UNIVERSIDAD DE LA REPUBLICA URUGUAY

UNIVERSIDAD DE LA REPÚBLICA

Facultad de Ingeniería

Instituto de Computación

Algoritmos de Aproximación para el Problema de Steiner con Penalidades

Informe de Proyecto de Grado

Cristian Bauza

Pablo Olivera

Dr. Ing. Franco Robledo

Tutor

Dr. Ing. Pablo Romero

Tutor

Montevideo, Marzo de 2017

Agradecimientos

Cristian Bauza

Quiero dedicar este trabajo a todas aquellas personas que me han acompañado a lo largo de estos 10 años de estudio.

A mis padres, Juan y Beatriz por haberme dado la oportunidad de estudiar durante gran parte de mi vida a costa de su trabajo y esfuerzo y junto con ellos a mis hermanos Facundo y Yoana por siempre haber comprendido mis momentos de ausencia por dedicarle tiempo a esta carrera.

También quiero dedicar y agradecer a mi pareja, Natalia quien ha estado presente en gran parte de esta etapa de mi vida y que ha sabido comprenderme, ayudarme y ha aceptado postergar muchas cosas en pos de que yo llegara a este punto. Estas cinco personas sin dudas han sido un pilar fundamental para mi.

Estoy muy agradecido a Pablo Olivera y Martín Berguer por haber trabajado codo a codo todo este tiempo, ha sido un placer enorme haber trabajado con ellos y haber compartido los distintos momentos.

Por último quiero agradecer a los tutores Pablo Romero y Franco Robledo por haber confiado este proyecto a nosotros y por habernos acompañado, guiado y alentado a lo largo de todo este proceso.

Pablo Olivera

Este trabajo está dedicado principalmente a mis padres y familiares que me han acompañado de forma incondicional durante la carrera. Y a las personas que de una manera u otra han sido partícipes.

Quiero agradecer a Cristian Bauza y a Martín Berguer por todo el esfuerzo dedicado al proyecto, ha sido un privilegio haber compartido este tiempo con ellos. A mis tutores Pablo Romero y Franco Robledo por la confianza, dedicación y aliento.

RESUMEN

Algoritmos de Aproximación para el Problema de Steiner con Penalidades

Cristian Bauza y Pablo Olivera

Marzo/2017

El objeto de estudio de este informe es algoritmos de aproximación aplicados al problema del árbol de Steiner con penalidades (PCST por sus siglas en inglés).
El problema consiste en diseñar una subred de costo mínimo que conecte a todos o algunos nodos clientes con una central, donde la no conexión de un cliente tiene una correspondiente penalización y la subred solución debe ser un árbol.

La motivación a este trabajo parte de la necesidad de asignar un conjunto de generadores eólicos en un parque, en ciertos lugares factibles predefinidos, donde además existen penalizaciones por aquellos lugares que queden sin un generador. El problema consiste en encontrar una subred que interconecte los lugares con generador donde se minimice el costo de la red más las penalizaciones por los lugares que queden vacíos. El problema puede considerar un nodo particular (nodo raíz) el cual debe pertenecer a la subred solución.

Como punto de partida se realiza un análisis de las principales metodologías para atacar el problema, un repaso de los conceptos más importantes de complejidad computacional y luego un estudio enfocado a algoritmos de aproximación existentes para el PCST. La parte fundamental de este trabajo está enfocada en el análisis, comprensión e implementación de un algoritmo de aproximación factor 2 que sigue el esquema primal-dual con sincronización (propuesto por *Michel X. Goemans* y *David P. Williamson*), así como sus posibles variantes que incluyan aleatorización. Se estudian resultados obtenidos sobre instancias estándares ya estudiadas en otros trabajos académicos. Los resultados que se obtienen determinan que el algoritmo tiene un comportamiento muy bueno en cuanto al compromiso entre tiempos de ejecución y valores objetivo. En cuanto a las variantes planteadas, existen algunos casos en los que se obtienen mejores resultados que con el algoritmo original.

Por último se presentan las conclusiones y trabajo a futuro.

Índice

Lista de Abreviaturas y Siglas

PCST	*Prize Collecting Steiner Tree*
ILP	*Integer Linear Program*
LP	*Linear Program*
GW	Algoritmo $Goemans - Williamson$
RN	Algoritmo Random por Nodos
RA	Algoritmo Random por Aristas
GAP, gap	Distancia entre dos valores
PCTSP	*Prize Collecting Traveling Salesman Problem*
TSP	*Traveling Salesman Problem*
DIMACS	*Center for Discrete Mathematics and Theoretical Computer Science*
ICERM	*Institute for Computational and Experimental Research in Mathematics*
DOE	Despacho Óptimo de Energía
DES	*Dynamic Edge Splitting*

Capítulo 1

Introducción

1.1. Motivación

La planificación energética es un tema estratégico de los operadores de energía eléctrica de cada país. En Uruguay ha tomado altísima relevancia en la planificación del despacho de carga debido a la integración de energías renovables y otras fuentes de generación adicionales a las centrales hidroeléctricas, así como las perspectivas futuras de crecimiento de esta nueva realidad.

En Uruguay las herramientas de planificación energética básicamente se descomponen en tres según el horizonte: corto plazo (menos de 24 horas), mediano plazo (entre 48 horas y dos semanas) y largo plazo (3 meses aproximadamente, o más). En la Facultad de Ingeniería, un grupo de Investigadores del Dpto. de Investigación Operativa y del IMERL están trabajando en un proyecto financiado por la Agencia Nacional de Investigación e Innovación denominado: "Planificación estocástica óptima para la generación y acumulación diaria de energía, integrada a políticas de control en *Smart Grids*". Bajo este contexto dicho proyecto busca los siguientes objetivos:

i) Desarrollar e implementar un modelo unificado de optimización estocástica y algoritmos para la resolución del problema de despacho óptimo de energía (DOE) en el corto plazo (24 horas), para sistemas con porcentajes significativos de energías renovables no acumulables, que integra elementos para acumulación y la posibilidad

de afectar dinámicamente el consumo mediante cambios en los precios.
ii) Generar escenarios prospectivos representativos de las distintas inversiones y/o políticas a consideración de las autoridades competentes, como: bombear agua hacia embalses de centrales hidráulicas, nuevas centrales de bombeo, uso residencial de baterías y/o dispositivos telecontrolados para ciertos tipos de electrodomésticos.
iii) Resolver las instancias asociadas a esos escenarios para evaluar la conveniencia económica en el ecosistema eléctrico nacional (*social welfare*) de las inversiones y políticas bajo consideración. Recientemente el país ha incrementado en forma sostenida el porcentaje de potencia instalada proveniente de energías renovables, se espera que para fines de 2017 el 90 % de la energía anualmente consumida provenga de fuentes renovables, que se descomponen en: hidráulica, eólica, solar y biomasa. La imposibilidad práctica de acumular energía eólica y solar, debe compensarse con la utilización racional de otras formas de energía, pero normalmente esto no es suficiente ni económico. El notable avance nacional en las tecnologías de la información y la comunicación abre espacio para explorar alternativas más eficientes, como impulsar un consumo eléctrico inteligente mediante la fijación dinámica de precios, o incluso recurrir a la acumulación residencial como fuente de potencia hacia la red pública. El problema es intrínsecamente estocástico y como resultados esperados se tiene:
a) Motor de optimización estocástica que integre generación y demanda en un mismo DOE.
b) Instancias del modelo correspondientes a los escenarios prospectivos de políticas bajo evaluación.
c) Resultados cuantitativos para el retorno esperado de esos escenarios.
El proyecto es altamente complejo por las múltiples variables en juego y subproblemas que surgen de su análisis. El equipo de FING está en plena ejecución del proyecto con finalización estimada para mediados de 2018. Ahora bien, como una de las partes vinculadas al proyecto, surge el problema de localización óptima de un conjunto de generadores (eólicos) dentro de un parque preestablecido (un campo). Es así que el modelo conocido como *Prize-Collecting Steiner Tree Problem* (PCST), se considera adecuado para modelar este problema (en una variante simplificada)

de asignación de generadores a lugares factibles e interconectarlos topológicamente a costo mínimo. Con penalizaciones por el no uso de aquellos sitios que quedan "vacíos" (no se colocarán generadores). Es así que este proyecto de final de carrera ataca técnicas de solución basadas en algoritmos de aproximación para resolver el PCST, como un aporte más desde el enfoque académico a técnicas de optimización que resuelvan una componente de un problema mayor como lo es la planificación óptima del despacho de carga teniendo en cuenta políticas prospectivas. En este caso, el proyecto de grado busca aportar soluciones al problema de diseñar topologías de despliegue de parques eólicos, pensando más que nada en la expansión de este tipo de fuentes de generación en Uruguay y la apuesta que desde el gobierno se ha hecho con políticas de largo plazo al respecto. Sus resultados serán de ayuda comparativa para las soluciones que diseñen los integrantes del proyecto mayor, ejecutado por los investigadores de FING.

1.2. Definición del Problema

A continuación se presenta una definición formal del problema que será abordado a lo largo del documento. Dado un grafo no dirigido $G = (V, E)$, un nodo raíz $r \in V$, una función de costos sobre las aristas $c : E \to \mathbf{Q}^+$ y una función de ganancias sobre los nodos $\pi : V \to \mathbf{Q}^+$. El problema *Prize Collecting Steiner Tree* (de ahora en adelante PCST) consiste en encontrar un sub-grafo conexo que incluya al nodo r y que maximice la suma de las ganancias correspondientes a los nodos que están incluidos en el sub-grafo menos los costos de las aristas que están en el mismo [Ljubić et al., 2006].

La formulación matemática del problema es la siguiente:

$$
\begin{array}{rr}
\text{Dados} & \\
G = (V, E) & \text{Grafo no dirigido} \\
r \in V & \text{Nodo raíz} \\
c : E \to \mathbf{Q}^+ & \text{Función de Costos} \\
\pi : V \to \mathbf{Q}^+ & \text{Función de Ganancias} \\
\text{Encontrar} & \\
G' = (V', E') & \text{Sub-grafo de G} \\
\text{Tal que} & \\
G' & \text{Es conexo} \\
r \in V' & \\
G' \text{ es tal que maximiza } \; ganancia(G') = \sum_{v \in V'} \pi(v) - \sum_{e \in E'} c(e) &
\end{array}
\tag{1.1}
$$

1.3. Historia

Para tratar la historia del problema de Steiner primero es necesario repasar el problema de Fermat-Torricelli, cuyo origen se remonta al siglo XVII en el trabajo de Pierre de Fermat titulado *Method for Determining Maxima and Minima and Tangents to Curved Lines*. Cabe suponer que fue motivado por la pregunta de René Descartes sobre las curvas de la forma $\sum_{i=1}^{4} ||p_i - x|| = c$ partiendo de p_1 ... p_4 puntos dados y una constante c.

Para tres puntos, el problema de Fermat-Torricelli es el siguiente: dados los puntos A, B y C hay que encontrar un punto P tal que la suma de las distancias PA, PB y PC sea mínima. El primero en encontrar una solución al problema fue Torricelli, de aquí es que el punto P además de llamarlo el punto de Fermat también es conocido como el punto de Torricelli. La primera aparición del problema del árbol

de Steiner euclídeo data del año 1811 por el matemático y lógico francés Joseph Diaz Gergonne quien presenta algunas generalizaciones al problema de Fermat-Torricelli en el trabajo denominado *Annales de Gergonne*. Gergonne plantea la interrogante de encontrar una red de distancia mínima para conectar una serie de puntos dados, a diferencia del problema original se pueden incluir más de un punto extra [Brazil et al., 2014]. El problema de Steiner con cuatro puntos aparece en el año 1836 en la carta que le escribe Carl Friedrich Gauss a Heinrich Christian Schumacher donde se discute brevemente el problema del árbol de Steiner euclídeo, dicha carta se puede ver en la carátula de [Vazirani, 2001]. Luego, la generalización a n puntos fueron contribuciones de Karl Bopp y Eduard Hoffmann en la segunda mitad del siglo XIX. Sobre el árbol solución del problema de Steiner euclídeo, conocido como mínimo árbol de Steiner, se conocen algunas propiedades estructurales que debe cumplir. Si el mínimo árbol tiene n terminales, entonces a lo sumo puede tener $n-2$ nodos de Steiner. Todo par de líneas debe tener un ángulo de por lo menos 120°, esto implica que los nodos de Steiner incluidos deben tener grado tres y con ángulos de 120° [Gilbert y Pollak, 1968].

La definición formal del problema de Steiner típico es la siguiente: dado un grafo $G = (V, E)$, un subconjunto de nodos $T \subseteq V$ denominados terminales, una función de costos $c : E \to Q^+$, el problema consiste en minimizar el costo del árbol que conecta a todos los nodos de T, pudiendo usar nodos de $V \setminus T$, estos son llamados nodos de Steiner.

Hasta la actualidad se han planteado múltiples variantes al problema de Steiner, el presente trabajo trata sobre una de ellas, conocida como *Prize Collecting Steiner Tree*. Otras variantes son *Node-Weighted Steiner Tree Problem*, *Rectilinear Steiner Tree Problem* y *Generalized Steiner Problem*. La descripción de estas y otras extensiones del problema clásico se pueden consultar en [Garey y Johnson, 1977; Segev, 1987; Winter, 1987; Gordeev y Tarastsov, 1993].
La primera mención sobre el PCST se encuentra en [Bienstock et al., 1993] donde se plantea un algoritmo para el *Prize Collecting Salesman Problem* y además se presenta una extensión de dicho algoritmo que aplica para esta variante del problema de Steiner.

1.4. NP-Hardness

Cuando se plantea un algoritmo para encontrar la solución a un problema, habitualmente se trata de que sea el más eficiente, entonces la pregunta que surge es: ¿cuándo podemos afirmar que un algoritmo es eficiente? La noción de eficiente en el sentido amplio de la palabra está relacionado con la utilización de los recursos computacionales. La definición más clásica hace referencia al tiempo que tarda un algoritmo en retornar una solución en función del tamaño de la entrada. Para abordar más formalmente la respuesta a la pregunta, a continuación se exponen los conceptos más importantes.

Una función $f(n)$ es de orden $O(g(n))$ si existe una constante $c \in R$ y $n_0 \in N$ tal que $|f(n)| \leq c \cdot |g(n)|$ para todo $n \geq n_0$, siendo $f(n)$ la función de complejidad que representa el peor caso para todas las entradas posibles.

Definición 1.4.1. *Un algoritmo es de tiempo polinomial cuando su función de complejidad es $O(p(n))$ para algún polinomio p, en caso contrario es de tiempo exponencial.*

Definición 1.4.2. *Un problema es una pregunta general a ser respondida, usualmente posee múltiples parámetros cuyos valores no están especificados. La descripción de un problema está dada por (1) una especificación general de sus parámentros y (2) las propiedades que debe cumplir la solución. Una instancia de un problema se obtiene fijando sus parámetros [Garey y Johnson, 1979].*

Definición 1.4.3. *Un problema es intratable si ocurre que es tan complejo que no existe algoritmo que lo resuelva en tiempo polinomial.*

Definición 1.4.4. *Un problema de decisión Π consiste en un conjunto de instancias D_Π y un subconjunto de instancias $Y_\Pi \subseteq D_\Pi$ cuya respuesta es "sí" y para $D_\Pi \setminus Y_\Pi$ la respuesta es "no".*

Otras definiciones importantes de complejidad computacional hacen referencia a las clases de problemas.

Definición 1.4.5. *Un problema pertenece a la clase P si y solo si existe un algoritmo que lo resuelva en tiempo polinomial en función del tamaño de la instancia.*

Definición 1.4.6. *Un problema de decisión pertenece a la clase NP si y solo si dada una candidata a solución existe un algoritmo que en tiempo polinomial en función del tamaño de la instancia puede verificar si dicha candidata es solución.*

A partir de estas definiciones se puede ver que $P \subseteq NP$, ya que si un problema puede ser resuelto en tiempo polinomial, se observa que cumplirá con la Definición 1.4.6. También de aquí surge una de las conjeturas más importantes en la ciencia de la computación ¿es $P \neq NP$?

La principal técnica para demostrar que dos problemas están relacionados se conoce con el nombre de "reducción", la cual consiste en dar una transformación que permita mapear una instancia de un problema a una instancia equivalente del otro, cuando estas transformaciones se pueden ejecutar en tiempo polinomial se le llama "reducción de tiempo polinomial". Si existe una reducción de este tipo de un problema P_1 a un problema P_2 y existe un algoritmo de tiempo polinomial para solucionar P_2, entonces se podría encontrar un algoritmo de tiempo polinomial para solucionar P_1.

Definición 1.4.7. *Dado un problema de decisión π diremos que es NP-Difícil si y solo si para todo $\pi' \in NP$ se cumple que π' se reduce a π. Si $\pi \in NP$, entonces diremos que π es NP-Completo.*

Las bases para la teoría de NP-Completitud fueron introducidas en 1971 por Stephen Cook en su trabajo denominado *The complexity of theorem-proving procedures* [Cook, 1971]. Además, de introducir otros conceptos ya mencionados demuestra que el problema SAT (*satisfiability*) es NP-Completo, siendo el primero descubierto de esta clase de problemas.

Una consideración importante a tener en cuenta es que todo problema de optimización se puede transformar en un problema de decisión. Si un problema de optimización busca algún tipo de mínimo costo, entonces puede ser llevado a un problema de decisión incluyendo una cota numérica k, donde la decisión está en preguntar si la instancia dada tiene costo por debajo de k. Análogamente se puede aplicar lo mismo si el problema es de maximización.

Richard Karp se basa en el trabajo de Cook para presentar una lista con los primeros 21 problemas de decisión que demuestra que son NP-Completos [Karp, 1972]. Estos son versiones de problemas combinatorios conocidos como por ejemplo: el ciclo hamiltoniano (dirigido y no dirigido), el problema de la mochila, ordenamiento de tareas, corte máximo y el del número cromático de un grafo. Dentro de esta lista el principal interés para este informe está puesto sobre el problema del árbol de Steiner, ya que es ahí donde se demuestra que es NP-Completo. El problema de Steiner se trata de un caso particular del PCST donde para los nodos de Steiner se considera $\pi(v) = 0$ (ganancia cero) y los nodos terminales $\pi(v) = \infty$ (ganancia infinita) lo que implica que el PCST también es NP-Completo. Debido a que el PCST pertenece a esta clase de problemas la utilización de algoritmos exactos no siempre es viable para instancias formadas por un grafo de gran tamaño. A continuación se presentan dos alternativas para tratar el problema en grafos grandes, además, de los métodos exactos.

El principal material consultado para esta sección es [Garey y Johnson, 1979], donde además, se encuentra una amplia cobertura sobre estos temas y es uno de los trabajos más importantes en el área de complejidad computacional.

1.4.1. Métodos Exactos

Como se menciona en la sección anterior, el PCST es un problema NP-Difícil, por lo cual no podemos obtener una solución exacta en tiempo polinomial, lo que hace que encontrar una solución exacta no siempre sea factible para instancias de gran tamaño. Para obtener una solución exacta al problema un posible enfoque es considerar una formulación en forma de problema de programación lineal y luego aplicar técnicas y algoritmos conocidos para resolver este tipo de problemas.
Se plantea una formulación del problema basados en [Goemans y Williamson, 1995]. En primer lugar, para poder tener dicha formulación se exponen algunas consideraciones y definiciones:

- La función de maximización se puede cambiar a una de minimización, para ello se debe observar que se puede transformar la función objetivo de maximizar

la ganancia de los vértices incluidos en G' menos los costos de las aristas incluidas en G' a minimizar las perdidas generadas por los vértices que no son considerados en G' más los costos de las artistas consideradas en G'.

- De acuerdo a lo descripto en el punto anterior se considera la función π de ganancias sobre los nodos como una función de penalización y se define π_v como la penalización obtenida por dejar al nodo $v \in V$ fuera de la solución.
- Se define c_e como el costo de una arista $e \in E$.
- Se define un corte de V como un subconjunto de vértices de V.
- Dado un corte S de V, se define $\delta(S)$ como el conjunto de aristas que cruzan el corte S, es decir, una arista $e = (v, w)$ pertenece a $\delta(S)$ si $v \in S$ y $w \in \overline{S}$, donde $\overline{S}$ es el complemento de S.
- Se define una variable Z_T la cual tomará el valor 0 para todo $T \subseteq V$ que no sea exactamente el conjunto de vértices no considerados en la solución factible y 1 en el caso contrario.
- Se define una variable x_e que tomará el valor 1 si la arista $e \in E$ es considerada en la solución factible y 0 en caso contrario.
- Dado un corte S, se define $\chi(\delta(S))$ como la sumatoria de las variables x_e de las artistas que pertenecen a $\delta(S)$.

Dadas las definiciones anteriores, la formulación en forma de ILP es la siguiente:

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e + \sum_{T \subset V; r \notin T} Z_T \Big(\sum_{i \in T} \pi_i\Big) & & \\
\text{sujeto a} \quad & \chi(\delta(S)) + \sum_{T \supseteq S} Z_T \geq 1 & & S \subset V; r \notin S \\
& \sum_{T \subset V; r \notin T} Z_T \leq 1 & & \\
& x_e \in \{0, 1\} & & e \in E \\
& Z_T \in \{0, 1\} & & T \subset V; r \notin T
\end{aligned}
\tag{1.2}
$$

Como se observa en la Ecuación 1.2, el problema se transforma a un problema de programación lineal entera donde $\sum_{e \in E} c_e x_e$ representa los costos de las aristas incluidas en la solución y $\sum_{T \subset V; r \notin T} Z_T(\sum_{i \in T} \pi_i)$ representa la suma de las penalizaciones correspondientes a los nodos no incluidos en la solución, ya que como se definió anteriormente, Z_T es 0 a no ser cuando T es exactamente el conjunto de nodos no incluidos en la solución. Por otra parte, las restricciones se pueden leer de la siguiente manera, $\chi(\delta(S)) + \sum_{T \supseteq S} Z_T \geq 1$ indica que si se considera un corte S sobre el conjunto de nodos o bien tiene que haber por lo menos una arista que una dicho corte con su complemento, o de lo contrario en S solo hay nodos no considerados en la solución, si no fuera así quedarían nodos incluidos en la solución tanto en S como en su complemento $\overline{S}$ pero ninguna arista que los conecte, lo cual es absurdo ya que el problema exige que la solución sea conexa.

A partir de esta formulación del problema se puede obtener una relajación del mismo para poder abordarlo con alguno de los métodos exactos existentes, como puede ser el algoritmo *branch and cut*, un ejemplo de resolución con dicho método se puede ver en [Ljubić et al., 2006].

1.4.2. Metaheurísticas

Un posible enfoque para la resolución del problema en instancias grandes, es decir aquellas que dada la intratabilidad del problema no podemos encontrar una solución exacta en un tiempo razonable, es un enfoque metaheurístico. Los enfoques metaheurísticos buscan soluciones cercanas al óptimo a un costo computacional razonable, aunque no garantizan optimalidad y en muchos casos no se conoce el grado máximo de error que podrían tener.

Según se expresa en [Siarry, 2016], se pueden distinguir dos tipos de problemas de optimización, por un lado los problemas discretos en los cuales sus variables toman valores en un conjunto discreto de posibilidades y por otro los problemas con variables continuas, es decir aquellos en los que las variables toman valores en un intervalo continuo. No obstante, en la realidad también se pueden encontrar problemas mixtos, los cuales comprenden simultáneamente variables continuas y

discretas. En la literatura se hace referencia a dos tipos de problemas pertenecientes a la clase NP-Difícil:

- Ciertos tipos de problemas de optimización discretos donde no se conoce un algoritmo exacto de orden polinomial.
- Ciertos tipos de problemas de optimización de variable continua donde no se conoce un algoritmo que permita conocer el óptimo global en un número finito de cálculos.

Muchos esfuerzos han sido realizados a lo largo del tiempo para resolver estos problemas de forma separada. En el caso de la optimización continua existe un arsenal de métodos tradicionales, pero en general estas técnicas no son efectivas si el problema no posee ciertas propiedades estructurales como por ejemplo, que sea un problema convexo. En el caso de los problemas de optimización discretos existen un gran número de heurísticas desarrolladas, pero la mayoría de ellas fueron concebidas para un problema en particular. El arribo de las metaheurísticas marca una reconciliación entre los dos dominios, estas pueden aplicarse a todo tipo de problemas discretos y ser adaptadas a problemas continuos.

Los métodos metaheurísticos tienen en común las siguientes características:

- Son, al menos hasta cierto punto, estocásticos: este enfoque permite contrarrestar la cantidad exponencial de posibilidades.
- Son generalmente de origen discreto y tienen la ventaja decisiva en el caso de la continuidad de ser directos, es decir, no recurren a cálculos a menudo problemáticos de los gradientes de la función objetivo.
- Están inspirados por analogías: de procesos físicos (recocido del acero), biológicos (algoritmos evolutivos, búsqueda tabú), comportamientos etológicos (colonias de hormigas, enjambres de partículas).
- También comparten la misma desventaja, las dificultades de ajustes de los parámetros y grandes tiempos de cálculos.

- Normalmente comparten dos características de búsqueda, la diversificación (exploración a lo largo de todo el espacio de búsqueda con cierto grado de aleatoriedad) e intensificación (búsqueda alrededor de un punto de forma intensiva, en un espacio reducido).

La tendencia actual es la aparición de métodos híbridos que se esfuerzan por beneficiarse de las ventajas de combinar los diferentes enfoques.

Otro aspecto importante en las metaheurísticas es que presentan todo tipo de extensiones, se pueden citar por ejemplo:

- Optimización multiobjetivo: Se debe optimizar problemas con más de un objetivo y donde los mismos pueden ser contradictorios.
- Optimización multimodal: Se intenta buscar un conjunto de óptimos globales o locales.
- Optimización dinámica: La función objetivo puede ir evolucionando con el paso del tiempo.
- Uso de implementaciones paralelas.

A continuación se presenta una breve descripción de los principales métodos clásicos metaheurísticos:

- *Recocido Simulado* - Tiene como origen el proceso de recocido utilizado por los metalúrgicos para obtener un estado sólido "bien ordenado" de mínima energía (evitando las estructuras metaestables características de los mínimos locales de la energía). Esta técnica consiste en calentar un material a una temperatura elevada e ir bajando lentamente esta temperatura. El método de recocido simulado transpone el proceso de recocido a la solución de un problema de optimización: la función objetivo del problema, de manera similar a la energía de un material, se minimiza con la ayuda de la introducción de un temperatura ficticia, que en este caso es un simple parámetro controlable del algoritmo.

- *Tabu Search* - Fue formalizado en 1986 y su característica principal se basa en el uso de mecanismos inspirados en la memoria humana. El método tabú, desde este punto de vista, toma un camino opuesto al del recocido simulado, que no utiliza la memoria en absoluto y por lo tanto es incapaz de aprender lecciones del pasado. Por otra parte, el modelado de la memoria introduce múltiples grados de libertad, lo que dificulta cualquier análisis matemático riguroso.

- *Algoritmos evolutivos* - Los algoritmos evolutivos son técnicas de búsqueda inspiradas en la evolución biológica de las especies y aparecieron a finales de los años 50. Los algoritmos genéticos constituyen el ejemplo más conocido. Los métodos evolutivos despertaron inicialmente un interés limitado, debido a su alto costo de ejecución, pero en el siglo XXI han experimentado un desarrollo considerable, que puede atribuirse al aumento significativo de la evolución del hardware y la correspondiente capacidad de cómputo, en particular con la aparición de las arquitecturas con capacidades de cómputo paralelo. Un algoritmo evolutivo simple puede describirse de la siguiente forma: se considera un conjunto de N puntos en un espacio de búsqueda, estos son seleccionados a priori al azar y son considerados la población inicial. Cada punto individual tiene determinado rendimiento el cual es mayor cuanto mejor sea el resultado al evaluar la función objetivo en dicho punto. El algoritmo evolutivo consiste en evolucionar gradualmente durante sucesivas generaciones la población, manteniendo el número de individuos constante, el objetivo de dicha evolución es mejorar el rendimiento de los individuos.

- *Colonia de Hormigas* - Estos métodos se esfuerzan por simular la capacidad para resolver ciertos problemas observados en colonias de hormigas, cuyos miembros están equipados individualmente con facultades muy limitadas. Las colonias de hormigas siempre siguen el mismo camino y este camino es el más corto posible. Este es el resultado de un modo de comunicación indirecta a través del medio ambiente llamado *stigmergy*. Cada hormiga deposita a lo largo de su trayectoria una sustancia química (llamada feromona), todos los miembros de la colonia perciben esto y dirigen su caminata hacia las áreas más olorosas. Esto resulta particularmente en una capacidad colectiva de encontrar

el camino más corto rápidamente después de que la trayectoria original haya sido bloqueada por un obstáculo. Este comportamiento ha sido tomado como punto de partida para el diseño de algoritmos.

Según se expresa en [Osman y Kelly, 1996], las metaheurísticas tuvieron un desarrollo importante luego de su inserción entrada la década de 1980 y han tenido éxito en el ataque de problemas de optimización combinatoria prácticos y difíciles. Las metaheurísticas son métodos aproximados diseñados para atacar problemas de optimización combinatoria difíciles donde las heurísticas clásicas fallan en ser eficientes y efectivas. Las metaheurísticas proveen un *framework* general que permite crear nuevos métodos híbridos combinando diferentes conceptos derivados de las heurísticas clásicas, inteligencia artificial, evolución biológica, sistemas neuronales y mecanismos estadísticos. Se puede definir las metaheurísticas como un proceso de generación iterativo que es guiado por alguna heurística mediante la combinación inteligente de diferentes conceptos para explorar y explotar los espacios de búsqueda usando estrategias de aprendizaje para estructurar la información en orden y así poder encontrar soluciones cercanas al óptimo de forma eficiente.

1.4.3. Algoritmos de Aproximación

Una tercera forma de atacar problemas de optimización combinatoria NP-Difíciles es utilizar algoritmos de aproximación. Estos son utilizados tanto para problemas intratables como para situaciones donde el objetivo es llegar rápidamente a una solución cercana al óptimo y por alguna razón no se necesita la solución exacta. Un aspecto importante a tener en cuenta cuando se estudian este tipo de algoritmos es el factor de aproximación (también conocido como radio de aproximación), que es el radio entre el resultado obtenido al aplicar el algoritmo y la solución óptima. La presencia del factor marca una diferencia fundamental con respecto a los mecanismos mencionados en el punto anterior correspondiente a metaheurísticas, donde generalmente no se tiene asegurado una distancia máxima entre el valor hallado y el óptimo.

Más formalmente, cuando es un problema de optimización y en particular un problema de minimización, un algoritmo de aproximación A de factor fijo $c \geq 1$ produce para toda instancia I una solución factible $s(I)$ cuyo costo verifica $f(s(I)) \leq c * OPT(I)$, donde $OPT(I)$ representa la solución óptima para la instancia I de un problema. En el caso de los problemas de maximización, el factor c será tal que $c \leq 1$ y la solución factible $s(I)$ verificará $f(s(I)) \geq c * OPT(I)$.

Cuando se necesita resolver un problema lo ideal sería: encontrar soluciones óptimas en tiempo polinomial. Si se trata de problemas NP-Difíciles y en particular en instancias grandes, en la mayoría de los casos se debe de relajar alguno de los dos requerimientos. Una opción posible es ser más laxo con el tiempo que se tarda en obtener la solución. Si dada la realidad del problema es posible esperar el tiempo necesario hasta encontrar la solución óptima, entonces este es un enfoque posible. La segunda opción es estar dispuesto a obtener soluciones no óptimas, pero muy cercanas a estas y en un tiempo pequeño. Los algoritmos de aproximación utilizan este enfoque.

El objetivo a la hora de diseñar un algoritmo de aproximación, además, de obtener una solución factible al problema, es que la diferencia con respecto al óptimo sea tan pequeña como sea posible.

Las razones fundamentales por las que estudiar y aplicar algoritmos de aproximación son las siguientes:

- Son aplicables a muchos problemas del tipo NP-Difíciles en los cuales lo que interesa es obtener una solución suficientemente cercana al óptimo y no el óptimo necesariamente.
- Son utilizados en la práctica y están basados en una base matemática rigurosa que aseguran cierto grado de fiabilidad en la aproximación obtenida en el peor caso.

Se recomiendan dos libros para profundizar en el tema: [Williamson y Shmoys, 2011] y [Vazirani, 2001].

1.5. Conclusiones

Dado que el problema de Steiner con penalidades pertenece a la clase de problemas NP-Completos, existen tres caminos naturales para su resolución. La primera opción es utilizar métodos exactos que su principal inconveniente es el costo computacional prohibitivo para instancias de gran tamaño. Un segundo enfoque puede ser la utilización de algoritmos metaheurísticos. Una tercera opción la cual se va a adoptar en este proyecto es el desarrollo de algoritmos de aproximación. El objetivo de este proyecto consiste en familiarizarse con algoritmos de aproximación para el PCST. Teniendo en cuenta dicho objetivo se considera una solución que es conceptualmente simple, con factor de aproximación satisfactorio y además sobre ella se intentará conseguir mejoras agregando aleatorización.

1.6. Organización del Documento

En el Capítulo 2 se introducen los principales conceptos que son fundamentales para la comprensión del algoritmo que se presenta en el Capítulo 3, luego se describen trabajos publicados sobre los algoritmos de aproximación más importantes aplicados al PCST. El Capítulo 3 está exclusivamente dedicado al algoritmo de aproximación $Goemans-Williamson$ de factor 2 y a la demostración de dicho factor que cumple el algoritmo, cerrando con un ejemplo de aplicación al PCST en un grafo. El Capítulo 4 trata sobre algoritmos aleatorios, los cuales implican la utilización de un parámetro seleccionado al azar y para finalizar se detalla un algoritmo con aleatorización que aplica al PCST. El Capítulo 5 está dedicado al análisis experimental, múltiples comparativas entre los algoritmos implementados frente a resultados de aplicaciones externas que brindan soluciones exactas y se propone una nueva variante al algoritmo aleatorio que se presentó en el Capítulo 4. Por último en el Capítulo 6 se presentan las conclusiones y trabajo a futuro. En el Apéndice A se encuentra la información completa de las ejecuciones analizadas en el Capítulo 5 y también detalles sobre la implementación, tanto sobre las estructuras utilizadas como los parámetros que recibe el programa.

Capítulo 2

Algoritmos de Aproximación

2.1. Esquema Primal-Dual

Los conceptos fundamentales del problema dual surgieron por primera vez en una conversación entre John Von Neumann y George B. Dantzig en octubre de 1947. El concepto de dualidad aparece implícitamente en un trabajo publicado por Von Neumann unas semanas después [Dantzig, 1963].

George B. Dantzig es quien creó el método Simplex para la resolución de problemas de programación lineal, considerado como uno de los más importantes desarrollados en el siglo XX. Dados sus aportes se lo considera como el precursor de la programación lineal. En el libro [Dantzig, 1963] se puede tener una visión general de los logros de Dantzig en programación matemática hasta 1963 [Karp, 2008]. Este libro ha dado lugar a una serie de trabajos posteriores muy importantes, entre los cuales podemos mencionar:

- Teorema de máximo flujo y corte mínimo - Es un teorema aplicable al flujo de redes que indica que el flujo máximo desde un punto de origen en la red hasta un destino es igual al peso de las aristas en el corte mínimo, esto es el conjunto de aristas con menor peso que si fueran removidas se desconectaría el origen del destino. Este teorema tiene aplicaciones importantes en la logística de distribución así como redes de transporte, flujo de gases y líquidos entre

muchos otros.

- Teorema Minimax de Von Neumann - Es uno de los teoremas más importantes en el área de la teoría de juegos. El teorema establece que en un juego de dos personas de suma cero existe por lo menos un par de estrategias equilibradas [Szász, 2011].

- Algoritmos de aproximación - En el área de algoritmos de aproximación hay técnicas que toman como base la programación lineal y en particular el esquema primal-dual que será explicado en este capítulo para el diseño de algoritmos de aproximación [Vazirani, 2001].

A continuación se presentan los conceptos fundamentales de dualidad y el método primal-dual que es utilizado para el diseño de algoritmos de aproximación.

2.1.1. Dualidad

Se considera la siguiente formulación estándar de un problema de minimización.

$$\begin{aligned} &\text{Primal:} \\ &\min \quad \sum_{j=1}^{n} c_j x_j \\ &\text{sujeto a} \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \quad i = 1, ..., m \\ &\qquad\qquad x_j \geq 0, \qquad\qquad j = 1, ..., n \end{aligned} \tag{2.1}$$

$c_j,\ a_{ij},\ y\ b_i$ son números racionales dados.

Para obtener el dual se introducen las variables y_i para cada desigualdad de la primera restricción.

$$\begin{aligned} &\text{Dual:} \\ &\text{máx} \quad \sum_{i=1}^{m} b_i y_i \\ &\text{sujeto a} \quad \sum_{i=1}^{m} a_{ij} y_i \leq c_j, \quad j = 1, ..., n \\ &\qquad\qquad y_i \geq 0, \qquad\qquad i = 1, ..., m \end{aligned} \tag{2.2}$$

Teorema 1 (Teorema de Dualidad Fuerte)**.** *El problema primal tiene solución óptima finita si y solo si el problema dual tiene solución óptima finita. Además los valores óptimos de las funciones objetivo coinciden [Dantzig, 1963].*

El teorema anterior es útil para algoritmos que obtienen la solución exacta, mientras que para algoritmos de aproximación por lo general alcanza con el siguiente.

Teorema 2 (Teorema de Dualidad Débil)**.** *Sea* $x = (x_1, ..., x_n)$ *e* $y = (y_1, ..., y_m)$ *soluciones factibles del problema primal y del dual respectivamente, entonces:*

$$\sum_{j=1}^{n} c_j x_j \geq \sum_{i=1}^{m} b_i y_i$$

Prueba. Como y es una solución factible del dual y los x_j's son no negativos,

$$\sum_{j=1}^{n} c_j x_j \geq \sum_{j=1}^{n} (\sum_{i=1}^{m} a_{ij} y_i) x_j$$

Analogamente, como x es una solución factible del primal y los y_i's son no negativos,

$$\sum_{i=1}^{m} (\sum_{j=1}^{n} a_{ij} x_j) y_i \geq \sum_{i=1}^{m} b_i y_i$$

Luego alcanza con observar,

$$\sum_{j=1}^{n} (\sum_{i=1}^{m} a_{ij} y_i) x_j = \sum_{i=1}^{m} (\sum_{j=1}^{n} a_{ij} x_j) y_i$$

□

Teorema 3 (Condiciones de Holgura Complementaria)**.** *Sea* x *e* y *soluciones factibles del problema primal y del dual respectivamente, entonces* x *e* y *ambas son óptimas si y solo si se cumplen las siguientes dos condiciones:*

- ***Condición de holgura complementaria primal***
  Para todo $1 \leq j \leq n$*: si* $x_j \neq 0 \Rightarrow \sum_{i=1}^{m} a_{ij}y_i = c_j$

- ***Condición de holgura complementaria dual***
  Para todo $1 \leq i \leq m$*: si* $y_i \neq 0 \Rightarrow \sum_{j=1}^{n} a_{ij}x_j = b_i$

Prueba. Si x e y son soluciones óptimas, por el Teorema de Dualidad Fuerte se cumplen:

$$\sum_{j=1}^{n} c_j x_j = \sum_{j=1}^{n}(\sum_{i=1}^{m} a_{ij}y_i)x_j$$

$$\sum_{i=1}^{m}(\sum_{j=1}^{n} a_{ij}x_j)y_i = \sum_{i=1}^{m} b_i y_i$$

entonces se cumplen las condiciones de holgura complementarias.
En el caso del recíproco, si se cumplen las condiciones de holgura entonces se cumplen las igualdades mencionadas en el párrafo anterior, además, $\sum_{j=1}^{n} c_j x_j = \sum_{i=1}^{m} b_i y_i$. Por el Teorema de Dualidad Débil $\sum_{j=1}^{n} c_j \overline{x}_j \geq \sum_{i=1}^{m} b_i \overline{y}_i$ para toda solución factible $\overline{x}$ e $\overline{y}$, por lo tanto x e y deben ser soluciones óptimas. □

Las condiciones de holgura complementaria son muy útiles e importantes en el diseño de algoritmos, ya sean exactos o aproximados. De estas se desprenden las condiciones de holgura complementarias relajadas, que son fundamentales en los algoritmos de aproximación, formalizadas por primera vez en [Williamson et al., 1995].

Teorema 4 (Condiciones de Holgura Complementaria Relajadas)**.** *Sea* x *e* y *soluciones factibles del problema primal y del dual respectivamente que cumplen con las siguientes condiciones:*

- ***Condición de holgura complementaria relajada primal***
  Dado $\alpha \geq 1$*. Para todo* $1 \leq j \leq n$*: si* $x_j \neq 0 \Rightarrow c_j/\alpha \leq \sum_{i=1}^{m} a_{ij}y_i \leq c_j$

- ***Condición de holgura complementaria relajada dual***
  Dado $\beta \geq 1$*. Para todo* $1 \leq i \leq m$*: si* $y_i \neq 0 \Rightarrow b_i/\beta \leq \sum_{j=1}^{n} a_{ij}x_j \leq b_i$

Entonces:

$$\sum_{j=1}^{n} c_j x_j \geq \alpha \cdot \beta \cdot \sum_{j=1}^{m} b_i y_i$$

Prueba.

$$\sum_{j=1}^{n} c_j x_j \leq \alpha \sum_{j=1}^{n} \Big(\sum_{i=1}^{m} a_{ij} y_i \Big) x_j = \alpha \sum_{i=1}^{m} \Big(\sum_{j=1}^{n} a_{ij} x_j \Big) y_i \leq \alpha\beta \sum_{j=1}^{m} b_i y_i$$

□

2.1.2. Método Primal-Dual

Es un método utilizado para el diseño de algoritmos de aproximación y aplicable a una amplia variedad de problemas NP-Difíciles. Una forma común de hacerlo es basándose en las condiciones expuestas en el Teorema 4. Los algoritmos diseñados en base a este método parten desde una solución dual factible y una solución primal no factible y de manera iterativa va mejorando la factibilidad de la solución primal al mismo tiempo que se va acercando al óptimo de la solución dual. Se asegura que al final se llega a una solución factible del primal y ambas condiciones de holgura se cumplen. En cada iteración la mejora en el primal y dual van de la mano, la solución actual del primal se utiliza para mejorar el dual y viceversa. Cuando termina el algoritmo el costo de la solución del dual es utilizado como cota inferior del óptimo y por el Teorema 4 se asegura un factor de aproximación $\alpha\beta$.

Existe una forma denominada primal-dual con sincronización, donde las variables duales se incrementan todas en un mismo monto, a diferencia de la forma no sincronizada que estas crecen de maneras desiguales.

A continuación una breve historia sobre el esquema primal-dual el cual fue inicialmente propuesto para el diseño de algoritmos exactos, fue planteado por [Dantzig et al., 1956] inspirados en el trabajo de Kuhn que propuso el Método Húngaro para el problema de asignación. Al esquema original, con la introducción de algunas modificaciones permite el diseño de algoritmos de aproximación para una amplia

variedad de problemas NP-Difíciles. El primero conocido diseñado utilizando el esquema primal-dual es el trabajo de [Bar-Yehuda y Even, 1981] que da un algoritmo de aproximación de factor 2 para el problema de cobertura de vértices.

Para una historia más detallada ver [Goemans y Williamson, 1997].

2.2. Trabajos Relacionados

En la literatura se puede encontrar un amplio número de trabajos que aplican al PCST, desde diferentes enfoques: exactos, metaheurísticos y algoritmos de aproximación. La presente sección está dedicada principalmente a algoritmos de aproximación.

El primer algoritmo de aproximación presentado para el PCST fue realizado por [Bienstock et al., 1993]. En este trabajo se formula un algoritmo de aproximación para el problema del agente viajero con penalidades (PCTSP por sus siglas en inglés), en una versión donde el costo de las aristas cumplen la desigualdad triangular (Ver Sección A.1). La formulación del problema se plantea en forma de programación lineal donde agrega una restricción que asegura que un nodo j debe estar en el recorrido.

El algoritmo consta de tres fases:

Algoritmo 1:

1 Resolver el problema relajado variando $j \in V$ por todos los nodos. Se generan tantas soluciones como nodos tenga el grafo.

2 **para** *cada una de las soluciones* **hacer**

3 Se construye un sub-grafo inducido por los nodos que cumplen que su valor asociado en la solución óptima al problema relajado es mayor a una constante dada.

4 Al sub-grafo se le aplica el algoritmo de Christofides [Christofides, 1976] para obtener una solución al TSP. Al costo de la solución obtenida se le debe sumar la penalización por los nodos que quedaron afuera del sub-grafo.

5 **fin**

6 El resultado final retorna el mejor costo entre la mínima de las $|V|$ soluciones y la que no visita ningún nodo.

Luego se presenta una extensión al trabajo que indica que este algoritmo también es aplicable al PCST. Esto es posible realizando las modificaciones correspondientes a la formulación ILP y modificando el algoritmo aplicado en el paso 4, es decir,

sustituir Christofides por un algoritmo que aplique al problema del árbol de Steiner típico.

El siguiente trabajo es realizado por [Goemans y Williamson, 1995] donde se plantea un algoritmo de factor 2 utilizando el método primal-dual con sincronización, en adelante algoritmo $Goemans-Williamson$ (GW). A continuación se expone una breve descripción que luego se explicará en profundidad en el Capítulo 3.

El resultado final de aplicar el algoritmo en un grafo conexo no dirigido es un árbol, cuyas aristas se van agregando en cada iteración. Al comienzo se parte de un bosque donde todos sus nodos son aislados, luego iterativamente se agregan aristas reduciendo a lo largo de las iteraciones la cantidad de componentes conexas. El agregado de aristas nunca forma ciclos. Como último paso se ejecuta una fase de poda que remueve las ramas del árbol que no aportan mejoras a la solución.

Luego [Johnson et al., 2000] partiendo del algoritmo propuesto por [Goemans y Williamson, 1995] presentan una mejora para la fase de poda, la cual denominaron *strong pruning*. El algoritmo es recursivo y la idea general es podar aquellos sub-arboles que la suma de las penalizaciones de sus nodos menos el costo de las aristas sea menor o igual a cero. En el Capítulo 3 se expondrán los detalles del algoritmo.

Los siguientes dos trabajos están aplicados al *unrooted* PCST, donde la diferencia se presenta en que no existe un nodo raíz.

Una implementación al algoritmo $Goemans-Williamson$ dada por [Cole et al., 2001] obtiene un orden computacional $O(k(n+m)\log^2 n)$, lo cual es una mejora con respecto al algoritmo original que tiene un orden $O(n^3 \log n)$, ya que se debe ejecutar una vez por nodo, siendo k una constante dada, n el número de nodos y m la cantidad de aristas. Si bien se mejora el orden computacional hay una degradación en el factor de aproximación: $2+\frac{1}{n^k}$. Para esta mejora en la implementación, proponen un método denominado *Dynamic Edge Splitting (DES)* el cual se enfoca en la etapa del algoritmo $Goemans-Williamson$ que selecciona cuales subconjuntos se van a unir. Para ello se consideran tres categorías de aristas donde el criterio de partición toma en cuenta ciertas propiedades de las componentes conexas de los extremos de

las aristas. En lineas generales el *DES* consiste en eliminar una de estas categorías y esto lo logra dividiendo la arista en dos, agregando un nodo ficticio y haciendo que este nodo sea una nueva componente aislada. La aplicación del *DES* provoca que el criterio para considerar que arista tomar en cada iteración se vea modificado con respecto al algoritmo $Goemans - Williamson$ y como consecuencia la degradación del factor. Afirma el documento que es necesario estructuras de datos específicas para lograr el orden de ejecución mencionado.

Luego en [Feofiloff et al., 2007] se introduce un algoritmo basado en el esquema primal dual para el *unrooted* PCST, este algoritmo logra un orden $O(n^2 \log n)$ y un factor de aproximación de $2 - \frac{2}{n}$. La formulación del problema es la siguiente.

$$
\begin{aligned}
&\text{Primal:} \\
&\text{min} \quad \sum_{e \in E} c_e x_e + \sum_{L \subseteq V} \pi(L) Z_L \\
&\text{sujeto a} \quad \sum_{e \in \delta_G S} x_e + \sum_{L \supseteq S} Z_L + \sum_{L \supseteq \overline{S}} Z_L \geq 1 \quad S \subseteq V \\
&\qquad\qquad x_e \geq 0 \qquad\qquad e \in E \\
&\qquad\qquad Z_L \geq 0 \qquad\qquad L \subseteq V.
\end{aligned}
\tag{2.3}
$$

$\delta_G S$ es el conjunto de aristas del grafo G con un nodo en S y el otro en $\overline{S}$. Considerando $G_T = (V_T, E_T)$ el grafo solución, se introducen las variables: x_e es 1 si la arista $e \in E_T$, 0 en otro caso y Z_L es 1 si $L = V \setminus V_T$, 0 en otro caso. Una solución factible está formada por el par (x, Z).

$$
\begin{aligned}
&\text{Dual:} \\
&\text{máx} \quad y(R) \\
&\text{sujeto a} \quad y(R(e)) \leq c_e \qquad e \in E \\
&\qquad\qquad \sum_{S \subseteq L} y_S + \sum_{S \supseteq \overline{L}} y_S \leq \pi(L) \quad L \subseteq V \\
&\qquad\qquad y_S \geq 0 \qquad\qquad S \subseteq V.
\end{aligned}
\tag{2.4}
$$

R representa todos los posibles subconjuntos de V. Para todo conjunto S puede

ocurrir que: tiene una arista en $\delta_G S$, o bien S está incluido en un conjunto que penaliza, o bien $\overline{S}$ está incluido en un conjunto que penaliza.

A continuación se introducen algunos conceptos utilizados para describir el algoritmo. Una arista e está *tight* cuando se da la igualdad $y(R(e)) = c_e$, también aplica para conjuntos, cuando se cumple $\sum_{S \subseteq L} y_S + \sum_{S \supseteq \overline{L}} y_S = \pi(L)$ se dice que L está *tight*. Una arista es externa cuando sus nodos están en diferentes subconjuntos.
El algoritmo tiene dos fases claramente separadas, una de *crecimiento* y luego una de *poda*. La primera fase es iterativa, cada iteración comienza con un grafo F y una solución factible y del problema dual 2.4. Un subconjunto se considera activo si el conjunto de vértices no está *tight*. La fase termina cuando solo queda un subconjunto activo y se pasa a la segunda fase el árbol inducido por F en este subconjunto. Mientras los subconjuntos activos sean mayores que uno, entonces se incrementan de manera sincronizada los duales correspondientes a todos los conjuntos de A, siendo A el conjunto de los subconjuntos activos. El incremento se detiene cuando ocurre al menos una de las siguientes condiciones: una arista externa, un conjunto de A, o el complemento de un conjunto de A queda *tight*. Si ocurre la primera se añade la arista a F y se vuelve al comienzo de la iteración. Cuando se cumple la segunda condición se comienza una nueva iteración. El proceso iterativo se detiene si ocurre la tercera condición.

La segunda fase tiene como entrada un árbol T inducido por F y S que es una lista de conjuntos que en alguna iteración pasaron de activos a inactivos en la fase anterior. Luego se itera mientras exista algún elemento S_k de S que tenga una sola arista de T en la frontera, o sea, $|\delta_T S_k| = 1$, si esto ocurre se eliminan de T todas las aristas con al menos un nodo en S_k y al finalizar la iteración se obtiene el árbol solución.

En el año 2013 se publica una versión mejorada en la cual se brindan más detalles acerca de la implementación práctica del algoritmo.

El siguiente trabajo que logra bajar el factor 2 es presentado por [Archer et al., 2011], está basado en [Goemans y Williamson, 1995]. A continuación se incluye el pseudocódigo del algoritmo y una explicación de sus puntos principales.

I_β se obtiene multiplicando las penalizaciones de la instancia I por β, $\forall\ \beta > 0$.

Algoritmo 2: PCST(β)

1 $T^{GW} \leftarrow$ Aplicar $Goemans - Williamson$ a $I_{\frac{1}{2}}$.

2 Generar la solución T^{ST}:

3 Aplicar la fase de crecimiento de $Goemans - Williamson$ a I_β.

4 $D \leftarrow \{\ v \in V\ :$ Al terminar la etapa anterior v está incluido en alguna componente conexa que no contenga al nodo raíz. $\}$

5 $T^{ST} \leftarrow$ Aplicar algún algoritmo del problema de Steiner típico a la instancia I donde los terminales es el conjunto $V - D$.

6 Retorna la mejor solución de T^{GW} y T^{ST}.

El factor de aproximación es de 1.9672. En lineas generales para la demostración se plantean dos casos. Sea δ tal que $\pi(\overline{V(O)}) = \delta \cdot OPT$, siendo O el árbol de la solución óptima y δ representa la fracción del óptimo que es contribuida por el término de la penalización de la función objetivo, si $\delta \geq \epsilon\ con\ \epsilon > 0$ se cumple que la solución T^{GW} proporciona un factor $2\ -\ \epsilon$. En el otro caso, $\delta < \epsilon\ con\ \epsilon > 0$ se cumple que la solución T^{ST} proporciona un factor $2\ -\ \epsilon$. En este último caso observar la importancia de la elección del parámetro β y que depende del algoritmo aplicado en el paso 5 del algoritmo 2. Para lograr el factor 1.9672 en el paso 5 se utiliza el algoritmo desarrollado por [Byrka et al., 2010] el cual tiene un factor de aproximación de $Ln(4)$ para el problema del árbol de Steiner típico.

Otro posible enfoque para encontrar una solución aproximada al problema es planteada en [Williamson y Shmoys, 2011], la cual es una solución con una componente aleatoria. La forma planteada para encontrar una solución aproximada consiste en el siguiente proceso. En primer lugar, encontrar una solución exacta al problema relajado mediante algún método, por ejemplo, el método del elipsoide lo que produce una salida (x^*, s^*). Luego se obtiene cierto valor α de forma aleatoria y con distribución uniforme sobre $[\gamma, 1)$ con $0 \leq \gamma < 1$, se seleccionan los nodos cuyo valor de la variable s_i^* sea mayor o igual que α. Luego se define un problema de Steiner típico, donde los nodos terminales son los seleccionados anteriormente y el resto son nodos de Steiner. Por último, la solución final será la que surja de aplicar un algoritmo al

problema de Steiner típico. Se puede demostrar que este algoritmo tiene un factor de aproximación de 2,54.

Como fue expresado en Capítulo 1 existen básicamente tres enfoques para la resolución del problema PCST, a continuación se presentan otros trabajos relacionados basados en métodos exactos y heurísticas que si bien no aplican de forma directa a este proyecto, merecen la pena mencionar.

Entre los trabajos enfocados a resolver el problema en forma exacta se encuentra el de [Ljubić et al., 2006], en este trabajo se presenta un algoritmo basado en el método *branch-and-cut* y utiliza un modelo de programación lineal para un grafo dirigido y restricciones de conectividad basadas en desigualdades asociadas a cortes del conjunto de nodos. Utiliza el algoritmo de máximo flujo para detectar los subconjuntos cuyas desigualdades asociadas no se cumplen. El algoritmo planteado presenta una formulación ILP para un grafo dirigido, por lo que previo a esta formulación es necesario una transformación del grafo no dirigido a uno dirigido al cual se le agrega además un nodo raíz artificial llamado r. En la transformación del grafo se define un nuevo conjunto de nodos $V_{SA} = V \cup \{r\}$, luego se define el conjunto de aristas $A_{SA} = \{\bigcup_{(i,j)\in E}\{(i,j),(j,i)\}\} \cup \{(r,i) : i \in V, \pi_i > 0\}$ el cual contiene todas las aristas pertenecientes a E pero en ambos sentidos más un conjunto de aristas que parten desde el nodo raíz r hacia los nodos clientes, donde por nodo cliente se entiende todo aquel nodo que tienen penalización mayor que cero y el conjunto de nodos clientes se denomina R. Los costos de las aristas pertenecientes al grafo transformado se definen de la siguiente forma, $c'_{(r,i)} = -\pi_i$ (El costo de las aristas que parten desde el nodo r hacia un nodo i tienen como costo el opuesto de la penalización del nodo i), luego $c'_{(i,j)} = c_{(i,j)} - \pi_j$ (El costo del resto de las aristas es el costo original menos la penalización del nodo de llegada). En la formulación ILP se introduce una variable x_{ij} por cada arista (i,j) y una variable s_i para cada nodo i. Es necesario para introducir el modelo ILP utilizado en [Ljubić et al., 2006] definir lo siguiente, $\delta(S) = \{(i,j) : i \in \bar{S}, j \in S\}$.

$$\min \sum_{ij \in A_{SA}} c'_{ij} x_{ij} + \sum_{i \in V_{SA}} \pi_i \quad (1)$$

sujeto a

$$\begin{aligned}
&\sum_{ji \in A_{SA}} x_{ji} = s_i && \forall i \in V_{SA} \setminus \{r\} && (2)\\
&\chi(\delta(S)) \geq s_k && k \in S, r \notin S, \forall S \subset V_{SA} && (3)\\
&\sum_{ri \in A_{SA}} x_{ri} = 1 && && (4)\\
&s_i, x_{ij} \in \{0,1\} && \forall i \in V_{SA} \setminus \{r\}, \forall (i,j) \in A_{SA}. && (5)\\
&x_{rj} \leq 1 - s_i && i < j, i \in R && (6)\\
&\sum_{ji \in A_{SA}} x_{ji} \leq \sum_{ij \in A_{SA}} x_{ij} && \forall i \notin R, i \neq r && (7)
\end{aligned} \quad (2.5)$$

La restricción (3) es llamada *desigualdad de conectividad* y garantiza que para cada nodo incluido en la solución exista una ruta desde el nodo r, la restricción (2) asegura que todo nodo de la solución solo tenga un predecesor y por último la restricción (4) es relevante en el caso del problema sin nodo raíz y asegura que el nodo artificial r solo esté conectado por una arista. Posteriormente se agregan las llamadas restricciones de asimetría (6), estas restricciones evitan que se generen tantas soluciones como nodos clientes con la única variación de cual nodo cliente es conectado al nodo r. Por último, se agrega una restricción de flujo que asegura que para todo nodo que no pertenezca a R la cantidad de aristas entrantes tiene que ser menor o igual que la cantidad de aristas salientes, esta restricción es la llamada restricción de *balance de flujo*. El algoritmo *branch-and-cut* consta de los siguientes pasos, en cada nodo del árbol *branch-and-bound* se resuelve la relajación de 2.5, donde se sustituyen las restricciones (5) por $s_i \leq 1, x_{ij} \leq 1 \; \forall i \in V_{SA} \setminus \{r\}, \forall (i,j) \in A_{SA}$. El algoritmo tiene una fase de inicialización donde por cada par de aristas $(i,j),(j,i)$ puede estar a lo sumo una en la solución si es que el nodo i y/o el nodo j son considerados en la misma. Considerando que existe una cantidad exponencial de restricciones del tipo (3), estas no son agregadas al inicio. Durante la fase de separación la cual es aplicada para cada nodo del árbol correspondiente al algoritmo *branch-and-bound* se agrega una restricción del tipo (3) que es violada por la solución

del modelo ILP relajado. La violación a la restricción (3) puede ser buscada en tiempo polinomial por el algoritmo de flujo máximo.

En el año 2014 se desarrolló una competencia organizada en conjunto por DIMACS [1] e ICERM [2] dedicada a diferentes variantes del problema de Steiner, incluyendo el PCST, con un enfoque de investigación en la teoría y también en la práctica. Uno de los trabajos presentados que ganó en varias categorías de la competencia es el desarrollado por [Fischetti et al., 2016]. El ejecutable, sin acceso al código fuente, está accesible con fines de investigación y ha sido de gran utilidad para la fase experimental, su nombre es *Staynerd* [3]. Un aporte a destacar de [Fischetti et al., 2016] es sobre un modelo de programación lineal del problema donde solo tiene variables asociadas a los nodos, como consecuencia su aplicación está orientada a grafos en que los costos de las aristas es uniforme.

Previamente a la presentación del modelo, se introduce un nuevo concepto denominado "separador de nodos" que dice lo siguiente: dados dos nodos k y l pertenecientes a V, un subconjunto de nodos $N \cup V \setminus \{k, l\}$ es llamado $(\{k, l\})$-separador si y solo si luego de eliminar N de V no existe ningún camino entre k y l en el grafo. Al conjunto con todos los $(\{k, l\})$-separador se le llama $\mathscr{N}(l, k)$.

Se parte con un grafo $G = (V, E)$ con todas las aristas de costo c y con penalizaciones p_v para cada nodo v del grafo. En consecuencia que no hay variables asociadas a las aristas es necesario redefinir el costo de los nodos, incorporando los costos de las aristas y las penalizaciones de los nodos: $c_v = c - p_v,\ \forall v \in V$. Se definen las variables: $P = \sum_{v \in V} p_v,\ \forall v \in V$, y_v que vale 1 si el nodo v pertenece a la solución o 0 en caso contrario y por último $y(N) = \sum_{v \in N} y_v$.
Se considera un conjunto de terminales T_r que eventualmente puede ser vacío, de esta manera el problema es el PCST que se ha tratado a lo largo del documento. Se define un conjunto denominado "terminales potenciales": $T_p = \{v \in V \setminus T_r \ : \exists\{u, v\}\ sujeto\ a\ c_{uv} < p_v\}$ y el conjunto $T = T_p \cup T_r$.

[1] http://dimacs.rutgers.edu/
[2] http://icerm.brown.edu/
[3] http://homepage.univie.ac.at/ivana.ljubic/research/staynerd/StayNerd.html

$$
\begin{aligned}
\min \quad & \sum_{v \in V} c_v y_v + (P - c) & & \\
\text{sujeto a} \quad & y(N) \geq y_i + y_j - 1 & \forall i, j \in T, i \neq j, \forall N \in \mathscr{N}(i,j) & \\
& y_v = 1 & \forall v \in T_r & \\
& y_v \in \{0,1\} & \forall v \in V \setminus T_r &
\end{aligned}
\tag{2.6}
$$

La restricción $y(N) \geq y_i + y_j - 1$ asegura que la solución es conexa. Si se consideran dos nodos, $i, j \in T$ pertenecientes a la solución, al menos un nodo de alguno de los $N \in \mathscr{N}(i,j)$ también debe estar incluido, por lo que existe un camino que conecta a i con j.

La implementación dependiendo del tipo de instancia es el modelo que aplica, cuando los costos de las aristas son todos iguales, utiliza el modelo de la Ecuación 2.6 y en otros casos sigue el modelo utilizado en [Ljubić et al., 2006], el cual fue presentado en párrafos anteriores.

Se han tenido en cuenta los trabajos que se consideran más importantes dentro de la literatura investigada, ya sea porque son citados en múltiples investigaciones relacionadas con el PCST y/o también por que han marcado un avance importante en alcanzar mejores soluciones al problema.

Considerando que el objetivo del proyecto es familiarizarse con los algoritmos de aproximación aplicados al PCST y habiendo estudiado una serie de trabajos relacionados al mismo, se decide seleccionar a $Goemans-Williamson$ como el algoritmo a implementar. Entre los criterios evaluados para la selección está el factor de aproximación y debido a este se descartaron algunos trabajos, por otra parte, en los casos donde el factor es mejor al de GW existen algunas desventajas que llevó a descartarlos. En [Archer et al., 2011] la desventaja está por el lado del tiempo computacional y [Feofiloff et al., 2007] por la complejidad algorítmica. Es importante notar que el trabajo de [Goemans y Williamson, 1995] ha servido de inspiración para un número importante de trabajos relacionados al PCST, por lo que se considera que es fundamental comprenderlo, además, es una forma clara de aplicación el esquema primal-dual con sincronización. En resumen, se elige estudiar en profundidad e im-

plementar el algoritmo *GW* por su buen compromiso entre factor de aproximación, tiempo de ejecución y simplicidad algorítmica.

Capítulo 3

Goemans-Williamson

3.1. Algoritmo

En esta sección se describe el algoritmo propuesto por [Goemans y Williamson, 1995] aplicado al PCST con la modificación de la poda propuesta por [Johnson et al., 2000].

Se parte de un grafo $G = (V, E)$, un nodo raíz $r \in V$, cada arista e tiene un costo asociado c_e y cada vértice v una penalización π_v. Tomando como base la formulación del problema en forma de problema de programación lineal entera expuesta en la sección 1.2 quitamos la restricción $\sum_{T \subset V; r \notin T} Z_T \leq 1$, ya que esto no afecta la solución óptima. Por otra parte se cambia la función objetivo de modo de hacerla más intuitiva introduciendo una variable s_i, que toma el valor 1 si el nodo i está en la solución y 0 en caso contrario. Tomando en cuenta estas modificaciones mencionadas, se plantea la formulación del problema relajado.

Primal:

$$\begin{aligned} \min \quad & \sum_{e\in E} c_e x_e + \sum_{i\neq r}(1-s_i)\pi_i & & \\ \text{sujeto a} \quad & \chi(\delta(S)) \geq s_i & & i \in S;\ r \notin S \\ & x_e \geq 0 & & e \in E \\ & s_i \geq 0 & & i \in V;\ i \neq r. \end{aligned} \tag{3.1}$$

En esta formulación $\sum_{e\in E} c_e x_e$ representa la suma del costo de las aristas incluidas en la solución y $\sum_{i\neq r}(1-s_i)\pi_i$ representa la penalización por los nodos que no están incluidos en la solución, esto por la definición de s_i que se menciona anteriormente. Luego, la restricción $\chi(\delta(S)) \geq s_i \quad i \in S;\ r \notin S$ modela que para todo corte S que no contenga al nodo r y que tenga algún nodo perteneciente a la solución, entonces debe existir en la solución alguna arista que cruce el corte S, lo que asegura la conectividad de la solución. Luego, tenemos la formulación del problema dual asociado.

Dual:

$$\begin{aligned} \text{máx} \quad & \sum_{S:r\notin S} y_s & & \\ \text{sujeto a} \quad & \sum_{S:e\in\delta(S)} y_s \leq c_e & & e \in E \\ & \sum_{S\subseteq T} y_s \leq \sum_{i\in T} \pi_i & & T \subset V;\ r \neq T \\ & y_s \geq 0 & & S \subset V;\ r \neq S. \end{aligned} \tag{3.2}$$

$\delta(S)$ es el conjunto de aristas con exactamente un vértice en S y $\chi(F) = \sum_{e\in F} x_e$. En el problema dual tenemos dos importantes restricciones que son la base del algoritmo, la primera ($\sum_{S:e\in\delta(S)} y_s \leq c_e \quad e \in E$) restringe que para toda arista la suma de las variables duales de los cortes que cruza dicha arista no pueden superar el costo de la arista. La segunda restricción ($\sum_{S\subseteq T} y_s \leq \sum_{i\in T} \pi_i \quad T \subset V;\ r \neq T$) expresa que dado un subconjunto T de V, la sumatoria de las variables duales de todos los subconjuntos incluidos en T no debe superar la sumatoria de las penalizaciones

producidas por los nodos incluidos en T.

El algoritmo se divide en tres grandes fases:

- Una primera fase de inicialización de las estructuras que serán necesarias en la ejecución del algoritmo.
- La segunda es la de crecimiento, se agregan aristas a un conjunto que al principio es vacío.
- La última es la de poda, donde se quitan del conjunto solución aristas redundantes, el árbol resultante de esta etapa es la solución final.

El algoritmo sigue el esquema primal-dual con sincronización. A continuación se presenta el algoritmo en forma de pseudocódigo, primero se presentan las etapas y luego cada una de ellas en específico.

Algoritmo 3: Algoritmo GW

Entrada: Grafo no dirigido $G = (V, E)$, con aristas de costo $c_{ij} \geq 0$, la penalización de cada vértice $\pi_i \geq 0$ y el nodo raíz r.

Resultado: Un árbol F', que incluye el nodo raíz r y un conjunto X que corresponden a los vértices no alcanzados por F'

1 $(F, y_s, \Delta, d(v), w(\{v\})) \leftarrow$ Inicialización(G);
2 $F \leftarrow$ Crecimiento$(F, y_s, \Delta, d(v), w(\{v\})$;
3 $F' \leftarrow$ Poda(F);

Se definen estructuras que se utilizan en el algoritmo y algunos conceptos del mismo.

- Se define una componente como un conjunto de nodos.
- Se define Δ como un conjunto de componentes.
- Por cada nodo $i \in V$ se define d_i variable que guarda de forma acumulada el valor de las variables duales de las componentes a las que pertenece el nodo i $(\sum_{\delta: i \in \delta} y_\delta)$.

- Para cada componente δ se define w_δ variable que guarda de forma acumulada el valor de las variables duales de las componentes incluidas en δ ($\sum_{\alpha:\alpha\subseteq\delta} y_\alpha$).
- Dada una componente δ, se define el potencial de la componente como $\sum_{v\in\delta} \pi(v) - w_\delta$
- Para cada componente δ se define $\lambda(\delta)$ la cual toma el valor 0 si el nodo raíz pertenece a la componente o si la componente perdió el potencial (es decir, que su potencial es 0), si no se cumple ninguna de las dos condiciones anteriores, el valor de $\lambda(\delta)$ es 1 y además se dice que la componente está activa.
- Dada una arista $e = (v_i, v_j)$ y dos componentes δ_p y δ_q, con $v_i \in \delta_p$ y $v_j \in \delta_q$ se define $\epsilon_{1e} = \frac{c_e - d_{v_i} - d_{v_j}}{\lambda(\delta_p)+\lambda(\delta_q)}$.
- Dada una componente δ se define el potencial remanente de dicha componente como $\epsilon_{2\delta} = (\sum_{i:i\in\delta} \pi_i) - w(\delta)$.

A continuación se presentan cada una de las tres etapas del algoritmo.

Algoritmo 4: Algoritmo GW (Inicialización)

Entrada: Grafo G

Resultado: $F, y_s, \Delta, d(v), w(\{v\})$

```
1  F ← ∅;
2  Implícitamente se le asignan a todas las variables y_s el valor 0;
3  Δ ← {{v} : v ∈ V};
4  para v ∈ V hacer
5      d(v) ← 0;
6      w({v}) ← 0;
7      si v = r entonces
8          λ(v) ← 0;
9      en otro caso
10         λ(v) ← 1;
11     fin
12 fin
```

Luego de la inicialización comienza la fase de crecimiento, esta básicamente consta de ir incrementando las variables duales sin violar ninguna de las restricciones del problema dual y continúa hasta que no quede ninguna componente activa.

Algoritmo 5: Algoritmo GW (Crecimiento)

Entrada: $F, y_s, \Delta, d(v), w(\{v\})$

Resultado: F

mientras $\exists \delta \in \Delta : \lambda(\delta) = 1$ **hacer**
- Se busca $\epsilon_1 = min(\epsilon_{1_e})\ \forall\ e = (v_i, v_j)$ y $v_i \in \delta_p \in \Delta, v_j \in \delta_q \in \Delta$;
- Se busca $\epsilon_2 = min(\epsilon_{2_{\delta_j}})\ \forall\ \delta_j \in \Delta : \lambda(\delta) = 1$;
- $\epsilon = min(\epsilon_1, \epsilon_2)$;
- $w(\delta) = w(\delta) + \lambda(\delta) \cdot \epsilon\ \forall\ \delta \in \Delta$;
- $y(\delta) = y(\delta) + \lambda(\delta) \cdot \epsilon\ \forall\ \delta \in \Delta$;
- $d_v = d_v + \lambda(\delta_i) \cdot \epsilon\ \forall\ v \in \delta_i \in \Delta$;
- **si** $\epsilon_2 \leq \epsilon_1$ **entonces**
  - $\lambda(\delta_j) = 0$
  - Se marcan todos los vértices de δ_j con la etiqueta j.
- **en otro caso**
  - $F = F \cup \{e\}$;
  - $\Delta = \Delta \cup \{\delta_p \cup \delta_q\} - \delta_p - \delta_q$;
  - $w(\delta_p \cup \delta_q) = w(\delta_p) + w(\delta_q)$;
  - **si** $r \in \delta_p \cup \delta_q$ **entonces**
    - $\lambda(\delta_p \cup \delta_q) = 0$
  - **en otro caso**
    - $\lambda(\delta_p \cup \delta_q) = 1$
  - **fin**
- **fin**

fin

Luego, como última fase del algoritmo se encuentra la poda. Para comprender esta etapa, para todo $v \in F$, se define $nw(v)$ como el aporte del sub-árbol con raíz v a la solución. El aporte es la penalización de los nodos del sub-árbol menos el costo de las aristas incluidas en el sub-árbol. Luego todo sub-árbol con raíz v tal que su

aporte sea menor o igual que 0, se quita de la solución, ya que el mismo no aporta nada a la misma o en algún caso puede influir negativamente. Para ello, [Johnson et al., 2000] propone un algoritmo recursivo que en el proceso calcula los valores $nw(v)$ comenzando desde las hojas y subiendo, cuando encuentra que alguno de estos valores es menor o igual a 0,quita el sub-árbol correspondiente de la solución. Al comienzo del algoritmo se define $nw(v) = \pi_v$ para todo los nodos del árbol F.

A continuación se presenta el pseudocódigo de dicho algoritmo.

Algoritmo 6: Algoritmo GW (Strong Pruning)

Entrada: F

Resultado: F'

```
1  para v ∈ V hacer
2  |  nw(v) = π_v;
3  fin
4  Función StrongPrune (v):
5  |  para u / u es hijo de v hacer
6  |  |  StrongPrune (u);
7  |  |  si c({v, u}) ≥ nw(u) entonces
8  |  |  |  Eliminar la arista {v, u};
9  |  |  |  Eliminar el subárbol de T' que tiene como raíz a u;
10 |  |  en otro caso
11 |  |  |  nw(v) = nw(v) + nw(u) − c({v, u});
12 |  |  fin
13 |  fin
```

3.2. Factor 2

En la presente sección se presenta la demostración formal del factor de aproximación del algoritmo *Goemans-Williamson*, dicha demostración está basada en el documento [Goemans y Williamson, 1995]. Previo a la demostración principal se demuestran algunas proposiciones que serán útiles en la misma y se definen algunas variables que serán utilizadas a lo largo de la demostración.

Se denota Z^*_{PL} y Z^*_{PLR} como la solución óptima de la Ecuación 1.2 y la Ecuación 3.1 respectivamente. Luego, si se considera una solución factible del problema dual relajado, por el Teorema 2 se tiene: $\sum_{S \subseteq V} y_s \leq Z^*_{PLR} \leq Z^*_{PL}$.

Por otra parte se observa que por construcción todo vértice que no es alcanzado por F' forma parte de algún subconjunto que es desactivado (por la restricción de potencial) en algún paso del algoritmo, se define X como el conjunto de dichos vértices. El algoritmo etiqueta los nodos al desactivar un subconjunto por falta de potencial, entonces el conjunto X se puede dividir de acuerdo a sus etiquetas en subconjuntos disjuntos $C_1, C_2, ..., C_k$.

Definición 3.2.1. *Grafo H*
En el inicio de cada iteración del algoritmo se puede definir un grafo H de la siguiente forma:

- *Los nodos del grafo H son las componentes inactivas y activas existentes al inicio de la iteración.*
- *Las aristas del grafo H son las aristas del grafo original que cruzan cada componente activa intersección con las aristas de la solución brindada por el algoritmo:* $e \subset \delta(C) \cap F'$.
- *No se toman en cuenta las componentes inactivas que estén aisladas.*
- *Se define* N_a *como el conjunto de vértices correspondientes a componentes activas.*
- *Se define* N_i *como el conjunto de vértices correspondientes a componentes inactivas.*
- *Se define* d_v *como el grado del vértice v en el grafo H.*
- *Se define* N_d *como el conjunto de vértices correspondientes a componentes activas que al final del algoritmo estarán contenidas en alguna de las componentes* C_j. *Se puede observar que los vértices pertenecientes a* N_d *son los correspondientes a componentes activas y cuyo grado es 0:* $N_d = \{v \in N_a : d_v = 0\}$.

Lema 3.2.1. *Si se considera un incremento ϵ de las variables duales en un paso cualquiera del algoritmo, el incremento de*

$$\sum_S y_S |F' \cap \delta(S)| + \sum_j \sum_{S \subseteq C_j} y_S$$

en función del grafo H está dado por:

$$\epsilon \cdot ((\sum_{v \in N_a - N_d} d_v) + |N_d|)$$

y el incremento de

$$(2 - \frac{1}{n-1}) \sum_{S \subset V} y_S$$

en función del grafo H está dado por:

$$\epsilon \cdot (2 - \frac{1}{n-1}) \cdot |N_a|$$

Prueba. El incremento de $\sum_S y_S |F' \cap \delta(S)| + \sum_j \sum_{S \subseteq C_j} y_S$ tiene dos componentes:

- $\sum_j \sum_{S \subseteq C_j} y_s$ la cual corresponde con el incremento de las variables duales de las componentes correspondientes a los nodos N_d del grafo H. Una vez transcurrida la iteración correspondiente del algoritmo, estas componentes incrementarán sus variables duales en ϵ, por lo cual el incremento total de esta componente es de $\epsilon \cdot |N_d|$. Esto se debe a que el algoritmo incrementará en ϵ todas las variables duales correspondientes a componentes activas y por definición de H las componentes activas que al finalizar el algoritmo estarán en algún C_j son las correspondientes a los vértices N_d del grafo H.

- $\sum_S y_S |F' \cap \delta(S)|$ la cual está asociada con el incremento de las variables duales de las componentes correspondientes a los nodos N_a del grafo H que tienen alguna arista incidente, por lo cual se puede expresar su incremento como $\epsilon \cdot (\sum_{v \in N_a} d_v)$. Esto se debe a que si se observa la expresión, se puede ver que por cada componente que incrementemos el valor ϵ, este incremento será multiplicado por la cantidad de aristas pertenecientes a F' que corten dicha componente. Por la definición de H se corresponde con el grado d_v siendo v el vértice correspondiente en H a dicha componente.

Entonces se puede definir el incremento como $\epsilon \cdot ((\sum_{v \in N_a} d_v) + |N_d|)$, además, como el grado de los vértices N_d de H es 0 el incremento queda determinado por

$$\epsilon \cdot ((\sum_{v \in N_a - N_d} d_v) + |N_d|)$$

Luego se tiene que determinar el incremento de $(2 - \frac{1}{n-1}) \sum_{S \subset V} y_S$. Se puede observar que esta expresión puede reescribirse como:

$$(2 - \frac{1}{n-1})(\sum_{S \subset V : \lambda(S)=1} y_S + \sum_{S \subset V : \lambda(S)=0} y_S)$$

por lo cual si se considera un incremento ϵ de las variable y_S de las componentes activas, se tiene:

$$(2 - \frac{1}{n-1})(\sum_{S \subset V : \lambda(S)=1} (y_S + \epsilon) + \sum_{S \subset V : \lambda(S)=0} y_S)$$

para finalizar se desarrolla y se observa que el incremento es:

$$\epsilon \cdot (2 - \frac{1}{n-1}) \cdot (\sum_{S \subset V : \lambda(S)=1} 1)$$

lo que por construcción de H corresponde con

$$\epsilon \cdot (2 - \frac{1}{n-1}) \cdot |N_a|$$

□

Lema 3.2.2. *En el grafo H definido en la Definición 3.2.1 se cumple:*

$$\sum_{v \in N_a - N_d} d_v \leq (2 - \frac{1}{n-1})|N_a - N_d|$$

Prueba. Considerando que $(N_a - N_d) \cap N_i = \emptyset$, y los siguientes conjuntos de aristas, $E_1 = \{e \in N_a - N_d\}$, $E_2 = \{e \in N_i\}$ y $E_3 = \{e \in corte(N_a - N_d, N_i)\}$. Por Handshaking la suma de los grados de los vértices cumple que $2(|E_1| + |E_2| + |E_3|) = (2|E_1| + |E_3|) + (2|E_2| + |E_3|)$ donde el lado izquierdo de la igualdad corresponde a la suma de los grados de los vértices del grafo H y el lado derecho a la suma de los grados de los vértices de los sub-grafos $N_a - N_d$ y N_i. Sustituyendo en la fórmula de grados se obtiene $\sum_{v \in (N_a - N_d) \cup N_i} d_v = \sum_{v \in N_a - N_d} d_v + \sum_{v \in N_i} d_v$, luego reordenando los factores tenemos:

$$\sum_{v \in N_a - N_d} d_v = \sum_{v \in (N_a - N_d) \cup N_i} d_v - \sum_{v \in N_i} d_v$$

Teniendo además las siguientes consideraciones:

- Por construcción del algoritmo no puede existir una componente inactiva distinta de aquella que contenga al nodo r cuyo vértice correspondiente en H tenga grado 1, ya que de lo contrario seria una componente que debería ser removida en la etapa de *pruning* del algoritmo.
- Teniendo en cuenta la consideración anterior, se observa que $(2|N_i|-1)$ es cota inferior de $\sum_{v \in N_i} d_v$.
- $\sum_{v \in (N_a - N_d) \cup N_i} d_v$ corresponde con la suma de los grados de los vértices de un árbol y dado que en cualquier árbol la cantidad de aristas es la cantidad de nodos - 1, se puede reescribir esto como $2(|(N_a - N_d) \cup N_i| \ - \ 1)$.
- Teniendo en cuenta las consideraciones anteriores, se tiene como cota superior de la expresión $\sum_{v \in (N_a - N_d) \cup N_i} d_v - \sum_{v \in N_i} d_v$ el valor $2(|(N_a - N_d) \cup N_i| \ - \ 1) - (2|N_i| - 1)$.
- En cualquier etapa del algoritmo $|N_a|$ es a lo sumo $(n-1)$, en particular, esto se da en la primera iteración del algoritmo, luego el número de componentes activas va decreciendo. Por lo cual, $(|N_a - N_d|)$ tiene como cota superior $(n-1)$ y por lo tanto $(\frac{1}{|N_a - N_d|})$ tiene como cota inferior $(\frac{1}{n-1})$ y finalmente $(2 - \frac{1}{|N_a - N_d|})$ tiene como cota superior $(2 - \frac{1}{n-1})$.

Luego de tener en cuenta las consideraciones anteriores, se puede deducir lo siguiente:

$$\begin{aligned}
\sum_{v \in N_a - N_d} d_v &\leq 2(|(N_a - N_d) \cup N_i| \ - \ 1) - (2|N_i| - 1) \\
&= 2|N_a - N_d| \ - \ 1 \\
&= 2|N_a - N_d| \ - \ \frac{|N_a - N_d|}{|N_a - N_d|} \\
&= |N_a - N_d|(2 - \ \frac{1}{|N_a - N_d|}) \\
&\leq |N_a - N_d|(2 - \frac{1}{n-1})
\end{aligned}$$

□

Teorema 5 (Teorema PCST Factor 2)**.** *El algoritmo presentado en la Sección 3.1 genera un conjunto de vértices V' y un conjunto de aristas incidentes sobre los mismos F', así como un conjunto de vértices X no alcanzados por F', de forma tal que el árbol $G' = (V', F')$ y dicho conjunto de vértices X son solución factible de la Ecuación 1.2 y cumplen:*

$$\sum_{e \in F'} c_e + \sum_{i \in X} \pi_i \leq (2 - \frac{1}{n-1}) \sum_{S \subset V} y_s \leq (2 - \frac{1}{n-1}) Z^*_{PL}$$

Por lo tanto el algoritmo tiene factor de aproximación $(2 - \frac{1}{n-1})$ para el PCST.

Prueba. Ya que los componentes C_j fueron desactivados por la restricción de potencial entonces se cumple que $\sum_{S \subseteq C_j} y_s = \sum_{i \in C_j} \pi_i$, además, para las aristas que forman parte de la solución final F' están *tight* por lo que se cumple $\sum_{S: e \in \delta(S)} y_s = c_e$. Considerando ambas igualdades, se puede reescribir la ecuación del Teorema 5 de la siguiente manera:

$$\sum_{e \in F'} \sum_{S: e \in \delta(S)} y_s + \sum_j \sum_{S \subseteq C_j} y_s \leq (2 - \frac{1}{n-1}) \sum_{S \subset V} y_s$$

Al igual que en el teorema 2.4 de [Goemans y Williamson, 1995], se puede observar que en $\sum_{e \in F'} \sum_{S: e \in \delta(S)} y_s$ se suma tantas veces la variable dual de una componente S como aristas pertenecientes a F' corten dicha componente. Entonces se pueden reescribir los términos de la siguiente forma:

$$\sum_S y_S |F' \cap \delta(S)| + \sum_j \sum_{S \subseteq C_j} y_S \leq (2 - \frac{1}{n-1}) \sum_{S \subset V} y_S \tag{3.3}$$

Luego se puede probar el teorema por inducción en el ciclo principal del algoritmo. Por el Lema 3.2.1 se puede determinar que ante un incremento ϵ de las variables duales los incrementos en la parte izquierda y derecha de la Ecuación 3.3 están dados por $\epsilon \cdot ((\sum_{v \in N_a - N_d} d_v) + |N_d|)$ y $\epsilon \cdot (2 - \frac{1}{n-1}) \cdot |N_a|$ respectivamente, por lo cual se debe demostrar que en cada iteración se cumple:

$$\epsilon \cdot ((\sum_{v \in N_a - N_d} d_v) + |N_d|) \leq \epsilon \cdot (2 - \frac{1}{n-1}) \cdot |N_a|$$

reordenando los términos se tiene lo siguiente:

$$\sum_{v \in N_a - N_d} d_v \leq (2 - \frac{1}{n-1}) |N_a - N_d| \leq (2 - \frac{1}{n-1}) |N_a - \frac{N_d}{(2 - \frac{1}{n-1})}|$$

para completar la demostración se debe probar que para todo grafo H se cumple:

$$\sum_{v \in N_a - N_d} d_v \leq (2 - \frac{1}{n-1})|N_a - N_d|$$

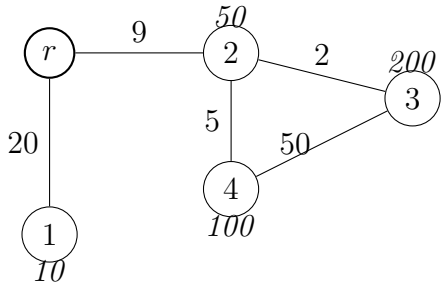
Luego por el Lema 3.2.2 se sabe que esta afirmación es cierta para todo H, por lo cual queda demostrado el teorema.

□

3.3. Ejemplos

En esta sección se presenta un ejemplo de la aplicación del algoritmo $Goemans-Williamson$.

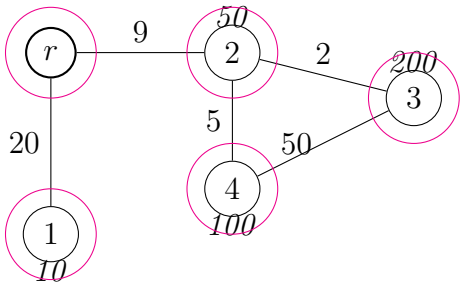
Para cada iteración se muestra el grafo con las componentes, los cálculos de ϵ_1 y ϵ_2 así como también las aristas que se van agregando a F, la evolución de d y en la tabla se incluye el valor de w, de y (variable dual) y de λ que indica si está o no activa la componente.



Se inicializan las variables y componentes.

$F = \{\}$. $d(r) = 0,\ d(1) = 0,\ d(2) = 0,\ d(3) = 0,\ d(4) = 0$

	{r}	{1}	{2}	{3}	{4}
w	0	0	0	0	0
y	0	0	0	0	0
λ	0	1	1	1	1



Primera iteración:

$\epsilon_1 = min\{\frac{20-0-0}{0+1}, \frac{9-0-0}{0+1}, \frac{2-0-0}{1+1}, \frac{5-0-0}{1+1}, \frac{50-0-0}{1+1}\} = 1{,}0$

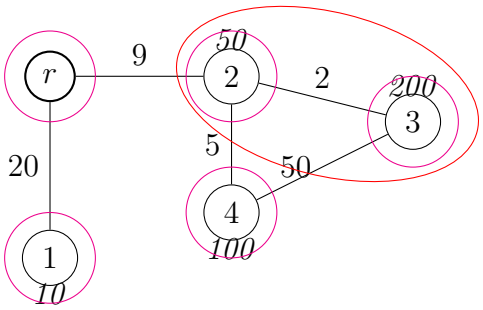
$\epsilon_2 = min\{10-0, 50-0, 200-0, 100-0\} = 10{,}0$

$\epsilon = min\{1{,}0, 10{,}0\} = 1{,}0$

La arista agregada es $(2,3)$ y las componentes $\{2\}$ y $\{3\}$ se unen para formar $\{2,3\}$.

$F = \{(2,3)\} \quad d(r) = 0,\ d(1) = 1,\ d(2) = 1,\ d(3) = 1,\ d(4) = 1$

	{r}	{1}	{2}	{3}	{4}	{2,3}
w	0	1	1	1	1	2
y	0	1	1	1	1	0
λ	0	1	0	0	1	1



Segunda iteración:

$\epsilon_1 = min\{\frac{20-0-1}{0+1}, \frac{9-0-1}{0+1}, \frac{5-1-1}{1+1}, \frac{50-1-1}{1+1}\} = 1{,}5$

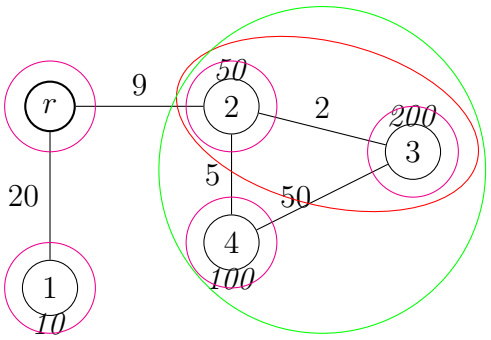
$\epsilon_2 = min\{10-1, (200+50)-2, 100-1\} = 9$

$\epsilon = min\{1{,}5, 9\} = 1{,}5$

La arista agregada es $(2,4)$ y las componentes $\{2,3\}$ y $\{4\}$ se unen para formar $\{2,3,4\}$.

$F = \{(2,3),(2,4)\}$ $d(r) = 0,\ d(1) = 2{,}5,\ d(2) = 2{,}5,\ d(3) = 2{,}5,\ d(4) = 2{,}5$

	{r}	{1}	{2}	{3}	{4}	{2,3}	{2,3,4}
w	0	2.5	1	1	2.5	3.5	6
y	0	2.5	1	1	2.5	1.5	0
λ	0	1	0	0	0	0	1



Tercer iteración:

$\epsilon_1 = min\{\frac{20-0-2{,}5}{0+1}, \frac{9-0-2{,}5}{0+1}\} = 6{,}5$

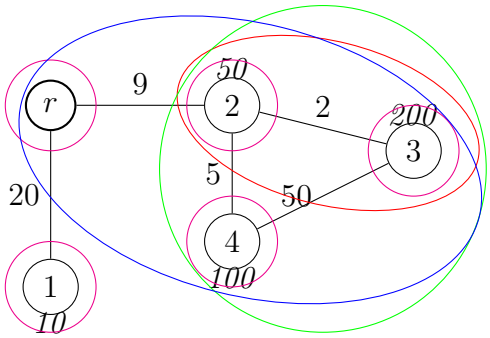
$\epsilon_2 = min\{10 - 2{,}5, (200 + 50 + 100) - 6\} = 7{,}5$

$\epsilon = min\{6{,}5, 7{,}5\} = 6{,}5$

La arista agregada es $(r,2)$ y las componentes $\{r\}$ y $\{2,3,4\}\{4\}$ se unen para formar $\{r,2,3,4\}$.

$F = \{(2,3),(2,4),(r,2)\}$ $d(r) = 0,\ d(1) = 9,\ d(2) = 9,\ d(3) = 9,\ d(4) = 9$

	{r}	{1}	{2}	{3}	{4}	{2,3}	{2,3,4}	{r,2,3,4,}
w	0	9	1	1	2.5	3.5	12.5	12.5
y	0	9	1	1	2.5	1.5	6.5	0
λ	0	1	0	0	0	0	0	0



Cuarta iteración:

$\epsilon_1 = min\{\frac{20-0-9}{0+1}\} = 11$

$\epsilon_2 = min\{10-9\} = 1{,}0$

$\epsilon = min\{11{,}0, 1{,}0\} = 1{,}0$

No se agregan aristas a la solución ya que la componente $\{1\}$ se desactiva por la restricción de potencial.

$F = \{(2,3),(2,4),(r,2)\} \quad d(r) = 0,\ d(1) = 10,\ d(2) = 9,\ d(3) = 9,\ d(4) = 9$

	{r}	{1}	{2}	{3}	{4}	{2,3}	{2,3,4}	{r,2,3,4,}
w	0	10	1	1	2.5	3.5	12.5	12.5
y	0	10	1	1	2.5	1.5	6.5	0
λ	0	0	0	0	0	0	0	0

La solución es:

$F' = \{(2,3),(2,4),(r,2)\}$

$X = \{1\}$

Capítulo 4

Aleatorización

4.1. Introducción

En el presente capítulo se estudiará otro algoritmo para encontrar una solución al problema PCST. Este algoritmo agrega una componente aleatoria lo cual es una novedad con respecto al estudiado en el Capítulo 3. Se demuestra que tiene un factor de aproximación de 2.54.

4.2. Algoritmos Aleatorios

En algunos casos puede ser interesante permitir que los algoritmos tomen decisiones aleatorias. De este modo la garantía de aproximación del algoritmo dependerá de las posibles opciones aleatorias del algoritmo. Esta clase de algoritmos con aleatorización podrían parecer más débiles en cuanto a la garantía de aproximación, sin embargo en la mayoría de los casos a partir de estos se pueden construir versiones deterministas de los algoritmos con la misma garantía de aproximación que la versión aleatoria. En general la garantía de aproximación se puede conseguir utilizando el *método de las esperanzas condicionales*. Considerando lo mencionado anteriormente no pareciera tener mucho sentido utilizar algoritmos con aleatorización, sin embargo, muchas veces es más fácil establecer y analizar una versión de este tipo que la versión determinista que resulta de la construcción antes mencionada [Williamson

y Shmoys, 2011].

La técnica de *randomized rounding* fue presentada por [Raghavan y Tompson, 1987]. De acuerdo con su formulación original, dado Π_I un problema de programación lineal 0-1, cuyas variables $x_i \in \{0, 1\}$, se define Π_R como el problema relajado con variables $x_i \in [0, 1]$. Sus dos fases consisten en, primero resolver Π_R, donde las variables toman los valores $\hat{x_i} \in [0, 1]$, luego como segunda fase se asignan valores cero o uno a x_i de forma aleatoria de acuerdo con la siguiente regla: $Prob.[x_i = 1] = \hat{x_i}$.

El libro *Randomized Algorithms* de [Motwani y Raghavan, 1995] cuenta con una extensa lista de aplicaciones de esta técnica a múltiples problemas de diversas áreas, por ejemplo: estructuras de datos, algoritmos en grafos y algoritmos de teoría de números.

4.3. Aleatorización en PCST

En el caso del PCST es posible aplicar el método de redondeo aleatorio pero con algunas modificaciones. En este problema no basta con redondear los vectores resultado ya que al aplicar este redondeo a ambos vectores el resultado podría quedar inconsistente. Por ejemplo, podrían quedar nodos en la solución pero sin aristas que lo conecten. Una forma de resolver este problema es la planteada en [Williamson y Shmoys, 2011], a continuación se presenta un pseudocódigo del algoritmo:

Algoritmo 7: Algoritmo con aleatorización

Entrada: Grafo no dirigido $G = (V, E)$, con aristas de costo $c_{ij} \geq 0$, la penalización de cada vértice $\pi_i \geq 0$ y el nodo raíz r.

Resultado: Un árbol F', que incluye el nodo raíz r

1 Se determina (X^*, S^*) solución de la relajación lineal. ;

2 Se elije α de forma aleatoria y uniforme en el intervalo $[\gamma, 1)$ con $0 \leq \gamma < 1$;

3 Se determina el conjunto de nodos terminales $U = \{v \in V : s_v^* \geq \alpha\}$;

4 Se resuelve el problema de Steiner típico donde se consideran como nodos terminales los pertenecientes a U y el resto como nodos de Steiner;

Existen múltiples maneras de resolver el primer paso del algoritmo, el planteo original de [Williamson y Shmoys, 2011] es utilizar el método del elipsoide. Dado que no se dispone de una implementación del método del elipsoide, se decide utilizar la herramienta *CPLEX*, la cual brinda una librería capaz de resolver problemas de programación linear tanto entera como mixta. Luego para el último paso, se utilizará el esquema primal dual para la resolución del problema de Steiner típico. En particular, dado que es un subproblema del PCST se puede utilizar el algoritmo descripto en el Capítulo 3. En consecuencia de que el modelo ILP planteado en [Williamson y Shmoys, 2011] contiene una cantidad de restricciones exponenciales en el orden de los nodos, no es posible implementar el mismo de forma directa en *CPLEX*, por lo cual se buscó un modelo alternativo. Para resolver el problema se utiliza un modelo ILP que se basa en una transformación previa del grafo, dicha transformación consiste en pasar el grado no dirigido a uno dirigido según lo mostrado en [Ljubić et al., 2006]. Luego de la transformación se construye un modelo basado en flujos según lo estudiado en [Segev, 1987] adaptado al problema PCST. A continuación se presenta la transformación y el modelo mencionado.

Para construir el modelo de ILP basado en flujos se transforma el grafo original $G = (V, E)$ en un $G_{SA} = (V_{SA}, A_{SA}, c'')$. El conjunto $V_{SA} = V \cup \{r\}$, r es un nodo raíz artificial. El conjunto de aristas A_{SA} está formado por aristas dirigidas (i, j) y (j, i) para toda arista $(i, j) \in E$ más las aristas desde el nodo raíz r hasta los vértices $R_{SA} = \{i \in V \mid \pi_i > 0\}$ (vértices del grafo original con peso mayor a cero). El costo c''_{ij} de las aristas se define de la siguiente manera:

$$c''_{ij} = \begin{cases} c_{ij} - \pi_j & \forall (i,j) \in A_{SA}, i \neq r \\ -\pi_j & \forall (r,j) \in A_{SA} \end{cases}$$

Se considera T_{SA} un árbol dirigido con raíz r que es subgrafo de G_{SA}. Si el grado del nodo raíz r es 1, entonces T_{SA} se corresponde con la solución del PCST.

El modelo ILP tiene como objetivo buscar el mínimo T_{SA}, para ello es necesario definir las siguientes variables: x_{ij} que vale 1 si la arista $(i, j) \in T_{SA}$ o cero si no y s_i que indica si el nodo i es tenido o no en cuenta en la solución. Se introduce una variable de flujo f_{ij} que indica el monto del flujo en la arista (i, j), para todo

$(i,j) \in A_{SA}$. Una unidad de flujo es enviado desde el nodo raíz a todos los nodos clientes de la solución, donde f_{ij} representa la suma de todos los flujos que pasan por la arista (i,j).

$$\min \sum_{ij \in A_{SA}} c''_{ij} x_{ij} + \sum_{i \in R_{SA}} \pi_i$$

sujeto a

$$\begin{aligned}
&\sum_{ji \in A_{SA}} x_{ji} = s_i && \forall i \in V_{SA} \setminus \{r\} \quad (1)\\
&\sum_{ji \in A_{SA}} f_{ji} - \sum_{ij \in A_{SA}} f_{ij} = s_i && \forall i \in R_{SA} \quad (2)\\
&\sum_{ji \in A_{SA}} f_{ji} - \sum_{ij \in A_{SA}} f_{ij} = 0 && \forall i \in V_{SA} \setminus R_{SA}, i \neq r \quad (3)\\
&0 \leq f_{ij} \leq (|V_{SA}| - 1) x_{ij} && \forall (i,j) \in A_{SA} \quad (4)\\
&\sum_{ri \in A_{SA}} x_{ri} = 1 && (5)\\
&s_i, x_{ij} \in \{0,1\} && \forall i \in V_{SA} \setminus \{r\}, \forall (i,j) \in A_{SA}.
\end{aligned} \tag{4.1}$$

La restricción (1) asegura que todo vértice seleccionado tiene exactamente un nodo predecesor en la ruta desde la raíz. (2) Para todos los nodos con penalización mayor que cero se debe cumplir que la diferencia entre flujo entrante y saliente es cero o uno dependiendo de si el nodo es considerado o no en la solución final. (3) Para todos los nodos con penalización cero el flujo entrante y saliente deben ser iguales. (4) Garantiza que si el flujo por una arista es mayor que cero, la arista debe estar incluida en la solución. Por último la (5) preserva que la solución final es conexa restringiendo el grado del nodo raíz en uno.

Se define el modelo ILP relajado simplemente cambiando la restricción $s_i, x_{ij} \in \{0,1\}$ por $0 \leq s_i \leq 1, 0 \leq x_{ij} \leq 1$.

4.4. Demostración Factor 2.54

Lema 4.4.1. *El árbol retornado por el paso 4 del algoritmo cumple lo siguiente:*

$$\sum_{e \in T} c_e \leq \frac{2}{\alpha} \sum_{e \in E} c_e x_e^*$$

Prueba. Se observa que la cota inferior de $\frac{2}{\alpha} \sum_{e \in E} c_e x_e^*$ es $2 \sum_{e \in E} c_e x_e^*$ ya que $\alpha \in [\gamma, 1)$ con $0 \leq \gamma < 1$. Como se considera el problema de Steiner típico, en este caso las penalizaciones de los nodos no se tienen en cuenta. Como se denota en la definición del problema expresada en la Sección 1.3, lo que se considera es el costo de las aristas, por lo cual si se puede encontrar un algoritmo de factor de aproximación 2 queda demostrado. Luego, dado que el problema de Steiner típico es un caso particular del PCST, si se aplica el algoritmo de la Sección 3.1 queda demostrado el Lema. □

Lema 4.4.2. *El árbol retornado por el paso 4 del algoritmo cumple lo siguiente:*

$$\sum_{i \in (V - V(T))} \pi_i \leq \frac{1}{1-\alpha} \sum_{i \in V} \pi_i (1 - s_i^*)$$

Prueba. Si el nodo i no pertenece al árbol T entonces tampoco pertenece al conjunto de terminales U, por lo que se cumple que $s_i^* < \alpha$. Entonces $1 - s_i^* > 1 - \alpha$, por lo tanto $\frac{1-s_i^*}{1-\alpha} > 1$ □

Lema 4.4.3.

$$E\left[\sum_{e \in T} c_e\right] \leq \left(\frac{2}{1-\gamma} ln \frac{1}{\gamma}\right) \cdot \sum_{e \in E} c_e x_e^*$$

Prueba. Si se aplica el valor esperado en ambos lados de la desigualdad del Lema 4.4.1 y luego calculando se tiene lo siguiente:

$$\begin{aligned} E\left[\sum_{e\in T} c_e\right] &\leq E\left[\frac{2}{\alpha}\sum_{e\in E} c_e x_e^*\right] \\ &= E\left[\frac{2}{\alpha}\right]\cdot\sum_{e\in E} c_e x_e^* \\ &= \left(\frac{1}{1-\gamma}\int_\gamma^1 \frac{2}{x}dx\right)\cdot\sum_{e\in E} c_e x_e^* \\ &= \left[\frac{2}{1-\gamma}ln(x)\right]_\gamma^1\cdot\sum_{e\in E} c_e x_e^* \\ &= \left(\frac{2}{1-\gamma}ln\frac{1}{\gamma}\right)\cdot\sum_{e\in E} c_e x_e^* \end{aligned}$$

□

Lema 4.4.4.

$$E\left[\sum_{i\in(V-V(T))}\pi_i\right] \leq \frac{1}{1-\gamma}\sum_{i\in V}\pi_i(1-s_i^*)$$

Prueba. Dado $U = \{i \in V : s_i^* \geq \alpha\}$, todo vértices que no esté en el árbol no está en U (ya que los elementos de U son los nodos terminales), entonces se cumple $\sum_{i\in(V-V(T))}\pi_i \leq \sum_{i\notin U}\pi_i$. Observar que si $s_i^* \geq \gamma$, entonces la probabilidad de que $i \notin U$ es $(1-s_i^*)/(1-\gamma)$. Si $s_i^* < \gamma$, entonces $i \notin U$ con probabilidad 1. Pero entonces se cumple $1 \leq (1-s_i^*)/(1-\gamma)$ □

Teorema 6. *El costo esperado de la solución entregada por el algoritmo 7 es:*

$$E\left[\sum_{e\in T} c_e + \sum_{i\in(V-V(T))}\pi_i\right] \leq \left(\frac{2}{1-\gamma}ln\frac{1}{\gamma}\right)\sum_{e\in E} c_e x_e^* + \frac{1}{1-\gamma}\sum_{i\in V}\pi_i(1-s_i^*)$$

Prueba. El teorema queda demostrado al aplicar el Lema 4.4.3, Lema 4.4.4 y por la propiedad de esperanza matemática que dice: si X e Y son variables aleatorias se cumple que: $E[X+Y] = E[X] + E[Y]$. □

Corolario 4.4.1. *Utilizando el algoritmo 7 con* $\gamma = e^{-1/2}$ *se obtiene un factor de aproximación de* $\frac{1}{1-e^{-1/2}} \approx 2{,}54$

Prueba. El objetivo es encontrar un valor de γ tal que el valor máximo de $\{\frac{2}{1-\gamma} ln \frac{1}{\gamma}, \frac{1}{1-\gamma}\}$ sea lo más pequeño posible. Para ello observar que $\frac{2}{1-\gamma} ln \frac{1}{\gamma}$ es decreciente y $\frac{1}{1-\gamma}$ es creciente en función de γ. Para minimizar se iguala $\frac{2}{1-\gamma} ln \frac{1}{\gamma} = \frac{1}{1-\gamma}$, luego se obtiene que $\gamma = e^{-1/2}$. Por lo tanto por el Teorema 6 el costo esperado del árbol solución no supera a:

$$\frac{1}{1-e^{-1/2}} \left(\sum_{e \in E} c_e x_e^* + \sum_{i \in V} \pi_i (1 - s_i^*) \right) \leq \frac{1}{1-e^{-1/2}} . OPT \leq (2{,}54) . OPT$$

□

Capítulo 5

Análisis Experimental

5.1. Introducción

El presente capítulo está enfocado en el estudio de los resultados de las ejecuciones de los algoritmos presentados en capítulos anteriores. Se especifica el entorno y cuales son los casos de prueba utilizados. Se presenta un análisis preliminar del algoritmo aleatorio y una posible variante a la aleatorización. Luego en las últimas secciones se analiza de forma detallada los múltiples enfoques utilizando comparativas de tiempo y valores objetivos.

5.2. Medidas de Méritos

En la presente sección se presentan algunas definiciones que son utilizadas en el capítulo.

Definición 5.2.1 (Densidad de un Grafo)**.** *Dado un grafo $G = (V, E)$ se define su densidad como:*

$$d(G) = \frac{2|E|}{|V|(|V| - 1)}$$

Este concepto de densidad fue extraído de [Coleman y Moré, 1983].

Definición 5.2.2 (Promedio de Variables Asociadas a las Aristas en modelo ILP Relajado)**.** *Dada la versión relajada del modelo planteado en la Ecuación 4.1, se*

define:

$$promAristas = \frac{\sum_{x_{ij}>0} x_{ij}}{\sum_{x_{ij}>0} 1}$$

Definición 5.2.3 (Mínimo de Variables Asociadas a las Aristas en modelo ILP Relajado)**.** *Dada la versión relajada del modelo planteado en la Ecuación 4.1, se define:*

$$minAristas = min\{x_{ij} : x_{ij} > 0\}$$

Definición 5.2.4 (GAP)**.** *El gap porcentual de un valor X con respecto a un valor Y se define en la siguiente fórmula:*

$$gap = \frac{(X - Y) \cdot 100}{Y}$$

Definición 5.2.5 (Medida *cna*)**.** *Dado un grafo $G = (V, E, \pi_v, c_e)$ según se especifica en la Ecuación 1.1, se define cna como el coeficiente entre las penalizaciones promedio de los nodos y los costos promedios de las aristas:*

$$cna = \frac{(\sum_{v \in V} \pi_v)/|V|}{(\sum_{e \in E} c_e)/|E|}$$

Definición 5.2.6 (Medida *pnp*)**.** *Dado un grafo $G = (V, E, \pi_v, c_e)$ según se especifica en la Ecuación 1.1, se define pnp como la proporción de nodos con penalización mayor que cero con respecto a la cantidad de total de nodos:*

$$pnp = \frac{|\{v \in V : \pi_v > 0\}|}{|V|}$$

5.3. Entorno e Instancias de Pruebas

5.3.1. Entorno de Pruebas

Para realizar las pruebas se utilizó un servidor virtual contratado en *DigitalOcean* [1] que cuenta con las siguientes características: 8 GB de memoria RAM, 4 CPU's Intel(R) Xeon(R) CPU E5-2650L v3 @ 1.80GHz y el sistema operativo utilizado es la distribución de Linux Ubuntu 14.04. El programa fue desarrollado utilizando el lenguaje C++ y compilado con la versión de *gcc 4.8.4* y la versión de *IBM ILOG CPLEX Optimization Studio V12.6.3. CPLEX* es la herramienta utilizada para resolver en forma exacta el modelo ILP relajado 4.1.

[1] https://www.digitalocean.com/

5.3.2. Instancias

Las instancias están en el formato STP (*Steiner Tree Problem File*) [2], estándar para la representación de grafos para el problema de Steiner y sus variantes.

Se obtienen desde la web de DIMACS [3] las cuales fueron propuestas para una competencia de algoritmos aplicados a problemas de grafos.
Instancias provistas a la competencia por Mauricio Resende.

- JMP: Instancias introducidas por Johnson et al. [2000]. Las del tipo P no tienen una estructura específica y las K son aleatorias que simulan un mapa de calles.
- CRR: Instancias aleatorias introducidas por Canuto et al. [2001], basadas en las familias C y D de *SteinLib* [4] y *OR-Library* [5].

Instancias provistas a la competencia por Biazzo et al. [2013].

- i640: Grafos aleatorios propuestos originalmente por Cees W. Duin de la Universidad de Amsterdam.
- H: Hipercubos propuestos originalmente por Rosseti et al. [2004]. Las instancias para el PCST fueron introducidas por Ljubić et al. [2006].
- H2: Hipercubos propuestos originalmente por Rosseti et al. [2004]. Similares a H.
- RANDOM: Grafos aleatorios.

Instancias provistas a la competencia por Gunnar W. Klau y Mohammed El-Kebir.

- ACTMODPC: Instancias a partir de redes de análisis biológicos donde el objetivo es encontrar conjuntos de genes que probablemente estén involucrados en algún mecanismo común de la célula. Fueron creadas por Daniel Juhl.

[2] http://steinlib.zib.de/format.php
[3] http://dimacs.rutgers.edu/
[4] http://steinlib.zib.de/steinlib.php
[5] http://people.brunel.ac.uk/ mastjjb/jeb/info.html

Instancias provistas a la competencia por Ljubić et al. [2006]

- Cologne: Instancias con nodo raíz asignado basadas en algunas ciudades alemanas utilizadas para el diseño de redes de fibra óptica.
- PUCNU: Instancias basadas en las PUC correspondientes el problema de Steiner típico. Todas las aristas tiene costo uno y la penalización de los terminales es de uno o dos.

Otras instancias tenidas en cuentas para los experimentos son generadas a partir de las topologías de las redes: ARPANET, RAU2, una red de parques eólicos de UTE prevista para 2023 e instancias aleatorias generadas por el programa implementado, así como otras provistas por [Berguer, 2017].

5.4. Análisis Preliminar

Se presenta un análisis de los resultados obtenidos en la solución relajada del modelo representado en la Ecuación 4.1. Los resultados analizados se corresponden con la salida del primer paso del algoritmo 7, en el cual se obtiene (x^*, s^*) que es solución óptima de la relajación. En la Tabla 5.1 se incluye un representante de los grupos de instancias utilizados en el experimento, cada entrada de la tabla indica la cantidad de nodos que el valor de s^* pertenece al intervalo indicado en la columna.

Tabla 5.1: Intervalos

Instancia	0	(0, 0.6065)	[0.6065, 1)	1
a0200RandGraph.1.2	152	15	0	33
bip42nu	1010	84	0	106
C01-A	477	18	0	5
hc10p	246	271	0	507
i101M1	735	11	0	2
i640-345	454	46	0	140
K100.1	74	16	0	10
lymphoma	1908	59	0	67

El estudio realizado en 297 instancias muestra que la cantidad de nodos cuya variable asociada s^* toma un valor continuo en el intervalo $(e^{-1/2} \approx 0{,}6065, 1)$ es

cero. Luego de constatar en base a estos experimentos que en todos los casos s^* no está en el intervalo determinado en el Corolario 4.4.1, es simple de observar que la aleatorización de α para la selección de nodos tal como lo indica el algoritmo 7 no va a tener incidencia en el resultado final. Ya que $\alpha \in [\gamma, 1)$ con $0 \leq \gamma < 1$ y al aplicar la selección de nodos de acuerdo a $\{v \in V : s_v^* \geq \alpha\}$ los terminales obtenidos van a ser siempre los mismos, esto no altera el factor 2.54, pero resulta imposible realizar un análisis del impacto del valor de alfa en la solución final.

Los resultados de la ejecución de 297 instancias se encuentra en la Sección A.3 del apéndice.

5.5. Variantes del Algoritmo Aleatorio

Considerando lo expresado en la sección anterior, se busca una variante del algoritmo donde se pueda aplicar una aleatorización que tenga efecto en el resultado para poder estudiar la incidencia del parámetro. Dado que en el resultado obtenido del modelo lineal relajado se observa que las variables asociadas a las aristas en general toman valores continuos entre 0 y 1, se decide implementar una versión del algoritmo que tome en consideración las mismas. Al igual que al algoritmo presentado en la Capítulo 4, el problema consiste en seleccionar los nodos terminales de acuerdo a cierto criterio para luego aplicar el algoritmo $Goemans - Williamson$. Se plantean algunas alternativas para el sorteo del parámetro aleatorio y se estudia la viabilidad de las mismas mediante resultados empíricos.

Los criterios planteados para seleccionar los nodos terminales son los siguientes. Por un lado se tiene que sortear un parámetro aleatorio, para dicho sorteo se plantean tres variantes:

- Sortear el parámetro α aleatorio de la misma forma que en el algoritmo del Capítulo 4.
- Sortear el parámetro α aleatorio en el intervalo $(minAristas, 1)$.
- Sortear el parámetro α aleatorio en el intervalo $(promAristas, 1)$.

Luego, si la variable asociada a una arista es mayor que el parámetro aleatorio, se agregan los nodos correspondientes a los extremos de la arista siempre y cuando los mismos tengan penalización mayor que cero. Este criterio se parte de considerar que si una arista debe estar en la solución, entonces los nodos de los extremos también, pero por otra parte no tiene sentido seleccionar como nodo terminal un nodo que no tenga penalización asociada. Podemos expresar los nodos seleccionados como terminales de la siguiente forma: $\{v_i \in V : \pi_{v_i} > 0, \exists v_j \in V : x_{ij} \geq \alpha \text{ o } x_{ji} \geq \alpha\}$. Esta variante del algoritmo no tiene un factor de aproximación demostrado, por lo cual se estudiará empíricamente su comportamiento. Del resultado obtenido en el estudio de esta variante se puede concluir que no es apropiado realizar un sorteo del parámetro α con ninguna de las tres opciones planteadas. Como alternativa se plantea considerar una versión determinista donde se consideren 2 valores para α, el primero $\alpha = \frac{(promAristas+minAristas)}{2}$ el cual es el valor medio entre el valor mínimo de las variables asociadas a las aristas y el valor promedio. Esto debido a que en la mayoría de los casos estudiados es posible encontrar el mejor valor objetivo tomando dicho α. El segundo es el valor 1, esto debido a que en algunos casos particulares se observó que el mejor valor objetivo se obtiene con $\alpha = 1$.

5.6. Análisis en Instancias Pequeñas

Se cuenta con algunas instancias de tamaño reducido, donde por reducido se hace referencia a instancias de 20 o menos nodos, para estas instancias se estudia el comportamiento de las implementaciones desarrolladas en cuanto a valor objetivo y tiempo. Los resultados son analizados y comparados en relación a una implementación que brinda el resultado óptimo la cual es desarrollada en paralelo a nuestro trabajo en otro proyecto de grado, bajo la autoría de Martín Berguer [Berguer, 2017].

En la Figura 5.1 se muestra la cantidad de instancias para cada uno de los puntos porcentuales del gap. Como se puede observar, el pico más alto se da cuando el gap es cero, el cual abarca cerca del 90 % de las instancias ejecutadas y en ninguno de los casos supera el 20 %.

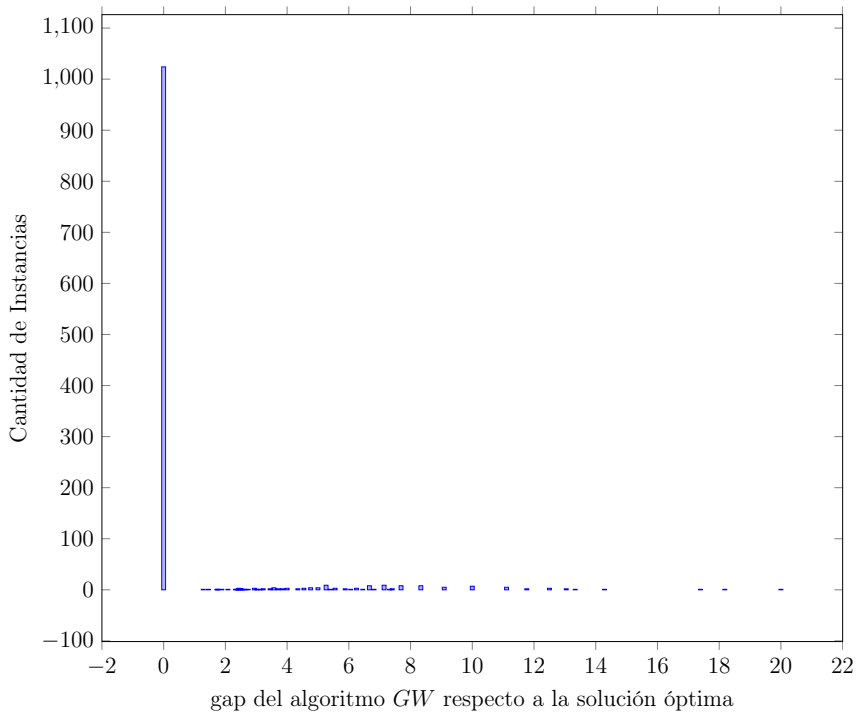


Figura 5.1: Indica la cantidad de instancias para cada valor porcentual del gap.

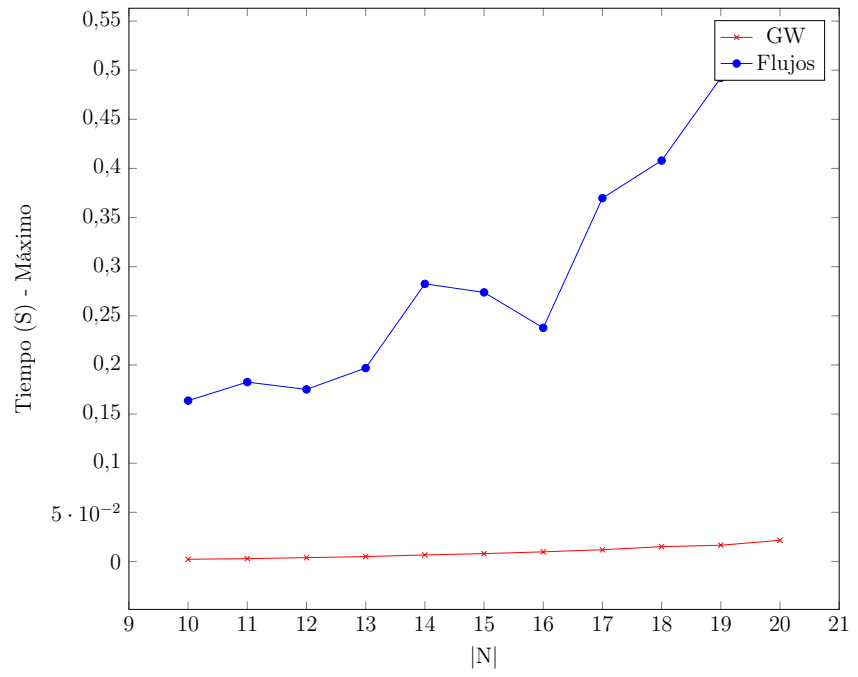


Figura 5.2: Comparativa de tiempos entre *GW* y un algoritmo exacto para grafos de 10 a 20 nodos.

En la Figura 5.2 se presenta una comparativa de tiempo en grafos con cantidad de nodos que van desde 10 hasta 20. Se observa que el crecimiento es más acelerado en la linea azul que corresponde al algoritmo exacto.

Debido a que el algoritmo exacto que se utiliza para la comparación está enfocado en la versión *unrooted* del PCST, para que fuera posible comparar gap y tiempo frente al algoritmo exacto se debió ejecutar el algoritmo GW tantas veces como nodos tenga la instancia. La solución final sale de tomar la que tiene el menor valor objetivo de todas las ejecuciones. El tiempo de ejecución utilizado en la comparación es el correspondiente a la suma de los tiempos de todas las ejecuciones.

5.7. Análisis en Instancias DIMACS

En esta sección se estudia el rendimiento de los algoritmos presentados en el Capítulo 3 y el Capítulo 4. Para ello se dispone además de los resultados obtenidos de las ejecuciones de los programas propios, los resultados óptimos obtenidos con el programa *staynerd* [6]. A continuación se presentan múltiples comparativas entre las ejecuciones de los algoritmos, midiendo tiempos de acuerdo a distintos criterios tales como el ńumero de aristas, la densidad de un grafo, entre otros.

La información completa de todas las ejecuciones que se toman como base para los análisis de este capítulo se encuentra en la Sección A.3 del apéndice, allí se puede ver la cantidad de nodos, aristas, los tiempos de ejecución, los valores objetivo obtenidos y el gap. Se puede observar en la Sección A.3 que no se dispone de todos los resultados óptimos, esto debido a que el programa *staynerd* no culminó para ciertas instancias. El no disponer de los resultados óptimos fue el principal problema encontrado a lo largo del trabajo, ya que limitaba la comparación de resultados.

La Tabla 5.2 muestra un resumen del tiempo de ejecución, mínimo, promedio y máximo por cada grupo de instancias para los algoritmos $Goemans-Williamson$ y (RN). En la Tabla 5.3 presenta el gap mínimo y máximo por cada grupo de instancia en comparación con el valor objetivo obtenido al aplicar el programa *staynerd*.

[6] http://homepage.univie.ac.at/ivana.ljubic/research/staynerd/StayNerd.html

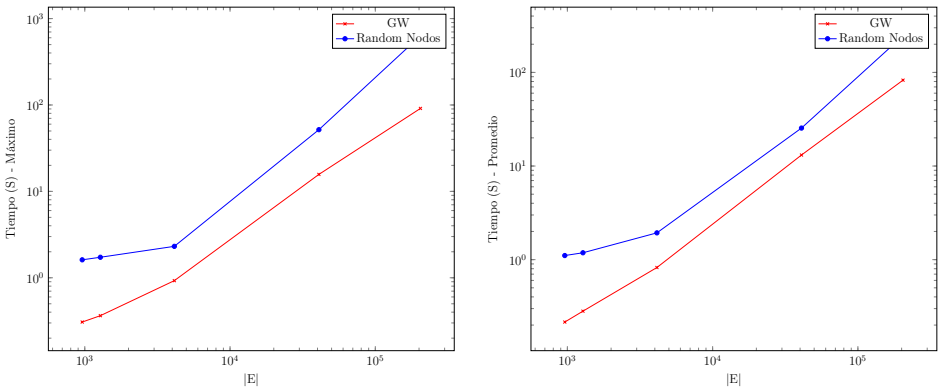
Observando los datos obtenidos se puede suponer que en cuanto a tiempos y gap siempre es más favorable el algoritmo $Goemans - Williamson$. Para validar esta hipótesis, a continuación se presentan tablas y gráficas con múltiples comparativas.

Tabla 5.2: Comparación de tiempos, mínimo, promedio y máximo, entre los algoritmos por grupos de instancias.

Grupo	min(GW)	promedio(GW)	max(GW)	min(RN)	promedio(RN)	max(RN)
cologne1	0.68	0.83	0.98	2.13	2.35	2.68
cologne2	7.27	8.04	9.92	15.07	16.61	19.53
CRR	0.09	1.8	10.44	0.57	4.06	19.49
i640	0.15	17.55	91.23	0.97	26.37	171.01
JMP	0.01	0.05	0.18	0.04	0.2	0.56
Random_graphs	0.06	9.12	60.31	0.2	49.12	429.62

Tabla 5.3: Comparación de gap, mínimo y máximo, entre los algoritmos por grupos de instancias.

Grupo	min(GW)	max(GW)	min(RN)	max(RN)
cologne1	0.00 %	11.98 %	0.00 %	75.31 %
cologne2	0.00 %	40.50 %	4.25 %	57.80 %
CRR	0.00 %	23.08 %	0.75 %	113.56 %
i640	1.25 %	43.57 %	10.41 %	46.28 %
JMP	0.00 %	8.16 %	5.05 %	117.85 %
Random_graphs	0.98 %	4.03 %	0.84 %	11.52 %



(a) Aristas vs Tiempo máximo (b) Aristas vs Tiempo promedio

Figura 5.3: Instancias i640. Variación de tiempo máximo y promedio al aumentar las aristas a nodos constantes.

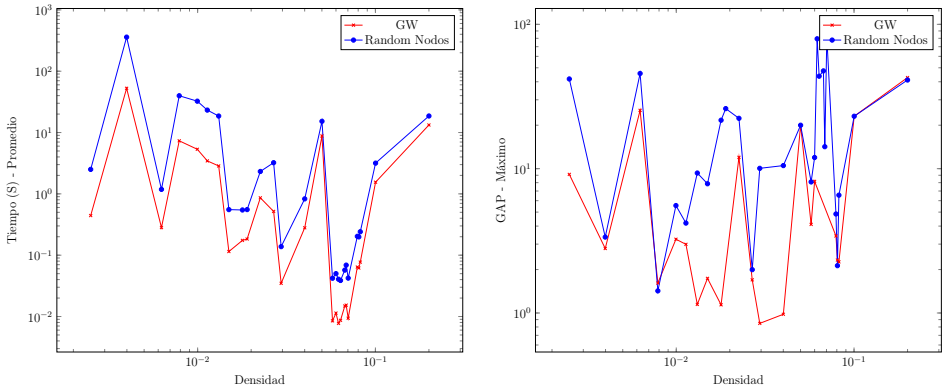
En la Figura 5.3 se presentan dos gráficas, una con tiempos máximos 5.3a y otra con tiempos promedio 5.3b. Este estudio tiene como objetivo ver como evoluciona el tiempo de ejecución de los algoritmos en instancias con una cantidad de nodos fija y aumentando la cantidad de aristas, para ello se seleccionaron las instancias i640 que tienen 640 nodos y con 960, 1280, 4135, 40896 y 204480 aristas. En la gráfica están representadas en los cinco puntos del eje x. Para cada uno de los números de aristas hay múltiples instancias, cuya diferencia está en la penalización que posee cada nodo y cantidad de nodos con penalización. En la gráfica se toma los tiempos máximos de los algoritmos para cada subconjunto de instancias con un mismo número de aristas y se observa que cuando aumenta la cantidad de aristas los tiempos aumentan y en todos los casos es mejor el tiempo de $Goemans - Williamson$.

En la Tabla 5.4 se puede ver de forma exacta cual es la diferencia en segundos entre los puntos del eje y, la última columna corresponde al factor entre los tiempos. El caso de mayor cantidad de aristas es el peor con un factor de casi siete veces mayor de t(RN) con respecto a t(GW).

Tabla 5.4: Instancias i640. Diferencias entre tiempos máximos

|E|	t(GW)	t(RN)	t(RN) - t(GW)	factor
960	0.30726	1.61367	1.30641	5.25181
1280	0.36421	1.7276	1.36339	4.74342
4135	0.92627	2.31079	1.38452	2.49473
40896	15.7205	51.7592	36.0387	3.29247
204480	91.2286	632.363	541.1344	6.93163

La gráfica 5.4a tiene como objetivo comparar el tiempo de ejecución de los algoritmos en relación con la densidad de los grafos de entrada. Se observa que independientemente de la densidad del grafo los tiempos del algoritmo $Goemans - Williamson$ son mejores a los de RN. La gráfica 5.4b compara el gap en relación con la densidad, se observa que para algunos casos es levemente mejor la solución provista por el Random por nodos, lo que nos conduce a profundizar en el análisis de cuales instancias el gap favorece a RN. En más de 250 instancias que se pudo comparar con *staynerd*, en aproximadamente 30 el gap del algoritmo RN es mejor que el de GW. Para estudiar en más detalles se seleccionan dos subconjuntos de 10



(a) Densidad vs Tiempo promedio (b) Densidad vs gap máximo

Figura 5.4: Comparativa de densidad de grafos versus tiempos promedios y gap máximos.

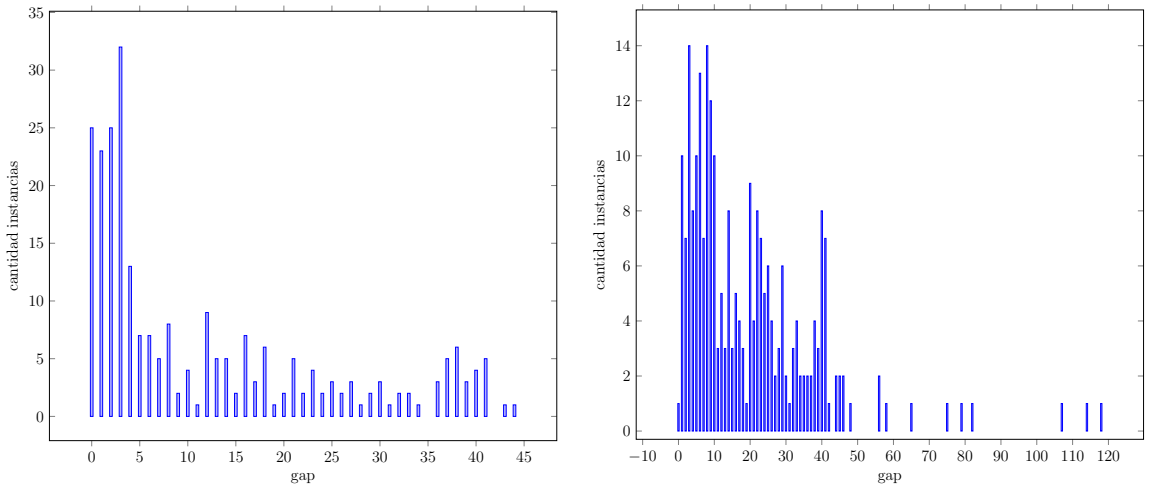
más significativos en los que es mejor RN (ver Tabla 5.5) y en los que es mejor GW (ver Tabla 5.6). En las tablas se puede comprobar que para los casos que RN tiene mejor gap con mayor diferencia, esta no es significativa, considerando que en tiempo es más lento. Por otro lado comparando los casos que la diferencia de gap favorece a GW, se puede observar que las mismas son bastante pronunciadas.

Tabla 5.5: 10 instancias para las cuales el gap de GW es mayor que RN y son las diferencias más altas

Instancia	nodos	aristas	staynerd	gap(GW)	gap(RN)	diferencia gap
i640-044	640	40896	1665	35.98	27.93	8.05
i640-245	640	40896	9448	42.77	37.9	4.86
i640-244	640	40896	8950	37.92	33.77	4.15
i640-143	640	40896	4321	37.91	35.11	2.80
i640-043	640	40896	1401	24.91	22.13	2.78
i640-225	640	204480	8386	43.57	41.22	2.34
i640-243	640	40896	9315	37.87	35.65	2.22
i640-241	640	40896	9716	39.47	37.58	1.89
C18-A	500	12500	111	15.32	13.51	1.80
i640-141	640	40896	5171	36.38	34.73	1.64

Tabla 5.6: 10 instancias para las cuales el gap de RN es mayor que GW y son las diferencias más altas

Instancia	nodos	aristas	staynerd	gap(GW)	gap(RN)	diferencia gap
K100.10	100	319	151731	0	117.85	117.85
C07-A	500	1000	59	0	113.56	113.55
D07-A	1000	2000	59	0	106.78	106.77
K100.1	100	348	149330	0	81.55	81.55
K100.6	100	307	157041	0	79.43	79.42
i105M1	741	6296	26717.2	0	75.31	75.31
C11-A	500	2500	26	3.85	65.38	61.53
i204M2	1801	16719	161700.55	0	57.8	57.79
D11-A	1000	5000	25	0	56	56
i102M2	749	6343	352538.82	3.05	55.53	52.47



(a) Gap algoritmo GW vs cantidad de instancias por puntos porcentuales

(b) Gap algoritmo RN vs cantidad de instancias por puntos porcentuales

Figura 5.5: Comparativa de gap.

Para finalizar el análisis de gap se presentan las gráficas de la figura Figura 5.5, donde en la gráfica 5.5a se observa que los mayores picos del gap de GW están entre el 0 % y el 3 %, que representa aproximadamente unas 90 instancias. Por encima del 40 % de gap son menos de 15 instancias, nunca superando el 50 %, sin embargo, si se observa la gráfica 5.5b que corresponde al algoritmo RN, los picos son menos pronunciados, pero la cantidad de instancias con gap menores al 3 % no superan las 30, siendo notoriamente inferior a GW. Si bien la mayoría está por debajo del

50 %, aproximadamente 10 instancias superan con creces ese límite, llegando hasta un 120 %.

A continuación se presenta un resumen comparativo entre los algoritmos *GW* y *RN*. Tomando el tiempo y gap como dos de las medidas principales se pueden sacar algunas conclusiones del desempeño de ambos algoritmos. Con respecto al tiempo de ejecución, el algoritmo $Goemans - Williamson$ es mejor. Una posible explicación es que el algoritmo de *RN*, primero debe resolver el problema relajado, luego seleccionar los nodos terminales y por último aplicar *GW* para resolver el problema de Steiner típico. Con respecto al gap como se observó en los datos existe un pequeño porcentaje donde el gap de *RN* es mejor que del algoritmo *GW*, pero esas mejoras no son significativas.

Considerando las comparativas entre los dos algoritmos implementados y observando que *GW* es el que mejor resultados brinda de los dos, a continuación se presenta un análisis entre los tiempos de ejecución de *GW* y *staynerd*. Aquí se tienen en cuenta instancias más grandes que las analizadas en la Sección 5.6. En resumen el punto de quiebre está en los grafos de 5000 nodos, a partir de aquí tal como se comenta al comienzo de la sección no fue posible obtener un resultado de las ejecuciones de *staynerd*, con *GW* si fue posible. Por debajo de esa cantidad de nodos depende de la topología del grafo, por ejemplo, como se puede comprobar en la Tabla 5.7 para las instancias del grupo i640 con 204480 aristas la comparativa es favorable a *GW*; sin embargo, en grafos del grupo Random_graphs con 4000 nodos pero menos densos, tal como lo indica la Tabla 5.8, el tiempo es favorable a *staynerd* y una posible interpretación al observar la topología es que estas instancias tienen penalización mayor a cero en todos sus nodos.

Tabla 5.7: Instancias i640 con 204480 aristas y el tiempo de *GW* es mejor a *staynerd*

Instancia	|V|	|E|	$\pi_v > 0$	t(GW)	t(staynerd)
i640-021	640	204480	9	83.0139	227.791
i640-022	640	204480	9	89.0416	234.906
i640-023	640	204480	9	88.8683	262.447
i640-225	640	204480	50	84.2184	1466.029
i640-223	640	204480	50	78.5136	1602.102
i640-224	640	204480	50	82.0833	2053.932
i640-221	640	204480	50	82.8221	2519.373

Tabla 5.8: Instancias Random_graphs, donde $|V| = |\pi_v > 0|$ y el tiempo de *staynerd* es mejor a *GW*

Instancia	$\|V\|$	$\|E\|$	$\pi_v > 0$	t(GW)	t(staynerd)
a4000RandGraph.2	4000	31880	4000	52.6689	7.502
a4000RandGraph.3	4000	32025	4000	48.1988	7.236
a4000RandGraph.1.2	4000	32087	4000	60.3061	21.127
a3000RandGraph.1.5	3000	23852	3000	23.6221	5.159
a3000RandGraph.2	3000	24065	3000	24.6345	6.957
a3000RandGraph.3	3000	24026	3000	22.2465	4.952
a3000RandGraph.1.2	3000	24045	3000	28.3301	18.232

5.8. Análisis de las Variantes del Algoritmo Aleatorio

En la presente sección se estudian las variantes planteadas al algoritmo aleatorio, en dichas variantes se pretende utilizar el resultado de las variables asociadas a las aristas para determinar cuales nodos seleccionar como terminales. La intención es determinar si el algoritmo brinda resultados aceptables y si las variantes planteadas para el sorteo del parámetro aleatorio α son apropiadas. En caso de ser necesario, se plantea encontrar una alternativa al problema de la selección de nodos terminales.

A continuación se presentan diversos análisis que tratan de brindar las respuestas a las dudas planteadas.

En la Tabla 5.9 y en la gráfica de la Figura 5.6 se muestra el comportamiento del algoritmo en un grafo de densidad 1 que contiene 100 nodos. Para ver el comportamiento en este tipo de grafos se realiza una discretización del parámetro α en el intervalo $(0, 1)$ tomando tramos de 0,01. El grafo de estudio es generado de forma

aleatoria, donde se asignan los costos de las aristas y las penalizaciones de los nodos con distribución uniforme en $(0, 1)$ y luego se alteran las penalizaciones de los nodos de manera de tomar distintas variantes mediante la alteración de la medida *cna* de la Definición 5.2.5. Para un primer análisis se consideran grafos donde el parámetro *cna* varíe entre 1.0 y 5.0. Otra medida a tener en cuenta es la cantidad de nodos que tienen penalización mayor que 0 con respecto del total, esta medida se encuentra expresada en la Definición 5.2.6. En este primer caso de estudio todos tienen penalización mayor que 0. En la gráfica de la Figura 5.6 se observa que ninguno de los criterios planteados para realizar un sorteo del parámetro α son aceptables. Se puede observar que cualquier valor del parámetro α seleccionado que esté cercano a 1 hace que el valor objetivo crezca. Esto es concordante con la topología del grafo, dado que al ser un grafo de densidad 1 con penalizaciones promedio de los nodos mayor o igual que el costo promedio de las aristas, es natural pensar que se puede encontrar un árbol de cubrimiento mínimo donde el costo de conectar a un nodo con otro sea, en general más pequeño que la penalización del nodo.

Tabla 5.9: Análisis *cna* 1 a 5.

cna	obj(GW)	promAristas	minAristas	Mejor Obj.	Intervalo Mejora α
1	1.2249	0.72058	0.0100	1.2249	(0.01, 0.60)
2	1.2264	0.72058	0.0100	1.2264	(0.01, 0.60)
3	1.2279	0.72058	0.0100	1.2279	(0.01, 0.60)
4	1.2295	0.72058	0.0100	1.2295	(0.00, 0.60)
5	1.2310	0.72058	0.0100	1.2310	(0.00, 0.60)

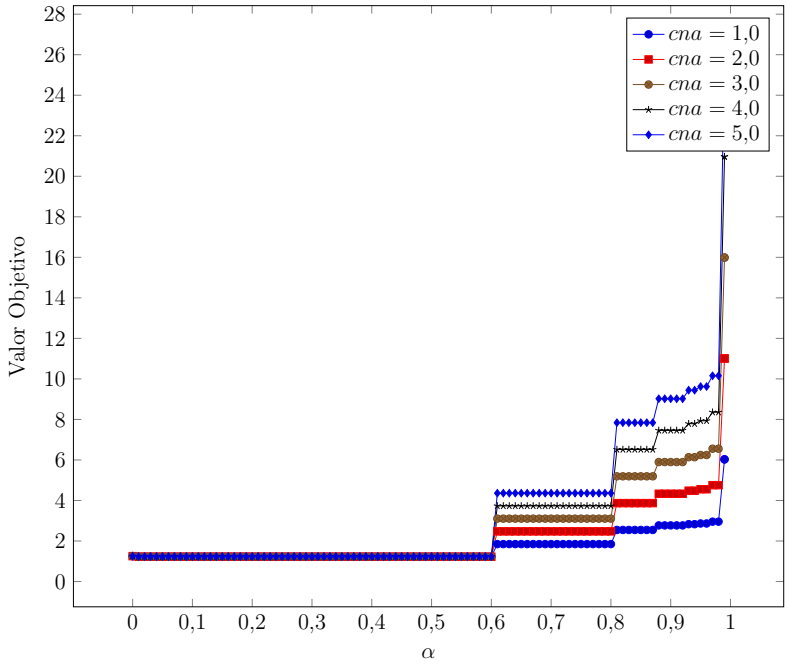


Figura 5.6: Gráfica Análisis *cna* 1 a 5.

El siguiente análisis es idéntico al anterior, pero en este caso se varían los valores tomados por el parámetro *cna*. Se toman valores de dicho parámetro entre 0.01 y 0.05. Esto se corresponde con grafos donde el costo promedio de aristas es superior a la penalización promedio de los nodos. En este caso se puede observar en la Tabla 5.10 y en la gráfica Figura 5.7 que en general, al igual que el caso anterior tomar valores del parámetro α cercanos a 1 aumenta el valor objetivo. Lo que varía con respecto al caso de estudio anterior es que si se toman valores de α cercanos a 0 también aumenta el valor objetivo. Como excepción se observa que cuando el parámetro *cna* toma el valor 0.01 o 0.02 el mejor valor objetivo se encuentra cuando α toma el valor 1.0.

Tabla 5.10: Análisis *cna* 0.01 a 0.05.

cna	obj(GW)	promAristas	minAristas	Mejor Obj.	Intervalo Mejora α
0.01	0.5188	0.5676	0.0100	0.5637	1.00
0.02	0.7775	0.6664	0.0100	0.8050	1.00
0.03	0.9048	0.6207	0.0100	0.9381	(0.04, 0.53)
0.04	0.9709	0.6478	0.0100	1.0007	(0.04, 0.57)
0.05	1.0164	0.6790	0.0100	1.0455	(0.01, 0.57)

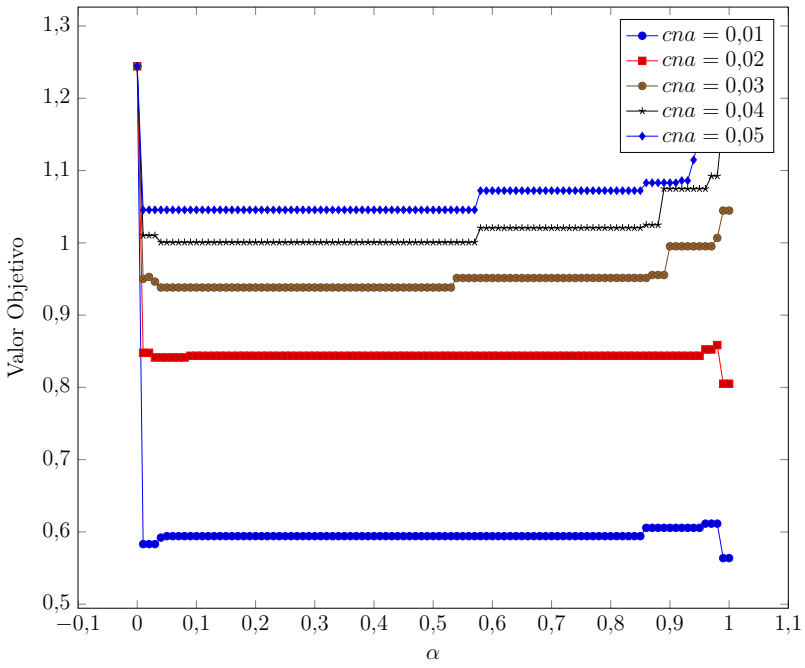


Figura 5.7: Análisis *cna* 0.01 a 0.05.

En los casos de estudio anteriores se consideró grafos aleatorios donde todos sus nodos tienen penalización mayor que 0. Como se observa, a pesar de los distintos valores tomados por el parámetro *cna*, en la mayoría de los casos no es beneficioso que el parámetro α tome valores cercanos a uno. A continuación se presenta un estudio considerando grafos con distintos valores de la medida *pnp*. En este caso también se consideran grafos con densidad 1. Se consideran grafos aleatorios con el parámetro *cna* fijo en 2. En la Tabla 5.11 y la gráfica Figura 5.8 se puede observar que al igual que en los casos anteriores, cuando el parámetro α toma valores cercanos a uno, el valor objetivo se incrementa. A diferencia de los casos anteriores, el mejor resultado aparece tomando valores α cercanos a 0.

Tabla 5.11: Análisis *pnp* 0.1 a 0.9.

pnp	obj(GW)	promAristas	minAristas	Mejor Obj.	Intervalo Mejora α
0.1	0.2555	0.3551	0.0100	0.2555	(0, 0.97)
0.3	0.5925	0.4689	0.0100	0.5925	(0, 0.77)
0.5	0.7845	0.5267	0.0100	0.7845	(0, 0.90)
0.7	0.9376	0.6306	0.0100	0.9376	(0, 0.72)
0.9	0.9726	0.6639	0.0100	0.9726	(0, 0.80)

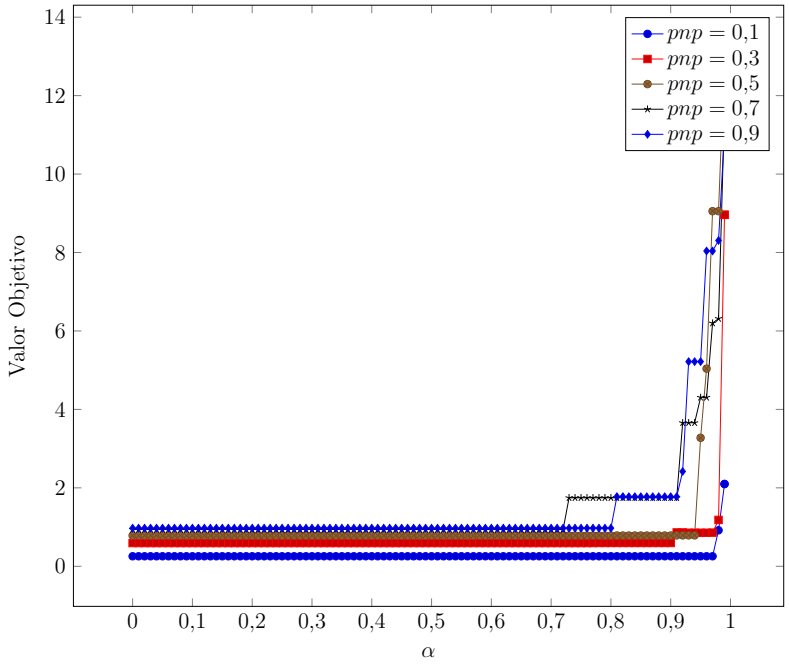
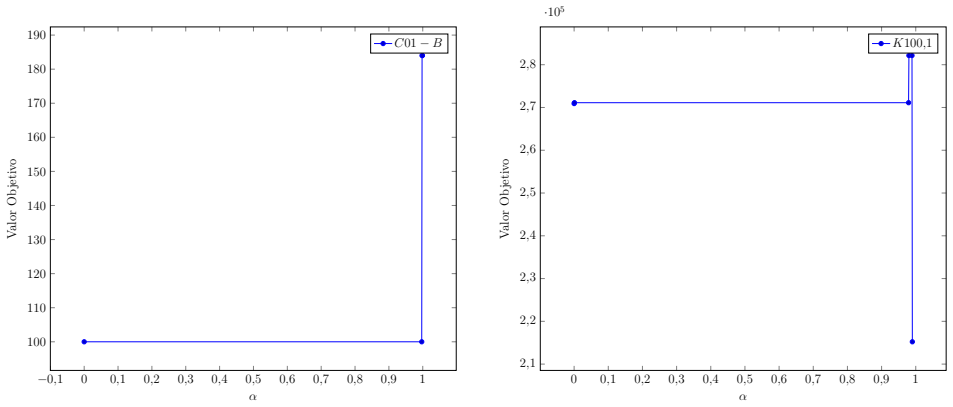


Figura 5.8: Gráfica Análisis *pnp* 0.1 a 0.9.

A continuación se realiza un estudio similar pero tomando dos combinaciones distintas, por un lado grafos con bajas densidades y baja proporción de nodos con penalizaciones mayores que 0 y por otro grafos con baja densidad y alta proporción de nodos con penalización mayor que 0. Para el primer caso se eligen dos grafos de las instancias DIMACS con muy poca cantidad de nodos con penalización mayor que 0 (menor al 5 %). Dichos grafos son las instancias C01-B con densidad 0.005 y K100.1 con densidad 0.07. Para estos casos, dado que la discretización se realiza en 1000 intervalos, en la gráfica solo se muestran los puntos de cambio. Es posible observar en las gráficas de la Figura 5.9 se obtienen resultados distintos, en el caso de la instancia C01-B si α toma valores cercanos a 1 el valor objetivo aumenta, mientras que para la instancia K100.1 el mejor valor objetivo se obtiene cuando α toma el valor 1.

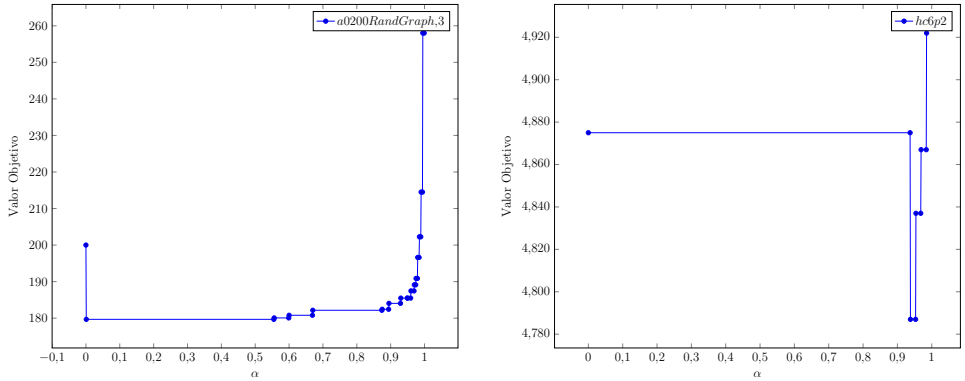
A continuación se presenta el mismo estudio pero para instancias de grafos poco densos donde todos los nodos tienen penalizaciones mayores que 0. Las instancias elegidas para dicho estudio son las instancias DIMACS hc6p2 con densidad 0.09, a0200RandGraph.3 con densidad 0.081, hc10u con densidad 0.009 y a0800RandGraph.3 con densidad 0.02. Para el caso de la instancia a0200RandGraph.3



(a) C01-B (b) K100.1

Figura 5.9: Discretización del parámetro α para las instancias C01-B y K100.1

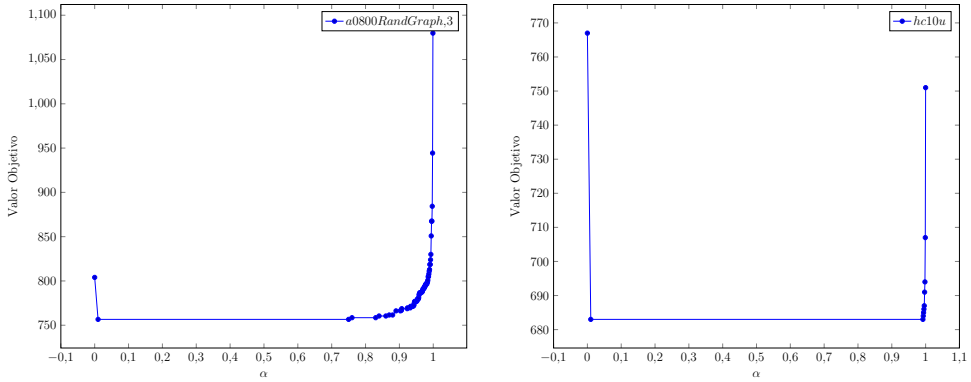
se observa en la gráfica de la Figura 5.10 que si α toma valores cercanos a 1 el valor objetivo aumenta, mientras que para el caso de la instancia hc6p2 existe un pequeño intervalo cercano a 1 donde si el parámetro α toma valores en él, se encuentra el mejor valor objetivo, pero si se acerca demasiado a 1 el valor objetivo aumenta. En el caso de las instancias a0800RandGraph.3 y hc10u, el análisis es muy similar, se observa en las gráficas de la Figura 5.11 que no es conveniente que α tome valores cercanos a 1.



(a) a0200RandGraph.3 (b) hc6p2

Figura 5.10: Discretización del parámetro α para las instancias a0200RandGraph.3 y hc6p2

En la Tabla 5.12 se puede observar que este criterio determinista planteado como alternativa para la selección del parámetro α es adecuado para los casos estudiados.



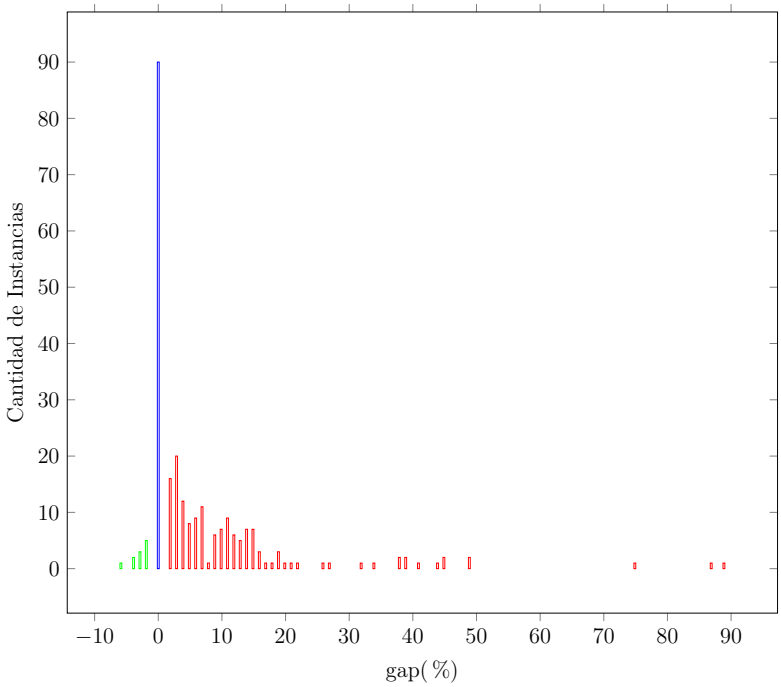
(a) a0800RandGraph.3 (b) hc10u

Figura 5.11: Discretización del parámetro α para las instancias a0800RandGraph.3 y hc10u

Tabla 5.12: Resumen Análisis Discretización del parámetro α

Instancia	Mejor Obj.	Obj. en $\alpha = \frac{(minAristas+promAristas)}{2}$	Obj. en $\alpha = 1$	Obj. GW	$d(G)$	cna	pnp
Aleatorio $cna = 1$	1.2249	1.2249	6.02869	1.2249	1	1	1
Aleatorio $cna = 2$	1.2264	1.2264	11.0085	1.2264	1	2	1
Aleatorio $cna = 3$	1.2279	1.2279	15.9883	1.2279	1	3	1
Aleatorio $cna = 4$	1.2295	1.2295	20.9681	1.2295	1	4	1
Aleatorio $cna = 5$	1.2310	1.2310	25.9479	1.2310	1	5	1
Aleatorio $cna = 0{,}01$	0.5637	0.5941	0.5637	0.5188	1	0.01	1
Aleatorio $cna = 0{,}02$	0.8050	0.8436	0.8050	0.7775	1	0.02	1
Aleatorio $cna = 0{,}03$	0.9381	0.9381	1.0445	0.9048	1	0.03	1
Aleatorio $cna = 0{,}04$	1.0007	1.0007	1.1756	0.9709	1	0.04	1
Aleatorio $cna = 0{,}05$	1.0455	1.0455	1.2568	1.0164	1	0.05	1
Aleatorio $pnp = 0{,}01$	0.2555	0.2555	2.0977	0.2555	1	2	0.1
Aleatorio $pnp = 0{,}03$	0.5925	0.5925	8.9613	0.5925	1	2	0.3
Aleatorio $pnp = 0{,}05$	0.7845	0.7845	13.0277	0.7845	1	2	0.5
Aleatorio $pnp = 0{,}07$	0.9376	0.9376	11.637	0.9376	1	2	0.7
Aleatorio $pnp = 0{,}09$	0.9726	0.9726	11.0669	0.9726	1	1	0.9
C01-B	67	100	67	100	0.005	9.79	0.01
k100.1	215216	271116	215216	149330	0.07	0.59	0.11
a0200RandGraph-3	179.68	179.68	258.006	173.19	0.081	0.66	1
hc6p2	4787	4875	4922	4572	0.009	0.72	1
a0800RandGraph-3	756.61	756.61	1079.8	674.28	0.002	0.63	1
hc10u	683	683	751	683	0.009	0.73	1

Para analizar el comportamiento del algoritmo más ampliamente, se realizan ejecuciones del mismo para las instancias que también fueron procesadas con el algoritmo *GW* y se realiza una comparativa con el mismo. Como se observa en la gráfica de la Figura 5.12 existen algunos casos donde se obtiene mejor resultado que en el algoritmo *GW* y algunos en los cuales se obtiene un gap con respecto al mismo superior al 20 %. En la gran mayoría de los casos el resultado tiene un gap respecto al algoritmo *GW* de entre 0 % y 20 %.



(a) Gap *RA* vs *GW*

Figura 5.12: Comparativa de gap Vs. *GW*

5.9. Aplicación del Algoritmo *GW*

En la presente sección se presentan los resultados de la aplicación del algoritmo *GW* a algunos ejemplos de grafos reales. Los nodos y aristas de color rojo de las figuras son los que conforman el grafo resultante. El primer grafo estudiado es el correspondiente a la red RAU2 (Red Académica Uruguaya versión 2), el mismo tiene un resultado óptimo de 51, el cual es el mismo que el obtenido por el algoritmo *GW*. En la Figura 5.13 se puede ver el resultado de la ejecución del algoritmo *GW* sobre la instancia RAU2.

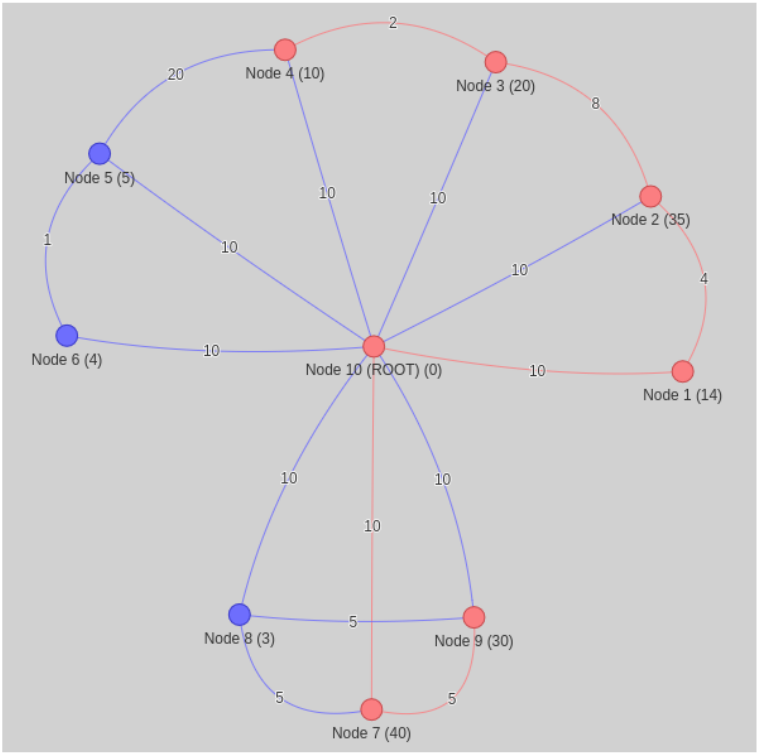


Figura 5.13: Red RAU2

Otro grafo estudiado es el correspondiente a la instancia de la red ARPANET (Primera red de internet creada en 1969), la misma tiene un resultado óptimo de 38 igual que el algoritmo *GW*. En la Figura 5.14 se muestra un ejemplo de ejecución sobre la red arpanet.

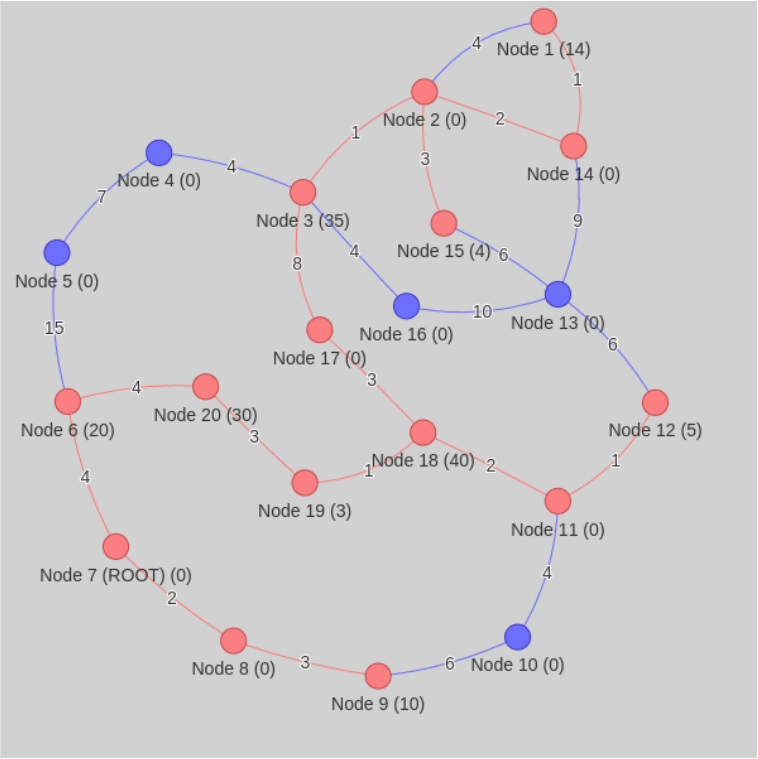


Figura 5.14: Arpanet

En ambos casos de estudio se asignaron valores aleatorios a los costos de las artistas y las penalizaciones por nodos ya que se dispone de las estructuras de las redes pero no de estos valores.

Para finalizar la sección de resultados experimentales se muestra un breve ejemplo de aplicación a una red de UTE. Se dispone de un diseño topológico de la red de UTE prevista para el año 2023 extraída de la propia web de UTE[7], la misma se puede observar en la Figura 5.15, a partir de ella se seleccionan los nodos correspondientes a parques eólicos y se agregan 5 aristas de forma aleatoria para aumentar la densidad del grafo y luego se aplica el algoritmo *GW* tomando como nodo raíz el punto situado en Montevideo. En la Figura 5.16 se puede observar el grafo de entrada y en la Figura 5.17 se puede ver el grafo en color rojo el correspondiente árbol solución.

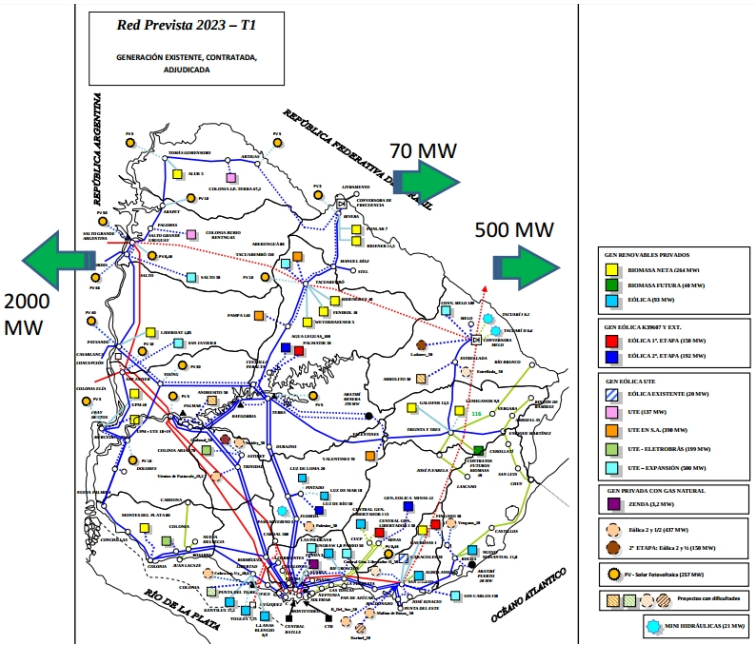


Figura 5.15: Topología de UTE esperada para el año 2023

El resultado de la aplicación del algoritmo *GW* sobre el grafo de los parques eólicos brinda un valor objetivo de 1469 en 0.000714 segundos, mientras que el valor óptimo en este caso es de 1454 lo que produce un gap de un 1 %.

[7]http://portal.ute.com.uy/sites/default/files/documents/files/mapa%202023.pdf

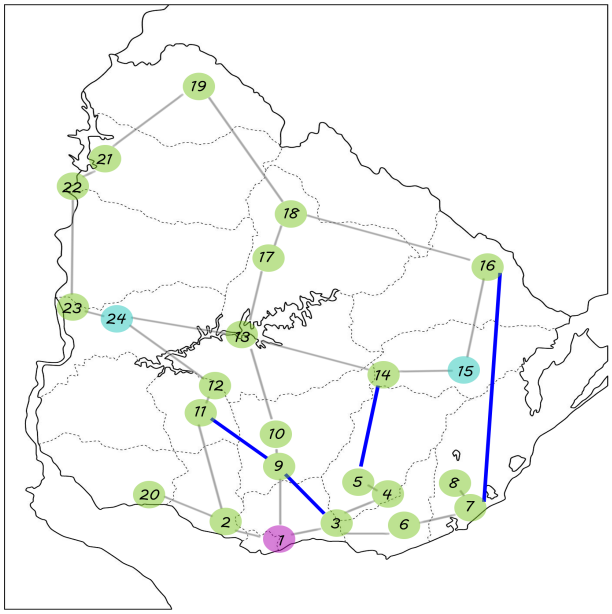


Figura 5.16: Grafo de los parques eólicos de UTE (extraído de [Berguer, 2017])

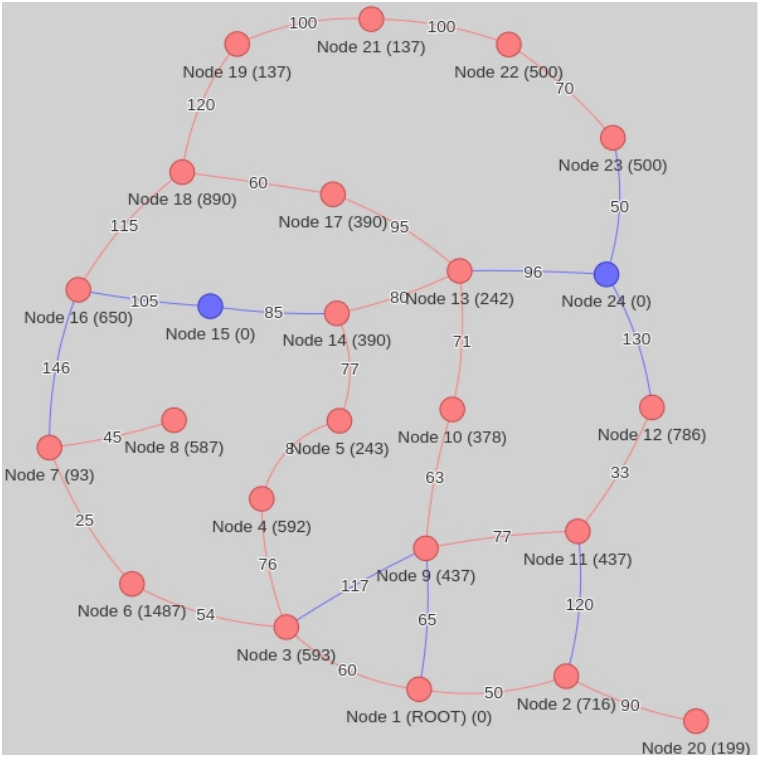


Figura 5.17: Grafo representativo de los parques eólicos de UTE con 5 aristas aleatorias

Capítulo 6

Conclusiones y Trabajo a Futuro

6.1. Conclusiones Generales

El problema estudiado presenta aplicación en nivel de logística y comunicaciones de diversa índole, por lo cual es de gran interés la búsqueda de soluciones al mismo. El problema categoriza dentro de la clase NP-Completo, por lo cual no es viable encontrar soluciones óptimas de manera eficiente para instancias de tamaño arbitrario y es prohibitivo para instancias de gran tamaño. Como alternativa para la resolución del problema se pueden considerar los algoritmos basados en metaheurísticas y los algoritmos de aproximación.

De acuerdo con lo planteado al comienzo, el principal objetivo de este proyecto es estudiar la aplicación de algoritmos de aproximación al PCST. Se han estudiado algoritmos de aproximación tanto determinísticos como aleatorios y se optó por el desarrollo del algoritmo $Goemans - Williamson$ de factor 2 por su simplicidad y su aparente eficiencia, lo que ha sido validado en este proyecto.

De acuerdo con los análisis de los experimentos realizados, las conclusiones que se obtienen se presentan a continuación. Como primer punto, al algoritmo $Goemans - Williamson$ tiene un comportamiento muy bueno en cuanto al compromiso entre tiempos de ejecución y valores objetivo, los cuales en muchos casos se acercan o son iguales a los óptimos globales. En específico, con respecto a los tiempos de ejecución,

en las pruebas realizadas se verifica que el tiempo de ejecución en general es menor con respecto a otras implementaciones de algoritmos exactos estudiados y en la mayoría de los casos los valores de gap tienden a cero, aunque se vieron algunas topologías de grafos particulares donde los tiempos no fueron favorables. Si bien el factor de aproximación teórico es 2, los resultados empíricos muestran que en general no se supera el factor 1.5. En cuanto a las alternativas planteadas, se puede concluir que son pocos los casos en los que se obtiene una mejora con respecto al algoritmo $Goemans-Williamson$ y además en general tienen un tiempo de ejecución superior debido a la necesidad de resolver el modelo ILP relajado.

Por otra parte, en cuanto al algoritmo RA, no se tiene un factor de aproximación formalmente demostrado, aunque los resultados empíricos muestran que en general se obtienen valores similares al algoritmo $Goemans - Williamson$.

De acuerdo a los resultados obtenidos de las tres implementaciones, se concluye que los algoritmos de aproximación son una excelente alternativa cuando se está dispuesto a ceder en un margen acotado el valor objetivo en pos de mejorar de forma contundente los tiempos de ejecución.

6.2. Trabajo a Futuro

Si bien los algoritmos RA y RN muestran resultados que en general no son mejores que $Goemans - Williamson$, existe una posibilidad de realizar un preprocesado del grafo que podría brindar mejoras tanto en tiempos de ejecución como en valor objetivo. En lineas generales el preprocesamiento consiste en una serie de pasos previos a la ejecución del algoritmo donde se quitan aristas y nodos redundantes. Esta técnica fue planteada por [Duin y Volgenant, 1987] y utilizada en un algoritmo con solución exacta para el PCST propuesto por [Ljubić et al., 2006].

Por otra parte, de futuro también se aspira a analizar algoritmos de mejor factor y analizar la aplicabilidad del PCST en problemas reales de interés para entidades estatales como UTE en el proyecto “Planificación estocástica óptima para la generación y acumulación diaria de energía, integrada a políticas de control en *Smart*

Grids".

Un desafío interesante es analizar de forma minuciosa posibles mejoras en los algoritmos, tanto en estructuras de datos, como en estrategias de ejecución en función del tipo de grafo (densidad, cantidad de aristas y nodos). Otra posible mejora es analizar las posibilidades de procesamiento en paralelo para así poder usar los múltiples núcleos de las cpu's actuales. En particular, por ejemplo, en el algoritmo $Goemans - Williamson$ es visible que se puede paralelizar los cálculos de los ϵ mínimos.

Bibliografía

Archer, A., M. Bateni, M. Hajiaghayi, y H. Karloff

2011. Improved Approximation Algorithms for Prize-Collecting Steiner Tree and TSP. *SIAM Journal on Computing*, 40(2):309–332.

Bar-Yehuda, R. y S. Even

1981. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198 – 203.

Berguer, M.

2017. Análisis, modelización, y resolución exacta vía programación lineal entera del prize-collecting steiner tree problem. Universidad de la República - Facultad de Ingeniería. Proyecto de Grado. Tutores: Dr. Ing. Franco Robledo y Dr. Ing. Pablo Romero.

Biazzo, I., A. Braunstein, y R. Zecchina

2013. On the performance of a cavity method based algorithm for the prize-collecting steiner tree problem on graphs. *CoRR*, abs/1309.0346.

Bienstock, D., M. X. Goemans, D. Simchi-Levi, y D. Williamson

1993. A note on the prize collecting traveling salesman problem. *Mathematical Programming*, 59(1):413–420.

Brazil, M., R. L. Graham, D. A. Thomas, y M. Zachariasen

2014. On the history of the euclidean steiner tree problem. *Archive for History of Exact Sciences*, 68(3):327–354.

Byrka, J., F. Grandoni, T. Rothvoß, y L. Sanità

2010. An improved lp-based approximation for steiner tree. In *Proceedings of the*

Forty-second ACM Symposium on Theory of Computing, STOC '10, Pp. 583–592, New York, NY, USA. ACM.

Canuto, S. A., M. G. C. Resende, y C. C. Ribeiro

2001. Local search with perturbations for the prize-collecting steiner tree problem in graphs. *Networks*, 38(1):50–58.

Christofides, N.

1976. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University.

Cole, R., R. Hariharan, M. Lewenstein, y E. Porat

2001. A faster implementation of the goemans-williamson clustering algorithm. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '01, Pp. 17–25, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Coleman, T. F. y J. J. Moré

1983. Estimation of sparse jacobian matrices and graph coloring problems. *SIAM Journal on Numerical Analysis*, 20(1):187–209.

Cook, S. A.

1971. The complexity of theorem-proving procedures. In *In STOC*, Pp. 151–158. ACM.

Dantzig, G. B.

1963. *Linear Programming and Extensions.*

Dantzig, G. B., L. R. Ford, y D. R. Fulkerson

1956. A primal-dual algorithm for linear programs. *Linear Inequalities and Related Systems. H. W. Kuhn and A. W. Tucker*, Pp. 171–181.

Duin, C. W. y A. Volgenant

1987. Some generalizations of the steiner problem in graphs. *Networks*, 17(3):353–364.

Feofiloff, P., C. G. Fernandes, C. E. Ferreira, y J. C. de Pina

2007. Primal-dual approximation algorithms for the prize-collecting steiner tree problem. *Information Processing Letters*, 103(5):195 – 202.

Fischetti, M., M. Leitner, I. Ljubić, M. Luipersbeck, M. Monaci, M. Resch, D. Salvagnin, y M. Sinnl

2016. Thinning out steiner trees: a node-based model for uniform edge costs. *Mathematical Programming Computation*, Pp. 1–27.

Garey, M. R. y D. S. Johnson

1977. The rectilinear steiner tree problem is np-complete. *SIAM Journal on Applied Mathematics*, 32(4):826–834.

Garey, M. R. y D. S. Johnson

1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co.

Gilbert, E. N. y H. O. Pollak

1968. Steiner minimal trees. *SIAM Journal on Applied Mathematics*, 16(1):1–29.

Goemans, M. X. y D. P. Williamson

1995. A General Approximation Technique for Constrained Forest Problems. *SIAM Journal on Computing*, 24(2):296–317.

Goemans, M. X. y D. P. Williamson

1997. Survey. The Primal-Dual Method for Approximation Algorithms and its Application to Network Design Problems. *Approximation Algorithms for NP-hard Problems, D. Hochbaum*, Pp. 144–191.

Gordeev, E. N. y O. Tarastsov

1993. The steiner problem: a survey. *Discrete Mathematics and Applications*, 3(4):339–364.

Johnson, D. S., M. Minkoff, y S. Phillips

2000. The prize collecting steiner tree problem: Theory and practice. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*,

SODA '00, Pp. 760–769, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

Karp, R. M.

1972. *Reducibility among Combinatorial Problems*, Pp. 85–103. Boston, MA: Springer US.

Karp, R. M.

2008. George dantzig's impact on the theory of computation. *Discrete Optimization*, 5(2):174 – 185. In Memory of George B. Dantzig.

Ljubić, I., R. Weiskircher, U. Pferschy, G. W. Klau, P. Mutzel, y M. Fischetti

2006. An algorithmic framework for the exact solution of the prize-collecting steiner tree problem. *Mathematical Programming*, 105(2/3):427 – 449.

Motwani, R. y P. Raghavan

1995. *Randomized Algorithms*. New York, NY, USA: Cambridge University Press.

Osman, I. H. y J. P. Kelly

1996. *Meta-Heuristics: An Overview*. Boston, MA: Springer US.

Raghavan, P. y C. D. Tompson

1987. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4):365–374.

Rosseti, I., M. P. de Aragão, C. C. Ribeiro, E. Uchoa, y R. F. Werneck

2004. *New Benchmark Instances for The Steiner Problem in Graphs*, Pp. 601–614. Boston, MA: Springer US.

Segev, A.

1987. The node-weighted steiner tree problem. *Netw.*, 17(1):1–17.

Siarry, P.

2016. *Metaheuristics*. Cham: Springer International Publishing.

Szász, D.

2011. John von Neumann, the Mathematician. *Mathematical Intelligencer*, 33(2):42–51.

Vazirani, V. V.

2001. *Approximation Algorithms.* New York, NY, USA: Springer-Verlag New York, Inc.

Williamson, D. P., M. X. Goemans, M. Mihail, y V. V. Vazirani

1995. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica*, 15(3):435–454.

Williamson, D. P. y D. B. Shmoys

2011. *The Design of Approximation Algorithms*, 1st edition. New York, NY, USA: Cambridge University Press.

Winter, P.

1987. Steiner problem in networks: A survey. *Netw.*, 17(2):129–167.

Apéndice A

Apéndice

A.1. Problema del Agente Viajero con Penalidades

El problema del agente viajero con penalidades (PCTSP por sus siglas en inglés) se define de la siguiente manera. Dado un grafo $G = (V, E)$ completo, no dirigido, con un conjunto de vértices V y uno de aristas E. $\forall\ e = \{i, j\}, e \in E$ su costo es c_e, $\forall\ i \in V$ su penalización es π_i. En el trabajo de [Bienstock et al., 1993] se considera una versión reducida del problema. Se asume que el costo de las aristas cumplen con la desigualdad triangular, $c_{i,j} \leq c_{i,k} + c_{k,j},\ \forall\ i,\ j,\ k \in V$. El objetivo es encontrar un tour que visita un subconjunto de vértices talque la suma del largo del tour más la suma de las penalizaciones de los vértices no incluidos en el tour sea la menor posible.

A.2. Implementación

La implementación del sistema esta realizada en C++ y el código esta estructurado en distintas clases, las 2 fundamentales son las siguientes:

- Graph: Esta clase contiene la representación de los grafos, sus principales atributos son la lista de nodos, el nodo root, la lista de subconjuntos activos y no activos (Utilizados por el algoritmo GW). Su principal funcionalidad es la

resolución del algoritmo *GW*, dada una instancia de un grafo, esta función retorna otro instancia de grafo que es el resultado de la ejecución del algoritmo *GW* según se especifica en el capitulo 3. Cada nodo tiene una lista de aristas, donde cada arista guarda su peso y una referencia al nodo vecino. La implementación sirve tanto para grafos dirigidos como no dirigidos, en el caso de un grafo no dirigido se agregan las aristas en ambos sentidos con igual peso.

- ResolveCplex: Esta clase tiene como principal funcionalidad generar y resolver el modelo ILP 4.1 relajado para una instancia de un grafo, hallar una solución óptima a dicho problema, determinar cuales nodos serán terminales y cuales de Steiner y devolver una copia del grafo pero donde los nodos de Steiner tienen penalización 0 y los terminales infinito. Dicha copia es a la que se le aplica el algoritmo *GW* para completar el algoritmo planteado en el capítulo 4. La resolución del modelo lineal se realiza mediante la integración del programa con librerías de la herramienta *CPLEX*[1], dicha herramienta cuenta con un entorno de desarrollo propio pero también con librerías para distintos lenguajes de programación, entre ellos C++[2] . Estas librerías exponen una API para poder modelar y resolver problemas de programación lineal entera o mixta.

El programa posee la capacidad de ejecutarse tanto en modo manual como en modo batch, es decir, se puede iniciar el programa y desde dentro asignar los parámetros para luego ejecutar o se pueden pasar los parámetros como argumentos al invocar el programa desde la consola. A continuación se presenta la lista de parámetros que se pueden pasar al programa.

- **-h** Muestra la ayuda del programa.
- **-v** Imprime información adicional de la ejecución del programa.
- **-p** <**ruta**> Ruta donde se encuentra el archivo .stp que se desea cargar.

[1] `https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/`

[2] `https://www.ibm.com/support/knowledgecenter/en/SSSA5P_12.7.0/ilog.odms.ide.help/refcppopl/html/overview.html`

- **-f <nombre archivo>** Nombre del archivo .stp que contiene la instancia del grafo a cargar.
- **-rcsvf <nombre archivo>** Nombre del archivo donde se guardan los resultados de las ejecuciones.
- **-r <Id nodo root>** Es el identificador del nodo root.
- **-re** Indica si el modelo ILP generado por la clase ResolveCplex se relajan las variables asociadas a las aristas y los nodos. En el caso de desear ejecutar los algoritmos RN y RA, este argumento es necesario.
- **-gw** Indica que se ejecute el algoritmo GW.
- **-rm** Indica que se ejecute la versión aleatoria del algoritmo RN que realiza la selección por el valor de las variables asociadas a los nodos.
- **-ar** Indica que se ejecute la versión aleatoria del algoritmo RA que realiza la selección por el valor de las variables asociadas a las aristas.
- **-e** Indica que una vez culminada la ejecución de los algoritmos indicados el programa finalice.
- **-jsi** Indica que se guarde la instancia del grafo original en un archivo .json que puede ser utilizado posteriormente para la representación gráfica del grafo.
- **-jso** Indica que se guarde la instancia del grafo resultante de la aplicación del algoritmo en un archivo .json que puede ser utilizado posteriormente para la representación gráfica del grafo.
- **-ilpm** Indica que se guarde el modelo ILP en un archivo.
- **-batch** Indica que el programa se esta corriendo en modo batch.
- **-da** Indica que se realice un discretización del parámetro α y se ejecute el algoritmo AR para cada intervalo discreto.
- **-gr** Indica si se genera un grafo aleatorio. En caso de no utilizar un grafo proveniente de u archivo .stp se puede construir un grafo aleatorio.

- **-grn <cantidad nodos>** Indica la cantidad de nodos del grafo aleatorio. Se genera un grafo aleatorio con la cantidad de nodos que se le indique. El gafo es completo y se le asigna a cada nodo una penalización aleatoria en el intervalo (0, 1). Tambien se le asigna a cada arista un costo aleatoria en el intervalo (0, 1)

- **-grca <multiplicador>** Indica un multiplicador para el valor de los costos de las aristas del grafo aleatorio generado. Multiplica el costo de las aristas por este coeficiente.

- **-grcn <multiplicador>** Indica un multiplicador para el valor de las penalizaciones de los nodos del grafo aleatorio generado. Multiplica la penalización de los nodos por este coeficiente.

- **-daci <comienzo intervalo>** Indica en que punto del intervalo $(0, 1)$ comienza la discretización.

- **-cint <cantidad de intervalos>** Cantidad de valores que tomará el parámetro α en la discretización. Estos valores seran tomados entre el valor pasado en el argumento -daci y 1.

- **-th <cantidad de hilos>** Es la cantidad de hilos de ejecución que puede utilizar el programa.

- **-fsa <forma de selecció de nodos>** Indica la forma en la que se selecciona el parámetro α en el algoritmo AR. Las opciones son:

  - 0 - Se realiza un sorteo al igual que en el algoritmo RN (entre $e^{-1/2}$ y 1).
  - 1 - Se realiza un sorteo entre $promAristas$ y 1.
  - 2 - Se realiza un sorteo entre $minAristas$ y 1.
  - 3 - Se le asigna el valor $promAristas$.
  - 4 - Se le asigna el valor $minAristas$.
  - 5 - Se le asigna el valor $\frac{(promAristas+minAristas)}{2}$.
  - 6 - Se realizan dos ejecuciones, una con $\alpha = 1$ y con $\alpha = \frac{(promAristas+minAristas)}{2}$ El programa devuelve la mejor solución.

A continuación se presentan algunos ejemplos de ejecución del programa. Suponiendo que se tiene una estructura del tipo */opt/programas/pcst* y */opt/instanciasPCST/instancia1.stp* se puede ejecutar el programa de la siguiente forma:

```
$ ./pcst -gw -p "/opt/instanciasPCST/" -f "instancia1.stp"
    -rcsvf "resultadosGW.csv" -batch -e

$ ./pcst -rm -re -p "/opt/instanciasPCST/" -f "instancia1.
    stp" -rcsvf "resultadosRN.csv" -batch -e

$ ./pcst -ar -re -fsa 6 -p "/opt/instanciasPCST/" -f "
    instancia1.stp" -rcsvf "resultadosRA.csv" -batch -e

$ ./pcst -da -daci 0 -cint 100 -re -p "/opt/instanciasPCST
    /" -f "instancia1.stp" -rcsvf "resultadosDA.csv" -batch
    -e
```

El primer comando ejecuta solo el algoritmo *GW*, el segundo solo el *RN*, el tercero ejecuta el algoritmo *RA* y por último en cuarto comando ejecuta la discretización en el intervalo (0, 1) para 10 valores distintos de α. Si se desea ver los resultados por consola no son necesarios los argumentos -batch -e.

A.3. Resultados de Ejecuciones

Tabla A.1: Grupo ACTMODPC

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
drosophila001	5226	93394		323.756	8295.6	-			-	
drosophila005	5226	93394		320.562	8191.67	-			-	
drosophila0075	5226	93394		317.867	8114.56	-			-	
lymphoma	2034	7756		5.76643	3382.05	-	13.4613	3539.59	-	3505.05

Tabla A.2: Grupo Cologne

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
i101M1	748	6332	109271.5	0.69344	109272	0.00 %	2.34542	158419	44.98 %	158419
i101M2	748	6332	315925.31	0.89785	341795	8.19 %	2.39908	385704	22.09 %	385704

Tabla A.2: Grupo Cologne

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
i101M3	748	6332	355625.41	0.95479	385704	8.46 %	2.38392	385704	8.46 %	385704
i102M1	749	6343	104065.8	0.67572	104066	0.00 %	2.12677	114120	9.66 %	104066
i102M2	749	6343	352538.82	0.89188	363303	3.05 %	2.44384	548307	55.53 %	500542
i102M3	749	6343	454365.93	0.95665	506798	11.54 %	2.40114	548307	20.68 %	548307
i103M1	751	6343	139749.41	0.71834	139749	-0.00 %	2.12885	158393	13.34 %	158393
i103M2	751	6343	407834.23	0.89247	414414	1.61 %	2.4224	498888	22.33 %	498888
i103M3	751	6343	456125.49	0.9756	510786	11.98 %	2.40443	510786	11.98 %	510786
i104M2	741	6293	89920.84	0.77707	89920.8	-0.00 %	2.28136	97148.8	8.04 %	96174.4
i104M3	741	6293	97148.79	0.78083	97148.8	0.00 %	2.6764	97148.8	0.00 %	97148.8
i105M1	741	6296	26717.2	0.68202	26717.2	-0.00 %	2.25895	46839.1	75.31 %	26717.2
i105M2	741	6296	100269.62	0.84539	100270	0.00 %	2.24359	114983	14.67 %	114983
i105M3	741	6296	110351.16	0.88154	114983	4.20 %	2.36261	114983	4.20 %	114983
i201M2	1803	16743	355467.68	9.91641	355468	0.00 %	15.5357	405787	14.16 %	405787
i201M3	1803	16743	628833.61	8.36317	634951	0.97 %	18.6275	878130	39.64 %	878130
i201M4	1803	16743	773398.3	7.85803	819725	5.99 %	16.1058	894136	15.61 %	894136
i202M2	1804	16740	288946.83	7.38355	288947	0.00 %	15.9342	324025	12.14 %	324025
i202M3	1804	16740	419184.16	7.65672	430188	2.63 %	16.7298	490190	16.94 %	490190
i202M4	1804	16740	430034.26	8.79217	490190	13.99 %	16.6582	490190	13.99 %	490190
i203M2	1809	16762	459894.78	7.26514	459895	0.00 %	16.0533	507465	10.34 %	507465
i203M3	1809	16762	643062.02	7.68262	666415	3.63 %	15.906	671860	4.48 %	671860
i203M4	1809	16762	677733.07	7.27245	706531	4.25 %	17.5584	706531	4.25 %	706531
i204M2	1801	16719	161700.55	8.74706	161701	0.00 %	15.3124	255162	57.80 %	227025
i204M3	1801	16719	245287.2	8.06747	344623	40.50 %	19.5309	344623	40.50 %	344623
i204M4	1801	16719	245287.2	7.62357	344623	40.50 %	15.0736	344623	40.50 %	344623
i205M2	1810	16794	571031.42	7.84276	571459	0.07 %	16.0762	683942	19.77 %	677428
i205M3	1810	16794	672403.14	7.92067	778670	15.80 %	17.3077	778670	15.80 %	778670
i205M4	1810	16794	713973.62	8.25416	844756	18.32 %	16.7252	854959	19.75 %	854959

Tabla A.3: Grupo CRR

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
C01-B	500	625	91	0.08671	100	9.89 %	0.56564	100	9.89 %	100
C02-B	500	625	163	0.15554	173	6.13 %	0.63272	176	7.98 %	176
C03-A	500	625	434	0.24133	439	1.15 %	0.61999	630	45.16 %	610
C03-B	500	625	748	0.12161	774	3.48 %	0.64928	790	5.61 %	790
C04-A	500	625	624	0.10497	633	1.44 %	0.59811	829	32.85 %	769
C04-B	500	625	1063	0.10633	1100	3.48 %	0.64448	1104	3.86 %	1107
C05-A	500	625	1081	0.09682	1084	0.28 %	0.67534	1236	14.34 %	1235
C05-B	500	625	1528	0.11426	1550	1.44 %	0.62093	1567	2.55 %	1567
C06-B	500	1000	57	0.11058	62	8.77 %	0.63698	62	8.77 %	62
C07-A	500	1000	59	0.11737	59	0.00 %	0.72635	126	113.56 %	110
C07-B	500	1000	117	0.18082	131	11.97 %	0.7083	131	11.97 %	131
C08-A	500	1000	371	0.16314	384	3.50 %	0.80728	436	17.52 %	437
C08-B	500	1000	505	0.16269	528	4.55 %	0.69713	532	5.35 %	532
C09-A	500	1000	535	0.13015	550	2.80 %	0.70768	614	14.77 %	611
C09-B	500	1000	694	0.13108	727	4.76 %	0.68183	730	5.19 %	731
C10-A	500	1000	861	0.12923	883	2.56 %	0.72335	924	7.32 %	931
C10-B	500	1000	1069	0.15662	1108	3.65 %	0.89276	1108	3.65 %	1108
C11-A	500	2500	26	0.39206	27	3.85 %	1.20639	43	65.38 %	40
C11-B	500	2500	38	0.35147	43	13.16 %	1.04364	43	13.16 %	43
C12-A	500	2500	40	0.28779	45	12.50 %	0.8919	50	25.00 %	53
C12-B	500	2500	48	0.28706	53	10.42 %	0.91076	53	10.42 %	53
C13-A	500	2500	236	0.28135	249	5.51 %	0.95253	261	10.59 %	261
C13-B	500	2500	258	0.29112	276	6.98 %	0.95368	276	6.98 %	276
C14-A	500	2500	295	0.31708	313	6.10 %	0.96886	324	9.83 %	323
C14-B	500	2500	320	0.303	338	5.62 %	0.96318	338	5.62 %	338
C15-A	500	2500	502	0.27618	518	3.19 %	1.34286	528	5.18 %	528
C15-B	500	2500	552	0.26577	567	2.72 %	1.642	567	2.72 %	567
C16-A	500	12500	13	1.61157	16	23.08 %	2.75677	16	23.08 %	16
C16-B	500	12500	13	1.63226	16	23.08 %	2.78527	16	23.08 %	16

Tabla A.3: Grupo CRR

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
C17-A	500	12500	21	1.78143	22	4.76 %	2.86448	22	4.76 %	22
C17-B	500	12500	21	1.70102	22	4.76 %	2.87977	22	4.76 %	22
C18-A	500	12500	111	1.42546	128	15.32 %	3.03662	126	13.51 %	127
C18-B	500	12500	113	1.62589	131	15.93 %	2.91872	131	15.93 %	131
C19-A	500	12500	148	1.61675	157	6.08 %	3.42074	161	8.78 %	161
C19-B	500	12500	148	1.6157	159	7.43 %	3.34366	159	7.43 %	159
C20-A	500	12500	266	1.23807	268	0.75 %	3.99426	275	3.38 %	268
C20-B	500	12500	267	1.24334	269	0.75 %	3.71291	269	0.75 %	269
D01-B	1000	1250	122	0.43329	132	8.20 %	2.42671	132	8.20 %	132
D02-B	1000	1250	252	0.67316	275	9.13 %	2.56727	275	9.13 %	275
D03-A	1000	1250	820	0.41318	822	0.24 %	2.57711	1163	41.83 %	1095
D03-B	1000	1250	1509	0.43741	1582	4.84 %	2.74494	1606	6.43 %	1606
D04-A	1000	1250	1210	0.39308	1219	0.74 %	2.4038	1466	21.16 %	1459
D04-B	1000	1250	1882	0.40565	1952	3.72 %	2.38886	1999	6.22 %	1999
D05-A	1000	1250	2157	0.39105	2208	2.36 %	2.43322	2527	17.15 %	2513
D05-B	1000	1250	3135	0.39602	3208	2.33 %	2.50569	3236	3.22 %	3239
D06-B	1000	2000	82	0.73304	90	9.76 %	2.72394	90	9.76 %	90
D07-A	1000	2000	59	0.55342	59	0.00 %	2.81649	122	106.78 %	111
D07-B	1000	2000	119	0.73211	122	2.52 %	2.79493	122	2.52 %	122
D08-A	1000	2000	758	0.54204	779	2.77 %	2.70162	934	23.22 %	920
D08-B	1000	2000	1036	0.5424	1119	8.01 %	2.67205	1123	8.40 %	1122
D09-A	1000	2000	1077	0.72279	1120	3.99 %	2.84997	1292	19.96 %	1290
D09-B	1000	2000	1425	0.85739	1529	7.30 %	3.41688	1533	7.58 %	1533
D10-A	1000	2000	1671	0.57672	1707	2.15 %	3.01931	1821	8.98 %	1818
D10-B	1000	2000	2079	0.59439	2134	2.65 %	3.48439	2146	3.22 %	2144
D11-A	1000	5000	25	1.13316	25	0.00 %	3.51141	39	56.00 %	37
D11-B	1000	5000	33	1.21213	39	18.18 %	4.60032	39	18.18 %	39
D12-A	1000	5000	50	1.69096	50	0.00 %	3.92658	51	2.00 %	51
D12-B	1000	5000	50	2.60629	51	2.00 %	3.97049	51	2.00 %	51
D13-A	1000	5000	450	1.61099	480	6.67 %	4.11732	489	8.67 %	491
D13-B	1000	5000	490	1.42604	525	7.14 %	4.04134	527	7.55 %	527
D14-A	1000	5000	602	1.48866	634	5.32 %	4.74742	649	7.81 %	648
D14-B	1000	5000	665	1.43409	704	5.86 %	4.78079	705	6.02 %	705
D15-A	1000	5000	1042	1.2903	1084	4.03 %	5.89973	1102	5.76 %	1091
D15-B	1000	5000	1108	1.28542	1155	4.24 %	7.31268	1155	4.24 %	1155
D16-A	1000	25000	15	10.2604	17	13.33 %	14.6417	18	20.00 %	18
D16-B	1000	25000	15	10.4366	18	20.00 %	12.8432	18	20.00 %	18
D17-A	1000	25000	24	10.1321	26	8.33 %	12.6671	26	8.33 %	26
D17-B	1000	25000	24	8.61366	26	8.33 %	16.5006	26	8.33 %	26
D18-A	1000	25000	219	8.6448	246	12.33 %	15.252	248	13.24 %	245
D18-B	1000	25000	224	8.66045	255	13.84 %	14.1215	255	13.84 %	255
D19-A	1000	25000	307	8.40304	346	12.70 %	15.0616	358	16.61 %	345
D19-B	1000	25000	311	8.56314	353	13.50 %	16.6424	356	14.47 %	356
D20-A	1000	25000	537	6.65868	544	1.30 %	19.4897	558	3.91 %	552
D20-B	1000	25000	538	7.46694	545	1.30 %	15.2717	548	1.86 %	546

Tabla A.4: Grupo H

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
hc10p	1024	5120		1.24746	74455	-	3.49798	81673	-	81673
hc10u	1024	5120		1.17435	683	-	3.72315	683	-	683
hc11p	2048	11264		6.33447	147309	-	15.7688	162958	-	161712
hc11u	2048	11264		5.62395	1374	-	15.6491	1372	-	1372
hc12p	4096	24576		44.5548	293599	-	93.2156	324964	-	323176
hc12u	4096	24576		34.2646	2744	-	90.3876	2742	-	2742
hc6p	64	192		0.00463	4524	-	0.03447	4875	-	4875
hc6u	64	192		0.00369	41	-	0.02007	41	-	41
hc7p	128	448		0.01527	9108	-	0.06702	10012	-	9752
hc7u	128	448		0.01916	84	-	0.0722	85	-	85
hc8p	256	1024		0.0714	18570	-	0.27019	20608	-	19805

Tabla A.4: Grupo H

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
hc8u	256	1024		0.06319	168	-	0.23191	169	-	168
hc9p	512	2304		0.27527	35926	-	0.99254	40362	-	40327
hc9u	512	2304		0.27394	347	-	0.93956	346	-	346

Tabla A.5: Grupo H2

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
hc10p2	1024	5120		1.28922	75141	-	3.74097	81373	-	81373
hc10u2	1024	5120		1.23631	472	-	3.6502	471	-	470
hc11p2	2048	11264		5.93034	150276	-	16.004	162846	-	162846
hc11u2	2048	11264		5.79716	937	-	15.5149	936	-	936
hc12p2	4096	24576		46.0254	300163	-	91.1125	325269	-	325269
hc12u2	4096	24576		38.7329	1886	-	97.0936	1880	-	1880
hc6p2	64	192		0.00416	4572	-	0.03605	4875	-	4875
hc6u2	64	192		0.00385	25	-	0.01919	25	-	25
hc7p2	128	448		0.01432	9157	-	0.06638	10113	-	10114
hc7u2	128	448		0.0141	58	-	0.07281	59	-	59
hc8p2	256	1024		0.06513	18852	-	0.25998	20603	-	20607
hc8u2	256	1024		0.06211	116	-	0.23332	115	-	115
hc9p2	512	2304		0.28138	37662	-	0.94876	40361	-	40361
hc9u2	512	2304		0.29289	230	-	0.89941	232	-	232

Tabla A.6: Grupo i640

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
i640-001	640	960	3053	0.23753	3312	8.48 %	1.09204	4466	46.28 %	4191
i640-002	640	960	2795	0.15011	2940	5.19 %	0.9717	4020	43.83 %	3873
i640-003	640	960	2844	0.23409	3134	10.20 %	1.09082	3140	10.41 %	3140
i640-004	640	960	3356	0.23204	3769	12.31 %	1.04007	4316	28.61 %	4316
i640-005	640	960	3508	0.22225	3552	1.25 %	1.27605	4676	33.30 %	4101
i640-011	640	4135	2242	0.83994	2839	26.63 %	1.85271	2892	28.99 %	2892
i640-012	640	4135	2215	0.84652	2784	25.69 %	1.88583	2784	25.69 %	2784
i640-013	640	4135	2002	0.86704	2543	27.02 %	1.90193	2543	27.02 %	2543
i640-014	640	4135	2171	0.80602	3052	40.58 %	1.83927	3066	41.23 %	3066
i640-015	640	4135	2295	0.86862	2954	28.71 %	1.92251	2954	28.71 %	2954
i640-021	640	204480	1585	83.0139	2067	30.41 %	103.95	2095	32.18 %	2095
i640-022	640	204480	1704	89.0416	2202	29.23 %	104.878	2202	29.23 %	2202
i640-023	640	204480	1754	88.8683	2310	31.70 %	103.073	2351	34.04 %	2351
i640-024	640	204480	1575	85.6393	2067	31.24 %	108.662	2067	31.24 %	2067
i640-025	640	204480	1550	91.2286	2055	32.58 %	113.291	2055	32.58 %	2055
i640-031	640	1280	2400	0.32583	2954	23.08 %	1.22357	3496	45.67 %	3496
i640-032	640	1280	2053	0.20664	2288	11.45 %	1.0403	2508	22.16 %	2508
i640-033	640	1280	2789	0.25674	3144	12.73 %	1.07293	3469	24.38 %	3067
i640-034	640	1280	2757	0.23928	3371	22.27 %	1.08618	3451	25.17 %	3451
i640-035	640	1280	2510	0.32086	2870	14.34 %	1.14237	3010	19.92 %	3010
i640-041	640	40896	1792	15.7205	2253	25.73 %	15.6148	2253	25.73 %	2253
i640-042	640	40896	1621	12.9428	2288	41.15 %	16.3265	2288	41.15 %	2288
i640-043	640	40896	1401	12.384	1750	24.91 %	15.3749	1711	22.13 %	1711
i640-044	640	40896	1665	14.2713	2264	35.98 %	16.155	2130	27.93 %	2130
i640-045	640	40896	1569	14.4211	1932	23.14 %	22.4902	1932	23.14 %	1932
i640-101	640	960	8135	0.19061	9594	17.93 %	1.0464	9578	17.74 %	9578
i640-102	640	960	7791	0.29094	9309	19.48 %	1.60809	9807	25.88 %	9807
i640-103	640	960	7854	0.16033	9119	16.11 %	0.97187	9567	21.81 %	9567
i640-104	640	960	6965	0.30726	8434	21.09 %	1.61367	8997	29.17 %	8997
i640-105	640	960	8911	0.23367	10299	15.58 %	1.02974	11764	32.02 %	11764
i640-111	640	4135	5512	0.79263	7156	29.83 %	1.78275	7156	29.83 %	7156
i640-112	640	4135	5991	0.83514	8233	37.42 %	1.8354	8453	41.09 %	8453
i640-113	640	4135	5886	0.92107	7890	34.05 %	1.82335	7983	35.63 %	7983

Tabla A.6: Grupo i640

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
i640-114	640	4135	5767	0.87715	7508	30.19 %	1.8068	7445	29.10 %	7445
i640-115	640	4135	6040	0.81209	7694	27.38 %	1.81903	7974	32.02 %	7974
i640-121	640	204480	4379	84.9462	6006	37.15 %	119.571	6087	39.00 %	6087
i640-122	640	204480	4707	91.1524	6499	38.07 %	113.876	6499	38.07 %	6499
i640-123	640	204480	4509	82.508	6185	37.17 %	121.711	6185	37.17 %	6185
i640-124	640	204480	4723	81.6584	6584	39.40 %	118.487	6603	39.81 %	6603
i640-125	640	204480	4556	83.8595	6328	38.89 %	114.316	6367	39.75 %	6367
i640-131	640	1280	6490	0.22602	7671	18.20 %	1.17133	8307	28.00 %	8307
i640-132	640	1280	7800	0.26554	8758	12.28 %	1.09825	8964	14.92 %	8964
i640-133	640	1280	6808	0.30901	7767	14.09 %	1.10946	8282	21.65 %	8282
i640-134	640	1280	6507	0.27753	7624	17.17 %	1.0933	8050	23.71 %	8050
i640-135	640	1280	6618	0.26467	7821	18.18 %	1.11196	8113	22.59 %	8113
i640-141	640	40896	5171	13.0638	7052	36.38 %	18.6044	6967	34.73 %	6967
i640-142	640	40896	4733	12.9362	6239	31.82 %	20.0491	6170	30.36 %	6170
i640-143	640	40896	4321	15.0893	5959	37.91 %	17.8255	5838	35.11 %	5838
i640-144	640	40896	4562	12.3757	6244	36.87 %	18.2534	6244	36.87 %	6244
i640-145	640	40896	4907	13.5883	6534	33.16 %	18.9469	6534	33.16 %	6534
i640-201	640	960	14372	0.17408	16051	11.68 %	1.00402	17753	23.52 %	17753
i640-202	640	960	15059	0.16153	17753	17.89 %	1.00757	18168	20.65 %	18168
i640-203	640	960	13848	0.23543	16246	17.32 %	1.02242	17013	22.86 %	17013
i640-204	640	960	13309	0.18582	14874	11.76 %	0.99188	15517	16.59 %	15517
i640-205	640	960	15308	0.22594	18552	21.19 %	1.04564	18530	21.05 %	18530
i640-211	640	4135	11109	0.8279	15124	36.14 %	1.85219	15407	38.69 %	15407
i640-212	640	4135	10351	0.84235	14249	37.66 %	1.80579	14611	41.16 %	14611
i640-213	640	4135	10388	0.7974	14611	40.65 %	1.86477	14611	40.65 %	14611
i640-214	640	4135	10675	0.84036	14572	36.51 %	1.82414	14786	38.51 %	14786
i640-215	640	4135	10740	0.79565	13700	27.56 %	1.8327	13751	28.04 %	13751
i640-221	640	204480	8400	82.8221	11731	39.65 %	149.609	11731	39.65 %	11731
i640-222	640	204480	8993	80.2888	12548	39.53 %	144.282	12607	40.19 %	12607
i640-223	640	204480	9210	78.5136	13023	41.40 %	154.391	12911	40.18 %	12911
i640-224	640	204480	8870	82.0833	12492	40.83 %	171.012	12492	40.83 %	12492
i640-225	640	204480	8386	84.2184	12040	43.57 %	147.366	11843	41.22 %	11843
i640-231	640	1280	14279	0.33111	17688	23.87 %	1.7276	17888	25.27 %	17888
i640-232	640	1280	13526	0.23163	15708	16.13 %	1.06821	16491	21.92 %	16491
i640-233	640	1280	12948	0.29082	15076	16.43 %	1.09716	15894	22.75 %	15894
i640-234	640	1280	13645	0.2318	16468	20.69 %	1.07444	16872	23.65 %	16872
i640-235	640	1280	12720	0.28492	15737	23.72 %	1.09589	15959	25.46 %	15959
i640-241	640	40896	9716	11.8069	13551	39.47 %	18.768	13367	37.58 %	13367
i640-242	640	40896	9250	12.3371	12789	38.26 %	18.6068	12789	38.26 %	12789
i640-243	640	40896	9315	13.3809	12843	37.87 %	19.5588	12636	35.65 %	12636
i640-244	640	40896	8950	12.6442	12344	37.92 %	24.8031	11972	33.77 %	11972
i640-245	640	40896	9448	12.2443	13489	42.77 %	17.7296	13029	37.90 %	13029
i640-301	640	960	42822	0.24166	49813	16.33 %	1.10693	50838	18.72 %	50838
i640-302	640	960	42606	0.23243	47916	12.46 %	1.06291	49484	16.14 %	49336
i640-303	640	960	41286	0.19456	49372	19.59 %	1.02157	49728	20.45 %	49635
i640-304	640	960	42079	0.19062	49024	16.50 %	1.07806	48964	16.36 %	49334
i640-305	640	960	42798	0.20809	49094	14.71 %	1.041	48914	14.29 %	49206
i640-311	640	4135		0.74188	43728	-	2.21034	44340	-	44340
i640-312	640	4135		0.74763	43318	-	2.19915	42964	-	42964
i640-313	640	4135		0.75357	43133	-	2.31079	43444	-	43444
i640-314	640	4135		0.74194	43254	-	2.25169	42922	-	42922
i640-315	640	4135		0.92627	43114	-	2.03095	42662	-	42662
i640-321	640	204480		77.27	41335	-	602.794	41424	-	41424
i640-322	640	204480		76.1164	40783	-	615.524	40489	-	40489
i640-323	640	204480		81.9634	40492	-	632.363	40462	-	40462
i640-324	640	204480		75.8426	41255	-	599.408	41137	-	41137
i640-325	640	204480		72.5724	40252	-	585.758	40035	-	40035
i640-331	640	1280	39315	0.29669	47773	21.51 %	1.24522	47351	20.44 %	47351
i640-332	640	1280	39030	0.27662	48720	24.83 %	1.19141	48827	25.10 %	48827
i640-333	640	1280	39775	0.33484	48308	21.45 %	1.25685	49502	24.46 %	49485
i640-334	640	1280	39338	0.36421	49314	25.36 %	1.54963	50005	27.12 %	50005

Tabla A.6: Grupo i640

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
i640-335	640	1280	39601	0.2979	48020	21.26 %	1.22509	49383	24.70 %	49382
i640-341	640	40896		12.3824	41984	-	44.7224	41361	-	41361
i640-342	640	40896		12.3153	41944	-	45.8118	41910	-	41910
i640-343	640	40896		14.251	42295	-	42.9621	42212	-	42212
i640-344	640	40896		12.4732	42186	-	44.5062	42160	-	42160
i640-345	640	40896		11.5798	42366	-	51.7592	42310	-	42310

Tabla A.7: Grupo JMP

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
K100.1	100	348	149330	0.00939	149330	0.00 %	0.04231	271116	81.55 %	215216
K100.10	100	319	151731	0.00904	151731	0.00 %	0.04357	330550	117.85 %	264130
K100.2	100	339	243247	0.01533	243247	0.00 %	0.06884	277785	14.20 %	277785
K100.4	100	364	117107	0.00975	117107	-			-	
K100.6	100	307	157041	0.00773	157041	0.00 %	0.04057	281774	79.43 %	162819
K100.7	100	315	189219	0.00867	189219	0.00 %	0.03872	271936	43.71 %	271936
K100.9	100	333	146452	0.01484	146452	0.00 %	0.05716	216121	47.57 %	201145
K200	200	691	355797	0.03705	360306	-			-	
K400	400	1515	398802	0.18355	398802	0.00 %	0.55425	502760	26.07 %	501426
K400.2	400	1527	484327	0.17178	502999	3.86 %	0.56042	582498	20.27 %	570924
K400.4	400	1426	428200	0.1743	433079	1.14 %	0.546	520940	21.66 %	505195
K400.6	400	1576	397427	0.17721	397427	-			-	
K400.7	400	1442	482817	0.15823	485736	-			-	
K400.8	400	1516	440574	0.16154	441332	-			-	
K400.9	400	1500	404924	0.18019	407441	-			-	
P100	100	317	803300	0.00972	823026	2.46 %	0.04689	847997	5.56 %	847997
P100.1	100	284	968336	0.01021	1008120	4.11 %	0.04259	1017260	5.05 %	1017260
P100.2	100	297	425654	0.01137	460408	8.16 %	0.05005	476513	11.95 %	460408
P100.3	100	316	693527	0.00804	712866	2.79 %	0.04394	732708	5.65 %	732708
P100.4	100	284	851702	0.0068	866155	1.70 %	0.04164	920647	8.09 %	923925
P200	200	587	1317874	0.03481	1329040	0.85 %	0.13788	1450350	10.05 %	1450350
P400.1	400	1212	2809617	0.11219	2892130	2.94 %	0.53381	3019740	7.48 %	3019740
P400.2	400	1196	2534964	0.11475	2579020	1.74 %	0.55371	2734210	7.86 %	2720260

Tabla A.8: Grupo Random Graphs

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
a0200RandGraph.1.2	200	1636	122.21	0.07713	124.98	2.26 %	0.24116	130.22	6.55 %	129.44
a0200RandGraph.1.5	200	1575	141.88	0.06409	146.73	3.42 %	0.20317	148.76	4.85 %	157.67
a0200RandGraph.2	200	1605	157.02	0.06178	160.67	2.33 %	0.19875	160.36	2.13 %	170.51
a0200RandGraph.3	200	1616	170.29	0.06034	173.19	1.70 %	0.23802	171.72	0.84 %	179.68
a0400RandGraph.1.2	400	3194	234.98	0.27952	237.28	0.98 %	0.82856	259.66	10.50 %	245.88
a0400RandGraph.1.5	400	3231	272.87	0.26213	283.86	4.03 %	0.6897	295.61	8.33 %	301.25
a0400RandGraph.2	400	3292	300.92	0.24645	309.12	2.73 %	1.38876	315.67	4.90 %	343.22
a0400RandGraph.3	400	3222	337.6	0.23431	343.97	1.89 %	1.34204	343.33	1.70 %	380.5
a0600RandGraph.1.2	600	4821	361.06	0.63104	365.17	1.14 %	3.97511	402.24	11.41 %	383.67
a0600RandGraph.1.5	600	4845	408.29	0.60279	419.75	2.81 %	1.68051	433.52	6.18 %	454.11
a0600RandGraph.2	600	4831	460.02	0.56412	474.59	3.17 %	3.26025	471.84	2.57 %	523.59
a0600RandGraph.3	600	4808	507.94	0.51754	516.56	1.70 %	3.22165	518.08	2.00 %	571.73
a0800RandGraph.1.2	800	6453	465.25	1.20426	470.26	1.08 %	2.89491	506.47	8.86 %	489.94
a0800RandGraph.1.5	800	6301	531.22	1.65809	547.93	3.15 %	6.38308	561.45	5.69 %	595.6
a0800RandGraph.2	800	6465	603.33	1.02045	620.37	2.82 %	5.82247	622.36	3.16 %	676.55
a0800RandGraph.3	800	6385	663.61	1.03915	674.28	1.61 %	5.30148	676.62	1.96 %	755.55
a10000RandGraph.1.2	10000	80298		493.264	6003.51	-			-	
a10000RandGraph.1.5	10000	80288		447.245	6974.57	-			-	
a10000RandGraph.2	10000	79908		407.989	7786.2	-			-	
a10000RandGraph.3	10000	79778		387.832	8580.23	-			-	
a1000RandGraph.1.2	1000	8067	580.35	1.95755	588.4	1.39 %	4.79289	633.37	9.14 %	617.86

Tabla A.8: Grupo Random Graphs

Instancia	n	m	staynerd	t(GW)	GW	gap	t(RN)	RN	gap	RA
a1000RandGraph.1.5	1000	7868	674.07	1.82771	694	2.96 %	10.1642	721.05	6.97 %	761.09
a1000RandGraph.2	1000	8201	753.36	1.66984	772.91	2.60 %	8.2212	777.73	3.24 %	885.22
a1000RandGraph.3	1000	8107	831.58	1.601	846.75	1.82 %	7.51468	840.37	1.06 %	942.36
a12000RandGraph.1.2	12000	96093		700.505	7171.49	-			-	
a12000RandGraph.1.5	12000	96391		695.997	8311.96	-			-	
a12000RandGraph.2	12000	95987		641.529	9304.61	-			-	
a12000RandGraph.3	12000	96449		595.506	10243	-			-	
a1200RandGraph.1.2	1200	9448	705.67	2.85236	713.77	1.15 %	18.5895	771.49	9.33 %	731.14
a1200RandGraph.1.5	1200	9625	810.96	2.69508	836.05	3.09 %	7.83735	855.21	5.46 %	893.22
a1200RandGraph.2	1200	9546	907.26	2.53694	930.11	2.52 %	6.68966	948.19	4.51 %	1019.89
a1200RandGraph.3	1200	9451	1012.45	2.32031	1028.49	1.58 %	13.6537	1027.39	1.48 %	1147.84
a14000RandGraph.1.2	14000	112016		1021.43	8378.89	-			-	
a14000RandGraph.1.5	14000	112228		949.407	9745.07	-			-	
a14000RandGraph.2	14000	112369		905.244	10921.5	-			-	
a14000RandGraph.3	14000	111869		826.446	11984.4	-			-	
a1400RandGraph.1.2	1400	11192	810.63	4.05229	822.78	1.50 %	25.7468	904.05	11.52 %	847.85
a1400RandGraph.1.5	1400	11226	938.93	3.63869	964.41	2.71 %	19.3872	1008.37	7.40 %	1046.61
a1400RandGraph.2	1400	11100	1051.01	3.44876	1082.44	2.99 %	23.2226	1095.12	4.20 %	1198.77
a1400RandGraph.3	1400	11263	1159.89	3.27949	1177.9	1.55 %	17.5067	1172.03	1.05 %	1335.61
a1600RandGraph.1.2	1600	12869	944.06	5.54467	956.36	1.30 %	35.2726	1036.45	9.79 %	990.86
a1600RandGraph.1.5	1600	12739	1079.39	5.33279	1114.41	3.24 %	32.4568	1139.31	5.55 %	1184.68
a1600RandGraph.2	1600	12779	1217.15	4.63159	1247.21	2.47 %	31.3172	1254.82	3.10 %	1370.81
a1600RandGraph.3	1600	12963	1351.98	5.67968	1375.72	1.76 %	21.736	1365.04	0.97 %	1536.49
a1800RandGraph.1.2	1800	14473	1061.39	7.5153	1074.44	1.23 %	45.6677	1160.86	9.37 %	1107.24
a1800RandGraph.1.5	1800	14222	1218.95	7.0611	1255.38	2.99 %	31.2769	1294.74	6.22 %	1366.4
a1800RandGraph.2	1800	14329	1364.89	6.25448	1396.25	2.30 %	28.2761	1409.32	3.25 %	1574.33
a1800RandGraph.3	1800	14531	1507.27	5.78265	1535.59	1.88 %	29.2898	1520.24	0.86 %	1727.25
a2000RandGraph.1.2	2000	16008	1152.49	9.4537	1167	1.26 %	56.0883	1265.11	9.77 %	1194.13
a2000RandGraph.1.5	2000	15835	1330.79	9.90182	1366.04	2.65 %	43.2223	1449.13	8.89 %	1456
a2000RandGraph.2	2000	16062	1483.84	8.564	1521.31	2.53 %	37.5926	1530.87	3.17 %	1756.03
a2000RandGraph.3	2000	15751	1669.35	7.32777	1696.26	1.61 %	39.7952	1693.1	1.42 %	1876.46
a3000RandGraph.1.2	3000	24045	1781.61	28.3301	1807.15	1.43 %	130.859	1934.84	8.60 %	1857.91
a3000RandGraph.1.5	3000	23852	2028.62	23.6221	2088.75	2.96 %	117.537	2180.83	7.50 %	2232.99
a3000RandGraph.2	3000	24065	2283.6	24.6345	2337.81	2.37 %	109.934	2353.94	3.08 %	2615.55
a3000RandGraph.3	3000	24026	2537.2	22.2465	2585.06	1.89 %	125.874	2572.42	1.39 %	2858.34
a4000RandGraph.1.2	4000	32087	2397.53	60.3061	2427.67	1.26 %	429.619	2608.95	8.82 %	2509.95
a4000RandGraph.1.5	4000	32119	2735.37	54.2355	2810.17	2.73 %	230.682	2899.56	6.00 %	3095.99
a4000RandGraph.2	4000	31880	3072.61	52.6689	3158.59	2.80 %	359.62	3175.78	3.36 %	3613.91
a4000RandGraph.3	4000	32025	3406.62	48.1988	3466.99	1.77 %	220.884	3445.14	1.13 %	3662.63
a6000RandGraph.1.2	6000	47899		155.242	3589.69	-			-	
a6000RandGraph.1.5	6000	48077		148.892	4171.62	-			-	
a6000RandGraph.2	6000	48069		133.653	4672.55	-			-	
a6000RandGraph.3	6000	47915		128.38	5144.55	-			-	
a8000RandGraph.1.2	8000	64373		298.129	4784.26	-			-	
a8000RandGraph.1.5	8000	63812		287.36	5549.48	-			-	
a8000RandGraph.2	8000	63874		252.572	6206.75	-			-	
a8000RandGraph.3	8000	64177		247.08	6838.39	-			-	

Tabla A.9: Cologne

Instancia	0	(0, 0.6065)	[0.6065, 1)	1
i101M1	735	11	0	2
i101M2	688	50	0	10
i101M3	689	49	0	10
i102M1	748	0	0	1
i102M2	675	63	0	11
i102M3	675	63	0	11
i103M1	743	6	0	2
i103M2	684	55	0	12
i103M3	680	58	0	13
i104M2	729	9	0	3
i104M3	729	9	0	3
i105M1	740	0	0	1
i105M2	717	21	0	3
i105M3	717	21	0	3

Instancia	0	(0, 0.6065)	[0.6065, 1)	1
i201M2	1741	59	0	3
i201M3	1692	103	0	8
i201M4	1693	101	0	9
i202M2	1738	59	0	7
i202M3	1704	90	0	10
i202M4	1704	90	0	10
i203M2	1717	82	0	10
i203M3	1688	107	0	14
i203M4	1682	112	0	15
i204M2	1777	21	0	3
i204M3	1718	79	0	4
i204M4	1718	79	0	4
i205M2	1708	92	0	10
i205M3	1697	101	0	12
i205M4	1697	100	0	13

Tabla A.10: CRR

Instancia	0	(0, 0.6065)	[0.6065, 1)	1
C01-A	477	18	0	5
C01-B	477	18	0	5
C02-A	469	22	0	9
C02-B	468	22	0	10
C03-A	372	76	0	52
C03-B	329	88	0	83
C04-A	328	102	0	70
C04-B	261	119	0	120
C05-A	263	93	0	144
C05-B	160	101	0	239
C06-A	482	14	0	4
C06-B	479	16	0	5
C07-A	472	19	0	9
C07-B	467	23	0	10
C08-A	357	83	0	60
C08-B	328	90	0	82
C09-A	321	93	0	86
C09-B	287	92	0	121
C10-A	240	96	0	164
C10-B	177	82	0	241
C11-A	486	9	0	5
C11-B	486	9	0	5
C12-A	475	15	0	10
C12-B	474	16	0	10
C13-A	360	69	0	71
C13-B	354	63	0	83
C14-A	335	63	0	102
C14-B	320	58	0	122
C15-A	220	75	0	205
C15-B	193	60	0	247
C16-A	489	6	0	5
C16-B	489	6	0	5
C17-A	480	10	0	10
C17-B	478	12	0	10
C18-A	392	32	0	76
C18-B	377	40	0	83
C19-A	366	24	0	110
C19-B	359	17	0	124
C20-A	270	13	0	217
C20-B	247	6	0	247

Instancia	0	(0, 0.6065)	[0.6065, 1)	1
D01-A	983	14	0	3
D01-B	976	19	0	5
D02-A	959	33	0	8
D02-B	939	51	0	10
D03-A	752	152	0	96
D03-B	631	213	0	156
D04-A	697	162	0	141
D04-B	565	198	0	237
D05-A	536	187	0	277
D05-B	319	203	0	478
D06-A	975	20	0	5
D06-B	975	20	0	5
D07-A	969	21	0	10
D07-B	969	21	0	10
D08-A	753	138	0	109
D08-B	656	185	0	159
D09-A	642	193	0	165
D09-B	535	221	0	244
D10-A	486	169	0	345
D10-B	365	150	0	485
D11-A	983	12	0	5
D11-B	983	12	0	5
D12-A	970	20	0	10
D12-B	970	20	0	10
D13-A	733	131	0	136
D13-B	701	136	0	163
D14-A	642	153	0	205
D14-B	588	167	0	245
D15-A	442	154	0	404
D15-B	377	134	0	489
D16-A	988	7	0	5
D16-B	988	7	0	5
D17-A	975	15	0	10
D17-B	975	15	0	10
D18-A	772	84	0	144
D18-B	751	85	0	164
D19-A	696	86	0	218
D19-B	681	72	0	247
D20-A	524	40	0	436
D20-B	477	29	0	494

Tabla A.11: Hy H2

Instancia	0	(0, 0.6065)	[0.6065, 1)	1
hc10p	246	271	0	507
hc10u	578	204	0	242
hc11p	509	527	0	1012
hc11u	904	612	0	532
hc6p	13	19	0	32
hc6u	41	9	0	14
hc7p	35	31	0	62
hc7u	81	18	0	29
hc8p	67	65	0	124
hc8u	152	37	0	67
hc9p	126	132	0	254
hc9u	291	80	0	141

Instancia	0	(0, 0.6065)	[0.6065, 1)	1
hc10p2	253	265	0	506
hc10u2	717	129	0	178
hc11p2	491	543	0	1014
hc11u2	1445	254	0	349
hc6p2	13	19	0	32
hc6u2	48	7	0	9
hc7p2	33	33	0	62
hc7u2	90	16	0	22
hc8p2	67	64	0	125
hc8u2	183	33	0	40
hc9p2	123	136	0	253
hc9u2	372	58	0	82

Tabla A.12: i640

Instancia	0	(0, 0.6065)	[0.6065, 1)	1
i640-001	605	27	0	8
i640-002	611	22	0	7
i640-003	616	18	0	6
i640-004	605	27	0	8
i640-005	601	31	0	8
i640-011	625	9	0	6
i640-012	624	10	0	6
i640-013	623	10	0	7
i640-014	622	9	0	9
i640-015	620	11	0	9
i640-031	616	17	0	7
i640-032	624	10	0	6
i640-033	615	18	0	7
i640-034	612	20	0	8
i640-035	621	14	0	5
i640-041	628	5	0	7
i640-042	629	4	0	7
i640-043	632	3	0	5
i640-044	627	6	0	7
i640-045	630	4	0	6
i640-101	568	51	0	21
i640-102	575	45	0	20
i640-103	572	46	0	22
i640-104	573	48	0	19
i640-105	558	59	0	23
i640-111	602	20	0	18
i640-112	594	24	0	22
i640-113	596	22	0	22
i640-114	596	24	0	20
i640-115	594	22	0	24
i640-131	578	42	0	20
i640-132	566	50	0	24
i640-133	586	34	0	20
i640-134	579	42	0	19
i640-135	587	33	0	20
i640-141	601	15	0	24
i640-142	610	10	0	20
i640-143	612	10	0	18
i640-144	607	11	0	21
i640-145	606	12	0	22

Instancia	0	(0, 0.6065)	[0.6065, 1)	1
i640-201	529	70	0	41
i640-202	505	91	0	44
i640-203	533	69	0	38
i640-204	546	58	0	36
i640-205	514	84	0	42
i640-211	560	36	0	44
i640-212	567	34	0	39
i640-213	562	37	0	41
i640-214	558	41	0	41
i640-215	561	39	0	40
i640-231	534	60	0	46
i640-232	533	63	0	44
i640-233	538	61	0	41
i640-234	529	68	0	43
i640-235	547	58	0	35
i640-241	570	25	0	45
i640-242	580	17	0	43
i640-243	578	21	0	41
i640-244	582	20	0	38
i640-245	574	25	0	41
i640-301	376	124	0	140
i640-302	373	127	0	140
i640-303	374	130	0	136
i640-304	398	113	0	129
i640-305	360	143	0	137
i640-311	422	75	0	143
i640-312	451	56	0	133
i640-313	431	72	0	137
i640-314	437	70	0	133
i640-315	434	73	0	133
i640-331	371	134	0	135
i640-332	393	110	0	137
i640-333	373	123	0	144
i640-334	388	111	0	141
i640-335	379	124	0	137
i640-341	465	41	0	134
i640-342	456	46	0	138
i640-343	467	34	0	139
i640-344	460	40	0	140
i640-345	454	46	0	140

Tabla A.13: JMP

Instancia	0	(0, 0.6065)	[0.6065, 1)	1
K100.1	74	16	0	10
K100.10	70	22	0	8
K100.2	73	14	0	13
K100.6	74	17	0	9
K100.7	72	16	0	12
K100.9	81	11	0	8
K400	287	64	0	49
K400.2	279	66	0	55
K400.4	282	69	0	49
P100	53	14	0	33
P100.1	53	22	0	25
P100.2	76	10	0	14
P100.3	66	12	0	22
P100.4	63	9	0	28
P200	130	31	0	39
P400.1	231	64	0	105
P400.2	248	63	0	89

Tabla A.14: PUCNU

Instancia	0	(0, 0.6065)	[0.6065, 1)	1
bip42nu	1010	84	0	106
bip52nu	2027	71	0	102
bip62nu	1046	51	0	103
bipa2nu	3079	83	0	138
bipe2nu	509	16	0	25
cc10-2nu	871	88	0	65
cc11-2nu	1745	178	0	125
cc12-2nu	3538	332	0	226
cc3-10nu	958	24	0	18
cc3-11nu	1258	39	0	34
cc3-12nu	1643	44	0	41
cc3-4nu	54	5	0	5
cc3-5nu	109	8	0	8
cc5-3nu	209	21	0	13
cc6-2nu	52	7	0	5
cc6-3nu	628	63	0	38
cc7-3nu	1930	142	0	115
cc9-2nu	434	48	0	30

Tabla A.15: Random Graphs

Instancia	0	(0, 0.6065)	[0.6065, 1)	1	Instancia	0	(0, 0.6065)	[0.6065, 1)	1
a0200RandGraph.1.2	152	15	0	33	a1400RandGraph.1.2	1035	140	0	225
a0200RandGraph.1.5	116	8	0	76	a1400RandGraph.1.5	840	113	0	447
a0200RandGraph.2	78	6	0	116	a1400RandGraph.2	664	57	0	679
a0200RandGraph.3	57	3	0	140	a1400RandGraph.3	461	17	0	922
a0400RandGraph.1.2	285	51	0	63	a1600RandGraph.1.2	1170	160	0	270
a0400RandGraph.1.5	226	28	0	146	a1600RandGraph.1.5	988	98	0	514
a0400RandGraph.2	192	18	0	190	a1600RandGraph.2	777	54	0	769
a0400RandGraph.3	126	8	0	266	a1600RandGraph.3	505	18	0	1077
a0600RandGraph.1.2	426	67	0	107	a1800RandGraph.1.2	1311	182	0	307
a0600RandGraph.1.5	355	35	0	210	a1800RandGraph.1.5	1070	118	0	612
a0600RandGraph.2	280	22	0	298	a1800RandGraph.2	843	65	0	892
a0600RandGraph.3	191	15	0	394	a1800RandGraph.3	584	24	0	1192
a0800RandGraph.1.2	601	76	0	123	a2000RandGraph.1.2	1495	192	0	313
a0800RandGraph.1.5	490	44	0	266	a2000RandGraph.1.5	1221	154	0	625
a0800RandGraph.2	377	27	0	396	a2000RandGraph.2	964	71	0	965
a0800RandGraph.3	245	20	0	535	a2000RandGraph.3	669	37	0	1294
a1000RandGraph.1.2	728	109	0	163	a3000RandGraph.1.2	2223	288	0	489
a1000RandGraph.1.5	602	78	0	320	a3000RandGraph.1.5	1807	198	0	995
a1000RandGraph.2	472	37	0	491	a3000RandGraph.2	1374	111	0	1515
a1000RandGraph.3	334	16	0	650	a3000RandGraph.3	948	61	0	1991
a1200RandGraph.1.2	885	133	0	182	a4000RandGraph.1.2	2930	402	0	668
a1200RandGraph.1.5	715	83	0	402	a4000RandGraph.1.5	2363	260	0	1377
a1200RandGraph.2	556	58	0	586	a4000RandGraph.2	1834	126	0	2040
a1200RandGraph.3	364	26	0	810	a4000RandGraph.3	1244	65	0	2691