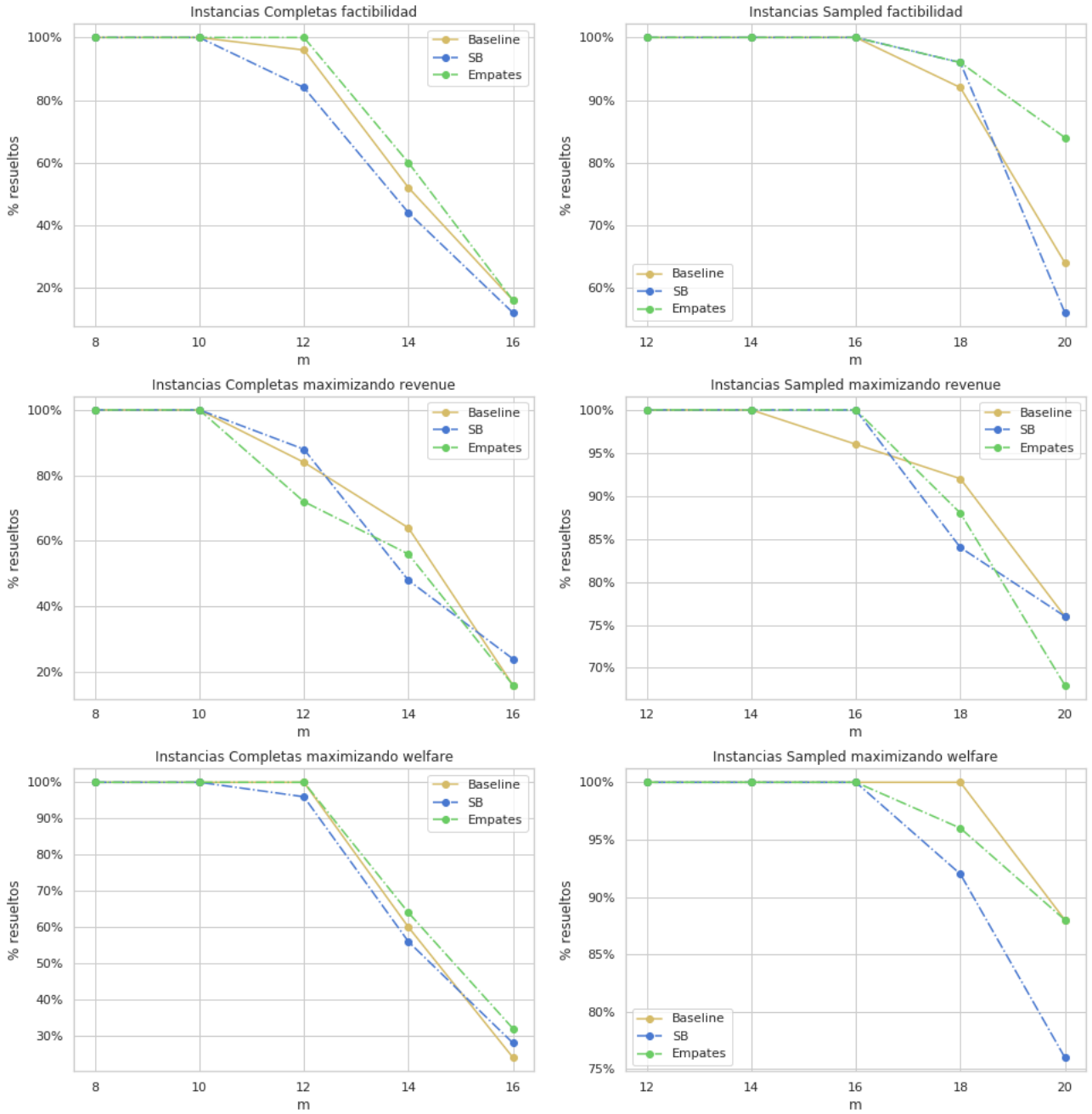


Fig. 21: Instancias *Completas* y *Sampled*, porcentaje de instancias resueltas.

7.6 Primeras combinaciones - sin cortes

Si bien pudimos observar el impacto de varias de las contribuciones por separado, sería interesante poder evaluar cómo funcionan en conjunto. Podría pasar que dos o más de ellas no logren impactos significativos por separado, pero sí lo hagan al combinarlas. Vamos a probar entonces combinar la primera familia de desigualdades de Rompimiento de Simetrías con el orden de prioridades de *branching* 1 y el agregar las variables de empates con sus respectivas restricciones.

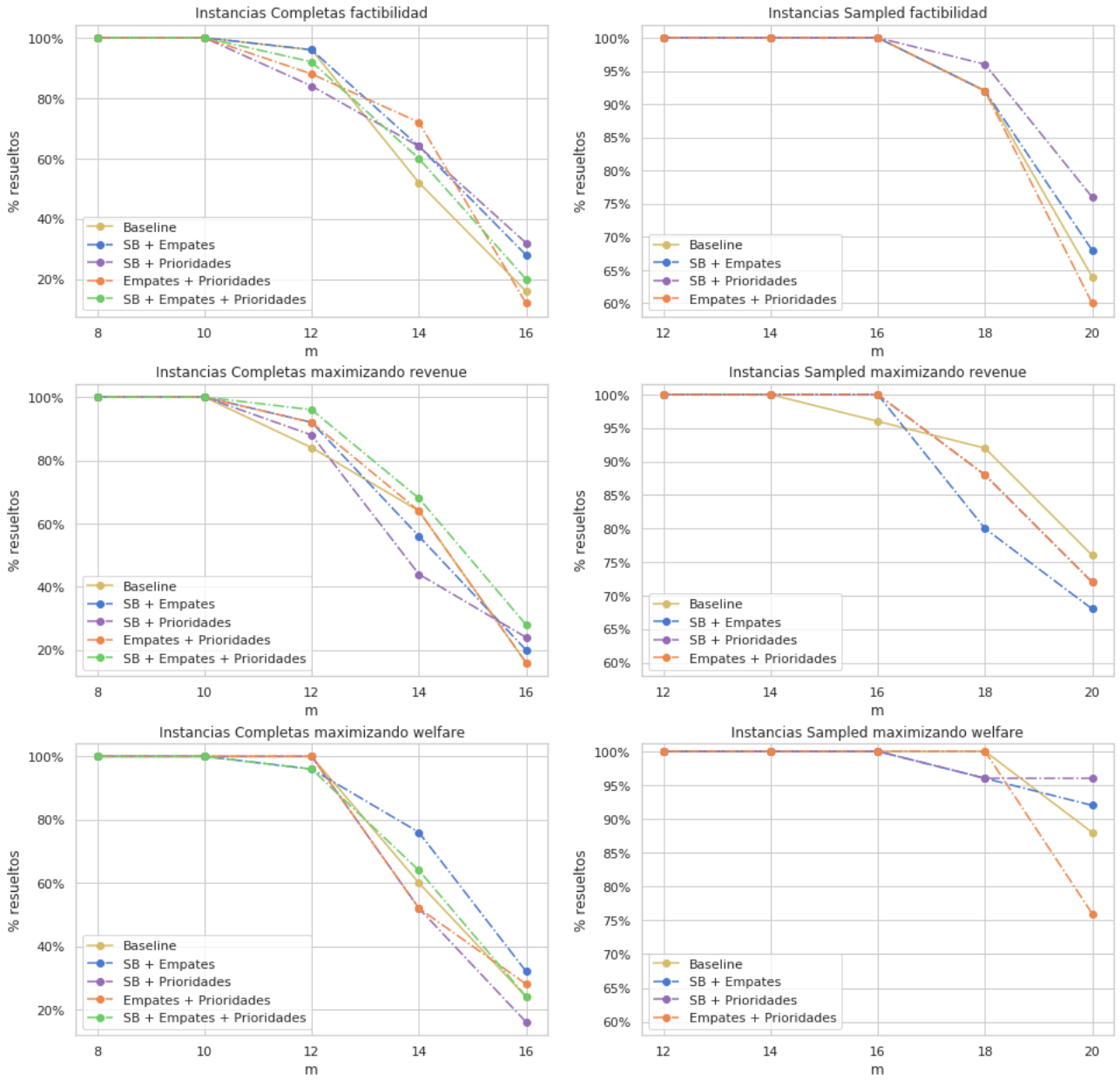


Fig. 22: Instancias *Completas* y *Sampled*, porcentaje de instancias resueltas. Para las instancias *Sampled* maximizando *revenue*, SB+Prioridades coincidió con Empates+Prioridades.

En la Figura 22 podemos ver que efectivamente, combinar las desigualdades válidas propuestas, el agregado de variables, y el fijado del orden de prioridad de *branching* tuvo (en algunos casos) impacto positivo en la cantidad de instancias resueltas.

En el caso de las instancias *Completas* con una función objetivo constante, combinar Rompimiento de Simetrías con agregar variables que representen los empates parece dar mejores resultados. Si bien es verdad que para algunos tamaños hubo otras combinaciones con mejor performance, esta combinación tiene la ventaja de que se mantuvo consistentemente por encima del *baseline*. Con z_{pw} también aparece esta combinación como ganadora, mientras que al maximizar z_{rev} tuvo mejores resultados utilizar las tres.

Para las instancias *Sampled*, por su parte, cuando elegimos la función objetivo constante resultó claro ganador tomar las desigualdades de Rompimiento de Simetrías, junto a las prioridades de *branching*. Al maximizar z_{pw} es más difícil la comparación, pero podemos declarar vencedor a la misma combinación, con la que queda sin resolución una única instancia para cada uno de los dos tamaños más grandes de m. Con función objetivo z_{rev} sucede que ninguna de las combinaciones puede superar al modelo base.

7.7 Familias de cortes

Mencionamos tres familias de desigualdades que decidimos incorporar al algoritmo de resolución como familias de cortes, motivados por lo observado en la Sección 7.3. Ellas son la segunda variante de Rompimiento de Simetrías (SB2)-(SB3), las cotas para cantidad de Ganadores (6.2) y las desigualdades (6.3) de Orden entre los α_i.

Los planos de corte se pueden incorporar de diversas maneras. Basados en resultados de experimentos preliminares, decidimos considerar aquí dos variantes:

- Buscar únicamente en el nodo raíz del árbol de exploración (C&B)
- Buscar en los primeros 1000 nodos del mismo (B&C)

Como algoritmo de separación, en cada nodo en cuestión recorremos todas las posibles desigualdades, agregando todas aquellas que se invalidan en el óptimo de la relajación lineal del mismo. En el nodo raíz realizamos tantas iteraciones de planos de corte como nos permita el *solver*, mientras que en los restantes nodos realizamos una única iteración. El algoritmo de B&C tiene más chances de encontrar más cortes para agregar durante la ejecución, pero a costa de un mayor tiempo empleado buscándolos.

Probamos, en primer lugar, las tres familias por separado. En la Figura 23 mostramos la cantidad de instancias *Completas* y *Sampled* resueltas, para cada una de las tres funciones objetivo. Para las instancias *Completas*, podemos observar que agregar los cortes de Rompimiento de Simetrías en el nodo raíz estableció una pequeña mejora en el caso de z_{pw}. Para el problema de factibilidad no hay un claro ganador, pero podemos mencionar

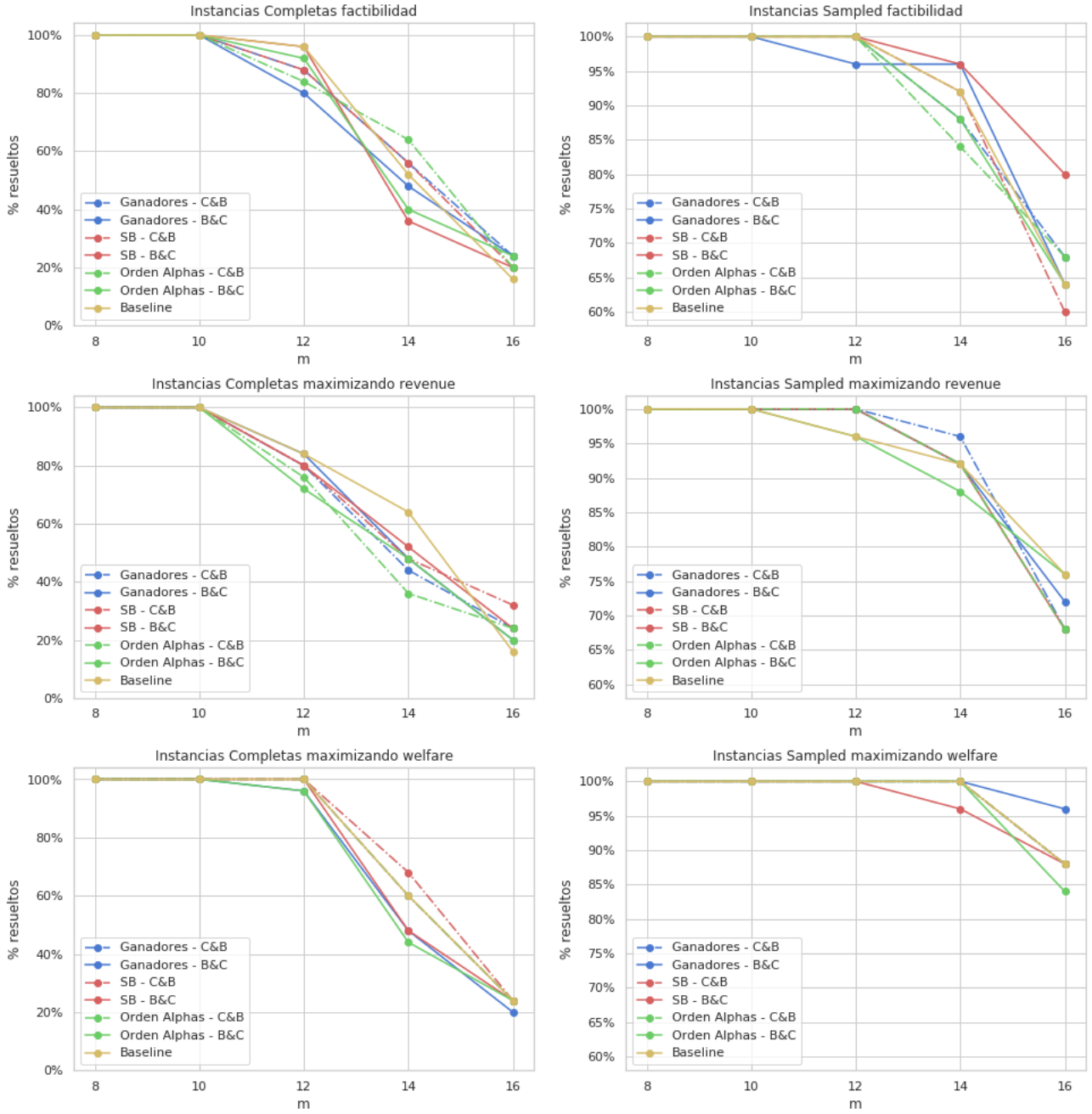


Fig. 23: Instancias *Completas* y *Sampled*, porcentaje de instancias resueltas. En las instancias *Sampled*, para z_{rev} SB coincidió con la versión C&B de cortes SB, y para z_{pw} todas las versiones de C&B coincidieron con Baseline

a los cortes de Ganadores y de Orden de los α_i como candidatos a combinar con otras desigualdades, mientras que para z_{rev} no podemos establecer ninguno que represente una mejora respecto al baseline.

En las *Sampled*, por su parte, tuvo mejores resultados agregar los cortes de Rompimiento de Simetrías para el problema de factibilidad y los de Ganadores para z_{pw}, en ambos casos utilizando B&C. Con z_{rev} como función objetivo, ninguna familia proporcionó una ventaja significativa.

7.8 Más combinaciones - incluyendo cortes

A esta altura, separamos nuestro análisis según tipo de instancia y función objetivo. De esta manera, dada un problema podemos elegir la combinación de parámetros que esperamos funcione mejor para él.

Podemos resumir lo analizado a lo largo de este capítulo para mostrar las mejores combinaciones encontradas hasta ahora.

	Completos		Sampled	
	Combinaciones	Cortes	Combinaciones	Cortes
Factibilidad	(SB1) + Empates	Ganadores + Orden α_i C&B	(SB1) + Priorities o Empates	(SB2)-(SB3) B&C
Revenue	(SB1) + Empates + Priorities	-	-	-
Welfare	(SB1) + Empates	(SB2)-(SB3) C&B	(SB1) + Priorities	Ganadores B&C

Tab. 7.1: Mejores combinaciones de parámetros según tipo de instancia y función objetivo, sin cortes y usando exclusivamente cortes.

Vamos a considerar, para cada tipo de instancia y función objetivo, una combinación entre ambas combinaciones ganadoras, según lo mostrado en la Tabla 7.1 y otros experimentos no reportados. En el caso de las *Completas*, para la función objetivo constante, incluimos ambas combinaciones a la vez, en z_{pw} consideramos (SB1) + Empates + (SB3) C&B, y para z_{rev} no incorporamos familias de cortes.

En cuanto a las *Sampled*, probamos, para la función objetivo constante, Empates + cortes de Rompimiento de Simetrías en un algoritmo de B&C, para z_{pw} la combinación entre ambas combinaciones ganadoras, y para z_{rev} los tres tipos de cortes, también usando B&C.

Por otro lado, consideramos una variante más, común a todos. Ésta es incluir **todas las contribuciones**, es decir:

- Preprocesamiento
- Desigualdad precio-apuesta más alta (22)

- Desigualdades de Rompimiento de Simetrías (SB1)
- Variables de Empates, con sus respectivas desigualdades (E1)-(E4)
- Orden de Prioridades 1 de *branching*
- Cortes de Orden entre los α_i (6.3), en los primeros 1000 nodos
- Cortes de cotas para la cantidad de Ganadores, en los primeros 1000 nodos

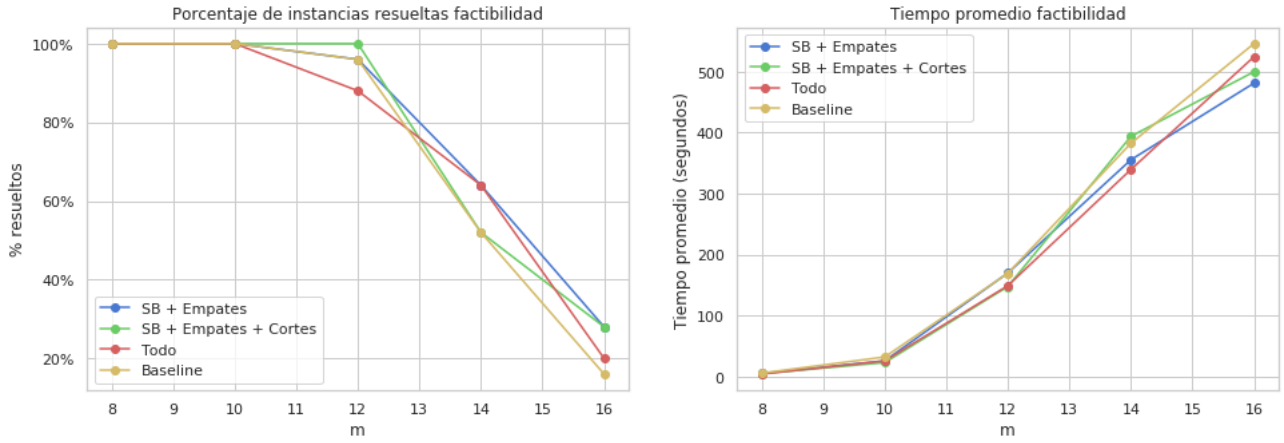


Fig. 24: Instancias *Completas*, función objetivo constante. A la izquierda, porcentaje de instancias resueltas según m, a la derecha tiempo promedio de resolución.

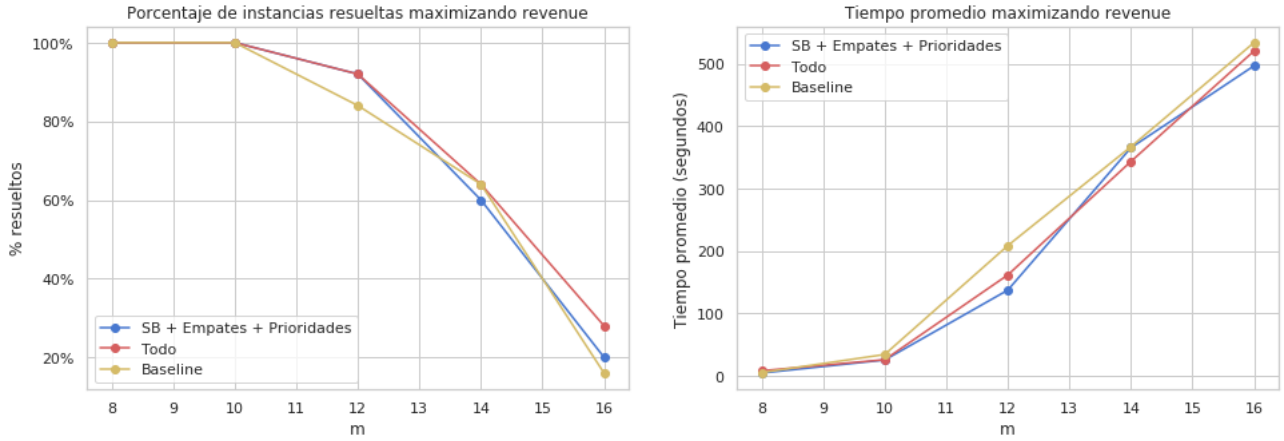


Fig. 25: Instancias *Completas*, maximizando *revenue*.

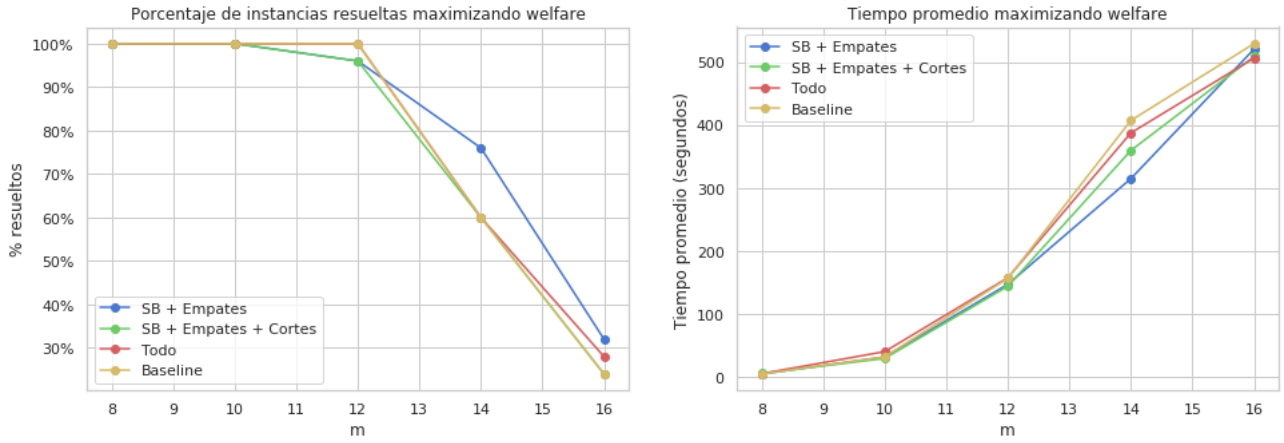


Fig. 26: Instancias *Completas*, maximizando *welfare*.

En la Figura 24 vemos los resultados para instancias *Completas*, con función objetivo constante. Podemos ver que la combinación de desigualdades de Rompimiento de Simetrías junto a Empates fue la que resolvió la mayor cantidad de instancias, y en menos tiempo que al agregarle cortes. En la Figura 26 podemos ver que lo mismo sucedió al maximizar z_{pw}. Para la función objetivo z_{rev}, por su parte, observamos en la Figura 25 que agregar todas las contribuciones fue la mejor combinación.

Veamos ahora las *Sampled*.

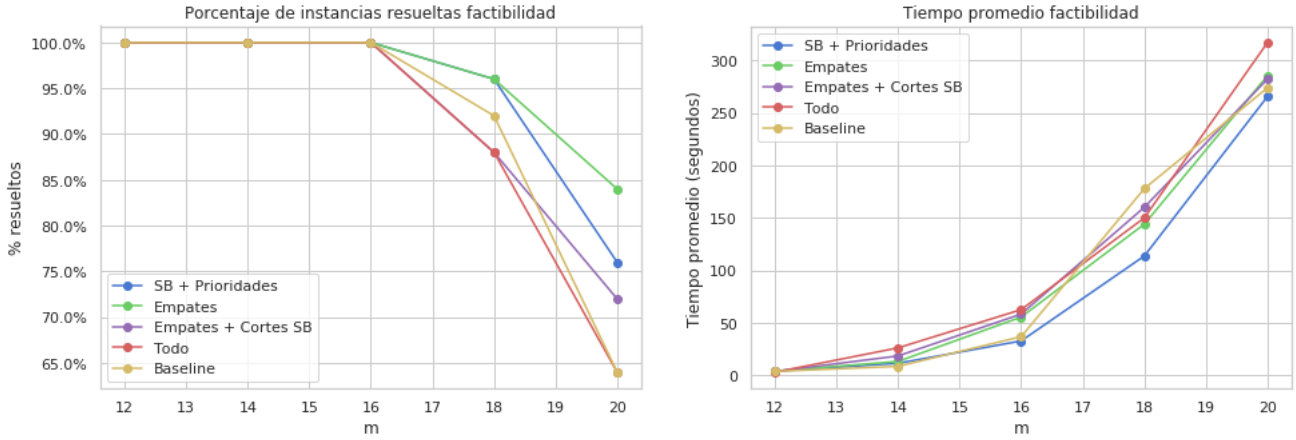


Fig. 27: Instancias *Sampled*, función objetivo constante. A la izquierda, porcentaje de instancias resueltas según m, a la derecha tiempo promedio de resolución.

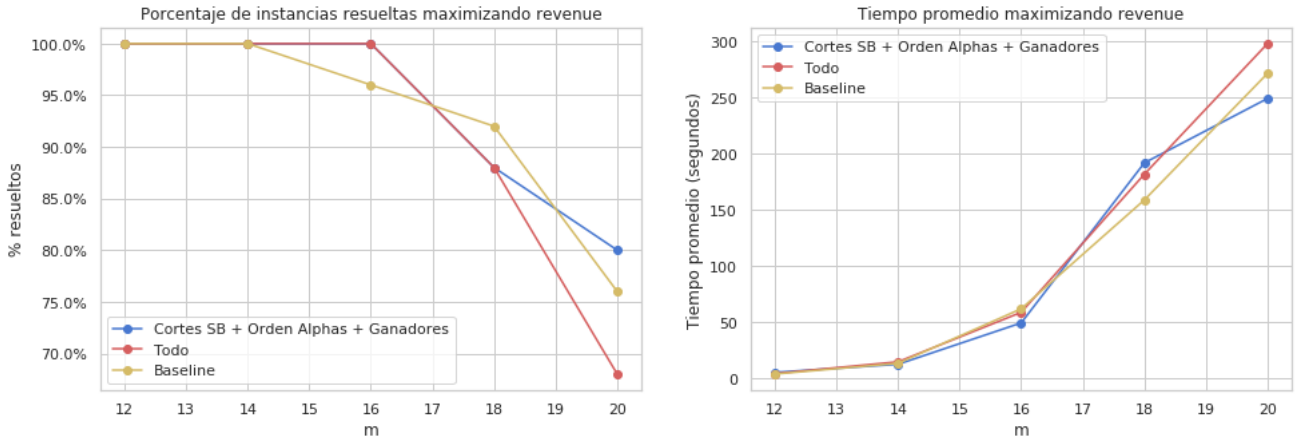


Fig. 28: Instancias *Sampled*, maximizando *revenue*.

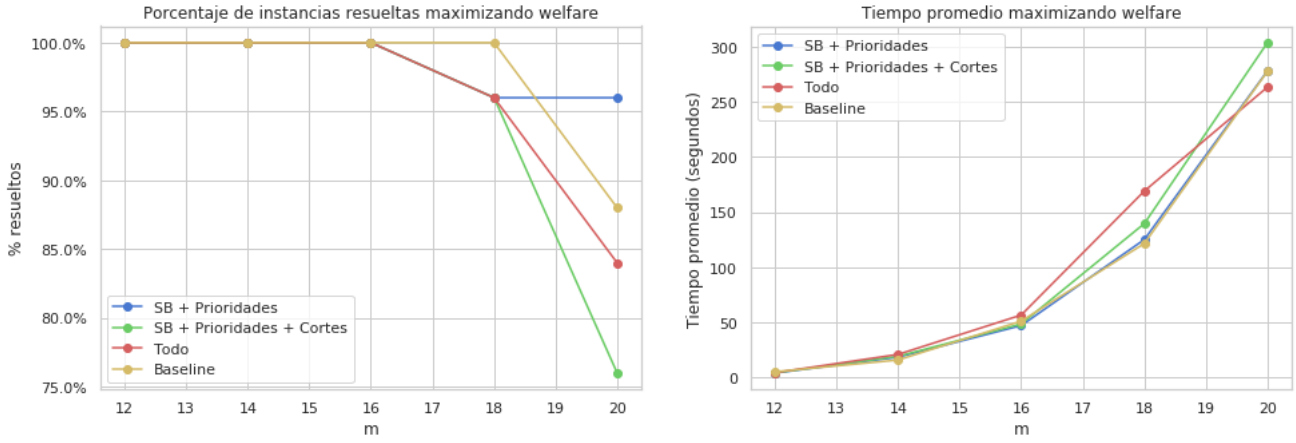


Fig. 29: Instancias *Sampled*, maximizando *welfare*.

En la Figura 27 vemos los resultados para instancias *Sampled*, con función objetivo constante. Podemos ver que la variante Empates resuelve mayor cantidad de instancias, pero la combinación de desigualdades de Rompimiento de Simetrías junto a Prioridades de *branching* resolvió en promedio en menor tiempo.

Para z_{rev}, hay una muy leve ventaja de la combinación de las tres familias de cortes, como se ve en la Figura 28. Finalmente, en el caso de z_{pw}, agregar cortes no otorgó ventaja alguna respecto a las combinaciones consideradas previamente (ver Figura 29).

7.9 Mejores combinaciones encontradas

Estamos en condiciones de exhibir las mejores combinaciones encontradas en cada caso, y comparar la cantidad de instancias que puede resolver, contra el modelo original.

	Completos	Sampled
Factibilidad	(22) + SB + Empates	(22) + Preprocesamiento + SB + Priorities
Revenue	Todo	(22) + Preprocesamiento + Todos los Cortes B&C
Welfare	(22) + SB + Empates	(22) + Preprocesamiento + SB + Priorities

Tab. 7.2: Mejores combinaciones encontradas, según tipo de instancia y función objetivo.

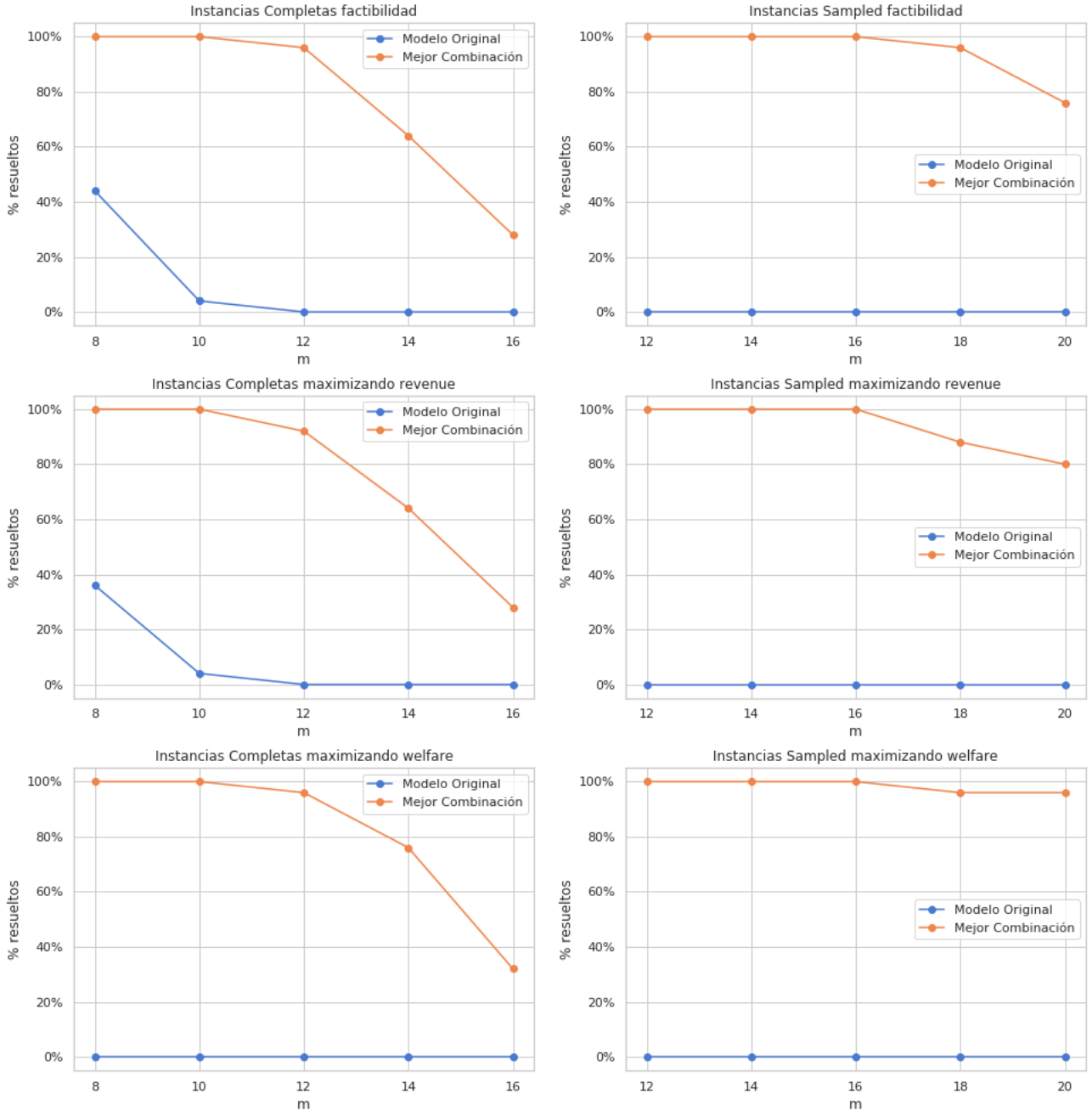


Fig. 30: Instancias *Completas* y *Sampled*, para las tres funciones objetivo consideradas. Comparamos el modelo original contra la mejor combinación de contribuciones que encontramos para cada una, siguiendo la Tabla 7.2.

La diferencia es realmente notable. En todos los casos, logramos una performance muy superior a la del modelo original. Para las instancias *Completas* y los valores de n considerados, recién a partir de $m = 12$ hay instancias que no podemos resolver, y para ese valor el modelo original no puede resolver ninguna. En el caso de las *Sampled* la diferencia es aún más significativa: para los tamaños considerados, logramos resolver un gran porcentaje de las instancias, en contraposición con el modelo original que no logra resolver ninguna.

7.10 Efecto en las instancias holgadas

Vimos que, para las instancias ajustadas, las desigualdades válidas de precio-apuesta más alta (22) provocaron un gran salto en la performance de resolución del modelo para las instancias ajustadas. ¿Sucederá algo parecido para las instancias holgadas? Decidimos evaluar esto realizando el mismo experimento que en la Sección 5.6, pero comparando el modelo original con aquel que incluye la desigualdad en cuestión.

En el caso de factibilidad, no observamos una gran diferencia entre ambas variantes. Agregar la desigualdad permitió resolver una pequeña cantidad de instancias nuevas, pero también hubo algunas instancias en las que resultó perjudicial. Más interesante fue cuando comparamos al tomar z_{rev} como función objetivo.

En primer lugar, podemos observar en la Figura 31 que el aporte de la desigualdad es más moderado que para las instancias ajustadas. Se logran resolver mayor cantidad de instancias, pero también hay unas pocas combinaciones de n y valores de b para los cuales implican un retroceso.

Pero, como dijimos, el aporte se ve más claramente cuando evaluamos la cantidad de instancias que podemos resolver a optimalidad. En la Figura 32 mostramos el número de

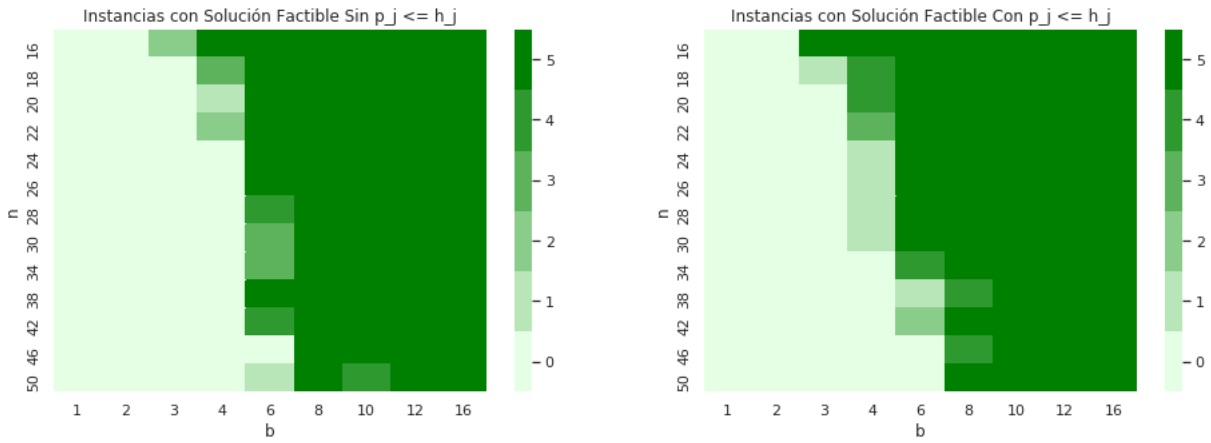


Fig. 31: Instancias *Completas*, con función objetivo *revenue*. A la izquierda, cantidad de instancias con solución factible sin la desigualdad (22), y a la derecha con dicha desigualdad.

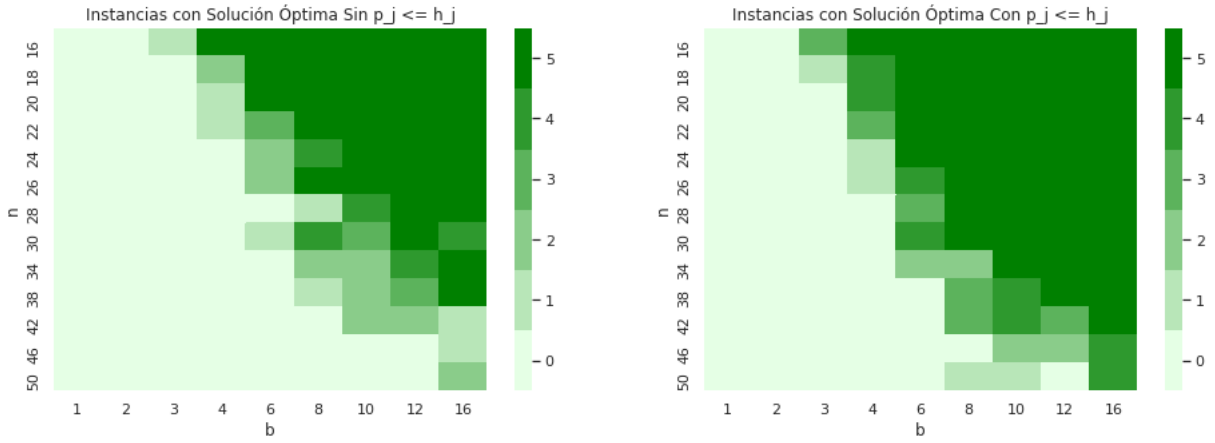


Fig. 32: Instancias *Completas*, con función objetivo *revenue*. A la izquierda, cantidad de instancias resueltas a optimalidad sin la desigualdad (22), y a la derecha con dicha desigualdad.

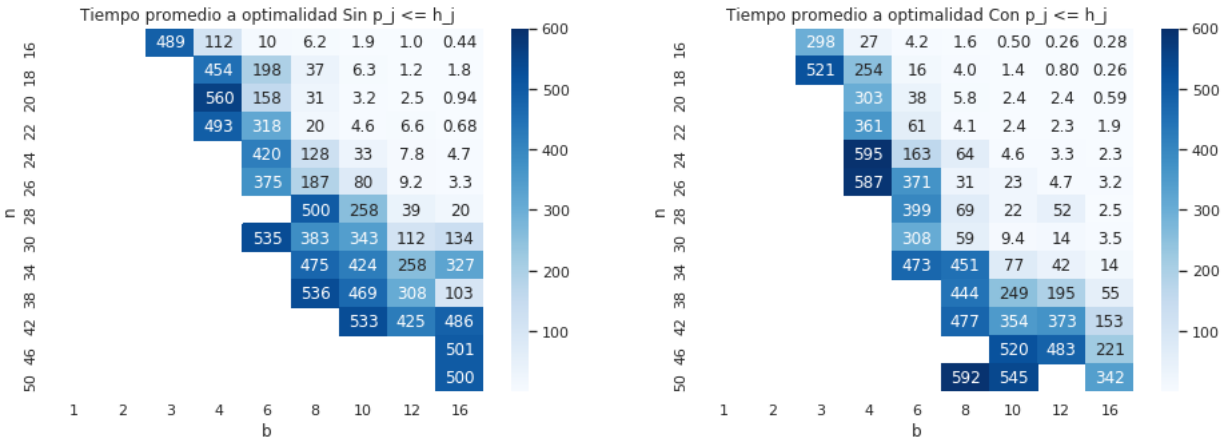


Fig. 33: Instancias *Completas*, con función objetivo *revenue*. Tiempos promedio hasta optimalidad, sin y con la desigualdad (22). Para las instancias no resueltas, se tomó t=600 segundos.

instancias resueltas sin y con la desigualdad (22), y se puede ver una clara diferencia en favor de la versión que la incluye. En total, se logran resolver 215 y 283 instancias respectivamente, lo que representa un incremento de más del 30 %. En la Figura 33 mostramos el tiempo promedio hasta obtener la solución óptima (considerando a las que no terminaron como 600 segundos para comparar más fielmente), y se ve la ventaja al agregar la desigualdad.

8. CONCLUSIONES Y TRABAJO FUTURO

En esta tesis se aborda el problema de búsqueda de *pacing equilibriums* para mercados de subastas en el contexto de las publicidades online mediante un algoritmo exacto basado en técnicas de PLEM. Para ello, se toma como punto de partida la formulación presentada en Conitzer et al. [8], que es reforzada a fin de tener un modelo más eficiente. Se proponen seis familias de desigualdades válidas que explotan características del modelo y se las evalúa experimentalmente a fin de encontrar la combinación de ellas que mejores resultados ofrece, según la función objetivo considerada y el tipo de instancia.

Los resultados obtenidos muestran que al agregar las mejores combinaciones encontradas, el algoritmo resulta muy superior, pudiendo resolver instancias de tamaños bastante más grandes. Destacamos la contribución de una familia de desigualdades válidas en particular, la que resultó determinante en este aspecto.

Por otro lado, analizamos experimentalmente la dificultad práctica de encontrar un PE en función de la magnitud de los presupuestos. Mostramos que cuando estos son ajustados el problema es en general difícil, y, a medida que se incrementan los presupuestos, esta dificultad decrece significativamente, a la par de la aparición de jugadores con presupuestos holgados.

Finalmente, esta tesis deja varias líneas de investigación a futuro abiertas, entre ellas:

1. *Heurísticas para las instancias ajustadas:* En el Capítulo 5 mostramos que, dada una instancia ajustada y un PE para ella, si disminuimos uniformemente los presupuestos de los jugadores, podemos deducir otro equilibrio para la nueva situación. Dado que las instancias con presupuestos más altos son más sencillas, se podría plantear una heurística para instancias que se intuyan ajustadas, que busque un reescalamiento de los presupuestos para el cual se pueda encontrar un PE. A partir de éste, se podría deducir un PE para la instancia original.

2. *Evaluar la dificultad según el porcentaje de jugadores holgados:* En este trabajo evaluamos la dificultad de las instancias en función de los presupuestos. Si pudiéramos generar consistentemente instancias cuyos equilibrios tengan el mismo porcentaje de jugadores holgados, ello permitiría un análisis de la dificultad en función del mismo.

3. *Embeber este programa en un esquema de resolución de instancias más grandes:* Si bien pudimos incrementar el tamaño de las instancias que podemos resolver, aún estamos lejos de los miles de millones de anunciantes e ítems que suceden en la realidad. Existen mecanismos de compactación de instancias reales, a fin de poder utilizar modelos como el analizado en este trabajo, y finalmente a partir de los PE encontrados realizar una asignación en la instancia original. Si logramos resolver de manera exacta instancias de tamaños más grandes, entonces podríamos utilizar sus

resultados para evaluar la calidad de los algoritmos de compactación de instancias y de extensión de sus Pacing Equilibriums.

Bibliografía

[1] Santiago R Balseiro, Omar Besbes, and Gabriel Y Weintraub. Repeated auctions with budgets in ad exchanges: Approximations and design. *Management Science*, 61(4):864–884, 2015.

[2] Santiago R Balseiro and Yonatan Gur. Learning in repeated auctions with budgets: Regret minimization and equilibrium. *Management Science*, 65(9):3952–3968, 2019.

[3] Santiago R. Balseiro, Anthony Kim, Mohammad Mahdian, and Vahab S. Mirrokni. Budget management strategies in repeated auctions. In *WWW*, 2017.

[4] Denis Charles, Deeparnab Chakrabarty, Max Chickering, Nikhil R Devanur, and Lei Wang. Budget smoothing for internet ad auctions: a game theoretic approach. In *Proceedings of the fourteenth ACM conference on Electronic commerce*, pages 163–180, 2013.

[5] Hana Choi, Carl F Mela, Santiago Balseiro, and Adam Leary. Online display advertising markets: A literature review and future directions. *Columbia Business School Research Paper*, (18-1), 2019.

[6] Vašek Chvátal. *Linear programming*. Macmillan, 1983.

[7] Vincent Conitzer, Christian Kroer, Debmalya Panigrahi, Okke Schrijvers, Eric Sodomka, Nicolás E Stier-Moses, and Chris Wilkens. Pacing equilibrium in first-price auction markets. *arXiv preprint arXiv:1811.07166*, 2018.

[8] Vincent Conitzer, Christian Kroer, Eric Sodomka, and Nicolás E Stier-Moses. Multiplicative pacing equilibria in auction markets. *arXiv preprint arXiv:1706.07151*, 2017.

[9] George Bernard Dantzig. *Linear programming and extensions*, volume 48. Princeton university press, 1998.

[10] Edmund Eisenberg and David Gale. Consensus of subjective probabilities: The pari-mutuel method. *The Annals of Mathematical Statistics*, 30(1):165–168, 1959.

[11] Matteo Fischetti, Fred Glover, and Andrea Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.

[12] Chinmay Karande, Aranyak Mehta, and Ramakrishnan Srikant. Optimizing budget constrained spend in search advertising. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 697–706, 2013.

[13] Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984.

[14] Christian Kroer, Alexander Peysakhovich, Eric Sodomka, and Nicolás E Stier-Moses. Computing large market equilibria using abstractions. *arXiv preprint arXiv:1901.06230*, 2019.

[15] IBM. IBM ILOG CPLEX 12.9 User's Manual. *https://www.ibm.com/analytics/cplex-optimizer*, 2019.

[16] William Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance*, 16(1):8–37, 1961.

Apéndice

A. EJEMPLOS

Ejemplo 6 (Puede haber *pacing* sin empates)**.** *Puede suceder que en un equilibrio todos gasten todos su presupuesto, pero aún así no haya empates. Ver, por ejemplo, la situación de la Figura 34.*

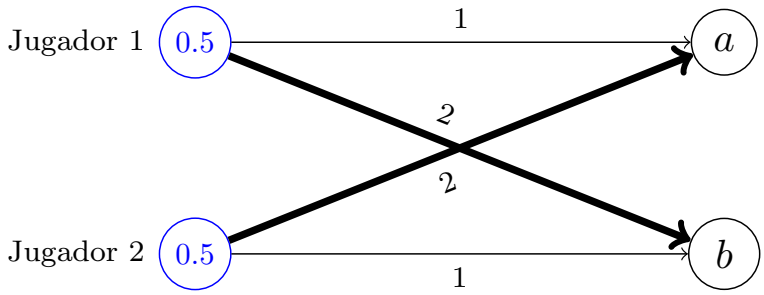


Fig. 34

Se puede ver que un equilibrio se obtiene considerando $\alpha_1 = \alpha_2 = \frac{1}{2}$. Con esos multiplicadores, el Jugador 2 gana la totalidad del ítem a, y el Jugador 1 la totalidad del ítem b, ambos por un precio de 0.5. Por lo tanto, ambos jugadores gastan todo su presupuesto, pero no hubo empate por ninguno de los dos ítems. Se puede ver que, además, dicho equilibrio es único.

B. RESULTADOS

B.1 Con vs. Sin desigualdad precio-apuesta más alta

Mostramos a continuación los resultados del experimento 7.1, agrupando según el valor de n.

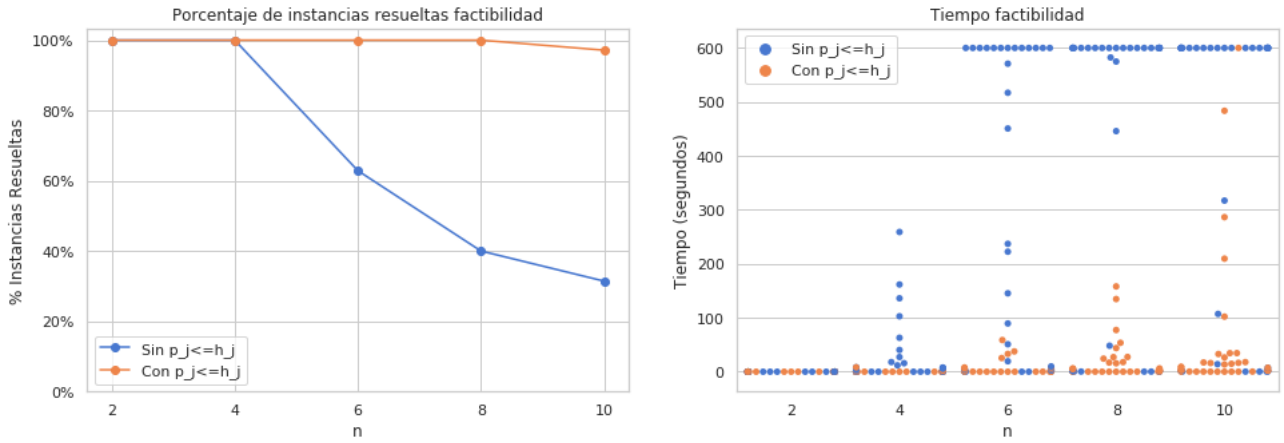


Fig. 35: Instancias *Completas*, factibilidad con y sin la desigualdad (22)

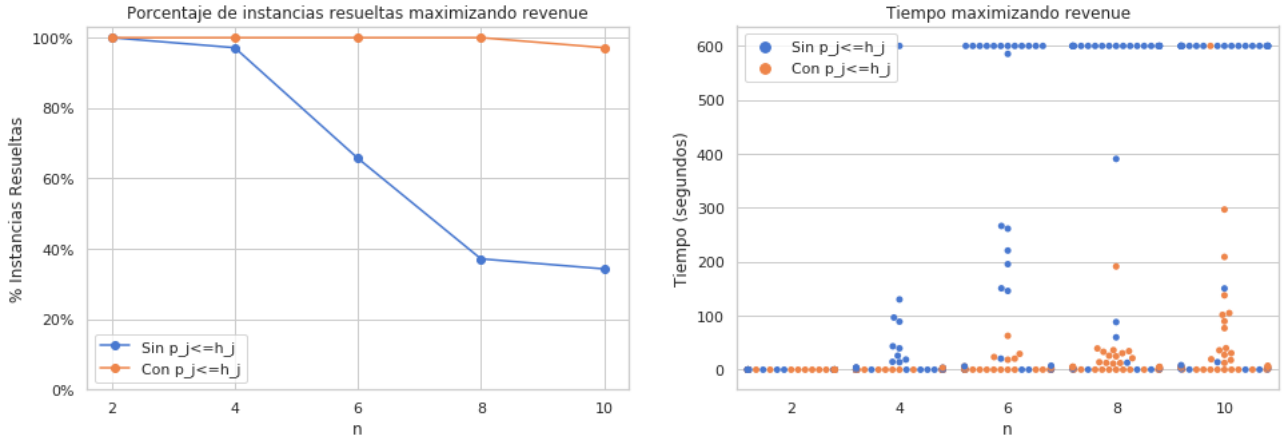


Fig. 36: Instancias *Completas*, maximizando z_{rev} con y sin la desigualdad (22)

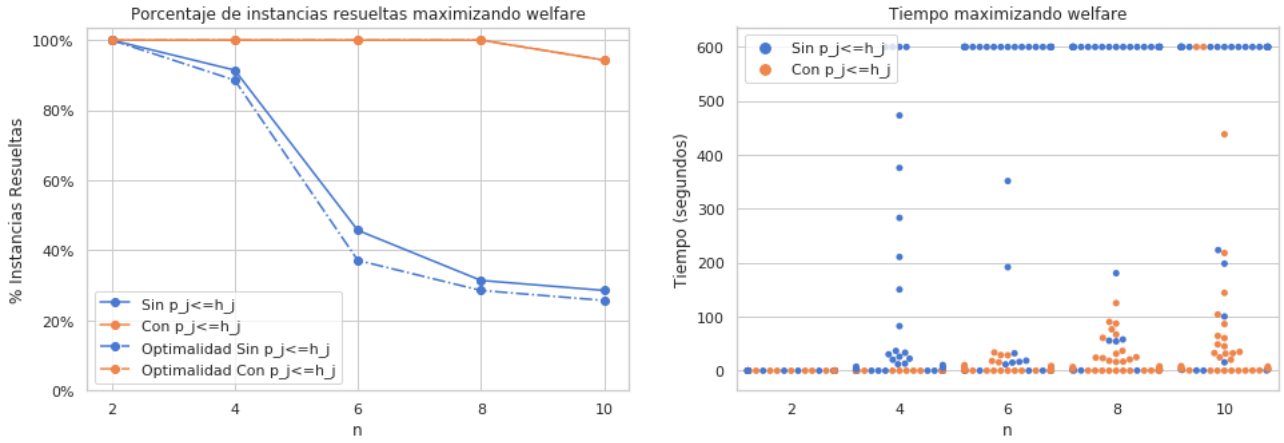


Fig. 37: Instancias *Completas*, maximizando z_{pw} con y sin la desigualdad (22)

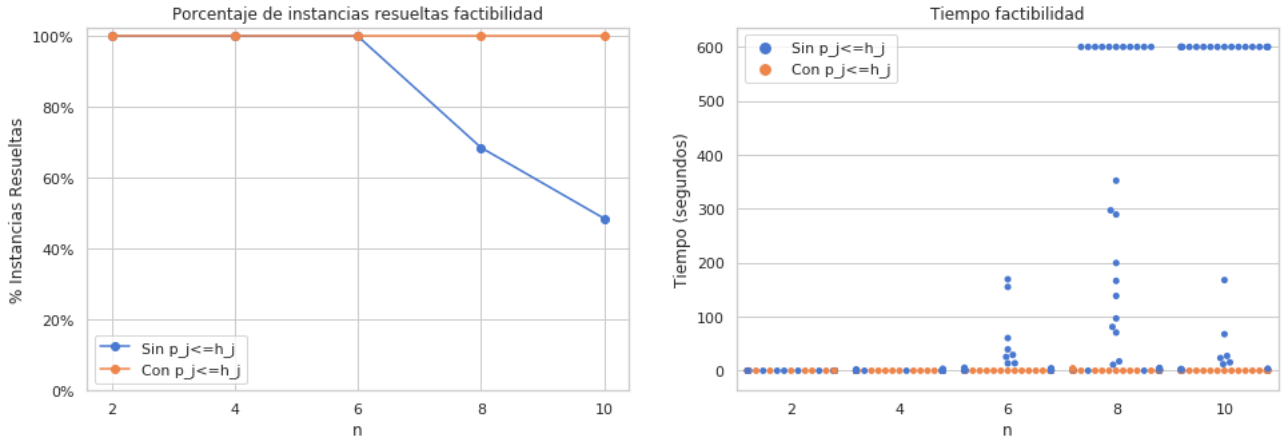


Fig. 38: Instancias *Sampled*, factibilidad con y sin la desigualdad (22)

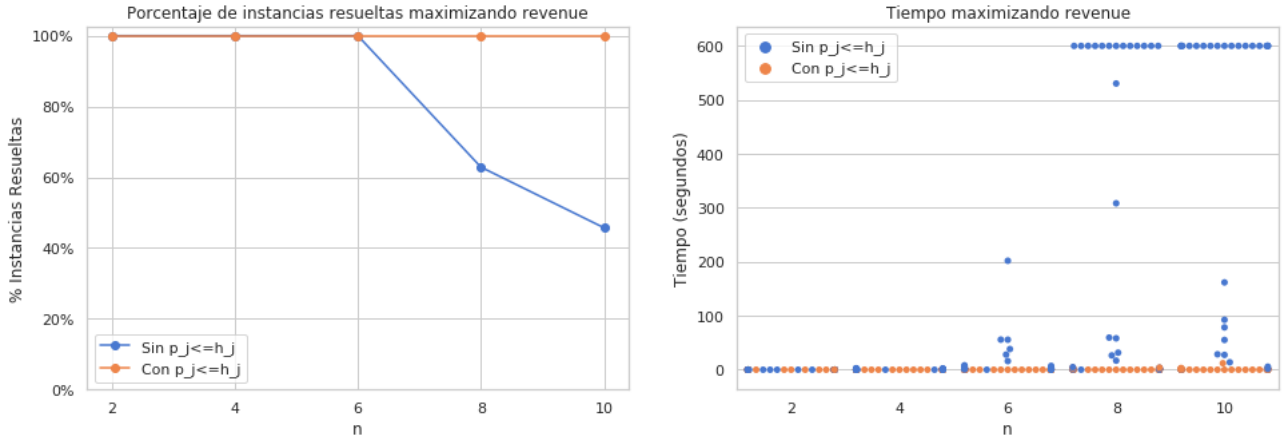


Fig. 39: Instancias *Sampled*, maximizando z_{rev} con y sin la desigualdad (22)

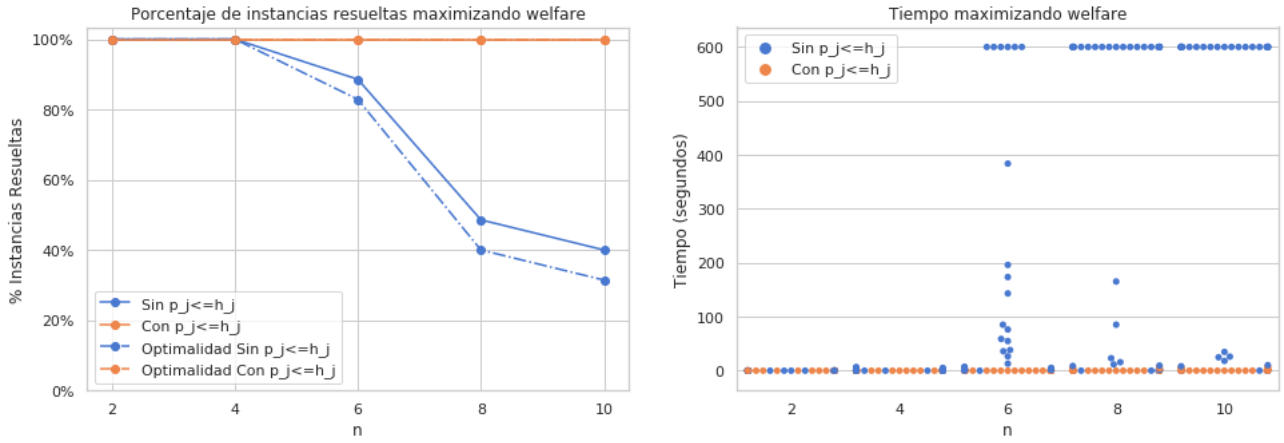


Fig. 40: Instancias *Sampled*, maximizando z_{pw} con y sin la desigualdad (22)

B.2 Prioridades

Mostramos los resultados de ambas prioridades de *branching* para las combinaciones restantes de tipo de instancia y función objetivo.

B.2.1 Instancias *Completas*

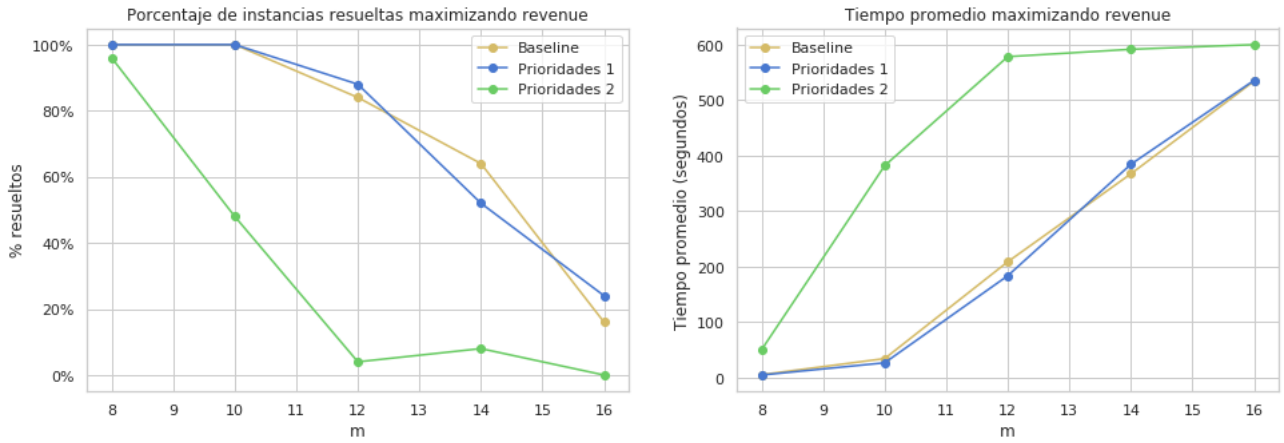


Fig. 41: Instancias *Completas*, maximizando *revenue*.

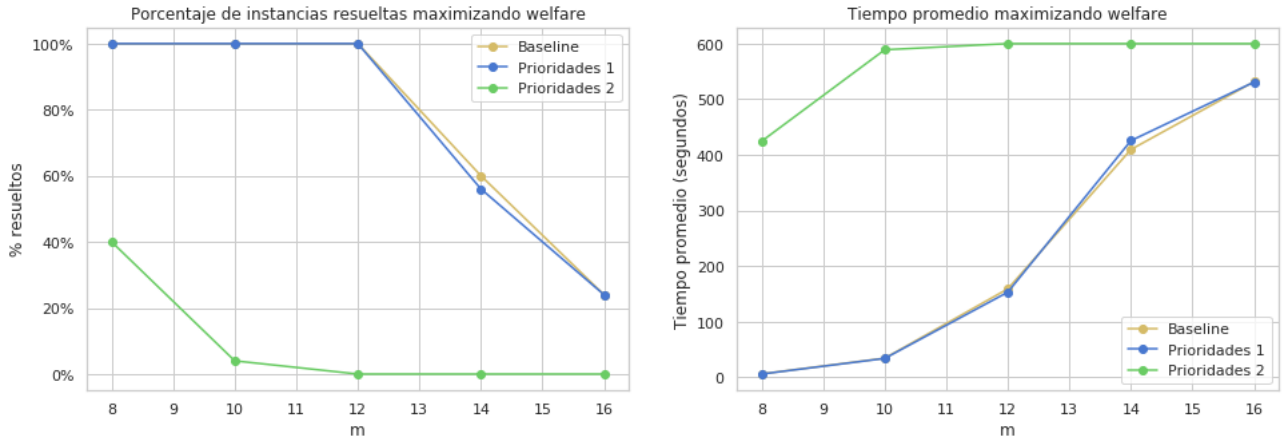


Fig. 42: Instancias *Completas*, maximizando *welfare*.

B.2.2 Instancias *Sampled*

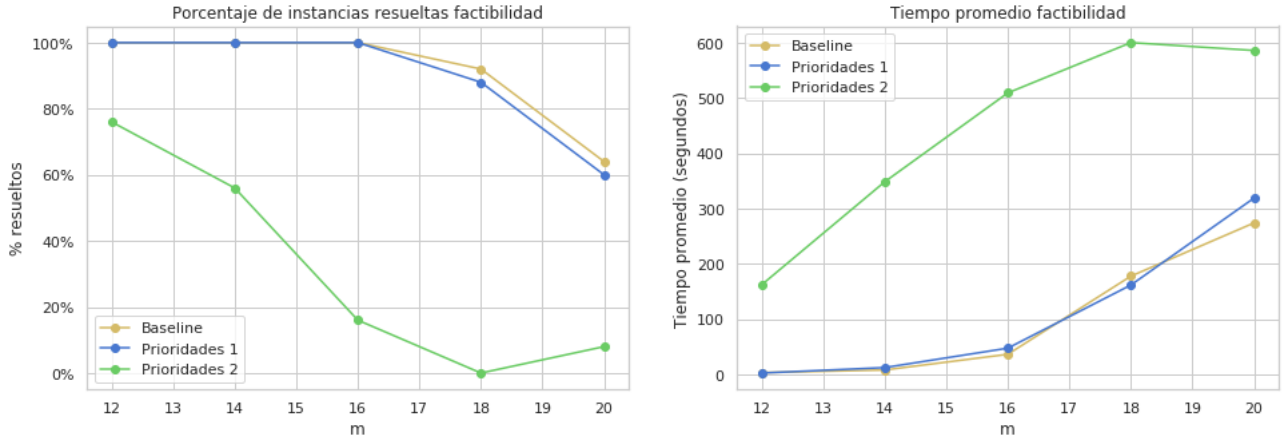


Fig. 43: Instancias *Sampled*, función objetivo constante.

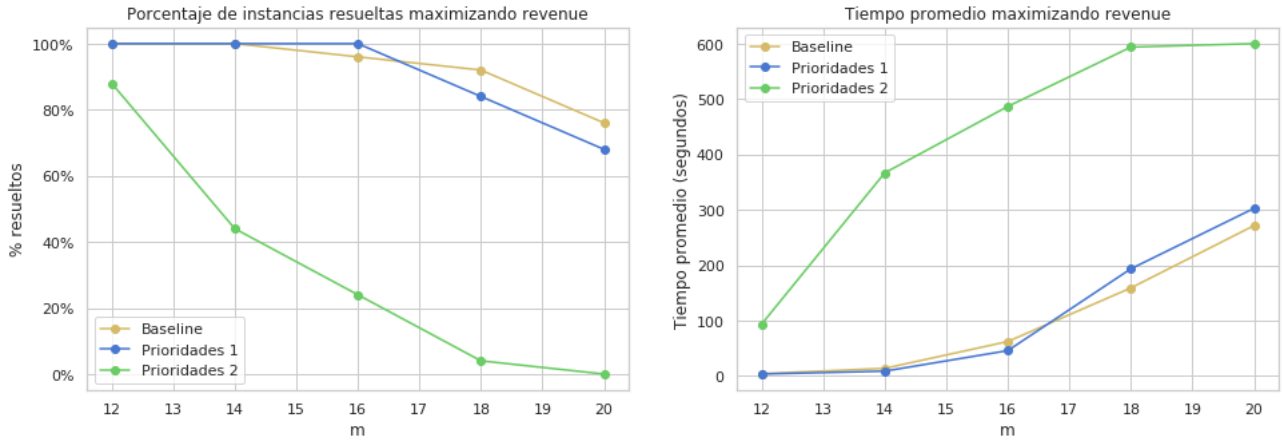


Fig. 44: Instancias *Sampled*, maximizando *revenue*.

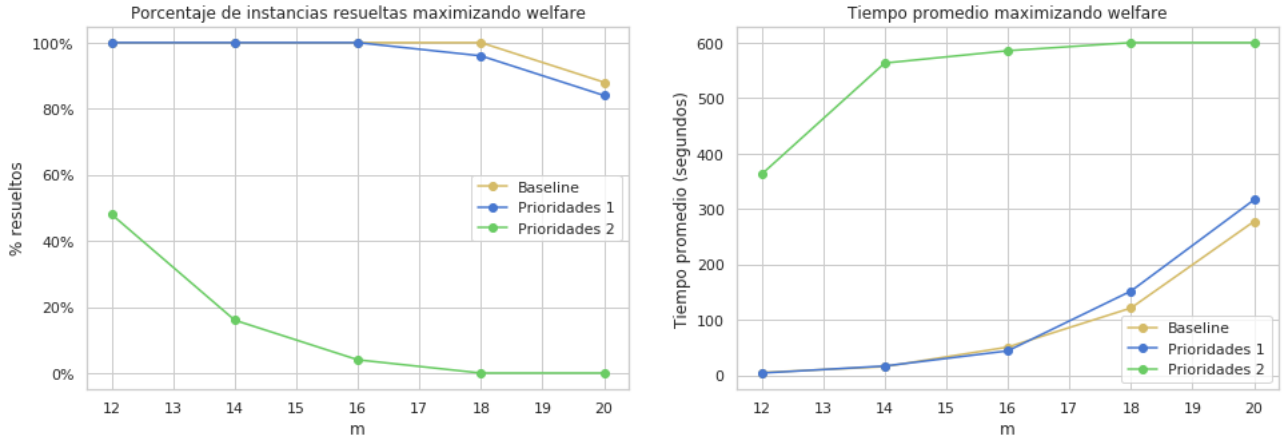


Fig. 45: Instancias *Sampled*, maximizando *welfare*.