UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"
DECANATO DE CIENCIA Y TECNOLOGÍA
MAESTRÍA EN CIENCIA/MENCIÓN OPTIMIZACIÓN

Trabajo de Grado

EL PROBLEMA DE ASIGNACIÓN LOCAL-VISITANTE EN CALENDARIOS DEPORTIVOS

Lic. Jorge Perdomo

Barquisimeto, Junio 2011

UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"
DECANATO DE CIENCIA Y TECNOLOGÍA
MAESTRÍA EN CIENCIA/MENCIÓN OPTIMIZACIÓN

Trabajo de Grado

EL PROBLEMA DE ASIGNACIÓN LOCAL-VISITANTE EN CALENDARIOS DEPORTIVOS

Trabajo de Grado presentado como Requisito Parcial para optar al Título de Magister Scientiarium, Mención Optimización.

Autor: Lic. Jorge Perdomo
Tutor: Dr. Hugo Lara

Barquisimeto, Junio 2011

DEDICATORIA

A mi madre, por quien soy.
A mi hija y a mi nieto, que son mi inspiración.

AGRADECIMIENTO

A esta prestigiosa universidad, que amablemente
me recibió en su seno.
A mis profesores, auténticos maestros.
A mis compañeros de estudio, de quienes
recibí gran apoyo.

UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"
DECANATO DE CIENCIA Y TECNOLOGÍA
MAESTRÍA EN CIENCIA/MENCIÓN OPTIMIZACIÓN

Trabajo de Grado

EL PROBLEMA DE ASIGNACIÓN LOCAL-VISITANTE EN CALENDARIOS DEPORTIVOS

Trabajo de Grado presentado como Requisito Parcial para optar al Título de Magister Scientiarium, Mención Optimización.

Autor: Lic. Jorge Perdomo
Tutor: Dr. Hugo Lara
Fecha: Junio, 2011

RESUMEN

En Investigación de Operaciones, el problema de elaboración de calendarios deportivos se ha convertido en un tópico sobresaliente, con diversas variantes que surgen según cuáles sean las condiciones del torneo deportivo y con cuál técnica de optimización abordarlo [8].
En la variante de torneos *double round robin*, de ida y vuelta, el problema se formula asignando la etiqueta de *local* (home) o *visitante* (away), a cada equipo y en cada partido de un itinerario preestablecido, de manera que se minimice el recorrido total de los equipos durante el torneo. En su formulación matemática original el problema de asignación local-visitante resulta ser un COP de grafo no dirigido, siendo definidos los conjuntos de vértices y aristas a partir de las condiciones del torneo. En este marco se aplica la formulación Min Res Cut, con la cual se deriva una formulación entera cuadrática binaria del problema. Se trata de un problema tipo NP-completo, por tanto computacionalmente inviable, para cuya resolución se proponen diversos algoritmos de aproximación. Con el software matlab MIQP [1], se resuelve mediante una técnica de ramificación y acotamiento, utilizando en cada iteración una relajación de programación cuadrática con restricciones lineales, o alternativamente, una relajación de programación no lineal con una única restricción lineal. Finalmente, con el software matlab SDPT3-01 [10], se resuelve en el marco de la programación semidefinida mediante el algoritmo de punto interior, obteniéndose una matriz definida positiva minimizadora no entera. Esta matriz produce, con muy buena precisión, una aproximación a la solución óptima del problema original mediante una adaptación [11] del algoritmo de aproximación aleatoria de Goemans y Williamson, el cual se corre en tiempo polinomial.

Índice general

Introducción

En el campo de la Investigación de Operaciones la confección de calendarios, sujeta a determinados criterios, se ha convertido en una prolífica disciplina de investigación que tiene su aplicación en la elaboración de diversos tipos de calendarios, como los escolares, de transporte, deportivos y laborales, entre otras utilidades.
La aplicación directamente relacionada con este trabajo, consiste en determinar el momento y el lugar en el que debe realizarse un juego en un torneo deportivo. En general, el problema trata de la elaboración de calendarios deportivos y de los diversos criterios que definen las condiciones exigidas para lograr un torneo satisfactorio en lo deportivo y en lo económico. En un trabajo relativamente reciente, Ribeiro [8] presenta una revisión introductoria de los aspectos relevantes a ser considerados en la determinación del mejor calendario para un torneo deportivo, cubriendo sus principales aplicaciones prácticas, así como métodos de solución y algoritmos para alcanzar estas soluciones. Entre los criterios destaca la situación de minimizar la distancia total de recorrido (*traveling tournament problem*) [2]. Otra situación se presenta cuando el objetivo es minimizar, para cada equipo, el número de pares de juegos consecutivos en casa o como visitante (*number of breaks*) [12]. Existe la variante en la que se busca minimizar el número de encuentros consecutivos contra los equipos más fuertes (*carry-over effects value*) [9].
En su inicio, este trabajo se inspiró en una tarea durante la escolaridad de la maestría consistente en analizar el documento de Suzuka, Miyashiro, Yoshise and Matsui, [11], en el cual, de manera bastante sucinta, muestran la resolución del problema de asignación local-visitante sobre la base de la programación semidefinida. El mismo está enfocado en la elaboración de un calendario deportivo para torneos *todos contra todos* del tipo *double round robin* en el que participa un número par de equipos, $2n, n \in \mathbb{N}$. Cada pareja de equipos afronta dos encuentros turnándose la sede, esto es, si en un partido uno de los dos equipos juega de local, en el siguiente lo hará como visitante, o viceversa. Es así como en este trabajo, se estudiará el diseño de un calendario deportivo bajo el criterio *traveling tournament problem*.
En el Capítulo 1. presentamos una interpretación exhaustiva del modelo *double round robin*. Definimos la *data* como matrices mostradas en formato de tablas o cuadros: la única matriz distancia, cuyas entradas son las distancias entre las sedes de los equipos, y la matriz itinerario, cuyas entradas representan el número asignado al equipo oponente en cada encuentro. Definimos también la matriz asignación, cuyas entradas de carácter binario, indican si un equipo juega de local o de visitante. El objetivo de nuestro problema es hallar la matriz asignación que minimice la distancia total de recorrido de todos los equipos.
La interpretación del modelo de Suzuka (y otros)[11] continúa en el Capítulo 4. Sin embargo, anteponemos el Capítulo 2. para expresar en términos matemáticos el problema de asignación local-visitante. Y el Capítulo 3. para expresar en términos combinatorios el modelo descrito hasta ahora. En el Capítulo 2., nos proponemos obtener una expresión matemática de la distancia total de recorrido de los equipos que permita resolver el problema de asignación mediante el recurso rudimentario de la

enumeración exhaustiva de todas las soluciones factibles. A sabiendas de que esta práctica es computacionalmente aplicable únicamente a pequeños valores de n, esperamos que estos resultados nos serán particularmente útiles para evaluar otros algoritmos aplicados para la resolución aproximada del problema.
En el capítulo anterior puede verse cómo la exposición del problema de asignación local-visitante puede realizarse formalmente en términos combinatorios. En el Capítulo 3. definimos un grafo, $G = (V, E)$, donde cada vértice está asociado a cada entrada de la matriz asignación. Para una solución factible, los vértices asociados a las entradas con valor A se agrupan en un subconjunto de vértices V', el cual define a su vez una bipartición $\delta(V')$ sobre el conjunto de vértices. En términos de estos conjuntos se expresan tanto el recorrido total como las restricciones.
En el Capítulo 4. continuamos analizando el modelo de Suzuka (y otros)[11], esto es, la aplicación de la técnica del problema Min Res Cut a nuestro problema de asignación. Ésta consiste en redefinir el conjunto de vértices V agregando un vértice específico r, e igualmente redefinir el conjunto de aristas E agregando las que tienen a r como vértice extremo. En esta formulación, tanto el recorrido total como las restricciones se expresan en término de las aristas de E. Su utilidad consiste en que, mediante manipulaciones algebraicas sencillas, se busca obtener una fórmula no condicionada para el recorrido total de los equipos.
En el Capítulo 5. nos proponemos desarrollar una formulación entera binaria del modelo combinatorio del problema de asignación local-visitante. El conjunto binario del par de valores que pueden asumir las variables asociadas a las aristas del grafo debe ser $\{0, 1\}$. Sin embargo, éstas deben expresarse en función de otras variables asociadas a los vértices, las cuales pueden asumir valores en $\{z_1, z_2\}$, con $z_1, z_2 \in \mathbb{Z}$. En este trabajo manejamos dos alternativas, $\{0, 1\}$ y $\{-1, +1\}$, aunque los experimentos computacionales los haremos sólo para la opción $\{0, 1\}$.
Dado el carácter NP-Completo del programa entero cuadrático binario, para la resolución del problema de asignación local-visitante proponemos algunas relajaciones, las cuales pueden dar un resultado óptimo con bastante precisión y en un tiempo relativamente corto. Está claro que estos resultados serían sólo burdas aproximaciones al óptimo del problema original. En el Capítulo 6. estudiamos una relajación de programación no lineal, con restricción no lineal, del programa entero cuadrático binario del problema de Asignación HA, ubicando el conjunto de soluciones factibles sobre una hiperesfera en un hiperespacio de dimensión igual al número de variables. Para su evaluación utilizaremos el código matlab fmincon.m [7]. Por otra parte, proponemos también la relajación de programación cuadrática (continua) con las restricciones lineales originales. Para esta relajación aplicaremos el código quadprog.m [7]. Por último, proponemos una relajación de programación no lineal, ahora con restricción lineal, en la que el conjunto de soluciones factibles se ubica sobre un hiperplano en el mismo hiperespacio del caso no lineal. En este mismo capítulo, utilizando estas relajaciones, intentamos resolver el programa entero cuadrático binario mediante el algoritmo de ramificación y acotamiento. El propósito de la aplicación de este algoritmo, es contar con una referencia para evaluar el programa semidefinido, lo cual constituye nuestro principal objetivo. Por último, ideamos una variante del algoritmo de ramificación y acotamiento que aplica una heurística codiciosa para la ramificación.
En el Capítulo 7., culminando aquí la interpretación del trabajo de Suzuka (y otros)[11], exponemos en detalle la aplicación del programa semidefinido como una relajación del programa entero cuadrático binario. Siguiendo el desarrollo propuesto por Helmberg [5] buscamos obtener una formulación primal-dual que nos pueda ser útil para resolver el programa semidefinido aplicando la técnica de punto interior. El resultado, (posiblemente) óptimo, se espera que sea una matriz definida positiva. Siendo así, mediante descomposición de Cholesky podrá obtenerse el conjunto de vectores que relajan las variables enteras binarias originales. Debido a la relajación, sólo se sabe de estos vectores que corresponderán a una aproximación de la solución óptima, sin tener mayores indicios de con cuáles

valores enteros se identifican. Se espera resolver este inconveniente mediante el método de aproximación aleatoria de Goemans y Williamson [4].
Reservamos el Capítulo 8. para mostrar los resultados de todos los cálculos computacionales llevados a cabo. Para cada algoritmo, se presenta una tabla de resultados donde se destacan, el número de iteraciones necesarias para obtener el óptimo, el tiempo invertido y el óptimo de la función objetivo.
Finalmente, en el Capítulo 9. presentamos las conclusiones de nuestro trabajo, destacando los principales logros alcanzados en cada capítulo tanto en los aspectos teóricos como en los experimentos computacionales, haciendo énfasis en la comparación entre el algoritmo de ramificación y acotamiento y la relajación semidefinida junto con el algoritmo de aproximación aleatoria de Goemans y Williamson [4]. Por último, indicamos algunas recomendaciones.
La data utilizada para los experimentos computacionales están en el Apéndice A. Pueden verse en formato de cuadros y de códigos.m, los itinerarios y las distancias que nos sirvieron para realizar todos los cálculos computacionales para los valores de n desde 2 hasta 6.
Todos los códigos diseñados para la experiencia computacional se encuentran en el Apéndice B. No se confeccionaron interactivos, de manera que cada aplicación fuese válida para cualquier valor de n, sino que mostramos los códigos para $n = 2$. Para los demás casos son totalmente similares.
En el Apéndice C. incluimos un estudio sobre la confección de matrices itinerario y de distancias. Ello con el interés de crear la data necesaria para realizar los experimentos computacionales para cualquier valor finito (manejable) de n.

Capítulo 1

El Problema de Asignación HA

1.1. Problema Original

Siguiendo la exposición de Suzuka (y otros) [11]), originalmente el problema consiste en elaborar el calendario deportivo correspondiente a un torneo del tipo *double round robin*, esto es, un torneo de exactamente una ida y una vuelta para cada par de equipos.

1.2. La Matriz Itinerario

Para su formulación se tiene preestablecido un itinerario, el cual puede presentarse como una matriz cuyas filas corresponden a los equipos participantes y cuyas columnas se reservan para las casillas, donde cada casilla define la ocurrencia de un partido. La dimensión de la matriz itinerario se obtiene de las siguientes condiciones del torneo:

- El número de equipos (o jugadores, etc.) es $2n$, donde $n \in \mathbb{N}$. Con lo cual se establece que participa una cantidad par de equipos en el torneo.
- El número de casillas es $2(2n-1)$. Lo cual resulta de establecer que cada pareja de equipos protagoniza dos encuentros.

Sean, T, el conjunto de equipos participantes y, S, el conjunto de casillas del torneo. Si mediante $\tau(t,s)$, $(t,s) \in T \times S$, se designa al oponente del equipo $t \in T$, en la casilla $s \in S$, entonces la matriz itinerario $\mathcal{T}$ tiene la forma plasmada en el Cuadro 1.1.

Cuadro 1.1: Matriz Itinerario.

$\mathcal{T}$	$s=1$	$s=2$	$s=3$	$\cdots$	$s=4n-2$
$t=1$	$\tau(1,1)$	$\tau(1,2)$	$\tau(1,3)$	$\cdots$	$\tau(1,4n-2)$
$t=2$	$\tau(2,1)$	$\tau(2,2)$	$\tau(2,3)$	$\cdots$	$\tau(2,4n-2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$t=2n$	$\tau(2n,1)$	$\tau(2n,2)$	$\tau(2n,3)$	$\cdots$	$\tau(2n,4n-2)$

Una matriz itinerario puede confeccionarse siguiendo las siguientes condiciones del torneo *double round robin*:

- Cada equipo juega un partido en cada casilla.
  Según esto, $\forall s \in S$, $\forall t \in T$, $\exists t' \in T$, $t \neq t'$, tal que, $t' = \tau(t, s)$.

- Cada equipo juega con cada otro equipo dos veces.
  Entonces, $\forall t \in T$, $\exists s, s' \in S$, $s \neq s'$ y $\exists t' \in T$, $t \neq t'$, tal que, $t' = \tau(t, s) = \tau(t, s')$.

Por ejemplo, para $n = 2$ se tienen: $|T| = 4$ y $|S| = 6$.
Para un valor cualquiera de n es posible confeccionar distintas matrices itinerario. Un ejemplo es el Cuadro 1.2., para $n = 2$:

Cuadro 1.2: Matriz Itinerario para $n = 2$.

$\mathcal{T}$	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$
$t=1$	2	3	2	4	3	4
$t=2$	1	4	1	3	4	3
$t=3$	4	1	4	2	1	2
$t=4$	3	2	3	1	2	1

En el Apéndice C. explicamos con más detalle la confección de matrices itinerario para torneos *single round robin* y *double round robin* y para cualesquiera valores de n. También aparece un código que puede ser aplicado con ese fin.

1.3. La Matriz Asignación

Dado un itinerario prefijado, el problema de asignación local-visitante consiste en determinar, para cada casilla y para cada pareja de equipos, cuál equipo hará las veces de local (***H**ome*) y cuál de visitante (***A**way*). Con este objetivo se construye la matriz asignación $\mathcal{A}$ de igual dimensión que la matriz itinerario tal como se muestra en el Cuadro 1.3.

Cuadro 1.3: Matriz Asignación.

$\mathcal{A}$	$s=1$	$s=2$	$s=3$	$\cdots$	$s=4n-2$
$t=1$	$a(1,1)$	$a(1,2)$	$a(1,3)$	$\cdots$	$a(1,4n-2)$
$t=2$	$a(2,1)$	$a(2,2)$	$a(2,3)$	$\cdots$	$a(2,4n-2)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$t=2n$	$a(2n,1)$	$a(2n,2)$	$a(2n,3)$	$\cdots$	$a(2n,4n-2)$

Las entradas de la matriz asignación, $a(t, s)$, o bien, $a_{t\,s}$, $(t, s) \in T \times S$, pueden asumir los siguientes valores:

$$a(t,s) = \begin{cases} H & \text{si el equipo } t \text{ juega como } \textit{local} \text{ en la casilla } s. \\ A & \text{si el equipo } t \text{ juega como } \textit{visitante} \text{ en la casilla } s. \end{cases}$$

La asignación de los valores H o A está sujeta a las siguientes condiciones del torneo *double round robin*. Atendiendo a que cada equipo tiene su sede, ocurre que:

- Cada partido se juega en la sede de alguno de los dos equipos.
  Es decir, en cada casilla $s \in S, \{a_{t\,s}, a_{\tau(t,s)\,s}\} = \{A, H\}, \forall t \in T$.
- Cada equipo juega en la sede de cada otro equipo exactamente una vez.
  O sea, $\forall t \in T$, si $\tau(t,s) = \tau(t,s')$, siendo $s \neq s'$, entonces $\{a_{t\,s}, a_{t\,s'}\} = \{A, H\}$.

Con estas condiciones se puede establecer el concepto de *consistencia*.
Dada una matriz itinerario $\mathcal{T}$, se dice que una matriz asignación $\mathcal{A}$ es consistente con $\mathcal{T}$, si se cumple que:

(C1) $\forall (t,s) \in T \times S,\ \left\{a_{t\,s}, a_{\tau(t,s)\,s}\right\} = \{A, H\}$

(C2) $\forall t \in T,\ [\tau(t,s) = \tau(t,s'),\ s \neq s'] \Rightarrow \{a_{t\,s}, a_{t\,s'}\} = \{A, H\}$

Una asignación, de las muchas posibles, consistente con el itinerario prefijado para $n = 2$ que aparece en el Cuadro 1.2., puede verse en el Cuadro 1.4.

Cuadro 1.4: Matriz Asignación consistente con el Cuadro 1.2.

$\mathcal{A}$	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$
$t=1$	H	H	A	A	A	H
$t=2$	A	H	H	H	A	A
$t=3$	A	A	H	A	H	H
$t=4$	H	A	A	H	H	A

1.4. Calendario Deportivo

Para un $n \in \mathbb{N}$ dado, sea $\mathcal{T}$ la matriz itinerario seleccionada, de entre las distintas que se pueden confeccionar de acuerdo con las condiciones establecidas para construir los Cuadros 1.1. y 1.2. Sea $\mathcal{A}$ la matriz asignación construida de acuerdo con las condiciones de consistencia con la matriz $\mathcal{T}$ seleccionada.
Un Calendario Deportivo es un par $(\mathcal{T}, \mathcal{A})$ de una matriz itinerario y una matriz asignación consistente con la matriz itinerario.
El problema de asignación local-visitante consiste en obtener el calendario que minimice el costo de recorrido de los equipos. En un sentido amplio, el costo puede hacer referencia a variables de gastos, de tiempo o de longitud de recorrido. En este trabajo, para obtener el mejor calendario, se evaluará la longitud de recorrido total de los equipos a lo largo de todo el torneo.

1.5. La Matriz Distancia

La Matriz Distancia $\mathcal{D}$ es una matriz con diagonal cero cuyas filas y columnas tienen índices en T y cada entrada $d(t, t')$ denota la distancia desde la sede del equipo t a la sede del equipo t'.
En principio, para el desarrollo de la formulación matemática en el siguiente capítulo, no se supone simetría para la matriz D, ni que sus elementos verifiquen la desigualdad triangular. Sin embargo, en la práctica, tales propiedades son necesarias para efectos de cálculo con matrices semidefinidas positivas, como ocurre con la formulación entera y con el programa semidefinido.

Cuadro 1.5: Matriz Distancia.

$\mathcal{D}$	$t=1$	$t=2$	$t=3$	$\cdots$	$t=2n$
$t=1$	0	$d(1,2)$	$d(1,3)$	$\cdots$	$d(1,2n)$
$t=2$	$d(2,1)$	0	$d(2,3)$	$\cdots$	$d(2,2n)$
$t=3$	$d(3,1)$	$d(3,2)$	0	$\cdots$	$d(2,2n)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$t=2n$	$d(2n,1)$	$d(2n,2)$	$d(2n,3)$	$\cdots$	0

En el Apéndice C. puede verse un código que fue diseñado para construir estas matrices, en las cuales las entradas son simétricas y verifican la desigualdad triangular.
Dado un par de matrices consistentes, itinerario y asignación local-visitante, la distancia de recorrido de un equipo t es la longitud de la ruta que comienza en la sede de t, la visita a las sedes donde los partidos se juegan en el orden definido por las matrices, itinerario y asignación local-visitante, y que culmina con el retorno a su sede al final del torneo.
La *distancia total de recorrido* es la suma de los recorridos de todos los equipos. De forma descriptiva, designando mediante d_t a la distancia total recorrida por el equipo t, y mediante d, a la distancia total de recorrido de todos los equipos, se tienen:

$$\begin{array}{rcl} d_t & = & d_{t0} + \sum_{s\in S} d_{ts} + d_{tf} \\ d & = & \sum_{t\in T} d_t \end{array} \tag{1.1}$$

donde, d_{t0}, es la distancia inicial recorrida desde la sede de t hasta la sede del equipo donde se realiza el primer partido, y d_{tf}, es la distancia de recorrido desde la sede del equipo donde se juega el último encuentro hasta la sede del equipo t.

1.6. El Problema Asignación HA

Una vez realizada la descripción detallada del torneo *double round robin*, puede plantearse la formulación del problema de asignación local(H)-visitante(A) como un problema de minimización:

Problema Asignación HA

Instancia: Una matriz itinerario $\mathcal{T}$ y una matriz distancia $\mathcal{D}$.

Tarea: Encontrar una matriz asignación $\mathcal{A}$, consistente con $\mathcal{T}$, que minimice a d.

Capítulo 2

Formulación Matemática

Formulación Matemática del Problema Asignación HA

Con la terminología utilizada para definir las matrices definidas para modelar el problema de asignación local-visitante, es posible plantear una formulación matemática del problema original. Se plantea conseguir una expresión matemática de la distancia total de recorrido de los equipos que participan en el torneo *double round robin*. Con ella se podrá determinar la matriz asignación, consistente con la matriz itinerario dada, que minimice dicha distancia total de recorrido.

2.1. Función Objetivo para el Problema Asignación HA

El recorrido de cada equipo, $t \in T$, ocurre desde la casilla $s = 1$ hasta la casilla $s = 4n - 2$ de la matriz itinerario, $\mathcal{T}$, a lo cual se agregan los recorridos inicial y final. Si se designa mediante, $l(t, s)$, al recorrido del equipo t desde la casilla s hasta la $s + 1$, éste es igual a:

$$l(t,s) = \begin{cases} d(t,t) = 0 & \text{si } (a_{t\,s}, a_{t\,s+1}) = (H,H) \\ d(\tau(t,s), \tau(t,s+1) & \text{si } (a_{t\,s}, a_{t\,s+1}) = (A,A) \\ d(t, \tau(t,s+1) & \text{si } (a_{t\,s}, a_{t\,s+1}) = (H,A) \\ d(\tau(t,s), t) & \text{si } (a_{t\,s}, a_{t\,s+1}) = (A,H) \end{cases} \tag{2.1}$$

Particularmente, el recorrido inicial es $l(t, 0)$, y el final, $l(t, 4n - 2)$. Aquí hay que disponer de las casillas adicionales, $s = 0$, cuando el equipo t está en su sede al inicio del campeonato, y $s = 4n - 1$, para describir el regreso a su sede, después del último encuentro desde la casilla $s = 4n - 2$, última de la matriz itinerario.
Como $d(t, t) = 0$, se tiene para estos recorridos, que:

$$l(t,0) = d(t, \tau(t,1)), \text{ si } (a_{t\,0}, a_{t\,1}) = (H,A) \tag{2.2}$$

$$l(t,4n-2) = d(\tau(t,4n-2), t), \text{ si } (a_{t\,4n-2}, a_{t\,4n-1}) = (A,H) \tag{2.3}$$

Con estas consideraciones, el recorrido total, d, de todos los equipos admite la expresión:

$$d = \sum_{t=1}^{2n} \sum_{s=0}^{4n-2} l(t,s) \tag{2.4}$$

Los valores de los términos que aparecen en el desarrollo de la suma (2.4), dependen de la solución factible que se esté evaluando. Esto es, de la matriz asignación. Por ejemplo, para $n = 2$, partiendo

del itinerario dado en el Cuadro 1.2., y evaluando la matriz asignación dada en el Cuadro 1.4., consistente con el primero, los recorridos $l(t,s)$ son los que aparecen en el Cuadro 2.1., recordando que $l(t,s)$ es el recorrido desde la casilla s hasta la $s+1$.

Cuadro 2.1: Recorridos de acuerdo con los Cuadros 1.2. y 1.4.

$\mathcal{L}$	$s=0$	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$
$t=1$	0	0	$d(1,2)$	$d(2,4)$	$d(4,3)$	$d(3,1)$	0
$t=2$	$d(2,1)$	$d(1,2)$	0	0	$d(2,4)$	$d(4,3)$	$d(3,2)$
$t=3$	$d(3,4)$	$d(4,1)$	$d(1,3)$	$d(3,2)$	$d(2,3)$	0	0
$t=4$	0	$d(4,2)$	$d(2,3)$	$d(3,4)$	0	$d(4,1)$	$d(1,4)$

(Aquí, por ejemplo: $l(3,4) = d(2,3)$)

2.2. Restricciones para el Problema Asignación HA

Como pudo verse, en la construcción de la matriz de recorridos $\mathcal{L}$, los valores que pueden asumir las entradas $l(t,s)$ dependen de los valores que asuman las $a(t,s)$ en una solución factible de la matriz asignación $\mathcal{A}$. A su vez, las condiciones de consistencia del par $(\mathcal{T},\mathcal{A})$, restringen los valores de estas entradas. Estas condiciones son:

$$\begin{cases} \forall (t,s) \in T \times S, & \{a_{t\,s}, a_{\tau(t,s)\,s}\} = \{A,H\} \\ \forall t \in T, [\tau(t,s) = \tau(t,s'),\ s \neq s'] \Rightarrow & \{a_{t\,s}, a_{t\,s'}\} = \{A,H\} \end{cases} \tag{2.5}$$

Siendo así, podemos formar grupos de cuatro entradas $a(t,s)$, cuyos valores dependen del valor que asuma cualquiera de ellas. Por tanto, es suficiente plantear un algoritmo en el cual se evalúe un *representante* de cada grupo. Llámese *grupo representante* al conjunto de estos representantes para un valor dado de n. Entonces, el algoritmo consiste en asignar, alternativamente a cada entrada del grupo representante, los valores A o H.
De acuerdo con lo discutido hasta aquí, la exposición formal del problema de optimización, adecuada para plantear el problema de asignación local-visitante, es la formulación combinatoria del mismo.

2.3. Resolución mediante Enumeración

En definitiva, lo que se ha planteado es un problema combinatorio que puede resolverse mediante enumeración exhaustiva. Como, $dim(\mathcal{A}) = (2n)(4n-2)$, entonces cada matriz asignación propuesta consta de $n(2n-1)$ grupos. La asignación alternativa de los valores A o H a los $n(2n-1)$ representantes, produce $2^{n(2n-1)}$ soluciones factibles. En el siguiente cuadro se muestra esta aseveración.

n	Soluciones Factibles	tiempo
2	64	0,03 seg.
3	32,768	6 seg.
4	268,435,456	14 horas
5	35,184,372,088,832	$\approx 10^8$ siglos
6	73,786,976,294,838,206,464	$\approx 10^{15}$ siglos
–	–	–

La explosión combinatoria es obvia. La enumeración completa de las soluciones posibles puede realizarse únicamente para pequeños valores de n. No obstante, un algoritmo para esta formulación arrojaría soluciones exactas, lo cual sería de mucha utilidad para evaluar algoritmos más *inteligentes*. En el Apéndice A. mostramos un código diseñado para resolver, mediante la enumeración de todas las soluciones factibles, el problema de asignación local-visitante para los caso más sencillo, $n = 2$, 3 y 4. A partir de $n = 5$, su resolución es computacionalmente inviable. Sin embargo, puede obtenerse una cota superior del valor objetivo óptimo mediante enumeración parcial aleatoria.

Capítulo 3

Formulación Combinatoria

3.1. Formulación Combinatoria del Problema Asignación HA

La formulación combinatoria del problema de asignación local-visitante, puede plantearse en los siguientes términos. Sea $G = (V, E)$ un grafo no dirigido, con un conjunto de vértices V y un conjunto de aristas E, como se muestran:

$$V = \{v_{t\,s} : (t, s) \in T \times S\} \tag{3.1}$$
$$E = \{\{v_{t\,s}, v_{t\,s+1}\} : (t, s) \in T \times (S \setminus \{4n - 2\})\} \tag{3.2}$$

Obviamente, el grafo es una representación de las matrices itinerario y asignación. De allí la conveniencia de organizar los vértices y las aristas, como se muestra en la Figura 3.1. para el caso $n = 2$, de acuerdo con el itinerario del Cuadro 1.2.

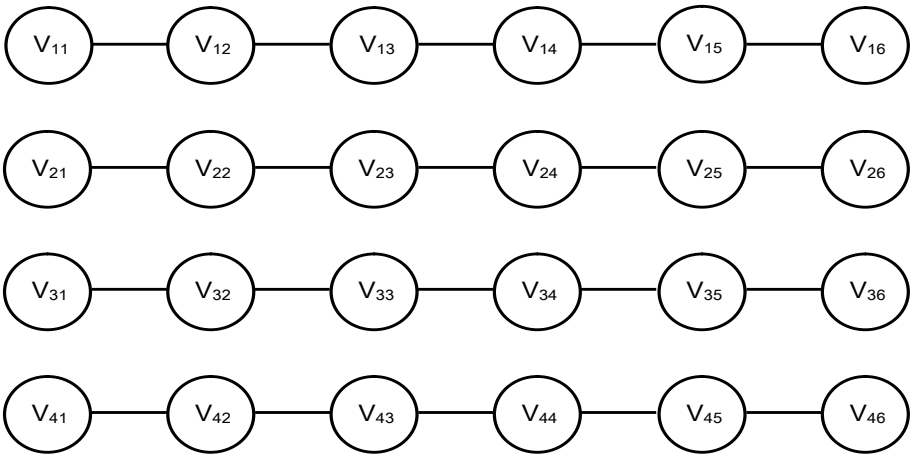

Figura 3.1: Grafo Asignación HA para $n = 2$.

En esta formulación, las soluciones factibles son las uniones de los conjuntos disjuntos de vértices, V' y $V \setminus V'$, con los cuales se construyen matrices asignación $\mathcal{A}$, como sigue:

$$\begin{aligned} v_{t\,s} \in V' &\quad \Leftrightarrow \quad a_{t\,s} = A \\ v_{t\,s} \notin V' &\quad \Leftrightarrow \quad a_{t\,s} = H \end{aligned} \tag{3.3}$$

3.2. La Función Objetivo

A la luz de esta formulación, el recorrido del equipo t entre las casillas s y $s+1$ puede redefinirse, a partir de la Fórmula 2.1., de la forma siguiente:

$$l(t,s) = \begin{cases} d(t,t) = 0 & \text{si} \quad v_{t\,s}, v_{t\,s+1} \notin V' \\ d(\tau(t,s),\tau(t,s+1)) & \text{si} \quad v_{t\,s}, v_{t\,s+1} \in V' \\ d(t,\tau(t,s+1)) & \text{si} \quad v_{t\,s} \notin V' \ \text{ y } \ v_{t\,s+1} \in V' \\ d(\tau(t,s),t) & \text{si} \quad v_{t\,s} \in V' \ \text{ y } \ v_{t\,s+1} \notin V' \end{cases} \tag{3.4}$$

De forma análoga, los trayectos inicial y final, reflejados en las Fórmulas 2.2. y 2.3., se transforman en:

$$l(t,0) = d(t,\tau(t,1)), \ \text{si} \ \ v_{t\,0} \notin V' \ \text{ y } \ v_{t\,1} \in V' \tag{3.5}$$

$$l(t,4n-2) = d(\tau(t,s),t), \ \text{si} \ \ v_{t\,4n-2} \in V' \ \text{ y } \ v_{t\,4n-1} \notin V' \tag{3.6}$$

3.3. Las Restricciones

El cálculo del recorrido total de los equipos sigue regido por (2.4), $d = \sum_{t=1}^{2n} \sum_{s=0}^{4n-2} l(t,s)$, donde los valores de los recorridos, $l(t,s)$, están restringidos según la pertenencia o no de los vértices $v_{t\,s}$ al conjunto $V' \subset V$.
Las condiciones contenidas en (2.5) conducen a la introducción, para cada V' dado, de los siguientes conjuntos.

I. La bipartición:

$$\delta(V') = \{\{v_i, v_j\} : \ \ v_i, v_j \in V, \ \ v_i \notin V' \ni v_j\} \tag{3.7}$$

II. El conjunto de corte de aristas:

$$\begin{aligned} E_{cut} &= E_{cut\,v} \cup E_{cut\,h}, \quad \text{donde,} \\ E_{cut\,v} &= \{\{v_{t\ s}, v_{\tau(t,s)\ s)}\} : (t,s) \in T \times S\} \\ E_{cut\,h} &= \{\{v_{t\ s}, v_{t\ s'}\} : t \in T, \ s, s' \in S, \ \tau(t,s) = \tau(t,s'), \ s \neq s'\} \end{aligned} \tag{3.8}$$

Por un lado, $\delta(V')$, separa los vértices asociados a las entradas de la matriz asignación que asumen el valor A, de aquellas que tendrán el valor H. Por el otro, $E_{cut\,v}$, es el conjunto de pares de vértices asociados a los pares de equipos oponentes en una casilla s, mientras que $E_{cut\,h}$, es el conjunto de pares de vértices asociados a cada oponente para cada equipo t.
Con estos conjuntos, las restricciones en la formulación combinatoria del problema de asignación local-visitante, se resumen en:

$$E_{cut} \subseteq \delta(V') \tag{3.9}$$

es decir, para una solución factible, $V' \cup (V \setminus V')$, el corte de aristas $\delta(V')$ se realiza sobre los pares de vértices de E_{cut}.
La Figura 3.2. muestra el conjunto $\delta(V')$, correspondiente a la solución factible propuesta en el Cuadro 1.4., como grafo bipartito, con corte sobre los vértices por casillas, $E_{cut\,v}$, a la izquierda en la figura, y con corte sobre los vértices por equipos, $E_{cut\,h}$, a la derecha.

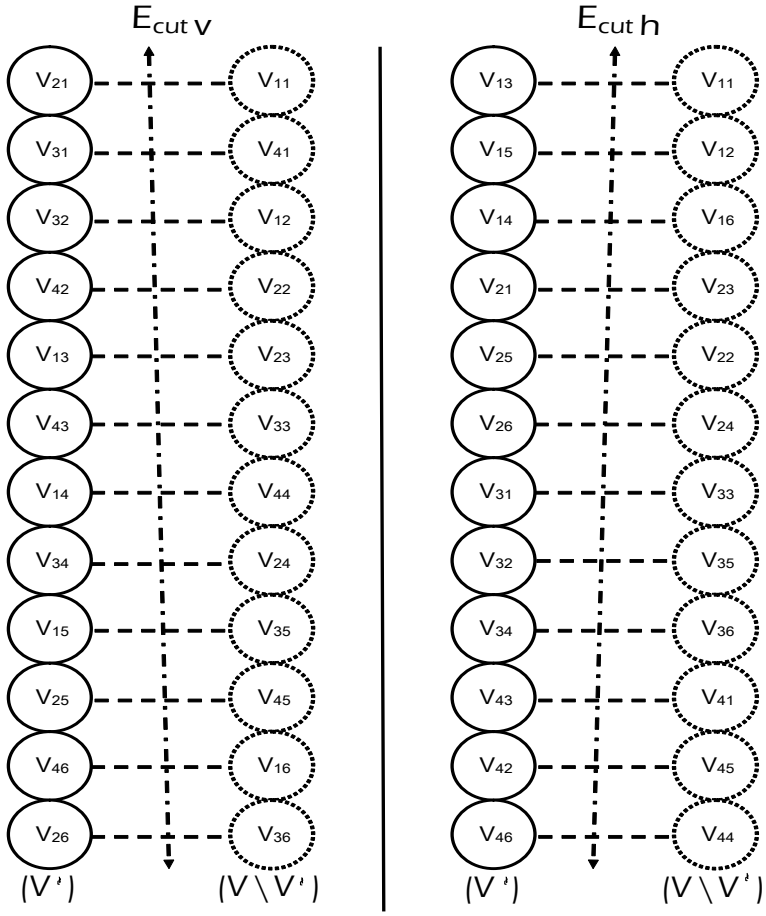

Figura 3.2: Cortes por Casillas y Equipos para $n = 2$.

Ahora bien, el algoritmo que resuelve por enumeración este problema, se simplifica enormemente al considerar que la expresión (3.9) permite formar grupos de cuatro vértices: dos de V' y dos de $V \setminus V'$. Por tanto, para evaluar la distancia (2.4), es suficiente considerar un vértice *representante* de cada grupo, para formar el *grupo representante* de los vértices. Como en el problema original, el número de posibilidades es $2^{n(2n-1)}$.

Por ejemplo, a continuación se muestran los grupos de cuatro vértices que pueden formarse de manera que se satisfagan las restricciones resumidas en (3.9), con los conjuntos $\delta(V')$ y E_{cut} definidos respectivamente mediante (3.7) y (3.8), ellos para el caso particular del itinerario prefijado para $n = 2$ de la Tabla 1.2.

$$\{v_{11} \quad v_{13} \quad v_{21} \quad v_{23}\}$$
$$\{v_{12} \quad v_{15} \quad v_{32} \quad v_{35}\}$$
$$\{v_{14} \quad v_{16} \quad v_{44} \quad v_{46}\}$$
$$\{v_{22} \quad v_{25} \quad v_{42} \quad v_{45}\}$$
$$\{v_{24} \quad v_{26} \quad v_{34} \quad v_{36}\}$$
$$\{v_{31} \quad v_{33} \quad v_{41} \quad v_{43}\}$$

Las restricciones aplicadas a estos grupos se pueden visualizar con el siguiente esquema, significando que las parejas de vértices son, o bien de V', o bien, de su complemento:

$$\left.\begin{array}{l}\{\{v_{11}, v_{23}\}, \{v_{21}, v_{13}\}\} \\ \{\{v_{12}, v_{35}\}, \{v_{15}, v_{32}\}\} \\ \{\{v_{14}, v_{46}\}, \{v_{16}, v_{44}\}\} \\ \{\{v_{22}, v_{45}\}, \{v_{25}, v_{42}\}\} \\ \{\{v_{24}, v_{36}\}, \{v_{26}, v_{34}\}\} \\ \{\{v_{31}, v_{43}\}, \{v_{33}, v_{41}\}\}\end{array}\right\} \subset \{V',\ V \setminus V'\}$$

Eligiendo un vértice de cada grupo, se puede constituir un arreglo del *grupo representante*. Éste podría ser:

$$(v_{11} \quad v_{12} \quad v_{14} \quad v_{22} \quad v_{24} \quad v_{31})$$

Para facilitar la notación, se define la variable $x_{t\,s}$,

$$x_{t\,s} = \begin{cases} 1 & \text{si } v_{t\,s} \in V' \\ 0 & \text{si } v_{t\,s} \in V \setminus V' \end{cases}$$

Entonces, al resolver por enumeración, hay que evaluar todos los vectores,

$$X \quad = \quad (x_{t\,s}) \in B^{n(2n-1)}$$

Continuando con el ejemplo para $n = 2$, se evaluarían,

$$X \quad = \quad (x_{11} \quad x_{12} \quad x_{14} \quad x_{22} \quad x_{24} \quad x_{31})$$

lo que implica un total de $2^{2(2(2)-1)} = 64$ iteraciones.
A la solución factible, ejemplificada en el Cuadro 1.4., le corresponde el arreglo,

$$X \quad = \quad (0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1)$$

cuya evaluación, en el marco de la formulación combinatoria, consiste en calcular la distancia total de recorrido de los equipos, mediante la Ecuación 2.4. para $n = 2$, siendo los términos, para esta solución factible, los que aparecen en el Cuadro 2.1. Obviamente, esta representación es la misma mostrada en el capítulo anterior para diseñar el código que permite calcular el recorrido total de los equipos mediante la enumeración de todas las soluciones factibles.
Para poder evolucionar en la búsqueda de un algoritmo más eficiente, será necesario modificar la Fórmula 2.4., de la distancia total de recorrido, a una más manejable. Esto se logra mediante la técnica de optimización conocida como *corte mínimo con restricciones* (Min Res Cut).

Capítulo 4

Formulación Min Res Cut

4.1. El Problema Min Res Cut

Siguiendo la exposición de Suzuka, (y otros)[11], el problema Min Res Cut puede plantearse en los siguientes términos:
Se considera, en primer lugar, un grafo no dirigido, $G = (V, E)$, con un conjunto de vértices V y un conjunto de aristas E. Para este grafo se define, para cualquier subconjunto, $V' \subseteq V$, el conjunto de corte de pares de vértices, $\delta(V') = \{\{v_i, v_j\} : v_i, v_j \in V, v_i \notin V' \ni v_j\}$.
A partir de este grafo, el problema Min Res Cut se define como sigue:

- Sea el conjunto de vértices, $V \cup \{r\}$, donde $r \notin V'$, es un vértice específico que se introduce como vértice auxiliar.
- Sea el conjunto de aristas, $\widehat{E} = E \cup \{\{v_{t\,s}, r\} : (t, s) \in T \times S\}$.
- Se da una función de peso sobre las aristas de $\widehat{E}$, de la forma, $\omega : \widehat{E} \to \mathbb{R}$
- Se da un conjunto, $E_{cut} \subseteq \{\mathcal{X} \subseteq V : |\mathcal{X}| = 2\}$

El problema consiste en encontrar un subconjunto de vértices V' que minimice la suma,

$$\sum_{e \in (\delta(V') \cap \widehat{E})} \omega(e) \tag{4.1}$$

sujeta a las condiciones,

$$\begin{aligned} r &\notin V' \\ E_{cut} &\subseteq \delta(V') \end{aligned}$$

4.2. Formulación Min Res Cut del Problema Asignación HA

4.2.1. El Grafo Min Res Cut

La formulación del problema Asignación HA como un problema Min Res Cut es la que se presenta a continuación.
Dada una Matriz Itinerario, $\mathcal{T}=(\tau(t,s))$, $(t,s) \in T \times S$, sea $G = (V, E)$ el grafo no dirigido correspondiente a la formulación combinatoria, propuesto mediante (3.1) y (3.2) en la Sección 3.1. Se introduce la variable artificial r, con la cual los conjuntos de vértices y aristas, para la formulación Min Res Cut del problema Asignación HA, serán:

$$V = \{v_{t\,s} : (t,s) \in T \times S\} \cup \{r\} \tag{4.2}$$

$$E = \{\{v_{t\,s}, v_{t\,s+1}\} : t \in T,\ s \in S \setminus \{4n-2\}\} \cup \{\{r, v_{t\,s}\} : (t,s) \in T \times S\} \tag{4.3}$$

Entonces, esta formulación coincide, salvo por los conjuntos V y E dados en 4.2 y 4.3, con la formulación combinatoria dada en la Sección 3.1. Significa esto que, el conjunto $\delta(V')$ se define igual, en función de los vértices $v_{t\,s}$. Igualmente, en función de ellos, se define el conjunto de soluciones factibles como en (3.3), el recorrido de los equipos como en (3.4), (3.5) y (3.6) y el conjunto de corte de aristas como en (3.8). Finalmente, las restricciones siguen resumiéndose con la condición, $E_{cut} \subseteq \delta(V')$.
Como ejemplo de grafo para la formulación Min Res Cut del problema de asignación local-visitante, la Figura 4.1. corresponde a la situación que se presenta para $n = 2$:

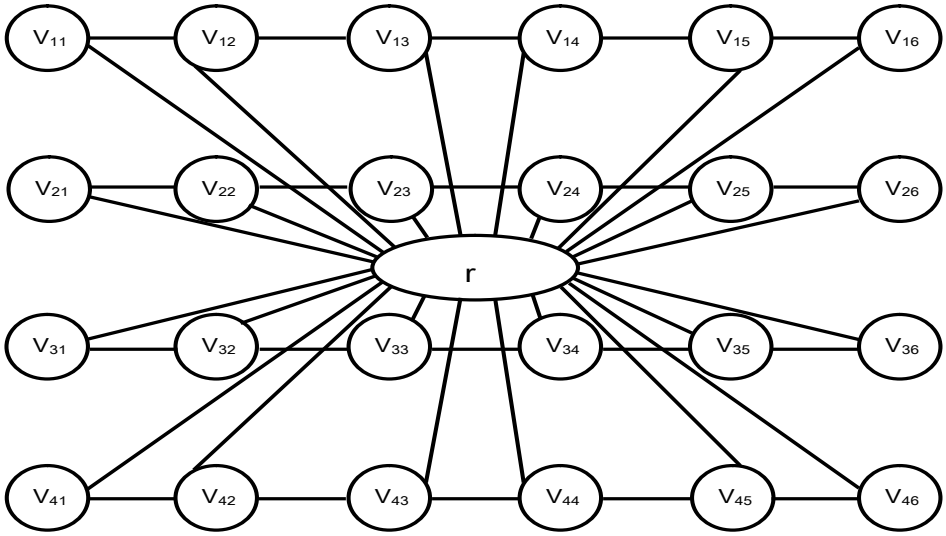

Figura 4.1: Grafo Min Res Cut para $n = 2$.

4.2.2. Las Restricciones Min Res Cut

Como se planteó en la Sección 2.2, las restricciones impuestas mediante las condiciones (2.5) en el problema original, y, consecuentemente, mediante la Ecuación 3.9., en la Sección 3.1., al estudiar la formulación combinatoria, restricciones que aplican también para la formulación Min Res Cut, permiten formar grupos de cuatro vértices, siendo entonces suficiente elegir un *representante* de cada grupo y constituir con ellos un *grupo representante* que, al ser evaluado por enumeración para un n dado, se ponen en juego $2^{n(2n-1)}$ posibilidades. Como el vértice específico r no interviene en el conjunto E_{cut}, las restricciones por corte de aristas son aquí exactamente las mismas que las

estudiadas en la formulación combinatoria.
Sin embargo, en la formulación Min Res Cut, $r \notin V'$, constituye otra restricción. Podría dibujarse un esquema similar al de la Figura 3.2., para el caso $n = 2$, con el fin de reflejar el efecto del corte de pares de vértices de $\delta(V')$, sin más que agregar el vértice r a la parte $V \setminus V'$ de la bipartición. De otra manera, la Figura 4.2. muestra el efecto directamente sobre el grafo $G = (V, E)$.

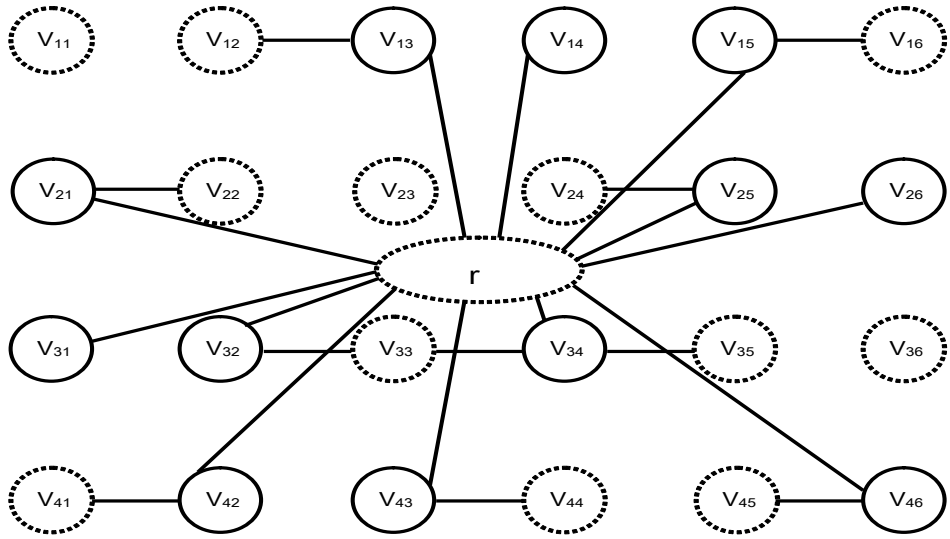

Figura 4.2: Restricciones E_{cut} y $r \notin V'$ para $n = 2$.

4.2.3. El Recorrido en la Formulación Min Res Cut

Como se indicó al final de la Sección 3.1., la técnica de optimización conocida como *corte mínimo con restricciones*, permitirá modificar la Fórmula 2.4., de la distancia total de recorrido, a una más manejable, con la idea de poder evolucionar en la búsqueda de un algoritmo más eficiente.
En este marco, se procede a modificar la expresión (3.4) para los recorridos de los equipos. Para ello, se utiliza una aritmética basada en la cardinalidad del conjunto, $v_{t\,s} \cap V'$. Está claro que si, $v_{t\,s} \in V'$, entonces, $|v_{t\,s} \cap V'| = 1$. Caso contrario, si $v_{t\,s} \notin V'$, entonces, $|v_{t\,s} \cap V'| = 0$.
Este sencillo resultado, permite expresar la Ecuación 3.4. para el recorrido del equipo t, desde la casilla s hasta la $s + 1$, como sigue. Sean, $t' = \tau(t, s)$ y $t'' = \tau(t, s + 1)$, entonces,

$$\begin{aligned} l(t,s) = \quad & d(t,t) \quad [1 - |v_{t\,s} \cap V'|] \;\; [1 - |v_{t\,s+1} \cap V'|] \;\; + \\ & d(t',t'') \quad |v_{t\,s} \cap V'| \;\; |v_{t\,s+1} \cap V'| \;\; + \\ & d(t,t'') \quad [1 - |v_{t\,s} \cap V'|] \;\; |v_{t\,s+1} \cap V'| \;\; + \\ & d(t',t) \quad |v_{t\,s} \cap V'| \;\; [1 - |v_{t\,s+1} \cap V'|] \end{aligned} \tag{4.4}$$

Es posible modificar esta fórmula tomando en cuenta varios aspectos. En primer lugar, es obvio que $d(t, t) = 0$, de manera que se puede prescindir del primer término. Además, como $r \notin V'$, se tiene la identidad, $|v_{t\,s} \cap V'| = |\{r, v_{t\,s}\} \cap \delta(V')|$. Introduciendo estos resultados, la Fórmula 4.4. queda de la siguiente forma, en término de las aristas $\{r, v_{t\,s}\}$ del conjunto de aristas, E, de la formulación Min Res Cut:

$$\begin{aligned} l(t,s) = \quad & d(t',t'') \quad |\{r, v_{t\,s}\} \cap \delta(V')| \;\; |\{r, v_{t\,s+1}\} \cap \delta(V')| \;\; + \\ & d(t,t'') \quad [1 - |\{r, v_{t\,s}\} \cap \delta(V')|] \;\; |\{r, v_{t\,s+1}\} \cap \delta(V')| \;\; + \\ & d(t',t) \quad |\{r, v_{t\,s}\} \cap \delta(V')| \;\; [1 - |\{r, v_{t\,s+1}\} \cap \delta(V')|] \end{aligned} \tag{4.5}$$

En segundo lugar, combinando las notaciones utilizadas en (2.1) y (3.4), con los cardinales aquí introducidos, se tiene la siguiente tabla de valores, sin considerar el caso $(a_{t\,s}, a_{t\,s+1}) = (H, H)$, que, ya se ha visto, es prescindible:

$(a_{t\,s}, a_{t\,s+1})$	$\lvert\{r, v_{t\,s}\} \cap \delta(V'\rvert$	$\lvert\{r, v_{t\,s+1}\} \cap \delta\,(V')\rvert$	$\lvert\{v_{t\,s}, v_{t\,s+1}\} \cap \delta\,(V')\rvert$
(A, A)	1	1	0
(A, H)	1	0	1
(H, A)	0	1	1

Para las tres situaciones descritas en esta tabla, los resultados allí escritos permiten establecer las siguientes relaciones,

I. $\quad \lvert\{r, v_{t\,s}\} \cap \delta(V')\rvert \;\; \lvert\{r, v_{t\,s+1}\} \cap \delta(V')\rvert = [1 - \lvert\{v_{t\,s}, v_{t\,s+1}\} \cap \delta(V')\rvert]$

II. $\quad \frac{1}{2}\,[\lvert\{r, v_{t\,s}\} \cap \delta(V')\rvert + \lvert\{r, v_{t\,s+1}\} \cap \delta(V')\rvert + \lvert\{v_{t\,s}, v_{t\,s+1}\} \cap \delta(V')\rvert] = 1$

Es así como se pueden desarrollar los cardinales de la Fórmula 4.5. hasta alcanzar los siguientes resultados,

$$\lvert\{r, v_{t\,s}\} \cap \delta\left(V'\right)\rvert \,\lvert\{r, v_{t\,s+1}\} \cap \delta\left(V'\right)\rvert = \frac{1}{2}[\lvert\{r, v_{t\,s}\} \cap \delta(V')\rvert + \lvert\{r, v_{t\,s+1}\} \cap \delta(V')\rvert - \lvert\{v_{t\,s}, v_{t\,s+1}\} \cap \delta(V')\rvert]$$

$$\left[1 - \lvert\{r, v_{t\,s}\} \cap \delta\left(V'\right)\rvert\right] \;\lvert\{r, v_{t\,s+1}\} \cap \delta\left(V'\right)\rvert = \frac{1}{2}[-\lvert\{r, v_{t\,s}\} \cap \delta(V')\rvert + \lvert\{r, v_{t\,s}\} \cap \delta(V')\rvert + \lvert\{v_{t\,s}, v_{t\,s+1}\} \cap \delta(V')\rvert]$$

$$\lvert\{r, v_{t\,s}\} \cap \delta\left(V'\right)\rvert\,[1 - \lvert\{r, v_{t\,s+1}\} \cap \delta\left(V'\right)\rvert] = \frac{1}{2}[\lvert\{r, v_{t\,s}\} \cap \delta(V')\rvert - \lvert\{r, v_{t\,s}\} \cap \delta(V')\rvert + \lvert\{v_{t\,s}, v_{t\,s+1}\} \cap \delta(V')\rvert]$$

Al sustituir en (4.5) se puede desarrollar luego para que los recorridos $l(t, s)$ resulten en términos de los cardinales de las aristas de E:

$$\begin{array}{rlll} l(t,s) = & \frac{1}{2}\lvert\{r, v_{t\,s}\} \cap \delta(V')\rvert & [d(t', t'') - d(t, t'') + d(t', t)] & + \\ & \frac{1}{2}\lvert\{r, v_{t\,s+1}\} \cap \delta(V')\rvert & [d(t', t'') + d(t, t'') - d(t', t)] & + \\ & \frac{1}{2}\lvert\{v_{t\,s}, v_{t\,s+1}\} \cap \delta(V')\rvert & [-d(t', t'') + d(t, t'') + d(t', t)] & \end{array} \tag{4.6}$$

Falta sólo obtener los recorridos inicial y final para cada equipo. Para ello se puede recurrir a la forma (4.4) de $l(t, s)$. Para obtener $l(t, 0)$ ha de tomarse en cuenta el resultado en (3.5), mientras que para $l(t, 4n-2)$, el resultado (3.6). Desarrollando se obtienen:

$$l(t, 0) \;=\; d(t, \tau(t, 1)) \;\; \lvert\{r, v_{t\,1}\} \cap \delta(V')\rvert \tag{4.7}$$

$$l(t, 4n-2) \;=\; d(\tau(t, 4n-2), t) \;\; \lvert\{r, v_{t\,4n-2}\} \cap \delta(V')\rvert \tag{4.8}$$

El terreno está ya abonado para obtener la fórmula del recorrido total de los equipos en la formulación Min Res Cut. Para ello se sustituye en la Fórmula (2.4) los resultados recogidos en (4.6), (4.7) y (4.8).

$$\begin{array}{rll} d \;=\; \sum_{t\in T} & \langle\, d(t, \tau(t, 1)) \;\lvert\{r, v_{t\,1}\} \cap \delta(V')\rvert \;+ \\ & \sum_{s=1}^{4n-3} \Big\{ \frac{1}{2}\,\lvert\{r, v_{t\,s}\} \cap \delta(V')\rvert \;[d(t', t'') - d(t, t'') + d(t', t)] \;+ \\ & \qquad \frac{1}{2}\,\lvert\{r, v_{t\,s+1}\} \cap \delta(V')\rvert \;[d(t', t'') + d(t, t'') - d(t', t)] \;+ \\ & \qquad \frac{1}{2}\,\lvert\{v_{t\,s}, v_{t\,s+1}\} \cap \delta(V')\rvert \;[-d(t', t'') + d(t, t'') + d(t', t)] \Big\} \;+ \\ & d(\tau(t, 4n-2), t) \;\lvert\{r, v_{t\,s}\} \cap \delta(V')\rvert\,\rangle \end{array} \tag{4.9}$$

El siguiente paso consiste en encontrar una expresión como la (4.1), en término de los pesos de las aristas de E. Para ello se desarrolla la Fórmula 4.9. sobre el índice s, tomando en cuenta que para

$s = 1$ y $s = 4n - 3$ hay que incorporar los resultados (4.7) y (4.8) para los recorridos inicial y final respectivamente. Para las aristas intermedias, se hace $s = \sigma$, con $t' = \tau(t, \sigma)$ y $t'' = \tau(t, \sigma + 1)$ y, para la siguiente casilla, $s = \sigma + 1$, con $t' = \tau(t, \sigma + 1)$ y $t'' = \tau(t, \sigma + 2)$
El resultado último es el siguiente:

$$\begin{aligned} d = \ \tfrac{1}{2} \textstyle\sum_{t \in T} \ & |\{r, v_{t\,1}\} \cap \delta(V')| \, [2d(t, \tau(t, 1)) + \\ & d(\tau(t, 1), \tau(t, 2)) - d(t, \tau(t, 2)) + d(\tau(t, 1), t)] + \\ & \textstyle\sum_{s=2}^{4n-3} |\{r, v_{t\,s}\} \cap \delta(V')| \\ & \qquad \{[d(\tau(t, s), \tau(t, s + 1)) - d(t, \tau(t, s + 1)) + d(\tau(t, s), t)] + \\ & \qquad [d(\tau(t, s - 1), \tau(t, s)) + d(t, \tau(t, s)) - d(\tau(t, s - 1), t)]\} + \\ & \textstyle\sum_{s=1}^{4n-3} |\{v_{t\,s}, v_{t\,s+1}\} \cap \delta(V')| \\ & \qquad [-d(\tau(t, s), \tau(t, s + 1)) + d(t, \tau(t, s + 1)) + d(\tau(t, s), t)] \ + \\ & \ |\{r, v_{t\,4n-2}\} \cap \delta(V')| \, [2d(t, \tau(t, 4n - 2)) + \\ & \ d(\tau(t, 4n - 3), \tau(t, 4n - 2)) + d(t, \tau(t, 4n - 2)) - d(\tau(t, 4n - 3), t)] \end{aligned} \tag{4.10}$$

A partir de la Fórmula 4.10. se definen los siguientes pesos de aristas de E:

$$\begin{aligned} \omega(\{r, v_{t\,1}\}) &= \tfrac{1}{2} \{2d(t, \tau(t, 1)) + d(\tau(t, 1), \tau(t, 2)) - d(t, \tau(t, 2)) + d(\tau(t, 1), t)\} \\ \omega(\{r, v_{t\,4n-2}\}) &= \tfrac{1}{2} \{2d(\tau(t, 4n - 2), t) + d(\tau(t, 4n - 3), \tau(t, 4n - 2)) + \\ &\quad + d(t, \tau(t, 4n - 2)) - d(\tau(t, 4n - 3), t)\} \\ \omega(\{r, v_{t\,s}\}) &= \tfrac{1}{2} \{d(\tau(t, s), \tau(t, s + 1)) - d(t, \tau(t, s + 1)) + d(\tau(t, s), t)\} + \\ &\quad + \tfrac{1}{2} \{d(\tau(t, s - 1), \tau(t, s)) + d(t, \tau(t, s)) - d(\tau(t, s - 1), t)\} \\ \omega(\{v_{t\,s}, v_{t\,s+1}\}) &= \tfrac{1}{2} \{-d(\tau(t, s), \tau(t, s + 1)) + d(t, \tau(t, s + 1)) + d(\tau(t, s), t)\} \end{aligned} \tag{4.11}$$

Con lo cual (4.10) puede escribirse, en términos de los pesos (4.11), de la forma:

$$\begin{aligned} d \ = \ & \textstyle\sum_{t \in T} \{ \, |\{r, v_{t\,1}\} \cap \delta(V')| \, \omega(\{r, v_{t\,1}\}) + \\ & + \textstyle\sum_{s=2}^{4n-3} |\{r, v_{t\,s}\} \cap \delta(V')| \, \omega(\{r, v_{t\,s}\}) + \\ & + \textstyle\sum_{s=1}^{4n-3} |\{v_{t\,s}, v_{t\,s+1}\} \cap \delta(V')| \, \omega(\{v_{t\,s}, v_{t\,s+1}\}) + \\ & + \qquad |\{r, v_{t\,4n-2}\} \cap \delta(V')| \, \omega(\{r, v_{t\,4n-2}\}) \ \} \end{aligned} \tag{4.12}$$

Puede verse en (4.12) que la distancia total de recorrido quedó expresada en términos de las aristas de E: $\{r, v_{t\,1}\}$, $\{r, v_{t\,4n-2}\}$, $\{r, v_{t\,s}\}$ y $\{v_{t\,s}, v_{t\,s+1}\}$. Designándolas de modo genérico como e, la Fórmula 4.12. se reduce a:

$$d \ = \sum_{e \in E} |e \cap \delta(V')| \ \omega(e) \tag{4.13}$$

que no es más que la forma Min Res Cut de la función objetivo tal como se presenta en la éxpresión (4.1) de la Sección 4.1, ya que (4.13) puede escribirse idénticamente como:

$$d \ = \sum_{e \in E \cap \delta(V')} \omega(e) \tag{4.14}$$

4.3. Resultado de la Formulación

El problema de Asignación HA queda planteado en la formulación Min Res Cut en los siguientes términos. Dados los conjuntos de vértices (4.2) y de aristas (4.3), se plantea el problema:

$$\begin{array}{lll} & \min & d = \sum\limits_{e \in E \cap \delta(V')} \omega(e) \\ \text{(MRCHA)} & \text{s.a.} & r \notin V' \\ & & E_{cut} \subseteq \delta(V') \end{array} \tag{4.15}$$

donde los conjuntos de pares de vértices $\delta(V')$ y E_{cut} están definidos en (3.7) y (3.8) respectivamente y los pesos $\omega(e)$, por el grupo de fórmulas (4.11).

Capítulo 5

Formulación Entera Binaria

5.1. Formulación Entera Binaria del Problema Asignación HA

La formulación matemática del problema de asignación local visitante, pasa por asociar una variable entera binaria a cada vértice de V. Estas variables son de la forma, X_r, $X_{t,s}$ y $X_{t,s+1}$. En términos de éstas, se proponen funciones elementales g_e asociadas a cada arista e de E, de forma tal que en la función objetivo (4.14), para un subconjunto dado V' de V, sólo apliquen las aristas de $E \cap \delta(V')$ que satisfagan las restricciones, $r \notin V'$ y $E_{cut} \subset \delta(V')$.
Se define de forma genérica la siguiente función de las variables:

$$g_e = \begin{cases} 1 & \text{si } e \in E \cap \delta(V') \\ 0 & \text{si } e \notin E \cap \delta(V') \end{cases} \tag{5.1}$$

con lo cual, en la reformulación matemática del problema, la función objetivo quedaría como sigue:

$$d = \sum_{e \in (E \cap \delta(V'))} g_e \, \omega(e)$$

Retomando las aristas de E que aparecen en la Fórmula 4.12., $\{r, v_{t\,1}\}$, $\{r, v_{t\,4n-2}\}$, $\{r, v_{t\,s}\}$ y $\{v_{t\,s}, v_{t\,s+1}\}$, se tiene las siguientes funciones:

$$g_1(X_r, X_{t,s}) = \begin{cases} 1 & \text{si } \{r, v_{t\,s}\} \in E \cap \delta(V') \\ 0 & \text{si } \{r, v_{t\,s}\} \notin E \cap \delta(V') \end{cases} \tag{5.2}$$

$$g_2(X_{t,s}, X_{t,s+1}) = \begin{cases} 1 & \text{si } \{v_{t\,s}, v_{t\,s+1}\} \in E \cap \delta(V') \\ 0 & \text{si } \{v_{t\,s}, v_{t\,s+1}\} \notin E \cap \delta(V') \end{cases} \tag{5.3}$$

y la función objetivo queda:

$$d = \sum_{t \in T} \sum_{s \in S} g_1(X_r, X_{t,s}) \omega(\{r, v_{t\,s}\}) + \sum_{t \in T} \sum_{s \in S \setminus \{4n-2\}} g_2(X_{t,s}, X_{t,s+1}) \omega(\{v_{t\,s}, v_{t\,s+1}\}) \tag{5.4}$$

5.2. Variables Enteras Binarias en {0,1}

Se define la siguiente variable entera binaria, asociada a cada vértice de V:

$$X_{t,s} = \begin{cases} 1 & \text{si } v_{ts} \in V' \\ 0 & \text{si } v_{ts} \notin V' \end{cases} \tag{5.5}$$

5.2.1. La Función Objetivo

Sabiendo que $r \notin V'$, expresiones adecuadas para las funciones (5.2) y (5.3) son las siguientes:

$$\begin{aligned} g_1(X_r, X_{t,s}) &= (X_r - X_{t,s})^2 \\ g_2(X_{t,s}, X_{t,s+1}) &= (X_{t,s} - X_{t,s+1})^2 \end{aligned} \tag{5.6}$$

con lo cual, en la reformulación matemática del problema, la función objetivo quedaría como sigue:

$$d = \sum_{t\in T}\sum_{s\in S}(X_r - X_{t,s})^2\omega(\{r, v_{t\,s}\}) + \sum_{t\in T}\sum_{s\in S\setminus\{4n-2\}}(X_{t,s} - X_{t,s+1})^2\omega(\{v_{t\,s}, v_{t\,s+1}\}) \tag{5.7}$$

5.2.2. Las Restricciones

En la Sección 3.1, al estudiar la formulación combinatoria del problema de asignación local-visitante, se introdujeron los conjuntos,

$$\begin{aligned} E_{cut\,v} &= \{\{v_{t\,s}, v_{\tau(t,s)\,s)}\} : (t,s) \in T \times S\} \\ E_{cut\,h} &= \{\{v_{t\,s}, v_{t\,s'}\} : t \in T,\ s, s' \in S,\ \tau(t,s) = \tau(t,s'),\ s \neq s'\} \end{aligned}$$

los cuales permiten establecer las restricciones para este problema, las cuales se resumen con la relación,

$$E_{cut} \subseteq \delta(V')$$

donde, $E_{cut} = E_{cut\,v} \cup E_{cut\,h}$.
En términos de la variable entera binaria, $X_{t\,s.}$, tales restricciones quedarían como sigue:

I. Por efecto de $E_{cut\,v}$,: $\forall(t,s) \in T \times S$,

$$X_{t\,s} + X_{\tau(t,s)\,s} = 1 \tag{5.8}$$

II. Por efecto de $E_{cut\,h}$,: $\forall(t,s),(t,s') \in T \times S,\ \tau(t,s) = \tau(t,s'),\ s \neq s'$,

$$X_{t\,s} + X_{t\,s'} = 1 \tag{5.9}$$

las cuales hacen un total de $4n(2n-1)$ ecuaciones.
En vista de que, $E_{cut\,v}$ y $E_{cut\,h}$, agrupan en parejas a todos los vértices de V, y éstos a su vez forman grupos de cuatro vértices relacionados, como se vio en la Sección 1.1., es posible replantear el problema de manera que se reduzca el número de ecuaciones en una cuarta parte. Una formulación equivalente puede hacerse, por ejemplo, manteniendo las $2n(2n-1)$ ecuaciones (5.8) y sustituyendo las ecuaciones (5.9) como sigue:

$$\begin{aligned} X_{t\,s} &+ X_{\tau(t,s)\,s} &&= 1 \\ X_{t\,s} &- X_{\tau(t,s')\,s'} &&= 0 \end{aligned}$$

para todo, $(t,s),(t,s') \in T \times S,\ \tau(t,s) = \tau(t,s'),\ s < s'$
Por último, la otra restricción, $r \notin V'$, se reduce a la expresión:

$$X_r = 0$$

Al sustituir esta restricción en (5.7), la misma puede reescribirse como sigue:

$$d \ = \ \sum_{t\in T}\sum_{s\in S}(X_{t\,s})^2\,\omega(\{r,v_{t,s}\}) \quad + \sum_{t\in T}\sum_{s\in S\setminus\{4n-2\}}(X_{t\,s}-X_{t\,s+1})^2\,\omega(\{v_{t,s},v_{t,s+1}\}) \tag{5.10}$$

Desarrollando se obtiene la ecuación cuadrática:

$$d \ = \ X^{'}\,Q\,X \tag{5.11}$$

donde Q es una matriz simétrica tridiagonal de orden, $[2n(4n-2)]$ y $X = (X_{t\,s})$, $t \in T$ y $s \in S$, es un vector de dimensión, $2n(4n-2)$.
En detalle, la matriz Q es de la forma:

$$Q = \begin{pmatrix} Q_1 & 0 & \dots & 0 & 0 \\ 0 & Q_2 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & Q_{2n-1} & 0 \\ 0 & 0 & \dots & 0 & Q_{2n} \end{pmatrix} \tag{5.12}$$

donde cada submatriz, Q_t, de orden $(4n-2)$, contiene los pesos, $\omega(\{r,v_{t\,s}\})$ y $\omega(\{v_{t\,s},v_{t\,s+1}\})$.
Para simplificar la notación, sean,

$$\begin{aligned} \omega_{t\,s} &= \omega(\{r,v_{t\,s}\}) \\ \omega_{t\,s\,s+1} &= \omega(\{v_{t\,s},v_{t\,s+1}\}) \\ \omega_{t\,s\,s+1\,s+2} &= \omega_{t\,s}+\omega_{t\,s\,s+1}+\omega_{t\,s+1\,s+2} \end{aligned}$$

siendo, particularmente,

$$\begin{aligned} \omega_{t\,0\,1\,2} &= \omega_{t\,1}+0+\omega_{t\,1\,2} \\ \omega_{t\,4n-3\,4n-2\,4n-1} &= \omega_{t\,4n-2}+\omega_{t\,4n-3\,4n-2}+0 \end{aligned}$$

De esta forma, las submatrices quedan de la forma:

$$Q_t = \begin{pmatrix} \omega_{t\,0\,1\,2} & -\omega_{t\,1\,2} & 0 & \dots & 0 & 0 & 0 \\ -\omega_{t\,1\,2} & \omega_{t\,1\,2\,3} & -\omega_{t\,2\,3} & \dots & 0 & 0 & 0 \\ 0 & -\omega_{t\,2\,3} & \omega_{t\,2\,3\,4} & \dots & 0 & 0 & 0 \\ 0 & 0 & -\omega_{t\,3\,4} & \ddots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & \omega_{t\,4n-5\,4n-4} & 0 & 0 \\ 0 & 0 & 0 & \dots & \omega_{t\,4n-5\,4n-4\,4n-3} & -\omega_{t\,4n-4\,4n-3} & 0 \\ 0 & 0 & 0 & \dots & -\omega_{t\,4n-4\,4n-3} & \omega_{t\,4n-4\,4n-3\,4n-2} & -\omega_{t\,4n-3\,4n-2} \\ 0 & 0 & 0 & \dots & 0 & -\omega_{t\,4n-3\,4n-2} & \omega_{t\,4n-3\,4n-2\,4n-1} \end{pmatrix} \tag{5.13}$$

5.2.3. Resultado de la Formulación

El problema de Asignación HA queda planteado en la formulación entera cuadrática binaria en los siguientes términos. Dados los conjuntos de vértices (4.2) y de aristas (4.3) y dada la siguiente variable entera binaria, asociada a cada vértice de V:

$$X_{t,s} = \begin{cases} 1 & \text{si } v_{ts} \in V' \\ 0 & \text{si } v_{ts} \notin V' \end{cases} \tag{5.14}$$

se plantea el problema:

$$\begin{array}{lll} & \min & d \;=\; X^{'}\, Q\, X \\ \text{(PEB01HA)} & \text{s.a.} & X_{t\,s} + X_{\tau(t,s)\,s} = 1 \\ & & X_{t\,s} - X_{\tau(t,s')\,s'} = 0 \\ & & \forall (t,s), (t,s') \in T \times S,\ \tau(t,s) = \tau(t,s'),\ s < s' \\ & & X_{t\,s} \in \{0,1\} \end{array} \tag{5.15}$$

donde la matriz Q está descrita mediante (5.12) y (5.13) y los pesos por el grupo de fórmulas 4.11. La resolución de este programa se efectuará en el siguiente capítulo.

5.3. Variables Enteras Binarias en {-1,+1}

Se define la siguiente variable entera binaria, asociada a cada vértice de V:

$$X_{t,s} = \begin{cases} +1 & \text{si } v_{ts} \in V' \\ -1 & \text{si } v_{ts} \notin V' \end{cases} \tag{5.16}$$

5.3.1. La Función Objetivo

Sabiendo que $r \notin V'$, expresiones adecuadas para las funciones (5.2) y (5.3) son las siguientes:

$$\begin{aligned} g_1(X_r, X_{t,s}) &= \tfrac{1}{4}(X_r - X_{t,s})^2 \\ g_2(X_{t,s}, X_{t,s+1}) &= \tfrac{1}{4}(X_{t,s} - X_{t,s+1})^2 \end{aligned} \tag{5.17}$$

con lo cual, en la reformulación matemática del problema, la función objetivo quedaría como sigue:

$$d = \frac{1}{4}\sum_{t\in T}\sum_{s\in S}(X_r - X_{t,s})^2\omega(\{r, v_{t\,s}\}) + \frac{1}{4}\sum_{t\in T}\sum_{s\in S\setminus\{4n-2\}}(X_{t,s} - X_{t,s+1})^2\omega(\{v_{t\,s}, v_{t\,s+1}\}) \tag{5.18}$$

5.3.2. Las Restricciones

Recordemos que las primeras restricciones para este problema se resumen con la relación:

$$E_{cut} \subseteq \delta(V')$$

donde, $E_{cut} = E_{cut\,v} \cup E_{cut\,h}$.
En términos de esta variable entera binaria, $X_{t\,s.}$, tales restricciones quedarían como sigue:

I. Por efecto de $E_{cut\,v}$,: $\forall (t,s) \in T \times S$,

$$X_{t\,s} + X_{\tau(t,s)\,s} = 0 \tag{5.19}$$

II. Por efecto de $E_{cut\,h}$,: $\forall (t,s),(t,s') \in T \times S,\ \tau(t,s) = \tau(t,s'),\ s \neq s'$,

$$X_{t\,s} + X_{t\,s'} = 0 \tag{5.20}$$

las cuales hacen un total de $4n(2n-1)$ ecuaciones.
Considerando los grupos de cuatro vértices relacionados, como se vio en la Sección 1.1., replanteamos el problema de manera que se reduzca el número de ecuaciones en una cuarta parte. Una formulación equivalente puede hacerse, por ejemplo, manteniendo las $2n(2n-1)$ ecuaciones (5.19) y sustituir las ecuaciones (5.20) como sigue:

$$\begin{aligned} X_{t\,s} &+ X_{\tau(t,s)\,s} &&= 0 \\ X_{t\,s} &- X_{\tau(t,s')\,s'} &&= 0 \end{aligned}$$

para todo, $(t,s),(t,s') \in T \times S,\ \tau(t,s) = \tau(t,s'),\ s < s'$
Por último, la otra restricción, $r \notin V'$, se reduce a la expresión:

$$X_r = -1$$

Desarrollando (5.18), manteniendo explícita la variable X_r, obtenemos la ecuación cuadrática:

$$d = \frac{1}{4}X^{'}\,L\,X \tag{5.21}$$

donde L es una matriz simétrica, ahora no tridiagonal, de orden, $[2n(4n-2)+1]$ y $X = (X_{t\,s}, X_r)$, $t \in T$ y $s \in S$, es un vector de dimensión, $2n(4n-2)+1$.
En detalle, la matriz L es de la forma:

$$L = \begin{pmatrix} L_1 & 0 & \dots & 0 & 0 & b_1 \\ 0 & L_2 & \dots & 0 & 0 & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & L_{2n-1} & 0 & b_{2n-1} \\ 0 & 0 & \dots & 0 & L_{2n} & b_{2n} \\ b_1^T & b_2^T & \dots & b_{2n-1}^T & b_{2n}^T & \sum \omega_{ts} \end{pmatrix} \tag{5.22}$$

donde cada vector es de la forma, $b_t = (-\omega_{t\,s})$, $s \in S$, $t \in T$ y donde cada submatriz, L_t, de orden $(4n-2)$, contiene los pesos, $\omega(\{r, v_{t\,s}\})$ y $\omega(\{v_{t\,s}, v_{t\,s+1}\})$ de igual manera que las submatrices Q_t del caso anterior. O sea, $L_t = Q_t$, para $t = 1,\ 2,\ ...,\ 2n$.

5.3.3. Resultado de la Formulación

El problema de Asignación HA queda planteado en la formulación entera cuadrática binaria en los siguientes términos. Dados los conjuntos de vértices (4.2) y de aristas (4.3) y dada la siguiente variable entera binaria, asociada a cada vértice de V:

$$X_{t,s} = \begin{cases} +1 & \text{si } v_{ts} \in V' \\ -1 & \text{si } v_{ts} \notin V' \end{cases} \tag{5.23}$$

se plantea el problema:

$$\begin{array}{lll} & \min & d = \frac{1}{4} X' \, L \, X \\ \text{(PEB11HA)} & \text{s.a.} & X_{t\,s} + X_{\tau(t,s)\,s} = 0 \\ & & X_{t\,s} - X_{\tau(t,s')\,s'} = 0 \\ & & \forall (t,s), (t,s') \in T \times S,\ \tau(t,s) = \tau(t,s'),\ s < s' \\ & & X_{t\,s} \in \{-1, +1\} \end{array} \tag{5.24}$$

donde la matriz L está descrita mediante (5.22) y (5.13), con los vectores $b_t = (-\omega_{t\,s})$, $s \in S$, $t \in T$, admitiendo que las submatrices L_t y Q_t, $t \in T$, son idénticas y donde los pesos están dados por las fórmulas 4.11.

Capítulo 6

Resolución de la Formulación Cuadrática Entera Binaria

En la literatura se acepta como un hecho que el programa entero, particularmente el programa entero cuadrático binario, es NP-completo. Al encarar su resolución se requiere de algoritmos de aproximación como los que se presentan aquí y en los próximos capítulos. El software matemático MatLab cuenta, entre sus herramientas, con funciones que permiten el cómputo de diversos programas de optimización. No es el caso para problemas como el que nos ocupa en este trabajo, sin embargo, dispone de otras funciones útiles que pueden aplicarse a relajaciones de nuestro problema. En este sentido, en este capítulo mostramos el comportamiento de tres diferentes relajaciones que pueden ser computadas con funciones matlab disponibles, sin entrar en detalles de los algoritmos involucrados siendo que no es el objetivo de nuestro trabajo. La función fmincon.m permite resolver una programación no lineal con restricciones no lineales (PNLC) y la función quadprog.m, un programa cuadrático (QP), aunque también es útil para aplicarse a una programa no lineal con restricciones lineales (PNLL). Para no abultar el contenido de este trabajo, nos limitaremos únicamente a la resolución del problema 5.24 para el caso que las variables asuman los valores en $\{0, 1\}$.

6.1. Relajación QP

Como es sabido, la complejidad del programa cuadrático entero binario reside, precisamente, en restringir a las variables a ser enteras. La relajación programa cuadrático consiste en eliminar esta restricción. La relajación QP del programa mostrado en 5.24, consistirá sencillamente en exigir que las variables estén restringidas al conjunto $[0, 1]$ y quedaría como sigue:

$$\begin{array}{lll} & \min & d = X' Q X \\ \text{(QPHA01)} & \text{s.a.} & X_{t\,s} + X_{\tau(t,s)\,s} = 1 \\ & & X_{t\,s} - X_{\tau(t,s')\,s'} = 0 \\ & & \forall (t,s), (t,s') \in T \times S,\ \tau(t,s) = \tau(t,s'),\ s < s' \\ & & X_{t\,s} \in [0,1] \end{array} \tag{6.1}$$

Aplicamos el código quadprog.m para el cómputo de esta última relajación. El código máster diseñado para su implementación, progquadn2.m, para el caso $n = 2$, lo mostramos en el Apéndice B. Los resultados numéricos aparecen en el Capítulo 8. El código matlab quadprog.m es una función que permite encontrar el mínimo de una función restringida no lineal multivaluada. El problema que

resuelve se especifica como sigue [7]:

$$\begin{aligned} \min \quad & \tfrac{1}{2}x^T Hx + f^T x \\ \text{s.a.} \quad & A \cdot x \leq b \\ & Aeq \cdot x = bed \\ & lb \leq x \leq ub \end{aligned}$$

En nuestro problema sólo utilizamos la restricción lineal de igualdad, $Aeq(x) = beq$, la cual constituye la expresión matricial de las restricciones de la formulación entera cuadrática dada en 6.1, y el acotamiento, $lb \leq x \leq ub$, de acuerdo con la misma formulación. La matriz en la función objetivo es $H = 2Q$ donde Q es la descrita en 5.12. Para la parte lineal, $f = 0$.

6.2. Relajación PNL

Como hemos adelantado, una aproximación al programa cuadrático entero binario la constituye la relajación de programación no lineal (PNL) del problema de Asignación HA. En este capítulo estudiaremos dos variantes de esta formulación. La relajación PNL con restricciones no lineales (PNLC) y con restricciones lineales (PNLL).

6.2.1. Relajación PNLC

En general, si restringimos las variables binarias a asumir valores en $\{z_1, z_2\}$, con $z_1, z_2 \in \mathbb{Z}$, la bipartición que implica el conjunto de restricciones del problema de asignación HA obliga a que la mitad de las soluciones factibles enteras tengan valor z_1 y la otra mitad z_2. La idea, para generar esta relajación de programación no lineal con restricciones no lineales, es asociar a cada una de estas soluciones factibles un vector con estos enteros como componentes en un hiperespacio euclidiano de dimensión $t \times s$, donde $t = |T| = 2n$ y $s = |S| = 2(2n-1)$. Claramente los extremos de estos vectores están sobre una hipersuperficie esférica de radio,

$$r^2 = \sum_t \sum_s X_{t,s}^2 = \frac{ts}{2} z_1^2 + \frac{ts}{2} z_2^2$$

Caso $\{-1, +1\}$

En la Sección 5.1, al plantear la formulación entera binaria del problema asignación HA, se propuso el conjunto $\{-1, +1\}$ como alternativa de soluciones para las variables enteras $X_{t,s}$, asociadas a cada vértice de V, en los siguientes términos:

$$X_{t,s} = \begin{cases} +1 & \text{si } v_{ts} \in V' \\ -1 & \text{si } v_{ts} \notin V' \end{cases}$$

En esta formulación se obtuvo, para la función objetivo, la siguiente:

$$d = \frac{1}{4} \sum_{t \in T} \sum_{s \in S} (X_r - X_{t,s})^2 \omega(\{r, v_{t\,s}\}) + \frac{1}{4} \sum_{t \in T} \sum_{s \in S \setminus \{4n-2\}} (X_{t,s} - X_{t,s+1})^2 \omega(\{v_{t\,s}, v_{t\,s+1}\})$$

En esta situación, como $r \notin V'$, se tiene que $X_r = -1$. Para efecto de la relajación utilizamos esta restricción sustituyendo el valor fijo de esta variable en la función objetivo, resultando:

$$d = \frac{1}{4} \sum_{t \in T} \sum_{s \in S} (1 + X_{t,s})^2 \omega(\{r, v_{t\,s}\}) + \frac{1}{4} \sum_{t \in T} \sum_{s \in S \setminus \{4n-2\}} (X_{t,s} - X_{t,s+1})^2 \omega(\{v_{t\,s}, v_{t\,s+1}\}) \tag{6.2}$$

Para un n dado, el número de variables es $2n(4n-2) = 4n(2n-1)$. Como las soluciones están en $\{-1,+1\}$ es inmediato ver que los puntos asociados a las mismas están ubicados sobre una hipersuperficie euclidiana esférica de radio $\sqrt{4n(2n-1)}$.
Obviamente, al optimizar la función objetivo sobre dicha hipersuperficie no se obtendrá una solución entera. En tal caso, lo mejor que procede es restringir las variables al intervalo $[-1,+1]$ sobre cada eje del hiperespacio.
Proponemos entonces la siguiente relajación del problema de asignación HA en el marco de la programación no lineal con restricción no lineal:

$$\begin{array}{lll} & \min & d \\ (PNLHA1) & \text{s.a.} & \sum_{i=1}^{t}\sum_{j=1}^{s} X_{ij}^2 = t\,s \\ & & X_{t,s} \in [-1,+1] \end{array}$$

donde d está dada por 6.2.

Caso $\{0,1\}$

En la misma Sección 5.1, propusimos el conjunto $\{0,1\}$ como alternativa de soluciones para las variables enteras $X_{t,s}$, como sigue:

$$X_{t,s} = \begin{cases} 1 & \text{si } v_{ts} \in V' \\ 0 & \text{si } v_{ts} \notin V' \end{cases}$$

Y obtuvimos la siguiente función objetivo:

$$d = \sum_{t\in T}\sum_{s\in S} X_{t,s}^2 \omega(\{r, v_{t\,s}\}) + \sum_{t\in T}\sum_{s\in S\setminus\{4n-2\}} (X_{t,s} - X_{t,s+1})^2 \omega(\{v_{t\,s}, v_{t\,s+1}\}) \tag{6.3}$$

En esta oportunidad los puntos asociados están ubicados sobre una hipersuperficie euclidiana esférica de radio $\sqrt{2n(2n-1)}$.
Proponemos ahora la siguiente relajación del problema de asignación HA en el marco de la programación no lineal con restricción no lineal, restringidas las variables al intervalo $[0,1]$ sobre cada eje del mismo hiperespacio.

$$\begin{array}{lll} & \min & d \\ (PNLCHA2) & \text{s.a.} & \sum_{i=1}^{t}\sum_{j=1}^{s} X_{ij}^2 = \frac{1}{2}t\,s \\ & & X_{t,s} \in [0,1] \end{array} \tag{6.4}$$

donde d está dada por 6.3 .
Limitamos la aplicación del código fmincon.m al cómputo de esta última relajación. El código máster, pnlhan2.m, para el caso $n = 2$, diseñado para su implementación lo mostramos en el Apéndice B. Los resultados numéricos aparecen en el Capítulo 8. El código matlab fmincon.m fue diseñado para encontrar el mínimo de una función restringida no lineal multivaluada. El problema que resuelve se especifica como sigue [7]:

$$\begin{array}{ll} \min & f(x) \\ \text{s.a.} & c(x) \le 0 \\ & ceq(x) = 0 \\ & A \cdot x \le b \\ & Aeq \cdot x = bed \\ & lb \le x \le ub \end{array}$$

En nuestro problema sólo utilizamos la restricción de igualdad no lineal, $ceq(x) = 0$, y el acotamiento, $lb \leq x \leq ub$, de acuerdo con la formulación dada en 6.4.

6.2.2. Relajación PNLL

Ahora la idea, para generar esta relajación de programación no lineal con restricciones lineales, es asociar a cada una de las soluciones factibles, un punto sobre un hiperplano en un hiperespacio euclidiano de dimensión $t \times s$, donde $t = |T| = 2n$ y $s = |S| = 2(2n-1)$. Dicho hiperplano, para el caso $\{0,1\}$, es el siguiente:

$$\sum_t \sum_s X_{t,s} = \frac{ts}{2}$$

En este caso proponemos la siguiente relajación del problema de asignación HA en el marco de la programación no lineal con una restricción lineal, restringidas las variables al intervalo $[0,1]$ sobre cada eje del hiperespacio de dimensión $t \times s$.

$$\begin{array}{lll} & \min & d \\ (PNLLHA2) & \text{s.a.} & \sum_{i=1}^{t} \sum_{j=1}^{s} X_{ij} = \frac{1}{2} t\, s \\ & & X_{t,s} \in [0,1] \end{array} \tag{6.5}$$

donde d está dada por 6.3 .
Por razones que se explican en la siguiente parte, utilizaremos el código quadprog.m para el cómputo de esta relajación. Su forma es la siguiente:

$$\begin{array}{ll} \min & \frac{1}{2} x^T H x + f^T x \\ \text{s.a.} & Aeq \cdot x = beq \\ & lb \leq x \leq ub \end{array}$$

donde las matrices H y f son las mismas referidas para la programación cuadrática, Aeq es una matriz fila de orden $(t\ s) \times 1$ con todas las componentes iguales a uno y $beq = \frac{1}{2} t\, s$.

6.3. Algoritmo de Ramificación y Acotamiento

En la práctica de la optimización se tiene establecido que el algoritmo de ramificación y acotamiento es una poderosa herramienta útil para resolver los programas enteros. Si el resultado que se obtenga es exacto o sólo una aproximación depende del tamaño del problema en cuanto al número de variables y restricciones. Este algoritmo es el que utilizamos en este trabajo para resolver nuestro programa cuadrático entero binario del problema de asignación local-visitante.
La importancia de resolver el programa entero es poder utilizar sus resultados como referencia para evaluar la relajación semidefinida del programa entero en cuestión, lo cual constituye el principal objetivo de nuestro trabajo.
Para poner en práctica el algoritmo de relajación y acotamiento se requiere relajar el problema, y los subproblemas, de manera que en cada iteración se busque una cota inferior de la función objetivo. Para tal efecto mostramos ya en los subcapítulos anteriores las relajaciones, programación no lineal y programa cuadrático.

6.3.1. Con la Relajación PNLC

La programación no lineal tiene la ventaja de que, al seleccionar la variable ramificadora, se crea un subproblema en el que sólo es necesario manipular la función objetivo y una única restricción.
Sin embargo, para el caso de la programación no lineal con restricción no lineal, la desventaja es que el programa resolvente fmincon.m requiere de muchas iteraciones para la evaluación de la función objetivo. Por otra parte, la función matlab MIQP [1], versión 1.07, no utiliza la función fmincon.m entre las opciones de función resolvente para evaluar la relajación y obtener las variables, una de las cuales será objeto de ramificación.
Debido a estos inconvenientes no se consideró en nuestro trabajo el uso de esta relajación en la aplicación del algoritmo de ramificación y acotamiento.

6.3.2. Con la Relajación QP

La desventaja del programa cuadrático es la necesidad, como es lo usual con este algoritmo, de manipular en cada iteración las matrices que se ponen en juego en la formulación de los subproblemas, lo cual podría traducirse en un número extraordinariamente alto de operaciones de cálculo. Sin embargo, tiene como gran ventaja la rapidez con la cual el resolvente, quadprog.m, evalúa la función cuadrática en cada iteración.
Para los efectos, la función matlab MIQP [1], permite resolver, entre otras opciones, el programa entero cuadrático binario con restricciones lineales, utilizando la técnica de ramificación y acotamiento, auxiliándose como resolvente de la función quadprog.m. El código MIQP utiliza una heurística en el algoritmo de ramificación y acotamiento que consiste en que, una vez seleccionada la variable ramificadora, se sustituyen, por separado, los dos valores fijos, 0 y 1, en la función objetivo y en las restricciones, resultando dos subproblemas de una dimensión menor que el problema padre. Este código es el que utilizamos aquí para resolver el programa 5.24 para el caso $\{0,1\}$.
Consideramos pertinente exponer brevemente la heurística aplicada por Bemporad [1] para la ramificación en el código MIQP. Supongamos que nos interesa resolver un programa cuadrático entero binario con restricciones lineales, de dimensión tres, como el siguiente:

$$\begin{aligned} \min \quad & 0{,}5\ [x_1x_2x_3] \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + [x_1x_2x_3] \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix} \\ \text{s.a.} \quad & \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \\ & 0^3 \leq [x_1x_2x_3] \leq 1^3 \end{aligned}$$

Este es el problema padre, que se almacena en un arreglo tipo estructura, donde podemos identificar, siguiendo la simbología de la función quadprog.m, las matrices,

$$prob.H \quad = \quad \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}$$

El lector podrá también identificar las matrices y vectores, $prob.f$, $prob.Aeq$ y $prob.beq$, donde la partícula eq se refiere a las restricciones de igualdad. Implícito se considera el arreglo de término independiente en la función objetivo, $prob.e = 0$.

Si para la ramificación asignamos, por ejemplo, el valor $x_2 = a$, se sustituye este valor en el problema padre y se *despeja*, resultando el subproblema,

$$\begin{array}{ll} \min & 0{,}5\ [x_1 x_3] \begin{pmatrix} h_{11} & h_{13} \\ h_{31} & h_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \end{pmatrix} + [x_1 x_3] \begin{pmatrix} f_1 \\ f_3 \end{pmatrix} + a\ [x_1 x_3] \begin{pmatrix} h_{12} \\ h_{32} \end{pmatrix} + 0{,}5\, a^2 h_{22} + a f_2 \\ \text{s.a.} & \begin{pmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 - a\, a_{12} \\ b_2 - a\, a_{22} \end{pmatrix} \\ & 0^2 \leq [x_1 x_3] \leq 1^2 \end{array}$$

Al sustituir alternativamente los valores, $a = 0$ y $a = 1$, la ramificación del problema padre en dos subproblemas, que en el código miqp.m se simbolizan $p0$ y $p1$, incluye las matrices:

$$\begin{array}{rr} p0.H = p1.H = \begin{pmatrix} h_{11} & h_{13} \\ h_{31} & h_{33} \end{pmatrix} & p0.Aeq = p1.Aeq = \begin{pmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{pmatrix} \\ p0.f = \begin{pmatrix} f_1 \\ f_3 \end{pmatrix} & p1.f = \begin{pmatrix} f_1 + h_{12} \\ f_3 + h_{32} \end{pmatrix} \\ p0.e = 0 & p1.e = 0{,}5\, h_{22} + f_2 \\ p0.beq = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} & p1.beq = \begin{pmatrix} b_1 - a_{12} \\ b_2 - a_{22} \end{pmatrix} \end{array}$$

El código máster, miqphan2.m, para aplicar el miqp.m al caso $n = 2$, aparece en el Apéndice B., y los resultados alcanzados para los casos más sencillos, $n = 2$ y 3, se muestran en el Capítulo 8. Observemos que, para un n dado, la ramificación consta de $4n(2n-1)$ niveles, lo que implica la existencia de $2^{4n(2n-1)+1}$ nodos que potencialmente invocarían el resolvente quadprog.m, en caso de factibilidad y de no superar el acotamiento, para posteriormente ramificarse. Esto constituye un grave inconveniente ya que la ramificación y acotamiento se constituye en una enumeración parcial de crecimiento no polinomial, a pesar de que la *poda* reduzca la enumeración en un 75 %. De allí que no hayamos logrado resultados a partir de $n = 4$.

6.3.3. Con la Relajación PNLL

Hay una diferencia clara en la aplicación del MIQP para el cómputo del QP y del PNLL como relajaciones. En el programa cuadrático, las restricciones del problema original de asignación local-visitante están contenidas en la ecuación matricial, $Aeq \cdot x = beq$. Contrariamente, en el PNLL las restricciones del problema original no aparecen en dicha ecuación matricial. Ello supone la necesidad de modificar en parte el código original del MIQP para considerar estas restricciones.
Brevemente, consideremos ahora el problema padre para un ejemplo sencillo de dimensión ocho. Obsérvese que la matriz restricción, Aeq, para el PNLL es una matriz fila de acuerdo con la formulación 6.5.

$$\min \quad 0{,}5\ [x_1 x_2 \cdots x_8] \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} & h_{16} & h_{17} & h_{18} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} & h_{26} & h_{27} & h_{28} \\ h_{31} & h_{32} & h_{33} & h_{34} & h_{35} & h_{36} & h_{37} & h_{38} \\ h_{41} & h_{42} & h_{43} & h_{44} & h_{45} & h_{46} & h_{47} & h_{48} \\ h_{51} & h_{52} & h_{53} & h_{54} & h_{55} & h_{56} & h_{57} & h_{58} \\ h_{61} & h_{62} & h_{63} & h_{64} & h_{65} & h_{66} & h_{67} & h_{68} \\ h_{71} & h_{72} & h_{73} & h_{74} & h_{75} & h_{76} & h_{77} & h_{78} \\ h_{81} & h_{82} & h_{83} & h_{84} & h_{85} & h_{86} & h_{87} & h_{88} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} + [x_1 x_2 \cdots x_8] \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ f_8 \end{pmatrix}$$

$$\text{s.a.} \qquad [a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7\ a_8] \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{pmatrix} = b$$

$$0^8 \leq [x_1\ x_2\ x_3\ x_4\ x_5\ x_6\ x_7\ x_8] \leq 1^8$$

Supongamos ahora que el grupo de variables $\{x_1 x_5 x_6 x_8\}$ forman un grupo restringido del problema original de asignación local-visitante. Para incluir estas restricciones en la ramificación se sustituyen las mismas por los valores $x_1 = 0$, $x_5 = 1$, $x_6 = 1$, $x_8 = 0$, con lo que obtenemos el subproblema $p0$. Igualmente, se sustituyen, $x_1 = 1$, $x_5 = 0$, $x_6 = 0$, $x_8 = 1$, para obtener el subproblema $p1$. Las matrices correspondientes a estos subproblemas son:

$$p0.H = p1.H = \begin{pmatrix} h_{22} & h_{23} & h_{24} & h_{27} \\ h_{32} & h_{33} & h_{34} & h_{37} \\ h_{42} & h_{43} & h_{44} & h_{47} \\ h_{72} & h_{77} & h_{74} & h_{77} \end{pmatrix} \qquad p0.Aeq = p1.Aeq = [a_2 a_3 a_4 a_7]$$

$$p0.f = \begin{pmatrix} f_2 + h_{25} + h_{26} \\ f_3 + h_{35} + h_{36} \\ f_4 + h_{45} + h_{46} \\ f_7 + h_{75} + h_{76} \end{pmatrix} \qquad p1.f = \begin{pmatrix} f_2 + h_{21} + h_{28} \\ f_3 + h_{31} + h_{38} \\ f_4 + h_{41} + h_{48} \\ f_7 + h_{71} + h_{78} \end{pmatrix}$$

$$p0.e = 0{,}5\, h_{55} + 0{,}5\, h_{66} + f_5 + f_6 + h_{56} \qquad p1.e = 0{,}5\, h_{11} + 0{,}5\, h_{88} + f_1 + f_8 + h_{18}$$

$$p0.beq = b - a_5 - a_6 \qquad p1.beq = b - a_1 - a_8$$

Al aplicar esta heurística, los niveles de la ramificación disminuyen en un 75 % en vista de que basta tomar la variable representante de cada grupo restringido de cuatro variables. Luego, para un valor dado de n, la ramificación consta sólo de $n(2n-1)$ niveles, lo que supone $2^{n(2n-1)+1}$ nodos. Desafortunadamente, al incluir las restricciones del problema original en la ramificación, no ocurre poda por infactibilidad, lo que produce una enumeración parcial de crecimiento exponencial al igual

que en el caso QP.
El código máster, miqphan2.m, para aplicar el miqp.m con esta modificación es el mismo que para la relajación QP, y los resultados alcanzados para los casos más sencillos, $n = 2$ y 3, se muestran en el Capítulo 8.

6.3.4. Ramificación Codiciosa

Hasta ahora no se ha tenido éxito en nuestro propósito de resolver el programa entero cuadrático binario utilizando el algoritmo de ramificación y acotamiento, debido al gran número de variables sujetas a ramificación a partir de $n = 4$. Lo que procede es idear una técnica que nos proporcione, al menos, un resultado aproximado. Proponemos aquí una técnica de ramificación y acotamiento en la que aplicamos una heurística codiciosa para la ramificación. Ésta consiste en seleccionar, en cada nivel de la ramificación, únicamente el grupo de variables restringidas a los valores $(0, 1, 1, 0)$ o $(1, 0, 0, 1)$ y que arroje la menor contribución a la función objetivo.
Para visualizar esta técnica aprovechamos el subproblema obtenido en el subcapítulo anterior para la instancia de ocho variables, donde a cuatro de ellas se les dió sendos valores (a, b, c, d). La función objetivo resultante es de la forma:

$$
\begin{aligned}
& z\left(x_1 = a, x_5 = b, x_6 = c, x_8 = d\right) \quad = \\
& = 0{,}5\left[x_2 x_3 x_4 x_7\right] \begin{pmatrix} h_{22} & h_{23} & h_{24} & h_{27} \\ h_{32} & h_{33} & h_{34} & h_{37} \\ h_{42} & h_{43} & h_{44} & h_{47} \\ h_{72} & h_{77} & h_{74} & h_{77} \end{pmatrix} \begin{pmatrix} x_2 \\ x_3 \\ x_4 \\ x_7 \end{pmatrix} + \\
& +0{,}5\left[x_2 x_3 x_4 x_7\right] \left\{ \begin{pmatrix} h_{21} & h_{25} & h_{26} & h_{28} \\ h_{31} & h_{35} & h_{36} & h_{38} \\ h_{41} & h_{45} & h_{46} & h_{48} \\ h_{71} & h_{75} & h_{76} & h_{78} \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} + \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_7 \end{pmatrix} \right\} + \\
& +0{,}5\ \left[\ a\ \ b\ \ c\ \ d\ \right] \begin{pmatrix} h_{11} & h_{15} & h_{16} & h_{18} \\ h_{51} & h_{55} & h_{56} & h_{58} \\ h_{61} & h_{65} & h_{66} & h_{68} \\ h_{81} & h_{85} & h_{86} & h_{88} \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} + \left[\ a\ \ b\ \ c\ \ d\ \right] \begin{pmatrix} f_2 \\ f_3 \\ f_4 \\ f_7 \end{pmatrix}
\end{aligned}
$$

La contribución particular del grupo $\{x_1 x_5 x_6 x_8\}$ se obtiene haciendo iguales a cero el resto de las variables. En este ejemplo serían, $x_2 = x_3 = x_4 = x_7 = 0$. La ramificación se obtiene dando al grupo los valores, $0, 1, 1, 0$, con lo que se obtiene la contribución $zp0$, y los valores, $1, 0, 0, 1$, para obtener la contribución $zp1$.

$$
\begin{aligned}
zp0\left(x_1 = 0, x_5 = 1, x_6 = 1, x_8 = 0\right) &= 0{,}5\left(h_{55} + h_{56} + h_{65} + h_{66}\right) + \left(f_5 + f_6\right) \\
zp1\left(x_1 = 1, x_5 = 0, x_6 = 0, x_8 = 1\right) &= 0{,}5\left(h_{11} + h_{18} + h_{81} + h_{88}\right) + \left(f_1 + f_8\right)
\end{aligned}
$$

La aplicación de esta técnica en nuestro problema de asignación local-visitante es como sigue. En la instancia $n = 2$, el total de variables es 24. Por tanto forman seis grupos de variables relacionadas. En el primer nivel de la ramificación se asignan los valores, $0, 1, 1, 0$ y $1, 0, 0, 1$ a cada uno de los seis grupos, obteniéndose así doce valores de z, la función objetivo. Se selecciona el menor valor y se crea un subproblema, utilizando la misma técnica de ramificación del miqp.m, para los restantes grupos de variables. En el segundo nivel, y los siguientes, se aplica la misma técnica para conseguir la menor

contribución entre las variables restantes.
La poda por acotamiento puede ocurrir si nos encontramos con más de una contribución mínima en un mismo nivel, lo cual es posible para los primeros niveles.
En la práctica hemos notado que la eficiencia de esta técnica es altamente influenciada por la data. Particularmente, a mayor número de cifras significativas con que se valoren las distancias entre las sedes de los equipos participantes en el torneo deportivo, menor es la ocurrencia de obtener más de una contribución mínima en un mismo nivel, lo que se traduce en un proceso computacional mucho más rápido. Sin embargo, en contrapocisión, es posible que la solución encontrada no sea la mejor aproximación.
En el Apéndice B. incluimos el código ramifacotha.m, y el código máster ramifacothan2.m para $n = 2$, diseñados con el propósito de aplicar esta variante codiciosa del algoritmo de ramificación y acotamiento. Nuestro aporte lo constituye la heurística de la ramificación codiciosa y la inclusión en el código de las restricciones del problema original, fuera de la matriz de restricciones. El resto, la creación de los subproblemas, su almacenamiento y, demás artes, pertenecen al MIQP.m de Bemporad y Mignone [1]. Los resultados numéricos se muestran en el Capítulo 8.

Capítulo 7

Programa Semidefinido

7.1. Programa Semidefinido del Problema Asignación HA

En [5], Helmberg desarrolla un programa semidefinido del problema Max-Cut, el cual puede adaptarse al problema Asignación HA, partiendo de la función objetivo encontrada con la formulación Min Res Cut del mismo,

$$d = \sum_{e \in E \cap \delta(V')} \omega(e) \tag{7.1}$$

Para ello definimos el conjunto de índices $I_V = \{1, 2, \cdots, m\}$, con $m = 2n(4n-2)+1$, donde cada $i \in I_V$ está asociado a cada vértice de V del grafo $G = (V, E)$ del Min Res Cut para el problema Asignación HA.
Para cada vértice de V asociamos una variable x_i, tal que, $i = 2(2n-1)(t-1)+s$, siendo $t \in T$ y $s \in S$, reservándose el último índice, $i = 2n(4n-2)+1$, para la variable asociada al vértice específico r. Tales variables se definen como componentes de un vector de corte:

$$x = (x_i) \in \{-1, +1\}^{2n(4n-2)+1}, \; i \in I_V$$

siendo,

$$x_i = \begin{cases} +1 & \text{si } \; v_{ts} \in V' \\ -1 & \text{si } \; v_{ts} \notin V' \end{cases} \tag{7.2}$$

Obviamente, $x_{2n(4n-2)+1} = -1$, ya que, por definición, $r \notin V'$. Sin embargo, el programa semidefinido requiere que esta componente permanezca explícita en la formulación.
Sea $\{i, j\} \in E$, la arista que une los vértices asociados a los índices i y j de I_V. Entonces, la función objetivo, expresada con esta notación, queda de la forma siguiente:

$$d = \sum_{\{i,j\} \in E \cap \delta(V')} h(x_i, x_j)\omega_{ij} \tag{7.3}$$

donde h es una función elemental que debe satisfacer:

$$h(x_i, x_j) = \begin{cases} 0 & \text{si } \; \{i, j\} \in E \cap \delta(V') \\ 1 & \text{si } \; \{i, j\} \notin E \cap \delta(V') \end{cases}$$

Por otra parte, $\omega_{ij} = \omega(\{i, j\})$ es el peso de la arista $\{i, j\} \in E$. La matriz $W = (\omega_{ij}) \in S_n$, es la matriz de adyacencia pesada del grafo G. Obviamente, es simétrica por ser un grafo no dirigido y, ademas, $\omega_{ij} = 0$ si $\{i, j\} \notin E$.

Proponemos una formulación algebraica del problema Asignación HA, expresando la función h en (7.3) como sigue:

$$h(x_i, x_j) = \frac{1 - x_i x_j}{2}$$

Con lo cual obtenemos:

$$d \;=\; \sum_{i<j} \frac{1 - x_i x_j}{2}\, \omega_{ij} \tag{7.4}$$

donde $i < j$ indica que $\{i, j\}$ y $\{j, i\}$ se cuentan una única vez, por tratarse de la misma arista ya que el grafo es no dirigido. Además, obsérvese que,

$$\frac{1 - x_i x_j}{2} \;=\; \begin{cases} 0 & \text{si } \; x_i = x_j \\ 1 & \text{si } \; x_i = -x_j \end{cases} \tag{7.5}$$

Por tanto, si $x_i = x_j$, significa que los vértices asociados a i y j están en el mismo conjunto, V' o $V \setminus V'$, mientras que, $x_i = -x_j$, indica que ambos vértices están en diferentes conjuntos. Entonces la expresión $\frac{1}{2}(1 - x_i x_j)$ produce el *vector incidente* asociado con el vector de corte $x = (x_i)$, $i \in I_V$, ya que evalúa a 1 si la arista $\{i, j\}$ está en el corte $\delta(V')$, y a 0 en caso contrario. Por consiguiente, (7.1) y (7.3) representan la misma función objetivo.
Aprovechando la simetría de W y el hecho de que $x_i x_i = 1, \forall i \in I_V$, podemos modificar la expresión (7.3) para alcanzar una forma matricial:

$$\frac{1}{2}\sum_{i<j} \omega_{ij}(1 - x_i x_j) = \frac{1}{4}\sum_{i,j=1}^{m} \omega_{ij}(x_i x_i - x_i x_j) = \frac{1}{4}\left[\sum_{i=1}^{m}\sum_{j=1}^{m} \omega_{ij} x_i x_i - \sum_{i=1}^{m}\sum_{j=1}^{m} \omega_{ij} x_i x_j\right]$$

Esta última no es más que el desarrollo matricial de la función objetivo expresada como,

$$d = \frac{1}{4} x^T (Diag(We) - W)x$$

donde, $e \in \mathbb{R}^m$, es el vector de componentes 1.
La matriz $Diag(We) - W$, se conoce como *matriz de Laplace* del grafo G. En términos de la misma nos resulta:

$$d = x^T C x \qquad \text{con} \qquad C = \frac{1}{4}(Diag(We) - W)$$

Por último, como $x^T C x \in \mathbb{R}$, se tiene que, $x^T C x = \mathrm{tr}(x^T C x) = \langle Cx, x\rangle$. Por la propiedad de invariancia de la traza ante la rotación en la multiplicación de matrices, $\mathrm{tr}(x^T C x) = \mathrm{tr}(C x x^T) = \left\langle C, (x x^T)^T \right\rangle = \left\langle C, x x^T \right\rangle$, de donde la función objetivo queda como el producto interno,

$$d = \langle C, X \rangle$$

donde, para $x \in \{-1, +1\}^m$, la matriz, $X = x x^T$ es semidefinida positiva, cuyas entradas de la diagonal son todas, $x_i x_i = 1$, esto es, $\mathrm{diag}(X) = e$. Ademas, como $\mathrm{rg}(x) = 1$, será también, $\mathrm{rg}(X) = 1$. En detalle, esta matriz es,

$$X = \begin{pmatrix} x_1 x_1 & x_1 x_2 & \dots & x_1 x_m \\ x_2 x_1 & x_2 x_2 & \dots & x_2 x_m \\ \vdots & \vdots & \ddots & \vdots \\ x_m x_1 & x_m x_2 & \dots & x_m x_m \end{pmatrix}$$

donde, $x_m = x_{2n(4n-2)+1} = -1$, es la variable asociada al vértice específico r.
Veamos la matriz de Laplace en detalle, donde los pesos $\omega_{ii} = 0$:

$$We = \begin{pmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1m} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{m1} & \omega_{m2} & \dots & \omega_{mm} \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} \omega_{11} + \omega_{12} + \cdots + \omega_{1m} \\ \omega_{21} + \omega_{22} + \cdots + \omega_{2m} \\ \vdots \\ \omega_{m1} + \omega_{m2} + \cdots + \omega_{mm} \end{pmatrix}$$

$$\text{Diag}(We) = \begin{pmatrix} \sum_{j=1}^{m} \omega_{1j} & 0 & \cdots & 0 \\ 0 & \sum_{j=1}^{m} \omega_{2j} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{j=1}^{m} \omega_{mj} \end{pmatrix}$$

con $\omega_{ii} = 0$, $i = 1, \cdots, m$. Consecuentemente,

$$\text{Diag}(We) - W = \begin{pmatrix} \sum_{j=1}^{m} \omega_{1j} & 0 & \cdots & 0 \\ 0 & \sum_{j=1}^{m} \omega_{2j} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{j=1}^{m} \omega_{mj} \end{pmatrix} - \begin{pmatrix} \omega_{11} & \omega_{12} & \dots & \omega_{1m} \\ \omega_{21} & \omega_{22} & \dots & \omega_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \omega_{m1} & \omega_{m2} & \dots & \omega_{mm} \end{pmatrix}$$

$$C = \frac{1}{4} \begin{pmatrix} \sum_{j=1}^{m} \omega_{1j} - \omega_{11} & -\omega_{12} & \dots & -\omega_{1m} \\ -\omega_{21} & \sum_{j=1}^{m} \omega_{2j} - \omega_{22} & \dots & -\omega_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ -\omega_{m1} & -\omega_{m2} & \dots & \sum_{j=1}^{m} \omega_{mj} - \omega_{mm} \end{pmatrix}$$

donde, $\omega_{ii} = 0, \forall i \in I_V$.
La expresión, $\text{diag}(X) = e$, constituye una restricción que surge naturalmente del desarrollo para obtener esta formulación matricial. Puede escribirse de la forma,

$$\langle E_{ii,ii}, X \rangle = \left\langle \begin{pmatrix} 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & 1^{(i)} & \dots & 0 \\ \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 \end{pmatrix}, \begin{pmatrix} x_1x_1 & \dots & x_1x_i & \dots & x_1x_m \\ \vdots & \ddots & \vdots & & \vdots \\ x_ix_1 & \dots & x_ix_i & \dots & x_ix_m \\ \vdots & & \vdots & \ddots & \vdots \\ x_mx_1 & \dots & x_mx_i & \dots & x_mx_m \end{pmatrix} \right\rangle = x_ix_i = 1$$

viéndose aquí que, $E_{ii,ii}, \forall i \in I_V$, es la matriz cuya entrada E_{ii} es 1 y toda otra entrada es 0. De éstas hay $|V|$ restricciones.

En cuanto a las otras restricciones, éstas están dadas por $E_{cut} \subseteq \delta(V')$ y $x_{2n(4n-2)+1} = -1$, con, $E_{cut} = E_{cut\,v} \cup E_{cut\,h}$, siendo,

$$\begin{aligned} E_{cut\,v} &= \{\{v_{t\,s}, v_{\tau(t,s)\,s)}\} : (t,s) \in T \times S\} \\ E_{cut\,h} &= \{\{v_{t\,s}, v_{t\,s'}\} : t \in T,\ s, s' \in S,\ \tau(t,s) = \tau(t,s'),\ s \neq s'\} \end{aligned}$$

Por consiguiente, de acuerdo con la matriz itinerario dada, se construye una matriz $E_{ij,j\,i}$ para cada arista $\{i,j\} \in E_{cut}$, de manera que la restricción $E_{cut} \subseteq \delta(V')$, se traduzca en $x_i x_j = x_j x_i = -1$. Esto es, $E_{ij,j\,i}$ es una de las $2n(4n-2)$ matrices con entradas E_{ij} y $E_{j\,i}$ iguales a 1, y toda otra entrada igual a 0. Entonces, al multiplicarla internamente por X se tendrá, para cada arista, que $x_i x_j + x_j x_i = -2$.
En detalle,

$$\langle E_{ij,j\,i}, X\rangle = \left\langle \begin{pmatrix} 0 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & 0 & \dots & 1 & \dots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \dots & 1 & \dots & 0 & \dots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 0 \end{pmatrix}, \begin{pmatrix} x_1x_1 & \dots & x_1x_i & \dots & x_1x_j & \dots & x_1x_m \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ x_ix_1 & \dots & x_ix_i & \dots & x_ix_j & \dots & x_ix_m \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ x_jx_1 & \dots & x_jx_i & \dots & x_jx_j & \dots & x_jx_m \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ x_mx_1 & \dots & x_mx_i & \dots & x_mx_j & \dots & x_mx_m \end{pmatrix} \right\rangle =$$

$$= x_i x_j + x_j x_i = -2$$

Con este resultado concluye la forma matricial semidefinida del programa entero binario del problema Asignación HA. No se trata aún de un programa semidefinido debido a la restricción, $\mathrm{rg}(X) = 1$. El programa semidefinido del problema Asignación HA se obtiene al relajar la forma matricial del programa entero, suprimiendo la restricción sobre el rango de X. Queda así,

$$(SDHA) \qquad \begin{aligned} \min\ & \langle C, X\rangle \\ \text{s.a.}\ & \langle E_{ii,ii}, X\rangle = 1,\ \forall i \in I_V \\ & \langle E_{ij,j\,i}, X\rangle = -2,\ \forall \{i,j\} \in E_{cut} \\ & X \succeq 0,\ X \in \mathbb{R}^{I_V \times I_V} \end{aligned} \tag{7.6}$$

Para intentar la formulación dual correspondiente, veamos que,

$$\langle C, X\rangle = \sum_{i=1}^{m} C_{ii} x_i x_i + \sum_{\substack{i,j=1 \\ i\neq j}}^{m} C_{ij} x_i x_j$$

Las primeras restricciones corresponden a las variables, $x_i x_i$, $i = 1, \cdots, m$ y se les asocia las variables duales, u_{ii}, $i = 1, \cdots, m$. Entonces, la función objetivo dual, tendrá los términos, $\sum_i 1 \cdot u_{ii}$. Las segundas restricciones corresponden a las sumas $x_i x_j + x_j x_i$ y se les asocia las variables duales, u_{ij}, $i, j = 1, \cdots, m$, $i \neq j$. Los términos que aportan a la función objetivo dual son de la forma, $\sum_{i\neq j}(-2) \cdot u_{ij} = \sum_{i>j}(-1) \cdot u_{ij} + \sum_{i<j}(-1) \cdot u_{j\,i}$.

La función objetivo dual quedaría de la forma,

$$\langle B, U\rangle = \left\langle \begin{pmatrix} 1 & -1 & \dots & -1 \\ -1 & 1 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \dots & 1 \end{pmatrix}, \begin{pmatrix} u_{11} & u_{12} & \dots & u_{1m} \\ u_{21} & u_{22} & \dots & u_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1} & u_{m2} & \dots & u_{mm} \end{pmatrix} \right\rangle$$

Como cada coeficiente, de todos los $x_i x_j$, es 1, entonces, las restricciones duales son todas de la forma $1 \cdot u_{ij} = u_{ij} \geq C_{ij},\ i, j = 1, \cdots, m$. Esto es,

$$U \succeq C$$

donde se introducirá una holgura $Z \succeq 0$. Es así como el par primal-dual del programa semidefinido del Problema Asignación HA, podría ser el siguiente

$$\begin{array}{lllll} & \min & \langle C, X\rangle & \max & \langle B, U\rangle \\ & \text{s.a.} & \langle E_{ii,ii}, X\rangle = 1,\ \forall i \in I_V & \text{s.a.} & U - Z = C \\ (PSDHA) & & \langle E_{ij,j\,i}, X\rangle = -2,\ \forall \{i,j\} \in E_{cut} & (DSDHA) & \\ & & X \succeq 0 & & Z \succeq 0 \end{array} \tag{7.7}$$

Sin embargo, esta formulación dual no es la fuerte para desarrollar un algoritmo que resuelva, al menos aproximadamente, el problema de Asignación HA. La función objetivo dual puede (o debe) modificarse para obtener otra función objetivo del tipo, $b^T y$, con $b, y \in \mathbb{R}^p$, donde p es el número de restricciones del primal.
En la formulación estándar propuesta por Helmberg [5], el par primal-dual del programa semidefinido es el siguiente:

$$\begin{array}{lll} & \min & \langle C, X\rangle \\ \text{(PSDP)} & \text{s.a} & \mathcal{A}\,X = b \\ & & X \succeq 0 \end{array}$$

$$\begin{array}{lll} & \max & \langle b, y\rangle \\ \text{(DSDP)} & \text{s.a} & \mathcal{A}^T\, y + Z = C \\ & & y \in \mathbb{R}^p, Z \succeq 0 \end{array}$$

donde, $\mathcal{A}X$, expresa la acción del operador lineal, $\mathcal{A}$:$S_m \to \mathbb{R}^p$, definido como sigue:

$$\mathcal{A}\,X \quad = \quad \begin{pmatrix} \langle \mathcal{A}_1, X\rangle \\ \langle \mathcal{A}_2, X\rangle \\ \vdots \\ \langle \mathcal{A}_p, X\rangle \end{pmatrix}$$

En nuestro problema se identifican,

$$\mathcal{A}\,X \quad = \quad \begin{pmatrix} \langle E_{ii,ii}, X\rangle\,,\ \forall i \in I_V \\ \langle E_{ij,j\,i}, X\rangle\,,\ \forall \{i,j\} \in E_{cut} \end{pmatrix}$$

y, $b \in \mathbb{R}^p$, es de la forma,

$$b = \left[(1)^{2n(4n-2)+1}, (-2)^{2n(4n-2)}\right]$$

ya que, $|I_V| = 2n(4n-2)+1$ y $|E_{cut}| = 2n(4n-2)$. O sea, $p = 4n(4n-2)+1$.
Se introducen la variables duales, y_k, $k = 1, \ldots, p$, asociadas a sendas restricciones. La expresión $\mathcal{A}^T y$, representa la acción del operador adjunto de $\mathcal{A}$, $\mathcal{A}^T : \mathbb{R}^p \rightarrow S_m$, como sigue,

$$\mathcal{A}^T\, y = \sum_{k=1}^{p} y_k \mathcal{A}_k$$

Con esta formulación, siguiendo la exposición de Laurent y Rendl [6] y de Helmberg [5], se puede desarrollar un algoritmo, basado en el método de punto interior, para resolver, en tiempo polinomial y con un grado de precisión prefijado, el programa semidefinido.

7.2. Algoritmo de Aproximación Aleatoria

Como se vió en la sección anterior, el programa semidefinido del problema Asignación HA se obtiene al relajar la forma matricial del programa entero, suprimiendo la restricción sobre el rango de X, resultando la formulación 7.6,

$$(SDHA) \qquad \begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.a.} & \langle E_{ii,ii}, X\rangle = 1,\ \forall i \in I_V \\ & \langle E_{ij,j\,i}, X\rangle = -2,\ \forall \{i,j\} \in E_{cut} \\ & X \succeq 0,\ X \in \mathbb{R}^{I_V \times I_V} \end{array}$$

Goemans [4] generó un algoritmo para obtener una solución aproximada del problema Max Cut, el cual fue aplicado por Suzuka (y otros) [11] para resolver el problema de asignación HA sin hacer consideraciones sobre la cota de aproximación, desconocida para la fecha de publicación del trabajo de Suzuka.
La idea es la siguiente: Una vez resuelto el programa semidefinido mediante el método del punto interior sugerido en la sección anterior, el resultado se espera que sea una matriz definida positiva X tal que rg$X \neq 1$ debido a la relajación sobre el programa cuadrático entero HA original. Por tanto, la solución $X = (X_{i,j})$ no será de la forma $X_{i,j} = x_i x_j$, con $x_i \in \{-1, +1\}$. Lo que se habría logrado es una relajación del producto $x_i x_j$ a un producto punto de vectores, $v_i^T v_j$, donde las componentes de cada v_i son tales que, $v_{i,k} \in [-1, +1]$, con $i, k = 1, 2, \cdots, m$.
Siguiendo la notación vectorial propuesta por Helmberg [5], la misma se consigue, asociando cada vértice $i \in V$, del grafo $G = (V, E)$, con un vector $v_i \in \mathbb{R}^m$, $i = 1, 2, \cdots, m$. Por consiguiente, cada solución factible X del SDHA, puede interpretarse como una matriz de Gram del conjunto de m vectores $\{v_1, \cdots, v_m\}$.
Como X es definida positiva, se sabe que existen matrices $B \in M_{m,n}$ tales que $X = B^T B$. Particularmente, en este caso, $B \in M_m$, estando formada B con los vectores v_i como vectores columna, esto es,

$$B = \begin{pmatrix} v_1 & v_2 & \cdots & v_m \end{pmatrix}$$

La matriz de Gram se construiría de la forma siguiente,

$$X \equiv B^T B = \begin{pmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_m^T \end{pmatrix} \begin{pmatrix} v_1 & v_2 & \cdots & v_m \end{pmatrix} = \begin{pmatrix} v_1^T v_1 & v_1^T v_2 & \dots & v_1^T v_m \\ v_2^T v_1 & v_2^T v_2 & \dots & v_2^T v_m \\ \vdots & \vdots & \ddots & \vdots \\ v_m^T v_1 & v_m^T v_2 & \dots & v_m^T v_m \end{pmatrix}$$

Por otra parte, la restricción original, $\text{diag}(X) = e$, exige que, $\sqrt{v_i^T v_i} = \|vi\| = 1$, $\forall i = 1, \cdots, m$. Es así como se consigue una relajación a una esfera unitaria de $\mathbb{R}^m$ y los productos, $x_i x_j$, se relajan a los productos escalares, $v_i^T v_j = 1$ al ser cada componente, $v_{i,k} \in [-1, +1]$.
Ahora puede presentarse el algoritmo de aproximación aleatoria de Goemans [4] para el Max Cut, adaptado aquí al problema de asignación HA, en los siguientes términos:

(1) Se resuelve el par primal-dual del programa semidefinido del Problema Asignación HA mediante el método de punto interior. Como solución (posiblemente) óptima se obtiene una matriz definida positiva $X_0 = (v_i v_j)$.

(2) Se efectúa una descomposición de Cholesky de esta solución de la forma, $X_0 = B^T B$, con $B = (v_1 \dots v_m)$, donde cada vector columna v_i de componentes en $[-1, +1]$ es una relajación de las variables $x_i \in \{-1, +1\}$, $i = 1, \dots m$ y además $\|v_i\| = 1$, para todo $i \in I_V$.

(3) Se genera un vector $u \in \mathbb{R}^m$ de forma uniformemente aleatoria sobre la superficie de una bola unitaria m-dimensional y se establece que, $V_1 = \left\{ i \in V : \, u^T v_i \geq 0 \right\}$. De aquí se obtiene el subconjunto de vértices:

$$V' = \begin{cases} V_1 & \text{si} \quad r \notin V_1, \text{ o bien si, } u^T v_m < 0 \\ V \setminus V_1, & \text{si} \quad r \in V_1, \text{ o bien si, } u^T v_m \geq 0 \end{cases}$$

ya que el vector v_m es la relajación del vértice específico r.
Obviamente habrá que evaluar si se cumplen las restricciones del problema original una vez que se hayan distribuidos los vértices en V' y $V \setminus V'$.

Los aspectos computacionales se describen en el Apéndice B, mientras que los resultados, en el Capítulo 8. Para la aplicación del programa semidefinido se utilizó el software SDPT3 [10], versión 3.0. En el mismo Apéndice mostramos el código máster, semidhan2.m, para su aplicación, el cual contiene a su vez la implementación del algoritmo aleatorio de Goemans, así como la evaluación de la factibilidad del minimizador resultante.

Capítulo 8

Resultados Numéricos

El problema de asignación local visitante fue resuelto de maneras exacta para algunas pocas instancias. En la mayoría de los casos fue posible obtener solamente aproximaciones al mínimo del recorrido total de todos los equipos. En este capítulo se recoge gran parte de los resultados numéricos obtenidos con todos los algoritmos que nos condujeron a la resolución del problema para los casos desde $n = 2$ hasta $n = 6$. Los códigos diseñados para la aplicación de los diferentes algoritmos aparecen en el Apéndice B. La data utilizada, las distancias y los itinerarios, en el Apéndice A.

Para cada algoritmo, primero mostramos una tabla resumen para estos cinco valores de n. Destacamos sobre todo el número de iteraciones, el tiempo y el valor óptimo de la función objetivo. Para los métodos de resolución aproximada, ramificación y acotamiento y programa semidefinido, incluimos una columna para reflejar la discrepancia con respecto al resultado obtenido por enumeración.

Seguidamente mostramos la salida que proporcionan los códigos aplicados. Sólo detallamos el caso $n = 2$. Los demás casos, por su tamaño, preferimos mostrarlos en forma fragmentada. No se muestran los casos que resultaron computacionalmente inviables. Estos corresponden a los valores $n = 5, 6$ para la enumeración y $n = 4, 5, 6$ para la ramificación y acotamiento con relajación QP y PNLL. Para la relajación codiciosa y el programa semidefinido con aproximación aleatoria de Goemans, el resultado es para todos los valores de n aquí tratados.

8.1. Resultados de la Enumeración

La solución exacta se alcanza mediante la enumeración exhaustiva de todas las soluciones factibles. Esto fue posible para los casos más sencillos, esto es, los valores más bajos de n. Para las situaciones más complejas, $n \geq 5$, se aplicó enumeración parcial aleatoria.
El resultado de aplicar la enumeración es el siguiente:

n	Iteraciones	Tiempo (aprox.)	dmin
2	64	0,023 seg.	108
3	32768	5,52 seg.	980,96
4	268435456	49615,5 seg.	1543,15
5	(50000000)	5800,2 seg.	$\leq$ 2068,437
6	(50000000)	7113,6 seg.	$\leq$ 2886,82

Debido a lo inviable del cómputo para valores grandes de n, se dispuso arbitrariamente de algunas, relativamente pocas, iteraciones de las 35,184,372,088,832 necesarias para $n = 5$ y de las 73,786,976,294,838,206,464 necesarias para $n = 6$.
La enumeración exhaustiva permite, además, ver todas las soluciones factibles. Para $n = 2$, pudimos observar que existen cuatro soluciones óptimas. Más aún, las 64 soluciones factibles se distribuyen en siete grupos que producen sendos valores del recorrido total de los equipos. La distribución es como sigue:

recorrido	108	110	114	117	119	120	122
nº de sol.	4	12	4	12	12	12	8

En detalle, se muestra a continuación el minimizador en forma de matriz $(X_{t,s})$, con $t = 4$ y $s = 6$. Esta solución se corresponde con la matriz asignación para el recorrido mínimo que mostramos seguidamente. El valor 1 (0) del minimizador, equivale al valor 'A' ('H') en la matriz asignación. Lo mismo es aplicable para los demás valores de n.
Resultado de la Enumeración para $n = 2$:

```
          VARIABLE X MINIMIZADORA
     1     0     0     0     1     1
     0     0     1     1     1     0
     1     1     0     0     0     1
     0     1     1     1     0     0
        ASIGNACIÓN DE MÍNIMA DISTANCIA
    'A'    'H'    'H'    'H'    'A'    'A'
    'H'    'H'    'A'    'A'    'A'    'H'
    'A'    'A'    'H'    'H'    'H'    'A'
    'H'    'A'    'A'    'A'    'H'    'H'
   DISTANCIA MÍNIMA DE RECORRIDO TOTAL
           dmin = 108
Iteraciones:   k = 64
Tiempo:        t = 0.0309999999999988 segundos
```

La inspección de todas las soluciones factibles para $n = 3$ mostró una única solución óptima. Los diferentes valores del recorrido total presentan una distribución aproximadamente normal.
Resultado de la Enumeración para $n = 3$:

```
     VARIABLE X MINIMIZADORA
     0     1     1     0     1     0     0     1     0     1
     1     0     0     0     0     1     1     1     1     0
     1     0     0     1     1     0     0     1     1     0
     0     1     1     0     0     0     1     0     1     1
     0     1     1     1     0     1     0     0     0     1
     1     0     0     1     1     1     1     0     0     0
     ASIGNACIÓN DE MÍNIMA DISTANCIA
    'H'    'A'    'A'    'H'    'A'    'H'    'H'    'A'    'H'    'A'
    'A'    'H'    'H'    'H'    'H'    'A'    'A'    'A'    'A'    'H'
    'A'    'H'    'H'    'A'    'A'    'H'    'H'    'A'    'A'    'H'
    'H'    'A'    'A'    'H'    'H'    'H'    'A'    'H'    'A'    'A'
    'H'    'A'    'A'    'A'    'H'    'A'    'H'    'H'    'H'    'A'
    'A'    'H'    'H'    'A'    'A'    'A'    'A'    'H'    'H'    'H'
   DISTANCIA MÍNIMA DE RECORRIDO TOTAL
           dmin = 980.96
Iteraciones:   k = 32768
Tiempo:        t = 5.52059598356774 segundos
```

El cómputo más ambicioso de la enumeración exhaustiva, de 13.34 horas, fue para $n = 4$.
Resultado de la Enumeración para n=4:

```
     VARIABLE X MINIMIZADORA
     1     0     0     1     1     0     0     1     0     1     0     1     0     1
     0     0     0     1     0     0     1     0     0     1     1     1     1     1
     0     1     1     0     1     1     0     0     0     1     1     0     0     1
     1     1     1     1     0     0     0     1     1     0     0     0     1     0
     1     1     1     0     1     0     0     1     1     0     0     1     0     0
     0     1     1     0     0     1     1     0     0     1     1     0     1     0
     0     0     0     1     1     1     1     1     1     0     1     0     0     0
     1     0     0     0     0     1     1     0     1     0     0     1     1     1
     ASIGNACIÓN DE MÍNIMA DISTANCIA
    'A'   'H'   'H'   'A'   'A'   'H'   'H'   'A'   'H'   'A'   'H'   'A'   'H'   'A'
    'H'   'H'   'H'   'A'   'H'   'H'   'A'   'H'   'H'   'A'   'A'   'A'   'A'   'A'
    'H'   'A'   'A'   'H'   'A'   'A'   'H'   'H'   'H'   'A'   'A'   'H'   'H'   'A'
    'A'   'A'   'A'   'A'   'H'   'H'   'H'   'A'   'A'   'H'   'H'   'H'   'A'   'H'
    'A'   'A'   'A'   'H'   'A'   'H'   'H'   'A'   'A'   'H'   'H'   'A'   'H'   'H'
    'H'   'A'   'A'   'H'   'H'   'A'   'A'   'H'   'H'   'A'   'A'   'H'   'A'   'H'
    'H'   'H'   'H'   'A'   'A'   'A'   'A'   'A'   'A'   'H'   'A'   'H'   'H'   'H'
    'A'   'H'   'H'   'H'   'H'   'A'   'A'   'H'   'A'   'H'   'H'   'A'   'A'   'A'
   DISTANCIA MÍNIMA DE RECORRIDO TOTAL
           dmin = 1543.15
Iteraciones:   k = 268435456
Tiempo:        t = 49615.4609234554 segundos
```

8.2. Resultados de la Relajación QP

Logramos otra aproximación mediante la relajación de programación cuadrática que consiste en suprimir la restricción sobre las variables de asumir sólo valores enteros. El algoritmo utilizado por la función quadprog.m [7] es una estrategia de conjunto activo. El algoritmo consta de dos fases. Primero se calcula un punto inicial factible. Luego se genera una sucesión de puntos factibles que convergen a la solución, lo cual consiste en buscar una dirección de búsqueda para generar el siguiente punto. Debido a la naturaleza cuadrática de la función objetivo, si resulta posible elegir un tamaño de paso igual a 1, sin violar las restricciones, entonces la dirección de búsqueda conseguida conduce al mínimo del programa cuadrático.
Tal parece que esto es lo que ocurre con nuestra función objetivo en la primera iteración, a la luz de los resultados que se muestran a continuación:

n	Iteraciones	Tiempo (aprox.)	fmin
2	1	0,013 seg.	42,3385
3	1	0,165 seg.	240,0990
4	1	1,558 seg.	393,2777
5	1	0,288 seg.	628,2923
6	1	0,920 seg.	874,6040

Resultado del QP para n=2:

```
Optimization terminated.
x =
    0.5426 0.5000 0.4574 0.5067 0.5000 0.4933 0.4574 0.5000
    0.5426 0.5242 0.5000 0.4758 0.4710 0.5000 0.5290 0.4758
    0.5000 0.5242 0.5290 0.5000 0.4710 0.4933 0.5000 0.5067
fval =
   42.3385
exitflag =
     1
output =
         iterations: 1
    constrviolation: -0.4574
          algorithm: 'medium-scale: active-set'
      firstorderopt: []
       cgiterations: []
            message: 'Optimization terminated.'
lambda =
      lower: [24x1 double]
      upper: [24x1 double]
      eqlin: [18x1 double]
    ineqlin: [0x1 double]
t = 0.0129316016421081 segundos
```

A continuación presentamos los resultados para el resto de valores de n. Debido a la gran longitud de los vectores columna en la salida del quadprog.m, editamos el resultados de manera que los mismos aparezcan como vectores fila.

Resultado del QP para n=3:

```
Optimization terminated.
x =
    0.5269 0.4731 0.5010 0.4636 0.5364 0.4990 0.4522 0.5162 0.4838 0.5478 0.4906 0.5094
    0.4819 0.4999 0.5021 0.4979 0.5478 0.5181 0.5001 0.4522 0.4731 0.5269 0.5053 0.4947
    0.4979 0.5021 0.5018 0.4982 0.5188 0.4812 0.5094 0.4906 0.4947 0.5053 0.4939 0.5111
    0.4889 0.4838 0.5162 0.5061 0.4954 0.5046 0.5181 0.5364 0.4636 0.4889 0.5111 0.4819
    0.4812 0.5188 0.5046 0.4954 0.4990 0.5001 0.5061 0.5010 0.4982 0.5018 0.4999 0.4939
fval =
  240.0990
exitflag =
     1
output =
         iterations: 1
    constrviolation: -0.4522
          algorithm: 'medium-scale: active-set'
      firstorderopt: []
       cgiterations: []
            message: 'Optimization terminated.'
lambda =
      lower: [60x1 double]
      upper: [60x1 double]
      eqlin: [45x1 double]
    ineqlin: [0x1 double]
t = 0.164750268539717 segundos
```

Resultado del QP para n=4:

```
Optimization terminated.
x =
    0.5274 0.4726 0.4827 0.5173 0.5029 0.4971 0.4938 0.5111 0.4999 0.5001 0.4748 0.5252
    0.4889 0.5062 0.4928 0.4888 0.4639 0.5361 0.5148 0.5129 0.4871 0.4889 0.4969 0.5031
    0.5112 0.4852 0.5111 0.5072 0.4726 0.5274 0.4653 0.4945 0.5055 0.4871 0.5129 0.5347
    0.4996 0.5004 0.5244 0.4756 0.5197 0.4803 0.4798 0.5112 0.5127 0.5055 0.4945 0.5202
    0.5051 0.4949 0.5001 0.4999 0.4888 0.5008 0.4992 0.4873 0.5072 0.4983 0.5173 0.4827
    0.5248 0.4752 0.4598 0.5402 0.5004 0.4996 0.5017 0.4992 0.5008 0.4928 0.4826 0.5174
    0.5361 0.4639 0.4752 0.5248 0.4949 0.5051 0.5046 0.4954 0.5252 0.4748 0.4803 0.5197
    0.5202 0.5017 0.5347 0.5032 0.4852 0.4798 0.5062 0.4653 0.4954 0.5046 0.4983 0.5148
    0.4968 0.4938 0.5174 0.4826 0.4873 0.4968 0.4971 0.5029 0.5402 0.4598 0.5031 0.4969
    0.4756 0.5244 0.5032 0.5127
fval =
  393.2777
exitflag =
     1
output =
```

```
         iterations: 1
    constrviolation: -0.4598
          algorithm: 'medium-scale: active-set'
      firstorderopt: []
       cgiterations: []
            message: 'Optimization terminated.'
lambda =
      lower: [112x1 double]
      upper: [112x1 double]
      eqlin: [84x1 double]
    ineqlin: [0x1 double]
t = 1.5583645407447 segundos
```

Resultado del QP para n=5:

```
Optimization terminated.
x =
    0.5124 0.5104 0.4860 0.5140 0.5064 0.4936 0.4896 0.4876 0.5122 0.4900 0.5240 0.4760
    0.4955 0.5045 0.4320 0.4878 0.5100 0.5680 0.4876 0.5204 0.5247 0.5258 0.4955 0.5045
    0.5038 0.5124 0.4796 0.4928 0.4742 0.5223 0.5063 0.4937 0.4777 0.4962 0.5072 0.4753
    0.4800 0.5200 0.5161 0.5154 0.5426 0.4839 0.4846 0.4925 0.5075 0.5100 0.5041 0.4959
    0.4937 0.5063 0.4943 0.5057 0.4900 0.4574 0.4644 0.4953 0.5122 0.5356 0.5045 0.4955
    0.4900 0.5100 0.5102 0.4898 0.4959 0.5041 0.5047 0.4878 0.5680 0.5183 0.4817 0.4320
    0.5076 0.4700 0.5140 0.4860 0.4574 0.4588 0.5100 0.4900 0.5412 0.5072 0.4896 0.5104
    0.5300 0.4924 0.4802 0.5198 0.4928 0.5426 0.4924 0.5047 0.4753 0.4846 0.4898 0.4630
    0.5154 0.4935 0.4878 0.4972 0.5028 0.5102 0.4953 0.5076 0.5370 0.5122 0.5065 0.5247
    0.4971 0.4896 0.4878 0.4742 0.5102 0.5412 0.5104 0.4787 0.4588 0.5152 0.5258 0.4898
    0.4848 0.5122 0.5057 0.4943 0.5213 0.5029 0.5356 0.4796 0.4839 0.4644 0.4633 0.5161
    0.4878 0.5065 0.5204 0.4848 0.4760 0.5240 0.5152 0.5122 0.5198 0.4802 0.4935 0.5367
    0.5200 0.4800 0.5038 0.4962 0.5367 0.5370 0.4962 0.5213 0.4898 0.5102 0.5104 0.4896
    0.5045 0.4955 0.4630 0.5038 0.4787 0.4633 0.5029 0.5300 0.4962 0.5038 0.4936 0.5064
    0.5122 0.5075 0.4925 0.5028 0.4972 0.4777 0.4700 0.4878 0.5223 0.4817 0.5183 0.4971
fval =
  628.2923
exitflag =
     1
output =
         iterations: 1
    constrviolation: -0.4320
          algorithm: 'medium-scale: active-set'
      firstorderopt: []
       cgiterations: []
            message: 'Optimization terminated.'
lambda =
      lower: [180x1 double]
      upper: [180x1 double]
      eqlin: [135x1 double]
    ineqlin: [0x1 double]
t = 0.287891407986211 segundos
```

Resultado del QP para n=6:

```
Optimization terminated.
x =
    0.4993 0.5007 0.4710 0.5290 0.5052 0.5072 0.4898 0.5102 0.4862 0.5138 0.5296 0.5058
    0.4928 0.4704 0.4513 0.4948 0.4986 0.5014 0.4993 0.5007 0.4942 0.5487 0.4956 0.5044
    0.5407 0.4949 0.5134 0.5012 0.5302 0.5431 0.5029 0.4971 0.4704 0.4698 0.5230 0.5296
    0.5017 0.4983 0.5051 0.4593 0.4988 0.4866 0.4569 0.4770 0.4809 0.4919 0.4939 0.5080
    0.4948 0.4988 0.5458 0.5061 0.5081 0.5041 0.5092 0.4908 0.4885 0.5115 0.4920 0.5052
    0.5034 0.4966 0.5012 0.5191 0.4959 0.4542 0.5191 0.4914 0.5081 0.4919 0.4762 0.4928
    0.5047 0.4953 0.4971 0.5029 0.4958 0.5042 0.5072 0.5238 0.5086 0.5155 0.4845 0.4902
    0.5098 0.4809 0.5182 0.4818 0.4934 0.5066 0.5000 0.5051 0.5238 0.4934 0.5238 0.4981
    0.4572 0.4959 0.5019 0.5428 0.5066 0.4762 0.5487 0.5635 0.4949 0.4762 0.4365 0.5000
    0.5041 0.4513 0.5044 0.4956 0.4919 0.5081 0.5343 0.4657 0.4566 0.5019 0.4452 0.4885
    0.4981 0.4942 0.5115 0.4885 0.5115 0.4762 0.5238 0.5548 0.5076 0.5434 0.5058 0.4924
    0.5046 0.5086 0.5290 0.4710 0.4685 0.4817 0.4762 0.4569 0.5183 0.4594 0.4908 0.5092
    0.4868 0.5132 0.4914 0.5238 0.4762 0.5238 0.5406 0.4954 0.5431 0.5315 0.5033 0.4967
    0.5000 0.4762 0.4657 0.5343 0.4542 0.4809 0.5138 0.4862 0.4980 0.5020 0.5132 0.4868
    0.4983 0.5017 0.5191 0.5098 0.4902 0.5000 0.5238 0.5458 0.4954 0.5081 0.5009 0.4626
    0.5125 0.4875 0.4953 0.5047 0.4919 0.5115 0.5020 0.4980 0.4770 0.4991 0.4885 0.4365
    0.5014 0.4986 0.5635 0.5046 0.5374 0.5230 0.5066 0.4934 0.4991 0.5238 0.4866 0.5183
    0.5102 0.4898 0.4817 0.5419 0.5042 0.4958 0.5792 0.5009 0.4581 0.4208 0.4966 0.5034
    0.4924 0.5134 0.4762 0.5076 0.4967 0.5033 0.5061 0.5374 0.5315 0.5066 0.4698 0.4939
    0.5548 0.4581 0.4588 0.5302 0.4934 0.5412 0.5419 0.4845 0.5155 0.4452 0.5007 0.4993
    0.4626 0.4685 0.5007 0.4993 0.4593 0.4920 0.4875 0.5125 0.5434 0.5191 0.5428 0.5406
    0.5412 0.4572 0.4208 0.4588 0.5080 0.5792 0.4809 0.5407 0.4594 0.4566 0.4818 0.5182
fval =
  874.6040
exitflag =
     1
output =
         iterations: 1
    constrviolation: -0.4208
          algorithm: 'medium-scale: active-set'
      firstorderopt: []
       cgiterations: []
            message: 'Optimization terminated.'
lambda =
      lower: [264x1 double]
      upper: [264x1 double]
      eqlin: [198x1 double]
    ineqlin: [0x1 double]
t = 0.919980332695915 segundos
```

8.3. Resultados de la Relajación PNLC

El problema original admite una formulación cuadrática entera binaria. La función objetivo es una función cuadrática en función de variables restringidas al conjunto de los enteros y limitadas a dos valores posibles. Las restricciones del problema original quedaron plasmadas en un sistema de ecuaciones lineales sobre estas variables.
Una aproximación, la menos afortunada en cuanto al número de iteraciones necesarias para obtener una solución óptima, se logra mediante la relajación de programación no lineal con una restricción no lineal, la cual consiste en sustituir el sistema de restricciones lineales por una única ecuación cuadrática compatible con las soluciones factibles del problema. Utilizando como herramienta el código fmincon.m, se logra el siguiente resultado donde, dentro de cada iteración, se distingue el total de número de veces que se evalúa la función objetivo (fcount).

n	Iteraciones	fcount	Tiempo (aprox.)	fmin
2	32	857	0,74218 seg.	58,2042
3	72	4571	6,6944 seg.	329,7404
4	97	11300	29,6740 seg.	497,0963
5	98	18174	107,8589 seg.	822,2018
6	98	26512	397,3562 seg.	1037,5574

Los minimizadores son vectores de tamaño $t \times s$. Se muestra los resultados completos dados por fmincon.m sólo para $n = 2$. El resultado es un vector de 24 componentes cuyos valores están en $[0, 1]$. Debido a la no pertinencia de esta relajación no mostramos los resultados para los demás valores de n.

Iter	F-count	f(x)	Max constraint	Line search steplength	Directional derivative	First-order optimality	Procedure
0	25	164	0				
1	54	156.641	0.04687	0.0625	-36.4	79.9	
2	82	145.307	0.1314	0.125	-41	80.1	
3	110	131.749	0.277	0.125	-38.6	72.6	
4	137	109.649	0.652	0.25	-43.1	55.5	
5	163	86.0473	1.773	0.5	-37.7	46.5	
6	190	78.7117	1.681	0.25	-23.3	31.9	
7	216	72.5201	1.525	0.5	-19.9	30.5	
8	242	71.1174	1.384	0.5	-16.8	27	
9	268	67.8048	1.092	0.5	-15.8	28.7	
10	295	65.7089	0.9828	0.25	-12.1	24.3	
11	321	66.0353	0.9006	0.5	-8.68	25.7	
12	346	63.8196	0.7692	1	-12.5	25.5	
13	372	63.7487	0.6896	0.5	-7.81	26.9	
14	398	62.8582	0.6488	0.5	-7.67	26.2	
15	424	61.6545	0.5461	0.5	-6.97	23.6	
16	450	61.4403	0.4763	0.5	-5.38	23.3	
17	475	59.1323	0.1373	1	-10.2	24.5	
18	503	59.0458	0.1341	0.125	-1.79	23.9	

```
   19    528      58.5938      0.05969           1        -3.91         23.1
   20    553      58.5089      0.04118           1        -2.24         23.2
   21    579      58.4148      0.03316         0.5        -1.95         23.2
   22    604      58.3024       0.0148           1        -2.05         23.1
   23    630      58.2724      0.01117         0.5       -0.973         23.2
   24    655      58.2825      0.01078           1        -0.79         23.2
   25    680      58.2549     0.008195           1        -1.08         23.1
   26    705      58.2366     0.003705           1       -0.992         23.1
   27    730      58.2285      0.00364           1        -0.73         23.2
   28    757      58.2241     0.003279        0.25       -0.451         23.1
   29    782      58.2076    0.0005779           1       -0.889         23.1
   30    807      58.2044   4.101e-005           1       -0.557         23.1
   31    832      58.2042   1.013e-006           1        -0.24         23.1  Hessi..
   32    857      58.2042   1.655e-007           1      -0.0151         23.1  Hessi..

Local minimum possible. Constraints satisfied.
fmincon stopped because the predicted change in the objective function
is less than the default value of the function tolerance and constraints
were satisfied to within the default value of the constraint tolerance.
<stopping criteria details>
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
              4
              5
              6
              9
             11
             13
             15
             21

     Mínimo x de la Relajación
Columns 1 through 9
0.0055   0.0457   0.1877   1.0000   1.0000   1.0000   0.5602   0.8386   1.0000
Columns 10 through 18
0.9485   1.0000   0.5270   1.0000   0.9450   1.0000   0.2752   0.3540   0.0995
Columns 19 through 24
0.1175   0.0546   1.0000   0.7183   0.2416   0.2705

   Distancia Mínima (Relajada)
           dmin = 58.2042014020948
t = 0.742186227579203 segundos
```

8.4. Resultados de la Relajación PNLL

El problema original admite una formulación cuadrática entera binaria. La función objetivo es una función cuadrática en función de variables restringidas al conjunto de los enteros y limitadas a dos valores posibles. Las restricciones del problema original quedaron plasmadas en un sistema de ecuaciones lineales sobre estas variables.
Se logra una aproximación mediante la relajación de programación no lineal con una restricción lineal, la cual consiste en sustituir el sistema de restricciones lineales por una única ecuación lineal compatible con las soluciones factibles del problema. Utilizando como herramienta el código quadprog.m, se logra el siguiente resultado:

n	Iteraciones	Tiempo (aprox.)	fmin
2	1	0,0504 seg.	34,7358
3	4	0,0388 seg.	191,4287
4	5	0,0613 seg.	299,2692
5	6	0,1429 seg.	483,6686
6	8	0,7842 seg.	610,0670

Los minimizadores son vectores de tamaño $t \times s$. Se muestra los resultados completos dados por quadprog.m sólo para $n = 2$. El resultado es un vector de 24 componentes cuyos valores están en $[0, 1]$.
Resultado del PNLL para $n = 2$:

```
Optimization terminated.
x =
    0.2313 0.3429 0.3033 0.6721 0.6756 0.6152 0.4272 0.5610
    0.6785 0.5866 0.6354 0.3830 0.6794 0.5907 0.8270 0.3544
    0.5317 0.2511 0.5004 0.2150 0.5906 0.5415 0.3386 0.4676
fval =
   34.7358
exitflag = 1
output =
         iterations: 1
    constrviolation: -0.1730
          algorithm: 'medium-scale: active-set'
      firstorderopt: []
       cgiterations: []
            message: 'Optimization terminated.'
lambda =
      lower: [24x1 double]
      upper: [24x1 double]
      eqlin: -5.7893
    ineqlin: [0x1 double]
t = 0.00433895928113769 segundos
```

Resultado del PNLL para $n = 3$:

```
Optimization terminated.
x =
    0.2589 0.2794 0.3769 0.7605 0.7618 0.4016 0.3925 0.4401 0.4355 0.3166
    0.3887 0.5014 0.4657 0.4616 0.3309 0.3452 0.6027 0.4264 0.4465 0.4016
    0.3868 0.5353 0.5568 0.5744 0.6001 0.6416 0.6890 0.6303 0.5147 0.3670
    0.3296 0.3833 0.3363 0.3516 0.3949 1.0000 1.0000 0.8257 0.7626 0.3555
    0.4776 0.5962 0.5436 1.0000 0.8844 0.7467 0.7513 0.2704 0.2661 0.2427
    0.4898 0.6206 0.6899 0.4728 0.3028 0.5063 0.3160 0.2933 0.3215 0.1779
fval =
  191.4287
exitflag =
     1
output =
         iterations: 4
    constrviolation: 0
          algorithm: 'medium-scale: active-set'
      firstorderopt: []
       cgiterations: []
            message: 'Optimization terminated.'
lambda =
      lower: [60x1 double]
      upper: [60x1 double]
      eqlin: -13.1074
    ineqlin: [0x1 double]
t = 0.0388135223890187 segundos
```

Resultado del PNLL para $n = 4$:

```
Optimization terminated.
x =
    0.2179 0.2397 0.7183 0.7976 0.8087 0.7139 0.4381 0.3199 0.3512 0.3371 0.5135 0.5587
    0.3795 0.4288 0.1831 0.2106 0.5270 0.6216 0.5704 0.3407 0.3307 0.5829 0.8344 0.7757
    0.3004 0.4506 0.5112 0.2779 0.3078 0.4195 0.4187 0.5213 0.5664 0.4618 0.5086 0.6123
    0.6051 0.6375 0.6940 0.6033 0.4974 0.3640 0.2292 0.3149 0.3493 0.4168 0.4520 0.4838
    0.6018 0.6556 0.8669 0.7887 0.4897 0.4837 0.4647 0.3279 0.2792 0.5729 1.0000 0.9835
    0.7539 0.6761 0.6743 0.7074 0.3465 0.3268 0.4039 0.3400 0.3294 0.2136 0.4456 0.5905
    0.6995 0.6003 0.6408 0.6790 0.4050 0.4195 0.9381 1.0000 0.8739 0.7146 0.3098 0.2703
    0.1518 0.2471 0.1757 0.3188 0.3690 0.2373 0.4801 0.2116 0.7125 0.7524 0.3929 0.3942
    0.4096 0.5373 0.3221 0.3436 0.2537 0.4879 1.0000 1.0000 0.8069 0.6563 0.5418 0.5316
    0.2505 0.2776 0.2793 0.1812
fval =
  299.2692
exitflag =
     1
output =
         iterations: 5
    constrviolation: 0
```

```
          algorithm: 'medium-scale: active-set'
      firstorderopt: []
       cgiterations: []
            message: 'Optimization terminated.'
lambda =
      lower: [112x1 double]
      upper: [112x1 double]
      eqlin: -10.7890
    ineqlin: [0x1 double]
t = 0.0613157538178735 segundos
```

Resultado del PNLL para $n = 5$:

```
Optimization terminated.
x =
    0.3052 0.6285 0.8244 0.7973 0.8744 0.8258 0.4810 0.3538 0.3668 0.2264 0.7352 0.7166
    0.8510 0.8218 0.1990 0.3088 0.1671 0.1945 0.3595 0.4996 0.4482 0.5119 0.4089 0.3840
    0.6079 0.5956 0.6772 0.4549 0.5023 0.5637 0.2704 0.2694 0.5497 0.4961 0.2928 0.2818
    0.2642 0.3549 0.4085 0.3966 0.4566 0.4534 0.4767 0.5558 0.6132 0.5749 0.5705 0.5657
    0.5364 0.5912 0.6872 0.6605 0.5195 0.3625 0.3659 0.4046 0.5383 0.7065 0.5824 0.5376
    0.8902 0.9316 0.5383 0.4680 0.3700 0.3819 0.4574 0.6273 0.7273 0.7227 0.6540 0.4460
    0.2792 0.5180 0.8661 0.7415 0.1675 0.3359 0.5354 0.5407 0.5325 0.4687 0.7257 0.6978
    0.5468 0.4397 0.8047 0.8378 0.3543 0.1554 0.2875 0.2465 0.2517 0.1479 0.4723 0.3129
    0.2049 0.4606 0.6991 0.7016 0.6334 0.6588 0.3537 0.4602 0.4868 0.6159 0.4904 0.3321
    0.3473 0.5704 0.2488 0.2950 0.4100 0.4573 0.6285 0.4903 0.4635 0.5007 0.3817 0.4521
    0.4422 0.3033 0.2725 0.2573 0.4178 0.3010 0.2039 0.2601 0.1344 0.2419 0.2312 0.1533
    0.3046 0.3265 0.4366 0.6359 0.9973 1.0000 0.6698 0.4814 0.7341 0.7105 0.3722 0.2742
    0.2224 0.2713 0.7082 0.7388 0.4565 0.3658 0.4657 0.5535 0.5270 0.6444 0.8875 0.8331
    1.0000 1.0000 0.4247 0.4674 0.4626 0.3095 0.3491 0.4545 0.8061 0.9237 1.0000 1.0000
    0.5601 0.3600 0.3660 0.5293 0.5027 0.4812 0.3426 0.4251 0.5168 0.3465 0.3629 0.3130
fval =
  483.6686
exitflag =
     1
output =
         iterations: 6
    constrviolation: 0
          algorithm: 'medium-scale: active-set'
      firstorderopt: []
       cgiterations: []
            message: 'Optimization terminated.'
lambda =
      lower: [180x1 double]
      upper: [180x1 double]
      eqlin: -10.9440
    ineqlin: [0x1 double]
t = 0.142922424498086 segundos
```

Resultado del PNLL para $n = 6$:

```
Optimization terminated.
x =
    0.7038 0.7361 0.6769 0.6626 0.1320 0.2022 0.8058 0.9363 0.8764 0.8243 0.4264 0.3172
    0.1935 0.2175 0.2676 0.1090 0.7927 0.9061 1.0000 1.0000 0.4459 0.2845 0.4202 0.5293
    0.7267 0.3314 0.4903 0.1566 0.5119 0.3875 0.3348 0.3680 0.5186 0.6971 0.6399 0.6156
    0.8281 0.8098 0.3983 0.5662 0.1477 0.4026 0.2901 0.3141 0.1773 0.3216 0.4001 0.4381
    0.4816 0.4085 0.4329 0.4404 0.4734 0.5207 0.6427 0.6555 0.6938 0.7267 0.6496 0.7018
    0.6823 0.6150 0.4057 0.2626 0.2343 0.1741 0.1213 0.2821 0.4657 0.4948 0.4622 0.6859
    0.9556 1.0000 0.8275 0.8597 0.9314 0.8749 0.7890 0.6140 0.6171 0.7861 0.6879 0.6400
    0.5579 0.1965 0.3147 0.2714 0.4447 0.5567 0.5315 0.3222 0.2568 0.4305 0.5429 0.3349
    0.3602 0.1636 0.3177 0.4456 0.4294 0.2814 0.5751 0.5292 0.3641 0.5149 0.4147 0.3795
    0.1480 0.3070 0.3647 0.4184 0.4150 0.4817 0.9462 0.8873 0.5661 0.3398 0.6029 0.5103
    0.3565 0.4170 0.2417 0.2252 0.4358 0.5650 0.6379 0.7827 0.5478 0.7158 0.5320 0.3640
    0.3176 0.3084 0.9217 0.9572 0.7160 0.3534 0.3651 0.3428 0.4296 0.4096 0.2384 0.2502
    0.5671 0.6295 0.3286 0.4835 0.4410 0.4220 0.4283 0.4779 0.4398 0.5027 0.3771 0.4368
    0.3726 0.4264 0.6979 0.6971 0.1705 0.5481 0.9275 0.9729 1.0000 0.9703 0.6534 0.6637
    0.7366 0.7510 0.5213 0.3791 0.3638 0.2607 0.3515 0.1054 0.2252 0.1391 0.3417 0.5022
    0.8228 0.7591 0.3534 0.3579 0.1485 0.5055 0.9511 0.9210 0.5238 0.4188 0.3741 0.4176
    0.9813 1.0000 0.5592 0.5208 0.5999 0.3965 0.4112 0.4898 0.4972 0.5553 0.5225 0.4941
    1.0000 1.0000 0.5897 0.5720 0.3991 0.3637 0.7332 0.5025 0.4607 0.5867 0.1978 0.2123
    0.2924 0.4179 0.4639 0.3094 0.3436 0.3698 0.1417 0.3462 0.4577 0.2810 0.3138 0.1250
    0.4524 0.4557 0.5510 0.4698 0.2690 0.4818 0.4353 0.3600 0.3426 0.4243 0.8616 0.9476
    0.5305 0.3716 0.6878 0.7042 0.3817 0.1280 0.7898 0.8477 0.5609 0.5020 0.4278 0.4731
    0.5097 0.3198 0.5440 0.3344 0.1385 0.5993 0.4170 0.4814 0.2990 0.3947 0.2854 0.2568
fval =
  610.0670
exitflag =
     1
output =
         iterations: 8
    constrviolation: 0
          algorithm: 'medium-scale: active-set'
      firstorderopt: []
       cgiterations: []
            message: 'Optimization terminated.'
lambda =
      lower: [264x1 double]
      upper: [264x1 double]
      eqlin: -9.3332
    ineqlin: [0x1 double]
t = 0.784234937680627 segundos
```

8.5. Resultados de la Ramificación y Acotamiento

8.5.1. Caso QP

Resolución del programa cuadrático entero binario mediante el algoritmo de ramificación y acotamiento utilizando en cada iteración la relajación programación cuadrática y utilizando como resolvente la función quadprog.m. Se muestra en la última columna el error relativo del mínimo aproximado, fmin, con respecto al valor exacto, dmin, obtenido por enumeración. Se obtuvo un resultado satisfactorio únicamente para $n = 2$.

n	Iteraciones	Tiempo (aprox.)	fmin	discrepancia(%)
2	35	3,9155 seg.	108	0

Se muestra los resultados sólo para $n = 2$ ya que fue el único que produjo un resultado completo sin la ayuda de un punto inicial diferente al predeterminado $x0 = 0$.
Resultados del miqp.m (caso QP) para n=2:

```
naplic = 1 si es original -- naplic = 2 si es modificado.
naplic = 1

Extendedflag =
    QPiter: 35
     optQP: 26
      time: 3.6847

xmin =
     1     1     0     0     0     1
     0     1     1     1     0     0
     1     0     0     0     1     1
     0     0     1     1     1     0

     ASIGNACIÓN DE MÍNIMA DISTANCIA
    'A'    'A'    'H'    'H'    'H'    'A'
    'H'    'A'    'A'    'A'    'H'    'H'
    'A'    'H'    'H'    'H'    'A'    'A'
    'H'    'H'    'A'    'A'    'A'    'H'

fmin =108

t = 3.91555707499137 segundos
```

8.5.2. Caso PNLL

Resolución del programa cuadrático entero binario mediante el algoritmo de ramificación y acotamiento utilizando en cada iteración la relajación programación no lineal con una restricción lineal y utilizando como resolvente la función quadprog.m. Se muestra en la última columna el error relativo del mínimo aproximado, fmin, con respecto al valor exacto, dmin, obtenido por enumeración. Los resultados fueron los siguientes:

n	Iteraciones	Tiempo (aprox.)	fmin	discrepancia(%)
2	63	0,2469 seg.	108	0
3	27955	57,2636 seg.	980,96	0

Se muestra los resultados sólo para $n = 2$ y $n = 3$, ya que fueron los únicos que produjeron un resultado satisfactorio.
Resultados del miqp.m (caso PNLL) para n=2:

```
naplic = 1 si es original -- naplic = 2 si es modificado.
naplic = 2

Extendedflag =
    QPiter: 63
     optQP: 54
      time: 0.2444

xmin =
     1     1     0     0     0     1
     0     1     1     1     0     0
     1     0     0     0     1     1
     0     0     1     1     1     0

     ASIGNACIÓN DE MÍNIMA DISTANCIA
    'A'    'A'    'H'    'H'    'H'    'A'
    'H'    'A'    'A'    'A'    'H'    'H'
    'A'    'H'    'H'    'H'    'A'    'A'
    'H'    'H'    'A'    'A'    'A'    'H'

fmin =108

t = 0.246942558341912 segundos
```

Resultados del miqp.m (caso PNLL) para n=3:

```
naplic = 1 si es original -- naplic = 2 si es modificado.
naplic = 2

Extendedflag =
    QPiter: 27955
     optQP: 17739
      time: 57.2539

xmin =
     0     1     1     0     1     0     0     1     0     1
     1     0     0     0     0     1     1     1     1     0
     1     0     0     1     1     0     0     1     1     0
     0     1     1     0     0     1     0     0     1     1
     0     1     1     1     0     0     1     0     0     1
     1     0     0     1     1     1     1     0     0     0

     ASIGNACIÓN DE MÍNIMA DISTANCIA
    'H'    'A'    'A'    'H'    'A'    'H'    'H'    'A'    'H'    'A'
    'A'    'H'    'H'    'H'    'H'    'A'    'A'    'A'    'A'    'H'
    'A'    'H'    'H'    'A'    'A'    'H'    'H'    'A'    'A'    'H'
    'H'    'A'    'A'    'H'    'H'    'A'    'H'    'H'    'A'    'A'
    'H'    'A'    'A'    'A'    'H'    'H'    'A'    'H'    'H'    'A'
    'A'    'H'    'H'    'A'    'A'    'A'    'A'    'H'    'H'    'H'

fmin =980.96

t = 57.2636301541118 segundos
```

8.5.3. Caso Heurística Codiciosa

Resolución del programa cuadrático entero binario mediante el algoritmo de ramificación y acotamiento, utilizando una heurística codiciosa para la ramificación y una variante del código MIQP.m de Bemporad [1]. La tabla de resultados muestra, en la última columna, el error relativo del mínimo aproximado, fmin, con respecto al valor exacto, dmin, obtenido por enumeración. Los resultados fueron los siguientes: Se muestran todos los resultados.

n	Iteraciones	Tiempo (aprox.)	fmin	discrepancia(%)
2	25	0,0382 seg.	108	0
3	92	0,1439 seg.	1001,45	2,089
4	147	0,5490 seg.	1634,43	5,915
5	121	1,2375 seg.	2056,728	(−0,566)
6	10430	76,9195 seg.	2870,76	(−0,566)

Resultados de la ramificación codiciosa para $n = 2$:

```
QPiter = 25

xmin =
     1     0     0     0     1     1
     0     1     1     1     0     0
     1     1     0     0     0     1
     0     0     1     1     1     0

     ASIGNACIÓN DE MÍNIMA DISTANCIA
    'A'    'H'    'H'    'H'    'A'    'A'
    'H'    'A'    'A'    'A'    'H'    'H'
    'A'    'A'    'H'    'H'    'H'    'A'
    'H'    'H'    'A'    'A'    'A'    'H'

fmin =108

dmin =
   108

t = 0.0381621826237692 segundos
```

Resultados de la ramificación codiciosa para $n = 3$:

```
QPiter = 92
xmin =
     0     1     1     1     0     0     0     0     1     1
     1     0     0     0     0     1     1     1     1     0
     1     0     0     1     1     0     0     1     1     0
     0     1     1     0     0     0     1     1     0     1
     0     1     1     0     1     1     0     0     0     1
     1     0     0     1     1     1     1     0     0     0
     ASIGNACIÓN DE MÍNIMA DISTANCIA
    'H'    'A'    'A'    'A'    'H'    'H'    'H'    'H'    'A'    'A'
    'A'    'H'    'H'    'H'    'H'    'A'    'A'    'A'    'A'    'H'
    'A'    'H'    'H'    'A'    'A'    'H'    'H'    'A'    'A'    'H'
    'H'    'A'    'A'    'H'    'H'    'H'    'A'    'A'    'H'    'A'
    'H'    'A'    'A'    'H'    'A'    'A'    'H'    'H'    'H'    'A'
    'A'    'H'    'H'    'A'    'A'    'A'    'A'    'H'    'H'    'H'
fmin =1001.45
dmin =
  1.0014e+003
t = 0.143939523039939 segundos
```

Resultados de la ramificación codiciosa para $n = 4$:

```
QPiter = 147
xmin =
     1     0     0     1     1     0     0     0     0     1     0     1     1     1
     0     0     0     1     1     1     0     1     1     0     1     0     0     1
     0     1     1     1     0     0     1     0     0     1     0     1     0     1
     1     1     0     0     1     0     0     1     1     0     0     0     1     1
     1     1     1     0     1     0     0     1     1     0     0     1     0     0
     0     1     1     0     0     1     1     0     0     1     1     0     1     0
     0     0     0     0     0     1     1     1     1     0     1     1     1     0
     1     0     1     1     0     1     1     0     0     1     1     0     0     0
     ASIGNACIÓN DE MÍNIMA DISTANCIA
    'A'   'H'   'H'   'A'   'A'   'H'   'H'   'H'   'H'   'A'   'H'   'A'   'A'   'A'
    'H'   'H'   'H'   'A'   'A'   'A'   'H'   'A'   'A'   'H'   'A'   'H'   'H'   'A'
    'H'   'A'   'A'   'A'   'H'   'H'   'A'   'H'   'H'   'A'   'H'   'A'   'H'   'A'
    'A'   'A'   'H'   'H'   'A'   'H'   'H'   'A'   'A'   'H'   'H'   'H'   'A'   'A'
    'A'   'A'   'A'   'H'   'A'   'H'   'H'   'A'   'A'   'H'   'H'   'A'   'H'   'H'
    'H'   'A'   'A'   'H'   'H'   'A'   'A'   'H'   'H'   'A'   'A'   'H'   'A'   'H'
    'H'   'H'   'H'   'H'   'H'   'A'   'A'   'A'   'A'   'H'   'A'   'A'   'A'   'H'
    'A'   'H'   'A'   'A'   'H'   'A'   'A'   'H'   'H'   'A'   'A'   'H'   'H'   'H'
fmin =1634.43
dmin =
  1.6344e+003
t = 0.548954755422826 segundos
```

Resultados de la ramificación codiciosa para $n = 5$:

```
QPiter = 121

xmin =
     0   0   0   1   0   1   1   1   1   1   1   0   0   1   0   0   0   1
     1   1   1   1   1   0   0   0   0   0   0   0   0   1   1   1   1   0
     1   0   0   0   0   1   1   1   0   0   0   1   1   0   0   1   1   1
     1   1   1   0   0   1   1   0   0   1   1   0   0   0   1   1   0   0
     1   1   1   0   1   0   0   1   1   1   1   0   0   0   1   0   0   0
     0   0   0   1   0   0   0   0   0   0   1   1   1   1   1   1   1   1
     1   1   0   0   1   1   0   0   0   1   1   0   0   1   1   0   1   0
     0   0   1   1   0   0   1   1   1   0   0   1   1   0   0   1   0   1
     0   1   1   0   1   1   1   1   1   0   0   1   1   0   0   0   0   0
     0   0   0   1   1   0   0   0   1   1   0   1   1   1   0   0   1   1

     ASIGNACIÓN DE MÍNIMA DISTANCIA
  'H' 'H' 'H' 'A' 'H' 'A' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'H' 'H' 'H' 'A'
  'A' 'A' 'A' 'A' 'A' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'A' 'A' 'A' 'A' 'H'
  'A' 'H' 'H' 'H' 'H' 'A' 'A' 'A' 'H' 'H' 'H' 'A' 'A' 'H' 'H' 'A' 'A' 'A'
  'A' 'A' 'A' 'H' 'H' 'A' 'A' 'H' 'H' 'A' 'A' 'H' 'H' 'H' 'A' 'A' 'H' 'H'
  'A' 'A' 'A' 'H' 'A' 'H' 'H' 'A' 'A' 'A' 'A' 'H' 'H' 'H' 'A' 'H' 'H' 'H'
  'H' 'H' 'H' 'A' 'H' 'H' 'H' 'H' 'H' 'H' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
  'A' 'A' 'H' 'H' 'A' 'A' 'H' 'H' 'H' 'A' 'A' 'H' 'H' 'A' 'A' 'H' 'A' 'H'
  'H' 'H' 'A' 'A' 'H' 'H' 'A' 'A' 'A' 'H' 'H' 'A' 'A' 'H' 'H' 'A' 'H' 'A'
  'H' 'A' 'A' 'H' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'A' 'H' 'H' 'H' 'H' 'H'
  'H' 'H' 'H' 'A' 'A' 'H' 'H' 'H' 'A' 'A' 'H' 'A' 'A' 'A' 'H' 'H' 'A' 'A'

fmin =2056.728

dmin =
  2.0567e+003

t = 1.2374774587273 segundos
```

Resultados de la ramificación codiciosa para $n = 6$:

```
QPiter = 10430

xmin =
     1  0  0  1  1  1  0  1  1  0  1  0  0  0  0  0  0  1  0  1  1  1
     1  0  0  0  0  0  0  0  0  1  0  1  1  1  1  0  1  1  1  1  1  0
     1  0  0  0  0  1  1  1  1  1  1  0  0  1  1  1  1  0  0  0  0  0
     0  0  1  0  0  0  0  1  1  0  0  1  1  1  1  1  0  0  1  1  0  1
     1  0  0  1  1  0  0  1  1  0  0  0  1  0  1  0  0  1  1  1  1  0
     0  1  0  1  1  0  1  0  1  1  1  1  1  0  0  1  0  0  0  0  0  1
     1  1  1  0  0  0  1  1  1  1  0  1  1  0  0  0  1  0  0  0  0  1
     0  1  1  0  0  1  0  0  0  1  1  0  0  1  0  1  1  1  0  0  1  1
     0  1  0  0  0  1  1  0  0  0  0  1  0  1  1  1  1  0  0  1  1  1
     0  1  1  1  1  1  1  0  0  0  1  0  0  0  1  1  0  1  1  0  0  0
     1  0  1  1  1  1  1  0  0  1  1  0  0  0  0  0  1  1  1  0  0  0
     0  1  1  1  1  0  0  1  0  0  0  1  1  1  0  0  0  0  1  1  1  0

     ASIGNACIÓN DE MÍNIMA DISTANCIA
'A' 'H' 'H' 'A' 'A' 'A' 'H' 'A' 'A' 'H' 'A' 'H' 'H' 'H' 'H' 'H' 'H' 'A' 'H' 'A' 'A' 'A'
'A' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'A' 'H' 'A' 'A' 'A' 'A' 'H' 'A' 'A' 'A' 'A' 'A' 'H'
'A' 'H' 'H' 'H' 'H' 'A' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'A' 'A' 'A' 'H' 'H' 'H' 'H' 'H'
'H' 'H' 'A' 'H' 'H' 'H' 'H' 'A' 'A' 'H' 'H' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'A' 'H' 'A'
'A' 'H' 'H' 'A' 'A' 'H' 'H' 'A' 'A' 'H' 'H' 'H' 'A' 'H' 'A' 'H' 'H' 'A' 'A' 'A' 'A' 'H'
'H' 'A' 'H' 'A' 'A' 'H' 'A' 'H' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'H' 'H' 'H' 'H' 'H' 'A'
'A' 'A' 'A' 'H' 'H' 'H' 'A' 'A' 'A' 'A' 'H' 'A' 'A' 'H' 'H' 'H' 'A' 'H' 'H' 'H' 'H' 'A'
'H' 'A' 'A' 'H' 'H' 'A' 'H' 'H' 'H' 'A' 'A' 'H' 'H' 'A' 'H' 'A' 'A' 'A' 'H' 'H' 'A' 'A'
'H' 'A' 'H' 'H' 'H' 'A' 'A' 'H' 'H' 'H' 'H' 'A' 'H' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'A' 'A'
'H' 'A' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'H' 'A' 'H' 'H' 'H' 'A' 'A' 'H' 'A' 'A' 'H' 'H' 'H'
'A' 'H' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'A' 'H' 'H' 'H' 'H' 'H' 'A' 'A' 'A' 'H' 'H' 'H'
'H' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'H' 'H' 'H' 'A' 'A' 'A' 'H' 'H' 'H' 'H' 'A' 'A' 'A' 'H'

fmin =2870.76

dmin =
  2.8708e+003

t = 76.9195496215301 segundos
```

8.6. Resultados de la Relajación Semidefinida y la Aproximación Aleatoria

Resolución del programa cuadrático entero binario mediante la relajación programa semidefinido utilizando la función matlab sqlp.m, la cual da como resultado una matriz definida positiva a la que se aplica el algoritmo de aproximación aleatoria de Goemans y Williamson. Para todos los casos se utilizaron 100 aplicaciones del algoritmo aleatorio, es decir, aleatoriamente se generaron 100 hiperplanos, los cuales, uno cada vez, separaron los dos grupos de vectores de $\mathbb{R}^m$, relajación de sendos tipos de soluciones del problema de asignación local visitante. La tabla de resultados muestra, en la última columna, el error relativo del mínimo aproximado, fmin, con respecto al valor exacto, dmin, obtenido por enumeración. Los resultados fueron los siguientes:

n	Iteraciones	Tiempo (aprox.)	fmin	discrepancia(%)
2	21	1,1541 seg.	108	0
3	22	1,3953 seg.	980,96	0
4	20	1,8591 seg.	1543,15	0
5	19	3,3247 seg.	2007,187	(−2,961)
6	19	6,61062 seg.	2738,69	(−5,131)

Mostramos aquí la salida completa del código máster semidhan2.m para $n = 2$. Puede verse la salida del código máster sqlp.m del software SDPT3-01 para el programa semidefinido. Luego la salida del semidefhan2.m, esta es la matriz definida positiva minimizadora, sus valores propios (positivos), la matriz resultante de la descomposición de Cholesky, la norma de los vectores columna (deben ser iguales a 1) y los resultados definitivos en el formato del problema de asignación local visitante.

```
Resultados del programa semidefinido y aproximación aleatoria para $n=2$:
num. of constraints = 43
 dim. of sdp    var  = 25,   num. of sdp  blk  =  1
*****************************************************************
   SDPT3: Infeasible path-following algorithms
*****************************************************************
 version  predcorr  gam  expon  scale_data
   HKM      1      0.000   1        0
it pstep dstep pinfeas dinfeas  gap      prim-obj      dual-obj    cputime
-----------------------------------------------------------------
 0|0.000|0.000|1.4e+001|4.6e+000|3.8e+004|3.619191e+003 0.000000e+000|0:0:00|chol 1  1
 1|0.953|1.000|6.5e-001|1.0e-002|2.8e+003|2.727883e+002-9.589592e+002|0:0:00|chol 1  1
 2|0.830|1.000|1.1e-001|1.0e-003|3.7e+002|1.402133e+002 2.088885e+001|0:0:00|chol 1  1
 3|0.846|1.000|1.7e-002|1.0e-004|7.5e+001|1.143879e+002 8.967464e+001|0:0:00|chol 1  1
 4|0.835|0.944|2.8e-003|3.4e-003|1.7e+001|1.098992e+002 1.047785e+002|0:0:00|chol 1  1
 5|0.740|1.000|7.3e-004|5.7e-004|5.9e+000|1.085327e+002 1.069152e+002|0:0:00|chol 1  1
 6|0.818|0.951|1.3e-004|1.7e-004|1.2e+000|1.078498e+002 1.076774e+002|0:0:00|chol 2  2
 7|0.529|0.566|6.3e-005|1.0e-004|8.0e-001|1.079282e+002 1.078072e+002|0:0:00|chol 2  2
 8|0.640|0.713|2.3e-005|4.2e-005|4.4e-001|1.079468e+002 1.078795e+002|0:0:00|chol 2  2
 9|0.684|0.598|7.3e-006|2.1e-005|2.2e-001|1.079582e+002 1.079207e+002|0:0:00|chol 3  3
10|0.484|0.564|3.8e-006|1.1e-005|1.5e-001|1.079652e+002 1.079489e+002|0:0:00|chol 3  3
11|0.393|1.000|2.3e-006|7.6e-007|1.4e-001|1.079775e+002 1.079713e+002|0:0:00|chol 3  3
12|0.686|1.000|9.0e-007|4.6e-007|6.9e-002|1.079732e+002 1.079758e+002|0:0:00|chol 3  3
13|0.833|0.873|2.7e-007|2.4e-007|1.9e-002|1.079757e+002 1.079834e+002|0:0:00|chol 3  4
14|0.982|1.000|2.0e-007|5.4e-008|9.6e-003|1.079771e+002 1.079863e+002|0:0:00|chol 5  5
15|1.000|0.994|1.2e-007|4.0e-008|4.6e-003|1.079801e+002 1.079896e+002|0:0:00|chol 7  7
16|0.665|1.000|8.9e-008|2.4e-008|3.0e-003|1.079830e+002 1.079904e+002|0:0:00|chol10 10
17|0.822|1.000|6.0e-008|1.8e-008|1.8e-003|1.079852e+002 1.079925e+002|0:0:00|chol
  warning: symqmr failed: 0.3
  switch to LU factor. lu 30   3
18|0.732|0.979|4.2e-008|1.2e-008|1.4e-003|1.079875e+002 1.079937e+002|0:0:00|lu 30   8
19|0.573|1.000|4.2e-008|8.3e-009|1.2e-003|1.079887e+002 1.079942e+002|0:0:00|lu 27 ^19
20|0.065|0.235|4.4e-008|1.5e-008|1.4e-003|1.079891e+002 1.079946e+002|0:0:00|lu 20 ^10
21|0.027|0.032|4.1e-008|2.3e-008|1.4e-003|1.079893e+002 1.079947e+002|0:0:00|
  stop: xxx progress is bad
  lack of progress in infeas
-----------------------------------------------------------------
 number of iterations   = 21
 primal objective value =  1.07987522e+002
 dual   objective value =  1.07993654e+002
 gap := trace(XZ)       = 1.36e-003
 relative gap           = 6.25e-006
 actual relative gap    = -2.83e-005
 rel. primal infeas     = 4.16e-008
 rel. dual   infeas     = 1.23e-008
 norm(X), norm(y), norm(Z) = 1.8e+001, 2.2e+004, 2.5e+004
 norm(A), norm(b), norm(C) = 8.8e+000, 1.1e+001, 4.9e+001
```

```
 Total CPU time (secs)  = 0.5
 CPU time per iteration = 0.0
 termination code       = -5
 DIMACS: 1.5e-007  0.0e+000  1.4e-008  0.0e+000  -2.8e-005  6.2e-006
------------------------------------------------------------------
  Columns 1 through 9
 1.0000    0.0007   -1.0000   -0.9999   -0.0012    0.9999   -1.0000   -0.0056    1.0000
 0.0007    1.0000   -0.0005   -0.0001   -1.0000   -0.0002   -0.0005    0.0010    0.0005
-1.0000   -0.0005    1.0000    0.9999    0.0010   -0.9999    1.0000    0.0056   -1.0000
-0.9999   -0.0001    0.9999    1.0000    0.0006   -1.0000    0.9999    0.0045   -0.9999
-0.0012   -1.0000    0.0010    0.0006    1.0000   -0.0003    0.0010   -0.0010   -0.0010
 0.9999   -0.0002   -0.9999   -1.0000   -0.0003    1.0000   -0.9999   -0.0045    0.9999
-1.0000   -0.0005    1.0000    0.9999    0.0010   -0.9999    1.0000    0.0056   -1.0000
-0.0056    0.0010    0.0056    0.0045   -0.0010   -0.0045    0.0056    1.0000   -0.0052
 1.0000    0.0005   -1.0000   -0.9999   -0.0010    0.9999   -1.0000   -0.0052    1.0000
 1.0000    0.0006   -1.0000   -0.9999   -0.0011    0.9999   -1.0000   -0.0057    1.0000
 0.0052   -0.0010   -0.0052   -0.0041    0.0010    0.0041   -0.0052   -1.0000    0.0048
-1.0000   -0.0006    1.0000    0.9999    0.0011   -0.9999    1.0000    0.0053   -1.0000
 0.9999    0.0000   -0.9999   -0.9999   -0.0005    0.9999   -0.9999   -0.0049    0.9999
-0.0016   -1.0000    0.0014    0.0010    1.0000   -0.0007    0.0014   -0.0010   -0.0014
-0.9999   -0.0003    0.9999    0.9999    0.0008   -0.9999    0.9999    0.0049   -0.9999
-1.0000   -0.0006    1.0000    0.9999    0.0011   -0.9999    1.0000    0.0053   -1.0000
 0.0013    1.0000   -0.0011   -0.0008   -1.0000    0.0005   -0.0011    0.0010    0.0011
 1.0000    0.0009   -1.0000   -0.9999   -0.0014    0.9999   -1.0000   -0.0053    1.0000
-0.9999   -0.0003    0.9999    0.9999    0.0008   -0.9999    0.9999    0.0049   -0.9999
 0.0048   -0.0010   -0.0048   -0.0037    0.0010    0.0037   -0.0048   -1.0000    0.0044
 0.9999    0.0003   -0.9999   -0.9999   -0.0008    0.9999   -0.9999   -0.0050    0.9999
 0.9999   -0.0002   -0.9999   -1.0000   -0.0003    1.0000   -0.9999   -0.0045    0.9999
-0.0053    0.0010    0.0053    0.0042   -0.0010   -0.0042    0.0053    1.0000   -0.0049
-0.9999    0.0002    0.9999    1.0000    0.0003   -1.0000    0.9999    0.0048   -0.9999
-1.0000    0.0002    1.0000    0.9998    0.0002   -0.9998    1.0000    0.0056   -1.0000
  Columns 10 through 18
 1.0000    0.0052   -1.0000    0.9999   -0.0016   -0.9999   -1.0000    0.0013    1.0000
 0.0006   -0.0010   -0.0006    0.0000   -1.0000   -0.0003   -0.0006    1.0000    0.0009
-1.0000   -0.0052    1.0000   -0.9999    0.0014    0.9999    1.0000   -0.0011   -1.0000
-0.9999   -0.0041    0.9999   -0.9999    0.0010    0.9999    0.9999   -0.0008   -0.9999
-0.0011    0.0010    0.0011   -0.0005    1.0000    0.0008    0.0011   -1.0000   -0.0014
 0.9999    0.0041   -0.9999    0.9999   -0.0007   -0.9999   -0.9999    0.0005    0.9999
-1.0000   -0.0052    1.0000   -0.9999    0.0014    0.9999    1.0000   -0.0011   -1.0000
-0.0057   -1.0000    0.0053   -0.0049   -0.0010    0.0049    0.0053    0.0010   -0.0053
 1.0000    0.0048   -1.0000    0.9999   -0.0014   -0.9999   -1.0000    0.0011    1.0000
 1.0000    0.0053   -1.0000    0.9999   -0.0015   -0.9999   -1.0000    0.0013    1.0000
 0.0053    1.0000   -0.0049    0.0045    0.0010   -0.0045   -0.0049   -0.0010    0.0048
-1.0000   -0.0049    1.0000   -0.9999    0.0015    0.9999    1.0000   -0.0013   -1.0000
 0.9999    0.0045   -0.9999    1.0000   -0.0009   -1.0000   -0.9999    0.0007    0.9999
-0.0015    0.0010    0.0015   -0.0009    1.0000    0.0012    0.0015   -1.0000   -0.0018
-0.9999   -0.0045    0.9999   -1.0000    0.0012    1.0000    0.9999   -0.0010   -0.9999
```

```
-1.0000   -0.0049    1.0000   -0.9999    0.0015    0.9999    1.0000   -0.0013   -1.0000
 0.0013   -0.0010   -0.0013    0.0007   -1.0000   -0.0010   -0.0013    1.0000    0.0016
 1.0000    0.0048   -1.0000    0.9999   -0.0018   -0.9999   -1.0000    0.0016    1.0000
-0.9999   -0.0045    0.9999   -1.0000    0.0012    1.0000    0.9999   -0.0010   -0.9999
 0.0049    1.0000   -0.0044    0.0040    0.0010   -0.0040   -0.0044   -0.0010    0.0044
 0.9999    0.0046   -0.9999    1.0000   -0.0012   -1.0000   -0.9999    0.0010    0.9999
 0.9999    0.0041   -0.9999    0.9999   -0.0007   -0.9999   -0.9999    0.0005    0.9999
-0.0054   -1.0000    0.0050   -0.0046   -0.0010    0.0046    0.0050    0.0010   -0.0050
-0.9999   -0.0044    0.9999   -0.9999    0.0007    0.9999    0.9999   -0.0005   -0.9999
-1.0000   -0.0052    1.0000   -0.9998    0.0007    0.9998    1.0000   -0.0004   -1.0000
  Columns 19 through 25
-0.9999    0.0048    0.9999    0.9999   -0.0053   -0.9999   -1.0000
-0.0003   -0.0010    0.0003   -0.0002    0.0010    0.0002    0.0002
 0.9999   -0.0048   -0.9999   -0.9999    0.0053    0.9999    1.0000
 0.9999   -0.0037   -0.9999   -1.0000    0.0042    1.0000    0.9998
 0.0008    0.0010   -0.0008   -0.0003   -0.0010    0.0003    0.0002
-0.9999    0.0037    0.9999    1.0000   -0.0042   -1.0000   -0.9998
 0.9999   -0.0048   -0.9999   -0.9999    0.0053    0.9999    1.0000
 0.0049   -1.0000   -0.0050   -0.0045    1.0000    0.0048    0.0056
-0.9999    0.0044    0.9999    0.9999   -0.0049   -0.9999   -1.0000
-0.9999    0.0049    0.9999    0.9999   -0.0054   -0.9999   -1.0000
-0.0045    1.0000    0.0046    0.0041   -1.0000   -0.0044   -0.0052
 0.9999   -0.0044   -0.9999   -0.9999    0.0050    0.9999    1.0000
-1.0000    0.0040    1.0000    0.9999   -0.0046   -0.9999   -0.9998
 0.0012    0.0010   -0.0012   -0.0007   -0.0010    0.0007    0.0007
 1.0000   -0.0040   -1.0000   -0.9999    0.0046    0.9999    0.9998
 0.9999   -0.0044   -0.9999   -0.9999    0.0050    0.9999    1.0000
-0.0010   -0.0010    0.0010    0.0005    0.0010   -0.0005   -0.0004
-0.9999    0.0044    0.9999    0.9999   -0.0050   -0.9999   -1.0000
 1.0000   -0.0040   -1.0000   -0.9999    0.0046    0.9999    0.9998
-0.0040    1.0000    0.0042    0.0037   -1.0000   -0.0040   -0.0047
-1.0000    0.0042    1.0000    0.9999   -0.0047   -0.9999   -0.9998
-0.9999    0.0037    0.9999    1.0000   -0.0042   -1.0000   -0.9998
 0.0046   -1.0000   -0.0047   -0.0042    1.0000    0.0045    0.0053
 0.9999   -0.0040   -0.9999   -1.0000    0.0045    1.0000    0.9998
 0.9998   -0.0047   -0.9998   -0.9998    0.0053    0.9998    1.0000
valores propios
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0000 0.0003 0.0008
3.9958 4.0041 16.9989
descomposición Cholesky
  Columns 1 through 9
 1.0000    0.0007   -1.0000   -0.9999   -0.0012    0.9999   -1.0000   -0.0056    1.0000
      0    1.0000    0.0002    0.0005   -1.0000   -0.0008    0.0002    0.0010   -0.0002
      0         0    0.0002   -0.0004    0.0000    0.0004    0.0002   -0.0049   -0.0002
      0         0         0    0.0138   -0.0000   -0.0138   -0.0000   -0.0820   -0.0000
      0         0         0         0    0.0001    0.0000    0.0000    0.0079    0.0000
```

0	0	0	0	0	0.0002	0.0000	-0.0082	-0.0000
0	0	0	0	0	0	0.0002	0.0015	-0.0002
0	0	0	0	0	0	0	0.9965	0.0004
0	0	0	0	0	0	0	0	0.0002
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Columns 10 through 18

1.0000	0.0052	-1.0000	0.9999	-0.0016	-0.9999	-1.0000	0.0013	1.0000
-0.0001	-0.0010	0.0001	-0.0006	-1.0000	0.0003	0.0001	1.0000	0.0002
-0.0002	0.0049	0.0002	0.0006	0.0000	-0.0006	0.0002	-0.0000	-0.0002
-0.0014	0.0820	0.0014	-0.0110	-0.0000	0.0110	0.0014	0.0000	-0.0014
-0.0000	-0.0079	0.0000	0.0001	0.0001	-0.0001	0.0000	-0.0001	-0.0000
0.0001	0.0082	-0.0000	0.0004	0.0000	-0.0004	-0.0000	-0.0000	0.0001
-0.0002	-0.0015	0.0002	-0.0000	0.0000	0.0000	0.0002	-0.0000	-0.0002
-0.0002	-0.9965	-0.0003	-0.0001	0.0000	0.0001	-0.0003	0.0000	0.0003
0.0001	0.0000	-0.0001	0.0002	0.0000	-0.0002	-0.0001	-0.0000	0.0001
0.0049	-0.0000	-0.0049	0.0049	-0.0000	-0.0049	-0.0049	0.0000	0.0049
0	0.0001	-0.0000	0.0001	0.0000	-0.0001	-0.0000	-0.0000	-0.0000
0	0	0.0001	-0.0001	-0.0000	0.0001	0.0001	0.0000	-0.0001
0	0	0	0.0112	0.0000	-0.0112	-0.0000	-0.0000	0.0000
0	0	0	0	0.0002	0.0000	0.0000	-0.0002	-0.0000
0	0	0	0	0	0.0002	-0.0000	0.0000	0.0000
0	0	0	0	0	0	0.0003	0.0000	-0.0003
0	0	0	0	0	0	0	0.0002	-0.0000
0	0	0	0	0	0	0	0	0.0001
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

```
  Columns 19 through 25
-0.9999    0.0048    0.9999    0.9999   -0.0053   -0.9999   -1.0000
 0.0003   -0.0010   -0.0003   -0.0008    0.0010    0.0008    0.0009
-0.0006    0.0050    0.0006    0.0004   -0.0050   -0.0004    0.0009
 0.0110    0.0820   -0.0110   -0.0138   -0.0820    0.0137   -0.0031
-0.0001   -0.0079    0.0001    0.0000    0.0079    0.0000    0.0002
-0.0004    0.0082    0.0004    0.0002   -0.0082   -0.0002    0.0003
 0.0000   -0.0015   -0.0000   -0.0000    0.0015    0.0000    0.0004
 0.0001   -0.9965   -0.0003    0.0000    0.9965    0.0003   -0.0003
-0.0002    0.0000    0.0003    0.0000    0.0000    0.0000    0.0001
-0.0049   -0.0000    0.0049    0.0000    0.0000   -0.0000   -0.0016
-0.0001    0.0001    0.0001    0.0000   -0.0002    0.0000    0.0001
 0.0001   -0.0000   -0.0001   -0.0000   -0.0000   -0.0000    0.0002
-0.0112    0.0000    0.0112    0.0000    0.0000    0.0000    0.0028
 0.0000    0.0000   -0.0000    0.0000   -0.0000    0.0000    0.0001
 0.0002    0.0000   -0.0002   -0.0000    0.0000   -0.0000    0.0001
-0.0000   -0.0000   -0.0000    0.0000    0.0000    0.0000   -0.0001
-0.0000   -0.0000   -0.0000    0.0000   -0.0000   -0.0000    0.0001
-0.0000   -0.0000   -0.0000   -0.0000   -0.0000    0.0000    0.0002
 0.0002   -0.0000   -0.0002   -0.0000   -0.0000   -0.0000    0.0000
      0    0.0002    0.0000   -0.0000   -0.0002    0.0000    0.0001
      0         0    0.0002    0.0000    0.0000    0.0000    0.0001
      0         0         0    0.0002    0.0000   -0.0002   -0.0000
      0         0         0         0    0.0001    0.0000    0.0002
      0         0         0         0         0    0.0002    0.0002
      0         0         0         0         0         0    0.0058
norma
1.0000   1.0000   1.0000   1.0000   1.0000   1.0000   1.0000   1.0000   1.0000
1.0000   1.0000   1.0000   1.0000   1.0000   1.0000   1.0000   1.0000   1.0000
1.0000   1.0000   1.0000   1.0000   1.0000   1.0000   1.0000

     VARIABLE X MINIMIZADORA
     1     0     0     0     1     1
     0     1     1     1     0     0
     1     1     0     0     0     1
     0     0     1     1     1     0
     ASIGNACIÓN DE MÍNIMA DISTANCIA
    'A'    'H'    'H'    'H'    'A'    'A'
    'H'    'A'    'A'    'A'    'H'    'H'
    'A'    'A'    'H'    'H'    'H'    'A'
    'H'    'H'    'A'    'A'    'A'    'H'
   DISTANCIA MÍNIMA DE RECORRIDO TOTAL
           dmin = 108
Tiempo:         t = 0.859324788485687 segundos
ans =
  107.9875  107.9937
```

Incluimos también los resultados para $n = 3$ hasta $n = 6$, pero sólo lo que corresponde al formato de asignación HA.

```
Resultado del programa semidefinido y la aproximación aleatoria para n=3:

    VARIABLE X MINIMIZADORA
     0     1     1     0     1     0     0     1     0     1
     1     0     0     0     0     1     1     1     1     0
     1     0     0     1     1     0     0     1     1     0
     0     1     1     0     0     0     1     0     1     1
     0     1     1     1     0     1     0     0     0     1
     1     0     0     1     1     1     1     0     0     0

     ASIGNACIÓN DE MÍNIMA DISTANCIA
    'H'    'A'    'A'    'H'    'A'    'H'    'H'    'A'    'H'    'A'
    'A'    'H'    'H'    'H'    'H'    'A'    'A'    'A'    'A'    'H'
    'A'    'H'    'H'    'A'    'A'    'H'    'H'    'A'    'A'    'H'
    'H'    'A'    'A'    'H'    'H'    'H'    'A'    'H'    'A'    'A'
    'H'    'A'    'A'    'A'    'H'    'A'    'H'    'H'    'H'    'A'
    'A'    'H'    'H'    'A'    'A'    'A'    'A'    'H'    'H'    'H'

   DISTANCIA MÍNIMA DE RECORRIDO TOTAL
           dmin = 980.96

Tiempo:        t = 1.39526741527205 segundos

ans =

  971.4136  971.3332

dmin =

  980.96
```

Resultado del programa semidefinido y la aproximación aleatoria para n=4:

VARIABLE X MINIMIZADORA

1	0	0	1	1	0	0	1	0	1	0	1	0	1
0	0	0	1	0	0	1	0	0	1	1	1	1	1
0	1	1	0	1	1	0	0	0	1	1	0	0	1
1	1	1	1	0	0	0	1	1	0	0	0	1	0
1	1	1	0	1	0	0	1	1	0	0	1	0	0
0	1	1	0	0	1	1	0	0	1	1	0	1	0
0	0	0	1	1	1	1	1	1	0	1	0	0	0
1	0	0	0	0	1	1	0	1	0	0	1	1	1

ASIGNACIÓN DE MÍNIMA DISTANCIA

'A'	'H'	'H'	'A'	'A'	'H'	'H'	'A'	'H'	'A'	'H'	'A'	'H'	'A'
'H'	'H'	'H'	'A'	'H'	'H'	'A'	'H'	'H'	'A'	'A'	'A'	'A'	'A'
'H'	'A'	'A'	'H'	'A'	'A'	'H'	'H'	'H'	'A'	'A'	'H'	'H'	'A'
'A'	'A'	'A'	'A'	'H'	'H'	'H'	'A'	'A'	'H'	'H'	'H'	'A'	'H'
'A'	'A'	'A'	'H'	'A'	'H'	'H'	'A'	'A'	'H'	'H'	'A'	'H'	'H'
'H'	'A'	'A'	'H'	'H'	'A'	'A'	'H'	'H'	'A'	'A'	'H'	'A'	'H'
'H'	'H'	'H'	'A'	'A'	'A'	'A'	'A'	'A'	'H'	'A'	'H'	'H'	'H'
'A'	'H'	'H'	'H'	'H'	'A'	'A'	'H'	'A'	'H'	'H'	'A'	'A'	'A'

DISTANCIA MÍNIMA DE RECORRIDO TOTAL

dmin = 1543.15

Tiempo: t = 1.8590557598801 segundos

```
ans =

  1.0e+003 *

    1.5279    1.5277


dmin =

  1.5432e+003
```

Resultado del programa semidefinido y la aproximación aleatoria para n=5:

```
     VARIABLE X MINIMIZADORA
1    1    1    0    1    0    0    0    0    0    1    0    1    0    0    1    1    1
0    0    0    0    0    1    1    1    1    1    1    1    1    0    0    0    0    1
0    1    1    1    1    0    0    0    1    1    1    0    0    1    1    0    0    0
0    0    0    1    1    0    0    1    1    0    0    1    1    1    1    1    0    0
0    0    0    1    0    1    1    0    0    0    0    1    1    1    0    1    1    1
1    1    1    0    1    1    1    1    1    1    0    0    0    0    0    0    0    0
0    0    1    1    0    0    1    1    1    1    0    1    0    0    0    1    0    1
1    1    0    0    0    1    0    0    0    0    0    1    1    1    1    0    1    1
1    0    0    1    1    0    0    0    0    1    1    0    0    1    1    1    1    0
1    1    1    0    0    1    1    1    0    0    1    0    0    0    1    0    1    0


     ASIGNACIÓN DE MÍNIMA DISTANCIA
'A'  'A'  'A'  'H'  'A'  'H'  'H'  'H'  'H'  'H'  'A'  'H'  'A'  'H'  'H'  'A'  'A'  'A'
'H'  'H'  'H'  'H'  'H'  'A'  'A'  'A'  'A'  'A'  'A'  'A'  'A'  'H'  'H'  'H'  'H'  'A'
'H'  'A'  'A'  'A'  'A'  'H'  'H'  'H'  'A'  'A'  'A'  'H'  'H'  'A'  'A'  'H'  'H'  'H'
'H'  'H'  'H'  'A'  'A'  'H'  'H'  'A'  'A'  'H'  'H'  'A'  'A'  'A'  'A'  'A'  'H'  'H'
'H'  'H'  'H'  'A'  'H'  'A'  'A'  'H'  'H'  'H'  'H'  'A'  'A'  'A'  'H'  'A'  'A'  'A'
'A'  'A'  'A'  'H'  'A'  'A'  'A'  'A'  'A'  'A'  'H'  'H'  'H'  'H'  'H'  'H'  'H'  'H'
'H'  'H'  'A'  'A'  'H'  'H'  'A'  'A'  'A'  'A'  'H'  'A'  'H'  'H'  'H'  'A'  'H'  'A'
'A'  'A'  'H'  'H'  'H'  'A'  'H'  'H'  'H'  'H'  'H'  'A'  'A'  'A'  'A'  'H'  'A'  'A'
'A'  'H'  'H'  'A'  'A'  'H'  'H'  'H'  'H'  'A'  'A'  'H'  'H'  'A'  'A'  'A'  'A'  'H'
'A'  'A'  'A'  'H'  'H'  'A'  'A'  'A'  'H'  'H'  'A'  'H'  'H'  'H'  'A'  'H'  'A'  'H'


   DISTANCIA MÍNIMA DE RECORRIDO TOTAL
          dmin = 2007.187
Tiempo:         t = 3.32472842218774 segundos

ans =

  1.0e+003 *

    1.9943    1.9944
```

Resultado del programa semidefinido y la aproximación aleatoria para n=6:

VARIABLE X MINIMIZADORA

```
1   0   0   1   1   1   0   1   1   0   0   0   0   1   1   0   1   0   0   1   1   0
1   0   0   0   0   0   0   0   0   1   1   1   0   0   0   1   1   1   1   1   1   1
1   0   0   0   0   1   1   1   1   1   1   0   0   1   1   1   1   0   0   0   0   0
0   0   0   1   0   0   0   1   1   0   0   1   1   1   1   1   0   0   1   1   1   0
0   1   1   1   1   1   1   1   1   0   0   0   0   0   0   1   0   0   0   0   1   1
0   1   1   0   0   1   1   0   1   1   1   1   1   0   0   1   0   0   0   0   0   1
1   1   1   0   0   0   0   1   1   1   0   1   1   0   0   0   1   1   0   0   0   1
1   0   0   0   1   0   0   0   0   1   1   0   0   1   1   0   1   1   0   1   1   1
0   1   0   0   0   1   1   0   0   0   0   1   1   1   1   0   0   1   1   1   1   0
1   0   1   1   1   1   1   0   0   1   1   0   0   0   0   1   0   1   1   0   0   0
0   1   1   1   1   0   1   0   0   0   0   0   1   1   1   0   1   1   1   0   0   0
0   1   1   1   1   0   0   1   0   0   1   1   1   0   0   0   0   0   1   1   0   1
```

ASIGNACIÓN DE MÍNIMA DISTANCIA

```
'A' 'H' 'H' 'A' 'A' 'A' 'H' 'A' 'A' 'H' 'H' 'H' 'H' 'A' 'A' 'H' 'A' 'H' 'H' 'A' 'A' 'H'
'A' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'H' 'A' 'A' 'A' 'H' 'H' 'H' 'A' 'A' 'A' 'A' 'A' 'A' 'A'
'A' 'H' 'H' 'H' 'H' 'A' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'A' 'A' 'A' 'H' 'H' 'H' 'H' 'H'
'H' 'H' 'H' 'A' 'H' 'H' 'H' 'A' 'A' 'H' 'H' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'A' 'A' 'H'
'H' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'H' 'H' 'H' 'H' 'A' 'H' 'H' 'H' 'H' 'A' 'A'
'H' 'A' 'A' 'H' 'H' 'A' 'A' 'H' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'H' 'H' 'H' 'H' 'H' 'A'
'A' 'A' 'A' 'H' 'H' 'H' 'H' 'A' 'A' 'A' 'H' 'A' 'A' 'H' 'H' 'H' 'A' 'A' 'H' 'H' 'H' 'A'
'A' 'H' 'H' 'H' 'A' 'H' 'H' 'H' 'H' 'A' 'A' 'H' 'H' 'A' 'A' 'H' 'A' 'A' 'H' 'A' 'A' 'A'
'H' 'A' 'H' 'H' 'H' 'A' 'A' 'H' 'H' 'H' 'H' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'A' 'A' 'A' 'H'
'A' 'H' 'A' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'A' 'H' 'H' 'H' 'H' 'A' 'H' 'A' 'A' 'H' 'H' 'H'
'H' 'A' 'A' 'A' 'A' 'H' 'A' 'H' 'H' 'H' 'H' 'H' 'A' 'A' 'A' 'H' 'A' 'A' 'A' 'H' 'H' 'H'
'H' 'A' 'A' 'A' 'A' 'H' 'H' 'A' 'H' 'H' 'A' 'A' 'A' 'H' 'H' 'H' 'H' 'H' 'A' 'A' 'H' 'A'
```

DISTANCIA MÍNIMA DE RECORRIDO TOTAL

```
         dmin = 2738.69
Tiempo:        t = 6.61067251564095 segundos
ans =

  1.0e+003 *

    2.6804    2.6806
```

Capítulo 9

Conclusiones y Recomendaciones

9.1. Conclusiones

En líneas generales, se logró el propósito principal de este trabajo. Se realizó una interpretación exhaustiva del problema de asignación local-visitante planteado por Suzuka (y otros) [11]. Esto es, la formulación del modelo en término de las distancias y el itinerario como data y de la asignación como variable para minimizar el recorrido total de los equipos en un torneo deportivo del tipo *double round robin*. La formulación Min Res Cut, mediante la cual se pudo expresar la función objetivo como una fórmula no condicionada, en término de los elementos de un grafo no dirigido, donde se anexa un vértice específico al conjunto de vértices originales del modelo y las aristas formadas con este vértice específico y los vértices del modelo original. Por último, la relajación del problema como un programa semidefinido y la aplicación del algoritmo de aproximación aleatoria de Goemans y Williamson [4] sobre la matriz definida positiva minimizadora resultante, hasta obtener una solución aproximada de nuestro problema de asignación, con una gran precisión y con gran rapidez de cálculo computacional.

Para cumplir con este propósito, comenzamos en el Capítulo 1. presentando la interpretación exhaustiva del modelo *double round robin*. Establecimos la data como matrices mostradas en formato de tablas o cuadros: Por un lado, la única matriz distancia, cuyas entradas $d_{i,j}$ son las distancias entre las sedes de los equipos i y j. Y, la matriz itinerario, de orden $t \times s$, donde t es la cantidad de equipos (*teams*) y s es el número de casillas (*slots*) o momentos cuando se efectúa cada partido. El posible itinerario no es único y determinar cuál será el itinerario no forma parte de nuestro problema. Es decir, de los múltiples itinerarios posibles escogimos uno cualquiera. Cada entrada $\tau_{k,l}$ de la matriz itinerario representa el número asignado al equipo que juega con el equipo k en la casilla l. La data que utilizamos, desde $n = 3$, se obtuvo con los códigos que se muestran en el Apéndice C. Definimos también la incógnita del problema. Ésta es la matriz asignación, donde cada entrada, $a_{k,l} = A$, si el equipo k debe jugar como visitante (*away*) en la casilla l, o bien, $a_{k,l} = H$, si debe hacerlo como local (*home*). Con esta notación planteamos el problema de asignación HA consistente en encontrar la matriz asignación que minimice la distancia total de recorrido de todos los equipos durante el desarrollo del torneo.

Si observamos, en el Capítulo 8 las matrices asignación minimizadoras resultantes, allí puede verse que, para cada equipo (cada fila), aparecen muchas 'H' seguidas (también muchas 'V'). Esto ocurre porque no se implementó el criterio *numbers of breaks* [8] en ninguna etapa del desarrollo de nuestro trabajo. Sin embargo, aquí aplicamos una solución un tanto simple, que consiste en mezclar los pareamientos (ver Apéndice C.) de los dos itinerarios *single round robin* que forman el itinerario *double round robin*, en lugar de concatenarlos. De esta forma, se consiguió disminuir la separación entre

las casillas correspondientes a un mismo oponente de cada equipo (en cada fila), siendo diferente el resultado para ambas casillas ya que las restricciones así lo exigen.
En el Capítulo 2., conseguimos una expresión matemática de la distancia total de recorrido de los equipos. Se obtuvo una expresión condicionada de esta función objetivo, pero con ella pudo resolverse el problema de asignación mediante la enumeración exhaustiva de todas las soluciones factibles. El resultado obtenido es exacto, pero con la limitante de que fue computacionalmente aplicable únicamente para los valores $n = 2$, 3 y 4. Estos resultados nos permitieron evaluar los algoritmos aplicados para la resolución aproximada del problema. Para los valores de n desde 5 en adelante, aplicamos una enumeración parcial aleatoria con la que obtuvimos una cota superior en cada instancia. Con la idea de sólo mostrar el comportamiento de los algoritmos, no ambicionamos su aplicación para valores muy grandes de n y optamos hasta $n = 6$ en todos los cálculos.
En el capítulo anterior puede verse cómo la exposición del problema de asignación local-visitante pudo realizarse formalmente en términos combinatorios. En el Capítulo 3. definimos un grafo, $G = (V, E)$, donde cada vértice está asociado a cada entrada de la matriz asignación. Para una solución factible, los vértices asociados a las entradas con valor A quedaron agrupadas en un subconjunto de vértices V', con el cual se definió a su vez una bipartición $\delta(V')$ sobre el conjunto de vértices. En términos de estos conjuntos se expresaron tanto el recorrido total como las restricciones. Sin embargo, la expresión del recorrido total continuó siendo la misma fórmula condicionada de la formulación matemática. Este formato no es apropiado para el desarrollo matemático progresivo en la búsqueda de algoritmos eficientes que nos permitan resolver el problema de asignación original.
Este inconveniente se salvó en el Capítulo 4. al aplicar la técnica del problema Min Res Cut. Ésta consiste en redefinir el conjunto de vértices V agregando un vértice específico r, que obviamente no pertenecerá a V'. E igualmente redefinir el conjunto de aristas E agregando las que tienen a r como vértice extremo. En esta formulación, tanto el recorrido total como las restricciones se expresaron en término de las aristas de E. Su utilidad consiste en que, mediante manipulaciones algebraicas sencillas, logramos una fórmula no condicionada para el recorrido total de los equipos. Dicha expresión consiste en la suma de los pesos de las aristas de E que pertenecen a la bipartición $\delta(V')$ que definimos en la formulación combinatoria. Conseguimos deducir las mismas fórmulas para estos pesos que muestra Suzuka en su trabajo.
El Capítulo 5 lo dedicamos a deducir la formulación entera a partir del modelo Min Res Cut del problema de asignación local-visitante. El resultado fue un programa entero cuadrático binario. Para ello se introdujo un conjunto binario de par de valores para asumir las variables asociadas a las aristas del grafo. Éste debe ser $\{0, 1\}$. Pero estas variables asociadas a las aristas debieron expresarse como funciones elementales de otras variables asociadas a los vértices, las cuales pueden asumir cualquier par de valores en $\{z_1, z_2\}$, con $z_1, z_2 \in \mathbb{Z}$. En este trabajo manejamos dos alternativas, $\{0, 1\}$ y $\{-1, +1\}$, aunque los experimentos computacionales los restringimos al caso $\{0, 1\}$. El desarrollo algebraico de la función objetivo con estas funciones elementales produjo las formulaciones cuadráticas enteras binarias del problema de asignación local-visitante original.
Como es sabido, el programa entero es NP-Completo. Dedicamos el Capítulo 6. a resolverlo mediante un algoritmo de aproximación. Es sabido también que el algoritmo de ramificación y acotamiento ha resultado ser eficiente en algunas instancias, dependiendo de las dimensiones del problema. Para aplicar esta técnica necesitamos utilizar una relajación del programa entero para evaluar la función y sus restricciones en cada ramificación. Por una lado estudiamos una relajación de programación no lineal (PNL), con restricción no lineal, del programa entero cuadrático del problema de Asignación HA, aprovechando la geometría (hiper)esférica del conjunto de soluciones factibles. Para su evaluación se utilizó el código matlab fmincon.m. Por otra parte, propusimos también la relajación de programación cuadrática (QP), de variables continuas, con las restricciones lineales originales. Para

esta relajación aplicamos el código quadprog.m. A la luz de los resultados obtenidos, que pueden verse en el Capítulo 8, la relajación QP, con una única iteración para dar el óptimo, prometía ser muchísimo mas eficiente que la relajación PNL, que requería de miles de iteraciones para el mismo resultado.
Para los cálculos computacionales utilizamos el software matlab MIQP [1], que utiliza la técnica de ramificación y acotamiento y aplica como resolvente la función quadprog.m. Desafortunadamente, se trata de una técnica basada en la enumeración parcial de las variables, con valores no enteros, que deben ser utilizadas como ramificadoras. El código corrió bastante bien solamente para $n = 2$. Procurando mejorar este resultado, optamos por aplicar una relajación de programación no lineal con una única restricción lineal del programa entero cuadrático binario, asociando el conjunto de soluciones factibles con los puntos de un hiperplano. Sin embargo, el resultado tampoco fue satisfactorio a partir de $n = 4$. Finalmente, ideamos una variante del algoritmo de ramificación y acotamiento aplicando una heurística codiciosa para la ramificación. Esta técnica permite salvar el inconveniente que significa la enumeración parcial, propia de la ramificación, del gran número de variables que se manejan en nuestro problema, pero con la salvedad de que se puede esperar sólo un resultado aproximado. A pesar de ello el resultado fue satisfactorio.
En el Capítulo 7. expusimos con bastante detalle la aplicación del programa semidefinido como una relajación del programa entero cuadrático binario. Siguiendo el desarrollo propuesto por Helmberg [5] logramos la misma expresión que Suzuka [11] en su trabajo para la formulación primal del programa semidefinido. Continuando con la teoría desarrollada por Helmberg, obtuvimos una formulación primal-dual particularmente útil para resolver el programa semidefinido aplicando la técnica de punto interior. Para resolver computacionalmente el programa semidefinido, utilizamos el software matlab SDPT3-01 [10], el cual utiliza la técnica de punto interior en los mismos términos que Helmberg en su obra, en cuanto a la función barrera logarítmica, la selección del punto inicial, el camino central y la condición de factibilidad de los resultados parciales.
Se esperaba que el resultado del programa semidefinido fuese una matriz definida positiva. En esta oportunidad los resultados fueron los esperados. Para todos los valores de n aquí tratados, se obtuvo una matriz definida positiva en un tiempo de cálculo computacional satisfactoriamente corto.
Seguidamente, mediante descomposición de Cholesky, pudimos obtener el conjunto de vectores que relajan las variables enteras binarias originales. Por último, aplicamos a estos vectores el método de aproximación aleatoria de Goemans y Williamson [4]. El resultado obtenido fue también satisfactoriamente preciso a pesar de que aplicamos esta técnica sólo 100 veces para todos los valores de n.
Reservamos el Capítulo 8. para mostrar un resumen de los resultados de todos los cálculos computacionales llevados a cabo.
A la vista de los resultados reflejados en el Capítulo 8., queda manifiesto que la resolución exacta del programa entero cuadrático binario mediante el algoritmo de ramificación y acotamiento, no produce buenos frutos en vista de que, aún para valores pequeños de n, el costo computacional es enorme, al menos en lo que respecta a esta instancia, el problema de asignación local-visitante.
Por el contrario, es indudable que el programa semidefinido, conjuntamente con el algoritmo de aproximación aleatoria de Goemans y Williamson, es una poderosa herramienta de cómputo para obtener un mínimo aproximado del programa entero cuadrático binario, con muy buena precisión y en muy corto tiempo computacional.

9.2. Recomendaciones

Previamente, consideramos oportuno informar que, para la aplicación de los software, se utilizó un procesador Intel Core2 Duo CPU 1.80GHz , modelo Vostro 1510, de 2048MB de memoria, 150GB de capacidad (80GB disponibles) y 1800MHz de velocidad. Es de esperarse que con un procesador de mayor capacidad se puede mejorar los resultados aquí alcanzados.

En vista de la gran eficiencia del uso del programa semidefinido para resolver el problema de asignación local-visitante, asumimos con mucho optimismo la posibilidad de que su aplicación a situaciones reales puede arrojar un resultado suficientemente preciso para aceptarlo como válido y en un tiempo relativamente corto que lo haga computacionalmente viable.

Siendo así, el programa semidefinido puede aplicarse a situaciones reales, como los torneos deportivos colectivos que se escenifican en nuestro país: baloncesto, beisbol y fútbol. Particularmente, el caso del beisbol es el más pertinente ya que los equipos juegan diariamente. Aunque su aplicación es mucho más compleja ya que requiere de muchas más variables, que las del tipo $X_{t,s}$ aquí estudiadas, ya que el torneo no es *double round robin*.

Una generalización del problema de asignación local-visitante aquí planteado, consistiría en seguir también el criterio *numbers of breaks*. Hacerlo significaría aumentar el número de restricciones y, seguramente, el de índices, como lo propone Ribeiro [8], ya que habría que considerar la matriz itinerario como variable además de la matriz asignación.

Apéndice A
La Data

Se muestra la data utilizada en los seis casos estudiados en este trabajo. Se trata, en cada uno, del itinerario $\mathcal{T}$ y de la matriz distancia $\mathcal{D}$ con las distancias entre las sedes de los diferentes equipos que participan en el torneo. Para $n = 4$ y superiores se utiliza el formato de código matlab debido al tamaño de las matrices.

Caso n=2

$\mathcal{T}$	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$
$t=1$	2	3	2	4	3	4
$t=2$	1	4	1	3	4	3
$t=3$	4	1	4	2	1	2
$t=4$	3	2	3	1	2	1

$\mathcal{D}$	$t=1$	$t=2$	$t=3$	$t=4$
$t=1$	0	7,0	5,0	4,0
$t=2$	7,0	0	10,0	8,0
$t=3$	5,0	10,0	0	3,0
$t=4$	4,0	8,0	3,0	0

Caso n=3

$\mathcal{T}$	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$	$s=7$	$s=8$	$s=9$	$s=10$
$t=1$	3	3	6	5	5	6	2	4	4	2
$t=2$	4	4	5	6	3	3	1	5	6	1
$t=3$	1	1	4	4	2	2	6	6	5	5
$t=4$	2	2	3	3	6	5	5	1	1	6
$t=5$	6	6	2	1	1	4	4	2	3	3
$t=6$	5	5	1	2	4	1	3	3	2	4

$\mathcal{D}$	$t=1$	$t=2$	$t=3$	$t=4$	$t=5$	$t=6$
$t=1$	0.	12,50	27,50	17,50	9,38	11,88
$t=2$	13,50	0.	35,00	23,75	20,63	15,00
$t=3$	27,50	35,00	0.	34,00	29,88	30,38
$t=4$	17,50	23,75	34,00	0.	9,88	26,38
$t=5$	9,38	20,63	29,88	9,88	0.	18,25
$t=6$	11,88	15,00	30,38	28,38	18,25	0.

Caso n=4

```
function T=itinhan4
T=[  3   3   5   5   8   8   7   2   4   4   6   6   2   7;
     5   4   6   6   7   3   3   1   8   8   4   7   1   5;
     1   1   7   4   4   2   2   7   5   5   8   8   6   6;
     7   2   8   3   3   7   6   6   1   1   2   5   5   8;
     2   7   1   1   6   6   8   8   3   3   7   4   4   2;
     8   8   2   2   5   5   4   4   7   7   1   1   3   3;
     4   5   3   8   2   4   1   3   6   6   5   2   8   1;
     6   6   4   7   1   1   5   5   2   2   3   3   7   4];
```

```
function d=disthan4
d=[ 0.     12.50   27.50   17.50    9.38   11.88    7.81    9.06;
   12.50    0.     35.00   23.75   20.63   15.00   13.44   10.63;
   27.50   35.00    0.     21.00   31.88   23.38   24.31   25.57;
   17.50   23.75   21.00    0.     23.88   28.38   23.31   19.56;
    9.38   20.63   31.88   23.88    0.     12.25   14.19   11.44;
   11.88   15.00   23.38   28.38   12.25    0.      9.69   17.95;
    7.81   13.44   24.31   23.31   14.18    9.69    0.     15.88;
    9.06   10.63   25.56   19.56   11.44   17.94   15.89    0. ];
```

Caso n=5

```
function T=itinhan5
 T=[2  7  5  5 10 10  7  2  6  3  8  8  9  9  4  6  3  4;
    1  8  6  7  4  4  9  1  8  5  7 10  3  3 10  9  5  6;
    9  9  8  6  5  8  6 10 10  1  4  4  2  2  7  7  1  5;
    8  6  7  8  2  2  5  5  9  9  3  3  6  7  1 10 10  1;
    6 10  1  1  3  7  4  4  7  2  9  9 10  6  8  8  2  3;
    5  4  2  3  7  9  3  8  1 10 10  7  4  5  9  1  8  2;
   10  1  4  2  6  5  1  9  5  8  2  6  8  4  3  3  9 10;
    4  2  3  4  9  3 10  6  2  7  1  1  7 10  5  5  6  9;
    3  3 10 10  8  6  2  7  4  4  5  5  1  1  6  2  7  8;
    7  5  9  9  1  1  8  3  3  6  6  2  5  8  2  4  4  7];
```

```
function d=disthan5
d=[ 0.    12.50  27.50  17.50   9.38  11.88   7.81   9.06   7.03   7.66;
   12.50   0.    35.00  23.75  20.63  15.00  13.44  10.63   9.84   8.44;
   27.50  35.00   0.    29.00  21.88  25.38  28.31  24.56  31.53  29.16;
   17.50  23.75  29.00   0.    10.88  22.38  18.31  13.56  23.53  21.16;
    9.38  20.63  21.88  10.88   0.    15.25   9.19  10.44  15.41  13.03;
   11.88  15.00  25.38  22.38  15.25   0.     8.689 13.94  15.91  16.53;
    7.81  13.44  28.31  18.31   9.19   8.69   0.     9.88   9.84  13.47;
    9.06  10.63  24.56  13.56  10.44  13.94   9.88   0.    14.09  12.72;
    7.03   9.84  31.53  23.53  15.41  15.91   9.84  14.09   0.    10.69;
    7.66   8.44  29.16  21.16  13.03  16.53  13.47  12.72  10.69   0.  ];
```

Caso n=6

```
function T=itinhan6
%Para n=6, 2n=12 equipos y 2(2n-1)=22 casillas:
%
T=[12 12  7  7  3  4 10 10  8  8  2  6  4  2  5  3  9  9 11 11  6  5;
    6  6 12  5 10  3 11  7  4  4  1 11  9  1  8  8  5 12  3 10  7  9;
    4  9 11 12  1  2  8 11  9  5  7  7  6  6 12  1 10 10  2  4  5  8;
    3  7  6  6  5  1  9  9  2  2 10 10  1  5  7 11 11  8  8  3 12 12;
   10 10  8  2  4 11  7  6 12  3  6 12 11  4  1  9  2  7  9  8  3  1;
    2  2  4  4  8  8 12  5 11  9  5  1  3  3  9  7  7 11 10 12  1 10;
    9  4  1  1 11 10  5  2 10 12  3  3  8  8  4  6  6  5 12  9  2 11;
   11 11  5 10  6  6  3 12  1  1  9  9  7  7  2  2 12  4  4  5 10  3;
    7  3 10 11 12 12  4  4  3  6  8  8  2 10  6  5  1  1  5  7 11  2;
    5  5  9  8  2  7  1  1  7 11  4  4 12  9 11 12  3  3  6  2  8  6;
    8  8  3  9  7  5  2  3  6 10 12  2  5 12 10  4  4  6  1  1  9  7;
    1  1  2  3  9  9  6  8  5  7 11  5 10 11  3 10  8  2  7  6  4  4];

function d=disthan6
%Para n=6, 2n=12 equipos y 2(2n-1)=22 casillas:
%
d=[ 0.   12.50 27.50 17.50  9.38 11.88  7.81  9.06  7.03  7.66  6.64  6.95;
   12.50  0.   35.00 23.75 20.63 15.00 13.44 10.63  9.84  8.44  8.05  7.34;
   27.50 35.00  0.   39.00 32.88 28.38 29.31 30.56 32.53 29.16 32.14 32.45;
   17.50 23.75 39.00  0.   23.88 20.38 24.31 14.56 14.53 20.16 17.14 20.45;
    9.38 20.63 32.88 23.88  0.   18.25  9.19 12.44 12.41 14.03 13.02 13.33;
   11.88 15.00 28.38 20.38 18.25  0.   14.69  7.94  9.91 12.53  7.52  8.83;
    7.81 13.44 29.31 24.31  9.19 14.69  0.   11.88 12.84 14.47  7.45 12.77;
    9.06 10.63 30.56 14.56 12.44  7.94 11.88  0.    7.09  8.72 14.70 10.02;
    7.03  9.84 32.53 14.53 12.41  9.91 12.84  7.09  0.   11.69  8.67  6.98;
    7.66  8.44 29.16 20.16 14.03 12.53 14.47  8.72 11.69  0.   11.30  5.61;
    6.64  8.05 32.14 17.14 13.02  7.52  7.45 14.70  8.67 11.30  0.    9.59;
    6.95  7.34 32.45 20.45 13.33  8.83 12.77 10.02  6.98  5.61  9.59  0.  ];
```

Apéndice B
Los Códigos

Mostramos todos los códigos que utilizamos para obtener los resultados que aparecen en el Capítulo 8. Algunos son de aplicación directa para obtener la matriz asignación que minimiza el recorrido total de los equipos. Tal es el caso de los enumhan_.m, no interactivos, donde el guión pié indica 2, 3, $\cdots$, 6, que fueron los únicos casos analizados para observar el comportamiento de los algoritmos. De la misma forma particularizada se diseñaron todos los demás códigos, de manera que sólo mostraremos los hechos para $n = 2$, siendo similares los códigos para los demás valores de n. Los enumhan_.m hallan el minimizador mediante la enumeración exhaustiva de las soluciones factibles. Esto fue computacionalmente viable únicamente para $n = 2$ hasta $n = 4$. Para el resto se optó por una enumeración parcial aleatoria.
Aparecen aquí también los códigos máster para ejecutar funciones matlab como el fmincon.n para la relajación PNL y el quadprog.m para la QP. Por otra parte, para resolver el programa entero cuadrático binario del nuestro problema de asignación local visitante, el código máster diseñado fue el miqphan_.m, para $n = 2$, para aplicar el software MIQP.m [1], el cual resuelve el programa entero mediante una variante de la técnica de ramificación y acotamiento que consiste en reducir, en una unidad, la dimensión de las matrices en cada ramificación, sustituyendo el valor entero, 0 o 1, asignado a la variable ramificadora, en lugar de aumentar en una unidad el número de restricciones. Incluimos el código ramifacotha.m, como una variante del MIQP.m en la que se aplica una heurística codiciosa para la ramificación. También el código máster ramifacothan_.m, para $n = 2$. Para resolver nuestro problema, relajado a un programa semidefinido, se utilizó el software SDPT3- 01 [10]. El código máster que diseñamos para su aplicación fue el semidefhan_.m. Dicho software se basa en la técnica del punto interior. La teoría que soporta dicha técnica, particularmente aplicable a los programas semidefinidos, puede verse en [5].
Podrá verse también los códigos diseñados para construir las matrices y las funciones de entradas para los software. La data, distancias e itinerarios, se introducen también como códigos.m y son los mostrados en el Apéndice A.

enumhan_.m

Código matlab que permite realizar los cálculos, mediante enumeración, para obtener la distancia mínima de recorrido en el problema Asignación HA. La data, el itinerario y las distancias, se introducen como archivos.m. Corresponde éste al caso más sencillo, $n = 2$, con enumeración exhaustiva. Para $n = 3$ y $n = 4$ es similar. Para valores superiores de n la enumeración es parcial aleatoria. El código es similar, salvo que se elije un tope, por ejemplo, tot= 50000000 y los datos se introducen aleatoriamente mediante, $Y = \text{randint}(1, m, [0, 1])$.

```
function enumhan2
tic
n=2; t=2*n; s=2*(2*n-1);
T=feval(@itinhan2); d=feval(@disthan2);
m=t*s/4; tot=2^m;
%        tot=50000000; % para n>=4
%SOLUCIONES FACTIBLES
k=1;
for ii=1:tot
    ly=0; X=ones(t,s); X=2*X; Y=dec2binvec(ii-1,m);
    %               Y = randint(1,m,[0,1]); % para n>=4
    for it=1:t-1
        for is=1:s-1
            ly=ly+1;
                if X(it,is)==2
                    X(it,is)=Y(ly); X(T(it,is),is)=1-X(it,is);
                    for ij=is+1:s
                        if T(it,ij)==T(it,is)
                            X(it,ij)=1-X(it,is); X(T(it,is),ij)=X(it,is);
                            break
                        end%if T
                    end%for ij
                else
                    ly=ly-1;
                end%if X
        end%for is
    end%for it
%DISTANCIA PARA LA ACTUAL SOLUCIÓN FACTIBLE
    dt=0;
    for it=1:t
        xts=X(it,1); dts=d(it,T(it,1));
        switch xts % TRAYECTO INICIAL
            case 0
                Lts=0;
            case 1
                Lts=dts;
        end%switch-1
        dt=dt+Lts; xts=X(it,s); dts=d(T(it,s),it);
        switch xts %TRAYECTO FINAL
```

```
            case 0
                Lts=0;
            case 1
                Lts=dts;
        end%switch-2
        dt=dt+Lts;
        for is=1:s-1 % TRAYECTOS EN EL ITINERARIO
            if X(it,is)==0 && X(it,is+1)==0
                Lts=0;
            elseif X(it,is)==1 && X(it,is+1)==1
                Lts=d(T(it,is),T(it,is+1));
            elseif X(it,is)==1 && X(it,is+1)==0
                Lts=d(T(it,is),it);
            elseif X(it,is)==0 && X(it,is+1)==1
                Lts=d(it,T(it,is+1));
            end%if xts
            dt=dt+Lts;
        end%for is
    end%for it
 %BÚSQUEDA DE LA DISTANCIA MÍNIMA
    if k==1
        dmin=dt; Xmin=X;
    elseif dt<dmin
        dmin=dt; Xmin=X;
    end
    k=k+1;
end%for ii
%MATRIZ ASIGNACIÓN ha MINIMIZADORA RESULTANTE
A=cell(t,s);
for it=1:t
    for is=1:s
        xts=Xmin(it,is);
        switch xts
            case 1
                A(it,is)={'A'};
            case 0
                A(it,is)={'H'};
        end
    end
end
disp(' '); disp('     VARIABLE X MINIMIZADORA'); disp(Xmin);
disp(' '); disp('     ASIGNACIÓN DE MÍNIMA DISTANCIA'); disp(A);
disp(' '); disp('   DISTANCIA MÍNIMA DE RECORRIDO TOTAL');
disp(['           dmin = ',mat2str(dmin)]);
disp(' '); k=k-1; disp(['Iteraciones:   k = ',int2str(k)]);
t=toc; disp(['Tiempo:        t = ',mat2str(t),' segundos']);
```

pnlhan_.m

Código máster que permite resolver la relajación de programación no lineal del programa entero binario de asignación HA, utilizando la función fmincon.m. Corresponde éste al caso más sencillo, $n = 2$. Para los demás casos es similar.

```
function pnlhan2
tic
n=2; t=2*n; s=2*(2*n-1); ts=t*s;
lb=zeros(1,ts); ub=ones(1,ts);
al=lb(1:ts/2); au=ub(1:ts/2); lu=[al;au]; lu=lu(:);
x0=lu'; % vector del tipo [1 0 1 0 1 ...]
options=optimset('LargeScale','off','Display','iter','Algorithm','active-set');
%
[x,dmin,exitflag]=fmincon(@bfun2,x0,[],[],[],[],lb,ub,@radion2,options);
%
disp(' '); disp('     Mínimo x de la Relajación'); disp(x);
disp(' '); disp('   Distancia Mínima (Relajada)');
disp(['           dmin = ',mat2str(dmin)]);
t=toc;
disp(['t = ',mat2str(t),' segundos']);
```

Se muestra, además, los códigos para los pesos de las aristas, $w(v_{t,s}, r)$ y $w(v_{t,s}, v_{t,s+1})$, que hacen de coeficientes en la función objetivo, bfunn.m, y la restricción no lineal, radionn.m.

```
function wts=pesoswtsn2
n=2;
T=feval(@itinhan2); d=feval(@disthan2);
[t,s]=size(T);
wts=zeros(t,s);
for it=1:t
    for is=1:s
        if is==1
            wts(it,1)=0.5*(2*d(it,T(it,1))+d(T(it,1),T(it,2))-d(it,T(it,2))+ ...
            d(T(it,1),it));
        elseif is==s
            wts(it,s)=0.5*(2*d(T(it,s),it)+d(T(it,s-1),T(it,s))+d(it,T(it,s))- ...
            d(T(it,s-1),it));
        else
            wts(it,is)=0.5*(d(T(it,is),T(it,is+1))-d(it,T(it,is+1))+d(T(it,is),it)+ ...
            d(T(it,is-1),T(it,is))+d(it,T(it,is))-d(T(it,is-1),it));
        end %if
    end %for is
end %for it
%disp('wts = '); disp(wts);
```

```
function wtss=pesoswtssn2
T=feval(@itinhan2); d=feval(@disthan2);
[t,s]=size(T);
wtss=zeros(t,s-1);
for it=1:t
    for is=1:s-1
        wtss(it,is)=0.5*(-d(T(it,is),T(it,is+1))+d(it,T(it,is+1))+d(T(it,is),it));
    end %for is
end %for it
%disp('wtss = '); disp(wtss);

function f=bfun2(x)
n=2; t=2*n; s=2*(2*n-1);
wts=feval(@pesoswtsn2); wtss=feval(@pesoswtssn2);
sum1=0.0;
for it=1:t
    for is=1:s
        i=2*(2*n-1)*(it-1)+is;
        sum1=sum1+x(i)^2*wts(it,is);
    end %for is
end %for it
sum2=0.0;
for it=1:t
    for is=1:s-1
        i=2*(2*n-1)*(it-1)+is;
        sum2=sum2+(x(i)-x(i+1))^2*wtss(it,is);
    end %for is
end %for it
f=sum1+sum2;

function [c,ceq]=radion2(x)
n=2; t=2*n; s=2*(2*n-1); ts=t*s;
c=[];
sum3=0.0;
for it=1:t
    for is=1:s
        i=2*(2*n-1)*(it-1)+is;
        sum3=sum3+x(i)^2;
    end %for is
end %for it
radion2=sum3-0.5*ts;
ceq=radion2;
```

progquadn_.m

Código máster que permite resolver la relajación de programación cuadrática del programa entero binario de asignación HA, utilizando la función quadprog.m. Corresponde éste al caso más sencillo, $n = 2$. Para los demás casos es similar.

```
function  progquadn2
tic
n=2; t=2*n; s=2*(2*n-1); ts=t*s;
lb=zeros(1,ts); ub=ones(1,ts);
%
options=[];
x0=[];
H=feval(@matrizHn2); f=zeros(ts,1);
b1=ones(ts/2,1); b2=zeros(ts/4,1);
beq=[b1;b2];
Aeq=feval(@matrizAeqn2);
%
[x,fval,exitflag,output,lambda] = quadprog(H,f,[],[],Aeq,beq,lb,ub,x0,options)
%
t=toc;
disp(['t = ',mat2str(t),' segundos']);
```

Se muestra, además, los códigos para obtener las matrices, H y Aeq.

```
function H=matrizHn2
%Coeficiente del términos cuadrático en:  1/2X'H X + f'X
n=2; t=2*n; s=2*(2*n-1); ts=t*s;
wts=feval(@pesoswtsn2); wtss=feval(@pesoswtssn2);
H=zeros(ts,ts);
for it=1:t
    for is=1:s
        h=s*(it-1)+is;
        if is==1
            H(h,h)=2*(wts(it,1)+wtss(it,1)); H(h,h+1)=-2*wtss(it,1);
        elseif is==s
            H(h,h-1)=-2*wtss(it,is-1); H(h,h)=2*(wts(it,s)+wtss(it,s-1));
        else
            H(h,h-1)=-2*wtss(it,is-1); H(h,h+1)=-2*wtss(it,is);
            H(h,h)=2*(wts(it,is)+wtss(it,is-1)+wtss(it,is));
        end %if
    end %is
end %it
% disp('H =  '); disp(H);
```

```
function Aeq=matrizAeqn2
% Las restricciones Aeq X = beq
T=feval(@itinhan2);
n=2; t=2*n; s=2*(2*n-1); ts=t*s;
a1=1; b1=1+t*s/2; m=3*t*s/4;
Aeq=zeros(m,ts);
Ctrl=zeros(t,s);
for it=1:t-1;
    for is=1:s-1;
         if Ctrl(it,is)==0
            for ij=is+1:s
                if T(it,ij)==T(it,is)
                    Ctrl(it,is)=1; Ctrl(it,ij)=1;
                    Ctrl(T(it,is),is)=1; Ctrl(T(it,is),ij)=1;
                    hs1=s*(it-1)+is; hj1=s*(it-1)+ij;
                    Aeq(a1,hs1)=1; Aeq(a1,hj1)=1;
                    a2=a1+1; hs2=s*(T(it,is)-1)+is; hj2=s*(T(it,is)-1)+ij;
                    Aeq(a2,hs2)=1; Aeq(a2,hj2)=1; Aeq(b1,hs1)=1; Aeq(b1,hj2)=-1;
                    a1=a1+2; b1=b1+1;
                    break
                end %if T(it,ij)==T(it,is)
            end %for ij
         end %if Ctrl
    end %for is
end %for it
% disp(Aeq);
```

miqphan_.m

Código máster que permite resolver el programa entero binario de asignación HA mediante el código miqp.m. Corresponde éste al caso más sencillo, $n = 2$. Para los demás casos es similar. Los códigos para obtener las matrices, H y Aeq, son los mismos que para el programa cuadrático. Tiene dos opciones. La aplicación *original*, para la relajación QP. La aplicación *modificada*, para la relajación PNLL, en la que se introducen las restricciones originales del problema original, pero fuera de la matriz de restricciones. Esta modificación se utiliza también en el código ramifacotha.m que se muestra más adelante.

```
function miqphan2
tic
n=2; t=2*n; s=2*(2*n-1); ts=t*s; T=feval(@itinhan2);
options.method='bestdepth'; options.order='0'; options.verbose='2';
lb=zeros(1,ts); ub=ones(1,ts); vartype=1:1:ts;
H=feval(@matrizHn2); f=zeros(ts,1);
disp('naplic = 1 si es original -- naplic = 2 si es modificado.');
naplic=input('naplic = ');
switch naplic
    case 1
%XXXXXXXXXXX Aplicación 'original'XXXXXXXXXXXXXXXXXX
Aeq=feval(@matrizAeqn2); b1=ones(ts/2,1); b2=zeros(ts/4,1);
beq=[b1;b2]; % vector del tipo [1 1 1...1 0 ...0]
options.branchrule='max'; options.aplication='original';
[xmin,fmin]=miqp(H,f,[],[],Aeq,beq,vartype,lb,ub,[],[],options);
    case 2
%XXXXXXXXXXX Aplicación 'modificado'XXXXXXXXXXXXXXXXXX
options.branchrule='min'; options.aplication='modificado';
Aeq=ones(1,ts); beq=[ts/2];
[xmin,fmin]=miqp(H,f,[],[],Aeq,beq,vartype,lb,ub,[],options);
    otherwise
    disp('Valor de naplic inválido: ctrl+C y comience de nuevo');
end
xmin=xmin', xmin=reshape(xmin,s,t); xmin=xmin'
A=cell(t,s);
for it=1:t
    for is=1:s
        xts=xmin(it,is);
        switch xts
            case 1
                A(it,is)={'A'};
            case 0
                A(it,is)={'H'};
end, end, end
disp(' '); disp('     ASIGNACIÓN DE MÍNIMA DISTANCIA'); disp(A);
disp(['fmin =',mat2str(fmin)]); t=toc; disp(['t = ',mat2str(t),' segundos']);
```

ramifacotha.m

Código, inspirado en en la técnica utilizada por Bemporad y Mignone [1] en el MIQP.m, para aplicar el algoritmo de ramificación y acotamiento utilizando una heurística codiciosa para la ramificación.

```
function [xmin, fmin] = ramifacotha(H, f, vartype, T)
if nargin == 0
   disp('Ramificación Codiciosa'); return
end
[t,s]=size(T); f  = f(:) ;
nx = size(H,1); vartype = vartype(:); nivar = length(vartype);
nha=0.25*nx; zstar = inf*ones(1,nha); xstar = NaN*ones(nx,1);
QPiter  = 0;
global STACK
global STACKSIZE
global STACKCOST
% Initialization of STACK
STACKSIZE = 1;
STACK     = struct('H',H, 'f',f, 'e',0, 'vartype', vartype, ...
       'ivalues',-ones(nivar,1), 'level',0, 'varnum', vartype);
STACKCOST = 0;
%XXXXXXX Main Loop XXXXXXXXX
while STACKSIZE > 0
    subprob = pop;
    if size(subprob.H,1)>0
        QPiter=QPiter+1; lev=subprob.level; nv=length(subprob.vartype);
        ctrl=zeros(nv,1); zctrl=inf;
        for k=1:nv
            if ctrl(k)==0
                xpartic=zeros(nv,1); this=subprob.varnum(k);
                iti=floor((this-1)/s)+1; isi=this-(s*(iti-1));
                ctrts=0;
                for it=1:t-1
                    for is=1:s-1
                        if it==T(iti,isi) & iti==T(it,is) & ctrts==0
                            isi=is; ctrts=1;
                        end
                        if it<=iti & ctrts==1
                            iti=it;
                            break
                        end
                        if ctrts==1
                            break
                        end
                    end
                    if ctrts==1
                        break
                    end
```

```
        end
        thisha(1)=isi+(iti-1)*s;
        for is=isi:s
           if T(iti,isi)==T(iti,is);
              thisha(2)=s*(iti-1)+is;
              thisha(3)=s*(T(iti,isi)-1)+isi;
              thisha(4)=s*(T(iti,isi)-1)+is;
           end
        end
        indha(1)=thisha(1);
        for ih=1:4
            nul=subprob.varnum-thisha(ih); indha(ih)=find(nul==0);
            ctrl(indha(ih))=1;
        end
        iturn=1;
        while iturn==1 || iturn==2
            if iturn==1
                zpartic0=0.5*(subprob.H(indha(1),indha(1))...
                  +subprob.H(indha(1),indha(4))...
                  + subprob.H(indha(4),indha(1))+subprob.H(indha(4),indha(4)))...
                  + subprob.f(indha(1))+subprob.f(indha(4));
                xpartic(indha(1))=1; xpartic(indha(4))=1;
                iturn=2; zpartic=zpartic0;
            else
                zpartic1=0.5*(subprob.H(indha(2),indha(2))...
                  +subprob.H(indha(2),indha(3))...
                  + subprob.H(indha(3),indha(2))+subprob.H(indha(3),indha(3)))...
                  + subprob.f(indha(2))+subprob.f(indha(3));
                xpartic(indha(1))=0; xpartic(indha(4))=0;
                xpartic(indha(2))=1; xpartic(indha(3))=1;
                iturn=0; zpartic=zpartic1;
            end
            if zpartic<zctrl
                q=1; zctrl=zpartic; zctrlmin=zpartic; xctrlmin=xpartic;
                branchvar1min=thisha; branchvar2min=indha;
            elseif zpartic==zctrl
                q=q+1; zctrlmin(q)=zpartic; xctrlmin(:,q)=xpartic ;
                branchvar1min(q,:)=thisha; branchvar2min(q,:)=indha;
end, end, end, end
for iq=1:q
    xctrl=xctrlmin(:,iq);
    branchvar1=branchvar1min(iq,:); branchvar2=branchvar2min(iq,:);
    if ~isempty(xctrl)
        zpi=.5*xctrl'*subprob.H*xctrl+xctrl'*subprob.f+subprob.e;
    end
    if zstar(lev+1)==inf
        zstar(lev+1)=zpi;
```

```
            end
            p = separate(subprob, T, branchvar1, branchvar2, xctrl);
            if zpi<=zstar(lev+1)
                zstar(lev+1) = zpi;
                iset = find(p.ivalues>-1);
                xstar(iset) = p.ivalues(iset);
                if lev==nha-1
                    xestar=xstar;
                end
            end
            if subprob.level ~=nha-1
                cost = 1/(subprob.level+1);
                push(p,cost);
end, end, end, end
disp(['QPiter = ',int2str(QPiter)]);
xmin   = xestar;
fmin   = zstar(nha);
%XXXXXX Subroutines  XXXXXXXX
function subprob = pop %!!!!!!!!
global STACK
global STACKSIZE
global STACKCOST
subprob = STACK(STACKSIZE);
STACKSIZE = STACKSIZE-1;
STACKCOST(end) = [];
function push(element,cost) %!!!!!!!!
global STACK
global STACKSIZE
global STACKCOST
ii = find(STACKCOST>=cost);
if ~isempty(ii)
   i=ii(end)+1 ;
else
   i=1;
end
for j = STACKSIZE:-1:i
   STACK(j+1) = STACK(j);
   STACKCOST(j+1) = STACKCOST(j) ;
end
   STACK(i)     = element;
   STACKSIZE    = STACKSIZE+1;
   STACKCOST(i) = cost;
function p = separate(prob, T, br1, br2, xctrl) %!!!!!!!!
if (length(prob.vartype) >= 1)
    nx = size(prob.H,1); [t,s]=size(T); thisha=br1;
    for ih=1:4
        nul=prob.varnum-thisha(ih); indha(ih)=find(nul==0);
```

```
    end
    others= [1:indha(1)-1,indha(1)+1:indha(2)-1,indha(2)+1:indha(3)-1,...
       indha(3)+1:indha(4)-1,indha(4)+1:nx];
    H00 = prob.H(others, others);
    H01 = prob.H(others, indha(1)); H02 = prob.H(others, indha(2));
    H03 = prob.H(others, indha(3)); H04 = prob.H(others, indha(4));
    H11 = prob.H(indha(1), indha(1)); H22 = prob.H(indha(2), indha(2));
    H33 = prob.H(indha(3), indha(3)); H44 = prob.H(indha(4), indha(4));
    H14 = prob.H(indha(1), indha(4)); H23 = prob.H(indha(2), indha(3));
    p0.H = H00; p1.H = H00;
    f0  = prob.f(others);
    f1  = prob.f(indha(1)); f2  = prob.f(indha(2));
    f3  = prob.f(indha(3)); f4  = prob.f(indha(4));
    p0.f = f0(:)+H02(:)+H03(:); p1.f = f0(:)+H01(:)+H04(:);
    p0.e = prob.e + f2 + f3 + H23 + 0.5 * (H22 + H33);
    p1.e = prob.e + f1 + f4 + H14 + 0.5 * (H11 + H44);
    vartype = [prob.vartype(1:indha(1)-1); ...
         prob.vartype(indha(1)+1:indha(2)-1)-1; ...
         prob.vartype(indha(2)+1:indha(3)-1)-2; ...
         prob.vartype(indha(3)+1:indha(4)-1)-3; ...
         prob.vartype(indha(4)+1:length(prob.vartype))-4];
    p0.vartype = vartype; p1.vartype = vartype;
    ifree = find(prob.ivalues==-1);
    aux = prob.ivalues;
    aux(thisha(1))= 0; aux(thisha(2))= 1; aux(thisha(3))= 1; aux(thisha(4))= 0;
    p0.ivalues = aux ;
    aux(thisha(1))= 1; aux(thisha(2))= 0; aux(thisha(3))= 0; aux(thisha(4))= 1;
    p1.ivalues = aux;
    level = prob.level+1;
    p0.level = level; p1.level = level;
    varnum = [prob.varnum(1:indha(1)-1); ...
         prob.varnum(indha(1)+1:indha(2)-1); ...
         prob.varnum(indha(2)+1:indha(3)-1); ...
         prob.varnum(indha(3)+1:indha(4)-1); ...
         prob.varnum(indha(4)+1:length(prob.varnum))];
    p0.varnum = varnum; p1.varnum = varnum;
    bin = xctrl(br2(1));
    if bin==0
        p=p0;
    else
        p=p1;
    end
else
    error('no more integer variables to branch on');
end
```

ramifacothan__.m

Código máster que permite resolver el programa entero binario de asignación HA mediante el algoritmo de ramificación y acotamiento aplicando una heurística codiciosa para la ramificación. Corresponde éste al caso más sencillo, $n = 2$. Para los demás casos es similar.

```
function ramifacothan2
tic
n=2; t=2*n; s=2*(2*n-1); ts=t*s;
T=feval(@itinhan2); H=feval(@matrizHn2); f=zeros(ts,1);
vartype=1:1:ts;
[xmin,fmin]=ramifacotha(H,f,vartype,T);
xmin=xmin', xmin=reshape(xmin,s,t); xmin=xmin'
A=cell(t,s);
for it=1:t
    for is=1:s
        xts=xmin(it,is);
        switch xts
            case 1, A(it,is)={'A'};
            case 0, A(it,is)={'H'};
end, end, end
disp(' '); disp('     ASIGNACIÓN DE MÍNIMA DISTANCIA'); disp(A);
disp(['fmin =',mat2str(fmin)]);
    d=feval(@disthan2); X=xmin; dt=0;
    for it=1:t
        xts=X(it,1); dts=d(it,T(it,1));
        switch xts % TRAYECTO INICIAL
            case 0, Lts=0;
            case 1, Lts=dts;
        end%switch-1
        dt=dt+Lts; xts=X(it,s); dts=d(T(it,s),it);
        switch xts %TRAYECTO FINAL
            case 0, Lts=0;
            case 1, Lts=dts;
        end, dt=dt+Lts;
        for is=1:s-1 % TRAYECTOS EN EL ITINERARIO
            if X(it,is)==0 && X(it,is+1)==0
                Lts=0;
            elseif X(it,is)==1 && X(it,is+1)==1
                Lts=d(T(it,is),T(it,is+1));
            elseif X(it,is)==1 && X(it,is+1)==0
                Lts=d(T(it,is),it);
            elseif X(it,is)==0 && X(it,is+1)==1
                Lts=d(it,T(it,is+1));
            end, dt=dt+Lts;
        end, end, dmin=dt
t=toc; disp(['t = ',mat2str(t),' segundos']);
```

semidhan__.m

Código máster que permite resolver el programa entero binario de asignación HA mediante el software SDPT3-4.0. Corresponde éste al caso más sencillo, $n = 2$. Para los demás casos es similar.
Se aplica el algoritmo aleatorio de Goemans a la matriz definida positiva resultante. Al final procede un análisis de la factibilidad de las solución mínima encontrada, es decir, se valida que corresponda a una solución factible de problema de asignación original.

```
function [obj,X,y,Z,info,runhist] = semidhan2
tic
n=2; t=2*n; s=2*(2*n-1); m=t*s+1;
T=feval(@itinhan2); d=feval(@disthan2);
blk{1,1} = 's';
blk{1,2} = m;
A=feval(@bloquesAn2);
At=svec(blk(1,:),A(1,:));
C=feval(@matrizCn2);
b=feval(@vectorbn2);
OPTIONS=[];
X0=[];y0=[];Z0=[];
%
[obj,Xbest,y,Z,info,runhist] = sqlp(blk,At,C,b,OPTIONS,X0,y0,Z0);
%
X=Xbest{1}; disp(X);
% Se aplica el algoritmo de aproximación aleatoria de Goemans...
% Se espera que X sea definida positiva...
valores=eig(X); % valores propios deben ser todos positivos...
disp('valores propios'); disp(valores);
Xchol=chol(X);% descomposición de Cholesky...
disp(Xchol);
for i=1:m
    norma(i)=norm(Xchol(:,i));
end
disp(norma);
vm=Xchol(:,m);%Vector columna asociado a variable específica...
tot=100;
for k=1:tot
     u=2*rand(1,m)-1;%Vector fila aleatorio en [-1,+1]...
     signo=u*vm;
     if signo>=0
         signo=0;
     else
         signo=1;
     end
     for it=1:t
        for is=1:s
            j=s*(it-1)+is;
            prod=u*Xchol(:,j);
```

```
        if prod>=0
            prod=0;
        else
            prod=1;
        end
        if prod==signo
            Y(it,is)=0;
        else
            Y(it,is)=1;
        end
    end%for is
end%for it
Z=2*ones(t,s); ctrl=0;
for it=1:t-1
    for is=1:s-1
        if Z(it,is)==2
            Z(it,is)=1; Z(T(it,is),is)=1;
            if Y(T(it,is),is)==Y(it,is)
                ctrl=1;
            end
            for ij=is+1:s
                if T(it,ij)==T(it,is)
                    Z(it,ij)=1; Z(T(it,is),ij)=1;
                    if Y(it,ij)==Y(it,is)
                        ctrl=1;
                    end
                    if Y(T(it,is),ij)==Y(it,ij)
                        ctrl=1;
                    end
                    if Y(T(it,is),ij)==Y(T(it,is),is)
                        ctrl=1;
                    end
                end%if T
            end%for ij
        end%if Y
    end%for is
end%for it
if ctrl==1
    ctrl=0
break
end
dt=0;
for it=1:t
    xts=Y(it,1); dts=d(it,T(it,1));
    switch xts % TRAYECTO INICIAL
        case 0
            Lts=0;
```

```
            case 1
                Lts=dts;
        end%switch
        dt=dt+Lts; xts=Y(it,s); dts=d(T(it,s),it);
        switch xts %TRAYECTO FINAL
            case 0
                Lts=0;
            case 1
                Lts=dts;
        end%switch
        dt=dt+Lts;
        for is=1:s-1 % TRAYECTOS EN EL ITINERARIO
            if Y(it,is)==0 && Y(it,is+1)==0
                Lts=0;
            elseif Y(it,is)==1 && Y(it,is+1)==1
                Lts=d(T(it,is),T(it,is+1));
            elseif Y(it,is)==1 && Y(it,is+1)==0
                Lts=d(T(it,is),it);
            elseif Y(it,is)==0 && Y(it,is+1)==1
                Lts=d(it,T(it,is+1));
            end%if xts
            dt=dt+Lts;
        end%for is
    end%for it
%BÚSQUEDA DE LA DISTANCIA MÍNIMA
    if k==1
        dmin=dt; Xmin=Y;
    elseif dt<dmin
        dmin=dt; Xmin=Y;
    end
end%for k
A=cell(t,s);
for it=1:t
    for is=1:s
        xts=Xmin(it,is);
        switch xts
            case 1
                A(it,is)={'A'};
            case 0
                A(it,is)={'H'};
        end
    end
end
disp(' ');
disp('     VARIABLE X MINIMIZADORA');
disp(Xmin);
disp(' ');
```

```
disp('     ASIGNACIÓN DE MÍNIMA DISTANCIA');
disp(A);
disp(' ');
disp('   DISTANCIA MÍNIMA DE RECORRIDO TOTAL');
disp(['           dmin = ',mat2str(dmin)]);
t=toc;
disp(['Tiempo:        t = ',mat2str(t),' segundos']);
```

Se muestra, además, los códigos para obtener las matrices, bloquesAn2, matrizWn2 y matrizCn2.

```
function A=bloquesAn2
n=2; m1=4*n*(2*n-1)+1;
A1 = cell(1,m1);
for k = 1:m1
    A1{k} = sparse(k,k,1,m1,m1);
end
T=feval(@itinhan2); [t,s]=size(T);
Ctrl=zeros(t,s);
k=0;
m2=3*(m1-1)/4;
A2 = cell(1,m2);
for it=1:t-1;
    for is=1:s-1;
         if Ctrl(it,is)==0
            for ij=is+1:s
                if T(it,ij)==T(it,is)
                    Ctrl(it,is)=1; Ctrl(it,ij)=1;
                    Ctrl(T(it,is),is)=1; Ctrl(T(it,is),ij)=1;
                    i=s*(it-1)+is; j=s*(it-1)+ij;
                    k=k+1;
                    A2{k}=sparse([i,j],[j,i],[1,1],m1,m1);
                    i=s*(T(it,is)-1)+is; j=s*(T(it,is)-1)+ij
                    k=k+1;
                    A2{k}=sparse([i,j],[j,i],[1,1],m1,m1);
                    i=s*(it-1)+ij; j=s*(T(it,is)-1)+is;
                    k=k+1;
                    A2{k}=sparse([i,j],[j,i],[1,1],m1,m1);
                    break
                end %if T(it,ij)==T(it,is)
            end %for ij
         end %if Ctrl
    end %for is
end %for it
A=cat(2,A1,A2);
```

```
function w=matrizWn2
n=2; t=2*n; s=4*n-2; m=t*s+1;
wts=feval(@pesoswtsn2); wtss=feval(@pesoswtssn2);
w=zeros(m,m);
for it=1:t
    for is=1:s
        i=s*(it-1)+is;
        w(i,m)=wts(it,is);
        w(m,i)=w(i,m);
    end%for is
end%for it
for it=1:t
    for is=1:s-1
        i=s*(it-1)+is;
        w(i,i+1)=wtss(it,is);
        w(i+1,i)=w(i,i+1);
    end%for is
end%for it

function C=matrizCn2
n=2; t=2*n; s=4*n-2; m=t*s+1;
w=feval(@matrizWn2);
C=-w;
for i=1:m
    C(i,i)=0;
    for j=1:m
        C(i,i)=C(i,i)+w(i,j);
    end%for j
end%for i
C=0.25*C;
```

Apéndice C
Itinerarios y Distancias

En situaciones reales, donde sea necesario aplicar el programa cuadrático entero binario, lo usual es que el *cliente* proporcione la data sobre la cual se requiere conseguir las variables de decisión que optimicen, o el costo, o el rendimiento. Sin embargo, para el análisis de validación de los códigos confeccionados para resolver dicho programa, de manera exacta o aproximada, se requiere ponerlos a prueba en experimentos computacionales utilizando una data ficticia.
En este sentido, para nuestro trabajo se crearon, para cada valor de n, sendas tablas de itinerario y distancias. Para las instancias de menor dimensión el trabajo puede hacerse sin el auxilio de un procesador digital. Sin embargo, ya desde pequeños valores de n, el trabajo manual se hace harto laborioso. De allí que fue necesario estudiar ambos temas con el propósito de automatizar su confección.

Los Itinerarios

El itinerario para un torneo deportivo pude presentarse como una matriz, o un cuadro, cuyas filas estás determinadas por cada uno de todos los equipos participantes, a cada uno de los cuales se le ha asignado un número, desde 1 en adelante, y cuyas columnas corresponden a los momentos (casillas) en que se realizará cada juego con dos de los equipos participantes.
El tipo de torneo más sencillo es aquel en el cual cada equipo juega una única vez con cada uno de los otros equipos participantes. Este se conoce como *single round robin*. La modalidad de torneo estudiada en este trabajo es el *double round robin*, en el cual cada equipo juega dos veces con cada otro de los equipos oponentes participantes en el torneo. Una vez actuando como visitante y, otra vez, como local. Para efectos prácticos, un *doble round robin* no es más que la concatenación de dos *single round robin*.
En términos combinatorios, el itinerario correspondiente a un *single round robin* puede plantearse como una equipartición del conjunto de todas las parejas de equipos. Es decir, dado el conjunto de todos los pares de equipos para una instancia n dada, esto es, $t = 2n$ equipos, dicho conjunto consta de $\frac{t}{2}(t-1)$ pares. Con dichos pares podemos construir diferentes pareamientos, entendiendo por tales, los conjuntos de $\frac{t}{2}$ pares donde cada equipo aparece una única vez. El total de pareamientos posibles con t equipos es $3 \cdot 5 \cdot 7 \cdots t-1$. Un itinerario de un torneo *single round robin* es un conjunto integrado por $t-1$ pareamientos disjuntos, que unidos, forman el total de pares.
Utilizamos como ejemplo la situación para $n = 3$. El número de equipos participantes es $t = 2n = 2(3) = 6$. El número total posible de pares de equipos es $\frac{t}{2}(t-1) = \frac{6}{2}(6-1) = 3 \cdot 5 = 15$. Éstos aparecen en la siguiente página:

$$
\begin{array}{ccccc}
\{1\ \ 2\} & \{1\ \ 3\} & \{1\ \ 4\} & \{1\ \ 5\} & \{1\ \ 6\} \\
 & \{2\ \ 3\} & \{2\ \ 4\} & \{2\ \ 5\} & \{2\ \ 6\} \\
 & & \{3\ \ 4\} & \{3\ \ 5\} & \{3\ \ 6\} \\
 & & & \{4\ \ 5\} & \{4\ \ 6\} \\
 & & & & \{5\ \ 6\}
\end{array}
$$

Con estos pares se pueden construir $3 \cdot 5 \cdots t-1 = 3 \cdot (6-1) = 3 \cdot 5 = 15$ pareamientos diferentes, a saber:

$$
\begin{aligned}
P_1 &= \{\{1\ \ 2\}, \{3\ \ 4\}, \{5\ \ 6\}\} \\
P_2 &= \{\{1\ \ 2\}, \{3\ \ 5\}, \{4\ \ 6\}\} \\
P_3 &= \{\{1\ \ 2\}, \{3\ \ 6\}, \{4\ \ 5\}\} \\
P_4 &= \{\{1\ \ 3\}, \{2\ \ 4\}, \{5\ \ 6\}\} \\
P_5 &= \{\{1\ \ 3\}, \{2\ \ 5\}, \{4\ \ 6\}\} \\
P_6 &= \{\{1\ \ 3\}, \{2\ \ 6\}, \{4\ \ 5\}\} \\
P_7 &= \{\{1\ \ 4\}, \{2\ \ 3\}, \{5\ \ 6\}\} \\
P_8 &= \{\{1\ \ 4\}, \{2\ \ 5\}, \{3\ \ 6\}\} \\
P_9 &= \{\{1\ \ 4\}, \{2\ \ 6\}, \{3\ \ 5\}\} \\
P_{10} &= \{\{1\ \ 5\}, \{2\ \ 3\}, \{4\ \ 6\}\} \\
P_{11} &= \{\{1\ \ 5\}, \{2\ \ 4\}, \{3\ \ 6\}\} \\
P_{12} &= \{\{1\ \ 5\}, \{2\ \ 6\}, \{3\ \ 4\}\} \\
P_{13} &= \{\{1\ \ 6\}, \{2\ \ 3\}, \{4\ \ 5\}\} \\
P_{14} &= \{\{1\ \ 6\}, \{2\ \ 4\}, \{3\ \ 5\}\} \\
P_{15} &= \{\{1\ \ 6\}, \{2\ \ 5\}, \{3\ \ 4\}\}
\end{aligned}
$$

De aquí surgen los siguientes conjuntos de pareamientos disjuntos que conducen a sendos itinerarios *single round robin*. Cada conjunto consta de $t-1 = 6-1 = 5$ pareamientos.

$$
\begin{aligned}
I_1 &= P_1 \cup P_5 \cup P_9 \cup P_{11} \cup P_{13} \\
I_2 &= P_1 \cup P_6 \cup P_8 \cup P_{10} \cup P_{14} \\
I_3 &= P_2 \cup P_4 \cup P_8 \cup P_{12} \cup P_{13} \\
I_4 &= P_2 \cup P_6 \cup P_7 \cup P_{11} \cup P_{15} \\
I_5 &= P_3 \cup P_4 \cup P_9 \cup P_{10} \cup P_{15} \\
I_6 &= P_3 \cup P_5 \cup P_7 \cup P_{12} \cup P_{14}
\end{aligned}
$$

Por ejemplo, el primer itinerario *single round robin* se construye como sigue:

I_1	P_1	P_5	P_9	P_{11}	P_{13}
	$\{1\ \ 2\}$	$\{1\ \ 3\}$	$\{1\ \ 4\}$	$\{1\ \ 5\}$	$\{1\ \ 6\}$
	$\{3\ \ 4\}$	$\{2\ \ 5\}$	$\{2\ \ 6\}$	$\{2\ \ 4\}$	$\{2\ \ 3\}$
	$\{5\ \ 6\}$	$\{4\ \ 6\}$	$\{3\ \ 5\}$	$\{3\ \ 6\}$	$\{4\ \ 5\}$

Cada pareamiento, P_j, corresponde a una casilla del itinerario. Cada par, $\{a \;\; b\}$ indica que en la casilla j el equipo b es el oponente del equipo a, y viceversa. Queda así este itinerario, como sigue:

I_1	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$
$t=1$	2	3	4	5	6
$t=2$	1	5	6	4	3
$t=3$	4	1	5	6	2
$t=4$	3	6	1	2	5
$t=5$	6	2	3	1	4
$t=6$	5	4	2	3	1

Por último, conociendo dos itinerarios de éstos, I_i e I_j, el correspondiente a un torneo *double round robin* no es más que la concatenación de los mismos, $I_k = I_i \cup I_j$.
El código, que se muestra en la siguiente página, permite crear un itinerario para un *double round robin* para cualquier valor de n.
En lugar de simplemente concatenar los itinerarios I_i e I_j, se puede intercalar las casillas de uno y otro buscando reducir la repetición de resultados $'A'$ o $'H'$ en las matrices asignación resultantes.
Este código se utilizó para crear itinerarios a partir de $n = 3$, los cuales aparecen en el Apéndice A.

```
function pareamitin
tic
n=input('Se introduce el valor de "n", n = ');
t=2*n; l=1; s=2*(2*n-1);
while l<=2
    ctrl=zeros(t,t); It=zeros(t,t-1); k=1;
    while k<=t-1
        cl=0; v=randperm(t);
        for i=1:2:t-1
            if ctrl(v(i),v(i+1))==1
                cl=1; k=k-1;
                break
            end
        end
        for j=1:2:t-1
            if cl==1
                cl=0;
                break
            end
            ctrl(v(j),v(j+1))=1; ctrl(v(j+1),v(j))=1;
            It(v(j),k)=v(j+1); It(v(j+1),k)=v(j);
        end
        k=k+1;
    end
    if l==1
        It1=It;
    else
        It2=It;
    end
    l=l+1;
end
ctrl=zeros(t-1,1);
It=zeros(t,s);
for j=1:t-1
    It(:,2*j-1)=It1(:,j);
    len1=0;
    for k=1:t-1
        if ctrl(k)==0
           dif=It1(:,j)-It2(:,k);
           adif=find(dif==0,t);
           len2=length(adif);
           if len2>=len1
               len1=len2;
               jk=k;
               It(:,2*j)=It2(:,k);
           end
        end
```

```
    end
    ctrl(jk)=1;
end
% It=cat(2,It1,It2);
disp(['Un itinerario para n = ',int2str(n)]);
disp(It);
time=toc;
disp(['tiempo = ',mat2str(time),' segundos']);
```

La Matriz Distancia

Para efecto del experimento computacional, se crea un conjunto de sedes ficticias y se da un valor numérico a la distancia entre cada una de los sitios sedes de los equipos. Para obtener la formulación del problema de asignación local-visitante como un programa cuadrático entero binario, en ningún momento se hizo referencia a que las distancias debiesen verificar la desigualdad triangular. Sin embargo, la práctica obliga a su cumplimiento. Si se observa la fórmula de los pesos de las aristas (4.11), ellos quedaron expresados en término de las distancias, de manera que, para cada tres de ellas, $d(a,b)$, $d(b,c)$ y $d(a,c)$, aparecen en los pesos de la forma, $d(a,b)+d(b,c)-d(a,c)$, $d(a,b)-d(b,c)+d(a,c)$ y $-d(a,b)+d(b,c)+d(a,c)$. Por tanto, si no se cumple la desigualdad triangular, algunos de los pesos podrían resultar negativo. Las matrices que aparecen en el cálculo computacional, tanto en el algoritmo de ramificación y acotamiento, como en el programa semidefinido, tienen en sus diagonales estos pesos o expresiones lineales de los mismos. Si algunos pesos fuesen negativos, se correría el riesgo de que tales matrices no fuesen semidefinidas positivas, lo cual es una exigencia.
Tenemos entonces la obligación de crear lugares ficticios separados por distancias que verifiquen la desigualdad triangular. Como es obvio, para pequeños valores de n, la tarea manual es laboriosa, pero realizable. No así para valores medianos de n. Por tal razón se hace necesario diseñar un código y dejar este trabajo para un procesador digital.
Para cualquiera que sea el número de equipos en el torneo, por tanto de sitios sedes, el análisis pasa por las tres primeras sedes. Se debe obtener $d(1,2)$, $d(2,3)$ y $d(1,3)$, tales que verifiquen la desigualdad triangular. Esto equivale a tener tres números positivos, a, b y c, tales que,

$$\begin{aligned} d(1,2)+d(1,3)-d(2,3) &= a \\ d(1,2)-d(1,3)+d(2,3) &= b \\ -d(1,2)+d(1,3)+d(2,3) &= c \end{aligned}$$

Matricialmente, sería

$$\begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Las soluciones de este sistema son: $x_1 = \frac{1}{2}(a+b)$, $x_2 = \frac{1}{2}(a+c)$ y $x_3 = \frac{1}{2}(b+c)$. Es fácil ver que, cualquiera que sea el orden de crecimiento de los términos a, b y c, se cumple que $Li \leq x_i \leq Ls$, $i = 1, 2$ y 3, con $Li = min\{a,b,c\}$ y $Ls = max\{a,b,c\}$. Entonces, este sistema de ecuaciones permite establecer un orden de magnitud para las distancias, a través del orden de magnitud de a, b y c.
Al incorporar una cuarta sede aparecen nuevas incógnitas, $d(1,4)$, $d(2,4)$ y $d(3,4)$, y tres ternas de ecuaciones de desigualdad triangular. Tomemos cualquiera de estas ternas de ecuaciones,

$$\begin{aligned} d(1,2)+d(1,4)-d(2,4) &= d \\ d(1,2)-d(1,4)+d(2,4) &= e \\ -d(1,2)+d(1,4)+d(2,4) &= f \end{aligned}$$

donde los términos independientes deben ser no negativos: $d, e, f \geq 0$. Como $d(1,2)$ es conocida, se plantea el sistema,

$$\begin{aligned} -d(1,4)+d(2,4) &= e-d(1,2) \\ d(1,4)+d(2,4) &= f+d(1,2) \end{aligned}$$

el cual tiene la forma matricial,

$$\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} e - d(1,2) \\ f + d(1,2) \end{bmatrix}$$

cuyas soluciones son: $y_1 = d(1,2) - \frac{1}{2}(e-f)$ y $y_2 = \frac{1}{2}(e+f)$.
Como debe resultar $y_1 > 0$, la manipulación algebraica de la desigualdad $d(1,2) - \frac{1}{2}(e-f) > 0$ conduce a que, $(e - d(1,2)) < (f + d(1,2))$. Por tanto, al plantear el sistema de la forma,

$$\begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

debe exigirse que $b_1 < b_2$.
Más aún, deben ser tales que $|b_1| < |b_2|$, según puede verse de las soluciones de este sistema: $y_1 = \frac{1}{2}(b_2 - b_1)$ y $y_2 = \frac{1}{2}(b_1 + b_2)$ y la exigencia de que $y_1 = d(1,4)$ y $y_2 = d(2,4)$ deben ser cantidades positivas. También se tiene, de las expresiones de b_1 y b_2, que $b_1 < d(1,2) < b_2$.
Al utilizar la siguiente terna de ecuaciones triangulares, sólo queda $d(3,4)$ como incógnita:

$$\begin{aligned} d(1,3) + d(1,4) - d(3,4) &= g \\ d(1,3) - d(1,4) + d(3,4) &= h \\ -d(1,3) + d(1,4) + d(3,4) &= f \end{aligned}$$

Eligiendo la primera, se resuelve,

$$d(3,4) = d(1,3) + d(1,4) - g$$

Ahora como, $d(3,4) > 0$, debe ser $g < d(1,3) + d(1,4)$.
El grupo de ecuaciones sobrantes será útil para evaluar los resultados conseguidos hasta ahora, debiéndose verificar la desigualdad triangular.

$$\begin{aligned} d(1,2) + d(1,4) &> d(2,4) \\ d(1,3) + d(3,4) &> d(1,4) \\ d(1,4) + d(3,4) &> d(1,3) \end{aligned}$$

Si existe otra sede, el procedimiento es exactamente el mismo que el aplicado para la cuarta sede. Siguiendo estas instrucciones diseñamos el código que se muestra en la siguiente página.

```
function distriang
tic
n=input('Se introduce el valor de "n", n = ');
t=2*n;
% Se eligen términos a,b y c cualesquiera de acuerdo
% con el orden de magnitud deseado:
a=5; b=20; c=50; A1=[1 1 -1;1 -1 1;-1 1 1];
bind1=[a b c]; x=bind1/A1;
dist=zeros(t,t);
dist(1,2)=x(1); dist(1,3)=x(2); dist(2,3)=x(3);
dist(2,1)=x(1); dist(3,1)=x(2); dist(3,2)=x(3);
i=4; ctrl=0; maxctrl=100*t;
while i>=4 && i<=t
    dist(1,i)=0; dist(2,i)=0; dist(i,1)=0; dist(i,2)=0;
    A2=[-1 1; 1 1];
    bind2= [min(x)-.5*dist(1,2) max(x)+.5*dist(1,2)];
    x=bind2/A2;
    dist(1,i)=x(1); dist(2,i)=x(2); dist(i,1)=x(1); dist(i,2)=x(2);
    if  dist(1,2)+dist(1,i)<dist(2,i)
        ctrl=ctrl+1;
        if ctrl==maxctrl
            disp('   ¡¡Pulse Ctrl+C y Haga otro Intento!! ');
            pause
        end
        continue
    end
    i=i+1;
end
ctrl1=0; ctrl2=0;
for i=3:t-1
    j=i+1;   maxctrl=500*t;
    while j>=i+1 && j<=t
        ai=dist(1,i); aj=dist(1,j); ak=ai+aj; ak=round(ak);
        b=randint(1,1,[1,ak-1]);
        dist(i,j)=dist(1,i)+dist(1,j)-b; dist(j,i)=dist(i,j);
        k=1;
        while k>=1 && k<=i-1
            ac=dist(i,j); bc=dist(k,i); cc=dist(k,j);
            if ac+bc<cc || ac+cc<bc || bc+cc<ac
                ctrl1=ctrl1+1;
                ctrl2=1;
                if ctrl1==maxctrl
                    disp('   ¡¡Pulse Ctrl+C y Haga otro Intento!! ');
                    pause
                end
                j=j-1;
            end
```

```
            if ctrl2==1
                ctrl2=0;
                break
            end
            k=k+1;
        end
        j=j+1;
    end
end
disp(['Matriz Distancia para n = ',int2str(n)]);
disp(dist);
time=toc;
disp(['tiempo = ',mat2str(time),' segundos']);
```

Bibliografía

[1] Bemporad, A., Mignone, D. MIQP-Development of a Mixed Integer Quadratic Program Solver for Matlab. *Automatic Control Laboratory of ETH Zürich* Switzerland, 2001
http://control.ethz.ch/ hybrid/miqp/

[2] Easton, K., Nemhauser, G. and Trick, M. The travelling tournament problem: description and benchmarks. *In T. Walsh, editor, Principles and Practice of Constraint Programming, volume 2239 of Lecture Notes in Computer Science*, pages 580–585. Springer, Berlin, 2001.

[3] Easton, K., Nemhauser, G. and Trick, M. Solving the travelling tournament problem: a combined integer programming and constraint programming approach. *In Handbook of Scheduling: Algorithms, Models and Performance Analysis*, Leung, J. Y-T. and Anderson, J. H (Eds), Chapman and Hall, 2004.

[4] Goemans, M. X. and Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42 (1995) 1115–1145.

[5] Helmberg, C. Semidefinite Programming for Combinatorial Optimization. *Konrad-Zuse-Zentrum für Informationstechnik*, Berlin, Enero 2000.
http://www.zib.de/helmberg.

[6] Laurent, M. y Rendl, F. Semidefinite Programming and Integer Programming. *Universität Klagenfurt, Institut für Mathematik*, pp.7-9. Austria, 2002.

[7] The MathWorks, Inc. MatLab R2009b. User Guide.

[8] Ribeiro, C. Sport Scheduling: a tutorial on fundamental problems and applications. *Department of Computer Science* Universidade Federal Fluminense, Brazil, 2010.
http://www.ic.uff.br/ celso/artigas/sport-scheduling

[9] Russell, K. Balancing carry-over effects in round robin tournaments. *Biometrika, 67:127–131,* 1980.

[10] TüTüncü, R., Toh, K. and Todd, M. SDPT3-A Matlab Software Package for Semidefinite-Quadratic-Linear Programming, Version 3.0 (2001)
http://www.math.nus.edu.sg/ mattohkc/index.html
http://www.math.cmu.edu/ reha/sdpt3.html

[11] Suzuka, A., Miyashiro, R., Yoshise, A. and Matsui, T. Semidefinite Programming Based Approaches to Home-Away Assignment Problems in Sports Scheduling. *Mathematical Engineering Technical Reports, Department of Mathematical Informatics*, The University of Tokyo. Japan.
http://www.i.u-tokyo.ac.jp/mi/mi-e.htm

[12] Trick, M. A Schedule-then-Breack Approach to Sport Timetabling *GSIA, Carnegie Mellon,* Pittsburgh
http://mat.gsia.cmu.edu