TRABAJO FIN DE GRADO

Introducción a la optimización robusta

Realizado por: Cristina Consolación Martínez Ortega

Supervisado por: Eduardo Conde Sánchez

FACULTAD DE MATEMÁTICAS

DPTO DE ESTADÍSTICA E INVESTIGACIÓN OPERATIVA

Índice general

Abstract

Sometimes, we may found a decision problem, different situations in which we have different options and we must decide. The treatment we must give to the problem must take into account the frequency with which the decision maker is in that situation. Sometimes, it may be appropriate to use expected values whereas other times it may not be a good choice. If we speak of decisions that must be made in concrete situations that will not be repeated, at least under the same probabilistic conditions, such as evacuations, emergency services assistance, etc, the decision criterion should not be to select the least expected cost.

In this way, the stochastic programming may not be able to satisfy the requirements of decision making in decision environments characterized by significant uncertainty since it requires assigning probability distributions to the scenarios considered. This is not a trivial exercise for many decision makers because in many cases it can be difficult to calculate the probability of a future scenario occurring when the related factors refer to the behavior of companies, agencies or governments for which we do not have information. In other cases, changes in the attitudes and priorities of potential clients related to the decision problem may have to be considered.

The approach followed in this work, unlike that used in stochastic programming, has the objective of protecting the decision maker without using a probability model on the performance of the system. In the framework of robustness, what the decision maker wants is not the optimal long-term or optimal for a single scenario (although it is the most likely scenario), but a decision that works well in all cases. So a robust decision will be one that is

characterized by significant uncertainty, works well in all cases and is protected against the worst possible scenarios.

The robust discrete optimization problems are in general more difficult to solve than their deterministic counterparts. The major source of complexity comes from the extra degree of freedom - the scenario set. For many polynomially solvable classical optimization problems such as assignment, minimum spanning tree, shortest path, resource allocation, single machine scheduling with sum of flow times criterion, their corresponding robust versions are weakly or strongly $NP-$hard.

The problems we will see have a high complexity, so we will see a numerical method of resolution that can be stopped before reaching the optimum that will lead us to a $\varepsilon-$optimal solution. With this approach we can obtain a method that is computationally more efficient in order to compute good solutions in real life problems.

Capítulo 1

Decisión en ambiente de incertidumbre

1.1. Planteamiento del problema

A menudo podemos encontrarnos con un problema de decisión, distintas situaciones en las que se nos presentan distintas opciones y debemos decidir. El tratamiento que debemos dar al problema deberá tener en cuenta la frecuencia con la que el decisor se encuentra en esa situación. A veces, puede ser adecuado usar valores esperados y en cambio, otras veces puede no ser una buena opción (como veremos basándonos en el libro de S. French [5]). Un ejemplo de esto podría ser elegir la mejor ruta para ir de casa al trabajo. El tiempo requerido para recorrer una ruta concreta es una variable aleatoria debido por ejemplo a la intensidad de tráfico o al efecto de los semáforos, pero si en este caso tenemos en cuenta valores esperados, y optamos por elegir la ruta con la que, en media, tardamos menos, la Ley Fuerte de los Grandes Números nos dice que si aplicamos este criterio de decisión reiteradamente en las mismas condiciones probabilísticas, el tiempo promedio

empleado se acerca cada vez más al óptimo. Pero en cambio, si hablamos de decisiones que deben ser tomadas en situaciones concretas que no se volverán a repetir, al menos bajo las mismas condiciones probabilísticas, como evacuaciones, asistencia de servicios de emergencia, etc. el criterio de decisión no debería ser el de seleccionar la ruta de menor tiempo esperado. En este caso es necesario reforzar la toma de decisiones de forma que se proteja al decisor contra eventualiades que pudiesen ocurrir. Esto da lugar a lo que se denomina optimización robusta. Además, dentro del marco de la programación estocástica, se necesita conocer las distribuciones de probabilidad de cada una de las rutas posibles y eso puede no ser una tarea fácil. Por lo tanto, como podemos ver, tomar una buena decisión no es tan sencillo.

1.2. Tablas de decisión

En esta sección veremos cómo podemos representar los problemas de decisión sin perder ninguna de sus características esenciales mediante lo que llamaremos una tabla de decisión. La consecuencia de cualquier acción viene determinada no sólo por la acción en sí misma sino también por un número de factores externos. Estos factores están habitualmente fuera del control del decisor y suelen ser desconocidos en el momento en el que se toma la decisión. A cada realización posible la denominaremos escenario y conocido el escenario podríamos predecir las consecuencias de cada acción con certeza. En la tabla de decisión representaremos los distintos escenarios que nos podríamos encontrar como el conjunto $\{\theta^s : s \in S\} = \left\{\theta^1, \theta^2, \ldots, \theta^n\right\}$ y las distintas acciones o alternativas que podemos llevar a cabo por el conjunto $\{X : x \in X\} = \left\{x^1, x^2, \ldots, x^m\right\}$ con $n, m \in \mathbb{Z}$. En el caso en el que las acciones sean elegir qué persona ocupará un determinado puesto de trabajo, como ocurre en los problemas de contratación de personal, los θ^j se suelen

identificar, no como posibles escenarios sino como cualidades o atributos de los diferentes candidatos al puesto de trabajo. En esta situación un criterio de decisión determinará de manera objetiva la decisión óptima de manera objetiva y cuantificable en función de la valoración de los candidatos en cada una de las características tenidas en cuenta.

Para los criterios de decisión que presentaremos en este tema, asumiremos que tanto los escenarios como las acciones son finitos pero, podríamos esperar que fuesen infinitos y en tal caso, las herramientas matemáticas que deberán usarse serán diferentes.

Como hemos dicho, los factores externos no son conocidos por el decisor en el momento en que éste toma la decisión. Pero una vez elegida la acción x^i, tiene lugar uno y sólo uno de los escenarios. A este escenario lo llamaremos escenario real. De esta forma, llamaremos v_{ij} al valor de la consecuencia de elegir la acción x^i cuando θ^j es el escenario real. Estas valoraciones representan algo positivo, es decir, cuanto mayores son, mejor. Se trata por tanto de beneficios obtenidos por el decisor que elige una acción bajo un determinado escenario o de la puntuación de una característica positiva para un posible candidato a un puesto de trabajo.

Por tanto, una tabla de decisión viene representada como vemos en la tabla 1,1.

Los problemas de decisión se clasifican según el conocimiento de quien toma la decisión sobre el escenario real de la siguiente forma.

- *Decisiones bajo certeza.* En este caso, el decisor sabe cuál es el escenario real y plantea las distintas acciones. Es equivalente a tomar $n = 1$ y considerar un único escenario. En dicho escenario, v_{i1} representa el

	Escenarios	θ^1	θ^2	$\cdots$	θ^n
Acciones	x^1	v_{11}	v_{12}	$\cdots$	v_{1n}
	x^2	v_{21}	v_{22}	$\cdots$	v_{2n}
	.	.	.	$\cdots$	.
	.	.	.	$\cdots$	.
	.	.	.	$\cdots$	.
	x^m	v_{m1}	v_{m2}	$\cdots$	v_{mn}

Tabla 1.1: Tabla de decisión en la que se valora cada acción según cada uno de los escenarios o atributos.

valor objetivo que es optimizado en el proceso de toma de decisiones.

- *Decisiones con riesgo.* Aunque en este caso el decisor no conoce el escenario real con certeza, puede cuantificar la incertidumbre a través de distribuciones de probabilidad $(P(\theta^1), \ldots, P(\theta^n))$. Esta información puede ser utilizada para optimizar criterios como el del valor esperado, es decir, elegir x^i tal que maximice $\sum P(\theta^j) \cdot v_{ij}$.

- *Decisiones bajo incertidumbre estricta.* En este otro caso, el decisor solo sabe el conjunto de posibles escenarios que pueden ocurrir tras tomar la decisión, pero no puede cuantificar su incertidumbre en modo alguno. Es en este tipo de situaciones ante una decisión en las que nos vamos a centrar a lo largo de este trabajo.

1.3. Criterios de decisión bajo incertidumbre estricta

Dada una situación de incertidumbre estricta, vamos a analizar algunos criterios que se han utilizado en la literatura a la hora de elegir una de las decisiones posibles y la *racionalidad* que conlleva su uso.

1.3.1. Criterio maximin de Wald.

A la hora de valorar la acción x^i tendremos en cuenta el escenario que le sería más desfavorable y al valor que obtiene bajo dicho escenario lo llamaremos s_i. Puesto que es un problema de beneficios, buscamos el menor valor que será de la forma

$$s_i = \min_{j=1,\dots,n} \{v_{ij}\}$$

Llamaremos a s_i el nivel de seguridad para la acción x^i. Esto garantiza al decisor un retorno de al menos s_i. Wald propuso en 1950 el criterio en el que el decisor debía elegir x^k de manera que el nivel de seguridad fuese lo mayor posible. Por tanto, el criterio de máxima recompensa de Wald consiste en:

Elegir x^k de manera que

$$s_k = \max_{i=1,\dots,m} \{s_i\} = \max_{i=1,\dots,m} \left\{ \min_{j=1,\dots,m} \{v_{ij}\} \right\}.$$

Podemos apreciar que este es un criterio de decisión muy pesimista, ya que el procedimiento general es asumir que lo que va a ocurrir es lo peor posible para cada una de las acciones consideradas. En la literatura se dice que este tipo de criterios es de carácter "*muy conservador*" por lo que su empleo en problemas de toma de decisión es bastante limitado.

1.3.2. Criterio maximax de Hurwicz.

En contraposición con este criterio podemos encontrar un criterio optimista considerando la mejor consecuencia posible para cada acción. De esta forma, se define el nivel optimista para x^i como

$$o_i = \max_{j=1,\dots,n} \{v_{ij}\}.$$

Así, o_i es el valor de la acción x^i bajo el mejor escenario que se podría dar. Este criterio es de la forma:

Elegir x^k tal que

$$o_k = \max_{i=1,\ldots,m} \{o_i\} = \max_{i=1,\ldots,m} \left\{ \max_{j=1,\ldots,n} \{v_{ij}\} \right\}.$$

Quizás, tanto el criterio optimista como el pesimista no sean demasiado adecuados para la toma de decisiones en problemas reales. Hurwicz en 1951 expone que tanto el criterio optimista como el pesimista son demasiado extremos ante la situación que podría ocurrir. Por eso, propone un criterio intermedio. De hecho, argumentó que el proceso de toma de decisones debe ser de acuerdo a un promedio ponderado de los niveles de seguridad y optimista : $s_i + (1-\alpha)o_i$, donde α, $0 \leq \alpha \leq 1$, es el índice optimista-pesimista del decisor. Este α, es un número específico e individual para el decisor y, además, se puede aplicar para cualquier otro problema de decisión al que se enfrente. Hurwicz recomienda que la decisión debe ser tomada de la forma:

$$\text{elegir } x^k \text{ tal que} \quad \alpha s_k + (1-\alpha)o_k = \max_{i=1,\ldots,m} \{\alpha s_i + (1-\alpha)o_i\}.$$

Por supuesto, para utilizar este criterio es necesario determinar α. Como ya hemos dicho, α es específico para el individuo y aplicable para todos los problemas. Para determinar el valor α es necesario realizar un experimento en el que el decisor debe poner "*el precio justo*" a un juego en el que ganara una unidad si se diese uno de los estados de la naturaleza y no ganara nada en caso contrario.

	θ^1	θ^2	s_i	o_i	$\alpha s_i + (1-\alpha)o_i$
x^1	1	0	0	1	$(1-\alpha)$
x^2	v	v	v	v	v

De donde sacamos

$$(1-\alpha) = v \Rightarrow \alpha = (1-v)$$

Así el decisor determina el α, y la cantidad v varía hasta expresar la indiferencia.

1.3.3. Criterio minimax regret de Savage.

En 1951, Savage propone comparar el valor de la consecuencia de una acción bajo un escenario con el valor de la acción que es óptima para dicho escenario.

Así, Savage define el regret de una consecuencia (r_{ij}) como la diferencia entre el valor resultante de la mejor acción dado que θ^j es el escenario real y el valor resultante de x^i bajo θ^j. Entendiendo siempre el regret como pérdida de beneficios frente a la decisión que deberíamos haber tomado si hubiésemos conocido el escenario que iba a ocurrir (arrepentimiento, "*regret*", frente a lo que deberíamos haber hecho).

$$r_{ij} = \underset{l=1,\ldots,m}{\text{máx}} \{v_{lj}\} - v_{ij}$$

Razona que el valor de v_{ij} debería ser reemplazado en la tabla de decisiones por r_{ij} y que en esta nueva tabla de arrepentimiento el decisor debería elegir siguiendo un enfoque pesimista, pero recordando que el arrepentimiento no son ganancias sino pérdidas. Así, este criterio aconseja que cada acción debe tener asignado el índice

$$\rho_i = \underset{j=1,\ldots,n}{\text{máx}} \{r_{ij}\} = \text{peor regret resultante bajo la acción } x^i$$

y entonces elegir una acción que minimiza ρ_i.

Elegir x^k tal que

$$\rho_k = \underset{i=1,\ldots,m}{\text{mín}} \{\rho_i\} = \underset{i=1,\ldots,m}{\text{mín}} \left\{ \underset{i=1,\ldots,n}{\text{máx}} \{r_{ij}\} \right\}$$

1.3.4. Criterio maxisum de Laplace.

Este criterio es conocido como criterio de razón insuficiente. En él Laplace expone que no conocer nada en absoluto sobre el escenario real es equivalente a considerar que todos los escenarios tienen la misma probabilidad de ocurrir. De esta forma, podría justificar la elección de una acción maximizando el valor de la media.

Así, lo que buscamos es maximizar el valor esperado de la acción x^i. Por lo tanto, debemos

elegir x^k tal que

$$\sum_{j=1}^{n} \left(\frac{1}{n}\right) \cdot v_{kj} = \max_{i=1,\ldots,n} \left\{ \sum_{j=1}^{n} \frac{1}{n} \cdot v_{ij} \right\}.$$

O directamente, elegir la acción que maximiza la suma de valoraciones.

1.3.5. Ejemplo de Milnor.

Lo primero que podemos apreciar al ver estos criterios de decisión, es que pueden discrepar a la hora de elegir la alternativa óptima. Veamos un problema donde realmente, cada uno de los criterios tendría una elección distina. Sean cuatro escenarios θ^j y cuatro acciones x^i propuestas por Milnor en 1954 como se muestra en la tabla 1.2.

	θ^1	θ^2	θ^3	θ^4	s_i	o_i	$\sum_j (1/n)v_{ij}$
x^1	2	2	0	1	0	2	5/4
x^2	1	1	1	1	1	1	1
x^3	0	4	0	0	0	4	1
x^4	1	3	0	0	0	3	1

Tabla 1.2: Ejemplo de Milnor.

Para este problema, el criterio de Laplace elegirá la acción x^1, mientras que el criterio de Wald elegirá x^2. El criterio de Hurwicz asigna a las acciones x^1,x^2, x^3 y x^4 los valores $2(1-\alpha)$, 1, $4(1-\alpha)$ y $3(1-\alpha)$ respectivamente. Se tiene que $4(1-\alpha) > 2(1-\alpha)$ y $4(1-\alpha) > 3(1-\alpha)$ $\forall 0 \leq \alpha < 1$, luego $\forall \alpha < \frac{3}{4}$, $4(1-\alpha) > 1$. Por tanto, el criterio de Hurwicz elige x^3 $\forall \alpha < \frac{3}{4}$. El criterio de minimax regret como hemos visto, elige la acción según el mínimo de los ρ_i y por tanto elige la acción x^4.

r_{ij}	θ^1	θ^2	θ^3	θ^4	ρ_i
x^1	0	2	1	0	2
x^2	1	3	0	0	3
x^3	2	0	1	1	2
x^4	1	1	1	0	1

Tabla 1.3: Regret para el ejemplo de Milnor.

Por lo que podemos ver que es posible que cada criterio elija un acción distinta.

1.4. Principios de racionalidad

En este capítulo hemos mencionado cuatro criterios de decisión en situaciones de incertidumbre estricta. Como se ha visto, los criterios discrepan, para compararlos establecemos principios de racionalidad que actúen como hipótesis que serán asumidas o no por un decisor "*racional*". Se verán a continuación los principales principios de racionalidad que consideramos que deben cumplir estos criterios.

1.4.1. Principio: Clasificación completa.

Este principio implica que una regla de decisión debe, explícita o implícitamente, asignar un índice numérico a cada acción y clasificarlas en orden

creciente (o decreciente) de dicho índice. De aquí en adelante podemos considerar una regla de decisión no especificada y suponer que se asigna el índice V_i a la acción x^i, de tal forma que si $V_i > V_j \ \forall i \neq j$ el criterio considera x^i mejor que x^j.

Si expresamos de esta forma los cuatro criterios que hemos visto obtenemos lo siguiente:

Criterio de máxima recompensa de Wald	$V_i = s_i$;
Criterio optimista pesimista de Hurwicz	$V_i = \alpha s_i + (1-\alpha) o_i$;
Criterio de minimax regret	$V_i = -\rho_i$;
Criterio de razón insuficiente de Laplace	$V_i = \sum_j \left(\frac{1}{n}\right) v_{ij}$.

1.4.2. Principio: Independencia de etiquetado.

Este principio establece el requisito de que la elección sugerida por un criterio no debe verse influenciada por la forma en que vienen representadas las alternativas frente a los distintos escenarios. Por tanto, se exige que la clasificación final de las acciones sea independiente tanto del orden en el que hemos etiquetado la acción en la tabla de decisiones, como del orden en el que hemos etiquetado los escenarios.

De manera general, sea una tabla de decisión $m \times n$ con escenarios θ^j, acciones x^i y valores v_{ij} dados. Sea una segunda tabla de decisión $m \times n$, con escenarios $\theta^{j'}$, acciones $x^{'}$ y valores $v^{'}_{ij}$ construida a partir de la primera permutando las filas y las columnas tales que, la fila i se convierte en la fila $\pi(i)$ y la columna j en la columna $\pi(j)$.

$$v^{'}_{\pi(i)\pi(j)} = v_{ij}.$$

De esta forma, una regla de decisión asigna los valores V_r y $V_r^{'}$ respectivamente a las acciones en las dos tablas de forma que

$$V_i > V_k \Leftrightarrow V_{\pi(i)}^{'} > V_{\pi(k)}^{'}, \quad \forall \quad 1 \leq i, k \leq m.$$

1.4.3. Principio: Independencia de escala de valores.

Observemos que si cambiamos de una escala de medida a otra, cada uno de los valores v_{ij} se convertiría en $(\delta v_{ij} + \beta)$, donde $\delta > 0$ es el factor de conversión y β el ajuste del cambio desde el cero. Por lo que deberíamos exigir que la elección sugerida por una regla de decisión sea independiente de la escala en la que medimos el valor.

De manera general, sea una tabla de decisión $m \times n$ con escenarios θ_j, acciones a_i y valores v_{ij} dados. Sea una segunda tabla de decisión $m \times n$, con escenarios θ_j, acciones a_i, pero diferentes valores $v_{ij}^{'}$ construidos a partir de la primera cambiando la escala de medición del valor, con $\delta > 0$ y β fijos como sigue

$$v_{ij}^{'} = \delta v_{ij} + \beta$$

Bajo este principio de racionalidad, una regla de decisión asigna los valores V_r y $V_r^{'}$ respectivamente a las acciones en las dos tablas de forma que

$$V_i > V_k \Leftrightarrow V_i^{'} > V_k^{'}, \quad \forall \quad 1 \leq i, k \leq m.$$

1.4.4. Principio: Dominación fuerte.

Este principio establece que si una acción tiene mejor valoración estrictamente que otra, cualquiera que sea el escenario, entonces claramente la primera acción debe ser preferida a la segunda. Consideremos, por ejemplo, la tabla de decisión 2×5 siguiente

	θ_1	θ_2	θ_3	θ_4	θ_5
a_1	7	8	6	3	5
a_2	4	2	5	-7	1

En este caso, el principio establece que la acción a_1 debería ser preferida a la acción a_2, porque cualquiera que sea el escenario conduce a una mejor consecuencia en sentido estricto.

Supongamos que en una tabla de decisión hay dos alternativas a_i y a_k tal que $\forall$ escenario θ_j

$$v_{ij} > v_{kj}.$$

Entonces una regla de decisión debe asignar los valores V_r tal que

$$V_i > V_k.$$

1.4.5. Principio: Independencia de alternativas irrelevantes.

Este principio exige que la clasificación de dos acciones en una tabla de decisión debe ser independiente a las demás acciones posibles. De manera general, sea una tabla de decisión $m \times n$ con escenarios θ_j, acciones a_i y valores v_{ij} dados. Sea una segunda tabla construida a partir de la primera simplemente añadiendo una acción extra. Así, la segunda tabla es $(m+1) \times n$ y

$$v_{ij}^{'} = v_{ij} \quad \forall \quad 1 \leq i \leq n \ , 1 \leq j \leq n.$$

Para $1 \leq j \leq n$, $v_{(m+1)j}^{'}$ puede tomar cualquier valor numérico. Entonces, una regla de decisión asigna los valores V_r y $V_r^{'}$ respectivamente a las acciones de las dos tablas tal que

$$V_i > V_k \Leftrightarrow V_i^{'} > V_k^{'} \quad \forall \quad 1 \leq i, k < m$$

Esto es que una regla de decisión debe conducir a la misma clasificación de las primeras m acciones en ambas tablas.

1.4.6. Principio: Independencia de la adición de una constante a una columna.

Considérense las dos tablas 2×2 siguientes

	θ_1	θ_2
a_1	6	4
a_2	3	8

	$\theta_1^{'}$	$\theta_2^{'}$
$a_1^{'}$	16	4
$a_2^{'}$	13	8

La diferencia entre ambas tablas es que se ha añadido a ambos elementos de la primera columna la constante 10 para pasar de la primera a la segunda. Podemos ver que la decisión que representa la primera tabla es exactamente la misma que representa la segunda. Bajo el primer escenario, la primera alternativa es 3 unidades mejor que la segunda; bajo el segundo escenario es 4 unidades peor. Por lo que cabría esperar que una regla de decisión eligiera la misma acción en ambas tablas.

Pues bien, sea una tabla de decisión $m \times n$ con escenarios θ_j, alternativas a_i y valores v_{ij} dados. Sea una segunda tabla con los mismos escenarios θ_j y alternativas a_i pero con diferentes valores v_{ij}, construidos a partir de la primera añadiendo una constante a todos los elementos de la $l-$ésima columna tal que para $j \neq l, 1 \leq j \leq n, 1 \leq i \leq m$:

$$v_{il}^{'} = v_{il} + c$$

$$v_{ij}^{'} = v_{ij}$$

Entonces, una regla de decisión debe asignar los valores V_r y $V_r^{'}$ respectivamente, a las acciones en las dos tablas tal que

$$V_i > V_k \Leftrightarrow V_i^{'} > V_k^{'} \quad \forall \quad 1 \leq i, k \leq m$$

Los principios vistos hasta ahora no precisan restricción acerca de la incertidumbre existente sobre el escenario que ocurrirá al tomar la decisión, es decir, son requisitos para cualquier regla de decisión, sin necesidad de encontrarnos bajo incertidumbre estricta. Los dos siguientes principios se centran claramente en el ámbito de la incertidumbre estricta.
Si efectivamente el decisor no conoce nada acerca del escenario real, es fácil suponer que no habría diferencia entre las alternativas de la tabla siguiente

	θ_1	θ_2	θ_3
a_1	6	0	3
a_2	0	6	3

Es claro que el conjunto de posibles consecuencias de a_1 es idéntico en términos de valor a la de a_2. La única diferencia razonable entre ambas es que se producen consecuencias iguales en valor pero bajo distintos escenarios. Parece razonable esperar, que si hacemos una permutación en una fila no debería verse afectada la decisión final.

1.4.7. Principio: Independencia de permutaciones por filas.

Supongamos que en una tabla de decisión $m \times n$ hay dos acciones a_i y a_k, y una permutación τ para los números $\{1, 2, \cdots, n\}$ tal que

$$v_{ij} = v_{k\tau(j)}.$$

Bajo este principio de racionalidad, una regla de decisión debería asignar el valor V_r a las acciones tal que

$$V_i = V_k.$$

Podría decirse que el principio 1.4.7 no es suficiente para caracterizar lo que entendemos por circunstancias de incertidumbre estricta. Considérense los dos problemas de decisión de las tablas 1.4 y 1.5.

Es de esperar que nos preguntemos ahora si existe realmente alguna diferencia entre estos problemas si es cierto que el escenario real es estrictamente incierto. Si los escenarios $\theta_2^{'}$, $\theta_3^{'}$, $\theta_4^{'}$ y $\theta_5^{'}$ se asocian y se identifican con el escenario compuesto $\theta_2^{''}$, entonces las tablas 1.4 y 1.5 implicarían idénticas consecuencias para dos alternativas consideradas. Realmente una elección que apunte que θ_2 en la tabla 1.4 sea diferente a $\theta_2^{''}$ en la tabla 1.5 no puede ser bajo una situación de incertidumbre estricta. Es cierto que $\theta_2^{''}$ se verifica bajo cualquiera de sus "*versiones*" $\theta_2^{'}$, $\theta_3^{'}$, $\theta_4^{'}$ o $\theta_5^{'}$, pero eso, bajo incertidumbre estricta, no es relevante a la hora de comparar la posibilidad de que ocurra uno de estos dos escenarios $\theta_2^{''}$ y $\theta_1^{'}$. Por lo tanto la acción elegida en la tabla 1.4 debe ser la misma que en la tabla 1.5 si este principio es verificado por el criterio de decisión.

1.4.8. Principio: Independencia de duplicación de columnas.

Sea una tabla de decisión $m \times n$ con escenarios θ_j, alternativas a_i y valores v_{ij} dados. Sea una tabla $m \times (n+1)$ con escenarios $\theta_j^{'}$, alternativas $a_i^{'}$ y valores $v_{ij}^{'}$ construidos a partir de los valores de la primera tabla duplicando la $n-$ésima columna de forma que $\quad \forall \quad 1 \leq j \leq n, \forall \quad 1 \leq i \leq m$

$$\begin{aligned} v_{ij}^{'} &= \quad v_{ij} \\ v_{i(n+1)}^{'} &= \quad v_{in}. \end{aligned}$$

Así, una regla de decisión debe asignar los valores V_r y $V_r^{'}$ respectivamente a las acciones en las dos tablas tal que

$$V_i > V_k \Leftrightarrow V_i^{''} > V_k^{''}, \forall \quad 1 \leq i, k \leq m$$

	θ_1	θ_2
a_1	9	4
a_2	2	6

Tabla 1.4: Ejemplo para el principio 1.4.8

			$\theta_2^{''}$		
	$\theta_1^{'}$	$\theta_2^{'}$	$\theta_3^{'}$	$\theta_4^{'}$	$\theta_5^{'}$
$a_1^{'}$	9	4	4	4	4
$a_2^{'}$	2	6	6	6	6

Tabla 1.5: Ejemplo para el principio 1.4.8

Nótese que aunque este principio sólo duplica la última columna una vez, una duplicación repetida permite que se aplique a situaciones como las de la tabla anterior. Además, la combinación de este principio con el principio 1.4.2 denota que se aplica a situaciones en las que cualquier columna es duplicada.

1.5. Análisis de los principios de racionalidad

En esta sección se comparan los cuatro criterios de decisión que se definieron en la sección 1.3 términos de la base axiomática establecida a través de los principios de racionalidad. Además veremos que no existe ningún criterio que sea compatible con todos los principios considerados.

Teorema 1.5.1 *Los criterios de decisión de Wald, Hurwicz, minimax regret y Laplace verifican la siguiente tabla de compatibilidades respecto a los principios de racionalidad 1.4.1-1.4.8.*

	Wald	Hurwicz	minimax	Laplace	
Principio 1.3.1	✓	✓	✓	✓	Clasificación completa
Principio 1.3.2	✓	✓	✓	✓	Independencia de etiquetado
Principio 1.3.3	✓	✓	✓	✓	Independencia de escala de valores
Principio 1.3.4	✓	✓	✓	✓	Dominación fuerte
Principio 1.3.5	✓	✓	✗	✓	Independencia de alternativas irrelevantes
Principio 1.3.6	✗	✗	✓	✓	Independencia de la adición de una constante a una columna
Principio 1.3.7	✓	✓	✗	✓	Independencia de permutaciones por filas
Principio 1.3.8	✓	✓	✓	✗	Independencia de duplicación de columnas

Tabla 1.6: Compatibilidad de los criterios con los principios de racionalidad. ✓indica que el criterio satisface el principio, ✗indica que no lo satisface.

Demostración 1.5.1

A continuación solo se demostrará la compatibilidad de los criterios con algunos de los principios que consideramos más relevantes, y veremos un contraejemplo para cada uno de los criterios y los principios que incumplen.
El principio 1.4.1 lo satisfacen todos los criterios puesto que, cada criterio asigna un valor numérico a cada alternativa y el orden de estos valores nos da una clasificación completa.
Para demostrar que el criterio maximin de Wald satisface el principio 1.4.2, vemos que si el nivel de seguridad de una acción es el mínimo de los valores de esa acción en los distintos escenarios, podemos asegurar que el nivel de seguridad no cambia en el caso en que permutemos las columnas. Esto se debe a que el mínimo de un conjunto de valores no depende de la posición en que se encuentren los distintos valores.
Veamos que el criterio de Laplace satisface el principio 1.4.3.

Sea $v_{ij} = \delta v_{ij} + \beta$, con $\delta > 0$.

Entonces,

$$\begin{aligned}
V_i &= \sum_{j=1}^{n} \frac{1}{n} v_{ij} > \sum_{j=1}^{n} \frac{1}{n} v_{kj} = V_k; \\
\Leftrightarrow \quad & \delta \left(\sum_{j=1}^{n} \frac{1}{n} v_{ij} \right) + \beta > \delta \left(\sum_{j=1}^{n} \frac{1}{n} v_{kj} \right) + \beta; \\
\Leftrightarrow \quad & \sum_{j=1}^{n} \frac{1}{n} (\delta v_{ij} + \beta) > \sum_{j=1}^{n} \frac{1}{n} (\delta v_{kj} + \beta); \\
\Leftrightarrow V_i^{'} &= \sum_{j=1}^{n} \frac{1}{n} v_{ij}^{'} > \sum_{j=1}^{n} \frac{1}{n} v_{kj}^{'} = V_k^{'}.
\end{aligned}$$

Para demostrar que el criterio maximin de Wald satisface el principio 1.4.4. Supongamos que $v_{ij} > v_{kj}$ para todos los escenarios θ_j. Entonces

$$\begin{aligned}
V_i = s_i &= \min_{j=1,\ldots,n} \{v_{ij}\} = v_{ij_0} \\
&> v_{kj_0} \\
&\geq \min_{j=1,\ldots,n} \{v_{kj}\} = s_k = V_k.
\end{aligned}$$

Veamos que el criterio de Hurwicz satisface el criterio 1.4.5. Para toda acción a_i, $\alpha s_i + (1-\alpha) o_i = \alpha \min_{j=1,\ldots,n} \{v_{ij}\} + (1-\alpha) \max_{j=1,\ldots,n} \{v_{ij}\}$ es independiente de cualquier otra acción. De esta forma, añadir una nueva acción a la tabla no puede afectar a la clasificación del resto de acciones $a_1, a_2, \ldots, a_m$.

Veamos que el criterio de Laplace satisface el principio 1.4.6. Sea

$$\begin{aligned}
v_{il}^{'} &= v_{il} + c \text{ para algún l} \\
v_{ij}^{'} &= v_{ij} \text{ para } j \neq l, 1 \leq j \leq n
\end{aligned}$$

Entonces

$$V_i = \sum_{j=1}^{n} \frac{1}{n} v_{ij} > \sum_{j=1}^{n} \frac{1}{n} v_{kj} = V_k;$$
$$\Leftrightarrow \frac{c}{n} + \sum \frac{1}{n} + v_{ij} > \frac{c}{n} + \sum \frac{1}{n} + v_{kj};$$
$$\Leftrightarrow V_i^{'} = \sum_{j=1}^{n} \frac{1}{n} v_{ij}^{'} > \sum_{j=1}^{n} \frac{1}{n} v_{kj}^{'} = V_k^{'}.$$

Es evidente que el mínimo elemento de la i-ésima fila es independiente de cualquier permutación por filas. Por tanto, el criterio maximin de Wald satisface el principio 1.4.7.

Para ver que el criterio minimax regret satisface el principio 1.4.8, supongamos que duplicamos la n-ésima columna. Entonces

$$r_{i(n+1)}^{'} = \max_{k=1,\ldots,n} \left\{ v_{k(n+1)}^{'} \right\} - v_{i(n+1)}^{'}, \quad 1 \leq i \leq m,$$
$$= \min_{k=1,\ldots,m} \{v_{kn}\} - v_{in} = r_{in}.$$

Ya que $r_{ij}^{'} = r_{ij}$ para $1 \leq i \leq m$ y $1 \leq j \leq n$, $\rho_i^{'} = \rho_i$ para cualquier a_i. Por tanto, la clasificación de las acciones no se ve afectada.

Veamos ahora un contraejemplo para mostrar que el criterio de minimax regret no cumple el criterio 1.4.5. Para ello, consideremos la tabla 1.7 con alternativas a_1 y a_2, luego consideremos la adición de la alternativa a_3. Es fácil comprobar que en el primer caso, $\rho_1 = 3 < 6 = \rho_2$, pero al añadir la acción a_3, $\rho_1^{'} = 9 > 6 = \rho_2^{'}$. Por tanto, no se cumple el criterio 1.4.5. Para mostrar que el criterio minimax regret no cumple el principio 1.4.7 conside-

ramos de nuevo la tabla 1.7. Las consecuencias resultantes de a_1 y a_2 son las mismas, ya que la fila 2 es simplemente una permutación de la fila 1. Pero a_1 y a_2 no mantienen su clasificación al añadir a_3. Luego no cumple el principio 1.4.7.

Veamos que el criterio de Laplace no satisface el principio 1.4.8. Si nos fijamos en las tablas 1.4 y 1.5, en la primera tabla el criterio elige la acción a_1 y en la segunda la acción a_2. Luego no cumple el principio 1.4.8. Para ver que tanto el criterio de máximo retorno de Wald como el criterio de Hurwicz no cumplen el principio 1.4.6, consideremos las tablas 2×2 1.8 y 1.9. Es claro que las clasificaciones proporcionadas en las dos tablas son distintas. Para ver el contraejemplo nos bastaba con encontrar un α de manera que las clasificaciones fueran distintas en cada tabla, en este caso $\alpha = \frac{1}{4}$, pero es fácil comprobar que se obtiene el mismo resultado para cualquier α de forma que $0 \leq \alpha \leq \frac{3}{7}$ y $\frac{2}{3} \leq \alpha \leq 1$.

	θ_1	θ_2	θ_3
a_1	6	0	3
a_2	0	3	6
a_3	2	9	4

Tabla 1.7: Contraejemplo que muestra que el criterio de Savage no satisface los principios 1.4.5 y 1.4.7

	θ_1	θ_2	s_i	0_i	$(\frac{1}{4}s_i + \frac{3}{4}o_i)$
a_1	6	4	4	6	22/4
a_2	3	8	3	8	27/4

Tabla 1.8: Contraejemplo que muestra que los criterios de Wald y Hurwicz no satisfacen el principio 1.4.6

□

	θ_1	θ_2	s_i	0_i	$(\frac{1}{4}s_i + \frac{3}{4}o_i)$
a_1	16	4	4	16	52/4
a_2	13	8	8	13	47/4

Tabla 1.9: Contraejemplo que muestra que los criterios de Wald y Hurwicz no satisfacen el principio 1.4.6

Por lo que ninguno de los cuatro criterios que hemos visto satisface todos los principios de racionalidad expuestos. Pero vamos más allá, ningún criterio es capaz de satisfacer todos los principios. Si somos capaces de encontrar un criterio que cumpla todos los principios desde el 1.4.1 hasta el 1.4.7, entonces este criterio es a todos los efectos el criterio de Laplace. Pero acabamos de dar un contraejemplo de que el criterio de Laplace no cumple el principio 1.4.8, por lo que se sigue que no habrá ningún criterio que satisfaga todos los principios.

Teorema 1.5.2 *Supongamos que un criterio de decisión satisface los principios 1.4.1, 1.4.4, 1.4.5, 1.4.6, 1.4.7. Entonces el criterio asigna un índice numérico V_i a cada acción a_i tal que*

$$V_i \geq V_k \Leftrightarrow \sum_{j=1}^{n} \frac{1}{n} v_{ij} \geq \sum_{j=1}^{n} \frac{1}{n} v_{kj}$$

En otras palabras, si un criterio cumple los principios

1.4.1 Clasificación completa

1.4.4 Dominación fuerte

1.4.5 Independencia de alternativas irrelevantes

1.4.6 Independencia de la adición de una constante a una columna

1.4.7 Independencia de permutaciones en las filas

$\Rightarrow$ *Es el criterio maxisum de Laplace.*

Demostración 1.5.2

En primer lugar observamos que el principio 1.4.1 se utiliza para suponer que el criterio de decision asigna un índice numérico V_i a cada acción a_i. En segundo lugar, suponemos que los $v_{ij} \geq 0 \quad \forall i, j$ puesto que una tabla de decisión podemos transformarla de forma equivalente en una tabla con entradas no negativas simplemente restando el mínimo de la columna a cada componente de dicha columna, como nos dice el principio 1.4.6. En términos generales, la demostración se basa en tomar una tabla de decisión e ir transformándola en una secuencia de tablas equivalentes a la primera hasta que se obtiene una tabla en la que la elección de dos acciones es obvia, es decir, hasta que en dos acciones nos encontramos filas de ceros.

Partimos primero de dos acciones a_i y a_k tal que

$$\sum_{j=1}^{n} \frac{1}{n} v_{ij} = \sum_{j=1}^{n} \frac{1}{n} v_{kj}$$

Es decir, partimos de dos acciones que son consideradas iguales por el criterio de Laplace.

Dejaremos estas acciones fijas pero renombradas como a_i^0 y a_k^0 y lo que haremos será ir añadiendo en cada paso otras acciones.

En el primer paso lo que haremos será, adjuntar una acción a_i^l , para $l = 1$, construida reordenando los elementos $(v_{ij}^{(l-1)})$ de la fila en orden ascendente. De manera análoga, añadimos una acción a_k^l construida reordenando los elementos $(v_{kj}^{(l-1)})$de manera ascendente. Por el principio 1.4.5 sabemos que podemos añadir alternativas sin que esto afecte a la clasificación de las anteriores ($V_i^{(l-1)}$ y $V_k^{(l-1)}$). Por el criterio 1.4.7 $\quad V_i^{(l)} = V_i^{(l-1)}$ y $V_k^{(l)} = V_k^{(l-1)}$. Entonces

$$V_i^{(l)} = V_k^{(l)} \Leftrightarrow V_i^{(l+1)} = V_k^{(l+1)}.$$

En el segundo paso, construimos una nueva tabla de decisión como sigue. Para cada $j = 1, 2, \ldots, n$ elegimos mín $\left\{v_{ij}^l, v_{kj}^l\right\}$ para la $j-$ésima columna, construyendo las acciones a_i^{l+1} y a_k^{l+1}. Por el principio 1.4.6,

$$V_i^{(l)} = V_k^{(l)} \Leftrightarrow V_i^{(l+1)} = V_k^{(l+1)}.$$

Sea ahora $l = l + 2$.

	θ_1	θ_2	θ_3	Proceso
a_i^0	8	6	10	
a_k^0	12	3	9	$(8+6+10)/3 = 8 = (12+3+9)/3$
a_i^1	6	8	10	
a_k^1	3	9	12	Paso 1 Permutar la fila en orden ascendente
a_i^2	3	0	0	
a_k^2	0	1	2	Paso 2 Elegir mín v_{ij}^3, v_{kj}^3
a_i^3	0	0	3	
a_k^3	0	1	2	Paso 1 Permutar la fila en orden ascendente
a_i^4	0	0	1	
a_k^4	0	1	0	Paso 2 Elegir mín v_{ij}^3, v_{kj}^3
a_i^5	0	0	1	
a_k^5	0	0	1	Paso 1 Permutar la fila en orden ascendente
a_i^6	0	0	0	
a_k^6	0	0	0	Paso 2 Elegir mín v_{ij}^5, v_{kj}^5

Tabla 1.10: Secuencia de transformaciones mediante la cuál dos acciones con igual beneficio medio se transforman en dos acciones equivalentes con filas enteras de ceros

Repetimos el proceso hasta que obtenemos una fila de ceros. Puesto que partimos de que las dos acciones eran iguales para el criterio de Laplace y hemos ido restando lo mismo a cada acción, vemos que ambas filas llegarán a ser una fila entera de ceros a la vez. Si consideramos a_i^δ y a_k^δ las dos acciones con una fila entera de ceros, entonces por el principio 1.4.7 $\quad V_i^\delta = V_k^\delta$.

Por lo tanto, la conclusión a la que llegamos

$$\begin{aligned} V_i^\delta = V_k^\delta \quad &\Rightarrow V_i^{\delta-1} = V_k^{\delta-1} \\ &\Rightarrow V_i^{\delta-2} = V_k^{\delta-2} \\ \Rightarrow \cdots \quad &\Rightarrow V_i^0 = V_k^0. \end{aligned}$$

Lo que quiere decir que si las dos acciones del principio son consideradas indiferentes por el criterio de Laplace, entonces cualquier criterio que verifique los principios de racionalidad considerados anteriormente también las considera indiferentes.

Consideremos ahora dos acciones tal que

$$\textstyle\sum_{j=1}^n \frac{1}{n} v_{ij} > \sum_{j=1}^n \frac{1}{n} v_{kj}$$

Es decir, dos acciones que no son indiferentes por el criterio de Laplace. Añadimos una acción a_l a la tabla de manera que

$$v_{lj} = v_{ij} - \textstyle\sum_{j'=1}^n \frac{1}{n}(v_{ij'} - v_{kj'}) \text{ para } j = 1, 2, \ldots$$

Observamos que la cantidad que restamos es constante por lo que

$$\textstyle\sum_{j=1}^n \frac{1}{n} v_{lj} = \sum_{j=1}^n \frac{1}{n} v_{kj} \Rightarrow V_l = V_k.$$

Pero por el principio 1.4.4 $\quad V_i > V_l$, y por el criterio 1.4.5 como añadir a_l no afecta a la clasificación de a_i y a_k, se tiene $V_i > V_k$.

Por lo que la conclusión será que las acciones finales son valoradas de igual forma que el criterio de Laplace. Es decir, si las acciones del principio no eran indiferentes para Laplace, tampoco lo serán para el criterio genérico que consideramos.

□

A partir de la demostración anterior se puede ver el siguiente

Corolario 1.5.1 *No existe ningún criterio de decisión que pueda satisfacer los principios 1.4.1-1.4.8*

Capítulo 2

Aplicaciones del criterio minimax regret

Como mencionábamos al principio del primer capítulo, la programación estocástica puede no ser capaz de satisfacer las necesidades de la toma de decisiones en entornos de decisión caracterizados por una incertidumbre significativa, ya que exige asignar distribuciones de probabilidad a los escenarios considerados y esto podría estar lejos de ser un ejercicio trivial para muchos decisores. En muchos casos puede ser difícil calcular la probabilidad de que ocurra un escenario futuro cuando los factores que intervienen se refieren al comportamiento de empresas, agencias o gobiernos de los que no disponemos de información. En otros casos puede que haya que considerar cambios en las actitudes y prioridades de los clientes potenciales relacionados con el problema de decisión. Además, en muchas ocasiones, los decisores están más interesados en protegerse del riesgo que supone tomar decisiones por las que deberán rendir cuentas en el futuro que en optimizar la eficacia esperada en todos los escenarios o simplemente en aquellos más probables. El decisor no solo se preocupa de cómo el beneficio varía en cada uno de los escenarios,

sino también de cómo el beneficio real bajo la decisión tomada se compara con el beneficio óptimo que podríamos haber logrado si hubiésemos conocido de antemano el escenario real. En el enfoque que se sigue en este trabajo, a diferencia del que se utiliza en programación estocástica, tiene como objetivo el de proteger al decisor sin necesidad de utilizar un modelo de probabilidad sobre el comportamiento del sistema.
En el marco de la robustez, lo que el decisor quiere no es el óptimo a largo plazo ni el óptimo para un único escenario (aunque sea el escenario más probable), sino una decisión que funcione bien en todos los casos. Por lo que una decisión robusta será aquella que caracterizada por una incertidumbre significativa, funciona bien en todos los casos y se protege contra el peor de los posibles escenarios.

Dentro del marco de la robustez, y basándonos en [14] nos vamos a centrar en los tres criterios de decisión siguientes:

- *Criterio maximin.*
- *Criterio minimax regret.*
- *Criterio minimax regret relativo.* En él se compara la eficacia relativa de cada decisión frente a la óptima en cada uno de los escenarios posibles. Este criterio podemos definirlo a partir del criterio minimax regret de Savage como sigue

  Definimos el regret relativo como

$$r'_{ij} = \frac{\max\limits_{l=1,\dots,m}\{v_{lj}\} - v_{ij}}{\max\limits_{l=1,\dots,m}\{v_{lj}\}} = \frac{r_{ij}}{\max\limits_{l=1,\dots,m}\{v_{lj}\}}$$

  Por lo que el peor regret relativo resultante bajo la acción x^i será

$$\rho_i^{'} = \max_{j=1,\ldots,n} \left\{ r_{ij}^{'} \right\}$$

y entonces deberíamos elegir una acción que minimiza $\rho_i^{'}$.

Elegir x^k tal que

$$\rho_k^{'} = \min_{i=1,\ldots,m} \left\{ \rho_i^{'} \right\} = \min_{i=1,\ldots,m} \left\{ \max_{i=1,\ldots,n} \frac{r_{ij}}{\max_{l=1,\ldots,m} \{v_{lj}\}} \right\}$$

Remarcamos que, igual que ocurría con los criterios del capítulo anterior, la elección de distintos criterios para la misma situación pueden dar lugar a decisiones óptimas distintas.

Además, tendremos en cuenta que, si los datos de entrada del problema se refieren a costes en lugar de a recompensas, la definición del $r_{ij}^{'}$ sería

$$r_{ij}^{'} = \frac{c_{ij} - \min_{l=1,\ldots,n} \{c_{lj}\}}{\min_{l=1,\ldots,n} \{c_{lj}\}}$$

De acuerdo con estas tres definiciones, para encontrar soluciones robustas nos centraremos en resolver los siguientes problemas de optimización:

Para el criterio maximin

$$z_M = \min \{y | f(X, \theta^s) \leq y, s \in S; x \in X = \cap_{s \in S} F_s\},$$

siendo F_s el conjunto de soluciones factibles bajo el escenario $s \in S$.

Para el criterio minimax regret

$$z_R = \min \{y | f(X, \theta^s) \leq y + z^s, s \in S; x \in X\}.$$

Para el criterio minimax regret relativo

$$z_{RL} = \min \{y | f(X, \theta^s) \leq (y+1) z^s, s \in S; x \in X\}.$$

Tenemos en cuenta en todas las definiciones que la intersección de los conjuntos factibles para s sea no vacía, $\cap_{s\in S} F_s \neq \emptyset$. Nos referimos a éstos como problemas de optimización robusta discreta.

El principal objetivo que planteamos en esta sección es discutir la formulación de un problema de optimización cuya solución conduzca a la identificación de decisiones robustas.

2.1. Modelo: Programación lineal robusta.

Un problema de programación lineal para un escenario específico s dado, se escribe de la forma

$$\begin{aligned} \text{mín} \quad & c^s x \\ \text{s.t} \quad & \\ & A^s x = b^s, \\ & x \geq 0. \end{aligned}$$

El problema minimax regret relativo de programación lineal robusta sería como sigue

$$\begin{aligned} \text{mín} \quad & y \\ \text{s.t} \quad & \\ & c^s x \leq (y+1) z^s, \quad s \in S, \\ & A^s x = b^s, \quad s \in S, \\ & x \geq 0. \end{aligned}$$

Que bien podría escribirse de la siguiente forma

$$\begin{aligned} &\text{mín máx} \quad \left\{ \frac{c^s x - z^s}{z^s} : s \in S \right\} \\ &\text{s.t} \\ &\qquad A^s x = b^s, \quad s \in S, \\ &\qquad x \geq 0. \end{aligned}$$

Para el problema minimax regret de programación lineal es necesario reemplazar el término de la derecha por $y + z^s$.

$$\begin{aligned} &\text{mín máx} \quad \{c^s x - z^s : s \in S\} \\ &\text{s.t} \\ &\qquad A^s x = b^s, \quad s \in S, \\ &\qquad x \geq 0. \end{aligned}$$

Y el problema maximin de programación lineal robusta tendría una formulación análoga a las anteriores quedando como sigue

$$\begin{aligned} &\text{máx mín} \quad \{-c^s x : s \in S\} \\ &\text{s.t} \\ &\qquad A^s x = b^s, \quad s \in S, \\ &\qquad x \geq 0. \end{aligned}$$

La programación lineal tiene una amplia variedad de aplicaciones económicas, como podemos ver en los libros de [10], [2] y [19], en muchos de ellos teniendo en cuenta la utilización de datos para los que existe incertidumbre en situaciones de planificación a medio y largo plazo.

Para mostrar cómo puede verse reflejada esta incertidumbre proponemos el siguiente problema de mezcla de productos. Una empresa fabrica n productos finales diferentes, que podrían ser variaciones del mismo producto, y utiliza m materias primas básicas para su fabricación. Generalmente, las cantidades a_{ij}^s indican la cantidad de materia prima básica i utilizada para la producción de una unidad del producto j, y las cantidades b_i^s indican la cantidad máxima de materia prima disponible. En otros casos, a_{ij}^s podría representar coeficientes de productividad (las horas de trabajo necesarias por unidad de producto final), y b_i^s podría ser horas de trabajo disponibles de un tipo específico de trabajo (cualificado, no cualificado o subcontratado). Pues bien, la empresa está interesada en maximizar su beneficio neto (el precio de venta menos la mano de obra y otros costos de producción) y, por lo tanto, las cantidades $-c_j^s$ representan el beneficio neto para un escenario de datos de entrada s específico.

La incertidumbre de los datos de entrada puede manifestarse de varias formas

- Incertidumbre en los coeficientes de beneficio neto. Podría reflejar las diferentes condiciones del mercado que pueden afectar al precio (por ejemplo, las posibles acciones de los competidores o la tasa de cambio y la incertidumbre inflacionaria si operan en un mercado extranjero), los costes de mano de obra (por ejemplo, la oferta limitada de mano de obra y / o huelgas laborales) y otros costes de producción (problemas inesperados de equipo tales como averías y problemas de calidad o rendimiento).

- Uso de tecnología de producción imprecisa. Esto influye en los coeficientes dependientes de la tecnología a_{ij} (usamos la interpretación de a_{ij} como coeficientes de productividad). Para una nueva tecnología es difícil estimar el impacto o los efectos de aprendizaje sobre la productividad. Esto induce una alta incertidumbre en los valores de a_{ij}.

- Incertidumbre de la oferta de materias primas. Esta incertidumbre podría ser causada por las fuerzas del mercado (industrias competidoras por la misma materia prima podrían limitar el suministro disponible) u otras razones sencillas tales como el clima o las restricciones gubernamentales a las importaciones.

2.2. Modelo: Problema robusto de asignación.

El problema de asignación es un problema de programación lineal entera con una estructura especial. Este problema tiene una amplia variedad de aplicaciones y se define como sigue

$$
\begin{aligned}
&\text{mín} \quad \sum_{i,j} c_{ij}^{s} x_{ij} \\
&\text{s.t} \\
&\qquad \textstyle\sum_i x_{ij} = 1, \quad j = 1, \ldots, n, \\
&\qquad \textstyle\sum_j x_{ij} = 1, \quad i = 1, \ldots, n, \\
&\qquad x_{ij} \geq 0, \qquad i, j = 1, \ldots, n.
\end{aligned}
$$

Observamos que la matriz de restricciones es totalmente unimodular por lo que no necesitamos exigir que las variables $x_{ij} \in \{0, 1\}$ sino que es suficiente con que sean no negativas. Además, no nos hace falta exigir que $x_{ij} \leq 1$ puesto que tal como aparecen las restricciones no nos vamos a encontrar

ningún x_{ij} que sea mayor o igual que 1.

El problema minimax regret relativo de asignación puede formularse de la forma

$$
\begin{aligned}
\text{mín} \quad & y \\
\text{s.t} \quad & \\
& \sum_{i,j} c_{ij}^s x_{ij} \leq (y+1) z^s, && s \in S, \\
& \textstyle\sum_i x_{ij} = 1, && j = 1, \ldots, n, \\
& \textstyle\sum_j x_{ij} = 1, && i = 1, \ldots, n, \\
& x_{ij} \in \{0, 1\}, && i, j = 1, \ldots, n.
\end{aligned}
$$

De manera análoga podemos encontrar la formulación maximin, y mediante los cambios apropiados en el miembro de la derecha de la primera restricción, al igual que ocurría en el modelo anterior, podemos obtener la formulación de minimax regret.

Para el problema de asignación, cada recurso (como sería un empleado o una máquina), debe ser asignado exclusivamente a una única actividad o tarea particular. En este caso, hay un coste c_{ij} asociado con la asignación i que realiza la tarea j. La incertidumbre de datos de entrada en este caso podría ser el nivel de eficiencia del empleado o máquina en cualquier día en particular de acuerdo con una planificación. Por ejemplo, en el caso de una máquina, el nivel de eficiencia de la misma se ve afectado por muchos factores desconocidos tales como las condiciones meteorológicas o la presencia de dispositivos auxiliares específicos (como accesorios, paletas y aceites lubricantes) que potencian la eficiencia de la máquina. Lo que intenta la una

decisión robusta es asignar los recursos a las tareas de manera que se proteja la actividad que se realiza contra la peor situación posible.

2.3. Modelo: Problema robusto de caminos más cortos.

Dado un grafo $G = (V, E)$, con longitud no negativa c_e asociada a cada eje $e \in E$, un nodo origen $v_0 \in V$ y un nodo destino $v_t \in V$, se define el problema de caminos más cortos como encontrar el camino de mínima longitud total para ir de v_0 a v_t. Este problema puede resolverse en tiempo polinomial con una complejidad $\mathcal{O}(|V|^2)$, mediante el algoritmo de Dijkstra en [3], aunque existen implementaciones que rebajan esta complejidad. El problema maximin de camino más corto puede definirse como encontrar el camino que tiene longitud máxima mínima en todos los escenarios. Para este tipo de problema, cada escenario corresponde a un conjunto predeterminado de longitudes de ejes. El problema minimax regret relativo (y de manera similar minimax regret) de caminos más cortos se define como encontrar entre todos los trayectos que van de v_0 a v_t el que minimiza la diferencia relativa máxima (diferencia máxima) de la longitud del trayecto frente a la longitud de trayecto óptima en cada escenario.

Una aplicación obvia de este modelo es la de obtención de rutas para el transporte de mercancías usando una red de carreteras. En la incertidumbre de los datos está reflejada la incertidumbre en las condiciones de tráfico en las distintas carreteras (como es la presencia de accidentes, atascos de tráfico en las horas punta o proyectos de construcción). Todos los escenarios anteriores podrían no ser apropiados para un día en particular, debido a la falta de información actualizada en todas las carreteras tendrá que considerar un subconjunto apropiado de escenarios, que podría afectar a todo o sólo a un

subconjunto de carreteras (longitudes de los ejes), el decisor al hacer esta elección de ruta apropiada lo hace de manera robusta.

Existen otras aplicaciones como la detección de tareas críticas en la ejecución de un proyecto complejo. En este caso, se utilizaría una formulación derivada del denominado *Critical Path Method*.

2.4. Modelo: Problema robusto de árbol de unión.

Dado un grafo conexo no dirigido de la forma $G = (V, E)$, un árbol de unión T es definido como un subgrafo conexo de G sin ciclos. En otras palabras, un árbol de unión es un subgrafo conexo minimal de G. Si a cada eje del grafo se le asocia un coste, entonces un árbol de unión mínimo se refiere a aquel árbol de unión cuya suma de los costes de los ejes es mínima. Un árbol de unión mínimo puede encontrarse fácilmente mediante el algoritmo de Prim con una complejidad de $\mathcal{O}(\text{mín}\left\{|V|^2, |E|\log|V|\right\})$ o mediante el algoritmo de Kruskal con una complejidad de $\mathcal{O}(|E|\log|E|)$. Para definir la versión maximin del mismo asociamos un coste no negativo a cada eje $e \in E$ bajo cada escenario $s \in S$. Por lo tanto, el problema maximin del mínimo árbol de unión se define como

$$\begin{aligned} &\max_{T} \quad \min_{s\in S} \sum_{e\in E} -c_e^s \\ &\text{s.t} \\ &\qquad T\text{es árbol de unión.} \end{aligned}$$

El problema de mínimo árbol de unión tiene aplicaciones principalmente en problemas de diseño de redes de transporte (como podemos ver en [15]) y

diseño de redes de comunicaciones ([7]), en el que por ejemplo una compañía desea abarcar una serie de localizaciones con el menor coste posible. Al igual que ocurría en el problema de caminos más cortos, la principal incertidumbre de datos de entrada está recogida en el coste de transporte de las diversas carreteras o líneas de comunicaciones.

2.5. Modelo: Problema robusto de mochila.

El problema de mochila para un escenario específico s es definido como sigue.

$$\begin{array}{ll} \text{máx} & \sum_{i=1}^{n} v_i^s x_i \\ \text{s.t} & \\ & \sum_{i=1}^{n} a_i x_i \leq b, \\ & x_i \in \{0,1\}, \qquad i = 1, \ldots, n. \end{array}$$

Es conocido que éste es un problema NP-completo que puede resolverse en tiempo pseudo-polinomial con una complejidad de $\mathcal{O}(nb)$ usando algoritmos de programación dinámica (véase [24]).

El problema maximin de mochila se define como sigue.

$$\begin{array}{ll} \text{máx} & y \\ \text{s.t} & \\ & \sum_{i=1}^{n} v_i^s x_i \geq y, \quad s \in S, \\ & \sum_{i=1}^{n} a_i x_i \leq b, \\ & x_i \in \{0,1\}, \qquad i = 1, \ldots, n. \end{array}$$

El problema de mochila modela la idea de llenar una mochila incapaz de soportar más de un peso determinado, con todo o parte de un conjunto de objetos, cada uno con un peso y valor específicos. Los objetos colocados en la mochila deben maximizar el valor total sin exceder el peso máximo. Este problema tiene una gran cantidad de aplicaciones como por ejemplo en el la gestión de presupuestos ([10]). Veamos por tanto, un ejemplo de esta aplicación. Si una empresa quiere seleccionar n proyectos, sea v_i^s el valor actual neto del producto i, a_i la inversión requerida ($a_i > 0$) y b ($b > 0$) el presupuesto disponible para invertir en nuevos proyectos. La incertidumbre en los coeficientes de la función objetivo podría ser causada por flujos de efectivo futuros y / o tasas de descuento de los diversos flujos de efectivo futuros. La incertidumbre del flujo podría ser el resultado de las acciones de los competidores, el crecimiento del mercado, y las condiciones macroeconómicas (como las condiciones inflacionarias o la presencia de una recesión que afecta al comportamiento de compra de los clientes). Las tasas de descuento podrían verse afectadas por una variedad de factores tales como oportunidades de inversión futuras potenciales y tasas de interés de las instituciones financieras.

Una discusión detallada sobre el problema de mochila podemos encontrarlo en [12].

2.6. Modelo: Problema robusto de asignación de recursos.

El problema de asignación de recursos se define como sigue: N unidades de un recurso dado se asignan a n actividades. La realización de cada actividad tiene un coste, que depende de la cantidad de recurso asignado para dicha actividad. Lo que buscamos es la asignación óptima de recursos tal que minimice el coste total. Sea por tanto x_i la cantidad de recurso asignado

a la actividad i. Sea $c_i(x_i)$ el coste producido por la actividad i cuando x_i unidades de recurso son asignadas a la actividad i. El problema de asignación de recursos para un escenario específico s puede formularse como el siguiente problema entero no lineal como sigue.

$$\begin{aligned}
\text{mín} \quad & \sum_{i=1}^{n} c_i^s(x_i) \\
\text{s.t} \quad & \\
& \sum_{i=1}^{n} x_i \leq N, \\
& x_i \in \mathbb{Z}_+ \qquad i = 1, \ldots, n.
\end{aligned}$$

Bajo determinadas condiciones, para un escenario específico s, podemos resolver este problema en tiempo polinomial mediante un simple algoritmo greedy, que es aquel que intenta hacer lo mejor posible para cada actividad sin tener en cuenta los recursos disponibles para el resto de actividades.
El problema maximin de asignación de recursos se define de la siguiente forma

$$\begin{aligned}
\text{máx} \quad & y \\
\text{s.t} \quad & \\
& \sum_{i=1}^{n} -c_i^s(x_i) \geq y, \\
& \sum_{i=1}^{n} x_i \leq N, \\
& x_i \in \mathbb{Z}_+ \qquad i = 1, \ldots, n.
\end{aligned}$$

El problema puede ser fácilmente motivado dentro el contexto de planificar la organización de varios artículos dentro de un almacén con espacio de

almacenamiento limitado. Se debe tener en cuenta que se tiene que satisfacer la demanda prevista en la planificación, los costes de economía de escala y el coste de almacenamiento de inventario. Supongamos que los artículos se ordenan en contenedores estandarizados que requieren una unidad de espacio de almacenamiento, y sea D_i la demanda del ítem i en término del número de contenedores. Tendremos además, un coste de pedido O_i por pedido del ítem i y un coste h_i de mantenimiento por contenedor por periodo del ítem i. Entonces, el problema de elegir el tamaño de pedido x_i del ítem i puede formularse como un problema discreto de asignación de recursos con $c_i^s(x_i) = \frac{D_i^s O_i^s}{x_i} + \frac{1}{2} h_i^s x_i$, con N la capacidad de almacenamiento total.

Para el entorno de planificación anterior, la incertidumbre de datos de entrada puede asumir muchas formas:

- Incertidumbre en la demanda de los distintos ítems, que puede ser causada por una variedad de razones, como pueden ser las acciones de los competidores que afectan a los precios y a la demanda, o incluso cambios en las preferencias de los consumidores debido a la introducción posterior de productos sustitutivos.

- La incertidumbre en los costes de pedido, lo que podría reflejar la flexibilidad de la empresa en la selección entre proveedores en un entorno competitivo de precios competitivos, o los costes de pedido adicionales asociados con la calidad de los pedidos entrantes (como la inspección de artículos de mala calidad).

- Incertidumbre en los costes de almacenamiento. Este coste puede verse afectado por las tasas de interés bancarias, e incluso por situaciones de escasez de capital, con dificultades asociadas para financiar en en-

tornos de países en desarrollo en los que podrían estar localizadas las instalaciones.

2.7. Modelo: Problema robusto de secuenciación.

Consideremos el conjunto $\{1, 2, \ldots, n\}$ de n trabajos independientes que requieren procesamiento en un conjunto $\{1, 2, \ldots, m\}$ de m máquinas. El interés de este problema es planificar situaciones en las que los tiempos de procesamiento de cada operación individual de trabajo son imprecisos, entendiendo tiempo de procesamiento como el tiempo que se requiere para cada operación. Esta incertidumbre en el tiempo de procesamiento será descrita a través de un conjunto de escenarios S de tiempos de procesamiento. Sea p_{ij}^s el tiempo de procesamiento del trabajo i en la máquina j bajo el escenario s, y $P^s = \left\{p_{ij}^s : i = 1, \ldots, m, j = 1, \ldots, n\right\}$ la matriz de tiempos de procesamiento de trabajo correspondiente al escenario s. Para ver ejemplos de formulación robusta de secuenciación, nos centraremos en principio en la programación de una sola máquina aplicando el criterio de tiempo medio de completación, después veremos el caso en que el problema de secuenciación requiere la programación de dos máquinas. Si por ejemplo estuviésemos hablando de una sucursal bancaria, estamos suponiendo que todos los clientes entran a la vez y van saliendo según van siendo atendidos, por lo que el tiempo medio de procesamiento es lo que tardan en media en atender a cada cliente, y el tiempo medio de completación es el tiempo medio que tarda el cliente desde que llega al banco hasta que éste sale atendido.

Formulemos el modelo minimax regret relativo para el problema de secuenciación con una sola máquina con el criterio de tiempo medio de completación. Sea $\sigma = \{\sigma(1), \sigma(2), \ldots, \sigma(n)\}$ una permutación que describe el

orden de procesamiento de los n trabajos, siendo $\sigma(k)$ la posición en la que es procesado el trabajo k en la secuencia σ. Definimos las variables $x_{ik} \in \{0,1\}$ como sigue

$$x_{ik} = \begin{cases} 1 \quad si \;\; \sigma(k) = i \qquad i = 1,\ldots,n; k = 1,\ldots,n. \\ \\ 0 \quad \text{en otro caso} \end{cases}$$

Es decir, x_{ik} vale 1 si el trabajo i se procesa en la $k-$ésima posición. Entonces, nuestro problema minimax regret relativo de secuenciación puede ser escrito como

$$\begin{array}{lll} \text{mín} & y & \\ \text{s.t} & & \\ & \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{n}(n-k+1)p_i^s x_{ik} \;\leq\; \frac{1}{n}(y+1)z^s, & s \in S, \\ & \sum_{k=1}^{n} x_{ik} = 1 & i = 1,\ldots,n, \\ & \sum_{i=1}^{n} x_{ik} = 1 & k = 1,\ldots,n, \\ & x_{ik} \in \{0,1\} & i,k = 1,\ldots,n. \end{array}$$

De esta forma, el lado izquierdo de la primera restricción representa el tiempo promedio que se tarda en atender a un cliente, y lo que se pretende es minimizar el regret relativo respecto del tiempo de completación promedio óptimo bajo cada escenario posible.

Al igual que en otros modelos, para la versión de minimax regret del problema el lado derecho de la primera restricción debe ser reemplazado por $y+z^s$, mientras que para el maximin, es decir, minimax en términos de coste lo reemplazaríamos por y.

Veamos ahora la formulación de un problema de secuenciación con más de una máquina. El problema de secuenciación de dos máquinas se ocupa de la programación de una instalación de producción en dos etapas (lo que podrían ser dos máquinas), con procesamiento fijo desde la etapa 1 hasta la etapa 2, y con espacio de almacenamiento infinito entre ambas etapas. Considerando las mismas variables indicadoras x_{ik} que antes, y las variables

B_k^s = tiempo de inicio del $k-$ésimo trabajo en la secuencia de la segunda máquina (donde el superíndice s se refiere al escenario de los tiempos de procesamiento).

Lo que quiere decir que B_k^s será el máximo entre el tiempo que se tarda en procesar el $k-$ésimo trabajo en la máquina 1 y el tiempo que tarda en quedarse libre la máquina 2 para el el $k-$ésimo trabajo. Pues bien, en estas condiciones podemos formular el problema de secuenciación para el criterio de tiempo de completación con el objetivo de fabricar en dos máquinas para un escenario de tiempos de procesamiento s como sigue

$$
\begin{array}{lllll}
\text{mín} & \sum_{i=1}^{n} p_{i2}^s x_{in} + B_n^s & & & \\
\text{s.t} & & & & \\
& \sum_{i=1}^{n} \sum_{j=1}^{k} p_{i1}^s x_{ij} & \leq & B_k^s, & k = 1, \ldots, n; s \in S, \\
& B_k^s + \sum_{i=1}^{n} p_{i2}^s x_{ik} & \leq & B_{k+1}^s & k = 1, \ldots, n-1; s \in S, \\
& \sum_{i=1}^{n} x_{ik} = 1 & & & k = 1, \ldots, n, \\
& \sum_{k=1}^{n} x_{ik} = 1 & & & i = 1, \ldots, n, \\
& x_{ik} \in \{0, 1\} & & & i, k = 1, \ldots, n.
\end{array}
$$

Si nos fijamos en la primera restricción, el miembro de la izquierda representa el tiempo total de procesamiento de los k primeros trabajos en la máquina 1. Con respecto a la segunda restricción, el miembro de la izquierda representa el tiempo en el que puede comenzar el procesamiento del $k-$ésimo trabajo en la máquina 2 más lo que tarda ese trabajo en ser procesado por la máquina 2. Lo que estaremos calculando será el mínimo tiempo total que se tarda en completar los n trabajos en las 2 máquinas. Así pues, el minimax regret en el problema de secuenciación para el criterio de tiempo de completación se define como

$$
\begin{array}{lll}
\text{mín} & y & \\
\text{s.t} & & \\
& \sum_{i=1}^{n} p_{i2}^{s} x_{in} + B_n^s \leq y + z^s, & s \in S, \\
& \sum_{i=1}^{n} \sum_{j=1}^{k} p_{i1}^{s} x_{ij} \leq B_k^s, & k = 1, \ldots, n; s \in S, \\
& B_k^s + \sum_{i=1}^{n} p_{i2}^{s} x_{ik} \leq B_{k+1}^s & k = 1, \ldots, n-1; s \in S, \\
& \sum_{i=1}^{n} x_{ik} = 1 & k = 1, \ldots, n, \\
& \sum_{k=1}^{n} x_{ik} = 1 & i = 1, \ldots, n, \\
& x_{ik} \in \{0, 1\} & i, k = 1, \ldots, n.
\end{array}
$$

La cantidad z^s representa el tiempo de completación óptimo del problema en cada escenario. Por lo tanto, lo que estamos minimizando es el mínimo del máximo regret de los tiempos de completación. Para obtener el minimax regret relativo para el problema de secuenciación, igualmente necesitamos reemplazar el lado derecho de la primera restricción por y $(y+1)z^s$, y de manera análoga podemos encontrar una formulación para el criterio maximin.

2.8. Modelo: Problema robusto de diseño de redes.

Para estudiar el problema de diseño de redes nos centraremos en entornos caracterizados por una incertidumbre significativa en los datos de entrada. El problema de diseño de redes intenta resolver la siguiente cuestión: ¿qué configuración de red minimiza la suma de los costes fijos de los arcos elegidos en dicha red y el coste de los enrutamientos de bienes a través de la red definida por estos arcos? Este tipo de problema tiene aplicaciones tan diversas como la toma de decisiones de inversión de capital para la planificación del transporte, la planificación de la ruta y la flota de vehículos, diseño de redes de telecomunicaciones, localización de instalaciones y diseño del sistema de distribución de la carga. Desde un punto de vista teórico, la versión más simple del problema de diseño de red sin capacidad de flujo en los arcos con costes de enrutamiento lineal, modela muchos de los problemas más conocidos en la optimización combinatoria (por ejemplo, caminos más cortos, mínimo árbol de unión, ramificación óptima, problema del viajante y problema de la red Steiner). Para un estudio más exhaustivo de problemas de diseño de red y sus aplicaciones, así como una cobertura de los casos especiales de modelos de diseño de redes sin capacidad de flujo, véase [18].

Los elementos básicos del modelo son un conjunto N de nodos y un conjunto A de arcos no dirigidos disponibles para diseñar la red. Denotamos por $\{i, j\}$ un arco no dirigido entre los nodos i y j, y por (i, j) y (j, i) sus correspondientes direcciones. Sea K el conjunto de mercancías para cada $k \in K$, d_k denota la cantidad requerida de flujo de la mercancía k que debe ser enviada desde su punto de origen, denotado por $O(k)$, hasta su punto de destino, denotado por $D(k)$. Dado c_{ij}^k denota el coste no negativo de ruta de la mercancía k en el arco (i, j) y F_{ij} es el coste fijo de construcción del

arco $\{i, j\}$. En general, c_{ij}^k y c_{ji}^k no tienen por qué ser iguales, por ejemplo si hablamos de una ruta de carreteras, ir de i a j podría ser un camino más largo o con más cuestas que el camino de ir de j a i, por lo que no tendrían el mismo coste de combustible.

Los parámetros de entrada más importantes para el problema son los coeficientes de coste de ruta c_{ij}^k y los volúmenes de los diversos productos a transportar d_k. En el modelo que consideramos, no tenemos limitada la capacidad de flujo sobre los arcos, por tanto podemos simplificar el problema asumiendo que todo el producto se transporta utilizando la misma ruta desde $O(k)$ hasta $D(k)$. Esto equivale a ajustar los costes que ahora no serán unitarios sino que se refieren al transporte de la cantidad total de cada producto.

Igual que hemos hecho en otros modelos, utilizaremos la notación $s \in S$ como el índice de escenario de datos de entrada y S el conjunto de todos los escenarios posibles. Diferentes escenarios de datos de entrada S implican diferentes costes de tomar el arco (i, j) para transportar todo el flujo de mercancía k desde el nodo i al nodo j, es decir, $c_{ij}^k(s)$. Sea $C^k(s) = \left\{c_{ij}^k(s), i, j \in N\right\}$ y $C(s) = \left\{C^k(s), k \in K\right\}$.

Para datos de entrada específicos, (es decir, $C(s)$, $s \in S$), el problema de diseño de red contiene dos tipos de variables de decisión, uno que modela decisiones discretas y otro que modela decisiones de flujo continuo. Sea y_{ij} una variable binaria que indica si se ha elegido o no el arco $\{i, j\}$, es decir

$$y_{ij} = \begin{cases} 1 & \text{si el arco } \{i, j\} \text{ es elegido para el diseño de red} \\ 0 & \text{en caso contrario} \end{cases}$$

Sea x_{ij}^k el flujo de la mercancía k en el arco dirigido (i,j). Entonces, si $y = (y_{ij})$ y $x = (x_{ij}^k)$ son las matrices de las variables de diseño y flujo, para un escenario específico $s \in S$, el problema de diseño de redes se formula como sigue:

$$\text{mín} \qquad \sum_{k \in K} \sum_{(i,j) \in A} (c_{ij}^k(s) x_{ij}^k + c_{ji}^k(s) x_{ji}^k) + \sum_{\{i,j\} \in A} F_{ij} y_{ij}$$

s.t

$$\sum_{i \in N} x_{ij}^k - \sum_{h \in N} x_{jh}^k = \left\{ \begin{array}{ll} -1 & \text{si } j = O(k); j \in N \\ 1 & \text{si } j = D(k); k \in K \\ 0 & \text{en otro caso.} \end{array} \right\}; \quad j \in N, k \in K,$$

$$x_{ij}^k \leq y_{ij}, x_{ji}^k \leq y_{ij}, \{i,j\} \in A, k \in K,$$

$$x_{ij}^k, x_{ji}^k \geq 0; y_{ij} \in \{0,1\}, \{i,j\} \in A, k \in K,$$

$$y \in \Gamma.$$

El primer bloque de restricciones para cada producto k representa las limitaciones habituales de conservación de flujo de red. El segundo bloque de restricciones rechaza cualquier flujo de mercancías a través de arco $\{i,j\}$ si este arco no está incluido en el diseño, es decir, $y_{ij} = 0$. El conjunto Γ incluye cualquier restricción lateral impuesta individual o conjuntamente en las variables de flujo y de diseño (por ejemplo, restricciones de elección

múltiple ($y_{ij} + y_{pq} \leq 1$) o restricciones de prioridad ($y_{ij} \leq y_{rs}$)).

Las decisiones en el diseño de redes tienen un impacto determinante en la eficacia del sistema diseñado para un largo periodo de tiempo. En muchas instancias, debido a los cortos ciclos de vida de los productos, los diseños de redes de fabricación deben desarrollarse sin ni siquiera saber el conjunto exacto de productos a fabricar. Esta incertidumbre de datos de entrada impulsa al diseñador a buscar diseños que sean "buenos" (es decir, cercanos al óptimo) para una variedad de escenarios operativos futuros. Esto se denomina "propiedad de robustez" de los diseños de redes.

Dado que estamos hablando de costes de enrutamiento, el criterio minimax para el diseño de redes se formula como sigue:

$$\min_{x,y} \max_{s \in S} \quad (\sum_{k \in K} \sum_{\{i,j\} \in A} (c_{ij}^k(s) x_{ij}^k + c_{ji}^k(s) x_{ji}^k) + \sum_{\{i,j\} \in A} F_{ij} y_{ij}) - z^s$$

s.t

$$\sum_{i \in N} x_{ij}^k - \sum_{h \in N} x_{jh}^k = \left\{ \begin{array}{ll} -1 & \text{si } j = O(k); j \in N \\ 1 & \text{si } j = D(k); k \in K \\ 0 & \text{en otro caso.} \end{array} \right\}; \quad j \in N, k \in K,$$

$$x_{ij}^k \leq y_{ij}, x_{ji}^k \leq y_{ij}, \{i,j\} \in A, k \in K,$$

$$x_{ij}^k, x_{ji}^k \geq 0; y_{ij} \in \{0,1\}, \{i,j\} \in A, k \in K,$$

$$y \in \Gamma.$$

Donde z^s representa el coste de diseño óptimo bajo el escenario $s \in S$.

2.9. Ejemplo numérico

Vamos a resolver el problema del diseño robusto de redes sobre el grafo de conexiones potenciales representado en la figura 2.1 Los coeficientes que

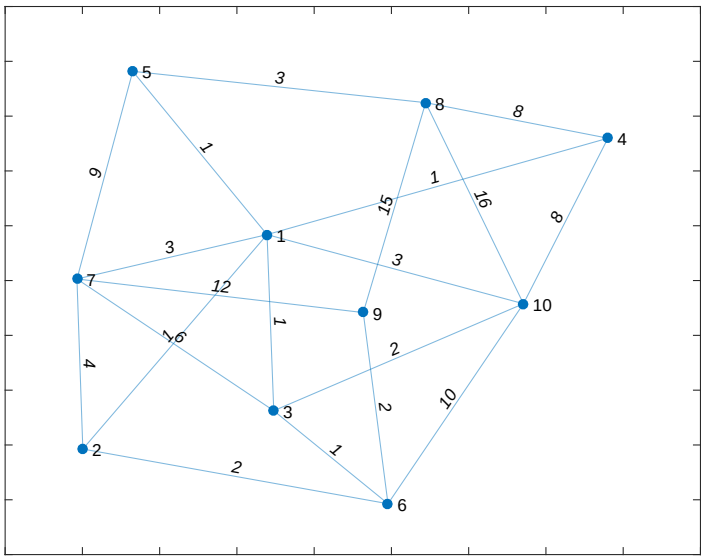


Figura 2.1: Red de conexiones potenciales con los costes fijos F_{ij}

aparecen asociados a los ejes corresponden con los costes fijos de construcción F_{ij}. Suponemos que a lo largo de cada eje se puede transportar producto en ambos sentidos. El software utilizado para resolver la formulación determinística del problema bajo cada escenario es AMPL. A continuación se muestra el código correspondiente a la resolución del problema bajo el primer escenario de costes de transporte

```
set NODOS:={1..10};
set ARCOS within {NODOS,NODOS}:={
```

```
(1,2), (1,3), (1,4), (1,5), (1,7), (2,6), (2,7), (5,7),
(7,9), (6,9), (3,6), (3,10), (1,10), (4,10), (3,7), (8,10),
(4,8), (5,8), (6,10), (8,9),
(2,1), (3,1), (4,1), (5,1), (7,1), (6,2), (7,2), (7,5),
(9,7), (9,6), (6,3), (10,3), (10,1), (10,4), (7,3), (10,8),
(8,4), (8,5), (10,6), (9,8)};

set PRODUCTOS:={1..3};

param c{ARCOS,PRODUCTOS};
param F{ARCOS};
param B{NODOS,PRODUCTOS};

var x{ARCOS,PRODUCTOS}>=0;

var y{ARCOS} binary;

minimize objetivo:sum{(i,j) in ARCOS,k in PRODUCTOS} c[i,j,k]*x[i,j,k]+
                  sum{(i,j) in ARCOS} F[i,j]*y[i,j];
subject to flow{j in NODOS, k in PRODUCTOS}:
sum{(i,j) in ARCOS}x[i,j,k]-sum{(j,h) in ARCOS}x[j,h,k]=B[j,k];
subject to bound{(i,j) in ARCOS,k in PRODUCTOS}:
x[i,j,k]<=y[i,j]+y[j,i];

data;
param c:=
[1,2,1]  1 [1,3,1]  2 [1,4,1]  3 [1,5,1]  3
```

```
[1,7,1]  5 [2,6,1]  6 [2,7,1]  7 [5,7,1]  8
[7,9,1]  1 [6,9,1]  2 [3,6,1]  3 [3,10,1] 3
[1,10,1] 5 [4,10,1] 6 [3,7,1]  7 [8,10,1] 8
[4,8,1]  1 [5,8,1]  2 [6,10,1] 3 [8,9,1]  8
[2,1,1]  1 [3,1,1]  2 [4,1,1]  3 [5,1,1]  3
[7,1,1]  5 [6,2,1]  6 [7,2,1]  7 [7,5,1]  8
[9,7,1]  1 [9,6,1]  2 [6,3,1]  3 [10,3,1] 3
[10,1,1] 5 [10,4,1] 6 [7,3,1]  7 [10,8,1] 8
[8,4,1]  1 [8,5,1]  2 [10,6,1] 3 [9,8,1]  8
[1,2,2]  5 [1,3,2]  2 [1,4,2]  1 [1,5,2]  3
[1,7,2]  2 [2,6,2]  6 [2,7,2]  1 [5,7,2]  8
[7,9,2]  3 [6,9,2]  2 [3,6,2]  2 [3,10,2] 3
[1,10,2] 5 [4,10,2] 3 [3,7,2]  7 [8,10,2] 8
[4,8,2]  2 [5,8,2]  2 [6,10,2] 3 [8,9,2]  1
[2,1,2]  5 [3,1,2]  2 [4,1,2]  1 [5,1,2]  3
[7,1,2]  2 [6,2,2]  6 [7,2,2]  1 [7,5,2]  8
[9,7,2]  3 [9,6,2]  2 [6,3,2]  2 [10,3,2] 3
[10,1,2] 5 [10,4,2] 3 [7,3,2]  7 [10,8,2] 8
[8,4,2]  2 [8,5,2]  2 [10,6,2] 3 [9,8,2]  1
[1,2,3]  2 [1,3,3]  2 [1,4,3]  9 [1,5,3]  1
[1,7,3]  2 [2,6,3]  1 [2,7,3]  1 [5,7,3]  1
[7,9,3]  3 [6,9,3]  1 [3,6,3]  2 [3,10,3] 3
[1,10,3] 5 [4,10,3] 3 [3,7,3]  7 [8,10,3] 2
[4,8,3]  5 [5,8,3]  2 [6,10,3] 3 [8,9,3]  2
[2,1,3]  2 [3,1,3]  2 [4,1,3]  9 [5,1,3]  1
[7,1,3]  2 [6,2,3]  1 [7,2,3]  1 [7,5,3]  1
[9,7,3]  3 [9,6,3]  1 [6,3,3]  2 [10,3,3] 3
```

```
[10,1,3] 5 [10,4,3] 3 [7,3,3]  7 [10,8,3] 2
[8,4,3]  5 [8,5,3]  2 [10,6,3] 3 [9,8,3]  2;


param F:=
[1,2]  1 [1,3]  1 [1,4]   1 [1,5]  1  [1,7]  3
[2,6]  3 [2,7]  2 [5,7]   4 [7,9]  1  [6,9]  6
[3,6]  2 [3,10] 8 [1,10]  8 [4,10] 9  [3,7]  3
[8,10] 2 [4,8] 10 [5,8]  12 [6,10] 15 [8,9] 16
[2,1]  1 [3,1]  1 [4,1]   1 [5,1]  1  [7,1]  3
[6,2]  3 [7,2]  2 [7,5]   4 [9,7]  1  [9,6]  6
[6,3]  2 [10,3] 8 [10,1]  8 [10,4] 9  [7,3]  3
[10,8] 2 [8,4] 10 [8,5]  12 [10,6] 15 [9,8] 16;
param B:=
[1,1] 0 [2,1] 0 [3,1] 0 [4,1] -1 [5,1] 0 [6,1] 0
[7,1] 1 [8,1] 0 [9,1] 0 [10,1] 0
[1,2] 0 [2,2] 0 [3,2] 0 [4,2]  0 [5,2] -1 [6,2] 1
[7,2] 0 [8,2] 0 [9,2] 0 [10,2] 0
[1,3] 0 [2,3] 0 [3,3] 0 [4,3]  0 [5,3] -1 [6,3] 0
[7,3] 0 [8,3] 0 [9,3] 0 [10,3] 1;
```

Al resolver el problema obtenemos las soluciones

```
x [*,*,1]
:    1   2   3   4   5   6   7   8   9  10    :=
1    .   0   0   0   0   .   1   .   .   0
2    0   .   .   .   .   0   0   .   .   .
3    0   .   .   .   .   0   0   .   .   0
4    1   .   .   .   .   .   .   0   .   0
```

```
5    0   .   .   .   .   .   0   0   .   .
6    .   0   0   .   .   .   .   .   0   0
7    0   0   0   .   0   .   .   .   0   .
8    .   .   .   0   0   .   .   .   0   0
9    .   .   .   .   .   0   0   0   .   .
10   0   .   0   0   .   0   .   0   .   .
```

```
 [*,*,2]
:    1   2   3   4   5   6   7   8   9   10    :=
1    .   0   1   0   0   .   0   .   .   0
2    0   .   .   .   .   0   0   .   .   .
3    0   .   .   .   .   1   0   .   .   0
4    0   .   .   .   .   .   .   0   .   0
5    1   .   .   .   .   .   0   0   .   .
6    .   0   0   .   .   .   .   .   0   0
7    0   0   0   .   0   .   .   .   0   .
8    .   .   .   0   0   .   .   .   0   0
9    .   .   .   .   .   0   0   0   .   .
10   0   .   0   0   .   0   .   0   .   .
```

```
 [*,*,3]
:    1   2   3   4   5   6   7   8   9   10    :=
1    .   0   0   0   0   .   0   .   .   1
2    0   .   .   .   .   0   0   .   .   .
3    0   .   .   .   .   0   0   .   .   0
4    0   .   .   .   .   .   .   0   .   0
5    1   .   .   .   .   .   0   0   .   .
```

```
6    .   0   0   .   .   .   .   .   0   0
7    0   0   0   .   0   .   .   .   0   .
8    .   .   .   0   0   .   .   .   0   0
9    .   .   .   .   .   0   0   0   .   .
10   0   .   0   0   .   0   .   0   .   .
;

y [*,*]
:    1   2   3   4   5   6   7   8   9   10    :=
1    .   0   0   1   0   .   0   .   .   1
2    0   .   .   .   .   0   0   .   .   .
3    1   .   .   .   .   0   0   .   .   0
4    0   .   .   .   .   .   .   0   .   0
5    1   .   .   .   .   .   0   0   .   .
6    .   0   1   .   .   .   .   .   0   0
7    1   0   0   .   0   .   .   .   0   .
8    .   .   .   0   0   .   .   .   0   0
9    .   .   .   .   .   0   0   0   .   .
10   0   .   0   0   .   0   .   0   .   .
;
```

Que se corresponde con la construcción de la red que aparece en la figura 2.2 en trazo grueso

El coste correspondiente a la construcción de la red óptima bajo el primer escenario es de 37 unidades.

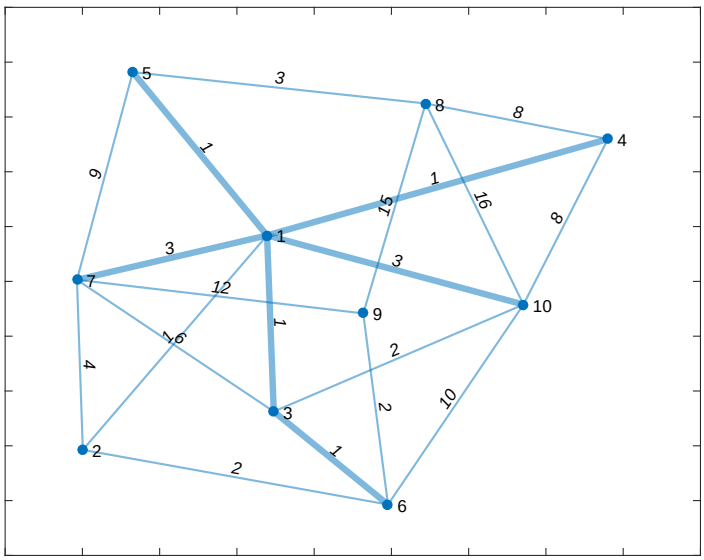


Figura 2.2: Diseño óptimo bajo el escenario s_1

Ahora resolvemos el problema bajo el escenario de costes de transporte dado por

```
param c:=
[1,2,1]  1 [1,3,1]  2 [1,4,1]  13 [1,5,1] 3
[1,7,1]  15 [2,6,1]  6 [2,7,1]  7 [5,7,1] 8
[7,9,1]  1 [6,9,1]  2 [3,6,1]  3 [3,10,1] 3
[1,10,1] 5 [4,10,1] 6 [3,7,1]  7 [8,10,1] 8
[4,8,1]  1 [5,8,1]  2 [6,10,1] 3 [8,9,1]  8
[2,1,1]  1 [3,1,1]  2 [4,1,1]  13 [5,1,1] 3
[7,1,1]  15 [6,2,1]  6 [7,2,1]  7 [7,5,1] 8
[9,7,1]  1 [9,6,1]  2 [6,3,1]  3 [10,3,1] 3
[10,1,1] 5 [10,4,1] 6 [7,3,1]  7 [10,8,1] 8
[8,4,1]  1 [8,5,1]  2 [10,6,1] 3 [9,8,1]  8
[1,2,2]  5 [1,3,2]  2 [1,4,2]  1 [1,5,2]  23
[1,7,2]  2 [2,6,2]  6 [2,7,2]  1 [5,7,2]  8
```

```
[7,9,2]  3 [6,9,2]  2 [3,6,2]  2 [3,10,2] 3
[1,10,2] 5 [4,10,2] 3 [3,7,2]  7 [8,10,2] 8
[4,8,2]  2 [5,8,2]  2 [6,10,2] 3 [8,9,2]  1
[2,1,2]  5 [3,1,2]  2 [4,1,2]  1 [5,1,2]  23
[7,1,2]  2 [6,2,2]  6 [7,2,2]  1 [7,5,2]  8
[9,7,2]  3 [9,6,2]  2 [6,3,2]  2 [10,3,2] 3
[10,1,2] 5 [10,4,2] 3 [7,3,2]  7 [10,8,2] 8
[8,4,2]  2 [8,5,2]  2 [10,6,2] 3 [9,8,2]  1
[1,2,3]  2 [1,3,3]  2 [1,4,3]  9 [1,5,3]  21
[1,7,3]  2 [2,6,3]  1 [2,7,3]  1 [5,7,3]  1
[7,9,3]  3 [6,9,3]  1 [3,6,3]  2 [3,10,3] 3
[1,10,3] 5 [4,10,3] 3 [3,7,3]  7 [8,10,3] 2
[4,8,3]  5 [5,8,3]  2 [6,10,3] 3 [8,9,3]  2
[2,1,3]  2 [3,1,3]  2 [4,1,3]  9 [5,1,3]  21
[7,1,3]  2 [6,2,3]  1 [7,2,3]  1 [7,5,3]  1
[9,7,3]  3 [9,6,3]  1 [6,3,3]  2 [10,3,3] 3
[10,1,3] 5 [10,4,3] 3 [7,3,3]  7 [10,8,3] 2
[8,4,3]  5 [8,5,3]  2 [10,6,3] 3 [9,8,3]  2;
```

En este nuevo escenario se han penalizado los costes de transporte del escenario anterior sobre los ejes usados para enviar la primera de las mercancías, desde el nodo 4 al nodo 7. Como se ve en la solución anterior se usan los ejes $(4, 1)$ y $(1, 7)$. Los costes de transporte unitario en ambos ejes han sido incrementados en 10 unidades en este segundo escenario. También hemos penalizado con 20 unidades de coste extra al de transporte asociado al eje $(1, 5)$ usado por las dos mercancías restantes bajo el primer escenario de costes. La solución que se obtiene ahora tiene coste 56 y corresponde con el valor de las variables

x [*,*,1]

:	1	2	3	4	5	6	7	8	9	10	:=
1	.	0	0	0	0	.	0	.	.	0	
2	0	.	.	.	.	0	0	.	.	.	
3	0	.	.	.	.	0	0	.	.	0	
4	0	.	.	.	.	.	.	1	.	0	
5	0	.	.	.	.	.	1	0	.	.	
6	.	0	0	.	.	.	.	.	0	0	
7	0	0	0	.	0	.	.	.	0	.	
8	.	.	.	0	1	.	.	.	0	0	
9	.	.	.	.	.	0	0	0	.	.	
10	0	.	0	0	.	0	.	0	.	.	

[*,*,2]

:	1	2	3	4	5	6	7	8	9	10	:=
1	.	0	1	0	0	.	0	.	.	0	
2	0	.	.	.	.	0	0	.	.	.	
3	0	.	.	.	.	1	0	.	.	0	
4	1	.	.	.	.	.	.	0	.	0	
5	0	.	.	.	.	.	0	1	.	.	
6	.	0	0	.	.	.	.	.	0	0	
7	0	0	0	.	0	.	.	.	0	.	
8	.	.	.	1	0	.	.	.	0	0	
9	.	.	.	.	.	0	0	0	.	.	
10	0	.	0	0	.	0	.	0	.	.	

[*,*,3]

:	1	2	3	4	5	6	7	8	9	10	:=
1	.	0	0	0	0	.	0	.	.	0	
2	0	.	.	.	.	0	0	.	.	.	
3	0	.	.	.	.	0	0	.	.	0	
4	0	.	.	.	.	.	.	0	.	0	
5	0	.	.	.	.	.	0	1	.	.	
6	.	0	0	.	.	.	.	.	0	0	
7	0	0	0	.	0	.	.	.	0	.	
8	.	.	.	0	0	.	.	.	0	1	
9	.	.	.	.	.	0	0	0	.	.	
10	0	.	0	0	.	0	.	0	.	.	

;

y [*,*]

:	1	2	3	4	5	6	7	8	9	10	:=
1	.	0	0	1	0	.	0	.	.	0	
2	0	.	.	.	.	0	0	.	.	.	
3	1	.	.	.	.	0	0	.	.	0	
4	0	.	.	.	.	.	.	1	.	0	
5	0	.	.	.	.	.	0	1	.	.	
6	.	0	1	.	.	.	.	.	0	0	
7	0	0	0	.	1	.	.	.	0	.	
8	.	.	.	0	0	.	.	.	0	1	
9	.	.	.	.	.	0	0	0	.	.	
10	0	.	0	0	.	0	.	0	.	.	

;

Gráficamente la nueva red óptima se corresponde con la representada en trazo grueso en la figura 2.3

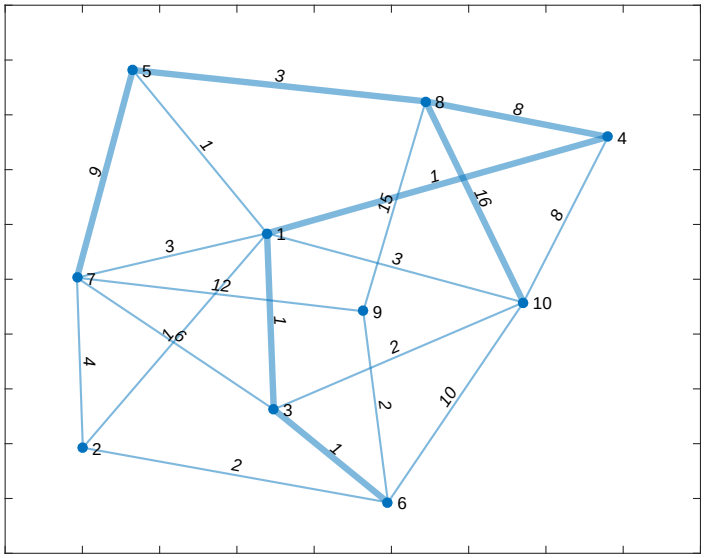


Figura 2.3: Diseño óptimo bajo el escenario s_2

Para resolver el problema minimax regret usaremos es siguiente código AMPL

```
set NODOS:={1..10};
set ARCOS within {NODOS,NODOS}:={
(1,2), (1,3), (1,4), (1,5), (1,7), (2,6), (2,7), (5,7),
(7,9), (6,9), (3,6), (3,10), (1,10), (4,10), (3,7), (8,10),
(4,8), (5,8), (6,10), (8,9),
(2,1), (3,1), (4,1), (5,1), (7,1), (6,2), (7,2), (7,5),
(9,7), (9,6), (6,3), (10,3), (10,1), (10,4), (7,3), (10,8),
(8,4), (8,5), (10,6), (9,8)};

set PRODUCTOS:={1..3};
set ESCENARIOS:={1..2};
```

```
param c{ARCOS,PRODUCTOS,ESCENARIOS};
param F{ARCOS};
param B{NODOS,PRODUCTOS};


var x{ARCOS,PRODUCTOS}>=0;
var y{ARCOS} binary;
var maxregret>=0;

minimize objetivo:maxregret;
subject to regret1:
sum{(i,j) in ARCOS,k in PRODUCTOS} c[i,j,k,1]*x[i,j,k]+
                  sum{(i,j) in ARCOS} F[i,j]*y[i,j]-37<=maxregret;
subject to regret2:
sum{(i,j) in ARCOS,k in PRODUCTOS} c[i,j,k,2]*x[i,j,k]+
                  sum{(i,j) in ARCOS} F[i,j]*y[i,j]-56<=maxregret;

subject to flow{j in NODOS, k in PRODUCTOS}:
sum{(i,j) in ARCOS}x[i,j,k]-sum{(j,h) in ARCOS}x[j,h,k]=B[j,k];
subject to bound{(i,j) in ARCOS,k in PRODUCTOS}:
x[i,j,k]<=y[i,j]+y[j,i];
```

Aquí se ha introducido el conjunto de índices para los escenarios ESCENARIOS y se ha añadido un nuevo índice al array de costes de transporte. Por último, se introducen dos nuevas restricciones denotadas por las etiquetas `regret1` y `regret2` que miden el regret bajo cada uno de los escenarios. El máximo de estos regrets se almacena en la variable `maxregret` que es mi-

nimizada. El valor objetivo óptimo de este problema es 12.8 que se obtiene en la solución

```
x [*,*,1]
:    1   2   3   4    5    6    7    8   9  10    :=
1    .   0   0   0  0.3    .  0.7    .   .   0
2    0   .   .   .    .    0    0    .   .   .
3    0   .   .   .    .    0    0    .   .   0
4    1   .   .   .    .    .    .    0   .   0
5    0   .   .   .    .    .  0.3    0   .   .
6    .   0   0   .    .    .    .    .   0   0
7    0   0   0   .    0    .    .    .   0   .
8    .   .   .   0    0    .    .    .   0   0
9    .   .   .   .    .    0    0    0   .   .
10   0   .   0   0    .    0    .    0   .   .


 [*,*,2]
:    1   2   3   4   5   6   7   8   9  10    :=
1    .   0   1   0   0   .   0   .   .   0
2    0   .   .   .   .   0   0   .   .   .
3    0   .   .   .   .   1   0   .   .   0
4    0   .   .   .   .   .   .   0   .   0
5    0   .   .   .   .   .   1   0   .   .
6    .   0   0   .   .   .   .   .   0   0
7    1   0   0   .   0   .   .   .   0   .
8    .   .   .   0   0   .   .   .   0   0
9    .   .   .   .   .   0   0   0   .   .
10   0   .   0   0   .   0   .   0   .   .
```

```
 [*,*,3]
:    1   2   3   4   5   6   7   8   9  10    :=
1    .   0   0   0   0   .   0   .   .   1
2    0   .   .   .   .   0   0   .   .   .
3    0   .   .   .   .   0   0   .   .   0
4    0   .   .   .   .   .   .   0   .   0
5    0   .   .   .   .   .   1   0   .   .
6    .   0   0   .   .   .   .   .   0   0
7    1   0   0   .   0   .   .   .   0   .
8    .   .   .   0   0   .   .   .   0   0
9    .   .   .   .   .   0   0   0   .   .
10   0   .   0   0   .   0   .   0   .   .
;
```

Es interesante observar que el problema proporciona soluciones x_{ij}^1 fraccionarias a la hora de determinar cómo se transporta la mercancía 1. La razón es que, fijado un diseño mediante las variables y_{ij} el conjunto de soluciones de transporte deja de ser un poliedro de vértices enteros al introducir las dos restricciones adicionales para computar el regret.

El diseño robusto de la red se obtiene a partir de las variables binarias y_{ij},

```
y [*,*]
:    1   2   3   4   5   6   7   8   9  10    :=
1    .   0   0   1   1   .   1   .   .   0
2    0   .   .   .   .   0   0   .   .   .
3    1   .   .   .   .   0   0   .   .   0
4    0   .   .   .   .   .   .   0   .   0
```

```
5    0   .   .   .   .   .   0   0   .   .
6    .   0   1   .   .   .   .   .   0   0
7    0   0   0   .   1   .   .   .   0   .
8    .   .   .   0   0   .   .   .   0   0
9    .   .   .   .   .   0   0   0   .   .
10   1   .   0   0   .   0   .   0   .   .
;
```

que definen el grafo representado en trazo grueso en la figura 2.4

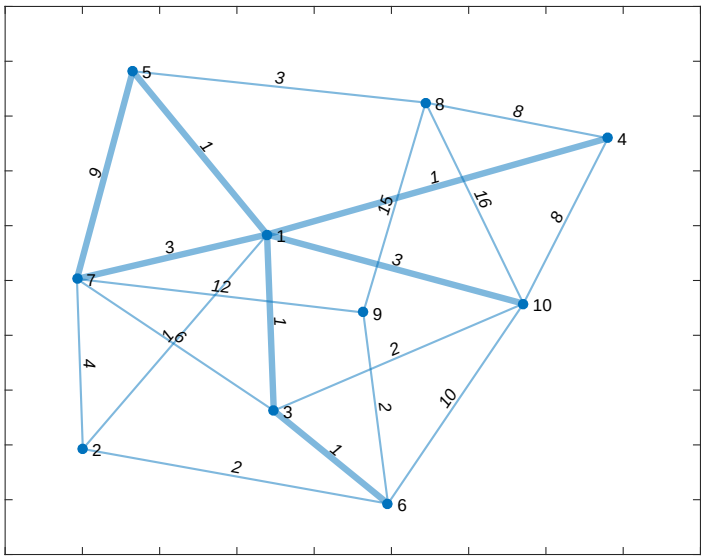
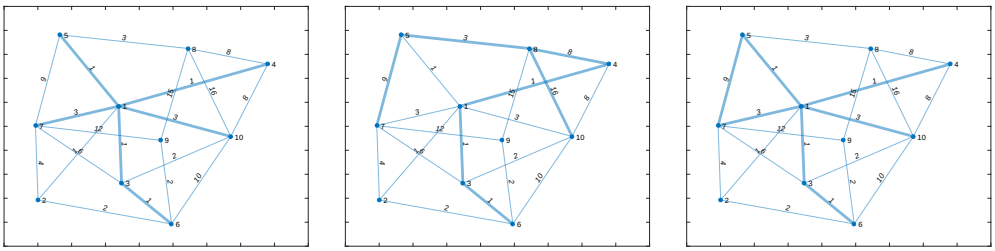


Figura 2.4: Diseño robusto de la red

Para tener mayor facilidad comparando los diseños óptimos bajo cada escenario y el diseño robusto representamos en la figura 2.5 los tres diseños



Escenario s_1 Escenario s_2 Solución robusta

Figura 2.5: Comparativa de los tres diseños

Capítulo 3

Complejidad y aproximación

3.1. Complejidad.

Como hemos mencionado en capítulos anteriores, los problemas robustos de optimización discreta son en general, más difíciles de resolver que sus formulaciones determinísticas. El principal factor es el aumento del grado adicional de libertad, es decir, de los escenarios establecidos. Para problemas de optimización clásicos en los que podemos hallar la solución en tiempo polinomial, como pueden ser asignación, mínimo árbol de unión, caminos más cortos, asignación de recursos o de secuenciación, las versiones robustas son débilmente o fuertemente $NP-$duros. Hemos encontrado, (véase [13]) que algunos de los problemas antes mencionados pueden resolverse de manera pseudo-polinomial basado en programación dinámica si el cardinal del conjunto de escenarios S es acotado. El conocido problema de mochila que es pseudo-polinómicamente resoluble también permanece pseudo-polinómicamente resoluble para un conjunto de escenarios S cuyo cardinal esté acotado, pero se convierte en $NP-$duro para un conjunto de escenarios en el que dicho cardinal no esté acotado, es decir, cuando el tamaño del conjunto de

escenarios forma parte del input.

Para determinar la complejidad de los problemas minimax usaremos reducciones polinomiales a problemas de optimización NP-duros considerados estándar en la Teoría de la Complejidad. A continuación describimos algunos de estos problemas.

- **El problema de la $2-$partición**

  - Datos: Sean un conjunto finito I y un tamaño $a_i \in \mathbb{Z}_+$ para $i \in I$.

  - Pregunta: ¿Existe un subconjunto $I' \subset I$ tal que $\sum_{i \in I'} a_i = \sum_{i \in I \setminus I'} a_i$?

  Es conocido que el problema de la $2-$partición es débilmente $NP-$duro incluso cuando $|I'| = \frac{|I|}{2}$.

- **El problema de la $3-$partición**

  - Datos: Sean un conjunto finito de $3l$ elementos I, una cota $B \in \mathbb{Z}_+$, y un tamaño $a_k \in \mathbb{Z}_+$ para $k \in I$, tal que cada a_k satisface $B/4 < a_k < B/2$ y tal que $\sum_{k \in I} a_k = lB$.

  - Pregunta: ¿Podemos dividir I en l conjuntos disjuntos tal que $\sum_{k \in I_i} a_k = B$ para $1 \leq i \leq l$?

  El problema de la $3-$partición es fuertemente $NP-$duro. (Véase [6]).

- **El problema de la partición par-impar**

  - Ejemplo: Sea una colección de números enteros positivos a_i, $1 \leq i \leq 2n$ tal que $\sum_i a_i = A$.

- Pregunta: ¿ Podemos dividir $\{a_i\}_{i=1}^{2n}$ en dos conjuntos disjuntos A_1, A_2 tal que $\sum_{a_i \in A_k} a_i = \frac{A}{2}$ para $k = 1, 2$ y exactamente uno de a_{2i}, a_{2i-1} pertenezca a A_1 para $1 \leq i \leq n$?

El problema de la partición par-impar es $NP-$duro. (Véase [6]).

- **El problema del conjunto de recubrimiento (SC)**

  - Datos: Sean un conjunto J de elementos y la familia I de subconjuntos finitos de J, y $k \geq 0$.

  - Pregunta: ¿Contiene I una subfamilia de conjuntos de tal manera que el número total de conjuntos en esta subfamilia es a lo más k, y cada elemento de J está incluido en al menos uno de los conjuntos seleccionados?

El problema del conjunto de recubrimiento es fuertemente $NP-$duro. (Véase [6]).

Veremos a continuación una serie de resultados de complejidad de algunos de los modelos vistos en el capítulo anterior.

Problema robusto de asignación

El problema robusto de asignación lo definimos en la sección 2.2. Para un escenario específico de datos de entrada el problema de asignación puede resolverse en tiempo polinomial ($\mathcal{O}(n^3)$).

Teorema 3.1.1 *Los problemas robustos de asignación minimax regret, minimax regret relativo y maximin son $NP-$duros incluso en el caso en el que $|S| = 2$.*

Problema robusto de caminos más cortos

El problema robusto de caminos más cortos visto en la sección 2.3 puede resolverse en su versión determinística, con una complejidad

$\mathcal{O}(min|V|^2, |A|\log(|V|))$ mediante el algoritmo de Dijkstra. En lo siguiente discutiremos la complejidad del problema de caminos más cortos sobre una red más restrictiva, una red laminada. Una red laminada es aquella que cumple las siguientes propiedades. El conjunto de los nodos puede dividirse en subconjuntos disjuntos como sigue $V = \{v_0\} \cup V_1 \cup V_2 \cup \cdots \cup V_m \cup \{v_t\}$ de manera que $V_i \cap V_j = \emptyset$, para todo $i \neq j$. Sólo existen arcos de v_0 a V_1, de V_m a v_t y de V_k a V_{k+1}, con $k = 1, \ldots, m-1$. Sea $\Delta = \max\{|V_k| : k = 1, \ldots, m\}$, Δ es conocido como la anchura de la red laminada. Así, las redes laminadas son casos especiales de las redes generales. Para el problema de caminos más cortos en redes laminadas se puede diseñar fácilmente un algoritmo aún más simple que el de Dijkstra basado en un procedimiento de programación dinámica estándar con una complejidad de $\mathcal{O}(|A|)$. Sobre este problema se tienen los siguientes resultados:

Teorema 3.1.2 *El problema robusto de caminos más cortos minimax regret es $NP-$completo incluso en redes laminadas de anchura 2 y con sólo 2 escenarios.*

Teorema 3.1.3 *El problema robusto de caminos más cortos maximin es $NP-$completo incluso para redes laminadas de anchura 2 y con sólo 2 escenarios.*

Teorema 3.1.4 *Los problemas robustos de caminos más cortos maximin y minimax regret se pueden resolver en tiempo pseudo-polinomial para redes laminadas en el caso en el que el conjunto de escenarios tenga cardinal acotado.*

Teorema 3.1.5 *El problema robusto de caminos más cortos maximin es fuertemente $NP-$duro para un conjunto de escenarios con cardinal no acotado.*

Teorema 3.1.6 *El problema robusto de caminos más cortos minimax regret es fuertemente $NP-$duro para un conjunto de escenarios con cardinal no acotado.*

Problema robusto de mínimo árbol de unión

El problema de mínimo árbol de unión visto en la sección 2.4 puede ser fácilmente resuelto mediante el algoritmo de Prim con una complejidad de $\mathcal{O}(\text{mín}\left\{|V|^2, |E|\log|V|\right\})$ o mediante el algoritmo de Kruskal con una complejidad de $\mathcal{O}(|E|\log|E|)$. En lo que sigue definiremos el problema de mínimo árbol de unión en un grafo muy restrictivo, una red regular (grafo rejilla).

Un grafo rejilla de orden (m, n) es definido mediante el conjunto de vértices

$$V = \{v_{ij} : i = 1, \cdots, m; j = 1, \cdots, n\}$$

y el conjunto de ejes $E = E_r \cup E_c$ con

$$E_r = \{(v_{ij}, v_{i,j+1}) : i = 1, \ldots, m; j = 1, \cdots, n-1\}$$

$$E_c = \{(v_{ij}, v_{i+1,j}) : i = 1, \ldots, m-1; j = 1, \cdots, n\}.$$

Los ejes en E_r son llamados ejes "fila", y los ejes en E_c son llamados ejes "columna". Un grafo rejilla de orden (m, n) tiene $m \cdot n$ nodos y $2mn - m - n$ ejes.

El siguiente teorema da el resultado de complejidad para el problema robusto de mínimo árbol de unión para el minimax regret (se pueden obtener resultados similares para los problemas de minimax regret relativo y maximin).

Teorema 3.1.7 *El problema robusto de mínimo árbol de unión minimax regret es $NP-$duro incluso bajo las siguientes restricciones:*

1. *G es un grafo rejilla con sólo dos filas, es decir, $m = 2$;*
2. *$c_e^s = 0$, $e \in E_c$, $s \in S$;*
3. *$|S| = 2$.*

Por lo que podemos observar que si bajo estas restricciones, el problema de mínimo árbol de unión es $NP-$duro, en general será, al menos, $NP-$duro.

Problema robusto de asignación de recursos

El problema robusto de asignación de recursos definido en la sección 2.6 puede resolverse en tiempo polinomial mediante un algoritmo greedy en $\mathcal{O}(n^2)$.

Teorema 3.1.8 *El problema robusto de asignación de recursos minimax regret relativo es $NP-$duro incluso con las siguientes restricciones:*

1. *Todas las funciones $c^s(\cdot)$ son lineales decrecientes.*
2. *x está restringido a tomar sólo valores binarios*
3. *$|S| = 2$.*

Teorema 3.1.9 *El problema robusto de asignación de recursos minimax regret con funciones de coste lineales decrecientes puede ser resuelto por*

un algoritmo pseudo-polinomial si el cardinal del conjunto de escenarios S está acotado.

Teorema 3.1.10 *El problema robusto de asignación de recursos minimax regret es fuertemente $NP-$duro para un conjunto de escenarios con cardinal no acotado.*

Problema robusto de mochila

El problema robusto de mochila visto en la sección 2.5 puede resolverse en tiempo pseudo-polinomial.

Teorema 3.1.11 *El problema robusto de mochila minimax regret es fuertemente $NP-$duro para un conjunto S de escenarios cuyo cardinal no esté acotado.*

Como hemos visto en estos resultados, los problemas presentados en el capítulo anterior tienen una complejidad elevada. Por lo que veremos en la próxima sección un método numérico de resolución que puede ser detenido antes de llegar a la optimalidad, en base a un ε que nos conducirá a una solución $\varepsilon-$óptima. Con esto obtenemos un método que es computacionalmente más eficiente para el cálculo de la solución óptima en los modelos propuestos, ya que no buscamos la solución óptima sino que podremos detener antes el proceso.

3.2. Método de descomposición de Benders.

En el capítulo anterior introdujimos el problema de diseño robusto de redes sin capacidad limitada en los arcos y, como dijimos, este problema tiene en cuenta los costes fijos de los arcos elegidos en una determinada configuración de red, así como el coste de enrutamiento de mercancías a

través de la red definida por dichos arcos. La principal incertidumbre a tener en cuenta para la formulación de este problema es que no se conocen los costes de los diversos productos básicos a transportar y que, a priori, no se conocen las cantidades de estos productos para transportarlos a través de la red. Las decisiones en los diseños de redes tienen un impacto determinante en la eficacia del sistema diseñado para un largo periodo de tiempo. En el caso del diseño de una red de transporte, el decisor se enfrenta a incertidumbre tanto en los costes de transporte como en las cantidades de producto que serán transportadas. En este último caso, depende de la oferta y la demanda que se den en el futuro por lo que una estrategia de actuación podría consistir en estimar dicha oferta y demanda bajo un conjunto finito de escenarios posibles y buscar una solución cercana al óptimo (diseño robusto).

Tal y como se ha visto en la literatura reciente ([9], [22]) muchos problemas combinatorios en los que resulta difícil encontrar una solución óptima adecuada, tales como asignación cuadrática y problema de diseño de redes multicomodidad, manifiestan la importante propiedad de tener múltiples soluciones, y significativamente diferentes, en un entorno del valor objetivo óptimo. Pues bien, a partir de la formulación del problema de diseño de red sin capacidad que vimos en el capítulo anterior, vamos a aplicarle el método de descomposición de Benders, que ha demostrado ser uno de los algoritmos más eficientes para el problema de diseño de red sin capacidad limitada en los arcos en el caso determinístico ([18]). Nuestro objetivo es encontrar una solución aproximada para el problema robusto de diseño de redes para un conjunto predeterminado de escenarios futuros operativos.

Partiendo de la formulación dada en (2.1), podemos ver que si ignoramos el operador de maximización en la función objetivo del problema de diseño

de redes y para un único escenario $s \in S$ fijo, la formulación resultante es la formulación del problema de diseño de redes estándar. Este problema lo denotaremos como $UNDP(s)$, sus ventajas frente a las formulaciones alternativas se discuten en [16]. La función objetivo de $UNDP(s)$ lo denotaremos por $z_s(x, y)$ y la solución óptima por $x^*(s), y^*(s)$. Con las técnicas de formulación que se utilizaron en el capítulo anterior, se puede resolver el problema del diseño robusto de una red, determinando previamente la solución óptima bajo cada escenario. Sin embargo, en problemas reales esta metodología podría ser ineficiente por el gran número de escenarios posibles. Por esta razón, vamos a desarrollar a continuación un método que determina una solución aproximada sin resolver el problema determinístico en cada escenario ($UNDP(s)$). Es a lo que llamaremos solución ε-óptima. Concretamente, nos referimos a (x, y) como un diseño de red, y ND denota el conjunto de todos los posibles diseños de red. Así, la solución óptima $(x^*(s), y^*(s))$ de $UNDP(s)$ será tal que

$$z_s(x^*(s), y^*(s)) = \min_{(x,y)\in ND} z_s(x, y)$$

En estas condiciones, para un conjunto de escenarios de entrada S dado, el diseño de red (x, y) es robusto sí y solo si

$$z_s(x, y) - z_s(x^*(s), y^*(s)) \leq \varepsilon z_s(x^*(s), y^*(s)), \quad s \in S$$

3.3. Método de descomposición de Benders aplicado al diseño robusto de redes.

El método de decomposición de Benders [1] es un algoritmo para la programación entera mixta que se ha aplicado con éxito a una variedad de problemas en el diseño de redes, con especial éxito en el caso de diseño de

red sin capacidad limitada en los arcos ([4], [8], [11], [16] y [21]). En esta sección discutiremos cómo el método de descomposición de Benders puede modificarse adecuadamente para la generación de diseños robustos de redes.

Cuando se aplica a problemas de diseño de redes, la descomposición de Benders va escogiendo iterativamente una configuración de red y, y a continuación va resolviendo el enrutamiento óptimo de los diversos productos a través de la red resultante, es decir, especificando x, y usa la solución del problema de enrutamiento para redefinir la configuración de red. Este último paso se produce con la generación de nuevas restricciones para el problema, que se denominan cortes de Benders. Estas restricciones forman parte de un problema entero mixto con las variables de elección de diseño de red discretas y una única variable continua que representa el coste total de la red (coste fijo y coste de enrutamiento). Este problema entero se conoce como el problema Maestro de Benders, en él se minimiza el coste total de la red sujeto a los cortes de Benders generados por cada configuración de red anterior. Cada vez que se resuelve el problema maestro, se obtiene una cota inferior para el coste del diseño de red óptimo. Además, cada solución al problema maestro determina una configuración de red, y el coste fijo, combinado con el de transporte óptimo sobre esta red es una cota superior para el valor óptimo del problema. Estas dos cotas permiten una terminación anticipada del algoritmo con nuestra evaluación del grado de suboptimalidad de los diseños de red obtenidos hasta ahora.

En la literatura se pueden encontrar resultados referidos al problema de diseño de red desde una perspectiva de "optimalidad" (es decir, determinar la solución del problema para un escenario de datos específicos). En sus conti-

nuos esfuerzos para resolver la complejidad del problema, los investigadores utilizaron diversos esquemas mejorando la descomposición de Benders (ver [16], [17]). Las versiones más refinadas de la metodología Benders aplicada al diseño de redes han sido obtenidas para encontrar y verificar soluciones óptimas para problemas con 30 nodos, 130 arcos con 40 arcos fijos abiertos (es decir, estos arcos ya existen en el diseño) y 58 productos básicos (véase [16]). Estos enfoques suelen ser ineficaces para resolver el problema con incertidumbre. Además, el enfoque de fuerza bruta solo generará una lista parcial de soluciones óptimas (o cercanas al óptimo en el caso de una terminación anticipada del algoritmo) para cada escenario de datos. Este enfoque no toma ninguna medida para tratar de generar en el proceso de optimización ninguna decisión robusta, y es probable que la intersección de estas listas no contenga ninguna solución robusta cualificada.

El enfoque que veremos a continuación apunta a generar diseños de redes robustos con un esfuerzo computacional significativamente menor que la ejecución repetida de aplicaciones independientes del Algoritmo de Benders. Podemos ver en la referencia [16] que los autores muestran que la mayor parte del esfuerzo computacional del Algoritmo de Benders (casi el 90 % de su tiempo de ejecución) se invierte en la solución del problema maestro. Esto se debe a que tenemos un problema maestro para cada escenario de datos. Cuando un problema maestro se resuelve, a partir de su solución obtenemos una configuración de red tentativa, y podemos resolver el enrutamiento óptimo de los diversos productos en ese diseño de red para cada escenario de datos de manera eficiente utilizando el algoritmo de Dijkstra ([3]). Después, usando teoría de dualidad ([20]), podemos generar un nuevo corte para cada uno de los problemas maestros correspondientes a cada uno de los escenarios de datos. Como veremos esta generación cruzada de cortes de Benders

(es decir, generando un corte para cada problema maestro correspondiente a cada escenario cada vez que se resuelve un problema maestro) acelera la convergencia del problema maestro para todos los escenarios. Un beneficio adicional de este procedimiento es que cada solución obtenida para cada problema maestro, en el proceso de cortes de generación cruzada que hemos mencionado, se compara repetidamente con las mejores cotas superior e inferior obtenidas para cada escenario de datos.

3.4. Generación de cortes para el Problema Maestro del Algoritmo de Benders

Consideramos un $y \in \Gamma$ fijo, así el problema $UNDP(s)$ se reduce al problema lineal

$$BP(s,y): \quad \min W_s(y) = \sum_{k\in K} \sum_{\{i,j\}\in A} (c_{ij}^k(s)x_{ij}^k + c_{ji}^k(s)x_{ji}^k)$$

s.t

$$\sum_{i\in N} x_{ij}^k - \sum_{h\in N} x_{jh}^k = \left\{ \begin{array}{ll} -1 & \text{si } \ j = O(k); j \in N \\ 1 & \text{si } \ j = D(k); k \in K \\ 0 & \text{en otro caso.} \end{array} \right\}; \quad j \in N, k \in K,$$

$$x_{ij}^k \leq y_{ij}, x_{ji}^k \leq y_{ij}, \{i,j\} \in A, k \in K,$$

$$x_{ij}^k, x_{ji}^k \geq 0; \{i,j\} \in A, k \in K.$$

De esta manera, $W_s(y)$ representa el coste de transporte para un diseño concreto de red dado por una configuración de red y fija.

En el enfoque de la descomposición de Benders, se buscan las soluciones del dual de $BP(s, y)$ y son utilizadas para generar cortes adicionales para el problema maestro de Benders. El conjunto factible de este problema dual no depende del diseño de la red, con lo cuál cualquier solución factible dual nos dará una acotación inferior del valor objetivo óptimo de problema $BP(s, y)$. Dado que los arcos no tienen capacidad, fijado y podemos descomponer el problema $BP(s, y)$ y su dual en K subproblemas independientes, uno para cada mercancía. Precisamente por no tener capacidades, la solución óptima puede ser encontrada mediante la resolución de un problema de caminos más cortos sobre la configuración de red y, y con matriz distancia $C^k(s)$. El dual para cada uno de los k subproblemas de $BP(s, y)$ viene dado por:

$$
\begin{aligned}
DBP_k(s,y): \quad & \text{máx}\, u^k_{D(k)}(s) - u^k_{O(k)}(s) + \sum_{\{i,j\}\in A} (v^k_{ij}(s) + v^k_{ji}(s))y_{ij} \\
& \text{s.t} \\
& u^k_j(s) - u^k_i(s) - v^k_{ij}(s) \leq c^k_{ij}(s), \;\; \{i,j\} \in A, \\
& u^k_i(s) \geq 0, \quad i \in N, \\
& v^k_{ij}(s), v^k_{ji}(s) \geq 0, \quad \{i,j\} \in A.
\end{aligned}
$$

donde $u^k_j(s)$ es la variable dual correspondiente a la restricción para el k-ésimo producto y nodo j en el primer bloque de restricciones del diseño de

redes, y $v_{ij}^k(s)$ es la variable dual correspondiente a la restricción para el arco $\{i, j\}$ para el segundo bloque de restricciones del problema de diseño de red. Observemos además, que las restricciones del problema dual no dependen de la configuración de red y.

Una vez resueltos todos los problemas duales, obtendremos unos valores para u y v, y sumando todos los valores objetivos de los problemas duales estaremos determinando un coste de enrutamiendo bajo una estructura y.

Si todos los K problemas de caminos más cortos en la red definida por y son factibles, obtendremos un corte de la forma

$$z \geq \sum_{\{i,j\} \in A} F_{ij} y_{ij} + \sum_{k \in K} \left[(u_{D(k)}^k(s) - u_{O(k)}^k(s)) - \sum_{\{i,j\} \in A} (v_{ij}^k(s) + v_{ji}^k(s)) y_{ij} \right] \tag{3.1}$$

donde el miembro de la derecha representa el coste correspondiente a transportar la mercancía k bajo el escenario s y a través de la estructura y de la red. Puesto que lo que queremos es minimizar el máximo de los posibles costes, z representa el máximo coste de transporte de la mercancía k bajo el escenario s a través de la estructura y.

Para un escenario específico de datos de entrada $s \in S$, la restricción anterior es una función de z e y, donde z es la variable continua en el problema maestro de Benders (BM_s) para cada escenario s

BM_s : $\min z$

s.t

$$z \geq \sum_{\{i,j\}\in A} F_{ij} y_{ij} + \sum_{k\in K} \left[(u^k_{D(k)}(s) - u^k_{O(k)}(s)) - \sum_{\{i,j\}\in A} (v^k_{ij}(s) + v^k_{ji}(s)) y_{ij} \right] \quad (3.2)$$

$$y_{ij} \in \{0,1\}, \quad \{i,j\} \in A,$$

$$y \in \Gamma.$$

Así pues, estamos minimizando el máximo coste que se produce bajo las estructuras de la red consideradas, es decir, se persigue encontrar aquella estructura y de tal forma que el máximo de todos ellos sea lo menor posible.

Las constantes $u^k_j(s)$ y $v^k_{ij}(s)$ usadas en la primera restricción, correspondiente al corte que hemos hecho, son los valores óptimos de las variables duales del problema $DBP_k(s,y)$. Estos valores se pueden calcular directamente como se indica a continuación. Sea $u^k_{O(k)}(s) = 0$ y $u^k_j(s)$ igual a la longitud del camino más corto desde $O(k)$ al nodo j en la red definida por y y matriz distancia $C^k(s)$. Una vez conocidos los valores de u, los valores de $v^k_{ij}(s)$ y $v^k_{ji}(s)$ vendrán dados por

$$\begin{aligned} v^k_{ij}(s) &= \text{máx}\left\{u^k_j(s) - u^k_i(s) - c^k_{ij}(s), 0\right\} \\ v^k_{ji}(s) &= \text{máx}\left\{u^k_i(s) - u^k_j(s) - c^k_{ji}(s), 0\right\} \end{aligned}$$

Podemos encontrar estos valores siempre que todos los problemas duales tengan solución factible. En el caso en el que para alguna mercancía la red definida por y no tuviese solución factible, querría decir que existiría un conjunto de arcos CS_k separando $O(k)$ de $D(k)$ con $y_{ij} = 0$ para todo $\{i, j\} \in CS_k$. A estos arcos se les denomina conjunto de corte, es decir, un subconjunto de arcos del grafo que desconecta los nodos de origen y destino para la mercancía k. Si esto ocurre, querría decir que no podemos mandar la mercancía del nodo $O(k)$ al nodo $D(k)$ y por lo tanto, el problema sería infactible. Por este motivo, buscamos imponer que se elija en el diseño de la red al menos, uno de los arcos que unen esos dos nodos. Es decir, añadiremos el siguiente corte de factibilidad al problema maestro de Benders

$$\sum_{\{i,j\} \in CS_k} y_{ij} \geq 1. \tag{3.3}$$

De esta forma, el problema maestro de Benders es de la forma (3.2), con el conjunto Γ incluyendo todos los cortes de factibilidad de la forma (3.3) para cada mercancía k para la cuál $D(k)$ no es accesible desde $O(k)$. El conjunto de corte CS_k puede ser fácilmente generado durante la ejecución del algoritmo de caminos más cortos para la mercancía k. El camino más corto para cada producto se calcula utilizando el algoritmo de Dijkstra. Este algoritmo, sin ninguna carga computacional adicional, puede encontrar el camino más corto desde $O(k)$ a todos los demás nodos $i \in N$, $i \neq O(k)$. Si el algoritmo no consigue obtener un camino desde $O(k)$ a $D(k)$, entonces podemos identificar todos los nodos que son accesibles desde $O(k)$. Los cortes establecidos CS_k se forman incluyendo todos los arcos, no incluidos en la solución actual y, que unen un nodo alcanzable de $O(k)$ a un nodo no

alcanzable de $O(k)$. Observamos que la adición de un solo corte de este tipo no garantiza la accesibilidad de $D(k)$ desde $O(k)$. Sin embargo, la aplicación repetida de estos cortes asegura que $D(k)$ sea alcanzado.

Es importante observar que la solución de BM_s para cada s, nos conducirá a la generación de un nuevo y. La solución de los problemas de caminos más cortos para las distintas mercancías en la red definida por y nos permitirá generar cortes de la forma (3.1) y (3.3) para todos los problemas maestros de Benders BM_s, $s \in S$.

Los cortes de Benders de la forma (3.1) son conocidos en la literatura de diseño de redes como cortes Estándar de Benders. Estos cortes pueden interpretarse como cotas sobre la solución óptima del problema maestro de Benders correspondiente. Magnanti y al. presentaron en [16] una técnica para fortalecer los cortes estándar de Benders que daba lugar a cotas inferiores más finas. Estos cortes, conocidos como cortes Fuertes de Benders, tienen la capacidad de acelerar la convergencia del algoritmo de Benders. Los cortes fuertes de Benders tienen la misma forma que (3.1) pero, en lugar de usar el valor de $u_j^k(s)$ y $v_{ij}^k(s)$ descritos anteriormente, utiliza una nueva solución dual $\left\{\overline{u}_i^k(s), \overline{v}_{ij}^k(s)\right\}$ especificada como sigue

$$
\begin{aligned}
\overline{u}^k_{O(k)}(s) &= 0, \\
\overline{u}^k_{D(k)}(s) &= u^k_{O(k)}(s), \\
\overline{u}^k_i(s) &= \text{mín}\left\{u^k_i(s), \overline{u}^k_{D(k)}(s) - D^k_i\right\}, \quad \text{para} \quad i \neq O(k), D(k), \\
\overline{v}^k_{ij}(s) &= \text{máx}\left\{0, u^k_j(s) - \overline{u}^k_i(s) - c^k_{ij}(s)\right\},
\end{aligned}
$$

donde D^k_i es el mínimo coste de ruta de una unidad de mercancía k desde el nodo i hasta $D(k)$ en una red que incluye todos los arcos candidatos. Obsérvese que el cálculo de cortes fuertes no es computacionalmente más exigente que el cálculo de los cortes estándar, ya que las cantidades D^k_i se pueden obtener durante una fase de preprocesamiento para todo $i \in N$, $k \in K$.

3.5. Algoritmo de Benders

Para describir de forma pseudo-algorítmica nuestro algoritmo de Benders para el problema de diseño robusto de redes sin capacidad, necesitaremos introducir la siguiente notación. Sean $UB(s)$ y $LB(s)$, que denotan respectivamente una cota superior e inferior de $UNDP(s)$. Usaremos la variable binaria $SO(s)$ para indicar si el problema $UNDP(s)$ ha sido resuelto de manera óptima en este paso de nuestro procedimiento (es decir, $SO(s) = 1$ si ha sido resuelto de manera óptima, y $SO(s) = 0$ en otro caso). Una lista LR_p almacena todas las soluciones robustas (es decir, que son ε-óptimas

para todo $s \in S$). Denotamos por $MP(s)$ el conjunto de restricciones del problema maestro de Benders para la formulación $UNDP(s)$ en este paso de nuestro procedimiento y $ZMP(s)$ el valor objetivo óptimo de ese problema maestro. Como es evidente, este conjunto se actualiza continuamente durante la ejecución de nuestro procedimiento con la adición de los demás cortes de Benders generados. También denotamos por $\overline{z}(s, y)$ el coste total de configuración de la red y bajo el escenario s donde $\overline{z}(s, y)$ viene dado por

$$\overline{z}(s, y) = W_s(y) + \sum_{\{i,j\} \in A} F_{ij} y_{ij}.$$

El algoritmo puede ser descrito formalmente como sigue

Algoritmo de Benders

- **Paso** 0**: Inicialización.**

  Sea $UB(s) = +\infty$, $LB(s) = 0$, y $SO(s) = 0$ para $s = 1, \ldots, |S|$.
  Sea $MP(s) = \emptyset$, para $s = 1, \ldots, |S|$, y $LR_p = \emptyset$.
  Calcular D_i^k para todo i y $D(k)$, $k \in K$.
  Sea $s = 1$, e ir al Paso 1.

- **Paso** 1**: Generación del diseño de redes.**

  Resolver el problema maestro para el escenario s con optimalidad. Sea $\overline{y} = \overline{y}_{ij}$ la configuración de red obtenida. Sea $LB(s) = ZMP(s)$, e ir al Paso 2.

- **Paso 2: Procedimiento de generación de cortes.**

  Paso 2.1: **Comprobación de la factibilidad.**

  Resolver el problema $BP(1, \overline{y})$. Si es factible, ir al paso 2.2. En otro caso, generar un conjunto de corte CS_k y una restricción de la forma (3.3) para cada mercancía k cuyo flujo en la configuración de red $\overline{y}$ es infactible. Añadir estas nuevas restricciones al conjunto de todas las restricciones $MP(s)$, $s = 1, \ldots, |S|$. Ir al paso 1.

  Paso 2.2: **Generación de cortes de Benders.**

  Resolver los problemas $BP(\gamma, \overline{y})$ para los escenarios $\gamma = 2, \ldots, |S|$. Para cada escenario introducir un corte de Benders de la forma (3.1) (corte estándar o fuerte).
  Añadir la nueva restricción al conjunto de restricciones $MP(\gamma)$ del correspondiente problema maestro.

  Paso 2.3: **Actualización de la lista de diseño robusto de red.**

  Para $\gamma = 1, \ldots, |S|$ hacer:

  - Sea $UB(\gamma) = \text{mín}\,\{UB(\gamma), \overline{z}(\gamma, \overline{y})\}$
  - Si $\dfrac{\overline{z}(\gamma, \overline{y}) - UB(\gamma)}{UB(\gamma)} \leq \varepsilon$ para $\gamma = 1, \ldots, |S|$, entonces sea $LR_p = LR_p + \overline{y}$.

    Para $\gamma = 1, \ldots, |S|$ hacer:

  - Para $y \in LR_p$ hacer:
  - Si $\overline{z}(\gamma, \overline{y}) > (1 + \varepsilon)UB(\gamma)$, entonces sea $LR_p = LR_p - y$.

  Ir al paso 3.

- **Paso 3: Comprobación de la optimalidad.**

  Si $UB(s) = LB(s)$, entonces $\overline{y}$ es la solución óptima del problema $UNDP(s)$. Sea $SO(s) = 1$. Si $SO(s) = 1$ para todo $s = 1, \ldots, |S|$, entonces parar; en otro caso, sea s igual al índice del siguiente problema $UNDP(s)$ no resuelto. Ir al paso 1.

La siguiente observación es importante para entender el procedimiento algorítmico anterior. En nuestras formulaciones $UNDP(s)$, $s = 1, \ldots, |S|$, mantenemos el mismo conjunto de mercancías K. Podría ocurrir que algún posible escenario s utilizara sólo un conjunto de mercancías, en tal caso, modelamos esa posibilidad con el simple ajuste $c_{ij}^k(s) = 0$ para toda mercancía k excluida del escenario s. Sin embargo, las restricciones de flujo para los productos excluidos k siguen apareciendo en la formulación $UNDP(s)$ para ese escenario. Dado que las restricciones de flujo para todos los productos k se incluyen en todas las formulaciones $UNDP(s)$, si una configuración de red $y = (y_{ij})$ es factible para una formulación $UNDP(s)$ es factible para todas ellas. Por lo tanto, en el paso 2.1 necesitamos verificar la viabilidad de una sola formulación. Si la configuración de red utilizada (es decir, y) no es factible para el escenario, los cortes de factibilidad generados serán aplicables a todos los problemas maestros de Benders.

En el paso 2.2, la solución del problema $BP(\gamma, \overline{y})$ implica la solución de los K problemas de caminos más cortos. En el paso 2.3 actualizamos primero la cota superior para cada escenario. Si nuestro diseño de red actual y, después de ser evaluado su rendimiento para cada escenario (es decir, el cálculo de $\overline{z}(\gamma, \overline{y})$), se encuentra cerca (en base a ε) de la mejor solución disponible hasta ahora para cada escenario, entonces estará incluida en la

lista de diseños de red robustos LR_p. Esta lista se actualiza continuamente a medida que se mejoran las soluciones disponibles para los diversos escenarios. Cuando el algoritmo se detiene, la solución óptima se ha obtenido para cada formulación $UNDP(s)$ y la lista LR_p contiene todos los diseños de red robustos encontrados.

El algoritmo se puede ejecutar utilizando tanto cortes estándar como fuertes de Benders (véase el paso 2.2). Sin embargo, la convergencia del algoritmo es significativamente más rápida cuando se usan cortes fuertes de Benders.

3.6. Coste esperado de los diseños robustos

Un enfoque bastante tradicional de ver el problema de diseño de redes sin capacidad limitada en los arcos en el que existe una incertidumbre en los datos de entrada es formularlo como un problema de diseño de redes estocástico en dos etapas. Este enfoque requiere que el decisor conozca no sólo el conjunto de escenarios posibles S, tal y como requería el enfoque robusto, sino además como habíamos mencionado anteriormente, debe identificar la probabilidad q_s de que cada escenario s pueda ocurrir. El modelado del problema de optimización consta de dos etapas ya que el decisor necesita especificar en una primera fase solo la configuración de red $y = (y_{ij})$, mientras que el enrutamiento real de los productos a través de la red, es decir, la determinación de $x = (x_{ij}^k)$, será especificada en una segunda etapa del proceso.

El coste esperado del problema de diseño de redes sin capacidad limitada en los arcos viene dado por

$$(ECP) \quad \min_{y} \sum_{s \in S} \left[W_s(y) + \sum_{\{i,j\} \in A} F_{ij} y_{ij} \right] q_s$$

s.t

$$\sum_{i \in N} x_{ij}^k - \sum_{h \in N} x_{jh}^k = \left\{ \begin{array}{ll} -1 & \text{si } \ j = O(k); j \in N \\ 1 & \text{si } \ j = D(k); k \in K \\ 0 & \text{en otro caso.} \end{array} \right\}; \quad j \in N, k \in K,$$

$$x_{ij}^k \leq y_{ij}, x_{ji}^k \leq y_{ij}, \{i,j\} \in A, k \in K,$$

$$x_{ij}^k, x_{ji}^k \geq 0; y_{ij} \in \{0,1\}, \{i,j\} \in A, k \in K,$$

$$y \in \Gamma.$$

El problema (ECP) es un problema entero estocástico y resulta muy complicado de resolver de manera óptima. Stougie en [23] presenta un marco general para el diseño y análisis de algoritmos jerárquicos y heurísticos para esa clase de problemas. Sin embargo, si todas las soluciones robustas son $\varepsilon-$óptimas para todos los escenarios que se hayan encontrado, es fácil calcular las cotas sobre el valor objetivo de (ECP).

Sea $(x^*(s), y^*(s))$ la solución óptima de $UNDP(s)$, y denotemos por y^{**} la solución óptima de (ECP). Entonces, la siguiente desigualdad es válida para todo diseño de red robusto y^{ε}

$$\frac{\overline{z}(s, y^{\varepsilon})}{1+\varepsilon} \leq \overline{z}(s, y^{*}(s)) \leq \overline{z}(s, y^{**}), \quad s \in S.$$

La primera parte de la desigualdad anterior se sigue inmediatamente de la definición de diseño de red robusto, y la segunda parte se sigue de la optimalidad de $y^{*}(s)$ para $UNDP(s)$. De la desigualdad anterior, y después de calcular el valor esperado sobre s obtenemos

$$\frac{\sum_{s \in S} \left[\overline{z}(s, y^{\varepsilon}) - \overline{z}(s, y^{**})\right] q_s}{\sum_{s \in S} \overline{z}(s, y^{**}) q_s} \leq \varepsilon$$

Así, (3.6) implica que cualquier diseño de red robusto y^{ε} es también $\varepsilon-$óptimo para el problema del coste esperado (ECP). En otras palabras, el algoritmo de descomposición de Benders presentado en la sección anterior, puede ser utilizado para resolver (ECP) con $\varepsilon-$optimalidad. Además, de (3.6) podemos concluir que si los diseños de redes robustos que están disponibles son ε-óptimos para cualquier escenario realizable, el valor de la información adicional sobre el valor exacto de las probabilidades q_s está limitado por la cantidad $\varepsilon \, \text{máx}_s \, \overline{z}(s, y^{*}(s))$.

3.7. Conclusiones

En este capítulo hemos visto como los modelos de optimización robusta introducidos en el capítulo anterior tienen una complejidad teórica *no polinomial*. En primer lugar, se debe indicar que la complejidad teórica es una cota superior del número de operaciones necesarias para obtener un óptimo sean cuales sean los datos de entrada. Solo se tiene en cuenta la cantidad de espacio necesario para almacenar dichos datos de entrada. En la práctica, muchos de estos problemas pueden ser resueltos para tamaños razonables

puesto que los datos guardan ciertas relaciones entre sí que los hacen más fáciles de tratar.

Por otro lado, el que sean problemas de alta complejidad teórica refuerza la idea de que es necesario utilizar aproximaciones y técnicas heurísticas que permitan inicializar los algoritmos con buenas soluciones. Esto acortará el proceso de cómputo. Además, como se ha dicho anteriormente al introducir el método de descomposición de Benders, se puede detener el método numérico antes de que se verifique el test de parada, es decir, antes de llegar a optimalidad y se tiene una cota de la bondad de la aproximación propuesta.

Incluso la obtención de soluciones aproximadas tiene mucho interés práctico. Recordemos que se está modelando un conjunto finito de escenarios posibles. Estos escenarios es muy posible que sean estimaciones obtenidas en base a distribuciones condicionadas de variables aleatorias. Por lo tanto, los escenarios considerados son, en sí mismos, aproximaciones a los valores reales que tomarán los costes, capacidades o recursos disponibles. En esta situación, cabe preguntarse si tiene sentido buscar optimalidad en las soluciones propuestas por el modelo cuando se basan en escenarios que son aproximaciones. Dicho de otro modo, el interés de las soluciones robustas, mas allá de si son óptimas o aproximadas, se basa en las características que los modelos de optimización estudiados en este trabajo infieren a dichas soluciones. En particular, las soluciones tendrán un comportamiento relativamente eficiente sea cual sea el escenario que ocurra sin necesidad de hacer uso de las probabilidades con las que ocurra, que por otro lado suelen ser difíciles de obtener o modelar.

Bibliografía

[1] J.J. Benders, *Partitioning procedure for solving mixed variable programming problems*, Numerische Mathematik, 1962.

[2] V. Chvatal, *Linear programming*, W.H. Freeman and Company, 1983.

[3] E.W. Dijkstra, *A note on two problems in connection with graphs*, Numerische Mathematik, 1959.

[4] G.G. Guerrin Florian, M. and G. Bushel, *The engine scheduling problem on a railway network*, INFOR J., 1976.

[5] S. French, *Decision theory*, John Wiley & Sons, 1986.

[6] M.R. Garey and D.S. Johnson, *Computers and intractability*, W.H. Freeman, 1979.

[7] B. Gavish, *Topological design of centralized computer networks - formulations and algorithms*, Networks, 1982.

[8] A.M. Geoffrion and G. Graves, *Multicommodity distribution system design by Benders decomposition*, Management Science, 1974.

[9] G.J. Gutierrez and P. Kouvelis, *Robust flowpath designs for automated guided vehicle systems*, Working Paper, Management Department, University of Texas at Austin, 1996.

[10] F.S. Hillier and G.J. Lieberman, *Introduction to mathematical programming*, McGraw Hill, 1990.

[11] H.H. Hoang, *Topological optimization of networks: A nonlinear mixed integer model employing generalized* Benders *decomposition*, Working Paper, 1982.

[12] P. Kouvelis Karabati, S. and G. Yu, *A min-max-sum resource allocation problem and its applications*, Working Paper, Fuqua School of Business, 1996.

[13] P. Kouvelis and G. Yu, *Robust discrete optimization and its applications*, Working Paper, Department of MSIS, Graduate School of Business, 1995.

[14] P. Kouvelis, P. Yu, G.Kouvelis, and G. Yu, *Robust discrete optimization and its applications*, Kluwer Academic Publisher, 1997.

[15] R.C. Larson and A.R. Odoni, *Urban operations research*, Prentice Hall, Englewood Cliffs, 1981.

[16] P. Mireault Magnanti, T.L. and R.T. Wong, *Tailoring* Benders *decomposition for uncapacitated network design*, Mathematical Programming Study, 1986.

[17] T.L. Magnanti and R.T. Wong, *Accelerating* Benders *decomposition: Algorithmic enhancement and model selection criteria*, Operations Research, 1981.

[18] ———, *Network design and tansportation planning: Models and algorithms*, Transportation Science, 1984.

[19] K.G. Murty, *Linear programming*, Wiley, 1983.

[20] G.L. Nemhauser and L.A. Wolsey, *Integer and combinatorial optimization*, Wiley, 1988.

[21] R. Richardson, *An optimization approach to routing aircraft*, Transportation Science, 1976.

[22] M.J. Rosenblatt and H.L. Lee, *A robustness approach to facilities design*, International Journal of Production Research, 1987.

[23] L. Stougie, *Design and analysis of algorithms for stochastic integer programming*, CWI Tract 37, 1987.

[24] P. Toth, *Dynamic programming algorithms for the zero-one knapsack problem*, Computing,, 1980.