Universidad de Murcia
Grado en Matemáticas
TRABAJO FIN DE GRADO

Árbitros viajeros y Flujos en redes

por

Damián Mompeán Rueda

Tutor: Alfredo Marín Pérez
Departamento de Estadística e Investigación Operativa

15 de junio de 2016

Declaración de originalidad

Yo, Damián Mompeán Rueda, autor del TFG "Árbitros viajeros y Flujos en redes", bajo la tutela del profesor Alfredo Marín Pérez, declaro que el trabajo que presento es original, en el sentido de que ha puesto el mayor empeño en citar debidamente todas las fuentes utilizadas.

Firmado: ..... Fecha: .....

Nota: En la Secretaría de la Facultad de Matemáticas se ha presentado una copia firmada de esta declaración.

Resumen

Si un comerciante, saliendo de su ciudad, debe visitar un conjunto de ciudades y regresar de nuevo a su ciudad, puede elegir el orden en el que visita dichas ciudades para que la distancia del tour sea lo más corta posible. Esta es la que se conoce como la versión clásica del problema del viajante de comercio que denotaremos TSP (*traveling salesman problem*). Sin embargo, en este trabajo nos centraremos en un caso más general en el que consideraremos que no todas las ciudades deban ser recorridas ni que se tenga que viajar directamente de una ciudad a otra sin pasar por una tercera, lo que se conoce como la versión de Steiner del TSP o STSP.

El otro problema a tratar en este trabajo será el que tantos quebraderos de cabeza da a la liga de béisbol norteamericana en la que se deben asignar todos los partidos de dicha liga a un conjunto de árbitros teniendo en cuenta que cada árbitro recorrerá la distancia entre los estadios donde arbitra en jornadas consecutivas. El TUP (*traveling umpire problem*) trata de hacer la asignación de un calendario deportivo dado a este equipo de árbitros, de manera que la distancia que recorren todos los árbitros a lo largo de toda la liga sea mínima.

Asumiremos, claro está, que nuestro comerciante y nuestros árbitros conocen las distancias entre todas las ciudades o estadios, por lo que, en principio, se encuentran en disposición de encontrar una solución óptima a sus problemas. En la Figura 1 podemos encontrar un ejemplo del TSP en el que dado un conjunto de puntos repartidos al azar nos proponemos dar un tour que pase por todos ellos cuya distancia euclídea total sea mínima. Se pone de manifiesto en dicho ejemplo que no se trata en absoluto de un problema trivial ya que solo con 25 puntos ya existen bastantes posibilidades, en concreto $\frac{24!}{2}$.

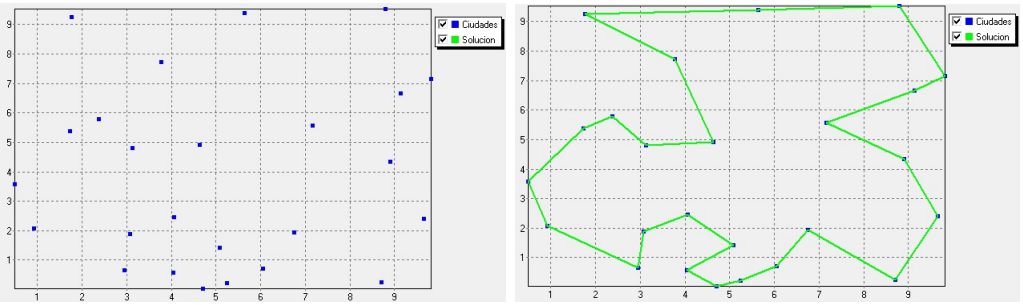

FIGURA 1: Ejemplo del TSP y su solución mediante Xpress

Llegados a este punto tenemos que tanto el STSP como el TUP son dos problemas de optimización. Sin embargo como el conjunto de rutas no es un continuo, no podemos aplicar las técnicas de diferenciación. Estamos ante dos problemas de optimización distintos, relativos a la Optimización Lineal. De hecho, dada la integridad de las variables con las que afrontamos ambos problemas, los incluiremos dentro del ámbito de la optimización lineal entera.

Como vemos, nuestros problemas no se escapan de esta tendencia científica de clasificar los objetos a nuestro alrededor para entenderlos mejor. Tenemos así la tabla periódica en química o la clasificación de los seres vivos en biología desde su reino hasta su especie que nos ayudan a establecer diferencias y similitudes entre todos los seres vivos que conocemos. En matemáticas ocurre exactamente lo mismo ya que con frecuencia clasificamos nuestras herramientas y problemas de multitud de formas como la integridad o la linealidad de un problema de optimización.

Esta idea de usar la clasificación como una herramienta organizativa nos propone clasificar también los métodos expuestos en este trabajo: ¿existe alguna vía de organizar nuestros algoritmos o los problemas a los que los queremos aplicar? Sin embargo desde la comunidad científica no se propuso ninguna vía para responder a esta cuestión hasta la década de los 60 cuando nació la Teoría de la Complejidad Computacional la cual se ocupa de construir esta clasificación. En este campo, se dice que un problema es fácil si se puede desarrollar un algoritmo que lo resuelva en un tiempo polinómico, es decir, que requiera de un número de operaciones que esté acotado por un polinomio en el tamaño de los datos del problema. En este sentido encontramos en la literatura muchos problemas que son fáciles como el problema del camino más corto o el del flujo máximo. Sin embargo, también tenemos otros problemas que han resistido hasta nuestros días los intentos de muchos y grandes científicos de construir un algoritmo eficiente capaz de resolverlos en un tiempo polinómico. Precisamente algunos de estos problemas son el problema del viajante de comercio y el problema del árbitro viajero. Todos estos

fracasos han llevado a la comunidad científica a preguntarse si estos problemas son difíciles en el sentido de que no se pueda dar un algoritmo eficiente que los resuelva. La teoría de la NP-completitud es fruto de esta cuestión aunque por el momento ha sido incapaz de dar respuesta a esta cuestión. No obstante, demuestra que la mayoría de estos problemas que parecen difíciles son equivalentes en el sentido de que si podemos dar un algoritmo polinómico para un problema de esta clase entonces también podemos encontrar un algoritmo polinómico para cualquier otro problema de esta clase. Estos problemas se conocen como problemas NP-completos entre los que ya se encuentran cientos de problemas por lo que se cree que todos ellos son imposibles de resolver en tiempo polinómico. Esto son malas noticias para nuestros propósitos.

Sin embargo la teoría de la NP-completitud tiene algunos aspectos positivos como se expone en Ahuja, Magnanti y Orlin (1993) [1]. Imaginemos que nuestro jefe nos pide que desarrollemos un algoritmo para un problema difícil. Tras varias semanas de trabajo duro no somos capaces de desarrollar un algoritmo eficiente para su resolución: Hay demasiadas posibilidades y tardaríamos años en llegar a la solución. Probablemente en este punto no sea una buena idea decirle a nuestro jefe que no hemos sido capaces de desarrollar su algoritmo.

A pesar de nuestro fracaso, estamos convencidos de que se trata de un problema demasiado difícil y que por muy inteligentes o creativos que fuésemos tampoco podríamos dar un algoritmo eficiente. Por el contrario, parece que no podemos probar nuestra conjetura ya que podría ser tan difícil como el problema original así que tampoco podemos decirle a nuestro jefe que no hemos encontrado un algoritmo eficiente porque no existe.

La teoría de la NP-completitud nos ofrece entonces una alternativa al proveernos de varias técnicas para probar que un problema es tan difícil como la larga lista de problemas NP-completos. Mediante estas técnicas quizás podamos probar que nuestro problema se reduce en tiempo polinomial a otro NP-completo demostrando que pertenece a esta clase. De este modo podremos decirle a nuestro jefe: “No he sido capaz de encontrar un algoritmo eficiente, pero esta larga lista de genios de la historia tampoco”. Esta declaración debería servir al menos para conservar nuestro empleo.

Aquí reside precisamente la importancia del TSP ya que con frecuencia estas técnicas desembocan en reducir mediante un algoritmo polinómico otro problema NP-completo al TSP.

Uno de los primeros hitos del TSP se debe al célebre matemático Euler que en 1759 estudió el problema del tour del caballo de ajedrez en el que se requiere mover esta pieza sobre las 64 casillas de un tablero pasando exactamente una vez por cada casilla, que no es otra cosa que dar un ciclo hamiltoniano sobre el grafo cuyos vértices son estas casillas que son adyacentes cuando el caballo puede moverse de una a la otra empleando solo uno de sus saltos. Sin embargo hasta 1856 no se empezó a considerar estos ciclos en un contexto general cuando Kirkman dio algunas condiciones suficientes para que un grafo poliédrico admitiese dicho ciclo.

El primer uso del término "problema del viajante de comercio" parece remontarse hasta 1932 cuando Tucker (conocido por las condiciones de Kunh-Tucker de minimalidad en optimización no lineal) recordaba habérselo oído a un estudiante de Princeton aunque decía no estar del todo seguro. Por otro lado, el primer trabajo importante sobre este problema data de 1954 y se debe a Dantzig, Fulkerson y Johnson que influyeron de manera significativa al publicar "Solution of a large-scale traveling-salesman problem" el desarrollo de la optimización combinatoria. La instancia más grande resuelta hasta la fecha de manera exacta es de 85900 ciudades.

En este trabajo nos enfrentamos a dos problemas NP-completos pudiéndose reducir, como veremos, el problema del árbitro viajero al TSP. Tras hacer una introducción a la teoría de grafos y explicar en el Capítulo 1 cómo funcionan los métodos que aplicaremos para resolver nuestros problemas, recogeremos y desarrollaremos en el Capítulo 2 los métodos más conocidos aunque rudimentarios para el STSP. En el Capítulo 3 daremos una introducción a la teoría de flujos en redes recopilando resultados bien conocidos como el Teorema del Flujo Máximo o el Teorema del corte Mínimo que nos servirán en el Capítulo 4 para dar mejores métodos para el STSP. Por otro lado en el Capítulo 6 explicaremos la relación entre el TSP y el TUP recogiendo algunos métodos, uno de ellos bien reciente, de marzo de 2016, para el TUP. Culminaremos poniendo a prueba la NP-completitud de estos dos problemas en los Capítulos 5 y 6 haciendo uso de las herramientas que tenemos a nuestra disposición de manera que seamos capaces de observar cómo varían los tiempos de resolución de algunas instancias de estos problemas en función de su tamaño y de la técnica que empleemos.

Abstract

If a salesman, starting from his home city, has to visit a set of cities and then return home, it is plausible for him to choose the order in which he visits those cities so that the total distance of the tour is as small as possible. This is known as the classic version of the *traveling salesman problem* (TSP). However, in this project we will focus in a more general case in which we consider that not all the cities have to be visited and we are not allowed to go directly from some of the cities to others. This is known as the Steiner version of the TSP or STSP.

The other problem that we will study in this project is the one that brings some headaches to the Major League Baseball. In this competition, all the matches are refereed by a set of umpires that have to travel the long distances between the teams venues where they work consecutively. In this situation, the *traveling umpire problem* (TUP) consists of assigning a league schedule to this set of umpires so that the total distances travelled by the umpires is minimum.

Let us assume our salesman and our umpires know the distances between the cities or the venues so that they have all the data necessary to find an optimal solution to their problems. In Figure 1 we can find an example for the TSP in which given a set of points of the plane we want to visit them exactly once in such a way that considering the Euclidean distance between the points, we travel as less as possible. We can see in this example that we are not dealing with a trivial problem because only with 25 points there are $\frac{24!}{2}$ possibilities for this tour.

At this point he have that the STSP and the TUP are both optimisation problems. However, we can not apply our differentiation techniques since the set of trips is not continuous. We are facing a different kind of problem related to the linear programming. In fact, they are included in the field of integer linear programing because of the integrity of the variables that will be used in their resolution.

As we can see, our problems are also influenced by the tendency for scientists to classify the objects in our environment. The periodic chart of elements in Chemistry and the genus/species nomenclature in Biology are examples that show how this classification helps us to understand the different existing relations between the elements of the universe or between the animals and plants. Mathematics and operation research are no different because we often classify our problems and tools by different ways such as the integrity or the linearity in linear programing.

This idea of using classification as an organising tool let us the following question: Is there a way to sort our algorithms or the problems to which we want to apply them? Perhaps the research community had not proposed any way to answer this question until 1960s with the birth of the computer complexity theory which tries to make this classification. In this field, it is said that a problem is *easy* if we can develop an algorithm to solve any instance of the problem in polynomial time (requires a number of operations that is bounded by a polynomial in the size of the input data for the problem). In this respect, we find several easy problems like the shortest path problem or the maximum flow problem. On the other hand, we also have a large list of problems that have resisted all the efforts of thousands of researchers to develop efficient algorithms to solve them in polynomial time. In fact, two of these problems are the TSP and the TUP. With all these unsuccessful attempts the scientist community has started to wonder if these problems are hard in the sense that it is impossible to give an efficient algorithm. The NP-completeness is an outgrowth of these inquiries although it has not been able to give us an answer yet. Nevertheless, using this theory we can show that a high proportion of these problems are equivalent in the sense that if we can find an efficient algorithm for one of them, then we can also find efficient algorithms for all the other problems. We refer to this class of problems as NP-complete problems which now includes thousands of problems. Thus, it is believed that all these problems are hard and it is impossible to find a polynomial algorithm for them. This is bad news about hard problems.

The theory of NP-completeness also has its positive aspects. To capture its usefulness, consider the following story told in Ahuja, Magnanti and Orlin (1993)[1]. Suppose that our boss asks us to develop an algorithm for a complex problem. After several weeks of hard work, we do not come up with an efficient algorithm: There are too many possibilities and in order to check them all we would need several years of computer time. Surely, it is not a good idea just say our boss that the task is too hard for us.

Despite our failure, we are confident that it turns out be a hard problem so it does not matter how smart or creative we are, we would have failed anyway. However, we cannot

prove our conjecture, because this proof could be as difficult as the original problem. Therefore, we cannot walk into our boss office and declare, 'I can't find an efficient algorithm because no such algorithm is possible!"

At this point, the NP-completeness theory gives us an alternative with many techniques for proving that our problem is as hard as the large list of NP-complete problems. With this strategies, perhaps, we will be able to show that our problem reduces in polynomial time to another NP-complete problem proving that it belongs to this class. Finally, we can announce that we cannot find an efficient algorithm, but neither can these list of famous people. This statement might be sufficient to save our job.

Here is were TSP plays a main role because most of this techniques are based in reducing our NP-complete problem to the TSP with a polynomial algorithm.

One of the first milestones in the history of the TSP is due to Euler. In 1759, this clever mathematician studied the problem of the *knight's tour*. This problem consists of move a chess knight exactly once over the 64 squares of a chessboard. This problem is equivalent of find a Hamiltonian cycle for the graph whose vertices are the 64 squares of a chessboard, where two vertices are adjacent if and only if a knight can move from one square to the other in one turn. However, this kind of cycles were not considered in a more general context until 1856 when Kirkman gave a sufficient condition for a polyhedral graph to admit such a cycle.

The first use of the term "traveling salesman problem"seems come from 1932 when Tucker (known for the conditions of Kuhn-Tucker for minimality in Nonlinear Programing) remembered hearing of it from a student of Princeton University although he was not sure. On the other hand, the first important work about this problem is "Solution of a large-scale traveling-salesman problem"published in 1954. With this work, its authors Dantzig, Fulkerson and Johnson influenced significantly the development of the combinatorial optimization. The greatest instance of the TSP ever solved has 85900 cities.

In this project we will face up two NP-complete problems because, as we will see, some instances of the TUP can be reduced to instances of the TSP. After introduce in Chapter 1 the graph theory and explaining our tools for solving problems, we will present and develop in Chapter 2 the most known formulations for the STSP. In Chapter 3 we will give an introduction to the network flows theory including well-known results

as the feasibility theorem or the minimum cut theorem that will allow us to give better formulations in Chapter 4. Moreover, in Chapter 6 we will explain the relationship between the TSP and the TUP and we will present some of the strategies for the TUP, one of them from March 2016. Finally we will solve some instances in Chapters 5 and 6 to show how the computer times depends on the size of the instance and the strategy chosen.

Índice general

1. Introducción **1**
- 1.1. Teoría de grafos . . . 1
- 1.2. Algoritmos y resolución . . . 4
- 1.3. Cotas y formulaciones . . . 5

2. Primeras formulaciones para nuestros problemas **7**
- 2.1. Formulaciones del TSP . . . 7
  - 2.1.1. Formulación clásica del TSP . . . 8
  - 2.1.2. Formulación en etapas . . . 9
- 2.2. Formulaciones del STSP . . . 9
  - 2.2.1. Formulación clásica del STSP . . . 10
  - 2.2.2. Formulación en etapas del STSP . . . 12

3. Flujos en redes **17**
- 3.1. Introducción . . . 17
- 3.2. El Teorema de Factibilidad y el Teorema del Corte Mínimo . . . 18
  - 3.2.1. Teorema del Corte Mínimo . . . 18
  - 3.2.2. Teorema de Factibilidad . . . 19

4. Formulaciones basadas en flujos **23**
- 4.1. Flujo simple . . . 23
- 4.2. Multiflujo . . . 27

5. Resolución de algunas instancias del STSP **31**
- 5.1. Tablas de resultados . . . 31

6. El problema del árbitro viajero **35**
- 6.1. Enunciado . . . 36
- 6.2. Formulación . . . 37
  - 6.2.1. Refuerzo . . . 39
  - 6.2.2. Simetría . . . 41
- 6.3. Un último procedimiento . . . 42
  - 6.3.1. Formulación . . . 42
  - 6.3.2. Desigualdades válidas . . . 45
- 6.4. Resolución de algunas instancias del TUP . . . 48

7. Conclusiones y trabajo futuro **51**

Capítulo 1

Introducción

En este capítulo introduciremos la teoría básica con la que enfocaremos nuestros problemas: Teoría de grafos y optimización lineal entera. De este modo nos encontraremos en los próximos capítulos en disposición de dar algunos resultados teóricos y herramientas que nos permitan calcular explícitamente soluciones para el STSP. No obstante, si uno entiende por resolver el problema dar un algoritmo eficiente que nos permita encontrar soluciones, tengamos presente que ni el STSP ni el TUP han sido aún resueltos en general. De hecho el TSP está reconocido como un problema NP-completo,como se puede ver en Garey y Johnson, 1979 [6], siendo uno de los problemas más conocidos de esta clase mientras que el TUP se puede particularizar en este último como veremos en el Capítulo 6.

1.1. Teoría de grafos

Una buena manera de plantear el problema es usar la Teoría de Grafos. Esto nos proporciona un buen enfoque del problema y nos permite distinguir más claramente entre las dos versiones del problema del viajante de comercio:

TSP: Sea $G = (V, E)$ un grafo completo no dirigido y c_e una función estrictamente positiva que nos indica el coste asociado a atravesar cada arista. En esta situación, el TSP trata de encontrar el ciclo hamiltoniano de coste mínimo.

Como vemos en esta definición buscamos un ciclo para resolver el TSP. Esto se debe a que si las distancias entre las ciudades satisfacen la desigualdad triangular, la solución óptima a este problema será un ciclo hamiltoniano.

Cuando lidiamos con problemas de rutas en la vida real, es más frecuente encontrarse con que solo son requeridos una parte de los nodos $V_R \subset V$ y nuestro grafo no es completo. Además, es probable que no precisemos un ciclo sino un paseo cerrado que incluya a estos V_R una vez o más y no necesariamente sea hamiltoniano. De este modo podemos atravesar aristas y nodos más de una vez en esta variante que llamaremos, siguiendo los autores Cornuéjols (1985) [2], Naddef (2002) [9] y Padberg (1991) [12], variante de Steiner o STSP.

STSP: Sea $G = (V, E)$ un grafo no dirigido conexo, c_e una función estrictamente positiva que nos indica el coste asociado a atravesar sus aristas y $V_R \subset V$. En esta situación, el STSP trata de encontrar el paseo cerrado de coste mínimo que contiene a todos los nodos de V_R al menos una vez.

Cabe aclarar que se puede convertir una instancia del STSP en otra equivalente del TSP calculando los caminos más cortos entre cada par de nodos, como se puede ver en Cornuéjols (1985) [2] deduciéndose la NP-completitud del STSP. Sin embargo, a la hora de resolver el problema tendremos una variable por cada arista de manera que esto aumentaría el número de variables lo cual es un inconveniente a la hora de resolver el problema. Por esta razón introduciremos y analizaremos formulaciones específicas para el STSP.

El proceso de dicha conversión consiste en calcular los caminos más cortos entre cada par de nodos de V_R aplicando un algoritmo como el de Johnson y sustituir en nuestro grafo el conjunto de nodos V por V_R colocando además una arista entre cada par de nodos con una longitud equivalente a la del camino más corto que los une (recordemos que dicho camino existe por la conexión del grafo).

A continuación resolvemos el TSP sobre este grafo y traducimos la solución obtenida al grafo inicial yuxtaponiendo los caminos correspondientes a las aristas del tour solución del TSP sobre V_R. Como esta transformación es polinómica deducimos que al ser el TSP NP-completo, el STSP también lo es.

Tenemos a nuestra disposición algunos resultados que garantizan que un grafo sea hamiltoniano. Sin embargo se desconoce una caracterización que podamos aplicar en nuestros problemas. Por esta razón, no se pierde ninguna propiedad esencial en la generalización al STSP y no nos resultará mucho más costoso resolver instancias del STSP.

Las estrategias para ambos problemas, como veremos en las formulaciones de los Capítulos 2 y 4 serán análogas. Para el TSP consistirán en buscar subgrafos en los que los nodos tengan grado exactamente dos, mientras que para el STSP buscaremos un subgrafo del multigrafo resultante al copiar las aristas de nuestro grafo un número suficiente de veces en el que los nodos de V_R tengan grado mayor que uno. Además exigiremos que dichos subgrafos sean conexos, lo cual supondrá un problema ya que resulta difícil traducir esta propiedad a una formulación. Por último, en el TSP impondremos que el grado de todos los nodos sea exactamente dos para que la solución sea un ciclo, mientras que en el STSP impondremos que todos los nodos tengan grado par. Esto último equivale a que el multigrafo sea euleriano:

Proposición 1.1. *Un multigrafo conexo con todos los nodos de grado par es euleriano.*

Demostración. Consideremos de entre todos los paseos del grafo que no repiten aristas uno de longitud máxima $(x_1, (x_1, x_2), ..., x_n)$. Dicho paseo debe ser cerrado. De lo contrario, podríamos ampliar el paseo ya que la cantidad de aristas del paseo incidentes en el último nodo sería impar y por tanto quedaría al menos una más. Veamos también por reducción al absurdo que debe ser euleriano. Supongamos que existe alguna arista (x_i, x_j) no perteneciente a dicho paseo. Si alguno de estos nodos pertenece al paseo, digamos x_i, como es cerrado podemos reordenarlo para que empiece y acabe en x_i para así yuxtaponerlo con $(x_i, (x_i, x_j), x_j)$ obteniendo un nuevo paseo más largo. Si ninguno de estos nodos está en el paseo, tomamos una cadena que una x_i con algún otro nodo del paseo. En dicha cadena deben existir dos nodos adyacentes de manera que uno pertenezca al paseo y el otro no por lo que la arista que los une no estaría en el paseo volviendo de nuevo al caso anterior. □

Este resultado se puede adaptar al caso de multigrafos dirigidos con una demostración análoga.

Proposición 1.2. *Un multigrafo $G = (V, A)$ dirigido conexo en el que para cada nodo i se satisfaga que*

$$\sum_{(i,j)\in A} n_{ij} = \sum_{(j,i)\in A} n_{ji},$$

siendo n_{ij} el número de copias del arco $(i, j) \in A$, es euleriano.

1.2. Algoritmos y resolución

En todo problema de matemáticas el primer paso para su resolución es conseguir extraer la naturaleza del problema en un ambiente apropiado, es decir, conseguir plasmarlo en papel u ordenador. Este proceso en Optimización Lineal se conoce como formular.

En la literatura encontramos varias maneras de formular el STSP, en este trabajo daremos en el Capítulo 2 unas primeras formulaciones que resultarán no ser muy eficientes. Más tarde, tras introducir en el Capítulo 3 una teoría de flujos en redes, introduciremos en el Capítulo 4 otras formulaciones que nos permitirán resolver instancias del STSP más grandes recogidas en el Capítulo 5. Una vez planteadas estas formulaciones, trataremos de compararlas y así determinar cuál es la mejor formulación teniendo en cuenta, claro está, el método para resolverlas. En el Capítulo 6 expondremos la relación existente entre el TUP y el TSP exponiendo dos formulaciones para el TUP.

El método a aplicar para resolver nuestras formulaciones consistirá en combinar el algoritmo del símplex con el algoritmo de ramificación y acotación ya que nuestras variables son enteras. Dicho procedimiento consiste en resolver el problema sin tener en cuenta la integridad de las variables. Denotaremos a esta variante del problema como su relajación lineal. A continuación, si una variable del óptimo de dicha relajación lineal toma un valor que no sea entero, dividiremos el problema en dos subproblemas restringiéndonos por un lado a que esta variable sea menor o igual que la parte entera del valor que tomó en el óptimo anterior, y por otro a que sea mayor o igual que dicha parte entera más uno. De este modo la solución anterior quedaría excluida de ambos subproblemas.

Este proceso se conoce como ramificar ya que a medida que dividimos el problema en subproblemas, podemos seguir obteniendo soluciones que tampoco sean enteras de manera que seguiríamos ramificando el problema en un árbol cuyas ramas acaben en soluciones enteras. Se trata de un proceso que puede ser largo ya que en un problema grande existen demasiadas posibilidades para las soluciones. Para acortar este proceso entrará en juego la parte de acotación.

En esta segunda parte, tendremos que guardar la mejor solución entera que vayamos encontrando para usarla como cota. Usaremos dicha cota para compararla con las soluciones que no sean enteras antes de ramificarlas en dos subproblemas. Si resulta que una de estas soluciones no enteras es peor que la cota, nos olvidaremos de esta parte del árbol

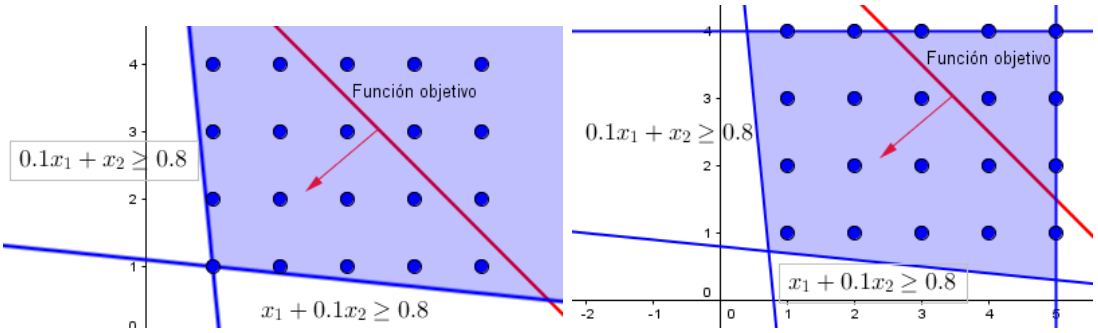

FIGURA 1.1: Comparación de dos formulaciones para un mismo problema

y no ramificaremos, pues si lo hiciésemos estaríamos introduciendo nuevas restricciones y obtendríamos soluciones aún peores.

1.3. Cotas y formulaciones

Debido a que para el STSP presentaremos varias formulaciones, vamos a explicar en esta sección cómo podemos comparar dos formulaciones para determinar cuál es más conveniente.

Cuando obtenemos un óptimo de una relajación lineal mediante el algoritmo del símplex, dicha solución es un punto extremo del conjunto factible de la relajación. Si dicho conjunto fuese la envolvente convexa de las soluciones enteras del problema inicial, nos aseguraríamos la integridad de dicho óptimo. El problema es que no siempre somos capaces de ajustar nuestra formulación para que la relajación lineal sea la envolvente convexa de las soluciones enteras por lo que probablemente obtendremos óptimos no enteros.

Por consiguiente, una forma de comparar dos formulaciones sería estudiar cuál de las dos se ajusta más a la mencionada envolvente convexa. Sin embargo, como solo ramificamos los óptimos lineales, lo importante para que una formulación sea eficiente es que no existan muchos óptimos lineales no enteros que ramificar. Veamos un ejemplo:

Supongamos que nos proponemos minimizar la función $x_1 + x_2$ en el conjunto de $x_1 \in \{1, \ldots, 5\}$, $x_2 \in \{1, \ldots, 4\}$ y que para ello disponemos de las dos formulaciones representadas en la Figura 1.1.

La primera formulación aparentemente es bastante mala ya que solo consta de dos restricciones y el poliedro de la relajación lineal resultante no está ni siquiera acotado. Sin embargo, como el óptimo de dicha relajación lineal es entero, a la hora de resolver el problema obtendríamos directamente el óptimo (1,1) sin necesidad de ramificar. Por otro lado, la segunda formulación pese a acotar el poliedro con cuatro restricciones, resultaría ser peor en la práctica ya que el simplex sobre la relajación lineal nos devolvería el punto (0.8,0.8) haciendo preciso hacer uso del algoritmo de ramificación y acotación.

En suma, cuando comparemos dos formulaciones, estudiaremos si los óptimos de la relajación lineal de una formulación pertenecen al conjunto factible de la otra y viceversa. Para ello nos bastará con determinar si alguno de los conjuntos factibles de las correspondientes relajaciones lineales está incluido en el otro.

Capítulo 2

Primeras formulaciones para nuestros problemas

2.1. Formulaciones del TSP

Nuestro objeto de estudio será el STSP pero antes trataremos de familiarizarnos con algunas formulaciones del TSP. Para ello introduciremos dos formulaciones bien conocidas de este problema. Podremos observar en ellas que, al igual que ocurrirá en el STSP, el principal problema será garantizar la conexión ya que al ser nuestras variables relativas a las aristas, será fácil exigir que el grado de cada nodo en la solución sea dos que equivaldría, una vez asegurada la conexión, a que la solución fuese un ciclo hamiltoniano.

A la hora de introducir formulaciones, para imponer restricciones sobre nuestras aristas denotaremos como $\delta(S)$ el conjunto de aristas que tienen un extremo en $S \subset V$ y el otro en su complementario. A este conjunto se denomina cociclo de S. Además acortaremos la expresión $\delta(\{i\})$ por $\delta(i)$.

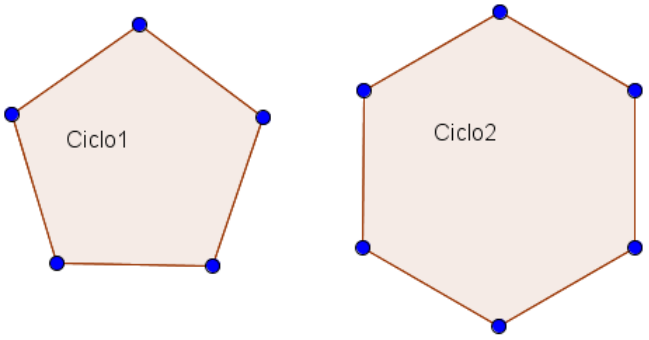

FIGURA 2.1: Sin la restricción (2.2) podemos obtener ciclos disjuntos que no constituirían una solución.

2.1.1. Formulación clásica del TSP

La formulación más común para el TSP se debe a Dantzig (1954)[3]:

$$\begin{aligned} & \text{mín} \sum_{e\in E} c_e x_e & & \\ s.a \quad & \sum_{e\in\delta(i)} x_e = 2 & (\forall i \in V) & \quad (2.1) \\ & \sum_{e\in\delta(S)} x_e \geq 2 & (\forall S \subset V : 2 \leq |S| \leq |V|/2) & \quad (2.2) \\ & x_e \in \{0,1\} & (\forall e \in E). & \end{aligned}$$

Aquí x_e denota a una variable binaria que toma el valor uno si la arista e pertenece al ciclo solución. En esta formulación la restricción (2.1) garantiza que la solución esté formada por uno o varios ciclos que contengan a todos los nodos. Si además imponemos (2.2) la solución será conexa y obtendremos por tanto un único ciclo hamiltoniano evitando casos como el de la Figura 2.1 en la que tenemos un grafo que satisface (2.1) pero que al no ser conexo no puede constituir una solución para el TSP.

Como decíamos, el problema de esta formulación es que relativas a (2.2) hay un número de restricciones del orden de $2^{|V|}$. Este crecimiento exponencial hace que tengamos problemas a la hora de su resolución con software y no seamos capaces de resolver instancias del problema grandes.

Veamos a continuación una formulación alternativa y estudiemos si es más eficiente a la hora de afrontar el problema de la conexión.

2.1.2. Formulación en etapas

La conocida como *time staged formulation* fue propuesta por Vadja (1961)[16] y Houk (1980)[7] independientemente. En esta formulación introducimos una nueva variable que nos indica el orden en el que debemos recorrer las aristas del tour. En este sentido, tenemos que $r_{ij}^k \in \{0,1\}$ toma el valor uno si la arista (i,j) es la k-ésima en ser atravesada en el tour y además se atraviesa en sentido $i-j$. Esto nos da una formulación que aparte de ser más manejable a la hora de su implementación, aborda con un menor número de restricciones el problema de la conexión. Denotemos con n el tamaño de V:

$$
\begin{array}{lll}
\text{mín} & \sum_{i=2}^{n} c_{1i} r_{1i}^1 + \sum_{k=2}^{n-1} \sum_{j=2}^{n} c_{ij} r_{ij}^k + \sum_{i=2}^{n} c_{i1} r_{i1}^n & \\
s.a & \sum_{j=2}^{n} r_{1j}^1 = 1 & (2.3) \\
& \sum_{j=2}^{n} r_{j1}^n = 1 & (2.4) \\
& \sum_{k=1}^{n-1} \sum_{i \neq j} r_{ij}^k = 1 \qquad (2 \leq i \leq n) & (2.5) \\
& \sum_{j \neq i}^{n} r_{ij}^k = \sum_{j \neq i}^{n} r_{ij}^{k+1} \qquad (i \geq 2; 2 \leq k \leq n-1) & (2.6) \\
& r_{ij}^k \in \{0,1\} \qquad (1 \leq i,j,k \leq n; i \neq j) &
\end{array}
$$

Las restricciones (2.3) y (2.4) aseguran que el viajante parte y llega al nodo uno. Esto agilizará los cálculos ya que una misma solución puede empezar a recorrerse desde cualquier nodo. Debido a que fijado i de V y $k \leq n$ el conjunto $\{r_{ij}^k\}_{j \neq i}$ se corresponde con el de las aristas que se salen de i hacia otro nodo y se recorren en la etapa k-ésima del tour solución, la restricción (2.5) asegura que nuestro comerciante llega a cada nodo exactamente una vez, mientras que (2.6) impone la condición de que si llega a un nodo en una etapa, entonces sale de él en la siguiente.

Observamos que en esta formulación tenemos del orden de n^3 variables y n^2 restricciones lo que hace más manejable el problema a la hora de su resolución. Sin embargo la cota de la relajación lineal es peor que en la formulación clásica (2.1.1), lo que nos llevará a una ramificación mayor.

2.2. Formulaciones del STSP

Como decíamos, cuando afrontamos un problema de rutas, puede ocurrir que convenga pasar más de dos veces por un mismo nodo o incluso que nuestro grafo no sea hamiltoniano. Por esto, a continuación comenzaremos con el estudio de la versión de

Steiner del TSP, presentando en este capítulo dos formulaciones análogas a las formulaciones clásica y en etapas de los Apartados 2.1.1 y 2.1.2 con unas adaptaciones para el STSP recogidas en Lechford (2013) [8] añadiendo pruebas de su validez.

2.2.1. Formulación clásica del STSP

Esta formulación es análoga a la vista en la sección anterior con algunas modificaciones propuestas por Fleischman (1985)[4] para adaptarla al STSP:

$$\begin{aligned}
\text{mín} \quad & \textstyle\sum_{e\in E} c_e x_e & & \\
s.a \quad & \textstyle\sum_{e\in\delta(i)} x_e = 2z_i & (\forall i \in V) & (2.7)\\
& \textstyle\sum_{e\in\delta(S)} x_e \geq 2 & (\forall S \subset V : S \cap V_R \neq \emptyset, V_R \cap S^c \neq \emptyset) & (2.8)\\
& x_e \in \mathbb{Z}_+ & (\forall e \in E) & \\
& z_i \in \mathbb{Z} & (\forall i \in V). &
\end{aligned}$$

Introducimos en esta formulación las variables z_i que al ser enteras, en la restricción (2.7) imponen que el grado de todos los nodos sea par. Denominaremos a partir de ahora como **subgrafo solución** al subgrafo de G formado por las aristas de la solución y los nodos de V que tienen grado mayor o igual que uno en esta solución. Dicho subgrafo solución será euleriano si es conexo debido a (2.7). Solo nos falta entonces asegurar su conexión y que contiene a todo V_R, lo cual se da gracias a (2.8). Veamos que efectivamente obtenemos soluciones del STSP al resolver esta formulación:

Proposición 2.1. *La formulación clásica es una formulación válida para el STSP.*

Demostración. Sea (V', E') el subgrafo solución con $E' = \{x_{ij} : x_{ij} = 1\}$ y V' el conjunto de nodos no aislados al considerar únicamente estas aristas. Si demostramos que es conexo, por (2.7) será euleriano y si además $V_R \subset V'$ podremos dar un paseo que recorra todas sus aristas y por tanto que pase por todos los nodos de V_R. En efecto, dado i de V_R el conjunto $\{i\}$ satisface las condiciones de la restricción (2.8) por lo que tendríamos al menos dos aristas de $\delta(i)$ en E' deduciéndose que i pertenece a V'. Para ver que es conexo supongamos por reducción al absurdo que existe una partición (V_1, V_2) en dos componentes conexas distintas de V de manera que ambas contengan nodos requeridos (al considerar el grafo solución óptima si solo hay nodos no requeridos en una componente conexa deberán ser nodos aislados por la optimalidad y no serán considerados en este subgrafo). En tal caso por (2.8) deberá haber al menos dos aristas

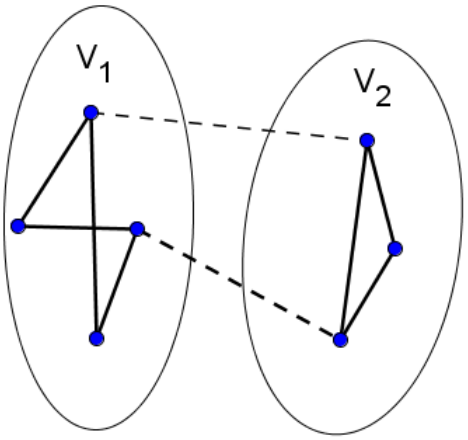

FIGURA 2.2: La restricción (2.8) asegura la conexión

entre V_1 y V_2, lo que contradice que sean componentes conexas distintas como muestra la Figura 2.2. □

Notemos que en esta formulación los nodos pueden ser visitados más de una vez. Las aristas, en principio, también podrían ser atravesadas varias veces. Sin embargo veamos a continuación un resultado que restringe este hecho y que será clave en el planteamiento de futuras formulaciones:

Lema 2.2 (Letchford (2013) [8])**.** *En una solución óptima del STSP, ninguna arista se atravesará más de una vez en el mismo sentido.*

Demostración. En primer lugar, si apareciera una arista repetida 3 veces o más, al eliminar 2 de ellas el grafo conservaría su conexión y la paridad en los grados de todos sus nodos. Por esta razón si eliminamos un par de ellas mejoraríamos la solución. Lo mismo ocurriría si una arista estuviera 2 veces y al eliminarla el grafo siguiese siendo conexo. Por último, si una arista aparece dos veces pero al retirarla desconectamos el subgrafo solución, en nuestra solución deberá ser recorrida en sentidos opuestos (de una primera componente conexa a la otra y luego de esta a la primera) como muestra la Figura 2.3. □

Como corolario obtenemos que las variables x_e de nuestra formulación están acotadas por 2. De este modo podemos incluir esta cota como restricción para reducir el conjunto de soluciones permitiendo a nuestro software encontrar soluciones en menos tiempo.

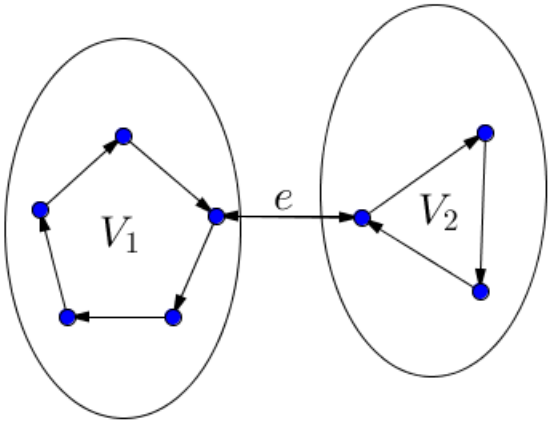

FIGURA 2.3: La arista e debe ser atravesada en sentidos opuestos.

El problema de esta formulación vuelve a ser el mismo que el de la del TSP: el número de restricciones relativas a (2.8). Así que trataremos de dar formulaciones alternativas que solucionen de manera más eficiente el problema de la conexión.

Al considerar el grafo como no dirigido, estamos perdiendo información ya que del Lema 2.2 no solo se extrae que no atravesaremos una arista más de dos veces sino que además si la atravesamos dos veces será en direcciones opuestas. Por esta razón en lo que sigue, consideraremos el grafo dirigido (V, A) donde el conjunto de aristas E se reemplaza por dos arcos dirigidos en A, uno en cada dirección. Haciendo uso de este resultado, consideraremos las variables relativas a estos arcos como binarias decidiendo si el arco (i, j) pertenece o no al tour. Siguiendo la notación de algunos autores como Letchford (2013) [8], para cada $S \subset V$ denotaremos como $\delta^+(S)$ al conjunto de arcos que parten de un nodo de S y llegan a otro de su complementario S^c y como $\delta^-(S)$ los que tienen sentido opuesto a estos. Para simplificar la notación escribiremos $\delta^+(i)$ o $\delta^-(i)$ para denotar a $\delta^+(\{i\})$ y $\delta^-(\{i\})$ ilustrados en la Figura 2.5.

Cabe aclarar que aunque vamos a transformar nuestro grafo en un grafo equivalente no dirigido, nuestros resultados y formulaciones no son aplicables a la versión dirigida del STSP ya que en ella el Lema 2.2 no siempre es cierto como muestra la Figura 2.4.

2.2.2. Formulación en etapas del STSP

En esta sección adaptaremos la formulación vista en el apartado anterior al STSP habiendo transformado nuestro grafo en un grafo dirigido. El primer problema que tendremos es que ahora el número de aristas del paseo no será conocido como en el caso del TSP que era igual a n. Sin embargo, el Lema 2.2 ya establece que no habrá más de

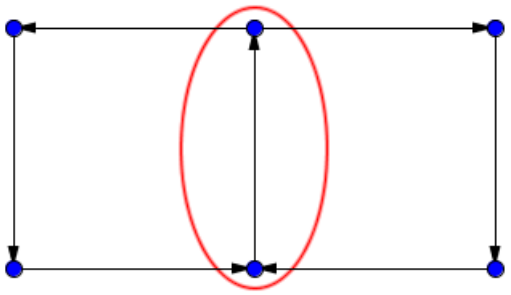

FIGURA 2.4: En el caso dirigido puede que se deban repetir arcos

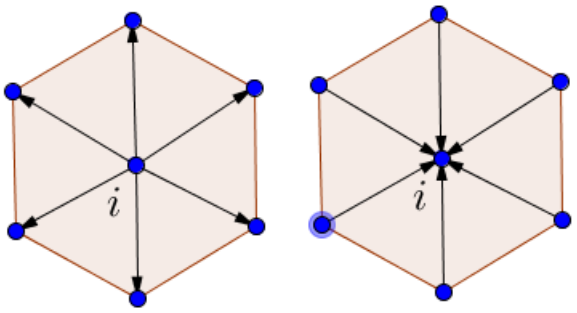

FIGURA 2.5: $\delta^+(i)$ y $\delta^-(i)$

$2|E| = |A|$ arcos en dicho paseo. Como esta cota es algo grande, en esta sección además de presentar la formulación daremos otra cota ya que de ella dependerá el número de variables:

$$\begin{aligned}
\text{mín} \quad & \textstyle\sum_{k=1}^{|A|} \sum_{a\in A} c_a r_a^k & & \\
s.a \quad & \textstyle\sum_{a\in\delta^+(1)} r_a^1 = 1 & & (2.9)\\
& r_a^1 = 0 & (\forall a \notin \delta^+(1)) & (2.10)\\
& \textstyle\sum_{k=1}^{|A|} \sum_{a\in\delta^+(1)} r_a^k = \sum_{k=1}^{|A|} \sum_{a\in\delta^-(1)} r_a^k & & (2.11)\\
& \textstyle\sum_{k=1}^{|A|} \sum_{a\in\delta^+(i)} r_a^k \geq 1 & (\forall i \in V_R) & (2.12)\\
& \textstyle\sum_{a\in\delta^+(i)} r_a^{k+1} = \sum_{a\in\delta^-(i)} r_a^k & (\forall i \neq 1; 1 \leq k \leq |A|-1) & (2.13)\\
& r_a^k \in \{0,1\} & (\forall a \in A, 1 \leq k \leq |A|). &
\end{aligned}$$

Las restricciones (2.9) y (2.10) aseguran que partimos únicamente del nodo uno y (2.11) que salimos y llegamos a este nodo el mismo número de veces. Esto también se satisface en el resto de nodos por (2.13), pero esta restricción además asegura que si llegamos a un nodo en una etapa, saldremos de él en la siguiente. Por último, la restricción (2.12) asegura que visitemos los nodos requeridos.

Además de estas restricciones hemos introducido dos desigualdades válidas para el problema de modo que se reduzca el conjunto factible de la relajación lineal y se obtengan mejores cotas:

$$\begin{aligned}
& \textstyle\sum_{a\in A} r_a^k = 1 \quad (\forall 1 \leq k \leq |V_R|), & (2.14)\\
& r_a^{|A|} = 0 \qquad (\forall a \notin \delta^-(1)). & (2.15)
\end{aligned}$$

La restricción (2.14) impone que se recorra exactamente una arista al menos durante las primeras $|V_R|$ etapas y (2.15) que en la etapa $|A|$ no se recorran aristas que no lleguen al primer nodo.

Veamos que obtenemos soluciones para el STSP:

Proposición 2.3. *La formulación en etapas es una formulación válida para el STSP.*

Demostración. Deducimos de (2.13) que

$$\sum_{k=2}^{|A|} \sum_{a\in\delta^+(i)} r_a^k = \sum_{k=1}^{|A|-1} \sum_{a\in\delta^-(i)} r_a^k \quad (\forall i \neq 1),$$

por lo que aplicando (2.10), (2.11) y (2.15) obtenemos que

$$\sum_{k=1}^{|A|} \sum_{a\in\delta^+(i)} r_a^k = \sum_{k=1}^{|A|} \sum_{a\in\delta^-(i)} r_a^k \quad (\forall i \in V). \tag{2.16}$$

Además por (2.12) todo V_R está en la solución así que solo falta ver la conexión para poder dar un circuito euleriano que contenga a V_R.

Por un lado, el primer nodo está conectado con todos los nodos de la solución ya que si cierto arco sale (o llega) de un nodo i en la etapa k-ésima, podremos construir un camino con los k (o $k-1$) primeros arcos de la solución que los una debido a (2.9), (2.13) y (2.14). Para ver que todos los nodos están conectados con 1 denotemos con L la longitud de la solución, es decir, el entero tal que existe $r_{ij}^L = 1$ y para todo $k \geq L+1$ se tiene que $r_a^k = 0$. Si dicho j es distinto de 1, como con los L primeros arcos podemos construir un camino que una 1 y j se violaría (2.16) pues habría un arco más en $\delta^-(j)$ que en $\delta^+(j)$. Así que si i está en la solución con $r_{ij}^k = 1$ con los últimos $L-k$ arcos podemos unir i con 1. □

Ahora introduciremos una cota para el número de variables de manera que reduzcamos el conjunto de soluciones. El resultado del que obtendremos esta cota está basado en el siguiente lema:

Lema 2.4 (Letchford (2013) [8])**.** *Sea H un grafo conexo con k nodos y ℓ aristas. Si $\ell > 2(k-1), \exists C$ un ciclo de manera que el grafo resultante al extraer de H las aristas de C sigue siendo conexo.*

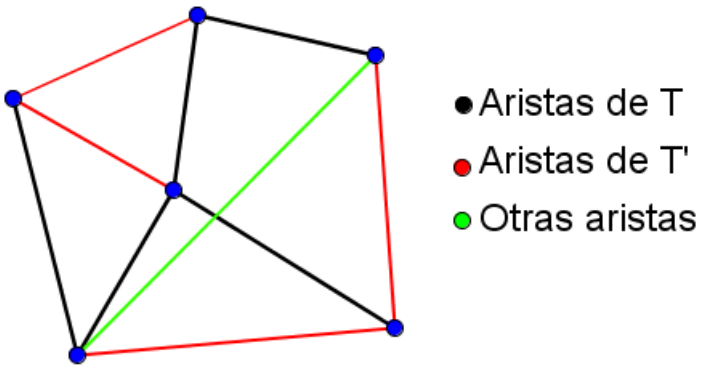

FIGURA 2.6: Ilustración del Lema 2.4

Demostración. Sea T un árbol generador de H, consideremos el grafo H' resultante al extraerle a H las aristas de T (que recordemos que por ser árbol son exactamente $k-1$). Sea además T' un conjunto de árboles generadores sobre cada una de las componentes conexas de H', que podrían ser más de una ya que H' no es necesariamente conexo (ver Figura 2.6).

Caso1: $\exists e \in H'$ una arista que no pertenece a dicho bosque. En este caso al añadir la arista al bosque T' se cierra un ciclo en alguna componente conexa de H'. Como ninguna de las aristas de este ciclo está en el árbol generador inicial T, al retirarlas conservamos la conexión ya que al menos quedarán las aristas de T como ilustra la Figura 2.6.

Caso2: No hay ninguna arista más en H' que no esté en T'. Esto no puede ser ya que $|E_{H'}| > k-1$ y si c es el número de componentes conexas de H', T' tiene $k-c$ aristas que es menor que $k-1$. □

Teorema 2.5 (Letchford (2013) [8])**.** *Existe una solución óptima del STSP que no contiene más de* $2(n-1)$ *aristas.*

Demostración. Sea X una solución óptima con un número de aristas mínimo, con más de $2(n-1)$ aristas y que no tenga nodos aislados no requeridos (en caso contrario los suprimimos). Aplicando el Lema 2.4 sabemos que existe C un ciclo tal que si eliminamos sus aristas E_C se conserva la conexión. En el grafo resultante al suprimir E_C de X todos los nodos tienen grado par ya que lo tenían en X y hemos suprimido las aristas de un ciclo. De este modo, al darse esta propiedad y mantenerse la conexión hemos mejorado la solución inicial X que supusimos óptima. □

Observamos que en esta formulación pasamos a tener una cantidad del orden de n^2 variables y n^2 restricciones lo que hace más manejable el problema a la hora de su resolución con software. Sin embargo las cotas obtenidas en la relajación lineal son muy débiles lo cual hace que los algoritmos de ramificación y acotación no sean eficientes.

En la próxima sección introduciremos la teoría de flujos, lo que nos permitirá más tarde dar formulaciones con flujos cuyas relajaciones lineales den mejores cotas sin un número excesivo de variables y restricciones.

Capítulo 3

Flujos en redes

3.1. Introducción

En este capítulo introduciremos los conceptos básicos sobre flujos que quedan fuera de lo estudiado en el grado y daremos algunos resultados que nos permitan en el Capítulo 4 establecer relaciones entre estos flujos y la conexión dando paso a formulaciones más potentes.

Como se expone en Gale (1957) [5], consideraremos (V, E) un grafo con una función $d : V \to \mathbb{R}$ que denominaremos **demanda**. Esta función representará si es positiva la cantidad de un cierto ítem que debe llegar hasta un cierto nodo que llamaremos sumidero. En caso de ser negativa, representará la cantidad de ítem que puede salir del mismo y en tal caso, al nodo se le llamará fuente.

Además consideraremos otra función $c : V^2 \to [0, +\infty)$ llamada **capacidad** que denotará la cantidad máxima de ítem que puede viajar de un nodo a otro a través de la arista que los une. Para abreviar usaremos la expresión $c(S, S')$ para denotar a $\sum_{s\in S} \sum_{s'\in S'} c(s, s')$. Llamaremos **red** al par (G, c).

Definición 3.1. Un **flujo** en (G, c) es una función $f : V^2 \to \mathbb{R}$ que satisface:

$$f(x, y) + f(y, x) = 0 \; ; f(x, y) \leq c(x, y) \; \forall x, y \in V.$$

Un flujo de s a s', con $s, s' \in V$, es un flujo tal que $\sum_{y\in V} f(y, x) = 0 (\forall x \neq s, s')$. Para abreviar si $S \subset V$, $f(S, x) := \sum_{y\in S} f(y, x)$.

Definición 3.2. Una demanda se dice **factible** si existe un flujo f satisfaciendo:

$$f(V, x) \geq d(x) \ \forall x \in V.$$

3.2. El Teorema de Factibilidad y el Teorema del Corte Mínimo

En esta sección daremos algunos resultados para caracterizar la factibilidad de las demandas, lo cual será muy útil a la hora de introducir formulaciones en las próximas secciones basadas en flujos. El primero será un conocido teorema de flujos en redes, el Teorema del Corte Mínimo.

3.2.1. Teorema del Corte Mínimo

Definición 3.3. Un **corte** respecto de dos nodos s y s', es una partición de V en (S, S^c) tal que $s \in S$ *y* $s' \in S^c$. Denotaremos como $Q_{s,s'} := \{(S, S^c) : (S, S^c)$ corte respecto de s y $s'\}$. Análogamente, $F_{s,s'}$ será el conjunto de flujos de s a s'.

Teorema 3.4 (Teorema del corte mínimo, Gale 1957 [5])**.** $\forall s, s' \in V$ *se satisface que*

$$\text{máx}\{f(s, V)\}_{F_{s,s'}} = \text{mín}\{c(S, S')\}_{Q_{s,s'}}.$$

Demostración. Sea f un flujo de s a s' y $(S, S') \in Q_{s,s'}$, se tiene que $\sum_{x \in S, x \neq s} f(x, V) = 0$ ya que si $x \neq s, s' \quad f(x, V) = 0$. De este modo $f(s, V) = f(s, V) + \sum_{x \in S, x \neq s} f(x, V) = f(S, V) = f(S, S) + f(S, S') \leq c(S, S') \quad \forall \ (S, S') \in Q_{s,s'}$.

Ahora falta ver que se satisface la igualdad para $\hat{f} := \text{máx}\{f(s, V)\}_{f \in F}$ y algún corte. Para ello deberemos construir la partición (S, S').

Definimos S el conjunto de $x \in V$ tales que $x = s$ o existe $x_0 = s, x_1, ..., x_n$ un subconjunto de nodos con $c(x_{i-1}, x_i) > \hat{f}(x_{i-1}, x_i)$. En esta situación s' no puede pertenecer a S ya que si $\mu = \text{mín}\{c(x_{i-1}, x_i) - \hat{f}(x_{i-1}, x_i)\}_{i=1}^n$, incrementando en $\frac{\mu}{2}$ el flujo en estas aristas conseguimos un flujo mayor. De este modo, $(S, S^c) \in Q_{S,S'} \Rightarrow \hat{f}(s, V) \leq c(S, S')$.

Supongamos que $\hat{f}(s,V) < c(S,S')$, como $\hat{f}(S,S') = \hat{f}(S,V) = \hat{f}(s,V) < c(S,S')$, $\exists x \in S, y \in S' / \hat{f}(x,y) < c(x,y)$ y por tanto $s = x_0, x_1, ..., x_n = x$ con $\hat{f}(x_{i-1}, x_i) < c(x_{i-1}, x_i)$ de manera que añadiendo el nodo y a tal conjunto deducimos que y también está en S llegando a una contradicción. □

3.2.2. Teorema de Factibilidad

Una vez probado el Teorema del Corte Mínimo nos encontramos en disposición de dar un resultado que caracterice las demandas factibles:

Teorema 3.5 (Teorema de factibilidad, Gale 1957 [5])**.** *Una demanda es factible si y solo si* $\forall S \subset V$*:*

$$d(S^c) \leq c(S, S^c).$$

Demostración.

$\boxed{\Rightarrow}$ Tenemos que por definición de factible: $d(S^c) = \sum_{s \in S^c} d(s) \leq \sum_{s \in S^c} f(V,s) = f(V, S^c) = f(S^c, S^c) + f(S, S^c)$. Como se desprende de la definición $\forall S \subset V$ resulta que $f(S,S) = 0$, por tanto la expresión anterior se reduce a $\sum_{s \in S, s' \in S^c} f(s,s') \leq \sum_{s \in S, s' \in S^c} c(s,s') = c(S, S^c)$.

$\boxed{\Leftarrow}$ En primer lugar, vamos a ampliar la red a $(\hat{G}, \hat{c})$ donde $\hat{G} = (\hat{V}, \hat{E})$ se obtiene añadiendo dos nuevos nodos ficticios, s y s' que solo son adyacentes a las fuentes y a los sumideros respectivamente. Extendemos la función c a esta nueva red como sigue (ver Figura 3.1) :

$$\hat{c}(x,y) = \begin{cases} c(x,y) & \text{si} \quad x, y \in V \\ -d(y) & \text{si} \quad d(y) \leq 0, x = s \\ d(x) & \text{si} \quad d(x) > 0, y = s'. \end{cases}$$

Veamos que $(\hat{V} \backslash \{s\}, \{s\})$ es un corte mínimo:

Sea $(\hat{S}, \hat{S}') \in Q_{s,s'}$, y consideremos los subconjuntos $S = \hat{S} \cap V$ y $S' = \hat{S}' \cap V$. Tenemos que:

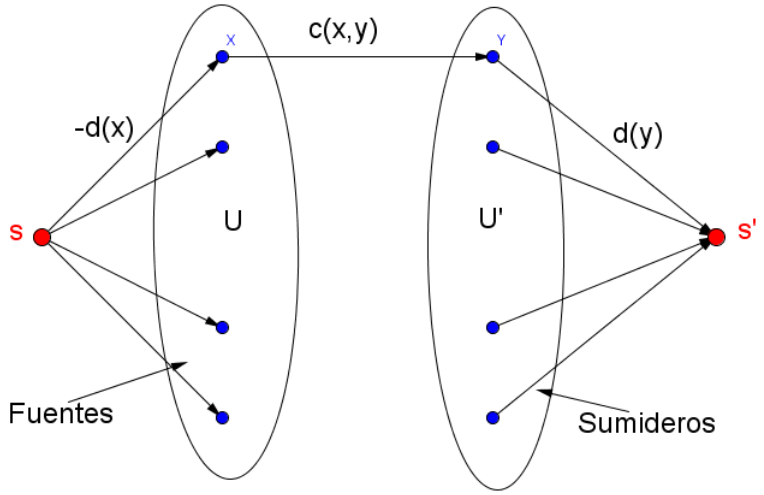

FIGURA 3.1: Ampliación de la red

$\hat{c}(\hat{S},\hat{S}') = c(S,S') + \hat{c}(s,S') + \hat{c}(S,s') + c(s,s')$ que al ser $c(s,s') = 0, \hat{c}(s,S') = -d(S'\cap U)$ y $\hat{c}(S,s') = d(S\cap U')$ equivale a $c(S,S') - d(S'\cap U) + d(S\cap U')$.

Por otro lado $\hat{c}(\hat{V}\backslash\{s'\},s') = d(U') = d(U'\cap S)+d(U'\cap S') \Rightarrow \hat{c}(\hat{S},\hat{S}')-\hat{c}(\hat{V}\backslash\{s'\},s') = d(U'\cap S') + d(S'\cap U) - c(S,S') = d(S') - c(S,S')$ que por hipótesis es ≤ 0 $\Rightarrow$ $(\hat{V}\backslash\{s'\},s')$ es un corte mínimo $\Rightarrow$ Por el Teorema del Corte Mínimo 3.4 $\exists \hat{f} \in F_{s,s'}/\hat{f}(s,\hat{V}) = \hat{c}(\hat{V}\backslash\{s'\},s') = d(U')$.

Así,

$$\begin{aligned} \hat{c}(\hat{V}\backslash\{s'\},s') = &\sum_{x\neq s'} \hat{c}(x,s') = \sum_{U'} \hat{c}(x,s') = \hat{f}(u',s') \\ \Rightarrow \hat{f}(x,s') = &\ \hat{c}(x,s) = d(x) \quad \forall x \in U'. \end{aligned}$$

Para finalizar la prueba basta observar que $\hat{f}$ restringido a V^2 es un flujo sobre la red inicial (G,c) por la manera en que lo hemos definido. Solo falta ver que además satisface d:

Si $x \in U$, $0 = \hat{f}(\hat{V},x) = \hat{f}(s,x) + f(V,x) \leq -d(x) + f(V,x) \Rightarrow d(x) \leq f(V,x)$

Si $x \in U'$, $0 = \hat{f}(\hat{V},x) = \hat{f}(s',x) + f(V,x) = -\hat{f}(x,s') + f(V,x) = -d(x) + f(V,x) \Rightarrow d(x) = f(V,x)$. □

En el Capítulo 4 haremos uso de esta teoría para introducir nuevas formulaciones y en el Capítulo 5 podremos comprobar su eficacia frente a las formulaciones del Capítulo 2.

Capítulo 4

Formulaciones basadas en flujos

Cuando formulamos el TSP o el STSP nuestro verdadero problema está en asegurar que nuestra solución es conexa y además hacerlo de manera eficiente. Para abordar esta cuestión nos imaginaremos que nuestro comerciante parte de un nodo, que consideraremos como depósito, con unas ciertas cantidades de algunos items o materiales y que debe repartirlas por los nodos requeridos, es decir, que estos materiales deben "fluir" por el grafo. Introduciremos dos formulaciones recogidas en Letchford (2013)[8] la primera de flujo simple en la que solo se reparte un tipo de material y otra de flujo compuesto en la que se reparten varios tipos de estos materiales.

4.1. Flujo simple

En esta formulación, conocida como formulación de flujo simple o *single commodity flow formulation*, supondremos que el primer nodo es requerido e imaginaremos que el comerciante parte de él con $|V_R| - 1$ unidades de un cierto ítem de las que deberá depositar una unidad en cada otro nodo requerido. Añadiremos a nuestras variables $\hat{x}_a$ una variable en cada arco f_a que representará las unidades de ítem que lo atravesarán.

$$
\begin{aligned}
\text{mín} \quad & \textstyle\sum_{a\in A} c_a \hat{x}_a & & \\
s.a \quad & \textstyle\sum_{a\in\delta^+(i)} \hat{x}_a \geq 1 & (\forall i \in V_R) & (4.1)\\
& \textstyle\sum_{a\in\delta^-(i)} \hat{x}_a = \sum_{a\in\delta^+(i)} \hat{x}_a & (\forall i \in V) & (4.2)\\
& \textstyle\sum_{a\in\delta^-(i)} f_a - \sum_{a\in\delta^+(i)} f_a = 1 & (\forall i \in V_R\backslash\{1\}) & (4.3)\\
& \textstyle\sum_{a\in\delta^-(i)} f_a - \sum_{a\in\delta^+(i)} f_a = 0 & (\forall i \notin V_R) & (4.4)\\
& 0 \leq f_a \leq (n_R - 1)\hat{x}_a & (\forall a \in A) & (4.5)\\
& \hat{x}_a \in \{0, 1\} & (\forall a \in A) &
\end{aligned}
$$

Cabe aclarar que a la hora de formular no consideraremos el flujo neto a través de una arista como en el Capítulo 3 ya que es menos apropiado a la hora de formular. Sin embargo ambas interpretaciones son equivalentes pues basta hacer el cambio $x_{ij} = x_{ij} - x_{ji}$ para que un flujo en las condiciones de este capítulo se convierta en un flujo en términos del Capítulo 3 y de Gale (1957) [5].

Veamos que obtendremos soluciones de STSP, como siempre al restringirnos al subgrafo solución formado por los nodos con grado mayor o igual que uno y los arcos a tales que $x_a = 1$:

Proposición 4.1. *La formulación flujo simple es una formulación válida para el STSP.*

Demostración. El multigrafo solución recorrerá todo V_R por (4.1) y será euleriano por (4.2) si es conexo, además (4.5) asegura que el flujo atraviese solo los arcos que estén en la solución y que por cada arco no pase más de $|V_R|$ unidades de material, que era lo que nos disponíamos a repartir. Razonando por reducción al absurdo, si no es conexo, existirá alguna componente conexa con algún nodo requerido en la que no está el primer nodo (si hay alguna componente conexa sin nodos de V_R la eliminamos y mejoramos la solución).

Sea S esta componente conexa, tenemos por (4.3) y (4.4) que :

$$\sum_{i \in S} \Big(\sum_{a \in \delta^-(i)} f_a - \sum_{a \in \delta^+(i)} f_a \Big) = |S \cap V_R| \geq 1.$$

Ahora bien, como S es una componente conexa $\{a \in \delta^+(i) : i \in S\} = \{a \in \delta^-(i) : i \in S\}$ y por tanto $\sum_{i \in S} \sum_{a \in \delta^-(i)} f_a = \sum_{a \in \delta^+(i)} f_a$, llegándose a una contradicción como muestra la Figura 4.1. □

Esta formulación contiene del orden de $|A|$ variables y $|A|$ restricciones. Sin embargo las cotas de la relajación lineal siguen siendo peores que en la formulación clásica como expondremos a continuación con los siguientes resultados.

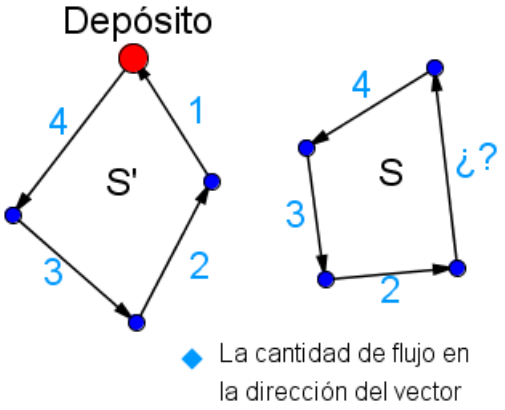

FIGURA 4.1: No se puede depositar una unidad de ítem en cada nodo de S.

Teorema 4.2 (Gale-Hoffman). *Sea $\hat{G} = (V, A)$ un grafo dirigido, s un nodo fuente, d una función demanda con $d_s = 0$ y ℓ, u cotas inferiores y superiores. Entonces existe un flujo satisfaciendo*

$$\begin{aligned} \ell \leq f \leq u \qquad & y \\ \textstyle\sum_{a\in\delta^-(i)} f_a = d_i + \sum_{a\in\delta^+(i)} f_a \quad & (\forall i \in V), \end{aligned}$$

si y solo si

$$\textstyle\sum_{a\in\delta^-(i)} u_a \geq \sum_{i\in S} d_i + \sum_{a\in\delta^+(S)} \ell_a \quad (\forall S \subset V\backslash\{s\}).$$

Teorema 4.3 (Letchford 2013 [8]). *Sea P^1 el subconjunto del $(\hat{x}, f)$-espacio definido por las restricciones (4.2)-(4.5) y la cota $\hat{x} \in [0, 1]$, P^2 la proyección de P^1 en el $\hat{x}$-espacio y P^3 la proyección de P^2 en el x-espacio mediante la aplicación lineal $x_{ij} = \hat{x}_{ij} + \hat{x}_{ji}$. Entonces P^3 está completamente descrito por las ecuaciones:*

$$\textstyle\sum_{e\in\delta(i)} x_e \geq 2 \qquad (i \in V_R), \tag{4.6}$$

$$\textstyle\sum_{e\in\delta(S)} x_e \geq 2\frac{|S\cap V_R|}{n_R - 1} \quad (\forall S \subset V\backslash\{1\} : |S \cap V_R| \neq 0), \tag{4.7}$$

$$x \in [0, 2]^{|E|}. \tag{4.8}$$

Demostración. Comencemos observando que la proyección de P^2 en P^3 está bien definida por ser simétrica y no importar escoger el representante (i, j) o (j, i) para cada arista de E. Esta misma proyección hace que se cumpla (4.8) en todo P^2. Haciendo $\ell_a = 0$ y $u_a = (|V_R| - 1)\hat{x}_a$ para cada a de A y fijando $s = 1$, $d_i = 1$ únicamente si i pertenece a

$V_R\backslash\{1\}$, aplicando el Teorema 4.2 tenemos que

$$\sum_{a\in\delta^-(S)} u_a \geq \sum_{i\in S} d_i \quad \forall S \subset V\backslash\{s\},$$

de donde se deduce que si $s \notin S$ se tiene que $(V_R - 1)\sum_{a\in\delta^-(S)} \hat{x}_a \geq |S \cap V_R|$, y aplicando (4.2) se llega a (4.7). Por otro lado, debido a (4.1) y (4.2) se deduce que $\sum_{a\in\delta^-(i)\cup\delta^+(i)} \hat{x}_a \geq 2$, que a su vez se traduce mediante nuestra proyección a (4.6).

En este punto de la demostración cabe aclarar que es justamente el recíproco del enunciado lo que nos resultará interesante ya que (4.6) -(4.8) son más débiles que las restricciones de la formulación clásica como veremos en el Corolario 4.3.

Si tenemos un elemento de este subconjunto del x-espacio, definiendo $\hat{x}_{ij} = \frac{x_{ij}}{2}$ tenemos un elemento del $\hat{x}$-espacio que efectivamente satisface la cota $x \in [0, 1]$ y evidentemente las restricciones (4.1) y (4.2). Además, por (4.7) también se cumple

$$\sum_{a\in\delta^-(S)} \hat{x}_e \geq \frac{|S \cap V_R|}{n_R - 1} \quad (\forall S \subset V\backslash\{1\} : |S \cap V_R| \neq 0),$$

donde tomando como función demanda $d(i) = 1$ para todo $i \in V_R\backslash\{1\}$, como cota superior $u_a = (|V_R| - 1)\hat{x}_a$ y como cota inferior $\ell = 0$ tenemos por el Teorema 4.2 que existe un flujo satisfaciendo (4.3)-(4.5). □

Como la relajación lineal de esta formulación está descrita por unas condiciones que son más débiles que las de la relajación lineal de la formulación clásica resulta que podemos comparar las cotas que proporcionan estas relajaciones:

Corolario 4.4 (Letchford (2013) [8])**.** *La cota proporcionada por la relajación lineal de la formulación flujo simple no es mejor que la proporcionada por la formulación clásica.*

Demostración. Como (2.8) implica (4.6) y (4.7) ya que $|S \cap V_R| \leq |V_R| - 1$ si $1 \notin S$, y debido a que todos los óptimos de la formulación clásica satisfacen la cota (4.8) como se deduce del Lema 2.2, se tiene que todo óptimo de la de la formulación clásica es solución factible de la relajación lineal de la formulación flujo simple y por tanto los óptimos en esta última formulación darán una cota más baja que los proporcionados por los de la clásica. □

4.2. Multiflujo

Ahora nos vamos a plantear la siguiente alternativa al flujo simple: imaginemos que el comerciante sale con $|V_R| - 1$ tipos de ítem distintos, pero que solo lleva una unidad de cada uno con el propósito de entregar en cada nodo de V_R la unidad del ítem correspondiente a dicho nodo. En esta situación tenemos un flujo asociado para cada tipo de material para lo cual introduciremos una variable binaria g_a^k que indicará si la mercancía del tipo k pasa a través del arco a:

$$
\begin{aligned}
\text{mín} \quad & \textstyle\sum_{a\in A} c_a \hat{x}_a & & \\
s.a \quad & \textstyle\sum_{a\in\delta^+(i)} \hat{x}_a \geq 1 & (\forall i \in V_R) & (4.9)\\
& \textstyle\sum_{a\in\delta^-(i)} \hat{x}_a = \sum_{a\in\delta^+(i)} \hat{x}_a & (\forall i \in V) & (4.10)\\
& \textstyle\sum_{a\in\delta^-(i)} g_a^k - \sum_{a\in\delta^+(i)} g_a^k = 0 & (\forall i \in V_R\backslash\{1\}; k \in V_R\backslash\{1, i\}) & (4.11)\\
& \textstyle\sum_{a\in\delta^-(k)} g_a^k - \sum_{a\in\delta^+(k)} g_a^k = 1 & (\forall k \in V_R\backslash\{1\}) & (4.12)\\
& \textstyle\sum_{a\in\delta^-(1)} g_1^k - \sum_{a\in\delta^+(i)} g_a^k = -1 & (\forall k \in V_R\backslash\{1\}) & (4.13)\\
& \hat{x}_a \geq g_a^k & (\forall a \in A; k \in V_R\backslash\{1\}) & (4.14)\\
& \hat{x}_a, g_a^k \in \{0, 1\} & (\forall a \in A, k \in V_R\backslash\{1\}). &
\end{aligned}
$$

La interpretación de esta formulación es similar a la del flujo simple. Cabe destacar las restricciones (4.11) y (4.12) que aseguran que el k-ésimo ítem se le entregue al k-ésimo nodo requerido y que no se entregue ninguna unidad a ningún otro nodo. En esta situación tenemos del orden de $|V_R||A|$ variables y restricciones.

A continuación se expondrán unos resultados para mostrar la eficiencia de esta formulación. El primero es análogo al Teorema 4.2 y caracteriza la relajación lineal de esta formulación:

Teorema 4.5 (Letchford (2013) [8]). *Sea P^1 el subconjunto del $(\hat{x}, g)$-espacio definido por las restricciones (4.9)-(4.14) y la cota $\hat{x}, g \in [0,1]$, P^2 la proyección de P^1 en el $\hat{x}$-espacio y P^3 la proyección de P^2 en el x-espacio mediante la aplicación lineal $x_{ij} = \hat{x}_{ij} + \hat{x}_{ji}\ \forall (i,j) \in E$.Entonces P^3 está completamente descrito por las ecuaciones:*

$$\sum_{e \in \delta(i)} x_e \geq 2 \qquad (\forall i \in V_R),$$
$$\sum_{e \in \delta(S)} x_e \geq 2 \quad (\forall S \subset V : S \cap V_R \neq \emptyset, V_R \cap S^c \neq \emptyset) \quad y$$
$$x \in [0,2]^{|E|};$$

que se corresponden con las ecuaciones (4.6), (2.8) y (4.8).

Demostración. Si fijamos k perteneciente a $V_R \backslash \{1\}$, las restricciones (4.12)-(4.14) impuestas sobre g^k equivalen a que g^k sea un flujo con cota superior x_a que satisface la demanda $d_k(k) = 1; d_k(1) = -1; d_k(i) = 0\ \forall i \neq k, 1$. Aplicando el Teorema de Factibilidad 3.5 tenemos que para todo $S \subset V$ se satisface que $c(S^c, S) \geq d_k(S)$. En particular,

$$c(S^c, S) = \textstyle\sum_{a \in \delta^+(S)} \hat{x}_a \geq d_k(S) = d_k(k) = 1 \quad \forall S : k \in S, 1 \notin S.$$

Combinando estas ecuaciones para todo $k \in V_R \backslash \{1\}$ tenemos que en P^2 se satisface que

$$\textstyle\sum_{a \in \delta^+(S)} \hat{x}_a \geq 1 \quad \forall S : S \cap V_R \neq \emptyset, 1 \notin S.$$

Si consideramos además (4.11) obtenemos que $\sum_{a \in \delta^+(S)} \hat{x}_a \geq 1$ para todo S tal que $V_R \cap S \neq \emptyset, V_R \backslash S \neq \emptyset$ ya que si $1 \notin S$ se deduce de lo anterior y si $1 \notin S$ entonces considerando S^c tenemos que $\sum_{a \in \delta^+(S^c)} \hat{x}_a \geq 1$ que por (4.11) equivale a $\sum_{a \in \delta^-(S^c)} \hat{x}_a \geq 1$ que no es otra cosa que $\sum_{a \in \delta^+(S)} \hat{x}_a \geq 1$. Finalmente, aplicando de nuevo (4.11) deducimos (2.8). Además, también de (4.9) y la cota de la relajación lineal $x \in [0,1]$ se deducen fácilmente (4.6) y (4.8).

Recíprocamente, si tenemos un elemento de P^3 satisfaciendo estos tres grupos de restricciones, al definir de nuevo $\hat{x}_{ij} = \frac{x_{ij}}{2}$ se satisfacen (4.9), (4.10) y $\hat{x} \in [0,1]$. Además por (2.8) también tendríamos que

$$\textstyle\sum_{a \in \delta^+(S)} \hat{x}_a \geq 1 \quad (\forall S \subset V : S \cap V_R \neq \emptyset, V_R \cap S^c \neq \emptyset).$$

A continuación, para ver que también se satisfacen el resto de restricciones fijamos k perteneciente a $V_R \backslash \{1\}$, y establecemos como cota superior $c = \hat{x}$. Considerando la demanda $d_k(k) = 1, d_k(1) = -1$, vamos a estudiar si estamos en condiciones de aplicar el Teorema de Factibilidad 3.5, es decir si para todo $S \subset V$ se tiene que $d(S) \leq c(S^c, S) = \sum_{a \in \delta^+(S)} \hat{x}_a$, lo cual se satisface ya que si $S \cap V_R = \emptyset$ ó $V_R \cap S^c = \emptyset$ entonces $d(S) = 0$. Concluimos que existe para cada k un flujo g^k que satisface la demanda d_k y por tanto las restricciones (4.11)-(4.14). □

Debido a las ecuaciones que describen las relajaciones lineales de la formulación multiflujo y clásica se deduce el siguiente corolario:

Corolario 4.6 (Letchford (2013) [8])**.** *Las cotas de las relajaciones lineales proporcionadas por la formulación multiflujo son iguales que las proporcionadas por la formulación clásica.*

Capítulo 5

Resolución de algunas instancias del STSP

En este capítulo comprobaremos el funcionamiento de nuestras formulaciones comparando el tiempo que tardan en resolver nuestros problemas implementados en el software de optimización lineal Xpress.

Tendremos que tener en cuenta que Xpress utiliza algunas técnicas para acelerar el proceso que incluyen un preprocesamiento en el que se introducen nuevas restricciones que reducen nuestro poliedro ajustándolo más a la envolvente convexa del conjunto de soluciones enteras. En este trabajo no hemos tenido en cuenta estos métodos por lo que para compararlas deberemos pedirle a Xpress que no los aplique a nuestros problemas. Para ello introducimos en el código Mosel los comandos:

- setparam("XPRS_CUTSTRATEGY",0)
- setparam("XPRS_HEURSTRATEGY",0)
- setparam("XPRS_PRESOLVE",0)
- setparam("XPRS_MIPPRESOLVE",0)

5.1. Tablas de resultados

Las instancias del problema las hemos generado con un programa implementado en Xpress repartiendo nodos de manera aleatoria en una región del plano y conectándolos de manera que las aristas no se crucen para darle un aspecto más real. Con este programa obtenemos grafos como el de la Figura 5.1.

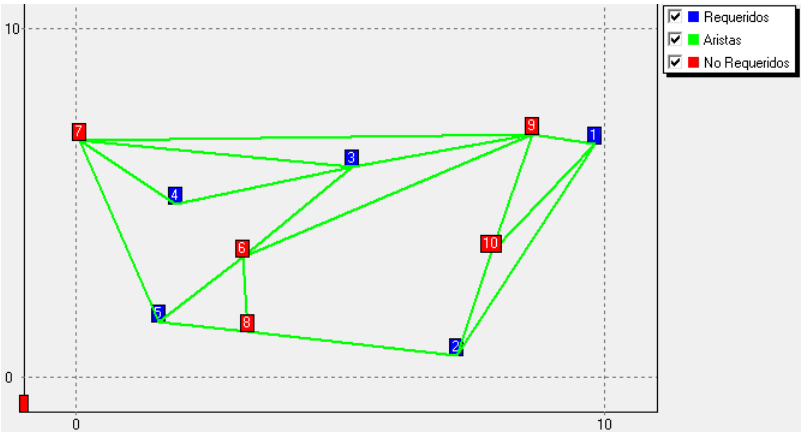

FIGURA 5.1: Ejemplo con 10 nodos, 5 requeridos

En los Cuadros 5.1 y 5.2 presentamos los datos obtenidos al resolver nuestras instancias indicando el tiempo en segundos (si no se indica lo contrario) que tarda cada formulación en función del tamaño de la instancia, $(|V_R|, |V|)$, y de si usamos o no el preprocesamiento de Xpress. En ellas podemos observar que la formulación en etapas comienza a tener problemas con las instancias a partir de 40 nodos incluso con nuestro refuerzo, denotado por (r), que parece suponer una mejora en la relajación lineal de un 4 por ciento aunque a veces empeora el tiempo de resolución. Por esta razón en el Cuadro 5.2 nos limitamos a resolver su relajación lineal. Así mismo vemos que sin ayuda del preprocesamiento podemos resolver hasta 90 nodos con las otras dos formulaciones aunque el tiempo de la formulación flujo simple varía mucho de una instancia a otra. No obstante si incluimos preprocesamiento podemos resolver instancias mucho más grandes como muestra la Figura 5.2 en la que mediante la formulación flujo simple resolvemos una instancia de 150 nodos en tan solo 3 segundos. Por otro lado, a pesar de que la formulación multiflujo tiene una cota casi óptima, dado que su tamaño es mayor, también tiene problemas con instancias a partir de 90 nodos. Con esto concluimos que aunque tener buenas cotas para la relajación lineal es importante ya que de esto depende el número de ramificaciones que realizaremos, también influye el tamaño de nuestra formulación ya que cuantas más variables y restricciones tengamos más tardaremos en resolver cada subproblema.

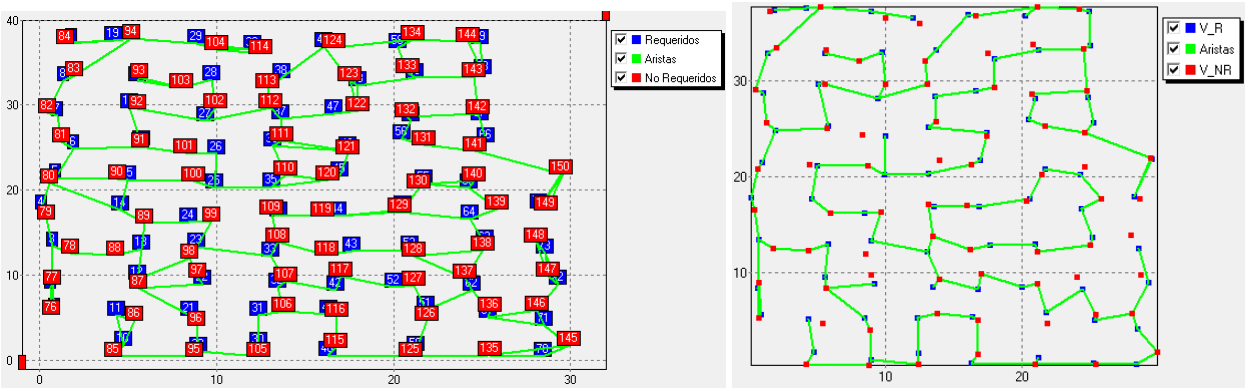

FIGURA 5.2: Instancia del STSP con 150 nodos resuelto con Xpress mediante la formulación flujo simple

CUADRO 5.1: De 20 a 60 nodos

Instancia	1	1sp	2	2 s.p	3	3 s.p.	4	4 s.p
$(\|V_R\|, \|V\|)$	(10,20)	(10,20)	(20,40)	(20,40)	(25,50	(25.50)	(30,60)	(30,60)
Valor obj.	23.88	23.88	26.49	26.49	91.08	91.08	34.50	34.50
Tiempo etapas	7.6	15.7	>2h	>2h	352	31 min	>2h	>2h
Rel. lin.	16.02	16.00	17.59	17.59	89.74	89.74	19.39	19.39
Gap %	33 %	33 %	34 %	34 %	2 %	2 %	44 %	44 %
Tiempo etapas (r)	10.1	11.0	>2h	>2h	272	37 min	>2h	>2h
R.L.	18.15	18.15	19.99	19.99	89.74	89.74	20.79	20.79
Gap %	24 %	24 %	25 %	25 %	2 %	2 %	40 %	40 %
Tiempo f. simple	0.1	0.1	0.3	16.2	0.1	0.6	2.0	>2h
R.L.	18.89	18.89	20.79	20.79	89.80	89.81	21.382	21.382
Gap %	21 %	21 %	22 %	22 %	2 %	2 %	39 %	39 %
Tiempo multiflujo	0.1	0.2	0.5	2.0	1.1	5.1	0.8	11.0
R.L.	23.88	23.88	26.49	26.49	91.08	91.08	34.50	34.50
Gap %	0 %	0 %	0 %	0 %	0 %	0 %	0 %	0 %

CUADRO 5.2: De 70 a 90 nodos.

Instancia	5	5sp	6	6sp	7	7s.p.
$(\|V_R\|, \|V\|)$	(35,70)	(35,70)	(40,80)	(40,80)	(45,90)	(45,90)
Valor obj.	65.62	65.62	85.60	85.60	128.42	128.42
Tiempo etapas	-	-	-	-	-	-
Rel. lin.	39.54	39.54	54.71	54.71	65.60	65.60
Gap %	40 %	40 %	36 %	36 %	49 %	49 %
Tiempo etapas (r)	-	-	-	-	-	-
R.L.	43.13	43.13	58.74	58.74	71.42	71.42
Gap %	35 %	35 %	32 %	32 %	45 %	45 %
Tiempo flujo simple	1.5	>2h	1.5	1.5h	2.9	>2h
R.L.	44.29	44.29	62.17	62.17	72.75	72.75
Gap %	33 %	33 %	28 %	28 %	43 %	43 %
Tiempo multiflujo	1.4	27.55	2.0	22.0	6.4	>2h
R.L.	65.62	65.62	85.60	85.60	127.87	127.87
Gap %	0 %	0 %	0 %	0 %	0.5 %	0.5 %

Capítulo 6

El problema del árbitro viajero

Otro problema de rutas relacionado con el TSP es el de la asignación de árbitros en un torneo deportivo o *Traveling Umpire Problem* (TUP). Este problema fue propuesto por Trick y Yildiz (2007) [14] y se inspira en la liga de béisbol de Estados Unidos.

Todos los partidos de esta liga se juegan durante una temporada que dura unos cuatro meses en los que tanto jugadores como árbitros deben recorrer las largas distancias que separan los estadios de los equipos. Cuando nos proponemos minimizar estas distancias, jugando con la asignación de equipos y árbitros en cada ronda, nos encontramos con que el calendario deportivo no es modificable ya que atiende a diversas razones como que los partidos importantes caigan en fechas propicias para la audiencia. Por este motivo el TUP trata de asignar los árbitros a los encuentros de este torneo de manera que estos recorran la menor distancia posible. A pesar de que en un partido hay un equipo de árbitros (compuesto por un árbitro y varios asistentes), estos equipos se forman al inicio de temporada y permanecen fijos hasta el final por lo que no necesitamos preocuparnos por formar los equipos de árbitros y cada equipo se considera como un árbitro solo.

El verdadero problema tiene muchos factores a tener en cuenta y su formulación podría llenar decenas de páginas teniendo en cuenta aspectos como los periodos de descanso que prefieren los árbitros. A pesar de excluir muchos detalles presentes en la vida real, el TUP recoge lo más importante de este problema y plantea un verdadero desafío a la hora de su resolución. Esto se puede comprobar a partir de las instancias sin resolver en Toffolo, Wauters y Trick (2013) [13].

6.1. Enunciado

Dados $2n$ equipos en una liga que enfrenta a cada equipo con sus rivales dos veces (una en casa y otra fuera) en un total de $4n-2$ jornadas y dos enteros $0 \leq d_1 \leq n-1$ y $0 \leq d_2 \leq E[\frac{n}{2}]$, donde E denota la parte entera de un número, una solución del TUP consiste en asignarle a n árbitros los encuentros de dicha liga satisfaciendo las siguientes condiciones:

1. En cada jornada, a cada árbitro se le debe asignar exactamente un partido y cada partido se debe asignar a un árbitro.
2. Cada árbitro debe arbitrar a todos los equipos en casa al menos una vez.
3. Cuando un árbitro arbitra en un estadio, no puede volver en las siguientes $n - d_1$ jornadas.
4. Cuando un árbitro arbitra a cierto equipo, no puede volver a hacerlo en las siguientes $E[\frac{n}{2}] - d_2$ jornadas.

El objetivo es minimizar las distancias recorridas por los árbitros durante toda la liga, entendiendo que si un árbitro arbitra en un estadio y la siguiente jornada en otro estadio, recorre la distancia que los separa. Cuando $d_1 = d_2 = 0$ las condiciones (3) y (4) se vuelven más estrictas y nos referimos a las instancias con estas características, siguiendo la nomenclatura de Oliveira, Souza y Yunes (2014) [10], como instancias dificiles (*hard instances*).

Puede parecer arbitrario que los parámetros en las condiciones (3) y (4) estén acotados por n y $E[\frac{n}{2}]$ respectivamente. Sin embargo, como se puede ver en Yildiz (2008) [17], a pesar de que no queramos que los árbitros visiten los estadios frecuentemente, incluso obviando las restricciones (2) y (4), esta frecuencia está acotada por n. Por otro lado, considerando solo (1) y (4) resulta siempre factible el problema cuando $d_2 = 0$.

Resulta que el TUP generaliza al TSP. Si en nuestra liga permitimos que se repitan equipos pero exigimos que una vez que se ha arbitrado en un estadio no se vuelva en n jornadas y además tenemos n rondas consecutivas en las que n de los equipos reciben a los otros en casa. La solución en estas jornadas consistiría en hacer recorrer a todos los árbitros la solución óptima del TSP que pase por todos los estadios de dichos equipos. Por esta razón si fuésemos capaces de desarrollar un método eficiente para este problema obtendríamos automáticamente otro para el TSP y por tanto para el STSP.

6.2. Formulación

La formulación presentada por Trick y Yildiz (2011) [15] comienza con los siguientes conjuntos de datos que debe introducir el usuario:

- El conjunto de árbitros que denotaremos $U = \{1, \ldots, n\}$.
- El conjunto de equipos $T = \{1, \ldots, 2n\}$.
- El conjunto de rondas $S = \{1, \ldots, 4n-2\}$.
- $OPP[s,i] = \begin{cases} j & \text{si } i \text{ juega contra } j \text{ en casa en la ronda } s, \\ -j & \text{si } i \text{ juega contra } j \text{ como visitante en la ronda } s. \end{cases}$
- d_{ij}= la distancia entre los campos de i y j.
- $CV_s = \{s, \ldots, s+n-d_1-1\} \ \ \forall s \in \{1, \ldots, 4n-2-(n-d_1-1)\}$, el conjunto de las d_1 jornadas siguientes a s empezando en s.
- $CT_s = \{s, \ldots, s+E[\frac{n}{2}]-d_2-1\} \ \ \forall s \in \{1, \ldots, 4n-2-(E[\frac{n}{2}]-d_2-1)\}$, el conjunto de las $E[\frac{n}{2}]-d_2$ jornadas siguientes a s empezando en s.

Nuestras variables de decisión que serán binarias:

- x_{isu} si el partido en el estadio del i-ésimo equipo se le asigna al árbitro u.
- z_{ijsu} si el árbitro u estaba en el campo de i en la ronda s y debe viajar al de j para arbitrar en la ronda $s+1$.

A continuación presentamos la formulación:

$$\text{mín} \quad \sum_{i\in T}\sum_{j\in T}\sum_{u\in U}\sum_{\substack{s\in S:\\ s\leq |S|-1}} d_{ij}z_{ijsu} \tag{6.1}$$

$$s.a \quad \sum_{u\in U} x_{isu} = 1, \qquad \forall i \in T, s \in S : OPP[s,i] \geq 1 \tag{6.2}$$

$$\sum_{\substack{i\in T:\\ OPP[s,i]\geq 1}} x_{isu} = 1, \qquad \forall s \in S, u \in U \tag{6.3}$$

$$\sum_{\substack{s\in S:\\ OPP[s,i]\geq 1}} x_{isu} \geq 1, \qquad \forall i \in T, u \in U \tag{6.4}$$

$$\sum_{\substack{c\in CV_s:\\ OPP[c,i]\geq 1}} x_{icu} \leq 1, \qquad \forall i, u, s : s \leq |S| - (n - d_1 - 1) \tag{6.5}$$

$$\sum_{c\in CT_s} (x_{icu} + \sum_{\substack{j\in T:\\ OPP[c,j]=i}} x_{jcu}) \leq 1, \quad \forall i, u, s : s \leq |S| - (E[\frac{n}{2}] - d_2 - 1) \tag{6.6}$$

$$x_{isu} + x_{j(s+1)u} - z_{ijsu} \leq 1, \qquad \forall i, j \in T, u \in U, s \in S : s \leq |S| - 1 \tag{6.7}$$

$$x_{isu} \in \{0,1\}, \qquad \forall i \in T, u \in U, s \in S \tag{6.8}$$

$$z_{ijsu} \in \{0,1\}, \qquad \forall i, j \in T, u \in U, s \in S : s \leq |S| - 1. \tag{6.9}$$

El objetivo (6.1) minimiza la distancia total recorrida por los árbitros que, como decíamos, no es otra que la que separa dos estadios en los que un árbitro debe actuar consecutivamente. En esta formulación, las variables importantes son las x_{isu} y nos indicarán las asignaciones de los árbitros, que es lo que estábamos buscando mientras que las z_{ijsu} solo indican las distancias a recorrer. Por esto solo intervienen en la función objetivo y en (6.7) donde se las fuerza a tomar el valor uno cuando $x_{isu} = x_{j(s+1)u} = 1$ para ciertos i, j, s y u añadiendo la distancia d_{ij} al objetivo. En otro caso, como estamos minimizando y $d \geq 0$, z_{ijsu} tomará el valor 0.

La condición (1) de nuestro problema se satisface gracias a (6.2) y (6.3) que aseguran por un lado que a cada partido se le asigne un árbitro, y por otro que en cada jornada se le asigne un partido a todos los árbitros.

La condición (2), *todos los árbitros deben intervenir en todos los campos*, estará modelada por las restricciones (6.4) ya que el subíndice s en el sumatorio se mueve dentro de los partidos que juega en casa el i-ésimo equipo.

La (3), sobre la frecuencia de los arbitrajes en casa, está modelada por las restricciones (6.5) al moverse el subíndice c en todos los partidos en casa del i-ésimo equipo en las rondas siguientes a s.

Finalmente la condición (4), sobre la frecuencia con la que un árbitro puede intervenir en los partidos de un equipo, se satisface al imponer (6.6) ya que $\sum_{c \in CT_s} x_{isu}$ se corresponde con los partidos jugados en casa y $\sum_{c \in CT_s} \sum_{j \in T: OPP[c,j]=-i} x_{jcu}$ con los jugados fuera por el equipo i-ésimo durante las rondas consecutivas correspondientes.

6.2.1. Refuerzo

Las restricciones (6.2)-(6.7) junto con la integridad de las variables nos describen el conjunto factible de nuestro problema. Sin embargo además de estas restricciones podemos añadir algunas adicionales que agilicen la búsqueda de soluciones al ajustar un poco más el conjunto factible a la envolvente convexa de nuestras soluciones enteras y fijar los valores de algunas variables. Para ello también en Trick y Yildiz (2011) [15] se presentan las siguientes restricciones:

$$x_{isu} = 0, \qquad \forall i \in T, u \in U, s \in S : OPP[s,i] \leq -1 \quad (6.10)$$

$$z_{ijsu} \leq x_{isu}, \qquad \forall i,j \in T, u \in U, s \in S : s \leq |S| - 1 \quad (6.11)$$

$$z_{ijsu} \leq x_{j(s+1)u}, \qquad \forall i,j \in T, u \in U, s \in S : s \leq |S| - 1 \quad (6.12)$$

$$\sum_{j \in T} z_{jisu} - \sum_{j \in T} z_{ij(s+1)u} = 0, \qquad \forall i \in T, u \in U, s \in S : s \leq |S| - 2 \quad (6.13)$$

$$\sum_{i,j \in T} z_{ijsu} = 1, \qquad \forall u \in U, s \in S : s \leq |S| - 1. \quad (6.14)$$

La restricción (6.10) prohíbe la asignación de un árbitro a cualquier campo donde no se juegue ningún partido en la ronda s.

Las restricciones (6.11) y (6.12) permiten a un árbitro desplazarse de un estadio a otro en cierta ronda solo si se juega en en el primer estadio en dicha ronda y en el segundo la siguiente. Para esto teníamos la restricción (6.7) pero para deducirlas de ella necesitamos usar la integridad de las variables y el hecho de que estamos minimizando.

La restricción (6.13) establece una conservación en el flujo de árbitros entre los estadios ya que impone que justamente cuando un árbitro haya viajado a un estadio en cierta ronda, salga de este estadio en la siguiente.

Por último la restricción (6.14) fuerza a que cada árbitro se desplace en cada ronda.

Más tarde, en Oliveira (2014) [10], se proponen más restricciones con el mismo propósito:

$$x_{i1u} = \sum_{j \in T} z_{ij1u}, \quad \forall i \in T, u \in U \tag{6.15}$$

$$x_{isu} = \sum_{j \in T} z_{ji(s-1)u}, \quad \forall i \in T, u \in U, s \in S : s \geq 2. \tag{6.16}$$

La restricción (6.15) asegura que en la primera ronda un árbitro salga de un estadio si y solo si arbitró en él. La restricción (6.16) es análoga a esta para el resto de rondas debido a la conservación de flujo que impone (6.13), un arbitro saldrá/llegará a un estadio en cierta ronda si y solo si arbitra en él en dicha ronda.

Esto permite sustituir nuestras variables x_{isu} en términos del flujo de árbitros convirtiéndose en un problema de flujos. Sin embargo, no las suprimiremos ya que es más sencillo trabajar con ellas y Xpress las elimina automáticamente al hacer preprocesamiento con las igualdades que le introduzcamos.

En este mismo artículo también podemos encontrar las siguientes igualdades relativas al flujo que también nos ayudarán a reducir nuestra cantidad de variables. Por un lado tenemos la restricción

$$z_{iisu} = 0, \quad \forall i \in T, u \in U, s \in S : s \leq |S| - 1, d_1 \leq n - 2 \tag{6.17}$$

y por otro lado si $d_2 \leq E[\frac{n}{2}] - 2$ también introducimos la restricción

$$z_{ijsu} = 0 \qquad \forall i \neq j \in T, u \in U, s \in S : s \leq |S| - 1, \text{ y} \tag{6.18}$$

se da alguna de estas condiciones:

$$OPP[s, i] = OPP[s + 1, j], \text{o } OPP[s, i] = j, \text{o bien } OPP[s + 1, j] = i.$$

La restricción (6.17) prohíbe que un árbitro repita campo mientras que la (6.18) asegura que un árbitro tras intervenir en un encuentro no se vaya a otro campo donde esté uno de los dos equipos para los que acaba de arbitrar siempre que hayamos impuesto la condición (4).

En la próxima sección reduciremos aun más el tamaño de nuestro conjunto factible rompiendo sus simetrías.

6.2.2. Simetría

Cuando una misma solución en la práctica tiene varias maneras de traducirse a nuestra formulación, decimos que nuestra formulación tiene simetrías. Este es un problema que con frecuencia resulta difícil de resolver a la hora de formular y que aumenta el tamaño del conjunto factible. Así, una solución de nuestro problema admitiría reordenaciones entre todos los árbitros ya que todos juegan el mismo papel. Por esto mismo, para cada solución que encontremos tendremos $n!$ formas distintas de reordenarla sin que deje de ser esencialmente la misma solución.

Para evitar esta simetría, como se hace en Yildiz (2008) [17], vamos a fijar los partidos de una cierta ronda k, que se puede elegir arbitrariamente en el intervalo $[1, 4n-2]$, repartiéndolos a priori entre los árbitros. Para introducir esta ronda contamos con la siguiente restricción :

$$x_{iku} = 1, \quad \forall (i,u) \in K. \tag{6.19}$$

En esta restricción K es el conjunto de uplas formadas por un partido de ronda fijada y el árbitro al que queremos asignárselo. Normalmente por comodidad realizaremos esta asignación en la primera ronda.

A continuación daremos un ejemplo, recogido en Oliveira (2014) [10], que sirve para comprobar la eficacia de las nuevas restricciones:

Consideremos una instancia del TUP con dos árbitros y cuatro equipos jugando una liga cuyo calendario viene en esta tabla:

Ronda	1	2	3	4	5	6
Encuentros	(1,2)	(2,3)	(1,3)	(2,1)	(1,4)	(3,1)
(Local,visitante)	(3,4)	(4,1)	(2,4)	(4,3)	(3,2)	(4,2)

Se tiene además que los estadios de los tres primeros equipos están cerca unos de otros, digamos a un kilómetro, mientras que el cuarto está a diez kilómetros de ellos. En esta situación, implementando el problema en Xpress, se obtiene que con $d_1 = d_2 = 0$ e introduciendo la restricción (6.19) en la primera ronda, asignando el primer partido al primer árbitro y el segundo al segundo ($K = \{(1,1),(3,2)\}$) , se tiene que para la relajación lineal de las restricciones (6.2)-(6.9) y (6.19) la solución óptima sería asignar el valor 0.5 a todas las variables correspondientes a los partidos que se juegan a partir de la

segunda ronda, uno para a las recogidas por (6.19) y cero al resto obteniéndose una cota con esta relajación de 11 kilómetros. Sin embargo al introducir las restricciones (6.10)-(6.18) esta solución se vuelve infactible y la cota asciende a 55 que aunque sigue sin ser una solución entera es una cota equivalente al valor óptimo sin obviar las restricciones de integridad.

6.3. Un último procedimiento

En esta sección vamos a exponer parte de un artículo publicado recientemente, Oliveira (2016)[11], en el que se muestra una nueva vía más elaborada para abordar el problema que requiere de una técnica específica para su implementación en el optimizador. Dada la complejidad de su construcción, añadiremos un ejemplo para facilitar la comprensión de la información recogida en dicho artículo.

El procedimiento a seguir consiste en una vez formulado el problema, construir otro conjunto de desigualdades válidas que no se implementa inicialmente dada su excesiva cantidad. A continuación, se resuelve la relajación lineal del problema y se elige de entre estas nuevas restricciones la que más se incumpla en este óptimo para introducirla al problema y repetir el proceso iterativamente. Para esta elección el procedimiento podrá ser heurístico ya que elegir exactamente la desigualdad que más se incumpla puede ser un problema dificil de resolver dada la cantidad de restricciones.

6.3.1. Formulación

Dividimos las rondas de la liga en varios subconjuntos de rondas consecutivas de una longitud fija. Consideramos en cada subconjunto todas las posibles rutas entre los partidos de estas rondas que contienen exactamente un partido de cada ronda y satisfacen las condiciones (3) y (4).

Así mismo, fijamos en $w \in \{2\ldots, 4n-2\}$ la longitud de estos subconjuntos y dividimos las jornadas en secciones indexadas por el conjunto $S = \{1, \ldots, E[\frac{4n-3}{w-1} + 1]\}$. De este modo, la s-ésima sección de la liga, denotada por T_s, se corresponderá con las consecutivas jornadas $(s-1)(w-1)+1$ hasta mín$\{s(w-1)+1, 4n-2\}$. Observamos que cada una de estas secciones tendría w jornadas, salvo la última que podría ser más corta. En el Ejemplo 6.1 se ilustra la elaboración de estas secciones con $n = 2$ y $w = 3$.

Ejemplo 6.1 Supongamos que tenemos organizada una liga para 4 equipos en la que arbitran 2 árbitros cuyas jornadas se exponen a continuación. Las secciones T_s quedan de este modo:

			T_2			
Rondas	1	2	3	4	5	6
Encuentros	(2,1)	(3,1)	(1,4)	(2,3)	(1,2)	(1,3)
(Local,visitante)	(3,4)	(4,2)	(3,2)	(4,1)	(4,3)	(2,4)
	T_1				T_3	

Como se ilustra en este ejemplo, para cada $2 \leq s \leq |S|$ coinciden la primera jornada de T_s con la última de T_{s-1}. Sigamos construyendo nuestra formulación:

Para cada sección s, sea P_s el conjunto de secuencias de viajes en las rondas de T_s conectando un partido de cada jornada con otro de la siguiente y satisfacen las condiciones (3) y (4) durante las rondas T_s.

Siguiendo con en el Ejemplo 6.1, a continuación calcularemos los P_1 suponiendo que $d_1 = 1$ y $d_2 = 1$ que imponen la condición de que los árbitros no puedan repetir estadio en rondas consecutivas pero sí equipos:

P_1 con $d_1 = d_2 = 1$ y $w = 3$
(2,1)-(3,2)-(1,4)
(2,1)-(4,2)-(1,4)
(2,1)-(4,2)-(3,2)
(3,4)-(4,2)-(1,4)
(3,4)-(4,2)-(3,2)

Llamaremos a todos estos elementos **rutas simples**. Denotemos por $P = \cup_{s \in S} P_s$ el conjunto de todas las rutas simples. En esta situación, para cada p de P, x_p será una variable binaria que valdrá uno si la ruta simple p pertenece a la solución o cero en caso contrario. Denotaremos por d_p la distancia total de la ruta simple, G_s el conjunto de partidos de T_s y P_{sg} el subconjunto de rutas simples de P_s que contienen al partido g para cada $g \in G_s$.

Para dar una solución al TUP con estas variables debemos juntar una ruta simple de cada sección T_s. Usaremos el término **ruta** para referirnos a una secuencia ordenada de

rutas simples $p_1, p_2, \ldots, p_m$ en la que p_i y p_{i+1} pertenecen a secciones consecutivas y el último partido de p_i coincide con el primero de p_{i+1}. Dada una ruta Q, $P(Q)$ denotará el conjunto de sus rutas simples. Llamamos **ruta completa** a una ruta que contiene una ruta simple de cada sección T_s. Por último diremos que una ruta es infactible cuando contiene dos o más juegos que incumplen las restricciones (3) y (4) o cuando es una ruta completa y no satisface (2) denotando por $\mathbb{U}$ el conjunto de todas las rutas infactibles.

Ahora ya estamos preparados para presentar nuestra formulación:

$$\begin{aligned}
\text{mín} \quad & \textstyle\sum_{p\in P} d_p x_p && && (6.20)\\
s.a \quad & \textstyle\sum_{p\in P_{sg}} x_p = 1, && \forall s \in S, g \in G_s && (6.21)\\
& \textstyle\sum_{p\in P(U)} x_p \leq |P(U)| - 1, && \forall U \in \mathbb{U} && (6.22)\\
& x_p \in \{0,1\}, && \forall p \in P. && (6.23)
\end{aligned}$$

Cuando formamos una ruta completa, su distancia total será la suma de todas las distancias de las rutas simples que la componen. De este modo, (6.20) minimiza la distancia recorrida por los árbitros. La restricción (6.21) asegura que a todos los partidos de todas las secciones llegue exactamente una ruta simple. Observemos que los partidos pertenecientes a dos secciones estarán incluidos en dos rutas simples; una comenzando y otra acabando en dicho partido de manera que podamos unirlas en una ruta. Si combinamos esta restricción con la integridad de las variables (6.23), aseguramos que en cada sección de la liga la solución esté formada por n rutas simples disjuntas con las que podremos formar n rutas completas satisfaciendo (1) y (2). Por último, observamos que todas nuestras rutas simples satisfacen (3) y (4) ya que la restricción (6.22) nos impide formar rutas que las incumplan al juntarlas puesto que impide que estén en la solución todas las rutas simples que constituyan una ruta infactible.

En el Ejemplo 6.1, las rutas simples

$$\hat{u_1} = (2,1) - (4,2) - (1,4), \quad \hat{u_2} = (1,4) - (2,3) - (4,3) \text{ y } \hat{u_3} = (4,3) - (2,4)$$

pertenecen a T_1, T_2 y T_3 respectivamente y juntas forman una ruta completa $\hat{U} = (\hat{u}_1, \hat{u}_2, \hat{u}_3)$ que es infactible por no satisfacer (2). Por tanto, debido a (6.22), podrían formar parte de la solución como máximo 2 de estas rutas simples.

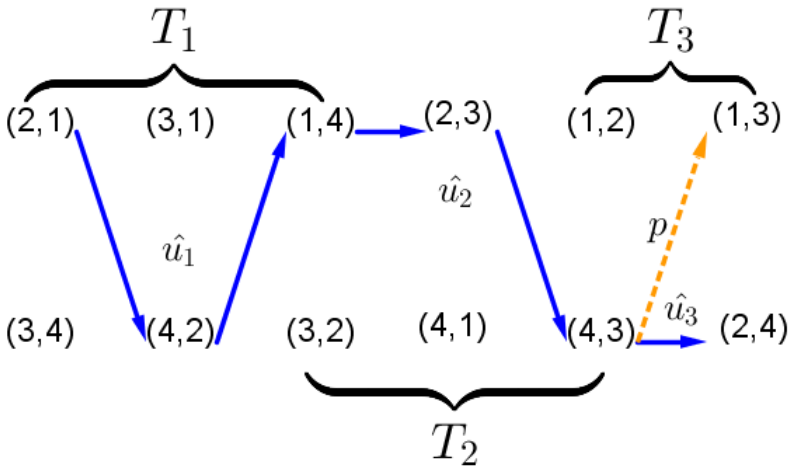

FIGURA 6.1: Representación de $H^+(\hat{U})$

6.3.2. Desigualdades válidas

Con la relajación lineal de esta formulación no obtenemos muy buenas cotas porque la restricción (6.22) es muy débil por lo que vamos a dar una restricción alternativa más exigente. Sea $U = (u_1, u_2, \dots)$ una ruta infactible y $H^+(U) = P(U) \cap \{p \in P | (u_1, \dots, u_i, p)$ es una ruta infactible para algún $i = 1...|P(U)| - 1\}$. Reforzamos la restricción (6.22) sustituyéndola por (6.24):

$$\sum_{p \in H^+(U)} x_p \leq |P(U)| - 1, \quad \forall U \in \mathbb{U}. \tag{6.24}$$

Para nuestra ruta infactible $\hat{U}$ el conjunto $H^+(\hat{U})$ estaría formado por las tres rutas simples de $P(\hat{U})$, junto con la ruta simple $(4,3) - (2,4)$ de P_3 ya que la ruta completa $(\hat{u_1}, \hat{u_2}, p)$ no contendría ningún partido jugado en el estadio del tercer equipo y por tanto también sería infactible como se puede ver en la Figura 6.1.

La validez de esta restricción reside en que si en la solución tenemos $|P(U)|$ rutas simples de $H^+(U)$ no pueden ser todas de $P(U)$ pues en tal caso no cumpliríamos (6.22). Por otro lado, no puede haber dos de estas rutas simples que pertenezcan a la misma sección debido a (6.21) ya que ambas cubrirían el mismo primer partido. Así que si tenemos en la solución $|P(U)|$ rutas simples de $H^+(U)$ deben ser cada una de un P_s distinto y por tanto entre esas rutas se debería encontrar u_1. En esta situación no podría haber ninguna ruta simple de $H^+(U)$ no perteneciente a $P(U)$ ya que por construcción si p es la primera ruta simple de tal conjunto ocupando la partición i-ésima tras la partición relativa a u_1, por definición de $H^+(U)$, la ruta $(u_1, \dots, u_{i-1}, p)$ que es infactible estaría en la solución, llegándose a una contradicción. De este modo deducimos la validez de (6.24).

Análogamente, aprovechando la simetría del TUP con respecto a organizar el torneo desde la primera ronda a la última o viceversa, definimos el conjunto $H^-(U) = P(U) \cup \{p \in P | (p, u_i, u_{i+1}, \ldots, u_{|P(U)|}) \text{ es una ruta infactible para algún } i = 2, ..., |P(U)| - 1\}$ obteniéndose la restricción (6.25) que restringe aún más el poliedro de la relajación lineal de las restricciones (6.21)-(6.24)como muestran las mejoras en la relajación lineal de las instancias resueltas en Oliveira (2016)[11].

$$\sum_{p \in H^-(U)} x_p \leq |P(U)| - 1, \quad \forall U \in \mathbb{U}. \tag{6.25}$$

A continuación presentaremos dos familias de desigualdades válidas para nuestro problema que se basan en traducir nuestro problema a un grafo y buscar subgrafos completos conflictivos en él.

Sea $1 \leq s \leq |S| - 1$ y $g \in G_s \cap G_{s+1}$, definimos A_{sg} el grafo cuyos vértices se corresponden con las rutas simples en P_s y P_{s+1} que empiezan o acaban con el partido g, denotando al vértice correspondiente a la ruta $p \in P_s$, como $v^A_{sg}(p)$. Dos de estos vértices en A_{sg}, $v^A_{sg}(p_1)$ y $v^A_{sg}(p_2)$, serán adyacentes si y solo si p_1 y p_2 pertenecen a la misma sección o constituyen juntos una ruta infactible. Denotando por $\mathbb{A}_{sg}$ el conjunto de subgrafos completos de A_{sg} se tiene la siguiente desigualdad válida:

$$\sum_{p: v^A_{sg}(p) \in C} x_p \leq 1, \quad \forall 1 \leq s \leq |S| - 1, g \in G_s \cap G_{s+1}, C \in \mathbb{A}_{sg}. \tag{6.26}$$

De manera análoga, para introducir la segunda familia de desigualdades tomamos $s \in S$ y definimos B_s el grafo cuyos vértices se corresponden con las rutas simples de P_s. Siguiendo una notación similar a la usada en la formulación (6.26) denotaremos al vértice de este grafo relativo a la ruta p por $v^B_s(p)$ y por $\mathbb{B}_s$ al conjunto de subgrafos completos de B_s. Dos de estos vértices, $v^B_s(p_1)$ y $v^B_s(p_2)$ serán adyacentes si y solo si p_1 y p_2 tienen un partido en común. De este modo, si dos vértices son adyacentes sus correspondientes rutas no pueden formar parte a la vez de la solución de donde se deduce la validez de (6.27) puesto que en un subgrafo completo todos los vertices son adyacentes.

$$\sum_{p: v^B_s(p) \in C} x_p \leq 1, \quad \forall 1 \leq s \leq |S|, C \in \mathbb{B}_s. \tag{6.27}$$

Volvamos sobre el Ejemplo 6.2 y calculemos B_1. Para simplificar, denotaremos cada $p \in P_1$ con un vector binario de tres coordenadas que indican si p contiene al primer o

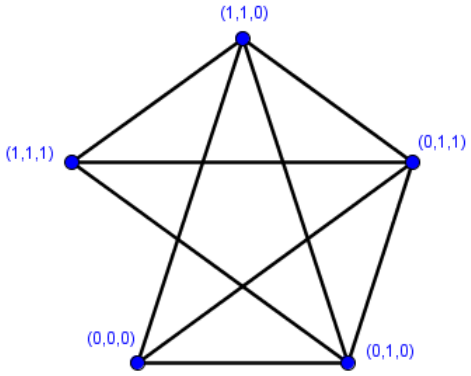

FIGURA 6.2: Representación de B_1

segundo partido de cada jornada facilitándonos comprobar si dos rutas simples contienen un partido en común:

P_1 con $d_1 = d_2 = 1$ y $w = 3$	Expresión en binario
(2,1)-(3,1)-(1,4)	(000)
(2,1)-(4,2)-(1,4)	(010)
(2,1)-(4,2)-(3,2)	(011)
(3,4)-(4,2)-(1,4)	(110)
(3,4)-(4,2)-(3,2)	(111)

El grafo B_1 representado en la Figura 6.2 contiene diversos subgrafos completos como por ejemplo el subgrafo inducido por todas las rutas simples menos (000) forzándose por (6.27) que no tengamos dos o más de estas cuatro rutas simples en la solución.

La razón por la que buscamos subgrafos completos es que al estar el lado derecho de la desigualdad fijo, intentaremos sumar todas las variables que podamos en el izquierdo para que la restricción resultante sea los más estricta posible. Así mismo si tuviésemos que tres variables binarias, x_1, x_2 y x_3, son incompatibles dos a dos tendríamos estas cuatro desigualdades válidas

$$\begin{aligned} x_1 + x_2 &\leq 1, \\ x_1 + x_3 &\leq 1, \\ x_2 + x_3 &\leq 1 \quad y \\ x_1 + x_2 + x_3 &\leq 1, \end{aligned}$$

siendo la más favorable la última de ellas ya que es la única que excluiría la solución $x_1 = x_2 = x_3 = 0{,}5$ de la relajación lineal por lo que es más estricta que las otras tres juntas.

El problema es que las restricciones (6.24)-(6.27) crecen de manera exponencial con n por lo que es inviable introducirlas todas en el software. Por esta razón, en Oliveira (2016)[11] se introduce un algoritmo de separación con el que tras resolver la relajación lineal de (6.21) se extrae una de estas restricciones que no se satisfaga y se introduce en el problema.

Dicho algoritmo queda fuera de las pretensiones de este trabajo y nos limitaremos solo a mencionar su existencia de modo que podamos hacernos una idea, dada la reciente publicación del artículo al que pertenece, de la situación actual del TUP.

6.4. Resolución de algunas instancias del TUP

En esta sección resolveremos algunas instancias del TUP y estudiaremos la influencia de las restricciones con las que se ha ido reforzando la formulación original.

Al igual que para el STSP, hemos elaborado un programa que elabora calendarios deportivos de una manera aleatoria, como el que se puede ver en la Figura 6.3 que muestra el calendario de una liga para 12 equipos que será el tamaño máximo que llegaremos a considerar en nuestras instancias.

En esta ocasión usaremos el preprocesamiento de Xpress para conseguir resolver instancias de mayor tamaño ya que tendremos problemas para resolver instancias del TUP para valores no demasiado grandes de n.

En el Cuadro 6.1 están recogidas las instancias resueltas indicando su tamaño mediante el parámetro n y el valor de los parámetros d_1 y d_2 que condicionan la frecuencia con la que los árbitros pueden intervenir en los partidos de cada equipo. Así mismo, exponemos en las filas $Tiempo1$, $Tiempo2$ y $Tiempo3$ el tiempo en segundos de resolución para las formulaciones (6.2)-(6.9), (6.2)-(6.14) y (6.2)-(6.18), incluyendo en todas ellas las restricciones (6.19) para romper la simetría en la primera jornada. Observamos que fijar d_2 a cero es muy restrictivo, sobre todo a medida que aumenta n. También podemos comprobar que el primer grupo de restricciones no es muy eficiente ya que funciona notablemente peor que los otros dos y además presenta una relajación lineal muy débil. Se pone de manifiesto la gran dificultad de este problema para el que hemos necesitado 39 horas en la resolución de una instancia de 12 equipos.

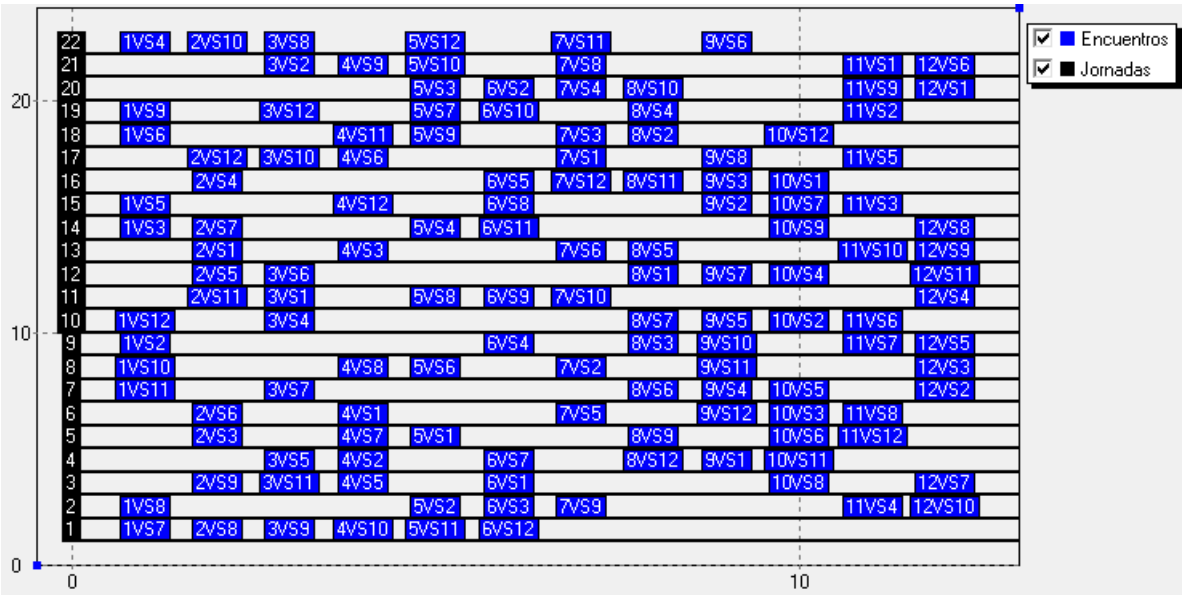

FIGURA 6.3: Ejemplo del calendario de una liga para 12 equipos

CUADRO 6.1: Resultados del TUP

Instancia	1	2	3	4	4	5	5	5	6
n	2	2	3	4	4	5	5	5	6
(d_1, d_2)	(0,0)	(0,0)	(0,0)	(0,0)	(0,1)	(1,1)	(0,1)	(1,0)	(1,1)
V. óptimo	7306	4913	12426	Inf.	20139	31208	32615	Inf.	45498
Tiempo1	0	0	0.5		1.0	>2h	>2h		>2h
RL1	7306	4913	1456		1139	1763	1763		2194
Gap1	0 %	0 %	88 %		94 %	94 %	95 %		95 %
Tiempo2	0	0	0.2		1.0	36.3	118.4		>2h
RL2	7306	4913	6621		9356	20827	21571		38965
Gap2	0 %	0 %	47 %		54 %	33 %	34 %		15 %
Tiempo3	0	0	0.1		4.8	23.4	92.7		39.5
RL3	7306	4913	11142		18345	26840	27107		38865
Gap3	0 %	0 %	11 %		9 %	14 %	17 %		15 %

Capítulo 7

Conclusiones y trabajo futuro

A lo largo del trabajo hemos desarrollado la teoría necesaria para elaborar varios métodos de resolución de nuestros problemas. Hemos visto cómo la teoría de flujos se convierte en una gran aliada en el STSP ya que gracias a ella hemos sido capaces de desarrollar formulaciones mucho más potentes para este problema sobre todo cuando las combinamos con el preprocesamiento de Xpress. Esta simbiosis nos propone estudiar estas técnicas de preprocesamiento y así comprender en qué consisten y por qué mejoran tanto en el STSP la eficacia de nuestras formulaciones.

Por otro lado, la posibilidad de reducir algunas instancias del TUP al TSP hace que el TUP cobre una importancia doble ya que las herramientas que hemos expuesto para este problema también nos sirven para resolver instancias del TSP y por tanto también del STSP. Sin embargo, pese a la gran elaboración de las formulaciones del TUP no podemos resolver grandes instancias incluso contando para ello con el apoyo del preprocesamiento de Xpress. Esto hace inviable adaptar estas técnicas a la resolución del TSP y del STSP ya que para doce ciudades ya tendríamos serios problemas mientras que hasta la formulación en etapas del STSP, que no se basa en la teoría de flujos, es capaz de resolver instancias de ese tamaño en menos de un minuto sin hacer uso de dicho preprocesamiento. No obstante, aún queda por explorar la formulación de la Sección 6.3 en la que podemos estudiar la influencia del parámetro w, que marca la longitud de las secciones en las que dividimos las rondas del torneo, en la eficacia de la formulación. Mediante este nuevo procedimiento además se presenta un método de resolución alternativo en el que se van introduciendo las restricciones de una en una. Esta técnica, que nos invita a adentrarnos en el ámbito heurístico para exprimir todo su potencial, es también aplicable al resto de problemas de Optimización Entera reforzando el algoritmo de ramificación y acotación.

Por otro lado, queda pendiente para el caso del TUP desarrollar la teoría expuesta en Yildiz (2008) [17], en donde se estudia la influencia de los parámetros d_1 y d_2 en la factibilidad del problema lo cual puede resultar útil ya que las instancias difíciles en las que d_1 y d_2 toman el valor cero suelen ser infactibles.

En el ámbito del STSP también podemos usar la heurística empleando como referencia el valor óptimo de la relajación de la formulación multiflujo ya que este valor se ajusta bastante bien al valor óptimo del problema.

Es probable que ninguno de nuestros problemas se pueda resolver en tiempo polinómico pero en tal caso no debemos desalentarnos pues también es posible que aún así seamos capaces de resolver instancias lo suficientemente grandes para abordar la mayoría de problemas de la vida real, bien de manera exacta o mediante procesos heurísticos en los que obtengamos soluciones aproximadas a nuestros problemas cuando su resolución exacta nos desborde debido a sus grandes dimensiones.

Bibliografía

[1] R. K. Ahuja, T. L. Magnanti y J. B. Orlin, *Network Flows: Theory, Algorithmsm and Applications*, Prentice Hall (1993).

[2] G. P. Cornuéjols, C. Fonlupt y D. Naddef, *The traveling salesman problem on a graph and some related integer polyedra*, Math. Program. 33 (1985), pp. 1-27.

[3] G. B. Dantzig, D. R. Fulkerson y S. M. Johnson, *Solution of a large-scale traveling salesman problem*, Oper. Res. 2 (1954), pp. 363-410.

[4] B. Fleischman, *A cutting plane procedure for the traveling salesman problem on a road network*, Eur. J. Oper. Res. 21 (1985), pp. 307-317.

[5] D. Gale, *A theorem of flows in networks*, Pacific J. Math 7 (1957), pp. 1073-1082.

[6] M. Garey y D. S. Johnson, *Computers and Intractability*, W.H. Freeman (1979).

[7] D. J. Houck, J. C. Picard y R.R. Vemuganti, *The traveling salesman problem as a shortest path problem: theory and computational experience*, Opsearch 17 (1980), pp. 93-109.

[8] A. N. Letchford, S. D. Nasiri y D. O. Theis, *Compact formulations of the Steiner Traveling Salesman Problem and related problems*, Eur. J. Oper. Res. 228 (2013), pp. 83-92.

[9] D. Naddef, *Polyhedral Theory and Branch and cut algotithms for the Symmetric TSP*, in G. Gutin, A Punnen (Eds.), *The Traveling Salesman Problem and its variations*, Kluwer Academic Publishers (2002), pp.29-116.

[10] L. Oliveira, C. C. Souza y textscT. Yunes, *Improved bounds for the traveling umpire problem: A stronger formulation and relax-and-fix heuristic*, Eur. J. Oper. Res. 236 (2014), pp. 592-600.

[11] L. Oliveira, C. C. Souza y H. Yunes, *Lower bounds for large traveling umpire instances: New valid inequalities and a branch-and-cut algorithm*, Comput. Oper. Res. 72 (2016), pp. 147-159.

[12] M. Padberg y G. Rinaldi, *A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems*, SIAM Rev. 33 (1991), pp. 60-100.

[13] T. Toffolo, T. Wauters y M. Trick, *TUP Benchmarks*. Disponible en: http://benchmark.gent.cs.kuleuven.be/tup/en/results/?page=2 [Accesible el 15/6/2016].

[14] M. A. Trick y H. Yildiz, *Benders' cuts guided larged neighborhood search for the traveling umpire problem.*, Lecture Notes in Computer Science 4510 (2007), pp. 332-345 .

[15] M. Trick y H. Yildiz, *Benders' cut guided large neighborhood search for the traveling umpire problem*, Naval Res. Log. 58 (2011), pp. 771-781 .

[16] S. Vajda, *Mathematical Programming*, Addison-Wesley (1961).

[17] H. Yildiz, *Methodologies and applications for scheduling, routing and related problems*, Ph.D. thesis Tepper School of Business. Carneige Mellon University (2008).