CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA

PROGRAMA DE POSGRADO EN CIENCIAS EN CIENCIAS DE LA COMPUTACIÓN

USO DE LA TEORÍA DE JUEGOS PARA MODELAR EL PROBLEMA DE ASIGNACIÓN DE GUARDAESPALDAS

TESIS

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

MAESTRO EN CIENCIAS

Presenta:

DANIEL BRUBECK SALCEDO

Ensenada, Baja California, México, Diciembre de 2011

RESUMEN de la tesis de **DANIEL BRUBECK SALCEDO**, presentada como requisito parcial para la obtención del grado de MAESTRO EN CIENCIAS en CIENCIAS DE LA COMPUTACIÓN . Ensenada, Baja California, Diciembre de 2011.

USO DE LA TEORÍA DE JUEGOS PARA MODELAR EL PROBLEMA DE ASIGNACIÓN DE GUARDAESPALDAS

Resumen aprobado por:

Dr. José Alberto Fernández Zepeda
Director de Tesis

En este trabajo se estudia el problema de asignación de guardaespaldas (PAG) desde la perspectiva de la teoría de juegos. Este problema lo propuso Fajardo-Delgado *et al.* (2010) y consiste en construir un árbol de esparcimiento en grafos conectados donde se presentan dos tipos de nodos: blancos y negros; los nodos blancos buscan minimizar su distancia a la raíz mientras que los nodos negros buscan maximizarla. Una solución del PAG es un árbol en el que se cumple una condición de equilibrio y se maximiza el beneficio social. En este trabajo se analizan los distintos tipos principales de juegos que existen en la literatura y se clasifica el juego del PAG con base en sus características.

Una característica importante del juego del PAG, es que puede tener más de una configuración que satisface una condición de equilibrio y en el que no todas ellas maximizan el beneficio social. Así, en este trabajo se proponen seis variantes del juego que buscan mejorar la calidad de los equilibrios encontrados. Cuatro de estas variantes, consisten en la implementación de las estrategias: voraz, ε-voraz, anti-voraz e IA-voraz. La diferencia entre estas estrategias radica en el mecanismo de decisión empleado por cada nodo. Otra variante es la implementación de un nuevo calendarizador denominado anti-voraz; este calendarizador está diseñado para trabajar en conjunto con la estrategia anti-voraz y su objetivo es minimizar los cambios en el beneficio social del sistema. Los resultados experimentales muestran que la estrategia anti-voraz bajo este calendarizador incrementa de forma significativa la calidad del equilibrio encontrado para el enfoque cooperativo. La última variante del juego, consiste en una nueva función de preferencias que utiliza el pago promedio obtenido en el tiempo con el fin de que los nodos sean más "precavidos" al evaluar sus opciones durante el juego.

Por otro lado, Fajardo-Delgado *et al.* (2010) también proponen el algoritmo CBAP para encontrar soluciones aproximadas para el PAG. Este algoritmo garantiza una condición de equilibrio para el PAG que no necesariamente maximiza el beneficio social. En este trabajo, se diseña la técnica de especulación; esta técnica se diseña para el enfoque cooperativo y permite romper de forma segura un equilibrio encontrado por el algoritmo CBAP. Se proponen cuatro variantes para el algoritmo CBAP que implementan la técnica de especulación. Las variantes se diferencian en la cantidad de información que utilizan y las reglas empleadas para realizar movimientos de especulación. Se demuestra

que estas variantes siempre terminan con una configuración que satisface una condición de equilibrio y que los beneficios sociales de estos equilibrios son iguales o mejores que los obtenidos por el algoritmo CBAP. Asimismo, se demuestra que las variantes propuestas mantienen el orden de complejidad temporal del algoritmo CBAP. Finalmente, se presentan resultados experimentales donde se comparan las variantes propuestas del juego del PAG y las variantes del algoritmos CBAP.

Palabras Clave: Teoría de juegos, problema de asignación de guardaespaldas, sistemas distribuidos.

ABSTRACT of the thesis presented by **DANIEL BRUBECK SALCEDO**, in partial fulfillment of the requirements of the MASTER IN SCIENCES degree in COMPUTER SCIENCE. Ensenada, Baja California, December 2011.

USING GAME THEORY TO MODEL THE BODYGUARD ALLOCATION PROBLEM

In this thesis, we study the bodyguard allocation problem (BAP) from a game theory perspective. This problem was proposed by Fajardo-Delgado *et al.* (2010) and its objective is to build a spanning tree in a connected graph in which there are two types of nodes: white and black. The white nodes seek to minimize their distance to the root while black nodes seek to maximize it. A solution of the BAP is a tree which satisfies a condition of equilibrium and maximizes social welfare. In this work, we analyze the main types of games that exist in literature and classify the BAP game based on its characteristics. An important feature of the BAP game is that it has more than one configuration that satisfies a condition of equilibrium, and not all of them maximize social welfare. Thus, in this thesis we propose six variants of the game which seeks to improve the quality of the equilibrium found by a previous existing algorithm. Four of these variants consist on the following strategies: greedy, ε-greedy, anti-greedy and RS-greedy. The difference between these strategies lies in the decision mechanism used by each node. Another variant is the implementation of a new scheduler called anti-greedy. We designed this scheduler to work with the anti-greedy strategy and its objective is to minimize changes in the social welfare of the system. Experimental results show that the anti-greedy strategy enhances the quality of the equilibrium found for the cooperative approach when it works in conjunction with this scheduler. The last variant of the proposed game is a new preference function that uses the average payment over time giving "cautious" behavior to the nodes when evaluating their options during the game.

On the other hand, Fajardo-Delgado *et al.* (2010) propose the CBAP algorithm to find approximate solutions for the BAP. This algorithm guarantees to finish with an equilibrium condition for the BAP, but it does not necessarily maximize social welfare. In this thesis, we design a new technique called the speculation technique, this technique works for the cooperative approach and it allows the algorithm to break safely the equilibrium found by the CBAP algorithm. We propose four variants of the CBAP algorithm that implement the speculation technique. The variants differ in the amount of information and the rules used in order to make speculation moves. We show that these variants always end with a configuration that satisfies a condition of equilibrium and the social welfare of these equilibria are equal to or better than those obtained by the CBAP algorithm. We also show that these variants maintain the time complexity of the CBAP algorithm. Finally, we present experimental results that compare the proposed variants of the BAP game and the variants of the CBAP.

Keywords: game theory, bodyguard allocation problem, distributed systems.

A mis padres

Agradecimientos

A mis padres, hermanos y mi tía Mónica, por todo el apoyo y comprensión durante esta etapa, así como aquellos sacrificios que hicieron para que pudiera alcanzar uno de mis sueños.

A mi titular de tesis, el Dr. José Alberto Fernández por guiarme y aconsejarme durante el desarrollo de esta tesis.

A los miembros del comité de tesis por sus valiosas aportaciones, en especial al *Dr. Carlos Alberto Brizuela*, por sus ideas y consejos de las juntas semanales.

A *Daniel Fajardo* por sus consejos, revisiones y apoyo durante la tesis.

A *Vicky* por alentarme a continuar cuando se comenzaba a notar el cansancio del tiempo invertido en la maestría y de escribir la tesis.

A mis compañeros de generación, *José, Eduardo, Gabriel, Ramón, Moises, Marco, Miguel, Martín, Hector, Karina, Gerardo, Limberg, Iván, Paúl, Mauricio, Ulises, Jessica* por los momentos compartidos, hacer de los momentos difíciles de la maestría más fáciles de llevar y todas aquellas noches de desvelo estudiando.

A *Hanna* por ayudarme a decidirme a entrar en la maestría y todo el apoyo que me dió durante el primer año.

A todos los investigadores, estudiantes y personal del departamento de ciencias de la computación por su enseñanza académica.

Al CONACyT y proyecto de investigación por su apoyo económico.

Contenido

Contenido (continuación)

Contenido (continuación)

Lista de Figuras

Lista de Figuras (continuación)

Lista de Tablas

Capítulo I

Introducción

En la literatura comúnmente se define un sistema distribuido como una colección de procesos autónomos que cooperan entre sí para alcanzar un objetivo global común. Un ejemplo de un sistema distribuido son las bases de datos distribuidas en las que la información se encuentra repartida entre distintos servidores de bases de datos. En los sistemas distribuidos convencionales, la cooperación entre los procesos es lo que permite llegar a un objetivo global. Sin embargo, esta cooperación requiere de algoritmos y protocolos que coordinen dicha cooperación.

Cuando se estudian sistemas distribuidos similares a Internet, se pueden encontrar dos situaciones que dificultan la cooperación y por tanto dificultan llegar a buenos resultados globales:

- Para poder tomar decisiones, cada nodo debe estar consciente del estado global en el que se encuentra, este traspaso de información es muy tardado cuando los sistemas son muy grandes. Por lo tanto, es deseable que los algoritmos tomen sus decisiones con base en conocimientos locales (Nisan *et al.*, 2007).

- Este tipo de sistemas distribuidos los han ido construyendo distintas entidades, por lo que el creador de un algoritmo o protocolo debe tomar en cuenta que las entidades que lo implementen tratarán de beneficiarse y utilizar el algoritmo a conveniencia; es decir, las entidades pueden comportarse de forma egoísta y cooperar poco con el sistema distribuido (Nisan *et al.*, 2007).

Para ejemplificar, este tipo de problemas se pueden observar en los sistemas de inter-

cambio de archivos basados en redes peer to peer (P2P), donde cada nodo del sistema distribuido sirve como cliente y servidor de forma que los nodos descargan archivos de otros nodos sin necesidad de un servidor central; así, el objetivo global del sistema es compartir la mayor cantidad de archivos y el objetivo local de cada nodo es descargar la mayor cantidad de archivos. Por lo general cada nodo se comporta de forma egoísta; es decir, cada nodo intenta maximizar su velocidad de descarga y al mismo tiempo intenta reducir su velocidad de subida (incluso puede llegar a no compartir archivos). En estos casos, si se permite que cada nodo se comporte en forma egoísta, entonces el sistema P2P no tiene un buen rendimiento global.

En estos casos, el uso de objetivos individuales en sistemas distribuidos generan conflictos debido a que cada proceso intenta beneficiarse a sí mismo, al mismo tiempo que intenta cooperar para alcanzar el objetivo global. La teoría de juegos ha estudiado de forma extensiva a los sistemas que siguen este tipo de comportamientos (von Neumann y Morgenstern, 1944). La teoría de juegos estudia situaciones de competencia y cooperación entre individuos racionales que toman sus decisiones con base en la interacción con otros individuos. Así, la teoría de juegos permite analizar distintos problemas en sistemas distribuidos donde está presente un comportamiento egoísta por parte de los nodos; problemas de ruteo de paquetes (Griffin *et al.* (2002), Roughgarden y Tardos (2002), Roughgarden (2005)), balanceo de cargas (Koutsoupias y Papadimitriou (2009), Grosu y Chronopoulos (2004)), redes de sensores (Machado y Tekinay (2008), Charilas y Panagopoulos (2010)) y servicios P2P (Moscibroda *et al.* (2006), Feldman *et al.* (2004)) son ejemplos de problemas en los que los procesos se comportan de forma egoísta.

Uno de los retos principales para poder aplicar la teoría de juegos en problemas de sistemas distribuidos es ver cómo modelar el problema aplicando los conceptos de la

teoría de juegos. En Nisan (1999) se presenta una serie de modelos para utilizarse en problemas de agentes en un sistema distribuido donde los problemas requieren calcular una función que depende de las entradas de n agentes; en el modelo propuesto se tiene que especificar la función a calcular (objetivo global), los objetivos locales de cada uno de los agentes y los pagos que los agentes manipulan. En Nisan y Ronen (1999) introducen el *diseño de mecanismos algorítmicos* para estudiar cómo combinar una conducta egoísta con una conducta cooperativa en los agentes para alcanzar una condición de equilibrio con máximo beneficio social; es decir, se utiliza el diseño de mecanismos para determinar las reglas del juego que permiten que los agentes lleguen a un objetivo global cuando maximizan su objetivo local. Los mecanismos propuestos en Nisan y Ronen (1999) suponen que existe una unidad central de confianza que recolecta la información de todos los procesos del sistema para poder tomar una decisión. Poco tiempo después, en (Feigenbaum y Shenker, 2002) se extiende el modelo propuesto por Nisan y Ronen (1999) donde al modelo centralizado, además de tener los mismos objetivos (estimular la compatibilidad y que al mismo tiempo que sea factible computacionalmente), los agentes y la información relevante deja de estar centralizada y pasa a ser distribuida. Creando así los diseños de mecanismos para algoritmos en ambientes distribuidos (DAMD).

Existen varios trabajos que estudian y modelan problemas de grafos o redes utilizando un enfoque basado en la teoría de juegos. En (Dürr y Thang, 2007) se estudia el problema de la ubicación de instalaciones en ambientes competitivos, el cual consiste en buscar la localización que maximice la cantidad de entidades que van a ese local. Los autores modelan este problema a través de la teoría de juegos, más específicamente de los juegos de Voronoi por ser este tipo de juegos un modelo geométrico simple para este problema; sin embargo, modelan el juego de Voronoi en un grafo de n vértices y k

jugadores. En dicho juego, cada jugador tiene que seleccionar un vértice (localización) para después añadir a cada localidad los vértices que representan a las entidades que van a la localidad, cada jugador intenta maximizar la cantidad de vértices que se añaden a su localidad seleccionada.

En (Alon *et al.*, 1995) se propone una solución basada en juegos sobre grafos. En este problema el algoritmo debe controlar el movimiento de un conjunto k de servidores, los cuales se representan por medio de un espacio métrico M, al mismo tiempo debe manejar los pedidos que igualmente se encuentran en el espacio métrico. Cada vez que llega un pedido, el algoritmo debe determinar a qué servidor debe mover al punto donde está el pedido para que se atienda, en el problema se desea minimizar el movimiento de servidores.

En (Boulogne y Altman, 2005) se propone un modelo para poder resolver el problema de ruteo competitivo en comunicaciones multicast; este problema trata sobre la decisión que tienen los proveedores de servicio para determinar la ruta a través de la cual van a enviar los paquetes y cómo dividir el flujo para que llegue a todos los nodos suscritos al servicio, utilizando herramientas de teoría de juegos para resolverlo.

En (Halpern, 2007) se realiza un estudio de los temas principales en la teoría de juegos, enfocándose en el trabajo a realizar en el área de ciencias de la computación. Dentro de sus tópicos considera las interacciones entre el cómputo distribuido y la teoría de juegos.

En (Kearns y Tan, 2008) se soluciona un problema de grafos no dirigidos que denominan el problema del democrático principal. Este problema establece que dado un conjunto de miembros de una población distribuida, éstos deben balancear sus preferencias sobre los candidatos para alcanzar la unidad colectiva; en este caso restringen a que cada nodo tome un color con la misma probabilidad (puede ser rojo o azul) y con

base en decisiones locales deben llegar todos los nodos a un solo color.

Por último, Chapman (2009) utiliza la teoría de juegos como marco de trabajo para resolver problemas de optimización en sistemas multi-agentes. Chapman (2009) modela los problemas como juegos con equililbrios de Nash por medio de estrategias puras y propone distintas estrategias distribuidas para crear algoritmos que encuentren el equilibrio de Nash del juego modelado.

I.1. Planteamiento del problema

En (Fajardo-Delgado *et al.*, 2010) se introduce un juego de red distribuido, en el cual se supone que existen dos tipos de procesos y cada tipo tiene objetivos individuales opuestos; sin embargo, estos procesos deben cooperar para poder llegar a un objetivo común global. La función de utilidad de cada proceso depende de ciertos parámetros en el sistema (grado de conectividad, ancho de banda, distancia, etc.) tal que un tipo de proceso intente maximizar el parámetro mientras que el otro intenta minimizarlo, y al mismo tiempo, los dos tipos intentan contribuir a la utilidad global del sistema. En este trabajo se supone un sistema distribuido cuya función de utilidad depende de la distancia entre los nodos a un nodo especial y se le denomina el *Problema de Asignación de Guardaespaldas* (PAG).

Fajardo-Delgado *et al.* (2010) definen el PAG de la siguiente forma: existe una persona (el cliente) cuya protección está a cargo de $(n-1)$ guardaespaldas en una casa de n cuartos. Los cuartos de la casa se conectan entre ellos a través de corredores. La casa se puede ver como un grafo, donde cada nodo representa un cuarto y cada corredor que conecta dos cuartos es una arista. El cliente y los guardaespaldas se colocan en cuartos distintos de la casa. El objetivo de los guardaespaldas es proteger al cliente de la mejor

forma posible. Cada guardaespaldas puede proteger únicamente un corredor. Los corredores sin protección se cierran. Cada corredor abierto permite la comunicación entre los cuartos que están conectados al mismo. Los objetivos son mantener comunicación en todos los cuartos de la casa y que el cliente esté lo más seguro posible. El criterio para determinar el nivel de seguridad del cliente se basa en la habilidad individual de cada guardaespaldas. Los guardaespaldas expertos requieren estar tan cerca como sea posible del cliente, mientras que los guardaespaldas novatos requieren estar tan lejos como sea posible del cliente. Por lo tanto, los guardaespaldas expertos desean minimizar la distancia con el cliente y los novatos maximizarla.

Formalmente, se tiene un grafo $G = (V_G, E_G)$ que representa a la casa, donde cada nodo $u \in V_G$ representa un cuarto y cada arista $e \in E_G$ representa un corredor que comunica dos cuartos. Se divide el conjunto de vértices V_G en tres conjuntos disjuntos V_1, V_2 y V_3 (de acuerdo con el tipo de persona que está en el cuarto), tal que $|V_1 \cup V_2 \cup V_3| = n$. El conjunto V_1 representa al conjunto de cuartos que tienen a un guardaespaldas experto y se representan con nodos de color blanco; el conjunto V_2 representa al conjunto de cuartos que tienen a un guardaespaldas novato y se representan con nodos de color negro. El conjunto V_3 está compuesto por un solo nodo $\{r\}$ que representa el cuarto que tiene al cliente.

Una solución del PAG representa un árbol de esparcimiento $T = (V_T, E_T)$ enraizado en r, tal que $V_T = V_G$ y $E_T \subseteq E_G$; donde el árbol T es un equilibrio con el máximo beneficio social (un equilibrio es una configuración del árbol en la que ningún nodo puede mejorar de forma individual su ganancia). Cada arista $e \in E_T$ representa un corredor protegido por un guardaespaldas. Maximizar el nivel de seguridad en la casa es equivalente a encontrar el árbol de esparcimiento T que maximiza la función de

beneficio social $g(T)$. La Ecuación 1 define el valor de $g(T)$.

$$g(T) = \sum_{\forall u \in V_1} f_1(u) + \sum_{\forall u \in V_2} f_2(u) \tag{1}$$

La Ecuación 1 representa la suma de la utilidad individual de todos los vértices de V_1 y V_2. Donde $f_1 : V_1 \to \mathbb{Z}$ y $f_2 : V_2 \to \mathbb{Z}$ denotan la utilidad de los nodos blancos y negros, respectivamente. Las Ecuaciones 2 y 3 definen a las funciones f_1 y f_2, respectivamente.

$$f_1(u) = n - d_u^T \tag{2}$$

$$f_2(u) = d_u^T \tag{3}$$

Donde d_u^T es la distancia entre los nodos u y r en el árbol T, para todo $u \in V_1 \cup V_2$.

Ejemplo 1. La Figura 1a muestra un grafo completo G que modela el PAG, donde $n = 3$, $V_1 = u$, $V_2 = v$ y $V_3 = r$. La Figura 1b muestra una solución para el PAG como un árbol T construido en G. Note que u minimiza su distancia a r, mientras que v la maximiza.

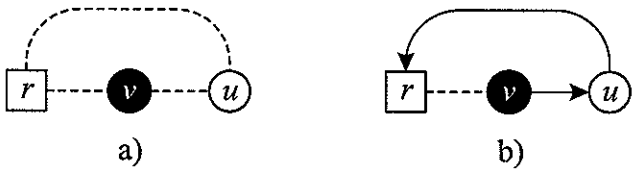

Figura 1. Problema de asignación de guardaespaldas; a) Un caso del PAG modelado como un grafo G; b) El árbol construido en G que representa una solución del PAG para este caso.

Fajardo-Delgado *et al.* (2010) modelan el PAG como un juego definido por el juego del PAG y lo estudian desde dos enfoques distintos: el enfoque no cooperativo y el cooperativo. La diferencia entre ambos enfoques se encuentra en cómo calculan sus pagos los jugadores. En el enfoque no cooperativo, los jugadores prefieren conexiones que incrementan el valor de su función objetivo local; mientras que en el enfoque cooperativo, los jugadores prefieren conexiones que mejoran el pago de su sub-árbol.

Juego 1. *Juego del PAG.* El Juego del PAG $\mathcal{G}$ es una tupla $(\Lambda, \{V_1 \cup V_2\}, S_u, \pi_u)$, donde:

- Λ representa un caso arbitrario de $\mathcal{G}$ que incluye un grafo conectado $G = (V_G, E_G)$ y una partición del conjunto de nodos V_G en tres subconjuntos V_1, V_2 y V_3.
- $\{V_1 \cup V_2\}$ son el conjunto de nodos que representan a los jugadores racionales del juego. El nodo especial r, donde $\{r\} = V_3$, es un jugador pasivo que no toma ninguna acción.
- S_u es el conjunto de *estrategias* posibles para cada jugador $u \in V_1 \cup V_2$.
- π_u es la función de pagos que calcula las preferencias para cada jugador $u \in V_1 \cup V_2$ en el grafo.

Fajardo-Delgado *et al.* (2010) también proponen dos algoritmos que encuentran un equilibrio para el juego del PAG en tiempo polinomial bajo los dos enfoques. En ambos algoritmos se construye un árbol inicial enraizado en r y cada nodo u sigue una estrategia del conjunto de estrategias S_u, donde $S_u = \{$maximizar la distancia a r, minimizar la distancia a $r\}$ hasta que el grafo alcanza una configuración en equilibrio. La diferencia entre los algoritmos es que un algoritmo está ideado para un sistema centralizado (CBAP); mientras que el segundo es un algoritmo ideado para un sistema distribuido (DBAP). Ambos algoritmos entregan soluciones aproximadas al PAG; es decir, el equilibrio encontrado no siempre es el equilibrio que maximiza el beneficio social.

I.2. Objetivo general

Los objetivos generales de este trabajo de tesis son clasificar el juego del PAG en los tipos de juegos que existen en la teoría de juegos y analizar variantes tanto del juego

como del algoritmo CBAP para que el algoritmo CBAP modificado pueda encontrar configuraciones en equilibrio con mejor beneficio social.

I.3. Contribución al conocimiento

Las contribuciones principales de esta tesis son las siguientes:

- Teoría de juegos en problemas computacionales: esta investigación va a profundizar en el empleo de la teoría de juegos para poder resolver problemas en el área de ciencias de la computación.
- Estudiar impacto de cooperación: esta investigación ayudará a entender para este problema qué tanto afecta al sistema cuando los nodos toman actitudes cooperativas o egoístas.
- Simulación del comportamiento del problema: esta investigación ayudará a comprender cómo se comporta este problema en un sistema distribuido utilizando un simulador que sea centralizado.

I.4. Organización de la tesis

El documento de tesis se organiza de la siguiente manera. En el Capítulo II se hace un estudio sobre la teoría de juegos y los tipos de juegos. En el Capítulo III se define el modelo del sistema, en donde se realizan las suposiciones básicas y la terminología que se utiliza a lo largo de este trabajo de investigación. Se realiza el análisis del juego del PAG y se proponen distintas variantes del juego para obtener configuraciones en equilibrio con mejor beneficio social. El Capítulo IV presenta una variante del algoritmo CBAP

para el PAG bajo el enfoque cooperativo que garantiza entregar resultados iguales o mejores que los resultados del algoritmo CBAP. En el Capítulo V se extiende la variante propuesta del Capítulo IV con una variante del algoritmo CBAP que obtiene mejores resultados que el algoritmo propuesto en el Capítulo IV a cambio de un incremento en el tiempo de ejecución. En el Capítulo VI se introduce una variante que puede aplicarse a las variantes de los capítulos IV y V. Esta variante garantiza resultados igual o mejores que los de las variantes que extiende. En el Capítulo VII se analizan resultados experimentales para las variantes del juego del PAG y del algoritmo CBAP. Finalmente, en el Capítulo VIII se presenta un resumen, las conclusiones de este trabajo de investigación y trabajo a futuro.

Capítulo II

Teoría de juegos

La teoría de juegos es una rama de las matemáticas que a través de sus modelos ayuda a comprender situaciones en las cuales interactúan distintos individuos *racionales* a través de sus tomas de decisiones. La racionalidad de un jugador se encuentra en la consistencia de sus decisiones cuando se encuentra con distintos conjuntos de acciones y no en la naturaleza de sus gustos o disgustos. La interacción de los individuos se hace a través de juegos. Un *juego* es una actividad donde compiten dos o más jugadores que buscan maximizar sus ganancias de acuerdo a un conjunto de reglas, es decir, los jugadores seleccionan la acción que maximice su beneficio de acuerdo a sus preferencias. Las *reglas* de un juego definen las características del juego; entre éstas se encuentran las acciones que pueden tomar los jugadores, los pagos de los jugadores (éstos definen las preferencias del jugador sobre el conjunto de acciones seleccionadas por todos los jugadores); los turnos de los jugadores y la información del juego que reciben los jugadores (Osborne, 2004).

En general, para representar un juego se tienen que definir al menos los siguientes aspectos:

- *Jugadores*: Son los agentes que toman las decisiones durante el juego, es necesario definir cuántos jugadores existen y los distintos tipos de jugadores.
- *Acciones*: Las decisiones que puede tomar un jugador; cada jugador tiene su conjunto de acciones.

- *Preferencias de los jugadores*: Cada jugador debe tener preferencias sobre sus acciones y tener alguna forma de evaluar si prefiere realizar una acción sobre otra, por lo general esto se realiza mediante la función de pago.

Un jugador realiza un *movimiento* cuando selecciona una acción en su turno. La *estrategia* de un jugador define el comportamiento del mismo durante el juego y está compuesta por el conjunto de movimientos que realiza el jugador desde que comienza hasta que termina el juego; la expresión s_i denota la estrategia para un jugador i. A lo largo de este capítulo, la expresión s_{-i} representa las estrategias de todos los jugadores excluyendo al jugador i. Una *solución* o *perfil de estrategias* de un juego son las estrategias de todos los jugadores que producen cierto pago para los jugadores. Los *pagos* de un jugador definen las preferencias del jugador para cada perfil de estrategias posible. La mejor estrategia que puede tomar un jugador depende, en general, de los movimientos realizados por los otros jugadores. Debido a esto, los jugadores deben tener en mente las posibles estrategias de los otros jugadores y formular creencias sobre éstas (Osborne, 2004).

II.1. Tipos de juegos

En la teoría de juegos, los juegos se agrupan en clases o tipos con base en características similares del juego (objetivos de los jugadores, intereses, estrategias, información del juego y conceptos de solución). Esta agrupación ayuda a identificar qué tipo de juego es el más apropiado para un problema específico y el concepto de solución adecuado; existen dos tipos de juegos principales que contienen a todos los tipos de juegos, los juegos *no cooperativos* y los juegos *cooperativos*; los objetivos de estos juegos son distintos entre sí, por lo tanto, el concepto de solución es diferente para estos tipos.

II.1.1. Juegos no cooperativos

Los juegos no cooperativos estudian la interacción de individuos con intereses propios, donde las decisiones que toma cada individuo afecta de forma positiva o negativa a los resultados de los otros individuos. Cada individuo tiene sus propias acciones y sus preferencias sobre dichas acciones. Sin embargo, las preferencias de cada individuo incluyen las acciones seleccionadas por todos los jugadores. Como las decisiones del jugador se ven afectadas por las acciones de los otros jugadores, necesita también formar creencias sobre las acciones que van a tomar los otros jugadores para determinar cuál es la acción que prefiere (Osborne, 2004).

El término "no cooperativo" puede malinterpretarse, ya que se puede pensar que estos juegos solo se aplican en situaciones donde los intereses de los individuos entran en conflicto; sin embargo, a éste tipo de juegos se les denomina como "no cooperativos" en el sentido de que los individuos son la unidad principal (incluyendo sus acciones, preferencias y creencias); es decir, cada individuo toma sus decisiones con base en sus preferencias personales sin importarle si afecta a otros jugadores; así, los jugadores pueden llegar a cooperar cuando la cooperación los lleve a mejorar sus pagos (Shoham y Leyton-Brown, 2009).

En los juegos no cooperativos, puede cambiar la forma de representar el juego, las estrategias o el concepto de solución dependiendo de las características del juego; así, estos juegos se dividen en sub-tipos de juegos que tienen características similares. Existen dos características que definen a grandes rasgos la representación y el concepto de solución de los juegos:

- La forma en que los jugadores toman decisiones.
- La cantidad de información que conocen los jugadores.

De acuerdo con la forma en que los jugadores toman sus decisiones, los juegos no cooperativos se dividen en dos sub-tipos: *juegos estáticos* o *simultáneos* y *juegos dinámicos*; la diferencia de estos sub-tipos se encuentra en si los jugadores pueden seleccionar su acción al mismo tiempo o deben esperar su turno. De igual forma, existen tres sub-tipos de acuerdo con la información que conocen los jugadores: *juegos de información completa*, *juegos de información incompleta* y *juegos de información perfecta*. En los juegos no cooperativos, estos sub-tipos de juegos se mezclan para formar cuatro sub-tipos de juegos principales, cada uno de estos sub-tipos tiene una representación y concepto de solución distinto.

Nota: En (Osborne, 2004) se tratan a los juegos estáticos con información completa como juegos de información perfecta. Esto es, como las decisiones se toman de forma simultánea, entonces la información del juego es perfecta (no se puede saber algo más). En este trabajo también se van a tratar como juegos de información perfecta cuando sean juegos estáticos para mantener uniformidad en los nombres de las clases de juegos.

Juegos estáticos con información perfecta

En los juegos estáticos con información perfecta se modela la interacción simultánea entre jugadores; los jugadores conocen las acciones y los pagos que obtiene cada jugador. Como la interacción es simultánea, los jugadores desconocen las acciones realizadas por los otros jugadores; sin embargo, cada jugador puede formular creencias sobre las estrategias que siguen los otros jugadores ya que conoce sus acciones y sus pagos (Osborne, 2004). Siguiendo la definición de Osborne (2004), un juego estático con información perfecta consiste de:

- Un conjunto de *jugadores*.

- Un conjunto de *acciones* para cada jugador.
- Las *preferencias* sobre el conjunto de perfil de acciones para cada jugador.

Un ejemplo de un juego estático de información perfecta es el *juego del dilema del prisionero*. El nombre del juego viene de una historia que involucra a dos sospechosos de un crimen; este juego es un ejemplo clásico de la teoría de juegos ya que existen muchas situaciones en las cuales dos participantes se encuentran con incentivos similares a los sospechosos en la historia (Osborne, 2004).

Juego 2. *Dilema del prisionero.* Se retiene a dos sospechosos de un crimen mayor en celdas distintas. La policía tiene evidencia suficiente para culpar a los dos sospechosos de un crimen menor; sin embargo, la evidencia no es suficiente para culpar a cualquier sospechoso del crimen mayor. La única forma de poder acusar a un sospechoso por el crimen mayor es que el otro sospechoso proporcione información sobre él (*confiese*). Si ambos sospechosos permanecen en silencio, a cada uno se la va a culpar por la ofensa menor y ambos van a pasar un año en prisión. Si solo uno de ellos confiesa, el sospechoso que confiesa sale libre y al sospechoso traicionado se le culpa por el crimen mayor por lo que pasa 3 años en prisión. Si ambos sospechosos confiesan, los dos son acusados del crimen mayor pero se les reduce un año de prisión por cooperar.

Modelando el juego de acuerdo a la definición de Osborne (2004):

- *Jugadores*: Los dos sospechosos.
- *Acciones*: Los dos jugadores tienen las mismas acciones Silencio, Confesar.
- *Preferencias*: Ambos sospechosos quieren minimizar el tiempo que pasen en la cárcel.

En general, estos juegos suelen representarse a través de una matriz de pagos de n dimensiones, donde n es el número de jugadores y cada componente de la dimensión representa una acción del jugador; cada celda de la matriz contiene el pago para todos los jugadores cuando seleccionan la acción que los lleva a esa celda. La Tabla II.1.1 muestra la matriz de pagos para el juego del dilema del prisionero. En la Tabla II.1.1, los renglones representan las acciones del sospechoso 1 y las columnas las acciones del sospechoso 2. Cada celda representa el par de acciones seleccionadas, donde el primer número es el pago obtenido (años de cárcel) por el sospechoso 1, y el segundo, el pago obtenido por el sospechoso 2.

Tabla I. Juego del dilema de prisionero utilizando matriz de pagos.

		Sospechoso 2	
		Confesar	Silencio
Sospechoso 1	Confesar	2, 2	0, 3
	Silencio	3, 0	1, 1

Una solución deseada para un juego no cooperativo es cuando los jugadores alcanzan el equilibrio de Nash. El *equilibrio de Nash* es una solución a un juego en el que todos los jugadores seleccionan la acción que maximiza sus pagos de acuerdo a la acción seleccionada de los otros jugadores; dado que los jugadores son racionales y todos los jugadores están en equilibrio, ningún jugador tiene incentivos para cambiar de acción ya que ésta es la que maximiza sus pagos; esto es, para cada jugador i,

$$u_i(a_i^*, a_{-i}^*) \geq u_i(a_i, a_{-i}^*) \text{ para cada acción } a_i \text{ del jugador } i \tag{4}$$

En el juego del dilema del prisionero, el equilibrio de Nash se encuentra cuando ambos jugadores *confiesan*. Note que en este juego, el equilibrio de ambos jugadores es inde-

pendiente de la acción seleccionada por el otro jugador; sin embargo, existen juegos en los que los jugadores no pueden llegar a un equilibrio seleccionando una sola acción.

Cuando los jugadores no pueden llegar a un equilibrio con una sola acción, se busca una solución estable donde los jugadores seleccionen sus estrategias de forma aleatoria. Una *estrategia mixta* es una distribución de probabilidades sobre las acciones de los jugadores. Dado que las estrategias mixtas utilizan decisiones aleatorias, los jugadores no pueden determinar cuál va a ser su pago; por lo tanto, en estos juegos se supone que los jugadores son *neutrales a riesgos* y seleccionan la distribución de probabilidad que maximiza su *pago esperado* (Nisan *et al.*, 2007). Así, se adapta el concepto de equilibrio de Nash para estrategias mixtas como una solución en la que todos los jugadores seleccionan una distribución de probabilidad para sus estrategias que maximizan su pago esperado dada la distribución de probabilidad que seleccionaron los otros jugadores sobre sus acciones. En Nash (1951) se demostró que en los juegos no cooperativos con estrategias finitas, siempre existe al menos un equilibrio de Nash para estrategias mixtas.

Juegos dinámicos con información perfecta

En los juegos estáticos con información perfecta, todos los jugadores seleccionan sus acciones al mismo tiempo; así, los jugadores seleccionan un solo plan de acción (estrategia) y una vez elegido no pueden cambiarlo. Sin embargo, existen situaciones en las que los jugadores pueden ir adaptando su estrategia conforme se va desarrollando el juego. En los juegos dinámicos se hace explícita la estructura secuencial de la toma de decisiones para permitir que los jugadores seleccionen su acción de acuerdo a las acciones que se han seleccionado a lo largo del juego (Osborne, 2004). En los *juegos dinámicos con información perfecta*, además de hacer explícito el orden en que los jugadores toman sus

decisiones, cada jugador conoce toda la información sobre las acciones previas del juego (Fudenberg y Tirole, 1991). En el Juego 3 se presenta un ejemplo donde es importante que la toma de decisiones sea dinámica.

Juego 3. *Juego de la entrada.* Un titular se enfrenta a la posible entrada de un retador. Las acciones del retador es *entrar* o *salir* de la competencia. Si el retador entra a la competencia, el titular puede *ceder* el título o *pelear* por mantenerlo; si el retador sale de la competencia, entonces el titular no tiene que hacer nada. Las preferencias de los jugadores son las siguientes: si el retador entra a la competencia, ambos jugadores prefieren que el titular *ceda* el título; es decir, pelear es costoso para ambos; el titular prefiere que el retador decida *salir* de la competencia; es decir, mantiene el título sin necesidad de pelear por él. El peor caso para ambos es cuando el retador decide *entrar* y el titular decide *pelear.*

Al igual que en los juegos estáticos de información perfecta, para modelar un juego dinámico con información perfecta se necesitan definir los jugadores y las preferencias de los jugadores sobre todos los resultados. Debido a la estructura secuencial en la toma de decisiones, en los juegos dinámicos con información perfecta no basta con especificar las acciones de los jugadores, ya que el orden de los jugadores y las acciones de los mismos puede ir cambiando conforme se toman decisiones en el juego; por lo tanto, se especifica el orden en que mueven los jugadores y las acciones disponibles para el jugador en cada toma de decisión. Así, para modelar un juego dinámico de información perfecta se tiene que definir el conjunto de todas las secuencias de acciones que pueden ocurrir y el jugador que toma una decisión en un punto de la secuencia dada. En estos juegos, la *historia terminal* representa una secuencia de posibles acciones para todos los jugadores; la *función jugador* define el orden en que juegan los jugadores conforme

se avanza en la historia terminal (Osborne, 2004). El modelo del Juego 3 de acuerdo a la definición de Osborne (2004) es el siguiente:

- *Jugadores*: El retador y el titular.
- *Historias terminales*: (*entrar,ceder*), (*entrar,pelear*), (*salir*).
- *Función jugador*: $P(\emptyset) = Retador$ y $P(entrar) = Titular$
- *Preferencias*: Las preferencias del retador están dadas por la función de pago u_1, donde $u_1(entrar, ceder) = 2$, $u_1(salir) = 1$ y $u_1(entrar, pelear) = 0$. Las preferencias del titular están dadas por la función de pago u_2, donde $u_2(salir) = 2$, $u_2(entrar, ceder) = 1$ y $u_2(entrar, pelear) = 0$.

Así como los juegos estáticos se representan a través de sus matrices de pagos, los juegos dinámicos con información perfecta se representan mediante árboles de juego. Un *árbol de juego* es un árbol enraizado, donde cada nodo del árbol representa el punto donde un jugador toma una decisión; los nodos del árbol se conocen como *nodos de decisión*; las hojas del árbol se conocen como *nodos terminales*. Dado que cada nodo terminal determina una ruta única hacia la raíz, los pagos de los jugadores se asignan en los nodos terminales con base en una función $u_i : Z \rightarrow \mathbb{R}$, donde $u_i(z)$ representa el pago del jugador i si se llega al nodo terminal z. En el árbol de juego, las aristas de salida de un nodo representan el conjunto finito de acciones A que puede realizar el jugador i en ese nodo de decisión (Fudenberg y Tirole, 1991). La Figura 2 modela el Juego 3 mediante un árbol de juego.

En los juegos dinámicos de información perfecta, la *estrategia* de un jugador es una función que asigna una acción para cada nodo de decisión donde el jugador tiene el turno. La solución de un juego es un perfil de estrategias s de todos los jugadores que

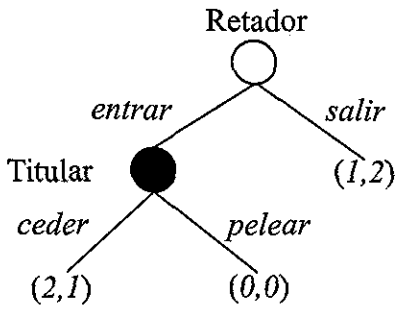

Figura 2. Árbol de juego para el Juego 3.

determina la historia terminal $O(s)$ que ocurre cuando se aplican las estrategias de todos los jugadores. El equilibrio de Nash se define como un perfil de estrategias en el que cada jugador utiliza la estrategia que maximiza su pago dadas las estrategias de los otros jugadores. Esto es, para cada jugador i

$$u_i(O(s_i^*, s_i^*)) \geq u_i(O(s_i, s_i^*)) \text{ para cada estrategia } s_i \text{ del jugador } i \qquad (5)$$

Donde u_i es la función de pago que representa las preferencias del jugador i y O es la función de salida del juego (Osborne, 2004). Dado que las estrategias tienen seleccionadas todas las acciones antes de comenzar el juego, al aplicar el equilibrio de Nash en los juegos dinámicos de información perfecta se pierde la estructura secuencial del juego; así se pierden ciertos equilibrios que dependen de los movimientos durante el juego. Un ejemplo se da en el Juego 3, donde la mejor estrategia para el titular es *pelear* por el puesto (intimidar al retador para que no entre); sin embargo, el titular puede *ceder* el puesto cuando el retador *entra* en la competencia.

Debido a la falta de robustez en la definición del equilibrio de Nash, los juegos dinámicos de información perfecta adaptan el concepto de equilibrio para que no se pierda la estructura secuencial del juego; así, la estrategia del jugador debe ser óptima conforme se va desarrollando el juego y no solo al inicio del juego (Osborne, 2004). Para poder obtener las estrategias óptimas a lo largo de un juego, el juego se descompone

en subjuegos denominados subjuegos propios, permitiendo así encontrar las acciones óptimas para un nodo de decisión específico (Fudenberg y Tirole, 1991). Un *subjuego propio* es un pedazo del juego que puede analizarse por si mismo; es decir, en un árbol de juego G, es un subárbol G' en el que se incluyen todos los nodos de decisión a partir de un nodo de decisión del árbol (Nisan *et al.*, 2007).

Para un juego dinámico con información perfecta G, el perfil de estrategias s es un *equilibrio de subjuegos perfectos* si s es un equilibrio de Nash para cada subjuego propio G' de G (Fudenberg y Tirole, 1991).

Juegos estáticos de información imperfecta

En los juegos estáticos de información perfecta los jugadores desconocen la acción que seleccionan los otros jugadores; es decir, se desconocen los movimientos, pero todos los jugadores conocen el juego. En los juegos estáticos de información imperfecta, los jugadores presentan incertidumbres acerca del juego que se está jugando; es decir, los juegos estáticos de información imperfecta modelan situaciones en las cuales los participantes desconocen características relevantes para un juego estático. Así, un jugador puede desconocer el número de jugadores, las acciones disponibles para los jugadores o el pago que obtiene por cada acción (Shoham y Leyton-Brown, 2009). El Juego 4 es un ejemplo de un juego estático de información imperfecta en el que los jugadores desconocen cierta información relevante para predecir la estrategia de los otros jugadores; es decir, cada jugador sólo conoce el valor que él le da al producto; sin embargo, desconoce el valor que le dan los otros jugadores y la cantidad de dinero que van a ofrecer por el producto. Dado que los jugadores no conocen toda la información relevante acerca del juego, los jugadores forman sus creencias sobre los otros jugadores utilizando probabilidad; por lo tanto, los juegos estáticos con información imperfecta se conocen como

juegos bayesianos.

Juego 4. *Subastas a primer precio.* Se tiene una subasta en la que se ofrece un producto a un grupo de jugadores, cada jugador i tiene un valor v_i del producto y de forma simultánea, todos los jugadores ofrecen una cantidad p_i por el producto. La ganancia del jugador i es $u_i = v_i - p_i$ si gana la subasta y $u_i = 0$ en otro caso.

Los juegos bayesianos se representan como un juego con distintos estados, dependiendo del estado, son los pagos que reciben los jugadores. Un *estado* es una descripción completa de las características relevantes de un juego (las preferencias y la información de los jugadores). El Juego 5 es un ejemplo de un juego bayesiano con dos estados, donde el jugador uno desconoce en qué estado se encuentra. Adicionalmente, en los juegos bayesianos los jugadores tienen información privada que desconocen los otros jugadores (la función de pagos del jugador, las creencias del jugador sobre los otros jugadores, las acciones del jugador); un ejemplo es el valor que da cada jugador al producto en el Juego 4. El *tipo* de cada jugador i representa la información privada de i y se denota por θ_i.

Juego 5. *Batalla de los sexos bayesiano.* Dos personas están decidiendo qué actividad realizar el fin de semana (ver un juego de béisbol o softbol), una de las personas desconoce si su compañero prefiere estar con él o prefiere evitarlo. La otra persona conoce todas las preferencias sobre su compañero. La Figura 3 modela la matriz de pagos para los jugadores. El jugador uno tiene 0.5 de probabilidad de que el jugador dos prefiera estar con el jugador uno o prefiera evitarlo.

Los juegos bayesianos permiten representar situaciones en las que los jugadores tienen incertidumbres acerca del juego mismo que se está jugando, las incertidumbres

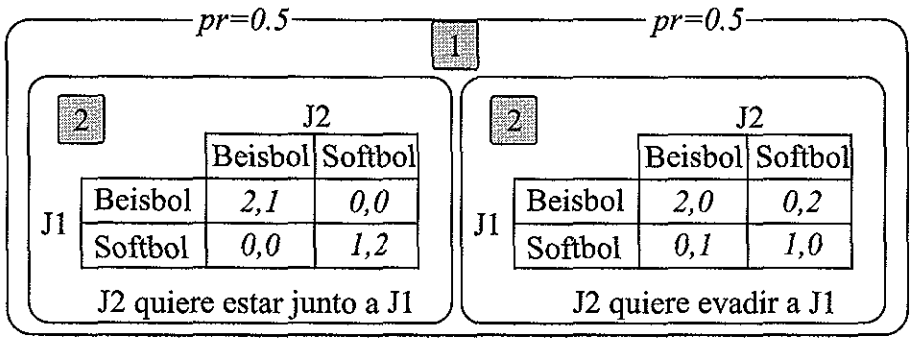

Figura 3. Matrices de pago para el juego 5.

se representan a través de distribuciones de probabilidad sobre un conjunto de juegos posibles donde se supone:

- Todos los juegos tienen el mismo número de agentes y el mismo espacio de estrategias para cada agente, solo difieren en los pagos.
- Las creencias de los jugadores se obtienen después del inicio del juego, obtenidas al recibir señales individuales y privadas.

La primer suposición parece restringir lo que puede desconocer un jugador; es decir, parece restringir a que solo pueden diferir los pagos de los jugadores. Sin embargo, los jugadores pueden desconocer distintos elementos en los juegos de información incompleta (el número de jugadores o las acciones de los jugadores). La suposición se hace ya que estas incertidumbres pueden reducirse a incertidumbres de pagos al reformular el problema (Shoham y Leyton-Brown, 2009). El segundo punto restringe a que las creencias se obtienen después del inicio del juego ya que al inicio del juego se define el estado del juego; sin embargo, los jugadores no observan este estado. Así, cada jugador recibe únicamente una señal que puede dar información sobre el juego. La señal que cada jugador i recibe al estar en el estado w se conoce como la *función señal* y se denota por $\tau_i(w)$. En los juegos bayesianos τ_i es una función determínistica; por lo

que para cada estado existe una señal finita. Los tamaños de los conjuntos de estados que son consistentes con cada una de las señales del jugador i reflejan la calidad de la información del jugador i (Osborne, 2004). Así, en el Juego 5, dado que el jugador uno desconoce el estado en el que se encuentra, entonces el jugador uno recibe la misma señal para cualquier estado del juego. Sin embargo, el jugador dos recibe una señal distinta por estado y conoce el estado del juego que está jugando.

Siguiendo la definición de Osborne (2004), para modelar un juego bayesiano, al menos se tiene que definir:

- Un conjunto de *jugadores.*
- Un conjunto de *estados.*
- Por cada jugador:
  - Un conjunto de *acciones.*
  - Un conjunto de *señales* que puede recibir y la *función señal* que asocia cada señal con un estado.
  - Por cada señal, una *creencia* acerca de los estados consistentes con la señal (distribución de probabilidad).
  - La *función de pagos esperados* sobre los pares (a, w), donde a representa un perfil de acciones y w representa un estado.

Una solución de un juego bayesiano es una colección de acciones por jugador, donde existe una acción para cada tipo de jugador; es decir, cada jugador selecciona una acción por cada señal que recibe. Así, la *estrategia pura* de cada jugador i es el mapeo $\alpha_i : \theta_i \mapsto A_i$; esto es, existe un mapeo de una acción para cada tipo de jugador.

Una *estrategia mixta* es una distribución de probabilidad sobre las estrategias puras del jugador, donde $s_i(a_i, \theta_i)$ denota la probabilidad de que el jugador i seleccione una acción a bajo la estrategia mixta s dado que es del tipo θ (Shoham y Leyton-Brown, 2009).

En los juegos estáticos de información perfecta, el equilibrio de Nash supone que cada jugador tiene la creencia correcta acerca de las acciones que van a realizar los otros jugadores. Sin embargo, en los juegos bayesianos se pierde esta propiedad ya que los jugadores desconocen información sobre el juego; de esta forma, se tienen que adaptar los conceptos de pago esperado, respuesta óptima y equilibrio de Nash para poder trabajar con la incertidumbre (Osborne, 2004). Dado que las probabilidades cambian dependiendo de la cantidad de información, en Shoham y Leyton-Brown (2009) definen tres tipos distintos de pago esperado: *ex post*, *ex interim* y *ex ante*:

- El pago esperado *ex post* representa el pago esperado obtenido cuando los jugadores conocen el tipo de todos los jugadores al recibir la señal; por lo tanto, el juego se puede modelar como un juego estático de información perfecta ya que solo se preocupa por el valor de las estrategias mixtas.

- El pago esperado *ex interim* representa el pago esperado obtenido cuando los jugadores conocen su propio tipo de jugador pero desconocen el tipo de los otros jugadores. Para que cada jugador i pueda evaluar su función de pago $u_i(a, \theta_i, \theta_{-i})$, i debe de considerar todas las asignaciones de los tipos de los otros jugadores y cada estrategia pura a.

- El pago esperado *ex ante* es el más general de los tres y representa el pago esperado obtenido cuando no se observan los tipos de los jugadores; es decir, el jugador desconoce su tipo y el de todos los jugadores.

En los juegos bayesianos, para cada jugador i, una *respuesta óptima* de i es el conjunto de mejor respuestas para un perfil de estrategias mixtas s_{-i}. La expresión $BR_i(s_{-i})$ denota la respuesta óptima del jugador i para el perfil de estrategias mixtas s_{-i} de los otros jugadores. es decir, la respuesta óptima es el conjunto de estrategias que maximizan el pago del jugador dado que los otros jugadores utilizaron cierta estrategia mixta (Shoham y Leyton-Brown, 2009). La Ecuación 6 define la respuesta óptima.

$$BR_i(s_{-i}) = max_{s'_i \in S_i} U_i(s'_i, s_{-i}) \tag{6}$$

La Ecuación 6 utiliza la definición de pago esperado ex ante ya que si una estrategia es óptima cuando ya recibió la señal, entonces debe de ser una respuesta óptima antes de recibir la señal (Shoham y Leyton-Brown, 2009). El concepto de respuesta óptima para juegos bayesianos permite adaptar el equilibrio de Nash a un nuevo concepto de equilibrio para juegos bayesianos conocido como equilibrio bayesiano de Nash. Un *equilibrio bayesiano de Nash* es una solución en la que todos los jugadores utilizan una respuesta óptima para el perfil de estrategias de los otros jugadores. Esto es, para cada jugador i:

$$s_i \in BR_i(s_{-i})$$

La diferencia del equilibrio bayesiano de Nash respecto al equilibrio de Nash es que el equilibrio bayesiano de Nash tiene definidas las bases de los juegos bayesianos (Osborne, 2004).

Dentro de los juegos bayesianos, existe una clase especial de juegos, los cuales se denominan *juegos de diseño de mecanismos (estáticos)* (Fudenberg y Tirole, 1991).

Diseño de mecanismos

El diseño de mecanismos emerge para resolver situaciones en las que la información privada de los jugadores permite que ellos se comporten de formas que no se pueden observar e incluso llegar a mentir sobre su información para alcanzar sus objetivos (Anthes, 2010). El diseño de mecanismos es una versión estratégica de la teoría de elección social; una *elección social* es una agregación de preferencias entre distintos participantes hacia una sola decisión conjunta (Nisan *et al.*, 2007). La diferencia con la teoría de elección social se encuentra en que en el diseño de mecanismos se supone que los jugadores se van a comportar de forma estratégica con el fin de maximizar sus pagos individuales (Shoham y Leyton-Brown, 2009). En estos juegos, existe una "entidad principal" que quiere condicionar sus acciones de acuerdo a la información que conocen los otros jugadores (agentes) de forma privada. La entidad puede preguntar la información a los jugadores; sin embargo, los jugadores pueden reportar a la entidad información falsa si no se les incentiva a reportar la información verdadera (Fudenberg y Tirole, 1991). El Juego 6 es un ejemplo en el que los jugadores pueden ocultar información para beneficiarse a si mismos. Si el jugador D conoce las preferencias de los otros jugadores, entonces el jugador puede ver que si vota por la actividad que prefiere (z), la actividad con más votos es y. Como el jugador D prefiere a x antes que a y, este jugador puede mentir sobre su preferencia y votar por la actividad x en vez de z, logrando así un empate entre las actividades x y y.

Juego 6. *Voto estratégico.* Se tienen cuatro jugadores (A, B, C, D) tratando de decidir cuál de tres actividades disponibles (x, y, z) van a realizar, los jugadores tienen un orden de preferencia sobre las actividades de forma que:

$$\mathbf{A} : y \succ x \succ z$$

$$\mathbf{B} : y \succ x \succ z$$

$$\mathbf{C} : x \succ z \succ y$$

$$\mathbf{D} : z \succ x \succ y$$

Donde $x \succ y$ significa que el jugador prefiere más la actividad x que la y.

El diseño de mecanismos permite diseñar un sistema donde los jugadores con el tiempo tiendan a comportarse de forma que la entidad central maximice su beneficio a pesar de estar en un ambiente de incertidumbre (Anthes, 2010). El diseño de mecanismos también se conoce como "teoría de juegos inversa" ya que en la teoría de juegos se intenta predecir las acciones que toman distintos jugadores para un juego definido, mientras que en el diseño de mecanismos se trata de predecir qué juego es el que va a dar ciertos comportamientos en los jugadores (Shoham y Leyton-Brown, 2009). Dado que el diseño de mecanismos busca lograr que los jugadores se comporten de cierta forma, los diseños de mecanismos son una buena herramienta para tratar de diseñar protocolos para sistemas distribuidos que tienen la propiedad de que los procesos pueden ser egoístas y por tanto se tenga que incentivarlos para cooperar (Shoham y Leyton-Brown, 2009).

Juegos dinámicos de información imperfecta

Los juegos dinámicos de información imperfecta modelan situaciones en las que la estructura secuencial de la toma de decisiones se hace explícita y los participantes no

tienen información completa sobre los movimientos anteriores; es decir, al momento de que un jugador toma una decisión, el jugador puede no estar informado acerca de las acciones previas de los otros jugadores que lo llevaron a un nodo de decisión específico. Los juegos dinámicos de información imperfecta se describen con el mismo modelo que los juegos dinámicos de información perfecta. Sin embargo, en los juegos dinámicos de información imperfecta también se tiene que definir la información que conocen los jugadores en cada nodo de decisión (Osborne, 2004).

La información que conoce cada jugador cuando está en un nodo de decisión se representa con conjuntos de información; donde los conjuntos de información $h_i \in H_i$ son una partición de los nodos de decisión del árbol de juego; es decir, cada nodo de decisión se encuentra exactamente dentro de un solo conjunto de información. Todos los nodos dentro de un mismo conjunto de información pertenecen al mismo jugador y tienen los mismos movimientos. Así, el jugador sólo conoce la información del conjunto de información en el que se encuentra el nodo de decisión; sin embargo, desconoce el nodo de decisión particular en el que se encuentra (Nisan *et al.*, 2007). Siguiendo la definición de Osborne (2004), estos juegos se definen de la misma forma que los juegos dinámicos de información perfecta; la única diferencia es que en estos juegos se definen los conjuntos de información para cada jugador.

El Juego 7 es un ejemplo de un juego dinámico de información imperfecta en el que el titular tiene en el mismo conjunto de información a los nodos de decisión que suceden después de que el retador decide "*prepararse* y *entrar*" o "*no prepararse* y *entrar*"; así, el titular sólo sabe que el retador decidió entrar en la pelea; sin embargo, desconoce si el retador está preparado o no para "*entrar*". La Figura 4 muestra el árbol de juego del Juego 7. En el árbol de juego, los nodos de decisión que pertenecen al mismo conjunto de información se unen a través de una arista punteada.

Juego 7. *Juego de la entrada bayesiano.* Un titular se enfrenta ante la posibilidad de la entrada de un retador. El retador tiene tres opciones, puede mantenerse fuera, prepararse y entrar o entrar sin prepararse. La preparación es costosa pero reduce la pérdida de una pelea. El titular puede pelear o ceder al retador. Una pelea es menos costosa si el retador no está preparado; sin embargo, el titular prefiere ceder a pelear sin importar que el retador esté preparado o no. El titular puede ver si el retador entra pero no puede ver si está preparado.

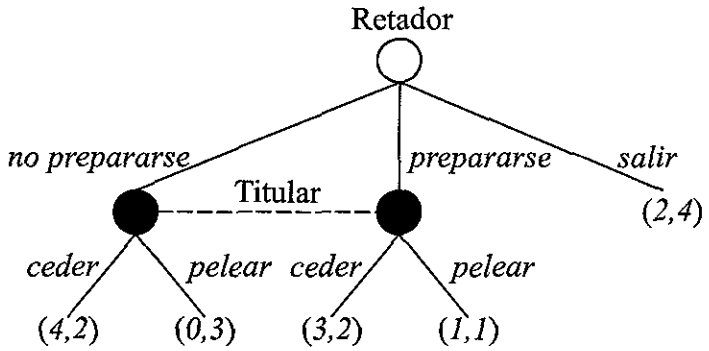

Figura 4. Árbol de juego del juego 7.

En los juegos dinámicos de información imperfecta, la *estrategia pura* de cada jugador i es una función que asigna a cada conjunto de información $h_i \in H_i$ una acción $A(h_i)$; donde $A(h_i)$ es el conjunto de acciones disponibles al jugador i cuando está en el conjunto de información h_i. Así, la *estrategia mixta* de cada jugador es una distribución de probabilidad sobre las estrategias puras de un jugador (Osborne, 2004). Así, en un juego dinámico de información imperfecta, un perfil de estrategias mixtas α^* es un *equilibrio de Nash* si para cada jugador i y cada estrategia mixta α_i; el pago esperado es al menos tan grande como el pago esperado de utilizar cualquier otra estrategia. Esto es para cada jugador i:

$$U_i(\alpha_i^*, \alpha_{-i}^*) \geq U_i(\alpha_i', \alpha_{-i}^*) \text{ para toda estrategia } \alpha_i' \text{ del jugador } i \qquad (7)$$

Las estrategias mixtas permiten a los jugadores aleatorizar en las estrategias a utilizar; sin embargo, en la Sección II.1.1 (ver juegos dinámicos de información perfecta) se muestra que las estrategias mixtas pierden la estructura secuencial del juego ya que en las estrategias mixtas se definen las probabilidades de seleccionar cada acción antes de comenzar el juego y no pueden cambiarse conforme se va desarrollando el juego. En los juegos dinámicos con información imperfecta se adapta este concepto para poder trabajar con conjuntos de información. Una *estrategia de comportamiento* está compuesta por una distribución de probabilidad en las acciones de un conjunto de información A_{h_i} para cada conjunto de información $h_i \in H_i$ para cada jugador i. La diferencia de las estrategias de comportamiento con las estrategias mixtas es que las estrategias de comportamiento permiten retrasar la selección hasta que suceda cierto evento (el jugador se encuentre dentro de un conjunto de información específico). Se dice que una estrategia de comportamiento es una *estrategia pura* cuando se seleccionan las acciones de cada conjunto de información de forma determinística (Nisan *et al.*, 2007).

Los juegos dinámicos de información perfecta utilizan el concepto de subjuego perfecto para adaptar el equilibrio de Nash; así, el equilibrio de Nash perfecto es robusto ante la naturaleza secuencial de los juegos dinámicos. De la misma forma, los juegos dinámicos de información imperfecta extienden el concepto de perfección de subjuegos para que sea robusto en estos juegos. Así, en los juegos dinámicos de información imperfecta aparecen dos conceptos de equilibrio nuevos: el *equilibrio bayesiano perfecto* y el *equilibrio secuencial* (Fudenberg y Tirole, 1991)

El equilibrio bayesiano perfecto extiende el concepto de subjuego perfecto de los juegos dinámicos de información perfecta al hacer que las estrategias utilizadas por los jugadores lleven a un equilibrio bayesiano de Nash. Sin embargo, el equilibrio bayesiano encontrado no debe de ser únicamente para el juego completo; es decir, debe de ser

un equilibrio bayesiano para cada *juego de continuación* que inicia en cada periodo t después de cada historia posible h^t.

Sin embargo, en Kreps y Wilson (1982) extienden el concepto de equilibrio bayesiano perfecto al dar más información a los jugadores al momento de generar sus creencias; este nuevo concepto de equilibrio lo definen como *equilibrio secuencial.* En el equilibrio secuencial suponen que si un conjunto de información contiene más de una historia, entonces el jugador puede crear una creencia acerca de la historia que ha ocurrido. Estas creencias se modelan como una distribución de probabilidad sobre las historias en los conjuntos de información.

II.1.2. Juegos cooperativos

Los juegos cooperativos estudian situaciones en las que se obtienen mejores resultados a través de la formación de grupos de individuos que los resultados obtenidos cuando cada individuo actúa de forma individual; por lo tanto, los juegos cooperativos estudian el comportamiento de grupos de jugadores (Osborne, 2004). Dado que el grupo es la unidad principal de los juegos cooperativos, los individuos dejan de tener acciones individuales y pasan a tener un conjunto de acciones asociadas a un grupo de jugadores. Sin embargo, los individuos siguen teniendo preferencias individuales, la diferencia con los juegos no cooperativos es que en los juegos cooperativos las preferencias de los individuos están asociadas a las acciones que tienen los grupos de individuos y no a las acciones individuales (Shoham y Leyton-Brown, 2009).

En los juegos cooperativos se define una *coalición* como un grupo de jugadores; la expresión S denota a una coalición. Existe una coalición que está compuesta por todos los jugadores del juego, esta coalición se conoce como *gran coalición* y la denota la

expresión N. Así, el resultado de un juego cooperativo es una partición del conjunto de los jugadores en coaliciones, donde se tiene una acción definida para cada coalición de la partición. El modelo más sencillo de los juegos cooperativos permite que los jugadores tomen en cuenta las acciones que se toman en cada coalición que afecta el resultado final; en general, a cada individuo le interesa las acciones que se toman en las coaliciones a las que pertenece. Tomando la definición de Osborne (2004), para definir un juego cooperativo se tienen que especificar al menos:

- Un conjunto de *jugadores.*
- Para cada coalición, un conjunto de *acciones.*
- Para cada jugador, las *preferencias* del conjunto de todas las acciones de todas las coaliciones a las cuales pertenece.

El Juego 8 es un ejemplo de un juego cooperativo en el que ambos jugadores reciben mejores pagos cuando forman una coalición.

Juego 8. *Juego de unanimidad para dos jugadores.* Se tienen dos personas que trabajando juntas pueden producir una unidad de salida, esta unidad la pueden compartir de la forma que quieran. Ninguna de las dos personas puede producir una salida por si misma y cada persona está interesada únicamente en la cantidad que recibe de salida, donde prefiere más a menos.

Se dice que los juegos cooperativos tienen *pagos transferibles* si existe una colección de funciones pago tal que para cada coalición S, cada acción de S genere una distribución de pagos entre los individuos donde la suma de los pagos de cada jugador de la coalición den el mismo monto. El pago total de una coalición en juegos cooperativos con pagos transferibles se denomina como el *valor* o *función característica* de S y se denota por

$v(S)$. Así, los juegos pueden definirse especificando únicamente al conjunto de jugadores y la colección de funciones valor o funciones características (Osborne, 2004). El Juego 8 se puede definir utilizando funciones características de la siguiente forma:

- $N = \{1, 2\}$
- Las funciones características:
  - $v(\{1\}) = v(\{2\}) = 0$
  - $v(\{1, 2\}) = 1$

La representación de juegos cooperativos con base en la función característica muestra que en los juegos cooperativos se busca observar el desempeño que tienen las coaliciones, por lo que no es relevante para el juego la forma en que los jugadores tomen las decisiones dentro de la coalición, cómo se coordinan o cualquier otro detalle que implique profundizar en lo que sucede dentro de la coalición, por lo tanto se puede tomar el pago de una coalición como un valor dado (Shoham y Leyton-Brown, 2009). Así, el objetivo de los juegos cooperativos es responder las preguntas:

- ¿Qué coaliciones se van a formar?
- ¿Cómo se va a repartir el pago entre los jugadores de cada coalición?

Se dice que una acción de una coalición es *estable* si ningún grupo de jugadores puede formar otra coalición y seleccionar una acción en la que todos los jugadores prefieran la nueva coalición a la coalición original. Así, cuando las acciones de la gran coalición son estables, se puede predecir la acción que se espera se seleccione dentro de la gran coalición (Osborne, 2004).

En los juegos cooperativos, el *núcleo* de un juego es el conjunto de acciones a_N de la gran coalición N tal que ninguna otra coalición tenga una acción donde todos sus

miembros la prefieran a a_N (Osborne, 2004); es decir, el conjunto de todas las acciones estables de la gran coalición. El núcleo en los juegos cooperativos es el concepto análogo del equilibrio de Nash para juegos no cooperativos. Dado que el núcleo se define como un conjunto de acciones, todos los juegos siempre van a tener un núcleo; sin embargo, pueden existir juegos donde el núcleo puede ser un conjunto vacío, en estos casos ninguna acción de la gran coalición es inmune a desviaciones (Osborne, 2004).

Los juegos cooperativos, al igual que en los juegos en general, tienen ciertas características que permiten agruparlos en distintas clases de juegos, permitiendo que sea más fácil el modelado y estudio del juego dependiendo de estas características.

Juegos de suma constante

Un juego cooperativo es de suma constante si la suma de los pagos de todas las coaliciones es la misma para cualquier resultado del juego, esto implica que el beneficio de una coalición está dada por la pérdida de otras coaliciones; es decir, un juego es de suma constante si para toda coalición $S \subset N, v(S)+v(N\backslash S) = v(N)$. Un caso particular de estos juegos son los juegos de suma cero, los cuales se obtienen al normalizar el resultado, este tipo de juegos es muy estudiado en los juegos no cooperativos (Shoham y Leyton-Brown, 2009).

Juegos Simples

Los juegos simples son aquellos en los que los pagos de las coaliciones pueden ser 1 o 0 (las coaliciones pueden ganar o perder). En general, se supone que los juegos simples no existe un conjunto vacío; es decir, un juego es simple si para toda coalición $S \in N$, $v(S) \in \{1, 0\}$ (Shoham y Leyton-Brown, 2009). El Juego 8 es un ejemplo de un juego simple en el que las coaliciones reciben un pago con valor cero si los jugadores

permanecen separados y un pago con valor 1 si ambos jugadores están en la misma coalición.

Se dice que los juegos simples son *monotónicos* si una coalición que contiene a una coalición ganadora, entonces la coalición también es ganadora; es decir, sea la coalición S, si $v(S) = 1$, entonces para toda coalición $T \supset S$, se cumple que $v(T) = 1$. Se dice que los juegos simples son *propios* si son juegos simples de suma constante; en estos juegos, si una coalición es ganadora, entonces su complemento es una coalición perdedora; es decir, si S es una coalición ganadora, entonces la coalición $N \setminus S$ es una coalición perdedora (Shoham y Leyton-Brown, 2009). El Juego 9 es un ejemplo de un juego simple monotónico propio.

Juego 9. *Juego de las mayorías para tres personas.* Se modelan tres individuos que desean tener acceso a una unidad de salida. Cualquier mayoría es la que controla el reparto de esta salida. A cada individuo le interesa únicamente la cantidad de salida que obtiene.

Juegos súper aditivos

Los juegos súper aditivos o juegos coalicionales cohesivos son juegos donde las coaliciones pueden trabajar sin interferir entre ellas; es decir, la unión de dos coaliciones tiene un pago garantizado de al menos la suma de los pagos de las coaliciones por separado. Así, Shoham y Leyton-Brown (2009) definen un juego súper aditivo como un juego en el que para todas las coaliciones $S, T \subset N$, si $S \cap T = \emptyset$, entonces $v(S \cup T) \geq v(S) + V(T)$. El Juego 10 es un ejemplo de un juego súper aditivo, donde el pago más grande se tiene cuando se da la gran coalición.

Dado que los juegos súper aditivos tienen la propiedad de que la unión de coaliciones garantizan pagos iguales o mejores que la suma de los pagos de las coaliciones antes de

unirse; entonces la gran coalición garantiza un pago que es al menos tan bueno como el mejor pago formado por cualquier otro conjunto de coaliciones. Por lo tanto, cuando se estudian los juegos súper aditivos, se puede restringir a buscar la solución cuando todos los jugadores forman la gran coalición.

Juego 10. *Terrateniente-trabajador.* La propiedad de un terrateniente produce una salida $f(k+1)m$ donde f es una función creciente, el número total de trabajadores es m. Al terrateniente y los trabajadores les importa la cantidad de salida que reciben y prefieren más a menos.

Juegos convexos

Los juegos convexos son una súper clase de los juegos súper aditivos; en los juegos convexos se quita la restricción que establece que la intersección de los conjuntos debe de ser vacía. Así, se permite que se den coaliciones donde existen jugadores en ambas coaliciones y por lo tanto se toma en cuenta el pago que producirían la coalición de los jugadores que están en ambas coaliciones (lo que se pierde si esos jugadores abandonan la coalición) al momento de garantizar el pago. Shoham y Leyton-Brown (2009) definen que un juego es convexo si para todas las coaliciones $S, T \subset N$, si $S \cap T \neq \emptyset$, entonces $v(S \cup T) \geq v(S) + v(T) - v(S \cap T)$. El Juego 11 es un ejemplo de un juego convexo.

Juego 11. *Juego del aeropuerto.* Se tiene un conjunto de ciudades que necesita un aeropuerto. Si abren un aeropuerto regional, las ciudades reparten el costo, el costo del aeropuerto regional depende del tamaño de la pista necesario para que pueda aterrizar el avión más grande. En otro caso, cada ciudad tendría que construir su propio aeropuerto.

Capítulo III

Juego del PAG

En este capítulo se define el modelo base del sistema y la terminología con la que se trabaja a lo largo de esta tesis; después se hace una clasificación del tipo de juego al que pertenece el PAG con base en sus características. Esta clasificación se basa en los conceptos descritos en el Capítulo II de esta tesis. Adicionalmente, se proponen algunas variantes del PAG que pueden lograr condiciones de equilibrio con un mayor bienestar social.

III.1. Modelo del sistema

El problema de asignación de guardaespaldas (PAG) se modela como un problema de grafos. Sea $G = (V_G, E_G)$ un grafo conectado no dirigido, una solución del PAG se modela como un árbol de esparcimiento $T = (V_T, E_T)$ enraizado en un nodo especial r, donde $V_T = V_G$ y $E_T \subseteq E_G$. Sea T_u un sub-árbol de T enraizado en el nodo u, la expresión $T_u - T_v$ denota el sub-árbol generado al remover de T_u el subárbol T_v, donde T_v es un sub-grafo de T_u. Cada nodo u registra en sus variables α_u y β_u, el número de nodos blancos y negros que existen en T_u, respectivamente. Se dice que el sub-árbol T_u es *blanco*$^+$ cuando $\alpha_u > \beta_u$; *negro*$^+$ cuando $\alpha_u < \beta_u$ y *gris* cuando $\alpha_u = \beta_u$. El conjunto de nodos vecinos del nodo u en el grafo G se denota como N_u^G. Se dice que un nodo v es *descendiente* de un nodo u, si el nodo u se encuentra en la trayectoria única de v a r en T (para el mismo caso, se dice que u es un *ancestro* de v). Para facilitar la notación, se supone que u es descendiente de sí mismo, pero, no un ancestro.

El conjunto de nodos descendientes del nodo u en T se denota como D_u^T. Otro conjunto de nodos importante es el conjunto de nodos no descendientes de u en el grafo G, el cual se denota como A_u^G, donde $A_u^G = \{v | v \in N_u^G \wedge v \notin D_u^T\}$.

Se define el beneficio social del sistema como la suma de las ganancias individuales de todos los nodos excepto la raíz. El beneficio social de un árbol de esparcimiento T se denota como $g(T)$. La expresión $g(T_t)$ representa el beneficio social para una configuración T en un instante t específico.

La ganancia individual de un nodo está relacionada con su distancia con respecto a la raíz. La distancia del nodo u a la raíz se denota como d_u^T y su ganancia individual por las funciones f_1 y f_2 cuando u es blanco y negro, respectivamente. Las Ecuaciones 8 y 9 definen estas funciones.

$$f_1(u) = n - d_u^T \tag{8}$$

$$f_2(u) = d_u^T \tag{9}$$

El juego del PAG se estudia bajo dos enfoques distintos. En el enfoque cooperativo (no cooperativo), las *preferencias* de un nodo determinan cuál de las acciones, acercarse o alejarse de la raíz reditúan en un mayor beneficio social (utilidad individual). Para el enfoque cooperativo, las preferencias de un nodo se establecen con base en los votos emitidos por sus descendientes (y que definen al subárbol del nodo como *blanco*$^+$, *negro*$^+$ o *gris*), por lo que un nodo bajo el enfoque cooperativo busca maximizar la ganancia de su subárbol. En el enfoque no cooperativo, las preferencias de cada nodo se establecen únicamente con base en su color, por lo que cada nodo busca maximizar su utilidad individual. Para ambos enfoques, la preferencia de cualquier nodo u se calcula a través de la función $\pi_u : M_u \rightarrow \mathbb{Z}$ expresada en la Ecuación 10, donde M_u representa

el movimiento del nodo u.

$$\pi_u(v) = (\alpha_u - \beta_u)(d^T_{P_u} - d^T_v),\ \forall v \in A^G_u \tag{10}$$

En el enfoque no cooperativo, cada nodo u evita la recolección de votos de sus descendientes, por lo que para este enfoque $\alpha_u = 1$ y $\beta_u = 0$ si u es blanco y $\alpha_u = 0$ y $\beta_u = 1$ si u es negro.

Se dice que el nodo u realiza un *movimiento* cuando u cambia su apuntador de $P_u = v$ a $P_u = v'$. Note que v' debe de ser un nodo de A^G_u, ya que de otra forma se desconecta del árbol T. Al cambio de distancia obtenido al realizar un movimiento se denota como $\Delta_{d^T_u} = d^T_{v'} - d^T_v$. Un *movimiento de optimización* es un movimiento en el que un nodo u incrementa, al menos en uno el valor de π_u; el nodo con el que se va a establecer la conexión se selecciona de acuerdo a la estrategia de u. Se dice que u está en un estado *estable* cuando no existe un nodo $v \in A^G_u$ al cual se pueda conectar u e incremente al menos en uno el valor de π_u. Cuando todos los nodos del sistema se encuentran en un estado estable, se dice que el árbol T se encuentra en *equilibrio.* Una restricción del PAG es que el árbol de esparcimiento solución T tiene que estar en equilibrio.

III.2. Clasificación del juego del PAG

En el juego del PAG, cada jugador u busca maximizar sus pagos de acuerdo a una función de pagos π_u. Esta función determina cuándo cada jugador actúa para obtener un beneficio individual o social. Sin embargo, aún bajo el enfoque cooperativo, los nodos maximizan sus pagos de forma individual; ya que la función de pagos π_u es la responsable de que el nodo coopere y seleccione la opción de conexión que incremente el

beneficio social. Dado que cada jugador actúa únicamente con base en dicha función, sin considerar si ésta afecta o no a otros nodos en el sistema, el juego del PAG se clasifica como un *juego no cooperativo.*

Una característica del PAG es que las opciones de conexión de los nodos cambian con la estructura de árbol en el sistema. El conjunto de vecinos no descendientes A_u^G, representa las opciones de conexión de u con las cuales puede maximizar π_u sin generar ciclos. Sin embargo, A_u^G depende de los movimientos que realicen los nodos vecinos de u; por lo que en cada movimiento de u, A_u^G debe estar siempre actualizado. Para cumplir con este requisito de actualización en el juego del PAG, los jugadores mueven por turnos. Note que el movimiento de un nodo no solo afecta en las decisiones de sus vecinos, sino que también puede afectar indirectamente a otros nodos. Por lo tanto, el juego no puede ser un juego estático ya que al tomar la decisión, los jugadores observan los cambios en sus nodos vecinos antes de evaluar. Dado que los jugadores mueven por turnos y la información se actualiza en cada turno, el juego del PAG se considera un *juego dinámico.*

Otra característica del juego del PAG, es que los jugadores toman decisiones con base en información local; por lo que desconocen cualquier información sobre las acciones mas allá de su vecindario. Cuando cada nodo u evalúa π_u, u sólo requiere conocer la distancia de los nodos vecinos a la raíz y del conjunto de vecinos no descendientes A_u^G; la información acerca del color de los nodos, los nodos en la trayectoria a la raíz y los movimientos anteriores de otros nodos no es necesaria. Bajo estas condiciones, u no puede determinar si el cambio en la distancia de un nodo vecino se debe al movimiento del nodo vecino o al de un ancestro (cuando se utiliza el enfoque cooperativo, cada nodo además requiere conocer la cantidad de nodos negros y blancos en su sub-árbol, sin saber qué color corresponde a sus vecinos.). Dado que los nodos toman sus decisiones con

base en información local, el juego del PAG se considera como un *juego de información incompleta.*

Con base en lo anterior, se concluye que el PAG pertenece a los tipos de juegos *no cooperativos, dinámicos y con información incompleta.*

III.3. Variantes del juego del PAG

En esta sección se proponen algunas variantes del juego del PAG que pueden lograr condiciones de equilibrio con mayor beneficio social que el juego del PAG propuesto por Fajardo-Delgado *et al.* (2010). Dado que el PAG es un problema ideado para sistemas distribuidos, los nodos solo toman decisiones con base en información local. Por lo tanto, las variantes propuestas no cambian el tipo de juego ni la información que conoce. Una variante interesante para el juego del PAG es cuando se modela como un juego cooperativo donde los nodos se juntan en coaliciones que buscan acercarse o alejarse de la raíz para alcanzar un beneficio en común en el grupo y repartir el pago en cada coalición. Sin embargo, los juegos cooperativos y los juegos de información completa no se exploran aquí, ya que éstos involucran el uso de información a nivel global.

Debido a las características del juego del PAG, sólo se pueden modificar algunas reglas. Las reglas que pueden cambiar son las estrategias de los jugadores, la función de pagos y la función jugador.

III.3.1. Estrategias

En el juego original del PAG, el comportamiento de cada jugador u se define por una estrategia del conjunto de estrategias S_u; donde S_u = {conectarse lo más cerca posible de r, conectarse lo más lejos posible de r}. Así, u utiliza la función de mejor

respuesta para seleccionar su estrategia en cada turno. Una característica deseable de este conjunto de estrategias es que los jugadores puedan llegar a un equilibrio utilizando la función de mejor respuesta. Sin embargo, esta estrategia limita la exploración ya que siempre se va a seleccionar la conexión que minimice/maximice la distancia a r (u siempre maximiza π_u); de esta forma, un nodo puede dejar sin explorar una conexión que posiblemente pudiera dar mejores pagos al llegar a un equilibrio.

Las variantes propuestas en este trabajo consisten en relajar las estrategias S_u para cada jugador u, tal que $S_u = \{\text{disminuir la distancia a } r, \text{ aumentar la distancia a } r\}$. Note que se omite la maximización/minimización de la distancia a r. Permitiendo a u utilizar una función de mejor respuesta débil (incrementar el valor de π_u sin obligar a maximizarlo). Así, se mantiene la característica de poder llegar a un equilibrio utilizando una función de mejor respuesta, pero además se permite que los jugadores tengan más alternativas de conexión al realizar un movimiento. De esta forma, los jugadores pueden explorar otras opciones antes de establecer la conexión que minimice/maximice su distancia a la raíz. Bajo estas nuevas estrategias, los jugadores tienen un mecanismo de decisión, el cual, establece la forma en que los jugadores van a ir disminuyendo/aumentando la distancia a r. En este trabajo se proponen tres estrategias distintas: ε-voraz, anti-voraz e IA-voraz; adicionalmente se adapta la estrategia propuesta por Fajardo-Delgado *et al.* (2010) para trabajar con la estrategia propuesta, esta adaptación se denomina estrategia voraz. En las cuatro estrategias, cada jugador u selecciona una estrategia del conjunto de estrategias S_u propuesto. Sin embargo, en cada una de estas estrategias cambia el mecanismo de decisión.

Estrategia Voraz

La estrategia voraz consiste en minimizar/maximizar la distancia del nodo a r; es decir, cada nodo u selecciona al nodo v tal que $\pi_u(v) = max(\pi_u(v'))$, $\forall v' \in A_u^G$ y $\pi_u(v) > \pi_u(P_u)$. Si dos o más nodos son elegibles por u, entonces u selecciona cualquiera de ellos con la misma probabilidad. Cuando ningún nodo vecino de u cumple con estas condiciones, se dice que el nodo está estable. Note que esta estrategia funciona de la misma forma en que funciona la estrategia propuesta por Fajardo-Delgado *et al.* (2010).

Estrategia ε-voraz

La estrategia ε-voraz basa su mecanismo de decisión en la regla de decisión ε-voraz definida por Chapman (2009). En la regla de decisión original, se selecciona la mejor opción con una probabilidad de $1-\varepsilon$ y se reparte el resto de la probabilidad en las otras opciones. La diferencia del mecanismo de decisión propuesto con la regla de decisión de Chapman (2009) es que únicamente se relaja el criterio de selección de la estrategia voraz, dando oportunidad de seleccionar a aquellos nodos que incrementan π_u pero no necesariamente la maximizan; es decir, no se toman en cuenta las opciones que decrementan el valor del pago actual. En esta estrategia, se forman dos conjuntos con los nodos elegibles. Para cada nodo u, el conjunto Z_u contiene a todos los nodos que maximizan π_u; es decir, $Z_u = \{v \in A_u^G | \pi_u(v) > \pi_u(P_u)$ y $\pi_u(v) \geq \pi_u(v'), \forall v' \in A_u^G\}$; el conjunto Y_u contiene a todos los nodos que incrementan π_u pero no la maximizan, es decir, $Y_u = \{v \in A_u^G | \pi_u(v) > \pi_u(P_u)$ y $v \notin Z_u\}$. Así, la estrategia asigna la probabilidad de seleccionar a un nodo $v \in A_u^G$ de acuerdo a los siguientes criterios:

- Si $v \in Z_u$, entonces la probabilidad de seleccionar al nodo v es $\frac{\varepsilon}{|Z_u|}$.
- Si $v \in Y_u$, entonces la probabilidad de seleccionar al nodo v es $\frac{1-\varepsilon}{|Y_u|}$.

- Para cualquier otro caso, la probabilidad de seleccionar al nodo v es 0.

donde ε es una constante con valores $0 < \varepsilon \leq 1$. Cuando $Z_u = Y_u = \emptyset$ entonces se dice que el nodo está estable.

En esta estrategia el valor de ε sirve para determinar la preferencia por seleccionar a los nodos del conjunto Z_u o Y_u.

- Si $\varepsilon < 0.5$ entonces los nodos del conjunto Y_u tienen más probabilidad de ser seleccionados.
- Si $\varepsilon = 0.5$ entonces los nodos de ambos conjuntos tienen la misma probabilidad de ser seleccionados.
- Si $\varepsilon > 0.5$ entonces los nodos del conjunto Z_u tienen más probabilidad de ser seleccionados; cuando $\varepsilon = 1$ esta estrategia trabaja de la misma forma que la estrategia voraz.

Note que los nodos dentro de los conjuntos Z_u y Y_u de cada nodo u, siempre incrementan π_u; de esta forma, cada movimiento disminuye la cantidad de nodos en dichos conjuntos. Así, aunque los nodos se seleccionan mediante probabilidad, u siempre llega a un equilibrio.

Estrategia Anti-voraz

La estrategia anti-voraz consiste en obligar al nodo a elegir la mayor cantidad de opciones conforme incrementa π_u. En esta estrategia, cada nodo u elige al nodo vecino v que incrementa al mínimo el cambio en π_u, donde $v \in A_u^G \mid \pi_u(v) = min(\pi_u(v'))$, $\forall v' \in A_u^G \mid \pi_u(v') > \pi_u(P_u)$. Si dos o más nodos son elegibles por u, entonces u selecciona cualquiera de ellos con la misma probabilidad. Cuando ningún nodo vecino de u cumple con estas condiciones, se dice que el nodo está estable.

Estrategia IA-Voraz

En el juego del PAG, los equilibrios pueden ser distintos dependiendo de los movimientos realizados por los jugadores a lo largo del juego. Sin embargo, el número de equilibrios posibles en un grafo se determina principalmente por el árbol de esparcimiento inicial a optimizar; es decir, se pueden generar árboles de esparcimiento iniciales que permitan pocos movimientos tal que los equilibrios resultantes tengan un beneficio social bajo. Chapman (2009) define una regla de decisión que selecciona una opción de forma aleatoria, donde las mejores opciones tienen mayor probabilidad de ser seleccionadas; así se puede seguir una estrategia aleatoria tal que el resultado no dependa tanto de la solución inicial. Sin embargo, Debido a que cada nodo asigna una probabilidad distinta de cero a sus vecinos en A_u^G, los nodos pueden mover indefinidamente y nunca llegar a un estado estable y mantenerse en dicho estado. La estrategia IA-Voraz (Inicio Aleatorio-Voraz) adapta la regla de probabilidad lineal de Chapman (2009) y permite a los nodos seguir una estrategia aleatoria durante un número finito de movimientos para asegurar la convergencia a un estado estable. En esta estrategia, cada nodo u utiliza un modelo de probabilidad lineal para asignar una mayor probabilidad de establecer su conexión con los nodos vecinos $v \in A_u^G$ que mejoran el pago de u; pero también existe una probabilidad de elegir a un nodo v que no mejore (e incluso empeore) el pago de u. Así, la estrategia IA-Voraz permite que el nodo decida aleatoriamente con base en el modelo de probabilidad lineal durante un número finito de movimientos; de forma que conforme incrementa el número de movimientos de un nodo, éste comienza a seguir una estrategia voraz.

En la estrategia IA-Voraz, cada nodo u forma dos conjuntos con los nodos elegibles: el conjunto Z_u y Y_u. El conjunto Z_u contiene a todos los nodos vecinos no descen-

dientes de u que maximizan y mejoran π_u; es decir, $Z_u = \{v \in A_u^G \mid \pi_u(v) > \pi_u(P_u)$ y $\pi_u(v) \geq \pi_u(v'), \forall v' \in A_u^G\}$. El conjunto Y_u contiene a todos los nodos vecinos no descendientes de u que no estén en Z_u; es decir, $Y_u = \{v \in A_u^G \mid v \notin Z_u\}$. Note que el conjunto Y_u incluye a los nodos vecinos de u que empeoran π_u. La Ecuación 11 define la probabilidad $\Pr_u(v)$ de que el nodo u seleccione a un nodo $v \in Z_u$.

$$\Pr_u(v) = \frac{1 - \sum_{\forall v' \in Y_u} Pr_u(v')}{|Z_u|} \tag{11}$$

Observe que en la Ecuación 11, la probabilidad de que u seleccione a v depende de la probabilidad de que u no seleccione a un nodo de Y_u. La probabilidad de seleccionar a un nodo de Y_u disminuye conforme se incrementa el número de movimientos del nodo u, tal que $\sum_{\forall v' \in Y_u} \Pr_u(v')$ tiende a cero. La Ecuación 12 define la probabilidad $\Pr_u(v)$ de que el nodo u seleccione a un nodo $v \in Y_u$.

$$\Pr_u(v) = \left(1 - \frac{min(t, movMax)}{movMax}\right) * \frac{norm(\pi_u(v))}{\sum_{\forall v' \in A_u^G} norm(\pi_u(v'))} \tag{12}$$

El primer factor de la Ecuación 12 representa el nivel de voracidad del nodo; es decir, conforme el nodo mueve, el nodo disminuye la probabilidad de seleccionar a un nodo del conjunto Y_u y adapta su estrategia para hacerla una estrategia voraz. El término $movMax$ es un parámetro del juego en el que se define el número máximo de movimientos en el que el nodo u puede elegir a un nodo de Y_u (Sea Δ_u el grado de conectividad de u, se puede suponer que $movMax = \Delta_u$). El término t representa el número de movimientos realizados por el nodo u. Note que el primer factor de la expresión de la Ecuación 12 disminuye conforme se incrementa el valor de t. El segundo factor del producto en la Ecuación 12 representa la probabilidad lineal que se le da al nodo de ser seleccionado. El término $norm(\pi_u(v))$ representa el valor normalizado del pago que entrega el nodo v a u. Dado que $\pi_u(v)$ puede tener pagos negativos; el lado derecho de la Ecuación 12

puede ser negativo; la normalización toma el pago más pequeño que puede obtener al establecer la conexión con un nodo $v \in Y_u$; es decir, $min_{v \in Y_u}(\pi_u(v))$ y le da el valor de 1; la normalización suma a los pagos de los nodos de Y_u la diferencia necesaria para lograr que el pago mínimo normalizado sea 1. Suponga que un nodo u tiene tres nodos vecinos en el conjunto Y_u con los siguientes pagos $\{-3, 0, 3\}$; entonces el valor normalizado para estos pagos es $\{1, 4, 7\}$. Así, la probabilidad de que u seleccione a un nodo $v \in Y_u$ siempre es positiva y se respeta el valor de los pagos originales.

Note que cuando $t \geq movMax$, la probabilidad de que el nodo u seleccione a un nodo $v \in Y_u$ es cero; de aquí en adelante, la probabilidad asignada a los nodos de Z_u es uniforme y cuando $Z_u = \emptyset$ y $t \geq movMax$, se dice que el nodo está estable y no realiza ningún movimiento.

III.3.2. Función de pagos

En el juego del PAG, cada jugador u cuenta con una función de pagos π_u que representa las preferencias de u para establecer su conexión con sus vecinos en A_u^G. Bajo el enfoque cooperativo, π_u mide el incremento en beneficio social en el sub-árbol T_u; mientras que en el enfoque no cooperativo, π_u mide el incremento de la utilidad individual de u. La función π_u define las preferencias de u para un momento específico; es decir, una buena opción de conexión en un turno puede dejar de serlo al siguiente turno. A continuación se presenta un ejemplo en el que se aprecia cómo un movimiento puede resultar en una mala decisión en el futuro.

Ejemplo 2. La Figura 5a muestra una configuración del sistema en el instante t del juego del PAG. En el instante $t+1$, el nodo w se aleja de la raíz debido al movimiento de un ancestro (ver Figura 5b). En el instante $t+2$, el nodo u evalúa sus opciones y

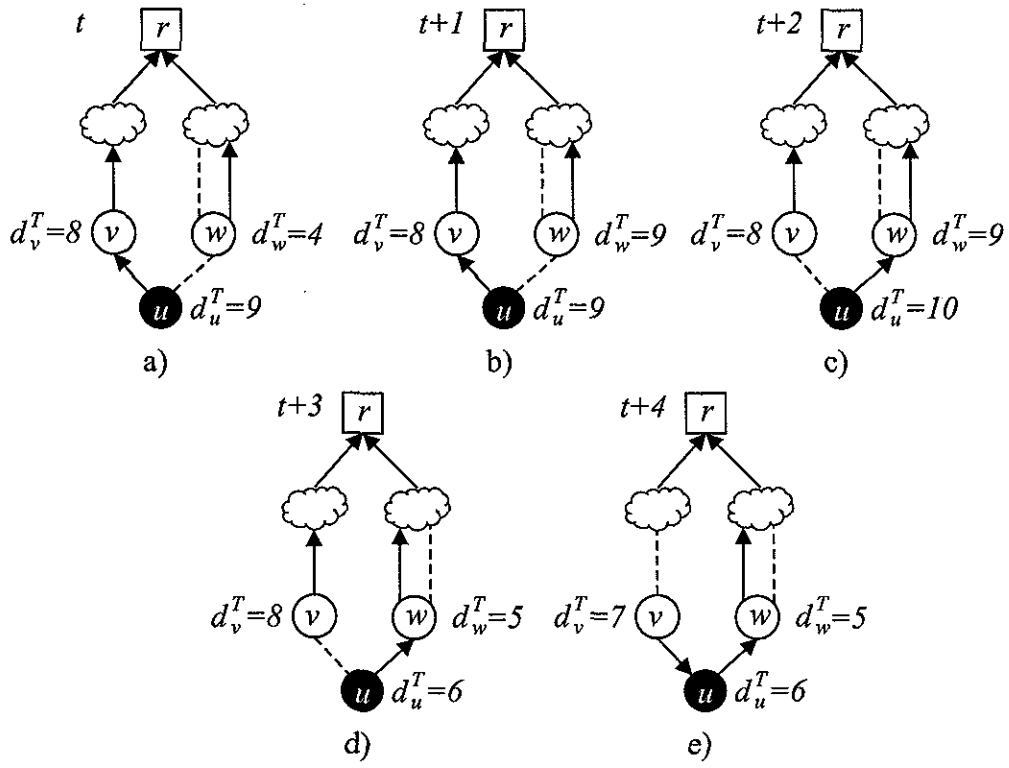

Figura 5. Ejemplo de un movimiento que es una mala decisión a futuro; a) configuración del sistema en el instante t; b) un ancestro de w se aleja de r; c) u cambia su apuntador hacia w; d) w cambia de conexión para acercarse a r; e) v cambia su apuntador hacia u.

cambia de conexión con el nodo w para alejarse de r; logrando un incremento de 1 en su utilidad individual (ver Figura 5c). En el instante $t+3$, el nodo w evalúa sus opciones y cambia su conexión con un vecino que disminuye su distancia a r; en este movimiento la utilidad de u decrementa en 4 unidades (ver Figura 5d). En el instante $t+4$, el nodo v evalúa sus opciones y cambia su conexión hacia el nodo u ya que éste disminuye su distancia a r (ver Figura 5e). Note que cuando el nodo v mueve, v deja de pertenecer al conjunto A_u^G y por lo tanto u no puede revertir su conexión para recuperar el valor de la utilidad individual que tiene en el instante t.

En este trabajo se propone una nueva función de pagos π' basada en la evaluación de estrategias utilizando el pago esperado sobre frecuencias históricas definida por

Chapman (2009), la cual hace que los nodos esperen cierta cantidad de turnos antes de cambiar su conexión. Esta nueva función permite al nodo evaluar el comportamiento de sus vecinos en el tiempo, a través de los pagos promedio de la función π. Así, cuando un nodo comienza a ser una buena (mala) opción, π' incrementa (decrementa) gradualmente el valor que entrega π para asegurar que realmente es una buena (mala) opción para el futuro. La función π' para el turno t se define en la Ecuación 13.

$$\pi_u^{'}(v)^t = \frac{\pi_u(v) + (t-1) * \pi_u^{'}(v)^{t-1}}{t} \tag{13}$$

La Ecuación 13 representa el valor promedio de la función $\pi_u(v)$ para el turno t de cada jugador u, donde $(t-1) * \pi_u^{'}(v)^{t-1}$ representa la suma de cada valor $\pi_u(v)$ en turnos anteriores; es decir, $(t-1) * \pi_u^{'}(v)^{t-1} = \sum_{i=1}^{t-1} \pi_u(v)^i$.

Una característica de la función π' es que puede adaptarse a cualquier función π; es decir, puede trabajar con los dos enfoques cooperativo y no cooperativo. Sin embargo, π' puede generar equilibrios distintos a los de π; esto se debe a que el incremento/decremento en el valor de π' es gradual conforme al nuevo valor en π. Note que si $t \rightarrow \infty$, entonces $\pi' = \pi$; así, π' y π tienen los mismos equilibrios.

III.3.3. Función jugador

En el juego del PAG, la función jugador es la que se encarga de dar el turno a un nodo para que realice un movimiento (en el algoritmo CBAP, el calendarizador representa a la función jugador). Fajardo-Delgado *et al.* (2010) proponen dos funciones jugador distintas:

- La función jugador *uniforme* selecciona con la misma probabilidad a cualquier nodo que no se encuentre estable.

- La función jugador *voraz* selecciona al nodo que reporta que puede obtener el mayor incremento en sus pagos al mover.

Aquí se propone una nueva función jugador; esta función está diseñada para trabajar en conjunto con la estrategia anti-voraz propuesta en este capítulo (ver Sección III.3.1) cuyo fin es maximizar la cantidad de veces que mueven los jugadores antes de llegar a un equilibrio. Esta nueva función se denomina *anti-voraz* y consiste en dar el turno al jugador que puede obtener el menor incremento en sus pagos al mover; es decir, selecciona al jugador $u \mid \pi_u(x) = min(\pi_{u'}(x'))$, $\forall u' \in Z$, $\forall x' \in A^G_{u'}$ y $\pi_{u'}(P_{u'}) < \pi_{u'}(x')$. Donde Z representa al conjunto de jugadores que no se encuentran en equilibrio. Si dos o más jugadores cumplen con esta condición, se selecciona cualquiera de estos jugadores con la misma probabilidad. Si los jugadores siguen una estrategia anti-voraz junto con la función jugador anti-voraz, entonces se incrementa el número de movimientos que realizan los jugadores antes de llegar a un equilibrio, ya que se le da oportunidad de mover al jugador que tiene la mejora mínima.

Capítulo IV

Técnica de especulación

Una característica del algoritmo CBAP propuesto por Fajardo-Delgado *et al.* (2010), es que siempre termina su ejecución en una configuración de equilibrio. Un *equilibrio* es una condición tal que ningún nodo puede mejorar su ganancia individual de manera unilateral. Sin embargo, los equilibrios pueden ser distintos, dependiendo del enfoque que se utilice. El algoritmo CBAP trabaja bajo dos tipos de enfoques: el cooperativo y el no cooperativo. En el enfoque cooperativo, cada nodo actúa únicamente en beneficio de la mayoría de sus decendientes; mientras que en el no cooperativo, los nodos buscan incrementar únicamente su ganancia individual. Bajo el enfoque cooperativo, los nodos requieren calcular el número de descendientes blancos y negros para así tomar decisiones (bajo el enfoque no-cooperativo, esta información no es necesaria). Independientemente del enfoque que se utilice, el algoritmo CBAP puede quedar atrapado en un máximo local. Un *máximo local* representa una condición de equilibrio con un valor de beneficio social distinto al óptimo. Para evitar algunos máximos locales, se propone una técnica que permita al algoritmo escapar de ciertos máximos locales y mejorar el beneficio social. Dicha técnica se analiza únicamente bajo el enfoque cooperativo (se conjetura que el análisis del enfoque no cooperativo es mucho más complejo), en donde se requiere que cada nodo utilice, además de la información de sus descendientes, la información de sus ancestros.

A continuación se define la terminología que se requiere para explicar la técnica propuesta.

IV.1. Modelo del sistema

En este capítulo se supone un modelo del sistema extendido del modelo presentado en el Capítulo III (ver Sección III.1) para el enfoque cooperativo. En este nuevo modelo, los nodos tienen un nuevo tipo de movimiento denominado movimiento de especulación. Un *movimiento de especulación* es un movimiento en el que un nodo "sacrifica" su ganancia individual para que otro nodo pueda mejorar el beneficio social. El decremento en la ganancia individual del nodo que realiza el movimiento de especulación se denomina *pérdida*. La ganancia obtenida después de recuperar la pérdida se denomina *dividendo*. En la técnica propuesta, el incremento en el beneficio social debe ser mayor que el decremento por el movimiento de especulación.

IV.2. Técnica de especulación

La intuición detrás de la técnica de especulación es la siguiente: se permite a un nodo "perturbar" la condición de equilibrio del sistema obtenida por una ejecución previa del algoritmo CBAP, para permitir que otros nodos realicen movimientos alternativos de optimización. La perturbación del equilibrio se realiza a través de un movimiento de especulación en los que se garantice que en movimientos posteriores se recupere esta "pérdida" y aumente el beneficio social. La técnica de especulación permite obtener soluciones mejores a las proporcionadas por el algoritmo CBAP y con un tiempo de ejecución del mismo orden. A continuación se presenta un ejemplo de esta técnica:

Ejemplo 3. La Figura 6a muestra una condición de equilibrio obtenida en el instante t de la ejecución del algoritmo CBAP y cuya ganancia es $g(T_t) = 11$. En el instante $t+1$, el nodo w perturba el equilibrio al cambiar su apuntador del nodo b al nodo a

(ver Figura 6b). Note que el nodo w cambia su distancia de $d_w^T = 2$ a $d_w^T = 3$, por lo que $\Delta_{d_w^T} = -1$ y el valor de la pérdida es 1. Cuando w realiza el movimiento de especulación, el beneficio social disminuye de $g(T_t) = 11$ a $g(T_{t+1}) = 10$. Posteriormente, en el instante $t+2$ el nodo b realiza un movimiento de optimización. Así, b cambia su apuntador del nodo r al nodo w (ver Figura 6c), y b cambia su distancia de $d_b^T = 1$ a $d_b^T = 4$, por lo que $\Delta_{d_b^T} = 3$. Note que el movimiento de optimización del nodo b recupera la pérdida del nodo w y aumenta el beneficio social en dos unidades. Así, después de aplicar la técnica de especulación, el beneficio social aumenta de $g(T_t) = 11$ a $g(T_{t+2}) = 13$.

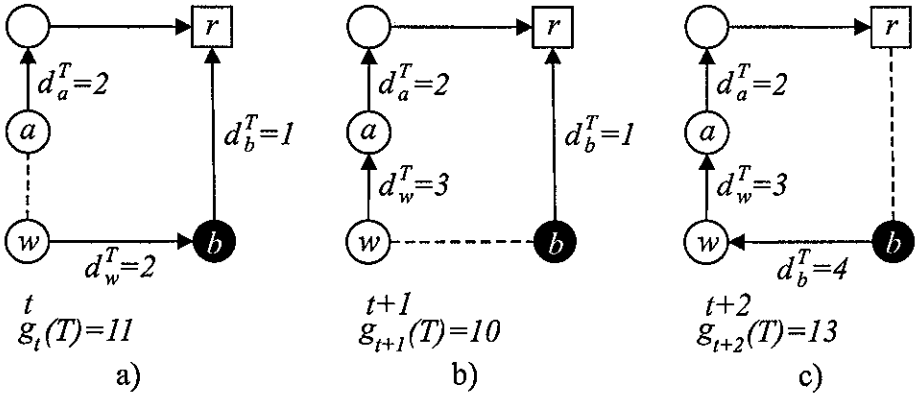

Figura 6. Técnica de especulación; a) condición de equilibrio encontrada por el algoritmo CBAP; b) movimiento de especulación del nodo w; c) movimiento de optimización del nodo b que recupera la pérdida y genera dividendos.

La Figura 7 muestra, de forma más general, una posible configuración de equilibrio en la que el algoritmo detectaría al nodo w como un posible candidato para aplicar la técnica de especulación; donde los nodos b y w tienen descendientes.

Cuando un nodo realiza un movimiento de especulación no siempre genera dividendos, incluso puede decrementar el beneficio social original. Un ejemplo de esto es cuando el nodo que va a realizar el movimiento de especulación y su padre, tienen ambos un sub-árbol $blanco^+$. Para determinar si es factible que un nodo u realice un movimiento de especulación que genere dividendos, u debe conocer además de α_u y β_u, los valores

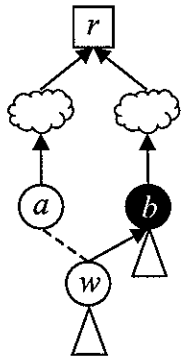

Figura 7. El algoritmo detecta que el nodo w puede ser un candidato para sacrificar parte de su ganancia individual a cambio de un incremento a futuro en el beneficio social.

de α_{P_u} y β_{P_u}. Así, al realizar el cálculo, u considera el número de nodos blancos y negros que existen tanto en T_u como en $T_{P_u} - T_u$ (el sub-árbol del padre de u sin tomar en cuenta los nodos del sub-árbol de u). Adicionalmente, debe considerar su distancia a la raíz y la distancia obtenida al cambiar de padre. En el Teorema 1 se establece la condición para la cual un movimiento de especulación siempre genera dividendos.

Teorema 1. *Dada una configuración de equilibrio en el sistema, cualquier nodo w que realice un movimiento de especulación permite que otro nodo b genere dividendos en el sistema, siempre y cuando se cumpla la siguiente condición:*

$$(\beta_w - \alpha_w)\Delta_{d_w^T} + ((\beta_b - \beta_w) - (\alpha_b - \alpha_w))\Delta_{d_b^T} > 0 \tag{14}$$

Demostración. Se demuestra por contradicción. Suponga que la Condición 14 se cumple y que cuando el nodo w realiza un movimiento de especulación, el nodo b no genera dividendos. La expresión $(\beta_w - \alpha_w)\Delta_{d_w^T}$ representa la pérdida en la ganancia individual del nodo w al realizar el movimiento de especulación y la expresión $((\beta_b - \beta_w) - (\alpha_b - \alpha_w))\Delta_{d_b^T}$ representa la ganancia del nodo b cuando éste realiza un movimiento de optimización, posterior al movimiento de especulación del nodo w, y se conecta a w. Debido a que el sistema se encuentra en equilibrio, ningún mo-

vimiento de w genera una ganancia individual mayor a la iteración actual, es decir, $(\beta_w - \alpha_w)\Delta_{d_w^T} \leq 0$. Dado que el nodo b no genera dividendos, se cumple que: $|(\beta_w - \alpha_w)\Delta_{d_w^T}| \geq |((\beta_b - \beta_w) - (\alpha_b - \alpha_w))\Delta_{d_b^T}|$.
Por lo tanto, $(\beta_w - \alpha_w)\Delta_{d_w^T} + ((\beta_b - \beta_w) - (\alpha_b - \alpha_w))\Delta_{d_b^T} \leq 0$ y existe una contradicción. □

En la técnica de especulación, cualquier nodo w verifica que se cumpla la Condición 14 antes de realizar un movimiento de especulación. El nodo w supone que el nodo b va a cambiar su apuntador hacia él cuando b realice el movimiento de optimización; sin embargo, b puede conectarse a cualquier nodo $u \in T_w$. Note que b podría obtener una ganancia mayor.

Cuando el nodo w realiza el movimiento de especulación, los nodos dentro de T_w y T_b pueden realizar movimientos de optimización que también incrementen el beneficio social. Por lo tanto, los dividendos estimados por los nodos que van a realizar movimientos de especulación representan una cota mínima de lo que puede obtenerse al encontrar un nuevo equilibrio.

A continuación se describe una versión del algoritmo CBAP junto con la técnica de especulación.

IV.3. Algoritmo CBAP$^+$

En esta sección se presenta una versión modificada del algoritmo CBAP, denominada CBAP$^+$. Esta nueva versión consiste en la implementación de la técnica de especulación. La entrada para el algoritmo CBAP$^+$ es un grafo conectado no dirigido $G = (V_G, E_G)$, donde $|V_G| = n$ y $|E_G| = m$, y una partición del conjunto de nodos $V = V_1 \cup V_2 \cup V_3$. La salida es un árbol de esparcimiento $T = (V_T, E_T)$ que satisface una condición de

equilibrio para el PAG, la cuál no siempre maximiza el beneficio social.

Pseudo-código 1 Algoritmo CBAP

1: $T \leftarrow$CREAR-ARBOL(G, r)
2: $T \leftarrow$ETIQUETADO-INICIAL(T)
3: $T \leftarrow$CONTEO-VOTOS(T)
4: **mientras** $\exists u$ tal que $u \leftarrow$CALENDARIZADOR(G, T) **hacer**
5: $P_u^{viejo} \leftarrow P_u$
6: $T' \leftarrow$MOVIMIENTO-OPTIMIZACION(G, T, u)
7: $T \leftarrow$ACTUALIZACION(T', u, P_u^{viejo})
8: **fin_mientras**

El algoritmo CBAP propuesto por Fajardo-Delgado *et al.* (2010) se muestra en el Pseudo-código 1. Los Pasos 1-3 construyen un árbol inicial T en el grafo de entrada e inicializan las variables de cada nodo del grafo. Los Pasos 4-8 realizan la optimización iterativa del árbol inicial hasta que el árbol se encuentra en equilibrio. El algoritmo CBAP$^+$ se muestra en el Pseudo-código 2. Los Pasos 1-4 sirven para inicializar de la misma forma que en el algoritmo CBAP. Los pasos 5-25 realizan la optimización iterativa del árbol inicial hasta que el árbol se encuentra en equilibrio y ningún nodo puede realizar un movimiento de especulación donde se generen dividendos. A continuación se explican cada uno de los procedimientos que se utilizan en el Pseudo-código 2 del algoritmo CBAP$^+$.

IV.3.1. Procedimiento CREAR-ARBOL()

Se utiliza el mismo procedimiento que el propuesto en (Fajardo-Delgado *et al.*, 2010). La entrada de este procedimiento es el grafo G y el nodo especial r. La salida es un árbol de esparcimiento $T = (V_T, E_T)$ construido en G y enraizado en r. La construcción del árbol de esparcimiento se puede realizar mediante una simple búsqueda por anchura o una búsqueda por profundidad; también pueden utilizarse algoritmos para generar

Pseudo-código 2 Algoritmo CBAP$^+$

1: $T \leftarrow$CREAR-ARBOL(G, r)
2: $T \leftarrow$ETIQUETADO-INICIAL(T)
3: $T \leftarrow$CONTEO-VOTOS(T)
4: $nodosNoEspeculadores \leftarrow 0$
5: **mientras** (EXISTE-EQUILIBRIO$(G, T) = falso$) o
($nodosNoEspeculadores < |V_G|$) **hacer**
6: **mientras** $\exists u$ tal que $u \leftarrow$CALENDARIZADOR(G, T) **hacer**
7: $P_u^{viejo} \leftarrow P_u$
8: $T' \leftarrow$MOVIMIENTO-OPTIMIZACION(G, T, u)
9: $T \leftarrow$ACTUALIZACION(T', u, P_u^{viejo})
10: **fin_mientras**
11: $nodosNoEspeculadores \leftarrow 0$
12: **para todo** $u \in V_G$ **hacer**
13: **si** PUEDE-ESPECULAR$(G, T, u) = verdadero$ **entonces**
14: $v \leftarrow P_u$
15: $T' \leftarrow$MOVIMIENTO-ESPECULACION(G, T, u)
16: $T \leftarrow$ACTUALIZACION(T', u, v)
17: $P_v^{viejo} \leftarrow P_v$
18: $T' \leftarrow$MOVIMIENTO-OPTIMIZACION(G, T, v)
19: $T \leftarrow$ACTUALIZACION(T', v, P_v^{viejo})
20: **si** EXISTE-EQUILIBRIO$(G, T) = falso$ **entonces**
21: termina ciclo
22: **fin_si**
23: **si no**
24: $nodosNoEspeculadores \leftarrow nodosNoEspeculadores + 1$
25: **fin_si**
26: **fin_para**
27: **fin_mientras**

árboles de esparcimiento aleatorios (ver Broder (1989), Raidl y Julstrom (2003)). El árbol que se genera en T debe estar enraizado en el nodo especial r, tal que cada nodo u apunte al único vecino que en la trayectoria que lo lleve a r. El tiempo de este procedimiento depende del algoritmo empleado para crear el árbol de esparcimiento. Si se usan los algoritmos propuestos en (Raidl y Julstrom, 2003) y se utilizan listas de adyacencias para representar el grafo, entonces el tiempo óptimo para generar un árbol de esparcimiento es $O(m)$ y $O(n+m)$ si no se utilizan listas de adyacencias.

IV.3.2. Procedimiento ETIQUETADO-INICIAL()

Se utiliza el mismo procedimiento que el propuesto en (Fajardo-Delgado *et al.*, 2010). La entrada de este procedimiento es un árbol T y la salida es un árbol T' donde cada nodo u cuenta con la información necesaria para evitar la conformación de ciclos y pueda realizar movimientos de optimización válidos. Para ello, se supone que u utiliza las variables id_u y d_u^T para almacenar un identificador temporal y la distancia a la raíz. Se utiliza un sistema de etiquetado similar al propuesto por Collin y Dolev (1994). Los identificadores siguen un orden lexicográfico de acuerdo a la estructura de T. Para el etiquetado, cada nodo u asigna un orden local arbitrario η_u a cada arista de u, tal que $\eta_u(v)$ denota el índice de la arista que conecta a u con el nodo v. La raíz siempre inicializa $id_r \leftarrow \perp$. La etiqueta para cualquier nodo u consiste de la concatenación de la etiqueta de su padre v con el índice $\eta_v(u)$, es decir, $id_u \leftarrow id_v \circ \eta_v(u)$ (El simbolo "$\circ$" significa concatenación). Note que cualquier descendiente del nodo u tiene el identificador de u como prefijo de su identificador. Así, el sistema de etiquetado permite que los nodos detecten qué vecinos son descendientes y preservar la estructura del árbol durante un movimiento (un nodo no puede conectarse a sus descendientes). La distancia a la raíz

se calcula con base en la distancia del padre, es decir, $d_u^T \leftarrow d_{P_u}^T + 1$.

Para calcular los valores de id_u y d_u^T para cada nodo u, el algoritmo hace un recorrido pre-orden de T. Este procedimiento toma $O(n)$ pasos.

IV.3.3. Procedimiento CONTEO-VOTOS()

Se utiliza el mismo procedimiento que el propuesto por Fajardo-Delgado *et al.* (2010). La entrada de este procedimiento es un árbol T. La salida es un árbol T' donde cada nodo u registra, en sus variables α_u y β_u, el número de nodos blancos y negros en T_u. El número de nodos de cada color representa el número de votos correspondientes a beneficiar a los nodos del mismo color a través de un movimiento de u. El conteo de votos se realiza mediante un recorrido post-orden del árbol T comenzando de las hojas de T y terminando en la raíz. Durante el recorrido, cada nodo u recolecta su voto y los votos registrados por sus descendientes directos para almacenarlos en α_u y β_u. Este procedimiento toma $O(n)$ pasos.

IV.3.4. Procedimiento EXISTE-EQUILIBRIO()

La entrada de este procedimiento es un grafo G y un árbol T construido en G. La salida de este procedimiento es un valor Booleano *verdadero* cuando todos los nodos en V_G se encuentran en equilibrio y *falso* en otro caso. Este procedimiento revisa que cada nodo u apunte al vecino que maximiza la ganancia de u, es decir, $\pi_u(v) \geq \pi_u(v'), \ \forall v' \in A_u^G$. Este procedimiento requiere que se revisen todos los nodos y que cada nodo evalúe a cada uno de sus vecinos. Por lo tanto, este procedimiento toma $O(n+m)$ pasos.

IV.3.5. Procedimiento CALENDARIZADOR()

Se utiliza el mismo procedimiento que el propuesto por Fajardo-Delgado *et al.* (2010). La entrada de este procedimiento es un grafo G y un árbol T construido en G. La salida de este procedimiento es un nodo u que puede realizar un movimiento de optimización en T. Este procedimiento evalúa cada nodo en V_G para formar un conjunto de nodos elegibles denominado Z, donde $Z = \{u \in V_1 \cup V_2 | \pi_u(x) > 0 \text{ y } x \in A_u^G\}$. Una vez formado el conjunto, se selecciona un nodo de acuerdo a una de las siguientes políticas de selección:

- *Uniforme*: Esta política selecciona aleatoriamente a exactamente un nodo del conjunto Z con la misma probabilidad.
- *Voraz*: Selecciona al nodo $u \in Z$ tal que un movimiento de optimización de u genere el incremento más grande en ganancia; es decir, $\pi_u(x) \geq \pi_{u'}(x')\ \forall u' \in Z$ y $\forall x' \in A_{u'}^G$. Si dos o más nodos cumplen con esta condición, se selecciona cualquiera de estos nodos con la misma probabilidad.
- *Anti-Voraz*: Selecciona al nodo $u \in Z$ tal que un movimiento de optimización de u genere el incremento más pequeño en ganancia; es decir, $\pi_u(x) \leq \pi_{u'}(x')\ \forall u' \in Z$, $\forall x' \in A_{u'}^G$, $\pi_u(x)$ y $\pi_{u'}(x') > 0$. Si dos o más nodos cumplen con esta condición, se selecciona cualquiera de estos nodos con la misma probabilidad.

Cualquiera de estas políticas puede ser *elitista*, si al aplicar la política el calendarizador selecciona a un nodo de V_1 antes que a uno de V_2 o viceversa; cuando no existe tal distinción en Z, las políticas son *no elitistas*.

La parte que consume más tiempo para este procedimiento es generar el conjunto de nodos Z, ya que cada nodo tiene que evaluar las preferencias de conexión con todos

sus vecinos. Por lo tanto, este procedimiento toma $O(n+m)$ pasos.

IV.3.6. Procedimiento MOVIMIENTO-OPTIMIZACION()

Se utiliza el mismo procedimiento que el propuesto en (Fajardo-Delgado *et al.*, 2010). La entrada de este procedimiento es el grafo G, un árbol T construido en G, y un nodo u seleccionado por el procedimiento CALENDARIZADOR(). El procedimiento MOVIMIENTO-OPTIMIZACION() regresa un árbol nuevo T', como resultado de un movimiento de optimización del nodo u. Dado que el nodo u evalúa cada vecino de acuerdo a sus preferencias, este procedimiento toma $O(n)$ pasos.

IV.3.7. Procedimiento ACTUALIZACION()

Se utiliza el mismo procedimiento que el propuesto en (Fajardo-Delgado *et al.*, 2010). Este procedimiento toma como entrada el árbol T', y a los nodos u y P_u^{viejo} (el padre que tenía u antes de realizar el movimiento de optimización). La salida de este procedimiento es un árbol T con las etiquetas, distancias y votos actualizados para todos los nodos. A diferencia del procedimiento ETIQUETADO-INICIAL(), el procedimiento ACTUALIZACION() sólo actualiza las distancias y etiquetas de los nodos en T_u. De igual forma, solo tiene que actualizar los votos de los viejos y nuevos ancestros de u. Por lo tanto, este procedimiento toma $O(n)$ pasos.

IV.3.8. Procedimiento PUEDE-ESPECULAR()

La entrada de este procedimiento es el grafo G, un árbol T construido en G y el nodo u que va a evaluar si puede realizar un movimiento de especulación que genere dividendos. La salida de este procedimiento es un valor Booleano *verdadero* cuando u

detecta un vecino con el cual puede realizar un movimiento de especulación que genere dividendos y *falso* en otro caso. Note que el nodo u evalúa todos sus vecinos que no estén en el sub-árbol T_{P_u}. Así, este procedimiento toma $O(n)$ pasos.

IV.3.9. Procedimiento MOVIMIENTO-ESPECULACION()

La entrada de este procedimiento es el grafo G, un árbol T construido en G y el nodo u que va a realizar el movimiento de especulación. El procedimiento MOVIMIENTO-ESPECULACION() regresa un árbol nuevo T', producto de un movimiento de especulación del nodo u. El nodo u evalúa cada vecino para determinar cuál de ellos le genera un mayor dividendo. Por lo tanto este procedimiento toma $O(n)$ pasos

IV.4. Análisis del algoritmo CBAP$^+$

En esta sección se demuestra que el algoritmo CBAP$^+$ obtiene una configuración de equilibrio con igual o mayor beneficio social que el CBAP y en el mismo orden de tiempo de ejecución que el algoritmo CBAP.

Sea $T = (V_T, E_T)$ un árbol de esparcimiento generado en un grafo arbitrario G, a través de la ejecución de los pasos 1-3 del Pseudo-código 2.

Lema 1. *El algoritmo CBAP$^+$ siempre termina con un árbol de esparcimiento con una configuración en equilibrio.*

Demostración. Se demuestra por contradicción. Suponga que el algoritmo CBAP$^+$ termina y que el árbol T resultante no tiene una configuración de equilibrio. Dado que T no tiene una configuración de equilibrio, al menos un nodo u puede incrementar su ganancia individual, debido a esto, la condición del ciclo en el paso 5

EXISTE-EQUILIBRIO()=*falso* (ver Pseudo-código 2) aún se cumple. Por lo tanto, el algoritmo aún no termina y existe una contradicción. □

A continuación se demuestra que el algoritmo CBAP$^+$ tiene un tiempo de ejecución del mismo orden que el algoritmo CBAP.

Lema 2. *Cualquier movimiento de especulación propicia un incremento, al menos en uno, en el beneficio social.*

Demostración. Cualquier nodo u realiza un movimiento de especulación sólo si cumple con la Condición 14. Por el Teorema 1 cuando u realiza un movimiento de especulación, siempre se generan dividendos. Por lo tanto, el beneficio social incrementa al menos en uno. □

Lema 3. *El algoritmo CBAP$^+$ siempre termina con una configuración con igual o mayor beneficio social que el algoritmo CBAP.*

Demostración. El algoritmo CBAP termina su ejecución cuando encuentra un equilibrio; mientras que el algoritmo CBAP$^+$ aplica el algoritmo CBAP hasta llegar a un equilibrio y después aplica la técnica de especulación. Dado que este proceso se repite de forma iterativa hasta llegar a un equilibrio en el que ningún nodo puede realizar movimientos de especulación y por el Lema 2, entonces el beneficio social mínimo que puede entregar el algoritmo CBAP$^+$ es la configuración que entrega el algoritmo CBAP; como los movimientos de especulación realizados durante la técnica de especulación siempre incrementan el beneficio social, entonces si se realizaron movimientos de especulación posteriores al primer equilibrio, la configuración generada por el algoritmo CBAP$^+$ tiene un beneficio social de mayor valor que el del algoritmo CBAP. □

Lema 4. *El número de movimientos de especulación que puede realizar el algoritmo CBAP$^+$ es a lo más de $O(n^2)$.*

Demostración. Suponga un árbol inicial T resultado de la ejecución del algoritmo CBAP tal que $g(T) = \tau$. Suponga que existe un nodo u que puede realizar un movimiento de especulación. Se sabe por Fajardo-Delgado *et al.* (2010) que el valor de $g(T)$ está acotado por $O(n^2)$. Dado que $\tau > 0$ y cada movimiento de especulación incrementa al menos en uno el valor de $g(T)$ (ver Lema 2), entonces el máximo número de movimientos de especulación que pueden hacerse al aplicar la técnica de especulación es $O(n^2)$. □

Corolario 1. *El algoritmo CBAP$^+$ realiza $O(n^2)$ movimientos.*

Demostración. Bajo el enfoque cooperativo, cada movimiento de optimización incrementa al menos en uno el beneficio social (Fajardo-Delgado *et al.*, 2010). Por lo tanto, el número de movimientos de optimización está acotado por $O(n^2)$. Por el Lema 4, el número de movimientos de especulación aplicando la técnica de especulación está acotado por $O(n^2)$. Dado que ambos movimientos incrementan el beneficio social, el número de movimientos que realiza el algoritmo CBAP$^+$ es el número de movimientos de optimización más el número de movimientos de especulación. Por lo tanto, el número de movimientos que realiza el algoritmo CBAP$^+$ es $O(n^2)$. □

Note que cada movimiento de optimización y/o especulación requiere de $O(n+m)$ pasos para completarse. Por lo tanto, el tiempo total que lleva hacer todos los movimientos es de $O(n^3 + n^2m)$. El algoritmo CBAP$^+$ termina cuando el grafo tiene una condición de equilibrio y no hay un nodo que pueda realizar un movimiento de especulación. El tiempo que tarda el algoritmo CBAP$^+$ es $O(n^3+n^2m)$, el cual tiene el mismo

orden que el tiempo de ejecución del algoritmo CBAP propuesto por Fajardo-Delgado *et al.* (2010) bajo el enfoque cooperativo.

Capítulo V

Técnica de especulación extendida

Una característica deseable de la técnica de especulación (ver Capítulo IV) es que los nodos puedan acceder en tiempo constante a la información adicional que requieren para determinar si pueden realizar un movimiento de especulación; esto se debe a que la información se calcula en pasos anteriores de la ejecución del algoritmo. Así, el algoritmo no utiliza tiempo adicional para obtener la información requerida por la técnica de especulación. Para aplicar la técnica de especulación, cuando el algoritmo llega a un equilibrio, cada nodo realiza un cálculo para saber si puede realizar un movimiento de especulación que genere dividendos. Cuando los nodos realizan el cálculo, se les permite conocer más información acerca del nodo que va a generar los dividendos (se les permite conocer la cantidad de nodos blancos y negros de su sub-árbol); sin embargo, los nodos hacen suposiciones sobre el comportamiento a futuro del nodo que va a generar dividendos. Debido a estas suposiciones, existen situaciones en las que un nodo puede realizar un movimiento de especulación y el cálculo indique que no se van a generar dividendos. Para evitar estas situaciones, en este capítulo se propone una ampliación de la técnica de especulación, la cual permite que los nodos tengan más información acerca del nodo que va a generar los dividendos. Sin embargo, obtener la información nueva requiere de un costo adicional. El algoritmo propuesto en este capítulo permite integrar el proceso de recolección de la información adicional requerida por la técnica propuesta con el proceso de actualización de los nodos; así, el tiempo de ejecución del algoritmo sigue siendo del mismo orden.

A continuación se define el modelo del sistema y la terminología que se requiere para explicar la técnica propuesta.

V.1. Modelo del sistema

En este capítulo se supone un modelo del sistema extendido del modelo presentado en el Capítulo IV. En este nuevo modelo, cada nodo u registra en la variable $P_u max$ el máximo cambio en la distancia que puede lograr P_u si u dejara de pertenecer a T_{P_u} y P_u se conecta con un nodo $v \in D_u^T \cup A_{P_u}^G$; es decir $P_u max = d_v^T - d_{P_u}^T$, note que $P_u max > 0$ ya que siempre va a incrementar la distancia al menos en uno (si se conecta a u) y que es un valor simulado ya que al momento de calcular $P_u max$, $D_u^T \cup A_{P_u}^G = \emptyset$ ya que $u \in T_{P_u}$. Por lo tanto se simula el caso en que u no pertenece a T_{P_u}. Adicionalmente, u registra en la variable $d_u^T min$ la distancia mínima a la raíz que puede tener y en la variable $idMin$ el identificador del nodo vecino que lo acerca más a la raíz. Adicionalmente se introduce un nuevo tipo de movimiento denominado movimiento de optimización voraz. Un *movimiento de optimización voraz* es un movimiento de optimización en el que siempre se maximiza el valor de la función de pagos π (este movimiento utiliza siempre una estrategia voraz independiente de la estrategia de selección del nodo).

V.2. Técnica de especulación extendida

La idea de la técnica de especulación extendida es la siguiente: al momento de que los nodos evalúan si pueden realizar un movimiento de especulación que genere dividendos, se le entrega al nodo más información sobre las opciones de conexión del nodo que va a generar los dividendos; de esta forma, el nodo puede hacer una suposición más

aproximada sobre el comportamiento del nodo que va a generar los dividendos después del movimiento de especulación. Así, a cada nodo u se le entrega la siguiente información adicional al momento de realizar el cálculo:

- El cambio en la distancia máxima que puede alcanzar P_u en T_u si $u \notin T_{P_u}$; es decir, el cambio en la distancia que podría alcanzar P_u si se conecta a un nodo $v \in T_u$ después de que u realice su movimiento de especulación. La diferencia con la técnica de especulación es que en la técnica de especulación u supone que el nodo que genera los dividendos se va a conectar a u; ahora u contempla que puede conectarse a cualquier nodo $v \in T_u$. Así, la distancia que calcula u para P_u después del movimiento de P_u que genere dividendos es $d^T_{P_u} = d^T_u + P_u max$.

  Ejemplo 4. La Figura 8 muestra un grafo con una condición de equilibrio. Cuando el nodo u realiza el cálculo para ver si puede realizar un movimiento de especulación utilizando la técnica de especulación extendida, $P_u max = 4$, ya que el nodo v puede conectarse con el nodo w. Así, cuando el nodo u realiza el cálculo usando como nueva conexión al nodo r, $\Delta_{d^T_v} = 4$, al aplicar la Ecuación 14 se obtienen que se van a generar dividendos con valor $+2$. Note que si se utiliza la técnica de especulación $\Delta_{d^T_v} = 1$; por lo tanto, al aplicar la Ecuación 14, u calcula que se van a generar dividendos con valor -1 y u no realiza el movimiento de especulación.

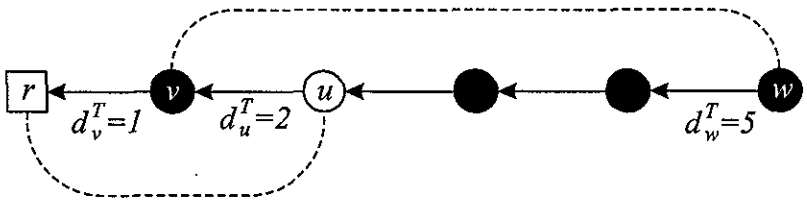

Figura 8. Movimiento de especulación extendida utilizando $P_u max$ en el cálculo.

- La distancia mínima a la raíz que puede tener P_u. Esta información permite que u contemple el caso en que P_u necesita acercarse a la raíz para generar dividendos; es decir, $T_{P_u} - T_u$ es $blanco^+$. Así, cuando u realiza el cálculo, u utiliza la distancia mínima a la raíz que puede obtener P_u; es decir, $d^T_{P_u} = min(d^T_u + 1, d^T_{P_u} min)$.

Ejemplo 5. La Figura 9 muestra un grafo con una condición de equilibrio. Cuando el nodo u realiza el cálculo para ver si puede realizar un movimiento de especulación utilizando la técnica de especulación extendida, $P_u min = 1$, ya que el nodo v puede conectarse directamente con el nodo r. Así, cuando el nodo u realiza el cálculo usando como nueva conexión al nodo w, $\Delta_{d^T_v} = -1$, al aplicar la Ecuación 14 se obtienen que se van a generar dividendos con valor $+1$. Note que si se utiliza la técnica de especulación $\Delta_{d^T_v} = 2$ ya que u supone que v se va a conectar a u al realizar el movimiento de optimización que genere dividendos; por lo tanto, al aplicar la Ecuación 14, u calcula que se van a generar dividendos con valor -2 y u no realiza el movimiento de especulación.

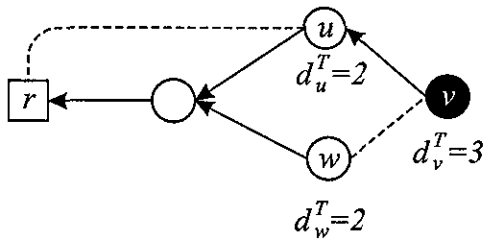

Figura 9. Movimiento de especulación extendida utilizando $P_u min$ en el cálculo.

Lema 5. *Cualquier movimiento de especulación que genere dividendos bajo la técnica de especulación también genera dividendos bajo la técnica de especulación extendida.*

Demostración. Tanto la técnica de especulación como la técnica de especulación extendida, utilizan la Ecuación 14 como una condición para evaluar si se van a generar

dividendos ante cada movimiento de especulación. La diferencia entre ambas, es que la técnica de especulación supone que P_u se conecta a u después del movimiento de especulación, mientras que la técnica de especulación extendida además, considera cualquier otra conexión alterna para P_u. Sea P_u el padre de u antes de realizar el movimiento de especulación, $\Delta_{d^T_{P_u}}$ es el cambio en la distancia de P_u cuando P_u realiza un movimiento de optimización que genera dividendos. En la técnica de especulación $\Delta_{d^T_{P_u}}$ depende de si se generan dividendos cuando P_u se acerca o aleja de la raíz. Sean d^T_u la distancia del nodo u después de hacer el movimiento de especulación y $d^T_{P_u}$ la distancia actual de P_u. Se analizan los siguientes casos:

- Cuando P_u genera dividendos al alejarse de la raíz, $\Delta_{d^T_{P_u}} = (d^T_u + P_u max) - d^T_{P_u}$. Al aplicar la técnica de especulación, u supone que P_u se va a conectar a u por lo tanto utiliza $P_u max = 1$. Dado que $P_u max > 0$, entonces el cálculo de la técnica de especulación extendida siempre calcula dividendos con valor mayor o igual a los calculados por la técnica de especulación.

- Cuando P_u genera dividendos al acercarse a la raíz, entonces se calculan los dividendos utilizando la distancia mínima a la raíz que puede obtener P_u; es decir, $\Delta_{d^T_{P_u}} = min(d^T_u + 1, d^T_{P_u} min) - d^T_{P_u}$. Al aplicar la técnica de especulación, u supone que P_u se va a conectar a u; por lo tanto, u calcula $\Delta_{d^T_{P_u}} = (d^T_u + 1) - d^T_{P_u}$. Dado que en la técnica de especulación extendida $\Delta_{d^T_{P_u}} = min(d^T_u + 1, d^T_{P_u} min) - d^T_{P_u}$ y P_u genera dividendos más grandes conforme se acerca a la raíz, entonces la técnica de especulación extendida siempre calcula dividendos con valor mayor o igual a los calculados por la técnica de especulación.

□

A continuación se describe una versión del algoritmo CBAP junto con la técnica de especulación extendida.

V.3. Algoritmo CBAP$^{\times}$

En esta sección se presenta una versión modificada del algoritmo CBAP^{+}, denominada CBAP$^{\times}$. Esta nueva versión consiste en la implementación de la técnica de especulación exhaustiva. La entrada para el algoritmo CBAP$^{\times}$ es un grafo conectado no dirigido $G = (V_G, E_G)$, donde $|V_G| = n$ y $|E_G| = m$, y una partición del conjunto de nodos $V = V_1 \cup V_2 \cup V_3$. La salida es un árbol de esparcimiento $T = (V_T, E_T)$ que satisface una condición de equilibrio para el PAG, la cuál no siempre maximiza el beneficio social.

El algoritmo CBAP$^{\times}$ se muestra en el Pseudo-código 3. Los Pasos 1-3 construyen un árbol inicial T en el grafo de entrada e inicializan las variables de cada nodo del grafo. Los pasos 5-28 realizan la optimización iterativa del árbol inicial hasta que el árbol se encuentra en equilibrio y ningún nodo puede realizar un movimiento de especulación donde se generen dividendos. A continuación se explican cada uno de los procedimientos que se utilizan en el Pseudo-código 3 del algoritmo CBAP$^{\times}$.

V.3.1. Procedimiento CREAR-ARBOL()

Este procedimiento se define en la Sección IV.3.1. El tiempo de ejecución de este procedimiento es $O(n + m)$ pasos.

Pseudo-código 3 Algoritmo CBAP$^{\times}$

```
1: T ←CREAR-ARBOL(G, r)
2: T ←ETIQUETADO-INICIAL(T)
3: T ←CONTEO-VOTOS(T)
4: nodosNoEspeculadores ← 0
5: mientras (EXISTE-EQUILIBRIO(G, T) = falso) o
   (nodosNoEspeculadores < |V_G|) hacer
6:   mientras ∃u tal que u ←CALENDARIZADOR(G, T) hacer
7:     P_u^viejo ← P_u
8:     T' ←MOVIMIENTO-OPTIMIZACION(G, T, u)
9:     T ←ACTUALIZACION(T', u, P_u^viejo)
10:  fin_mientras
11:  ACTUALIZACION-EXHAUSTIVA(T)
12:  nodosNoEspeculadores ← 0
13:  para todo u ∈ V_G hacer
14:    si PUEDE-ESPECULAR(G, T, u) = verdadero entonces
15:      v ← P_u
16:      T' ←MOVIMIENTO-ESPECULACION(G, T, u)
17:      T ←ACTUALIZACION-EXTENDIDA(T', u, v)
18:      P_v^viejo ← P_v
19:      T' ←MOVIMIENTO-VORAZ(G, T, v)
20:      T ←ACTUALIZACION-EXTENDIDA(T', v, P_v^viejo)
21:      si EXISTE-EQUILIBRIO(G, T) = falso entonces
22:        termina ciclo
23:      fin_si
24:    si no
25:      nodosNoEspeculadores ← nodosNoEspeculadores + 1
26:    fin_si
27:  fin_para
28: fin_mientras
```

V.3.2. Procedimiento ETIQUETADO-INICIAL()

Este procedimiento se define en la Sección IV.3.2. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

V.3.3. Procedimiento CONTEO-VOTOS()

Este procedimiento se define en la Sección IV.3.3. El tiempo de ejecución de este algoritmo es $O(n)$ pasos.

V.3.4. Procedimiento EXISTE-EQUILIBRIO()

Este procedimiento se define en la Sección IV.3.4. El tiempo de ejecución de este procedimiento es $O(n+m)$ pasos.

V.3.5. Procedimiento CALENDARIZADOR()

Este procedimiento se define en la Sección IV.3.5. El tiempo de ejecución de este procedimiento es $O(n+m)$ pasos.

V.3.6. Procedimiento MOVIMIENTO-OPTIMIZACION()

Este procedimiento se define en la Sección IV.3.6. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

V.3.7. Procedimiento ACTUALIZACION()

Este procedimiento se define en la Sección IV.3.7. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

V.3.8. Procedimiento ACTUALIZACION-EXHAUSTIVA()

La entrada de este procedimiento es un árbol T. La salida es un árbol T' donde cada nodo u registra en su variable $P_u max$ la distancia máxima que puede obtener P_u a partir de u si se conecta a un nodo v descendiente de u y de P_u; es decir, $d_v^T \geq d_{v'}^T \; \forall \; v, v' \in D_u^T \cap D_{P_u}^T$. Además registra en la variable $d_u^T min$ la distancia más corta que puede alcanzar u a r y en la variable $idMin$ el identificador del nodo que minimiza la distancia de u a r. El proceso de actualización exhaustiva se realiza mediante un recorrido post-orden del árbol T comenzando de las hojas de T y terminando en la raíz. Durante el recorrido, cada nodo u busca entre sus vecinos qué nodo es el que minimiza su distancia a la raíz y pregunta a P_u la distancia que puede alcanzar si se conecta a un nodo $v \in T_u$. Dado que el nodo u tiene que revisar en cada uno de sus vecinos cuál es la distancia mínima a la raíz, y el nodo P_u tiene que buscar en cada uno de sus vecinos al nodo $v \in D_u^T \cap D_{P_u}^T$ que lo aleja más de la raíz, entonces el tiempo de ejecución de este procedimiento es de $O(n+m)$ pasos.

V.3.9. Procedimiento PUEDE-ESPECULAR()

Este procedimiento se define en la Sección IV.3.8. La diferencia de este procedimiento con el procedimiento de la Sección IV.3.8 es que este nuevo procedimiento utiliza las reglas de la técnica de especulación extendida para calcular $\Delta_{d_{P_u}^T}$. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

V.3.10. Procedimiento MOVIMIENTO-ESPECULACION()

Este procedimiento se define en la Sección IV.3.9. La diferencia de este procedimiento con el procedimiento de la Sección IV.3.9 es que este nuevo procedimiento utiliza las

reglas de la técnica de especulación extendida para calcular $\Delta_{d^T_{P_u}}$. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

V.3.11. Procedimiento ACTUALIZACION-EXTENDIDA()

Este procedimiento toma como entrada el árbol T', y a los nodos u y P_u^{viejo} (el padre que tenía u antes de realizar el movimiento de optimización). La salida de este procedimiento es un árbol T con las etiquetas, distancias, la distancia mínima a la raíz, la distancia máxima del padre en el sub-árbol y votos actualizados para todos los nodos. A diferencia del procedimiento ACTUALIZACION(), el procedimiento ACTUALIZACION-EXTENDIDA() aprovecha la actualización de las etiquetas y distancia de los nodos en T_u para actualizar la distancia mínima a la raíz de todos los nodos vecinos de los nodos en T_u. Además, se aprovecha el proceso de actualización de votos para actualizar los descendientes más lejanos del padre de u y cada ancestro de u. Dado que cada nodo en T_u tiene que notificar su nueva distancia a sus nodos vecinos y cada ancestro de u tiene que calcular la máxima distancia de su padre por su sub-árbol, este procedimiento toma $O(n+m)$ pasos.

V.3.12. Procedimiento MOVIMIENTO-VORAZ()

La entrada de este procedimiento es el grafo G, un árbol T construido en G, y un nodo u. El procedimiento MOVIMIENTO-VORAZ() regresa un árbol nuevo T', como resultado de un movimiento de optimización voraz del nodo u. Dado que el nodo u evalúa cada vecino de acuerdo a sus preferencias, este procedimiento toma $O(n)$ pasos.

V.4. Análisis del algoritmo CBAP$^{\times}$

En esta sección se demuestra que el algoritmo CBAP$^{\times}$ obtiene una configuración de equilibrio con igual o mayor beneficio social que el CBAP y en el mismo orden de tiempo de ejecución que el algoritmo CBAP.

Sea $T = (V_T, E_T)$ un árbol de esparcimiento generado en un grafo arbitrario G, a través de la ejecución de los pasos 1-3 del Pseudo-código 3.

Lema 6. *El algoritmo CBAP$^{\times}$ siempre termina con un árbol de esparcimiento con una configuración en equilibrio.*

Demostración. Se demuestra por contradicción. Suponga que el algoritmo CBAP$^{\times}$ termina y que el árbol T resultante no tiene una configuración de equilibrio. Dado que T no tiene una configuración de equilibrio, al menos un nodo u puede incrementar su ganancia individual, debido a esto, la condición del ciclo en el paso 5, EXISTE-EQUILIBRIO() = *falso* (ver Pseudo-código 3) aún se cumple. Por lo tanto, el algoritmo aún no termina y existe una contradicción. □

Lema 7. *Cualquier movimiento de especulación realizado con la técnica de escape de especulación extendida propicia un incremento, al menos en uno, en el beneficio social.*

Demostración. Como el nodo que genera dividendos realiza un movimiento de optimización voraz, entonces el nodo que genera dividendos siempre selecciona la conexión que maximiza los pagos del nodo. Dado que el movimiento de especulación se realiza solo si cumple con la Condición 14, entonces por el Teorema 1 el beneficio social incrementa al menos en uno. □

Lema 8. *El algoritmo CBAP$^{\times}$ siempre termina con una configuración con igual o mayor beneficio social que el algoritmo CBAP.*

Demostración. El algoritmo CBAP termina su ejecución cuando encuentra un equilibrio; mientras que el algoritmo CBAP$^{\times}$ aplica el algoritmo CBAP hasta llegar a un equilibrio y después aplica la técnica de especulación extendida. Dado que este proceso se repite de forma iterativa hasta llegar a un equilibrio en el que ningún nodo puede realizar movimientos de especulación y por el Teorema 1, entonces el beneficio social mínimo que puede entregar el algoritmo CBAP$^{\times}$ es la configuración que entrega el algoritmo CBAP; como los movimientos de especulación realizados durante la técnica de especulación extendida siempre incrementan el beneficio social, entonces si se realizaron movimientos de especulación posteriores al primer equilibrio la configuración tiene un beneficio social de mayor valor que el de el algoritmo CBAP. □

A continuación se demuestra que el algoritmo CBAP$^{\times}$ tiene un tiempo de ejecución del mismo orden que el algoritmo CBAP.

Corolario 2. *El algoritmo CBAP$^{\times}$ realiza $O(n^2)$ movimientos.*

Demostración. Dado que el algoritmo CBAP$^{\times}$ es el mismo algoritmo que el algoritmo CBAP^{+} con la diferencia en el procedimiento de actualización de los nodos y el cálculo de la generación de dividendos al hacer un movimiento de especulación; entonces por el Lema 1 el número de movimientos que realiza el algoritmo CBAP$^{\times}$ es $O(n^2)$. □

Note que cada movimiento de optimización y/o especulación requiere de $O(n+m)$ pasos para completarse. Además, note que el tiempo adicional que requiere el algoritmo para realizar la actualización extendida de los nodos requiere de $= (n+m)$ pasos. Por lo tanto, el tiempo total que lleva hacer todos los movimientos es de $O(n^3 + n^2m)$. El algoritmo CBAP$^{\times}$ termina cuando el grafo tiene una condición de equilibrio y no hay un nodo que pueda realizar un movimiento de especulación. El tiempo que tarda el algorit-

mo CBAP$^{\times}$ es $O(n^3+n^2m)$, que es el mismo orden del tiempo de ejecución del algoritmo CBAP propuesto por Fajardo-Delgado *et al.* (2010) bajo el enfoque cooperativo.

Capítulo VI

Técnica de especulación con inversión cero

Una característica deseable de los algoritmos CBAP^{+} y CBAP$^{\times}$ (ver capítulos IV y V) es que cuando se realiza un movimiento de especulación, el algoritmo garantiza que el movimiento de optimización posterior va a generar dividendos; así, ambos algoritmos garantizan un resultado igual o mejor que el algoritmo CBAP. Para poder garantizar que los dividendos se mantienen en movimientos posteriores al movimiento de especulación, los algoritmos CBAP^{+} y CBAP$^{\times}$ trabajan únicamente bajo el enfoque cooperativo. Sin embargo, ninguno de los dos algoritmos explota completamente la propiedad de incremento en el beneficio social del enfoque cooperativo, ya que solo realizan el movimiento de especulación cuando saben que van a generarse dividendos; es decir, existen situaciones en las que el movimiento de especulación desencadena el movimiento de varios nodos en los que la suma de la ganancia obtenida por cada nodo que realizó un movimiento de optimización puede generar dividendos y no se realiza el movimiento de especulación ya que se calcula que el nodo que hace el primer movimiento de optimización posterior no genera dividendos. En este capítulo se proponen dos algoritmos que extienden al algoritmo CBAP^{+} y CBAP$^{\times}$ que intentan incrementar el beneficio social después de que ya no existen nodos que puedan realizar movimientos de especulación que generen dividendos.

A continuación se define el modelo del sistema y la terminología que se requiere para explicar la técnica propuesta.

VI.1. Modelo del sistema

En este capítulo, el modelo del sistema depende del la técnica de especulación que se está extendiendo. Cuando se trabaja con la técnica de especulación (algoritmo CBAP^{+}), se utiliza el modelo presentado en el Capítulo IV; cuando se trabaja con la técnica de especulación extendida (algoritmo CBAP$^{\times}$) se utiliza el modelo presentado en el Capítulo V. Adicionalmente, se introduce un nuevo tipo de movimiento denominado movimiento de especulación con inversión cero. Un *movimiento de especulación con inversión cero* es un movimiento de especulación en el que los dividendos calculados pueden ser nulos.

VI.2. Técnica de especulación con inversión cero

La intuición detrás de la técnica de especulación con inversión cero es la siguiente: se permite a un nodo "perturbar" la condición de equilibrio del sistema obtenida por una ejecución previa del algoritmo CBAP, para permitir que otros nodos realicen movimientos alternativos de optimización. La perturbación del equilibrio se realiza a través de un movimiento de especulación con inversión cero en el que se garantice que el movimiento posterior recupera la "pérdida". La diferencia con la técnica de especulación es que la técnica de especulación con inversión cero no supone que va a existir un incremento en el beneficio social; es decir, solo busca romper el equilibrio con una configuración en la que se mantenga el beneficio social que se tenía antes del movimiento de especulación con inversión cero; es decir, la técnica de especulación con inversión cero, relaja la desigualdad de la Ecuación 14 a:

$$(\beta_w - \alpha_w)\Delta_{d_w^T} + ((\beta_b - \beta_w) - (\alpha_b - \alpha_w))\Delta_{d_b^T} \geq 0 \tag{15}$$

Donde w es el nodo que va a realizar el movimiento de especulación y b el nodo que va a generar los dividendos. Note que ahora la desigualdad establece que puede ser mayor o igual a cero; es decir, cuando es igual a cero entonces se calcula que b solo recupera la pérdida de w al hacer el movimiento de especulación (no existe un cambio en el beneficio social).

Al permitir movimientos de especulación en los que no se generen dividendos también permite que se generen situaciones en las que un nodo puede quedar atrapado en un ciclo realizando movimientos de especulación con inversión cero. El Ejemplo 6 muestra una situación en la que puede generarse un ciclo de movimientos de especulación con inversión cero. Así, para evitar la conformación de ciclos a través de movimientos de especulación con inversión cero, se permite a los nodos realizar el movimiento de especulación con inversión cero solo cuando cumple con las siguientes condiciones:

- Se cumple con la Ecuación 15.
- La nueva conexión favorece al color del sub-árbol; es decir, si el sub-árbol es *negro* entonces la nueva conexión aleja al sub-árbol de la raíz y si es *blanco* lo acerca a la raíz.
- Cuando es un sub-árbol *gris*, la nueva conexión favorece al color del nodo; es decir, si el nodo que realiza el movimiento de especulación es *blanco*, entonces la nueva conexión tiene que estar más cerca a la raíz y si es *negro* tiene que estar mas lejos.

Note que con estas condiciones se limita la cantidad de movimientos de especulación con inversión cero que se pueden realizar; dado que el sistema se encuentra bajo una condición de equilibrio, los nodos con sub-árboles de color *blanco* (*negro*) ya se encuentran en la conexión que acerca (aleja) al sub-árbol de la raíz. Así, se pierden algunos

movimientos de especulación con inversión cero que se pueden realizar; sin embargo, se evita que se generen los ciclos.

Ejemplo 6. La Figura 10a muestra una configuración en equilibrio, en la que T_u, T_v y T_w son árboles de color *gris*; note los movimientos de estos nodos no perturban el beneficio social. En la Figura 10b, el nodo u calcula que puede hacer un movimiento de especulación con inversión cero si cambia su conexión de v a w y cambia su apuntador. Como v es un árbol gris, cuando realiza el movimiento de optimización que genera dividendos, v mantiene su conexión (por las reglas de los movimientos de optimización para el enfoque cooperativo, v no cambia su apuntador). En la Figura 10c, el nodo u vuelve a calcular que puede hacer un movimiento de especulación con inversión cero si cambia su conexión de w a v y cambia su apuntador, regresando a la configuración original y por lo tanto se genera un ciclo.

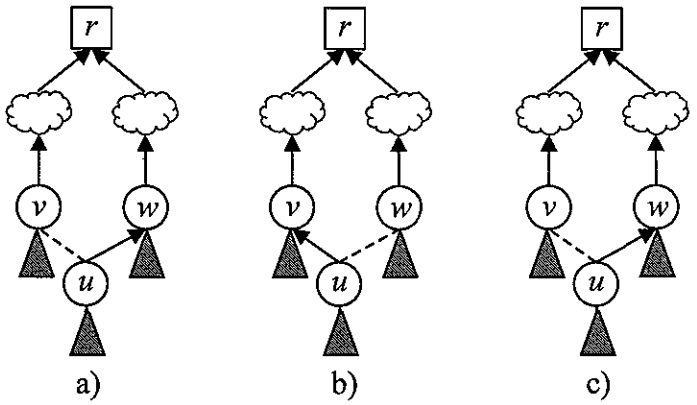

Figura 10. Ciclo formado al usar movimientos de especulación con inversión cero; a) condición de equilibrio encontrada por el algoritmo CBAP; b) movimiento de especulación del nodo u; c) movimiento de especulación del nodo u.

A continuación se describe una versión del algoritmo CBAP$^+$ y CBAP$^\times$ junto con la técnica de especulación con inversión cero, en esta versión se implemente la técnica de especulación y después la técnica de especulación con inversión cero; así, se le da

prioridad a los movimientos de especulación que si garantizan dividendos.

VI.3. Algoritmo CBAP^{++}

En esta sección se presenta una versión modificada del algoritmo CBAP^{+} (ver Sección IV), denominada CBAP^{++}. Esta nueva versión consiste en la implementación de la técnica de especulación con inversión cero. La entrada para el algoritmo CBAP^{++} es un grafo conectado no dirigido $G = (V_G, E_G)$, donde $|V_G| = n$ y $|E_G| = m$, y una partición del conjunto de nodos $V = V_1 \cup V_2 \cup V_3$. La salida es un árbol de esparcimiento $T = (V_T, E_T)$ que satisface una condición de equilibrio para el PAG, la cuál no siempre maximiza el beneficio social.

El algoritmo CBAP^{++} se muestra en el Pseudo-código 4. Los Pasos 1-3 construyen un árbol inicial T en el grafo de entrada e inicializan las variables de cada nodo del grafo. Los pasos 5-45 realizan la optimización iterativa del árbol inicial hasta que el árbol se encuentra en equilibrio y ningún nodo puede realizar un movimiento de especulación donde se generen dividendos. A continuación se explican cada uno de los procedimientos que se utilizan en el Pseudo-código 4 del algoritmo CBAP^{++}.

VI.3.1. Procedimiento CREAR-ARBOL()

Este procedimiento se define en la Sección IV.3.1. El tiempo de ejecución de este procedimiento es $O(n + m)$ pasos.

VI.3.2. Procedimiento ETIQUETADO-INICIAL()

Este procedimiento se define en la Sección IV.3.2. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

Pseudo-código 4 Algoritmo CBAP^{++}

```
1: t ←CREAR-ARBOL(g, r)
2: t ←ETIQUETADO-INICIAL(t)
3: t ←CONTEO-VOTOS(t)
4: nodosnoespeculadores ← 0
5: mientras (EXISTE-EQUILIBRIO(g, t) = falso) o
   (nodosNoEspeculadores < |v_g|) hacer
6:   mientras ∃u tal que u ←CALENDARIZADOR(g, t) hacer
7:     p_u^viejo ← p_u
8:     t' ←MOVIMIENTO-OPTIMIZACION(g, t, u)
9:     t ←ACTUALIZACION(t', u, p_u^viejo)
10:  fin_mientras
11:  nodosnoespeculadores ← 0
12:  para todo u ∈ v_g hacer
13:    si PUEDE-ESPECULAR(g, t, u) = verdadero entonces
14:      v ← P_u
15:      T' ←MOVIMIENTO-ESPECULACION(G, T, u)
16:      T ←ACTUALIZACION(T', u, v)
17:      P_v^viejo ← P_v
18:      T' ←MOVIMIENTO-OPTIMIZACION(G, T, v)
19:      T ←ACTUALIZACION(T', v, P_v^viejo)
20:      si EXISTE-EQUILIBRIO(G, T) = falso entonces
21:        termina ciclo
22:      fin_si
23:    si no
24:      nodosNoEspeculadores ← nodosNoEspeculadores + 1
25:    fin_si
26:  fin_para
27:  si EXISTE-EQUILIBRIO(G, T) = verdadero ∧ nodosNoEspeculadores = |V_G| en-
     tonces
28:    nodosNoEspeculadores ← 0
29:    para todo u ∈ V_G hacer
30:      si PUEDE-ESPECULAR-CERO(G, T, u) = verdadero entonces
31:        v ← P_u
32:        T' ←MOVIMIENTO-ESPECULACION-CERO(G, T, u)
33:        T ←ACTUALIZACION(T', u, v)
34:        P_v^viejo ← P_v
35:        T' ←MOVIMIENTO-OPTIMIZACION(G, T, v)
36:        T ←ACTUALIZACION(T', v, P_v^viejo)
37:        si EXISTE-EQUILIBRIO(G, T) = falso entonces
38:          termina ciclo
39:        fin_si
40:      si no
41:        nodosNoEspeculadores ← nodosNoEspeculadores + 1
42:      fin_si
43:    fin_para
44:  fin_si
45: fin_mientras
```

VI.3.3. Procedimiento CONTEO-VOTOS()

Este procedimiento se define en la Sección IV.3.3. El tiempo de ejecución de este algoritmo es $O(n)$ pasos.

VI.3.4. Procedimiento EXISTE-EQUILIBRIO()

Este procedimiento se define en la Sección IV.3.4. El tiempo de ejecución de este procedimiento es $O(n+m)$ pasos.

VI.3.5. Procedimiento CALENDARIZADOR()

Este procedimiento se define en la Sección IV.3.5. El tiempo de ejecución de este procedimiento es $O(n+m)$ pasos.

VI.3.6. Procedimiento MOVIMIENTO-OPTIMIZACION()

Este procedimiento se define en la Sección IV.3.6. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.3.7. Procedimiento ACTUALIZACION()

Este procedimiento se define en la Sección IV.3.7. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.3.8. Procedimiento PUEDE-ESPECULAR()

Este procedimiento se define en la Sección IV.3.8. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.3.9. Procedimiento MOVIMIENTO-ESPECULACION()

Este procedimiento se define en la Sección IV.3.9. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.3.10. Procedimiento PUEDE-ESPECULAR-CERO()

Este procedimiento se define en la Sección IV.3.8. La diferencia de este procedimiento con el procedimiento de la Sección IV.3.8 es que este nuevo procedimiento utiliza las reglas de la técnica de especulación cero para calcular si puede realizar un movimiento de especulación o no. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.3.11. Procedimiento MOVIMIENTO-ESPECULACION-CERO()

Este procedimiento se define en la Sección IV.3.9. La diferencia de este procedimiento con el procedimiento de la Sección IV.3.9 es que este nuevo procedimiento utiliza las reglas de la técnica de especulación cero para seleccionar la nueva conexión. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.4. Análisis del algoritmo CBAP^{++}

En esta sección se demuestra que el algoritmo CBAP^{++} obtiene una configuración de equilibrio con igual o mayor beneficio social que el CBAP^{+} y en el mismo orden de tiempo de ejecución que el algoritmo CBAP.

Sea $T = (V_T, E_T)$ un árbol de esparcimiento generado en un grafo arbitrario G, a través de la ejecución de los pasos 1-3 del Pseudo-código 2.

Lema 9. *El algoritmo CBAP^{++} siempre termina con un árbol de esparcimiento con una configuración en equilibrio.*

Demostración. Se demuestra por contradicción. Suponga que el algoritmo CBAP^{++} termina y que el árbol T resultante no tiene una configuración de equilibrio. Dado que T no tiene una configuración de equilibrio, al menos un nodo u puede incrementar su ganancia individual, debido a esto, la condición del ciclo en el paso 5, EXISTE-EQUILIBRIO()=$falso$ (ver Pseudo-código 4) aún se cumple. Por lo tanto, el algoritmo aún no termina y existe una contradicción. □

Lema 10. *El algoritmo CBAP^{++} siempre termina con una configuración con igual o mayor beneficio social que el algoritmo CBAP^{+}.*

Demostración. El algoritmo CBAP^{+} termina su ejecución cuando encuentra un equilibrio en el que no existe un nodo que puede realizar un movimiento de especulación; mientras que el algoritmo CBAP^{++} aplica la técnica de especulación con inversión cero una vez que llega a un equilibrio donde no puede aplicar la técnica de especulación. Dado que este proceso se repite de forma iterativa hasta llegar a un equilibrio en el que ningún nodo puede realizar movimientos de especulación cero y que por la Ecuación 15 se garantiza que no existe una pérdida en el beneficio social (el movimiento de optimización posterior al movimiento de especulación con inversión cero siempre recupera la pérdida), entonces el beneficio social mínimo que puede entregar el algoritmo CBAP^{++} es el beneficio social de la configuración que entrega el algoritmo CBAP^{+}; como los movimientos realizados al aplicar la técnica de especulación con inversión cero perturban el equilibrio; entonces, cualquier movimiento de optimización adicional al movimiento que recupera la pérdida siempre incrementan el beneficio social; así, cuando el algoritmo vuelve a entrar en el ciclo de movimientos de optimización (ver Pseudo-código 4,

línea 6) después de aplicar la técnica de especulación con inversión cero, entonces la configuración tiene un beneficio social de mayor valor que el del algoritmo CBAP^{+}. □

A continuación se demuestra que el algoritmo CBAP^{++} tiene un tiempo de ejecución del mismo orden que el algoritmo CBAP.

Conjetura 1. *El algoritmo CBAP^{++} realiza $O(n^2)$ movimientos de especulación con inversión cero.*

La idea de esta conjetura se basa en el siguiente razonamiento. Suponga que existe un grafo arbitrario en donde se encuentra un árbol que cumple con una condición de equilibrio para el PAG y que ningún nodo puede realizar un movimiento de especulación. Suponga además que existe un sub-árbol *gris* enraizado en el nodo u, en donde u puede realizar un movimiento de especulación con inversión cero; es decir, un movimiento donde el sub-árbol de u no gana ni pierde. Dado que todos los posibles movimientos de u son de inversión cero, entonces u rompe el equilibrio a través de un movimiento que maximice su función de utilidad individual (definida por las ecuaciones 2 y 3). La distancia de u después de realizar el movimiento de especulación con inversión cero, tiene una diferencia de $\Delta_{d_u^T}$ con respecto a la distancia original. Así, si u es negro, entonces $d < d'$ y cualquier otro futuro movimiento de especulación con inversión cero también. En el peor escenario, el nodo u está conectado directamente a la raíz por lo que el número máximo de movimientos de especulación con inversión cero que puede realizar u es de $O(n)$. Si todos los nodos negros se comportan de esta forma entonces se realizan $O(n^2)$ movimientos de especulación por inversión cero. Este razonamiento es similar para los nodos blancos.

Corolario 3. *El algoritmo CBAP^{++} realiza $O(n^2)$ movimientos.*

Demostración. Los movimientos que puede realizar el algoritmo CBAP^{++} son movimientos de optimización, de especulación y de especulación con inversión cero. Por el Corolario 1 (ver Capítulo IV) se sabe que el algoritmo realiza $O(n^2)$ movimientos de optimización y de especulación. Por la Conjetura 1 se tiene que el algoritmo CBAP^{++} realiza $O(n^2)$ movimientos de especulación con inversión cero. Por lo tanto, el número de movimientos que puede realizar el algoritmo CBAP^{++} es de $O(n^2)$ movimientos. □

Note que cada movimiento de optimización, especulación y/o especulación con inversión cero requiere de $O(n+m)$ pasos para completarse. Por lo tanto, el tiempo total que lleva hacer todos los movimientos es de $O(n^3+n^2m)$. El algoritmo CBAP^{++} termina cuando el grafo tiene una condición de equilibrio y no hay un nodo que pueda realizar un movimiento de especulación con inversión cero. El tiempo que tarda el algoritmo CBAP^{++} es $O(n^3+n^2m)$, que es el mismo orden del tiempo de ejecución del algoritmo CBAP propuesto por Fajardo-Delgado *et al.* (2010) bajo el enfoque cooperativo.

VI.5. Algoritmo CBAP$^{\times\times}$

En esta sección se presenta una versión modificada del algoritmo CBAP$^{\times}$, denominada CBAP$^{\times\times}$. Esta nueva versión consiste en la implementación de la técnica de especulación con inversión cero. La entrada para el algoritmo CBAP$^{\times\times}$ es un grafo conectado no dirigido $G = (V_G, E_G)$, donde $|V_G| = n$ y $|E_G| = m$, y una partición del conjunto de nodos $V = V_1 \cup V_2 \cup V_3$. La salida es un árbol de esparcimiento $T = (V_T, E_T)$ que satisface una condición de equilibrio para el PAG, la cuál no siempre maximiza el beneficio social.

Pseudo-código 5 Algoritmo CBAP$^{\times\times}$

```
1: T ←CREAR-ARBOL(G, r)
2: T ←ETIQUETADO-INICIAL(T)
3: T ←CONTEO-VOTOS(T)
4: nodosNoEspeculadores ← 0
5: mientras (EXISTE-EQUILIBRIO(G, T) = falso) o
   (nodosNoEspeculadores < |V_G|) hacer
6:   mientras ∃u tal que u ←CALENDARIZADOR(G, T) hacer
7:     P_u^viejo ← P_u
8:     T' ←MOVIMIENTO-OPTIMIZACION(G, T, u)
9:     T ←ACTUALIZACION(T', u, P_u^viejo)
10:  fin_mientras
11:  ACTUALIZACION-EXHAUSTIVA(T)
12:  nodosNoEspeculadores ← 0
13:  para todo u ∈ V_G hacer
14:    si PUEDE-ESPECULAR(G, T, u) = verdadero entonces
15:      v ← P_u
16:      T' ←MOVIMIENTO-ESPECULACION(G, T, u)
17:      T ←ACTUALIZACION-EXTENDIDA(T', u, v)
18:      P_v^viejo ← P_v
19:      T' ←MOVIMIENTO-VORAZ(G, T, v)
20:      T ←ACTUALIZACION-EXTENDIDA(T', v, P_v^viejo)
21:      si EXISTE-EQUILIBRIO(G, T) = falso entonces
22:        termina ciclo
23:      fin_si
24:    si no
25:      nodosNoEspeculadores ← nodosNoEspeculadores + 1
26:    fin_si
27:  fin_para
28:  si (EXISTE-EQUILIBRIO(G, T) = verdadero ∧ nodosNoEspeculadores = |V_G|)
     entonces
29:    nodosNoEspeculadores ← 0
30:    para todo u ∈ V_G hacer
31:      si (PUEDE-ESPECULAR-CERO(G, T, u) = verdadero) entonces
32:        v ← P_u
33:        T' ←MOVIMIENTO-ESPECULACION-CERO(G, T, u)
34:        T ←ACTUALIZACION-EXTENDIDA(T', u, v)
35:        P_v^viejo ← P_v
36:        T' ←MOVIMIENTO-VORAZ(G, T, v)
37:        T ←ACTUALIZACION-EXTENDIDA(T', v, P_v^viejo)
38:        si Existe-Equilibrio(G, T) = falso entonces
39:          termina ciclo
40:        fin_si
41:      si no
42:        nodosNoEspeculadores ← nodosNoEspeculadores + 1
43:      fin_si
44:    fin_para
45:  fin_si
46: fin_mientras
```

El algoritmo CBAP$^{\times\times}$ se muestra en el Pseudo-código 5. Los Pasos 1-3 construyen un árbol inicial T en el grafo de entrada e inicializan las variables de cada nodo del grafo. Los pasos 5-46 realizan la optimización iterativa del árbol inicial hasta que el árbol se encuentra en equilibrio y ningún nodo puede realizar un movimiento de especulación donde se generen dividendos. A continuación se explican cada uno de los procedimientos que se utilizan en el Pseudo-código 5 del algoritmo CBAP$^{\times\times}$.

VI.5.1. Procedimiento CREAR-ARBOL()

Este procedimiento se define en la Sección IV.3.1. El tiempo de ejecución de este procedimiento es $O(n+m)$ pasos.

VI.5.2. Procedimiento ETIQUETADO-INICIAL()

Este procedimiento se define en la Sección IV.3.2. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.5.3. Procedimiento CONTEO-VOTOS()

Este procedimiento se define en la Sección IV.3.3. El tiempo de ejecución de este algoritmo es $O(n)$ pasos.

VI.5.4. Procedimiento EXISTE-EQUILIBRIO()

Este procedimiento se define en la Sección IV.3.4. El tiempo de ejecución de este procedimiento es $O(n+m)$ pasos.

VI.5.5. Procedimiento CALENDARIZADOR()

Este procedimiento se define en la Sección IV.3.5. El tiempo de ejecución de este procedimiento es $O(n+m)$ pasos.

VI.5.6. Procedimiento MOVIMIENTO-OPTIMIZACION()

Este procedimiento se define en la Sección IV.3.6. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.5.7. Procedimiento ACTUALIZACION()

Este procedimiento se define en la Sección IV.3.7. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.5.8. Procedimiento ACTUALIZACION-EXHAUSTIVA()

Este procedimiento se define en la Sección V.3.8. El tiempo de ejecución de este procedimiento es $O(n+m)$ pasos.

VI.5.9. Procedimiento PUEDE-ESPECULAR()

Este procedimiento se define en la Sección V.3.9. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.5.10. Procedimiento MOVIMIENTO-ESPECULACION()

Este procedimiento se define en la Sección V.3.10. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.5.11. Procedimiento ACTUALIZACION-EXTENDIDA()

Este procedimiento se define en la Sección V.3.11. El tiempo de ejecución de este procedimiento es $O(n+m)$ pasos.

VI.5.12. Procedimiento MOVIMIENTO-VORAZ()

Este procedimiento se define en la Sección V.3.12. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.5.13. Procedimiento PUEDE-ESPECULAR-CERO()

Este procedimiento se define en la Sección V.3.9. La diferencia de este procedimiento con el procedimiento de la Sección V.3.9 es que este nuevo procedimiento utiliza las reglas de la técnica de especulación cero para calcular si puede realizar un movimiento de especulación o no. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.5.14. Procedimiento MOVIMIENTO-ESPECULACION-CERO()

Este procedimiento se define en la Sección V.3.10. La diferencia de este procedimiento con el procedimiento de la Sección V.3.10 es que este nuevo procedimiento utiliza las reglas de la técnica de especulación cero para seleccionar la nueva conexión. El tiempo de ejecución de este procedimiento es $O(n)$ pasos.

VI.6. Análisis del algoritmo CBAP$^{\times\times}$

En esta sección se demuestra que el algoritmo CBAP$^{\times\times}$ obtiene una configuración de equilibrio con igual o mayor beneficio social que el CBAP$^{\times}$ y en el mismo orden de

tiempo de ejecución que el algoritmo CBAP.

Sea $T = (V_T, E_T)$ un árbol de esparcimiento generado en un grafo arbitrario G, a través de la ejecución de los pasos 1-3 del Pseudo-código 5.

Lema 11. *El algoritmo CBAP$^{\times\times}$ siempre termina con un árbol de esparcimiento con una configuración en equilibrio.*

Demostración. Se demuestra por contradicción. Suponga que el algoritmo CBAP$^{\times\times}$ termina y que el árbol T resultante no tiene una configuración de equilibrio. Dado que T no tiene una configuración de equilibrio, al menos un nodo u puede incrementar su ganancia individual, debido a esto, la condicion del ciclo en el paso 5, ExisteEquilibrio()=$falso$ (ver Pseudo-código 5) aún se cumple. Por lo tanto, el algoritmo aún no termina y existe una contradicción. □

Lema 12. *El algoritmo CBAP$^{\times\times}$ siempre termina con una configuración con igual o mayor beneficio social que el algoritmo CBAP$^{\times}$.*

Demostración. El algoritmo CBAP$^{\times}$ termina su ejecución cuando encuentra un equilibrio en el que no existe un nodo que puede realizar un movimiento de especulación; mientras que el algoritmo CBAP$^{\times\times}$ aplica la técnica de especulación extendida con inversión cero una vez que llega a un equilibrio donde no puede aplicar la técnica de especulación extendida. Dado que este proceso se repite de forma iterativa hasta llegar a un equilibrio en el que ningún nodo puede realizar movimientos de especulación cero y que por la Ecuación 15 se garantiza que no existe una pérdida en el beneficio social (el movimiento de optimización posterior al movimiento de especulación con inversión cero siempre recupera la pérdida), entonces el beneficio social mínimo que puede entregar el algoritmo CBAP$^{\times\times}$ es el beneficio social de la configuración que entrega el algoritmo CBAP$^{\times}$; como los movimientos de especulación extendida con inversión cero

realizados perturban el equilibrio; entonces, cualquier movimiento de optimización adicional al movimiento que recupera la pérdida siempre incrementan el beneficio social; así, cuando el algoritmo vuelve a entrar en el ciclo de movimientos de optimización (ver Pseudo-código 5, línea 6) después de un movimiento de especulación con inversión cero, entonces la configuración con el CBAP$^{\times\times}$ tiene un beneficio social de mayor valor que el del algoritmo CBAP$^{\times}$. □

A continuación se demuestra que el algoritmo CBAP$^{\times\times}$ tiene un tiempo de ejecución del mismo orden que el algoritmo CBAP.

Conjetura 2. *El algoritmo CBAP$^{\times\times}$ realiza $O(n^2)$ movimientos de especulación con inversión cero.*

La idea de esta conjetura se basa en el siguiente razonamiento. Suponga que existe un grafo arbitrario en donde se encuentra un árbol que cumple con una condición de equilibrio para el PAG y que ningún nodo puede realizar un movimiento de especulación. Suponga además que existe un sub-árbol *gris* enraizado en el nodo u, en donde u puede realizar un movimiento de especulación con inversión cero; es decir, un movimiento donde el sub-árbol de u no gana ni pierde. Dado que todos los posibles movimientos de u son de inversión cero, entonces u rompe el equilibrio a través de un movimiento que maximice su función de utilidad individual (definida por las Ecuaciones 2 y 3). La distancia de u después de realizar el movimiento de especulación con inversión cero tiene una diferencia de $\Delta_{d_u^T}$ con respecto a la distancia original. Así, si u es negro, entonces $d < d'$ y cualquier otro futuro movimiento de especulación con inversión cero también. En el peor escenario, el nodo u está conectado directamente a la raíz por lo que el número máximo de movimientos de especulación con inversión cero que puede realizar u es de $O(n)$. Si todos los nodos negros se comportan de esta forma entonces

se realizan $O(n^2)$ movimientos de especulación por inversión cero. Este razonamiento es similar para los nodos blancos.

Corolario 4. *El algoritmo CBAP$^{\times\times}$ realiza $O(n^2)$ movimientos.*

Demostración. Los movimientos que puede realizar el algoritmo CBAP$^{\times\times}$ son movimientos de optimización, de especulación y de especulación con inversión cero. Por el Corolario 2 (ver Capítulo V) se sabe que el algoritmo realiza $O(n^2)$ movimientos de optimización y de especulación. Por la Conjetura 2 se tiene que el algoritmo CBAP$^{\times\times}$ realiza $O(n^2)$ movimientos de especulación con inversión cero. Por lo tanto, el número de movimientos que puede realizar el algoritmo CBAP$^{\times\times}$ es de $O(n^2)$ movimientos. □

Note que cada movimiento de optimización, especulación y/o especulación con inversión cero requiere de $O(n+m)$ pasos para completarse. Además, note que el tiempo adicional que requiere el algoritmo para realizar la actualización extendida de los nodos es de $O(n+m)$ pasos. Por lo tanto, el tiempo total que lleva hacer todos los movimientos es de $O(n^3+n^2m)$. El algoritmo CBAP$^{\times\times}$ termina cuando el grafo tiene una condición de equilibrio y no hay un nodo que pueda realizar un movimiento de especulación con inversión cero. El tiempo que tarda el algoritmo CBAP$^{\times\times}$ es $O(n^3+n^2m)$, que es el mismo orden del tiempo de ejecución del algoritmo CBAP propuesto por Fajardo-Delgado *et al.* (2010) bajo el enfoque cooperativo.

Capítulo VII

Resultados experimentales

En los capítulos III-VI se proponen distintas variantes para el juego del PAG y el algoritmo CBAP; estas variantes buscan encontrar equilibrios con mejor beneficio social que el entregado por el algoritmo CBAP utilizando el juego del PAG propuesto en (Fajardo-Delgado *et al.*, 2010). Sin embargo, no se puede demostrar que las variantes propuestas siempre dan configuraciones de equilibrio con mejor beneficio social (lo que sí se demuestra es que nunca dan uno peor). Por ello, se realizaron simulaciones en grafos generados de forma aleatoria para comparar los resultados de las variantes con los resultados obtenidos por el algoritmo CBAP cuando utiliza el juego del PAG original.

Se realizaron simulaciones en 450 grafos distintos de 65 nodos cada uno; para poder comparar, se utilizaron los mismos grafos que los utilizados por Fajardo-Delgado *et al.* (2010), de la misma forma, para que sea comparable el tiempo, las simulaciones se realizaron en una máquina con las mismas características que la utilizada en (Fajardo-

Tabla II. Caracterización de la máquina utilizada para las simulaciones.

Componente	Valor
Sistema Operativo	CentOS
Arquitectura	64 bits
Tipo de CPU	Intel(R) Xeon(R) CPU L7455 @2.13GHz
Número de CPU's	11
Memoria RAM	32 Gb

Delgado *et al.*, 2010); las características de la máquina se muestran en la Tabla II. Los grafos se generaron de forma aleatoria utilizando tres modelos de generación de grafos aleatorios distintos: Barabási-Albert, Erdős-Renyi y grafos crecidos aleatoriamente; por cada modelo se generaron 150 grafos distintos. De los grafos generados por cada modelo, se generaron grafos de cinco densidades distintas: 0.06, 0.12, 0.24, 0.49 y 0.98; donde la densidad de un grafo define la cantidad de aristas en el grafo. La Ecuación 16 define la densidad ρ para un grafo $G = (V, E)$. Así, por cada modelo, se generaron 30 grafos por cada densidad. En la Sección VII.1 se detalla el proceso de creación de los 450 grafos.

$$\rho(G) = \frac{|E|}{|V|(|V|-1)/2} \tag{16}$$

La metodología de los experimentos es la siguiente; por cada grafo, se realizan 30 simulaciones. Para determinar si los resultados de una configuración son mejores a los resultados de otra configuración, se juntan los resultados de todos los grafos de una densidad específica y se aplican las siguientes pruebas estadísticas:

1. Se aplica la prueba lilliefors para determinar si las muestras a comparar provienen de distribuciones normales.

2. Dependiendo de si es una distribución normal o no, se revisa si existe una diferencia significativa entre las muestras.

    - Si las dos muestras tienen una distribución normal, se aplica la prueba t-student.

    - Si al menos una de las muestras no tiene una distribución normal, se aplica la prueba de Wilcoxon para una y dos colas.

Para todas las pruebas se utilizó un nivel de confianza de 0.05.

En los experimentos también se obtiene el beneficio social promedio, número de movimientos promedio y el tiempo promedio en milisegundos que tardan las simulaciones en encontrar una configuración en equilibrio. También, para cada densidad se obtienen el *precio de la estabilidad* (PoS) y el *precio de la anarquía* (PoA). Donde el PoS representa la calidad del mejor equilibrio encontrado por el algoritmo y el PoA la calidad del peor equilibrio encontrado por el algoritmo. Las Ecuaciones 17 y 18 definen el PoS y PoA, respectivamente.

$$\text{PoS} = \frac{\text{beneficio social máximo}}{\text{beneficio social óptimo}} \tag{17}$$

$$\text{PoA} = \frac{\text{beneficio social mínimo}}{\text{beneficio social óptimo}} \tag{18}$$

Donde el beneficio social óptimo es igual a 3014 ya que es el valor máximo del beneficio social que pueden tener los grafos de 65 nodos. Por lo tanto, este valor se utiliza como el valor del mejor equilibrio posible.

En total, se realizaron cuatro tipos de experimentos distintos que buscan determinar distintos objetivos:

- ¿Qué método para generar un árbol de esparcimiento inicial da mejores resultados en el algoritmo CBAP?
- ¿Cuál es la mejor combinación de estrategia-calendarizador para el algoritmo CBAP?.
- ¿Cuánto mejora aplicar las técnicas de especulación y cuántos movimientos de especulación ocurren?
- ¿Cuál de los algoritmos CBAP, CBAP^{++} o CBAP$^{\times\times}$ da mejores resultados?

VII.1. Generación de grafos aleatorios

Para poder observar cómo se comportan las variantes propuestas del juego del PAG, se generaron 450 grafos aleatorios para realizar simulaciones y obtener estadísticas sobre los comportamientos de los algoritmos con distintas configuraciones. Los grafos se generaron utilizando el software *R* y el paquete *igraph* para el manejo de grafos. Cada grafo se guardó en formato *Graph Modelling Language* (GML) para así poder hacer distintas simulaciones y repetir experimentos con la misma entrada. Para generar los 450 grafos se utilizaron 3 modelos de generación de grafos aleatorios distintos (150 por modelo).

VII.1.1. Modelo Barabási-Albert

El modelo Barabási-Albert pretende modelar el crecimiento de los sistemas en el mundo real y utiliza el concepto de conexión preferencial; es decir, los nodos que se van conectando al grafo tienen una mayor tendencia a conectarse a los nodos de mayor grado en el grafo. El proceso comienza con un grafo aleatorio con $m_0 \geq m$ nodos. Los nuevos nodos se van añadiendo de uno a uno. Cada nodo es conectado a m nodos distintos ya existentes (Barabási y Albert, 1999). La librería *igraph* presenta una variante donde se comienza con un solo nodo sin conexiones y se van agregando los nuevos nodos de uno a uno.

El concepto de conexión preferencial se incorpora al modelo al momento de seleccionar los nodos a los que se va a conectar el nodo v. La Ecuación 19 define la probabilidad que da el nuevo nodo a conectarse con el nodo v.

$$\Pr(v) = \frac{k_v}{\sum_j k_j} + zappeal \tag{19}$$

Donde k_v es el número de aristas de entrada del nodo v; es decir, no se toman en cuenta las aristas que usó el nodo u al conectarse a los m nodos. La variable *zappeal* es un valor constante para dar probabilidad de conexión a los nodos que no tienen aristas de entrada; es decir, los nodos a los que ningún nodo se ha conectado. Note que los nodos recién incorporados al grafo no tienen aristas de entrada.

Los grafos generados con este modelo tienen la propiedad de que existen pocos nodos con un alto grado de conectividad y el resto de los nodos tiene un grado bajo de conectividad. Así, el modelo Barabási-Albert se caracteriza por generar grafos con forma de estrellas conectadas. La Figura 11 muestra un grafo generado por este método de 65 nodos con 128 aristas.

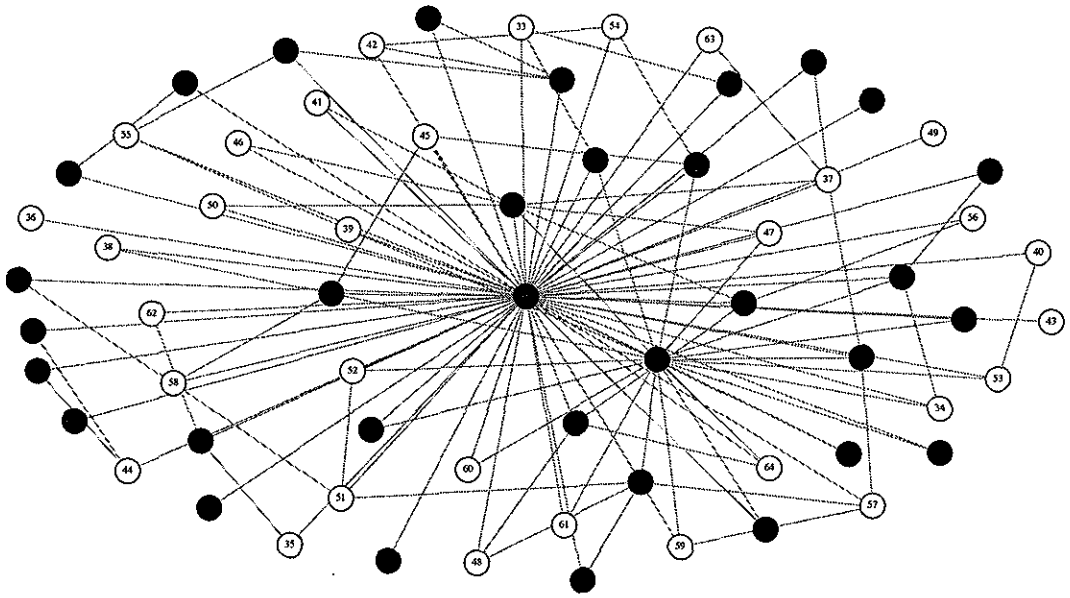

Figura 11. Grafo aleatorio de 65 nodos y 128 aristas generado por el modelo Barabási-Albert.

VII.1.2. Modelo Erdős-Rényi

El modelo Erdős-Rényi es un modelo sencillo para generar árboles aleatorios, en este modelo se tiene un grafo de n nodos y de todo el conjunto de aristas posibles, se

van seleccionando aleatoriamente m aristas sin reemplazo (Erdős y Rényi, 1960). La Figura 12 muestra un grafo aleatorio de 65 nodos y 128 aristas generado por el modelo de Erdős-Rényi.

VII.1.3. Modelo grafos crecidos aleatoriamente

El modelo de grafos crecidos aleatoriamente simula el crecimiento de una red. El proceso comienza con un grafo aleatorio con $m_0 \geq m$ nodos. En cada iteración se agrega un nodo y m aristas nuevas. Las nuevas aristas se seleccionan entre pares de nodos seleccionados aleatoriamente (Callaway *et al.*, 2001). En igraph se tiene una variante en la que el nodo añadido es el que selecciona conexiones con m nodos de forma aleatoria. Esta variante fue la utilizada para crear los grafos con este modelo, ya que la versión original permite que existan nodos aislados. La Figura 13 muestra un grafo aleatorio de 65 nodos y 128 aristas generado por el modelo de grafos crecidos aleatoriamente.

VII.2. Método de construcción de árbol inicial

En el procedimiento CREAR-ARBOL() del algoritmo CBAP$^+$ (ver Capítulo IV) se menciona que se pueden utilizar distintos algoritmos para crear el árbol de esparcimiento a optimizar. En Fajardo-Delgado *et al.* (2010) realizan simulaciones para ver el comportamiento del algoritmo CBAP utilizando el método Prim Random Spanning Tree (PrimRST) como método de generación de árbol de esparcimiento a optimizar. Así, en este experimento se busca observar el desempeño del algoritmo CBAP para el enfoque cooperativo cuando se utilizan los siguientes algoritmos para generar el árbol de esparcimiento a optimizar: PrimRST, Búsqueda por anchura (BFS) y Búsqueda por

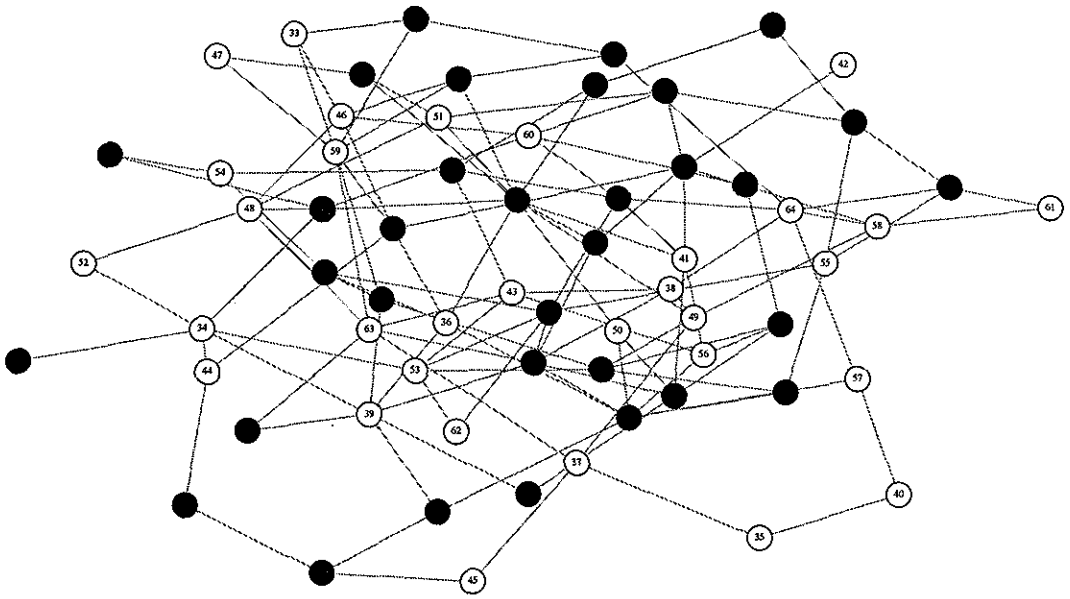

Figura 12. Grafo aleatorio de 65 nodos y 128 aristas generado por el modelo Erdős-Rényi.

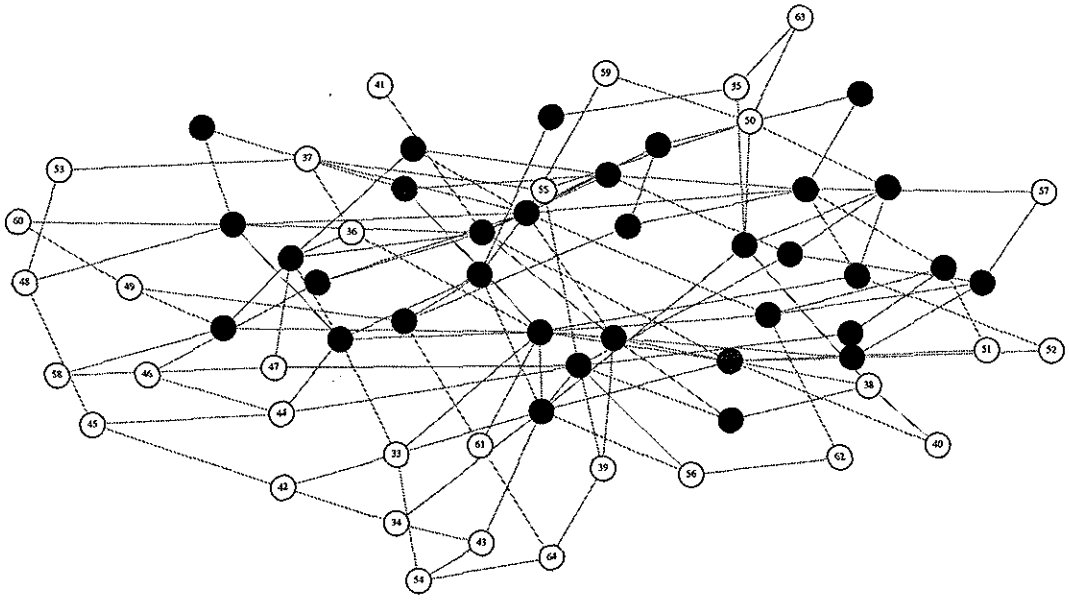

Figura 13. Grafo aleatorio de 65 nodos y 128 aristas generado por el modelo de grafos crecidos aleatoriamente.

profundidad (DFS).

La solución que entrega el algoritmo CBAP no depende únicamente del método para crear árboles de esparcimiento utilizado para generar la solución a optimizar, la estrategia y el calendarizador utilizados influyen en los resultados del algoritmo. Por lo tanto, se experimentó con distintas estrategias y calendarizadores para evitar que se probara con una combinación que favorece a un método de generación de árbol de esparcimiento. En el experimento, se utilizaron las estrategias: Voraz, Anti-voraz e IA-voraz; cada estrategia se probó con los calendarizadores: Voraz, Anti-voraz y Uniforme. Por lo tanto se experimentó por cada método para crear el árbol de esparcimiento a optimizar con cada estrategia y cada calendarizador mencionado.

Resultados

La metodología utilizada para analizar los resultados es la siguiente:

1. Por cada estrategia, se evalúa qué calendarizador es el que da mejores resultados.

2. Por cada método para generar el árbol de esparcimiento a optimizar, se evalúa qué combinación (estrategia-calendarizador) es la que da mejores resultados.

El resumen de los resultados de los experimentos se muestra en las tablas III-V. Donde las tablas muestran para cada estrategia, cuál es la configuración (algoritmo para generar árbol de esparcimiento - calendarizador) que dan los mejores resultados para las distintas densidades de los grafos.

Al observar las tablas III-V se puede ver que se obtienen mejores resultados con los métodos BFS y DFS en grafos con una densidad $\rho(G) = 0.06$; se conjetura que en este tipo de grafos, los movimientos que pueden realizar los nodos son pocos, por lo que el utilizar un método aleatorio puede dar malas configuraciones y los nodos no pueden

realizar suficientes movimientos de optimización para dar soluciones con una calidad igual que la que se entrega al utilizar los métodos determinísticos. Las tablas también muestran que en general, el método PrimRST es el que da mejores resultados para grafos con una densidad $\rho(G) \geq 0.12$. Esto se debe a que al utilizar un método aleatorio, los árboles de esparcimiento generados no van a ser siempre una buena solución; sin embargo, los árboles de esparcimiento generados por este método permite que los nodos tengan más movimientos de optimización antes de llegar a un equilibrio y por lo tanto puedan llegar a una solución de mayor calidad que las soluciones al utilizar los métodos BFS y DFS. Sin embargo, utilizar un método aleatorio también puede generar árboles de esparcimiento con una buena solución inicial.

Un resultado interesante es que comparada con las otras estrategias, la estrategia IA-voraz mejora los resultados cuando se utilizan los métodos BFS y DFS; sin embargo, no es suficiente para alcanzar la calidad de las soluciones obtenidas por el algoritmo al utilizar el método PrimRST en grafos con densidad $\rho(G) \geq 0.12$.

Tabla III: Resultados en modelo Barabási-Albert.

Densidad	Voraz	Anti-voraz	IA-voraz
0.06	DFS-Uniforme	DFS-Uniforme	DFS-Uniforme
0.12	DFS-Uniforme	DFS-Uniforme	DFS-Uniforme
0.24	PrimRST-Uniforme	PrimRST-Uniforme	PrimRST-Uniforme
0.49	PrimRST-Anti voraz	PrimRST-Anti voraz	PrimRST-Anti voraz
0.98	PrimRST-Anti voraz	PrimRST-Anti voraz	PrimRST-Anti voraz

Tabla IV. Resultados modelo Erdős-Renyi.

Densidad	Voraz	Anti-voraz	IA-voraz
0.06	DFS-Anti voraz	PrimRST-Uniforme	PrimRST-Uniforme
0.12	PrimRST-Uniforme	PrimRST-Anti voraz	PrimRST-Uniforme
0.24	PrimRST-Anti voraz	PrimRST-Anti voraz	PrimRST-Anti voraz
0.49	PrimRST-Anti voraz	PrimRST-Anti voraz	PrimRST-Anti voraz
0.98	PrimRST-Anti voraz	PrimRST-Anti voraz	PrimRST-Anti voraz

Tabla V. Resultados modelo grafos crecidos aleatoriamente.

Densidad	Voraz	Anti-voraz	IA-voraz
0.06	DFS-Uniforme	DFS-Uniforme	DFS-Uniforme
0.12	PrimRST-Uniforme	PrimRST-Uniforme	PrimRST-Uniforme
0.24	PrimRST-Anti voraz	PrimRST-Anti voraz	PrimRST-Anti voraz
0.49	PrimRST-Anti voraz	PrimRST-Anti voraz	PrimRST-Anti voraz
0.98	PrimRST-Anti voraz	PrimRST-Anti voraz	PrimRST-Anti voraz

VII.3. Estrategia y calendarizador

En la Sección VII.2 se menciona que la solución que entrega el algoritmo CBAP no depende completamente del método con el que se genera el árbol de esparcimiento a optimizar; depende de la estrategia utilizada por los nodos y el algoritmo utilizado por el calendarizador central para determinar qué nodo puede realizar un movimiento. Así, en este experimento se simulan las estrategias y funciones jugador (calendarizador) propuestas en el Capítulo III para determinar qué configuración (estrategia-calendarizador) es la que permite que el algoritmo CBAP encuentre equilibrios de mayor calidad para

los dos enfoques (no cooperativo y cooperativo)

Las configuraciones evaluadas con el algoritmo CBAP en este experimento son las siguientes:

- Anti-voraz: Esta configuración está compuesta por la estrategia Anti-voraz y un calendarizador anti-voraz.
- ε-voraz: Esta configuración está compuesta por la estrategia ε-voraz y un calendarizador uniforme.
- Voraz-I: Esta configuración está compuesta por la estrategia Voraz y un calendarizador uniforme. Los resultados utilizados de esta configuración son los resultados obtenidos en (Fajardo-Delgado *et al.*, 2010).
- Voraz-II: Esta configuración está compuesta por la estrategia Voraz y un calendarizador voraz. Los resultados utilizados de esta configuración son los resultados obtenidos en (Fajardo-Delgado *et al.*, 2010).

En todas las configuraciones se utiliza el método PrimRST para generar el árbol de esparcimiento a optimizar.

Note que la estrategia ε-voraz permite tener distintos niveles de voracidad dependiendo del valor de ε (ver Capítulo III). Debido a esto, se realizó un torneo para comparar qué nivel de voracidad es el que da mejores resultados para la configuración ε-voraz antes de compararla con las demás configuraciones. Se experimentó con los siguientes niveles de voracidad: 0.20, 0.50 y 0.80; en los resultados obtenidos de los torneos para ambos enfoques se obtuvo que cuando $\varepsilon = 0.50$ se obtienen resultados mejores o iguales al mejor de los otros dos niveles de voracidad. Por lo tanto, en las pruebas entre configuraciones se le da el valor de $\varepsilon = 0.50$ para ambos enfoques.

Resultados para el enfoque no cooperativo

Las figuras 14-16 muestran los resultados obtenidos de la comparación de configuraciones para el enfoque no cooperativo, donde las figuras muestran los promedios para el beneficio social, el número de movimientos de optimización realizados por los nodos y el tiempo en milisegundos que tarda el algoritmo en encontrar un equilibrio para las configuraciones evaluadas.

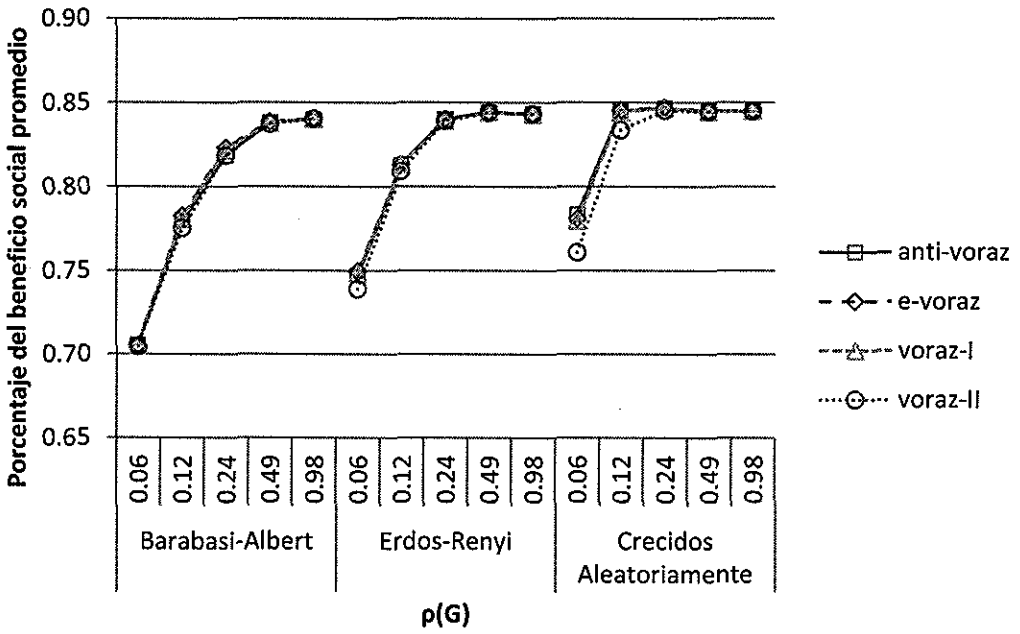

Figura 14. Beneficio social promedio del algoritmo CBAP para el enfoque no cooperativo.

Al observar la Figura 14 se puede ver que la calidad de las soluciones de todas las configuraciones tienen en promedio un valor similar; sin embargo, al realizar el análisis estadístico se obtuvo que sí existe una diferencia significativa entre las configuraciones. De las configuraciones evaluadas, la estrategia que en promedio da mejores resultados es la configuración ε-voraz. Se conjetura que esto se debe a que en los grafos poco densos ($\rho \leq 0.12$) el número de opciones de conexión de los nodos son pocos; así, un comportamiento voraz en los nodos lleva a mejores resultados. La propiedad que hace

que la configuración ε-voraz dé mejores resultados es que permite que los nodos sigan un comportamiento voraz; sin embargo, también permite en ciertas ocasiones que exploren opciones que no son las óptimas para ese momento específico, logrando soluciones con mejor beneficio social. La figura también muestra que conforme se hace más denso el grafo, se reduce la diferencia en la calidad del beneficio social entre las configuraciones; esto se debe a que en el enfoque no cooperativo, el número de equilibrios disminuye conforme incrementa el número de aristas.

Las figuras 15 y 16 muestran que el nivel de voracidad en los nodos está relacionado directamente con el número de movimientos y el tiempo. La configuración Anti-voraz es la que da la mayor cantidad de movimientos, mientras que la configuración voraz-II da la menor cantidad de movimientos.

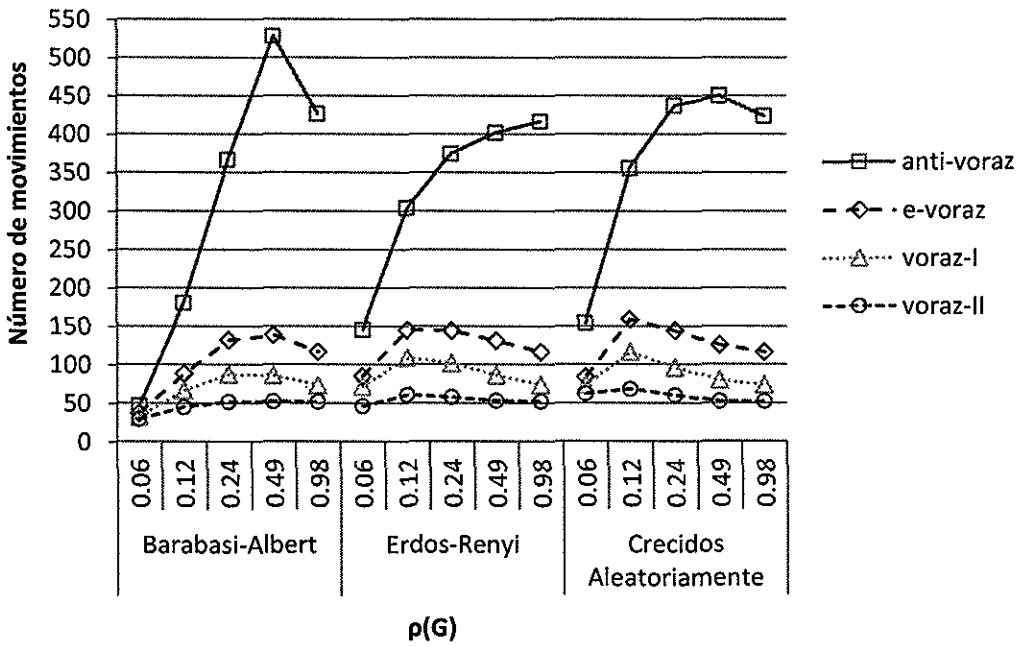

Figura 15. Número de movimientos promedio del algoritmo CBAP para el enfoque no cooperativo.

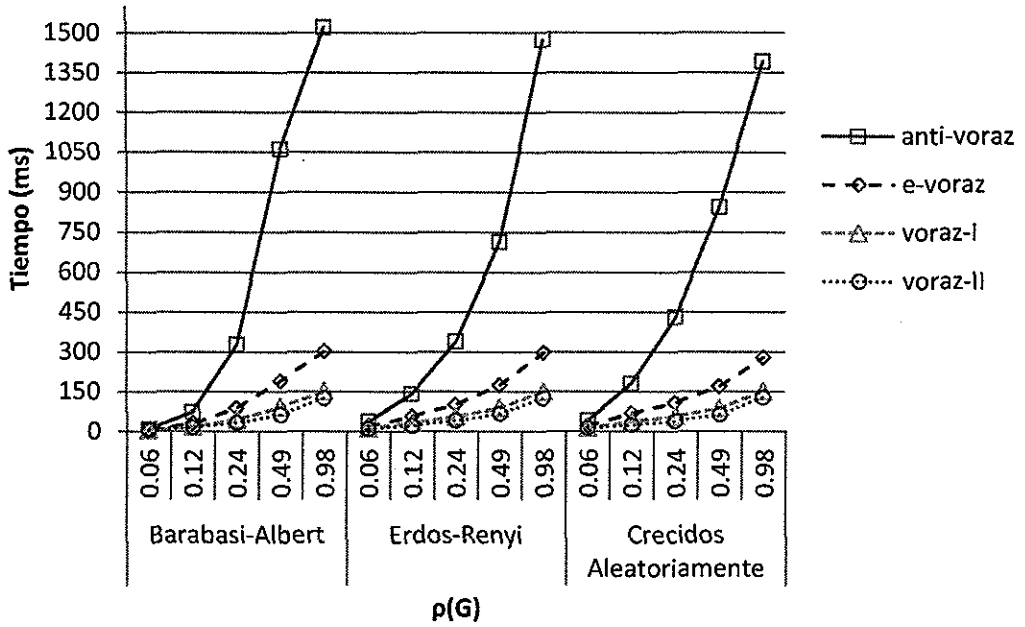

Figura 16. Tiempo promedio del algoritmo CBAP para el enfoque no cooperativo.

En promedio, el enfoque no cooperativo no depende tanto de la estrategia o del calendarizador; sin embargo, el número de movimientos que realizan los nodos y el tiempo que tarda el algoritmo sí lo es.

Resultados para el enfoque cooperativo

Las figuras 17-19 muestran los resultados obtenidos en la comparación de configuraciones para el enfoque cooperativo, donde las figuras muestran los promedios para el beneficio social, el número de movimientos de optimización realizados por los nodos y el tiempo en milisegundos que tarda el algoritmo en encontrar un equilibrio para las configuraciones evaluadas.

Al observar la Figura 17 se puede ver que de las configuraciones evaluadas, la configuración que en promedio da mejores resultados para grafos poco densos ($\rho(G) \leq 0.12$) es la configuración ε-voraz; se conjetura que esto se debe a que los nodos pueden realizar

pocos movimientos ya que tienen pocas conexiones, por lo que conviene que los nodos tengan un comportamiento voraz; sin embargo, la configuración ε-voraz permite que algunas veces el nodo deje de comportarse de forma voraz y explore algunas conexiones que de momento no son óptimas pero que pueden llegar a serlo en el futuro, dando oportunidad a formar configuraciones con mejor beneficio social. Cuando el grafo se hace más denso ($\rho(G) \geq 0.24$) la combinación que en promedio da mejores resultados es la Anti-voraz; se conjetura que esto se debe a que la configuración Anti-voraz permite realizar más movimientos de forma que se van explorando más opciones antes de llegar a un equilibrio. Al igual que en el enfoque no cooperativo, en la Figura 17 parece que la calidad del beneficio social promedio de las distintas configuraciones es muy similar; sin embargo, al aplicar el análisis estadístico se obtuvo que sí existe una diferencia significativa entre algunas de las configuraciones. Este resultado se aprecia con más facilidad conforme aumenta la densidad del grafo, donde la configuración Anti-voraz

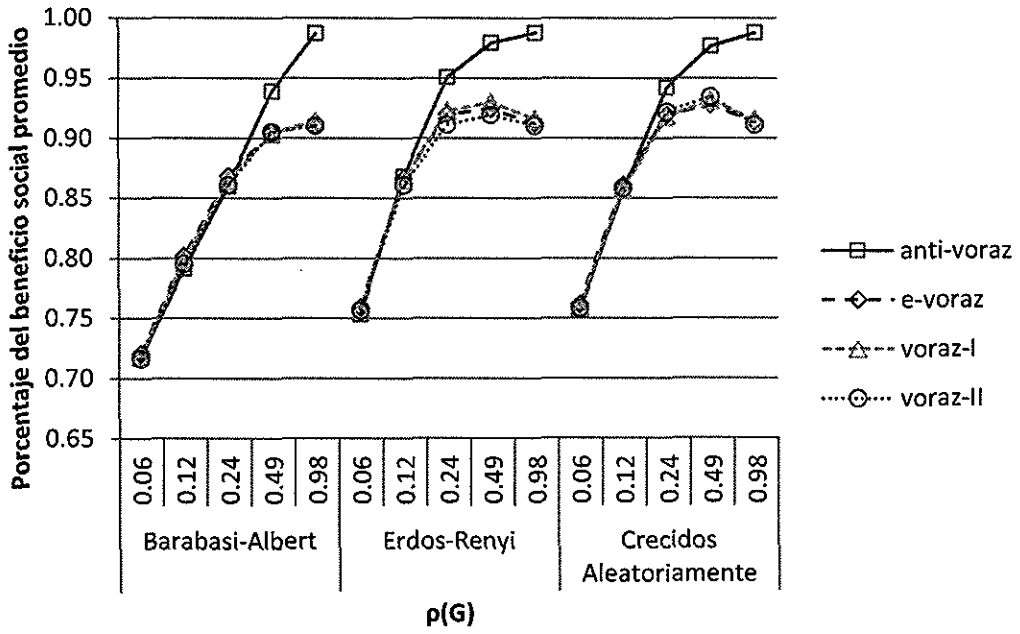

Figura 17. Beneficio social promedio del algoritmo CBAP para el enfoque cooperativo.

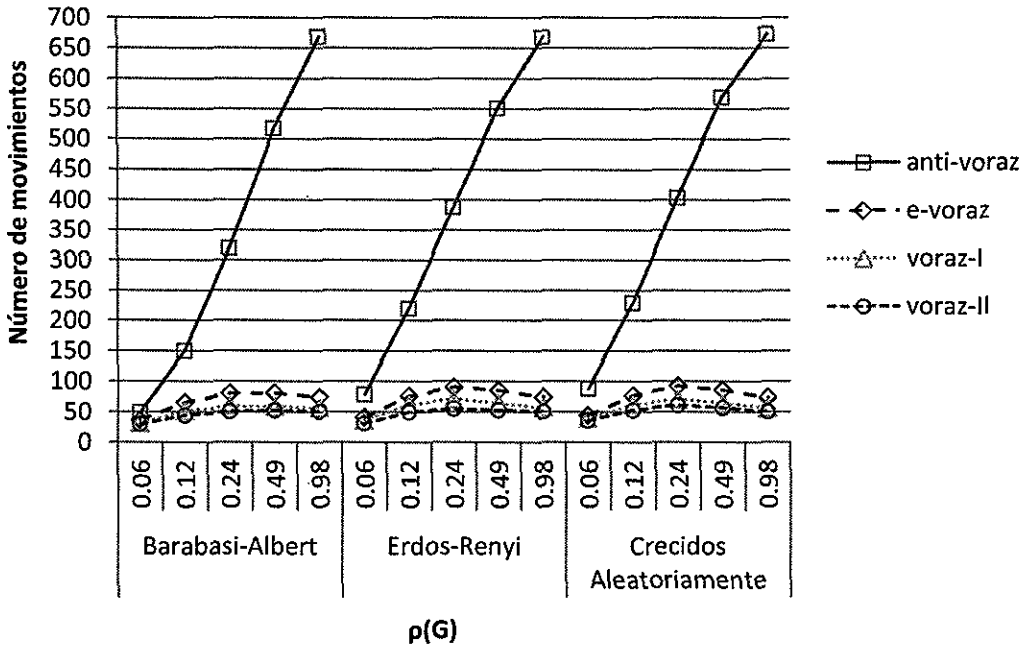

Figura 18. Número de movimientos promedio del algoritmo CBAP para el enfoque cooperativo.

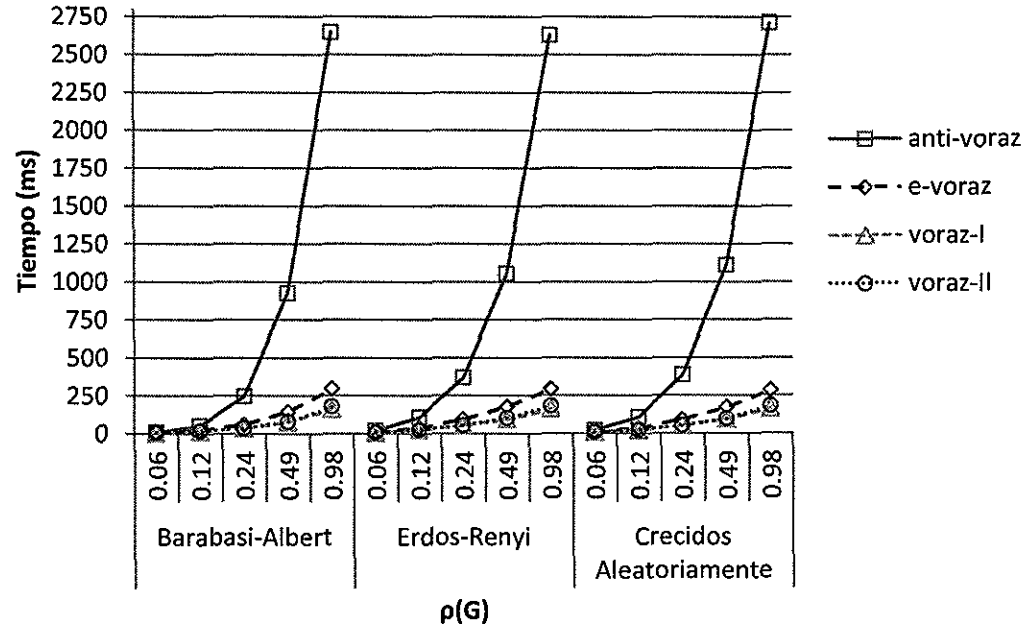

Figura 19. Tiempo promedio del algoritmo CBAP para el enfoque cooperativo.

comienza a dejar atrás al resto de las configuraciones al punto que se puede apreciar con la Figura 17. Un resultado interesante fue que la configuración Anti-voraz fue la única configuración que alcanzó a llegar a una solución con el beneficio social óptimo para grafos completos en algunos grafos con densidad $\rho(G) = 0.98$.

La Figura 18 muestra que al igual que en el enfoque no cooperativo, el nivel de voracidad en los nodos está relacionado directamente con el número de movimientos; es decir, la configuración Anti-voraz es la que da la mayor cantidad de movimientos y la configuración Voraz-II es la que da el menor número de movimientos. Sin embargo, en la Figura 19 se observa que no es igual para el tiempo que tardan los algoritmos; es decir, en promedio, la configuración Voraz-I tarda menos que la configuración Voraz-II; sin embargo, la configuración Voraz-II es la que tiene menos movimientos. Se conjetura que esto se debe a que es más pequeño el tiempo que le toma a la configuración Voraz-I realizar los movimientos adicionales que el tiempo que tarda el calendarizador de la configuración Voraz-II en determinar qué nodo va a realizar el siguiente movimiento.

VII.4. Técnica de especulación

El algoritmo CBAP$^+$ (ver Capítulo IV) garantiza terminar con soluciones con un beneficio social mayor o igual que el beneficio social de las soluciones que entrega el algoritmo CBAP. Sin embargo, para que las soluciones tengan un beneficio social mayor que el del algoritmo CBAP es necesario que se realicen movimientos de especulación. En este experimento se busca determinar qué tan frecuente es que se den configuraciones en equilibrio que permitan realizar movimientos de especulación y cuánto es lo que mejora el algoritmo CBAP$^+$ al realizar movimientos de especulación. Al final del experimento se vuelve a evaluar los resultados de las configuraciones para apreciar si la técnica de

especulación permite que una configuración supere a otras.

En este experimento, el algoritmo CBAP$^+$ utiliza las configuraciones descritas en la Sección VII.3.

Resultados

En la Tabla VI se observa que para todas las configuraciones y en todos los modelos se realizaron movimientos de especulación y por tanto el algoritmo CBAP$^+$ mejora en promedio los resultados del algoritmo CBAP. También se observa que el número de movimientos de especulación disminuye conforme el grafo se va haciendo más denso (note que cuando $\rho = 0.98$ el número de movimientos de especulación promedio es muy cercano a cero); este resultado se debe a que el equilibrio óptimo para un grafo completo para el enfoque cooperativo es una cadena de nodos blancos conectados a

Tabla VI. Movimientos de especulación promedio del algoritmo CBAP$^+$.

		Estrategia			
Modelo	$\rho(G)$	Anti-voraz	ε-voraz	Voraz-I	Voraz-II
Barabási-Albert	0.06	0.90	0.72	0.78	0.94
	0.12	1.65	0.81	0.78	0.96
	0.24	1.56	0.70	0.80	0.88
	0.49	1.11	0.59	0.58	0.66
	0.98	0.02	0.01	0.01	0.01
Erdős-Renyi	0.06	3.16	2.76	2.81	2.85
	0.12	2.38	1.92	1.90	1.88
	0.24	1.56	1.12	1.15	1.13
	0.49	0.89	0.51	0.55	0.54
	0.98	0.03	0.02	0.01	0.01
Crecidos aleatoriamente	0.06	4.26	3.52	3.47	3.83
	0.12	2.91	2.25	2.26	2.52
	0.24	1.85	1.32	1.35	1.32
	0.49	0.89	0.46	0.45	0.48
	0.98	0.00	0.01	0.00	0.01

Tabla VII. Incremento del beneficio social promedio después de la primera configuración en equilibrio encontrada por el algoritmo CBAP$^+$.

		Estrategia			
Modelo	$\rho(G)$	Anti-voraz	ε-voraz	Voraz-I	Voraz-II
Barabási-Albert	0.06	16.13	11.63	14.03	16.79
	0.12	27.07	12.43	12.90	16.15
	0.24	18.97	7.10	9.85	12.29
	0.49	6.52	2.54	2.80	3.76
	0.98	0.02	0.02	0.01	0.02
Erdős-Renyi	0.06	77.75	67.70	70.07	71.35
	0.12	84.15	54.35	60.96	55.60
	0.24	31.48	18.25	20.13	20.78
	0.49	5.80	2.46	2.78	3.47
	0.98	0.03	0.04	0.01	0.02
Crecidos aleatoriamente	0.06	109.27	96.63	97.18	105.35
	0.12	116.26	77.89	85.50	91.39
	0.24	50.31	31.43	35.92	32.45
	0.49	7.91	3.93	3.08	4.66
	0.98	0.00	0.01	0.01	0.00

la raíz, seguida por una cadena de nodos negros; debido a esto, conforme aumenta la densidad del grafo, los equilibrios de las soluciones comienzan a tomar la forma de este equilibrio óptimo. Por lo tanto, es difícil que al realizar el movimiento de especulación, el movimiento calculado por el nodo pueda generar dividendos (ver Capítulo V). Otro resultado que muestra la Tabla VI es que el número de movimientos de especulación es dependiente del modelo con el que fueron generados los grafos aleatorios. Esto se debe a la forma en que trabaja la técnica de especulación; ya que los nodos necesitan buscar entre sus vecinos no descendientes una opción de conexión con la que puedan realizar un movimiento de especulación que genere dividendos; así, entre más grande es el número de vecinos no descendientes, aumenta la probabilidad de tener un nodo con el que se pueda realizar el movimiento. Debido a esto, los grafos generados por el modelo Barabási-Albert permiten pocos movimientos de especulación ya que este modelo genera

Tabla VIII. Incremento en el tiempo promedio después de la primera configuración en equilibrio encontrada por el algoritmo CBAP$^+$.

		Estrategia			
Modelo	$\rho(G)$	Anti-voraz	ε-voraz	Voraz-I	Voraz-II
Barabási-Albert	0.06	0.54	0.46	0.47	0.58
	0.12	1.68	0.95	0.90	1.10
	0.24	3.60	2.06	2.19	2.53
	0.49	7.66	5.26	5.70	5.39
	0.98	5.87	5.82	5.70	5.68
Erdős-Renyi	0.06	4.29	3.29	3.15	3.60
	0.12	8.79	5.60	5.61	5.70
	0.24	10.65	6.39	6.77	6.63
	0.49	10.54	6.54	6.67	6.65
	0.98	5.89	5.93	5.68	5.71
Crecidos aleatoriamente	0.06	5.80	4.08	3.76	4.62
	0.12	12.08	7.08	7.11	7 73
	0.24	17.13	8.98	9.36	8.61
	0.49	12.26	7.18	6.77	6.84
	0.98	5.62	5.70	5.65	5.60

grafos con formas de estrellas enraizadas, por lo que existe una pequeña cantidad de nodos con un grado de conectividad alto, mientras que el grado de conectividad de los otros nodos es muy bajo. La Tabla VI también muestra que la configuración que más se beneficia de utilizar movimientos de especulación es la configuración Anti-voraz; se conjetura que este resultado se debe a que esta configuración da soluciones con un beneficio social más bajo cuando los grafos son poco densos. Por lo tanto, existen más oportunidades de que los nodos realicen movimientos de especulación.

La Tabla VII muestra que el incremento en el beneficio social después de la primera configuración en equilibrio es dependiente de la cantidad de movimientos de especulación. La Tabla VII también muestra que sí existe un incremento en el valor del beneficio social con el que termina el algoritmo CBAP$^+$ con respecto a la primera configuración en equilibrio que encuentra. Sin embargo, como el incremento en el beneficio social es

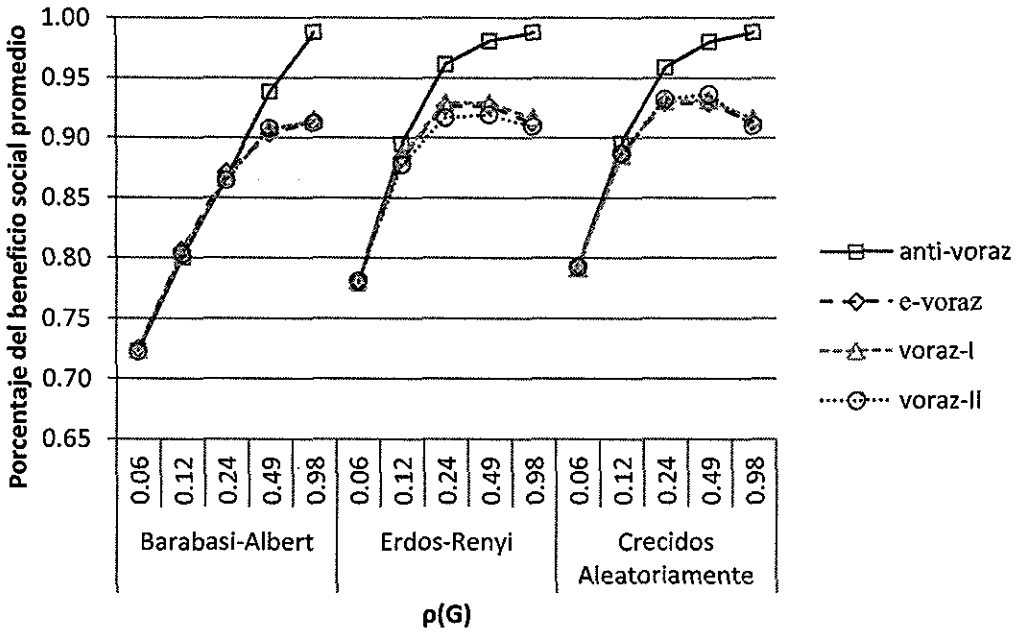

Figura 20. Beneficio social promedio del algoritmo CBAP$^+$.

dependiente del tipo de grafo, la técnica de especulación tiene un mejor desempeño para grafos generados por el modelo de grafos crecidos aleatoriamente.

Al observar la Tabla VIII se puede observar que aplicar la técnica de especulación tiene un incremento poco significativo en el tiempo adicional del algoritmo; por lo tanto, es más rápido ejecutar una vez el algoritmo CBAP$^+$ que ejecutar dos veces el algoritmo CBAP y entregar el mejor resultado.

La comparación de los resultados entre las distintas estrategias se muestran en la Figura 20 donde se muestra el beneficio social promedio que se obtuvieron con las distintas configuraciones. Al igual que en la Tabla VII, la Figura 20 muestra que la configuración Anti-voraz es la que más se beneficia de aplicar la técnica de especulación; esto se debe a que como se muestra en la Tabla VI, los movimientos de especulación son más frecuentes en grafos dispersos que es donde la configuración Anti-voraz tiende a dar soluciones con beneficio social más bajo comparada con las otras configuraciones (ver

Figura 17). Así, la técnica de especulación permite a la configuración Anti-voraz dar soluciones con un beneficio social similar al de las otras estrategias en grafos con una densidad $\rho(G) \leq 0.12$ y continúa dando resultados mejores a las otras configuraciones para grafos con densidad $\rho(G) \geq 0.24$; esto se debe a que en esas densidades comienza a haber pocos movimientos de especulación y por lo tanto las otras configuraciones no pueden alcanzar a la configuración Anti-voraz.

VII.5. Técnica de especulación extendida

En el Capítulo V se dan ejemplos de configuraciones en las que se puede realizar un movimiento de especulación y el cálculo realizado mediante la técnica de especulación indica que el movimiento no va a generar dividendos. Para evitar algunas de estas situaciones, el algoritmo CBAP$^{\times}$ da más información a los nodos cuando calculan si pueden realizar un movimiento de especulación. En este experimento se busca determinar en la práctica que tanto mejora la calidad de las soluciones cuando se calculan los movimientos de especulación con información adicional. Al final del experimento se comparan los resultados de las configuraciones para apreciar si la técnica de especulación extendida permite que una configuración supere a otras.

En este experimento, el algoritmo CBAP$^{\times}$ utiliza las configuraciones utilizadas en la Sección VII.3.

Resultados

En la Tabla IX se observa que en general; para todos los tipos de grafos, densidades del grafo y todas las configuraciones existe un incremento en el número de movimientos de especulación a comparación del número de movimientos de especulación

Tabla IX. Movimientos de especulación promedio del algoritmo CBAP$^{\times}$.

		Estrategia			
Modelo	$\rho(G)$	Anti-voraz	ε-voraz	Voraz-I	Voraz-II
Barabási-Albert	0.06	0.99	0.86	0.84	0.96
	0.12	1.77	1.11	1.16	1.12
	0.24	1.48	1.51	1.44	1.58
	0.49	1.42	5.55	5.20	4.34
	0.98	4.81	16.82	15.53	17.53
Erdős-Renyi	0.06	3.55	3.30	3.40	3.33
	0.12	3.32	3.00	3.03	3.02
	0.24	2.74	4.52	4.23	4.46
	0.49	3.52	9.75	9.06	10.02
	0.98	5.11	16.55	18.14	15.51
Crecidos aleatoriamente	0.06	4.18	3.59	3.76	3.91
	0.12	3.45	3.03	3.25	3.09
	0.24	2.81	4.14	4.10	3.55
	0.49	3.49	8.83	8.28	7.56
	0.98	4.90	16.74	15.72	17.84

de la Tabla VI y por tanto el algoritmo CBAP$^{\times}$ mejora en promedio los resultados del algoritmo CBAP^{+}. Un resultado interesante que muestra la Tabla IX es que a diferencia con el algoritmo CBAP^{+} donde el número de movimientos de especulación disminuye conforme el grafo se va haciendo más denso; el algoritmo CBAP$^{\times}$ incrementa el número de movimientos de especulación promedio conforme aumenta la densidad del grafo. Esto se debe a que conforme se hace denso el grafo, incrementa la cantidad de movimientos de especulación que se pueden realizar; es decir, los nodos pueden evaluar con más vecinos. Sin embargo, como se menciona en la Sección VII.3, conforme se va haciendo más denso el grafo, la solución óptima para el enfoque cooperativo tiende a formar una cadena de nodos blancos conectados a la raíz seguida de una cadena de nodos negros. Debido a esto, es necesario conocer el comportamiento a seguir del nodo que genera dividendos para saber si realmente puede recuperar la pérdida de la cadena de nodos

Tabla X. Incremento del beneficio social promedio después de la primera configuración en equilibrio encontrada por el algoritmo CBAP$^{\times}$.

		Estrategia			
Modelo	$\rho(G)$	Anti-voraz	ε-voraz	Voraz-I	Voraz-II
Barabási-Albert	0.06	15.96	13.89	13.21	16.12
	0.12	33.40	15.55	16.88	18.46
	0.24	19.92	17.06	18.02	19.68
	0.49	10.81	99.59	93.09	77.26
	0.98	32.26	275.88	273.50	260.69
Erdős-Renyi	0.06	81.20	74.17	79.59	81.71
	0.12	104.06	74.51	83.85	80.56
	0.24	47.20	75.96	70.07	79.35
	0.49	30.07	155.41	148.52	164.41
	0.98	39.23	271.94	261.19	281.06
Crecidos aleatoriamente	0.06	115.77	97.69	106.38	107.52
	0.12	127.47	94.74	107.67	107.03
	0.24	69.01	90.53	91.82	77.16
	0.49	38.05	161.61	153.31	132.82
	0.98	38.04	272.16	264.11	276.48

negros al minimizar/maximizar su distancia a r.

La Tabla X muestra que la técnica de especulación extendida beneficia a todas las configuraciones pero favorece a las configuraciones voraces cuando el grafo se hace denso; esto se debe a que la configuración Anti-voraz entrega soluciones muy buenas en las que ya no es posible realizar movimientos de especulación, es decir, las soluciones de la configuración Anti-voraz para grafos densos ya se encuentran cerca del óptimo. Otro resultado interesante se puede ver en la Tabla XI, donde se ve un incremento considerable de tiempo para todas las configuraciones conforme se hace denso el grafo (note que en grafos con densidad $\rho = 0.98$ existen incrementos promedio de 562 milisegundos). Sin embargo, estas configuraciones dan resultados similares a los resultados que se obtienen si se utiliza el algoritmo CBAP con una configuración Anti-voraz y son más rápidas.

En la Figura 21 se muestra el beneficio social promedio de las soluciones que dieron

Tabla XI. Incremento en el tiempo promedio después de la primera configuración en equilibrio encontrada por el algoritmo CBAP$^{\times}$.

		Estrategia			
Modelo	$\rho(G)$	Anti-voraz	ε-voraz	Voraz-I	Voraz-II
Barabási-Albert	0.06	0.67	0.56	0.52	0.65
	0.12	1.90	1.33	1.38	1.42
	0.24	3.61	4.09	3.90	4.36
	0.49	10.26	40.00	37.40	32.29
	0.98	142.43	477.41	442.24	545.03
Erdős-Renyi	0.06	4.72	4.25	4.35	4.76
	0.12	11.38	9.15	9.40	9.84
	0.24	17.81	27.41	24.83	27.61
	0.49	47.83	123.86	112.83	132.31
	0.98	152.04	472.27	443.58	570.09
Crecidos aleatoriamente	0.06	5.27	4.29	4.57	5.04
	0.12	12.70	9.85	10.42	10.39
	0.24	21.38	27.12	26.57	23.00
	0.49	52.31	117.44	106.83	100.63
	0.98	146.61	480.57	452.24	562.64

las distintas configuraciones. A diferencia de la técnica de especulación, en la figura se puede ver que al aplicar la técnica de especulación extendida, la calidad de las soluciones que dan todas las configuraciones tienen en promedio un valor similar. Sin embargo, a pesar de que todas las configuraciones se benefician de aplicar la técnica de especulación extendida; los resultados del análisis estadístico indican que la configuración Anti-voraz sigue dando resultados iguales o mejores a los resultados de las otras configuraciones en la mayoría de los casos. Un resultado interesante es que para grafos con una densidad $\rho = 0.98$, la configuración Anti-voraz dió una solución con el beneficio social óptimo para grafos completos con un porcentaje de 0.80.

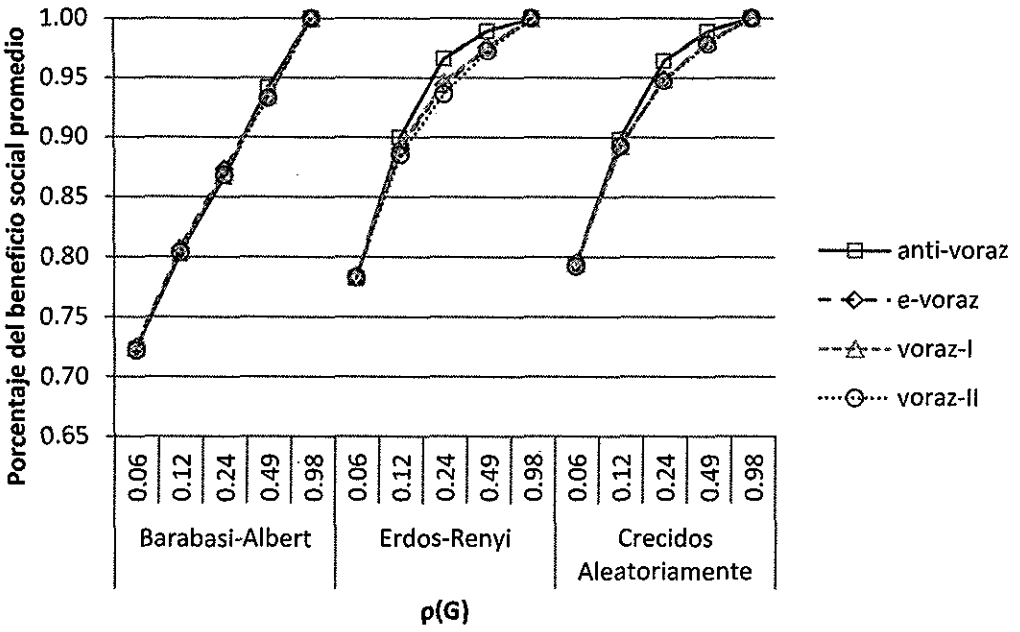

Figura 21. Beneficio social promedio del algoritmo CBAP$^{\times}$.

VII.6. Técnica de especulación con inversión cero

En el Capítulo VI se dan ejemplos de configuraciones en las que se puede realizar un movimiento de especulación con inversión cero en el que el cálculo del nodo indica que no se van a generar dividendos pero que movimientos posteriores de otros nodos pueden generar dividendos cuando se rompe el equilibrio. Para romper el equilibrio y permitir movimientos de optimización en algunas de estas situaciones, los algoritmos CBAP^{++} y CBAP$^{\times\times}$ establecen las reglas para poder realizar movimientos de especulación con inversión cero en los que no existen dividendos. En este experimento se busca determinar qué tan frecuente es encontrar configuraciones en las que se pueda dar movimientos de especulación con inversión cero y cuánto mejora la calidad de las soluciones cuando se aplican estos movimientos.

En este experimento, los algoritmos CBAP^{++} y CBAP$^{\times\times}$ utilizan las configuraciones descritas en la Sección VII.3.

Resultados

Los resultados de la Tabla XII muestran que las configuraciones en las que se pueden realizar movimientos de especulación con inversión cero para el algoritmo CBAP^{++} son pocas. Se conjetura a que el algoritmo realiza todos los movimientos de especulación antes de aplicar los movimientos de especulación cero; por lo tanto, debido a las reglas necesarias para realizar un movimiento de especulación con inversión cero y que ya se realizaron todos los movimientos de especulación posibles, entonces la probabilidad de que una configuración permita realizar un movimiento de especulación con inversión cero es muy baja. Sin embargo, al observar la Tabla XIII, se puede ver que si existe un incremento mínimo en el beneficio social promedio en grafos dispersos.

El algoritmo CBAP$^{\times\times}$ obtuvo resultados similares, la diferencia con el algoritmo CBAP^{++} es que disminuye ligeramente el número de movimientos de especulación con inversión cero que realiza y la ganancia promedio obtenida por estos movimientos.

Tabla XII. Movimientos de especulación con inversión cero promedio del algoritmo CBAP^{++}.

		Estrategia			
Modelo	$\rho(G)$	Anti-voraz	ε-voraz	Voraz-I	Voraz-II
Barabási-Albert	0.06	0.05	0.06	0.07	0.03
	0.12	0.03	0.04	0.03	0.01
	0.24	0.01	0.02	0.03	0.01
	0.49	0.00	0.00	0.01	0.00
	0.98	0.00	0.00	0.00	0.00
Erdős-Renyi	0.06	0.09	0.11	0.12	0.11
	0.12	0.02	0.04	0.03	0.02
	0.24	0.00	0.01	0.00	0.00
	0.49	0.00	0.00	0.00	0.00
	0.98	0.00	0.00	0.00	0.00
Crecidos aleatoriamente	0.06	0.01	0.01	0.01	0.00
	0.12	0.00	0.01	0.00	0.01
	0.24	0.00	0.01	0.00	0.01
	0.49	0.00	0.00	0.00	0.00
	0.98	0.00	0.00	0.00	0.00

Tabla XIII. Incremento promedio en el beneficio social después del primer movimiento especulación con inversión cero del algoritmo CBAP^{++}.

		Estrategia			
Modelo	$\rho(G)$	Anti-voraz	ε-voraz	Voraz-I	Voraz-II
Barabási-Albert	0.06	0.36	0.20	0.35	0.08
	0.12	0.74	0.52	0.59	0.19
	0.24	0.18	0.46	0.47	0.12
	0.49	0.00	0.00	0.05	0.00
	0.98	0.00	0.00	0.00	0.00
Erdős-Renyi	0.06	0.96	1.06	1.17	1.27
	0.12	0.16	0.43	0.44	0.44
	0.24	0.00	0.02	0.03	0.00
	0.49	0.00	0.00	0.00	0.00
	0.98	0.00	0.00	0.00	0.00
Crecidos aleatoriamente	0.06	0.10	0.03	0.02	0.04
	0.12	0.05	0.05	0.00	0.08
	0.24	0.00	0.03	0.00	0.02
	0.49	0.00	0.00	0.00	0.07
	0.98	0.00	0.00	0.00	0.00

VII.7. Comparación de algoritmos

En este experimento se comparan los algoritmos CBAP, CBAP^{++} y CBAP$^{\times\times}$ con el fin de determinar qué algoritmo es el que da soluciones con mejor calidad en el beneficio social. No se toman en cuenta los algoritmos CBAP^{+} y CBAP$^{\times}$ ya que se obtendrían resultados ligeramente menores a los de los algoritmos que aplican la técnica de especulación con inversión cero. Los algoritmos CBAP^{++} y CBAP$^{\times\times}$ se diferencian del algoritmo CBAP por el proceso de movimientos de especulación posterior al primer equilibrio encontrado; por lo tanto, la calidad de la solución para ambos algoritmos depende del primer equilibrio encontrado. Para este experimento, se ejecuta el algoritmo CBAP y posteriormente se ejecutan los algoritmos CBAP^{++} y CBAP$^{\times\times}$ con la configuración en equilibrio que dió el algoritmo CBAP. Así, se observa de forma justa la

mejora que tienen los algoritmos entre sí, ya que ambos algoritmos parten de la misma configuración en equilibrio.

En este experimento sólo se utiliza la configuración Anti-voraz de la Sección VII.3 para realizar la comparación entre los algoritmos; esto se debe a que el objetivo de este experimento es ver cuál es el algoritmo que da los mejores resultados; siendo la configuración Anti-voraz la que obtuvo mejores resultados para el enfoque cooperativo en los experimentos de las secciones VII.3-VII.6.

Resultados

En la Figura 22 se muestra el beneficio social promedio de las soluciones que dieron los distintos algoritmos. En ésta se puede observar que los algoritmos CBAP^{++} y CBAP$^{\times\times}$ superan en promedio la calidad de las soluciones del algoritmo CBAP. También se observa que el algoritmo CBAP^{++} tiene configuraciones con una calidad similar a la del algoritmo CBAP$^{\times\times}$ para grafos con una densidad $\rho(G) \leq 0.24$. Sin embargo, el algoritmo CBAP$^{\times\times}$ da soluciones con mejor calidad comparado con los otros dos algoritmos conforme el grafo se hace más denso. Adicionalmente, la Figura 24 muestra que para grafos con una densidad $\rho(G) \geq 0.24$, la peor solución que entrega el algoritmo CBAP$^{\times\times}$ es mejor que la peor solución de los otros dos algoritmos. Los resultados del análisis estadístico muestran que en general, las muestras del beneficio social que entrega el algoritmo CBAP$^{\times\times}$ son de mayor tamaño que las muestras que entregan los otros dos algoritmos.

La Figura 23 muestra el tiempo promedio que tardaron los algoritmos. En la figura, se puede observar que el algoritmo CBAP$^{\times\times}$ es más lento que los otros dos algoritmos; sin embargo, esta diferencia es pequeña comparada con el tiempo total que le lleva

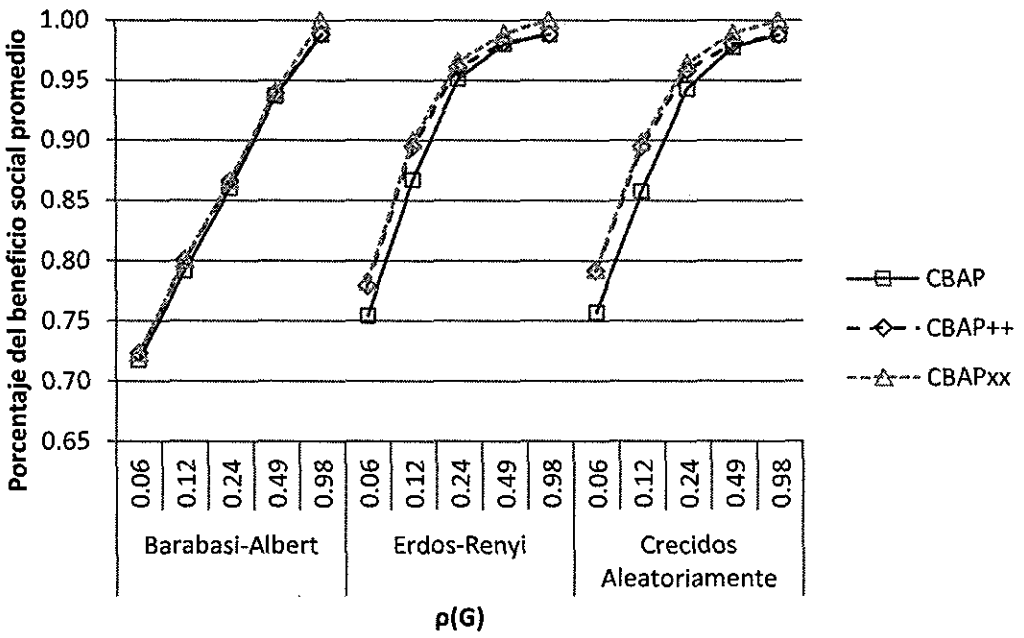

Figura 22. Beneficio social promedio de los algoritmos CBAP, CBAP^{++} y CBAP$^{\times\times}$.

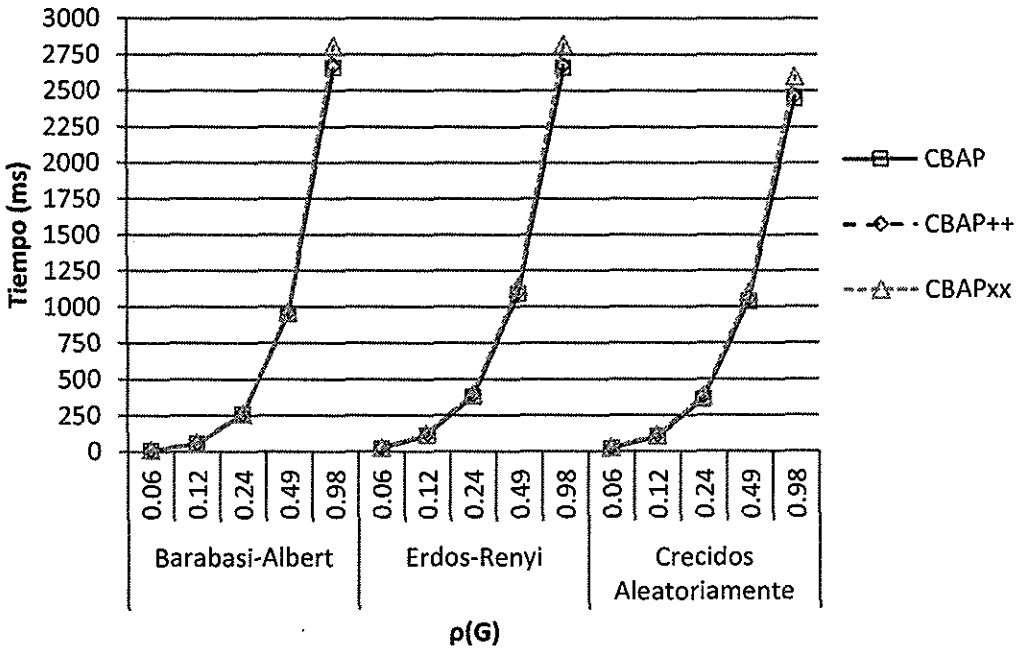

Figura 23. Tiempo total promedio de los algoritmos CBAP, CBAP^{++} y CBAP$^{\times\times}$.

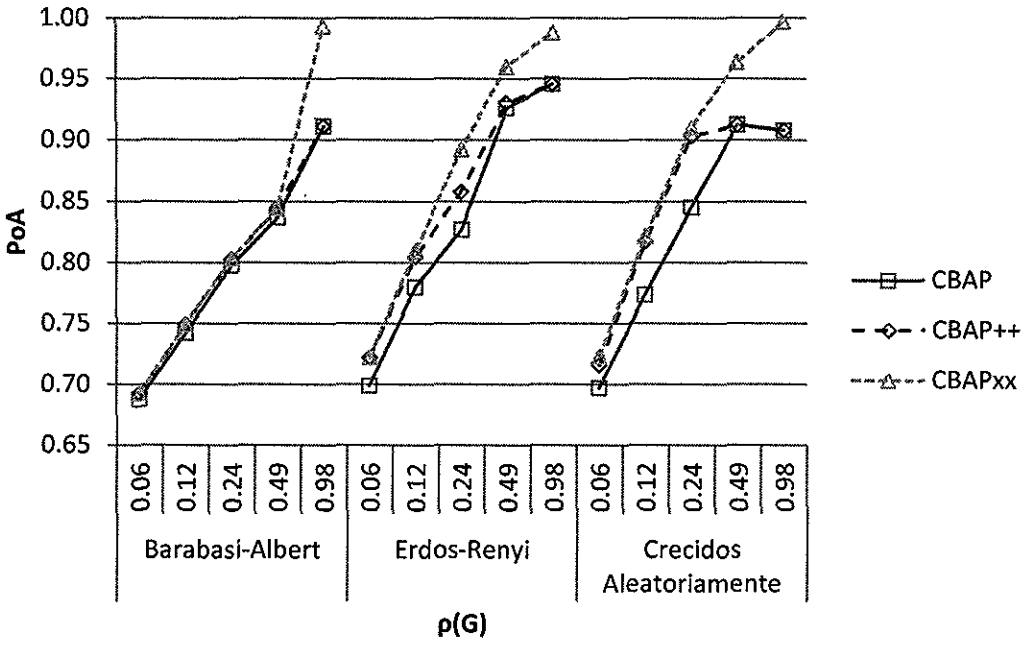

Figura 24. Precio de la anarquía de los algoritmos CBAP, CBAP^{++} y CBAP$^{\times\times}$.

al algoritmo CBAP encontrar el primer equilibrio bajo una configuración Anti-voraz. Debido a esto, el algoritmo CBAP$^{\times\times}$ entrega soluciones de mejor calidad con un incremento mínimo en el tiempo. Sin embargo, si se quiere optimizar el tiempo, se puede utilizar el algoritmo CBAP$^{\times\times}$ con cualquiera de las configuraciones voraces y obtener configuraciones similares a los del algoritmo CBAP con una configuración Anti-voraz y en un tiempo mucho menor.

Capítulo VIII

Conclusiones y trabajo a futuro

VIII.1. Resumen

El Problema de Asignación de Guardaespaldas (PAG) modela un problema de sistemas distribuidos en el que existen dos tipos de procesos con objetivos locales opuestos; sin embargo, los procesos deben cooperar para poder alcanzar un objetivo global. En este trabajo, el objetivo global del sistema es formar un árbol de esparcimiento en el que se maximice el beneficio social, mientras que el objetivo local de los nodos es minimizar (maximizar) su distancia a la raíz del árbol de esparcimiento dependiendo de si el nodo es del tipo blanco (negro). El beneficio social representa la suma de la ganancia individual de cada nodo; las ecuaciones 2 y 3 definen la función de ganancia para los nodos blancos y negros, respectivamente. La solución del PAG es un árbol de esparcimiento tal que todos los nodos se encuentran en un estado estable; es decir, ningún nodo puede incrementar su ganancia individual y maximiza el beneficio social. Fajardo-Delgado *et al.* (2010) modelan el PAG como un juego y lo estudian bajo dos enfoques: cooperativo y no cooperativo; también proponen el algoritmo CBAP para encontrar un equilibrio del juego del PAG en tiempo polinomial, este algoritmo puede trabajar bajo los dos enfoques y dan soluciones aproximadas al PAG; es decir, el algoritmo siempre encuentra un equilibrio pero no siempre maximiza el beneficio social.

El presente trabajo extiende el trabajo de Fajardo-Delgado *et al.* (2010), para mejorar la calidad de los equilibrios encontrados por el algoritmo CBAP para ambos

enfoques. En el trabajo, se analizan los distintos tipos principales de juegos que existen en la teoría de juegos y se clasifica el juego del PAG como un *juego no cooperativo dinámico con información incompleta.*

Se proponen seis variantes distintas para el juego del PAG que buscan mejorar la calidad de los equilibrios encontrados por el algoritmo CBAP; en las variantes propuestas, se modifican las estrategias de los jugadores y se crean 4 estrategias distintas: voraz, anti-voraz, ε-voraz e IA-voraz; la diferencia entre estas estrategias se encuentra en el mecanismo de decisión utilizado por los nodos para determinar con qué nodo van a establecer su conexión para un turno. Se diseña una función de preferencias para los jugadores que utiliza el pago promedio en el tiempo y se puede adaptar para los enfoques cooperativo y no cooperativo; la última variante es la función jugador (calendarizador) anti-voraz, esta función da el turno al jugador que reporta que va a tener el incremento mínimo al realizar su movimiento; la función jugador anti-voraz se diseña para trabajar en conjunto con la estrategia anti-voraz para maximizar el número de movimientos y por lo tanto explorar la mayor cantidad de configuraciones.

Adicionalmente, se estudian cuatro variantes del algoritmo CBAP. Estas variantes trabajan únicamente bajo el enfoque cooperativo y se basan en distintas implementaciones de la técnica de especulación; se demuestra que estos algoritmos siempre terminan en equilibrio, que sus resultados siempre son igual o mejores que los resultados que entrega el algoritmo CBAP y son del mismo orden de tiempo que el algoritmo CBAP.

Por último se presentan resultados experimentales donde se comparan las variantes propuestas del juego del PAG y las variantes propuestas del algoritmo CBAP.

VIII.2. Conclusiones y conjeturas

A continuación se listan las principales conclusiones del presente trabajo.

- El método utilizado para generar el árbol de esparcimiento a optimizar influye en la calidad de la solución. Para grafos con una densidad $\rho(G) = 0.06$ se obtienen mejores resultados al utilizar los métodos DFS y BFS. En las otras densidades se obtienen mejores resultados al utilizar el método PrimRST.

- Para los dos enfoques, el nivel de voracidad de los nodos y el calendarizador es directamente proporcional al número de movimientos de optimización que necesita realizar el algoritmo CBAP para encontrar un equilibrio. Sin embargo, el tiempo de ejecución del algoritmo no lo es, ya que dependiendo del calendarizador un algoritmo que realiza más movimientos puede tardar menos que otro con menos movimientos pero que tiene un calendarizador más lento.

- El enfoque no cooperativo obtiene mejores resultados cuando trabaja con configuraciones voraces; sin embargo, permitir que en ocasiones los nodos dejen su comportamiento voraz, para explorar otras conexiones que mejoran el pago actual del nodo pero no lo maximizan, permite que los nodos encuentran configuraciones en equilibrio con mayor beneficio social a costo de tiempo de ejecución. Así, la configuración que en general da mejores resultados es la ε-voraz.

- El enfoque cooperativo obtiene mejores resultados con la configuración ε-voraz cuando los grafos tienen una densidad $\rho(G) \leq 0.12$. Conforme el grafo se hace más denso, la configuración Anti-voraz comienza a dar mejores resultados; sin embargo, también comienza a tardar más.

- Se demostró que los cuatro algoritmos propuestos encuentran configuraciones en equilibrio con un beneficio social igual o mejor que el del algoritmo CBAP y en el mismo orden de tiempo de ejecución.

- En los experimentos, los cuatro algoritmos propuestos mejoran los resultados del algoritmo CBAP; sin embargo, los algoritmos CBAP^{+} y CBAP^{++} solo mejoran los resultados en grafos con una densidad $\rho(G) \leq 0.24$. Los algoritmos CBAP$^{\times}$ y CBAP$^{\times\times}$ mejoran los resultados en todas las densidades.

- El algoritmo CBAP$^{\times\times}$ da soluciones con mejor calidad que los algoritmos CBAP y CBAP^{++}; sin embargo, calcular la información adicional que requiere el algoritmo hace que sea más lento.

- La configuración Anti-voraz es la que obtiene mejores resultados con todas las variantes del algoritmo CBAP; sin embargo, esta configuración es muy lenta en grafos densos. Por lo tanto, si se quiere reducir el tiempo de ejecución, se pueden utilizar las configuraciones voraces con el algoritmo CBAP$^{\times\times}$ de forma que el equilibrio se encuentre mucho más rápido y tener configuraciones con un beneficio social igual o mejor que cuando se utiliza la configuración Anti-voraz con el algoritmo CBAP^{++}.

VIII.3. Trabajo a futuro

A partir de los resultados obtenidos en este trabajo, se presentan algunas ideas como propuestas para trabajo a futuro:

- En los experimentos se utilizaron grafos donde existen el mismo número de nodos blancos y negros; por lo tanto, se propone estudiar qué sucede con los equilibrios

cuando predominan nodos de un color específico.

- En el PAG, los nodos son blancos o negros y los pesos de las aristas son unitarios; se propone analizar el PAG cuando los nodos tienen pesos, es decir pueden existir distintos niveles en los colores y cuando las conexiones (aristas) tienen pesos.

- En los experimentos se obtuvo que el árbol de esparcimiento a optimizar determina en parte la calidad de la solución que encuentra el algoritmo CBAP. Se propone realizar un algoritmo para generar árboles de esparcimiento que permitan obtener buenos resultados al ejecutar el algoritmo CBAP.

- Proponer un mecanismo que permita que los nodos "mientan" en su distancia a la raíz cuando reportan su distancia a sus vecinos con el fin de que sus descendientes se desconecten de él y el nodo pueda conectarse a uno de sus descendientes para incrementar su ganancia individual. Este mecanismo es una versión no cooperativa de la técnica de especulación.

- Fajardo-Delgado *et al.* (2010) estudian el PAG desde un enfoque semi cooperativo, donde los nodos recolectan los votos hasta cierto nivel de su sub-árbol; sin embargo, demuestran que bajo este enfoque los nodos pueden quedar atrapados en ciclos. Se propone estudiar el PAG bajo un enfoque mixto que utilice los dos enfoques; es decir, en una misma ejecución del algoritmo CBAP, se permite que existan nodos que utilicen el enfoque cooperativo y nodos que utilicen el enfoque no cooperativo.

Referencias

Alon, N., Karp, R. M., Peleg, D., y West, D. B. (1995). A graph-theoretic game and its application to the k-server problem. *SIAM Journal on Computing*, **24**(1): 78–100.

Anthes, G. (2010). Mechanism design meets computer science. *Communications of the ACM*, **53**(8): 11–13.

Barabási, A. L. y Albert, R. (1999). Emergence of scaling in random networks. *Science*, **286**(5439): 509–512.

Boulogne, T. y Altman, E. (2005). Comptetitive routing in multicast communications. *Networks*, **46**(1): 22–35.

Broder, A. (1989). Generating random spanning trees. En *FOCS '89: 30th Annual Symposium on Foundations of Computer Science*, pp. 442–447, Octubre-1 Noviembre, Triangle Research Park, Carolina del Norte, Estados Unidos. IEEE Computer Society.

Callaway, D. S., Hopcroft, J. E., Kleinberg, J. M., Newman, M. E. J., y Strogatz, S. H. (2001). Are randomly grown graphs really random? *Physical Review E*, **64**(4): 1–7.

Chapman, A. C. (2009). *Control of Large Distributed Systems using Games with Pure Strategy Nash Equilibrium.* Tesis de doctorado, University of Southampton, Southampton, Inglaterra, p. 201.

Charilas, D. E. y Panagopoulos, A. D. (2010). A survey on game theory applications in wireless networks. *Computer Networks*, **54**(18): 3421–3430.

Collin, Z. y Dolev, S. (1994). Self-stabilizing depth-first search. *Information Processing Letters*, **49**(6): 297–301.

Dürr, C. y Thang, N. K. (2007). Nash equilibria in voronoi games on graphs. En L. Arge, M. Hoffmann, y E. Welzl (eds.), *ESA '07: 15th Annual European Symposium*, Vol. 4698 de *Lecture Notes in Computer Science*, pp. 17–28, Octubre, Eilat, Israel. Springer.

Erdős, P. y Rényi, A. (1960). On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, **5**: 17–61.

Fajardo-Delgado, D., Fernández-Zepeda, J. A., y Bourgeois, A. G. (2010). The bodyguards allocation problem. Manuscrito en proceso.

Feigenbaum, J. y Shenker, S. (2002). Distributed mechanism design: Recent results and future directions. En *DIAL-M '02: Proceedings of the 6th international workshop on Discrete algorithms and methods for mobile computing and communications*, pp. 1–13, Septiembre, Atlanta, Georgia, Estados Unidos.

Feldman, M., Lai, K., Stoica, I., y Chuang, J. (2004). Robust incentive techniques for peer-to-peer networks. En *EC '04: Proceedings of the 5th ACM Conference on Electronic Commerce*, Vol. 5385, pp. 102–111, Junio, Nueva York, NY, Estados Unidos. ACM.

Fudenberg, D. y Tirole, J. (1991). *Game Theory*. The MIT Press, primera edición. Cambridge, Inglaterra, pp. 1–579. ISBN 0-262-06141.

Griffin, T., Shepherd, F. B., y Wilfong, G. T. (2002). The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking*, **10**(2): 232–243.

Grosu, D. y Chronopoulos, A. T. (2004). Algorithmic mechanism design for load balancing in distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, **34**(1): 77–84.

Halpern, J. Y. (2007). Computer science and game theory: A brief survey. *Computer Research Repository*, **abs/cs/0703148**: 1–17. Publicación informal.

Kearns, M. y Tan, J. (2008). Biased voting and the democratic primary problem. En C. H. Papadimitriou y S. Zhang (eds.), *WINE '08: Proceedings of the 4th International Workshop on Internet and Network Economics*, Lecture Notes in Computer Science, pp. 639–652, Diciembre, Shangai, China. Springer.

Koutsoupias, E. y Papadimitriou, C. H. (2009). Worst-case equilibria. *Computer Science Review*, **3**(2): 65–69.

Kreps, D. M. y Wilson, R. (1982). Sequential equilibria. *Econometrica*, **50**(4): 863–894.

Machado, R. y Tekinay, S. (2008). A survey of game-theoretic approaches in wireless sensor networks. *Computer Networks*, **52**(16): 3047–3061.

Moscibroda, T., Schmid, S., y Wattenhofer, R. (2006). On the topologies formed by selfish peers. En *PODC '06: Proceedings of the 25th ACM Symposium on Principles of Distributed Computing*, pp. 133–142, Julio, Denver, Colorado, Estados Unidos. ACM.

Nash, J. (1951). Non-cooperative games. *The Annals of Mathematics*, **54**(2): 286–295.

Nisan, N. (1999). Algorithm for selfish agents: Mechanism design for distributed computation. En C. Meinel y S. Tison (eds.), *STACS '99: Proceedings of the 16th Annual Symposium on Theoretical Aspects of Computer Science Trier*, Vol. 1563 de *Lecture Notes in Computer Science*, pp. 1–15, Marzo, Universidad de Trier, Alemania. Springer Berlin / Heidelberg.

Nisan, N. y Ronen, A. (1999). Algorithmic mechanism design. En *STOC '99: Proceedings of the 31th annual ACM symposium on Theory of computing*, pp. 129–140, Mayo, Atlanta, Georgia, Estados Unidos. ACM.

Nisan, N., Roughgarden, T., Tardos, E., y Vazirani, V. V. (2007). *Algorithmic Game Theory*. Cambridge University Press. Nueva York, NY, Estados Unidos, pp. 1–753. ISBN 0521872820.

Osborne, M. J. (2004). *An Introduction to Game Theory*. Oxford University Press. Nueva York, NY, Estados Unidos, pp. 1–533. ISBN 9780195128963.

Raidl, G. R. y Julstrom, B. A. (2003). Edge sets: an effective evolutionary coding of spanning trees. *IEEE Transactions on Evolutionary Computation*, **7**(3): 225–239.

Roughgarden, T. (2005). *Selfish Routing and the Price of Anarchy*. The MIT Press. Cambridge, Inglaterra, pp. 1–196. ISBN 978-0-262-18243-0.

Roughgarden, T. y Tardos, É. (2002). How bad is selfish routing? *Journal of the ACM*, **49**(2): 236–259.

Shoham, Y. y Leyton-Brown, K. (2009). *Multiagent Systems: algorithmic, game-theoretic and logical foundations*. Cambridge University Press. Cambridge, Inglaterra, pp. 1–513. ISBN 978-0-521-89943-7.

von Neumann, J. y Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press, segunda edición. Princeton, Nueva Jersey, Estados Unidos, pp. 1–739. ISBN 9780691003627.