**C1 En términos de complejidad ¿cuándo se dice que un problema es difícil? Mencione un ejemplo**

A la vez es un problema complejo, si por complejidad nos referimos a la computacional. Un problema se cataloga como inherentemente difícil si su solución requiere de una cantidad significativa de recursos computacionales, sin importar el algoritmo utilizado . El problema de la mochila forma parte de una lista histórica de problemas NP Completos elaborada por Richard Karp en 1972 2 . En el caso del problema de la mochila, si contáramos con 4 productos, para saber cuál es la mejor solución podríamos probar las 24=16 posibilidades. El 2 se desprende del hecho de que cada decisión es incluir o no al producto y el 4 de la cantidad de productos. 16 posibilidades es un número manejable, sin embargo, si la cantidad de elementos por ejemplo ascendiera a 20, tendríamos que analizar nada más y nada menos que 220=1048576 posibilidades.

**C2 Si el problema de distribución se puede resolver de manera exacta como un problema de programación lineal continua ¿por qué se utilizan heurísticas de construcción para ese problema?**

Porque en la mayoría de los casos no se puede transportar partes fraccionarias de productos, es decir, el uso más conocido es para variables enteras. Por este motivo se utilizan heurísticas, ya que su resolución en programación lineal entera es un problema difícil. Muchas veces se requiere una solución aproximada, que se acerque lo suficiente a la exacta, pero sin perder demasiado tiempo en procesamiento.

**C1 ¿Es el problema de Asignación un problema difícil? ¿Por qué?.**

**C2 El problema de Asignación ¿es un problema difícil? ¿se puede resolver de manera exacta aplicando el método Simplex, sin tener que aplicar Branch & Bound u otro método para que las variables tengan valor entero?**

Asignación es un problema difícil, porque es un problema combinatorio donde el software tiene que evaluar cada posibilidad antes de dar con un resultado. En el caso de haber muchas combinaciones esto puede ser muy costoso. Sin embargo, su relajación lineal se resuelve en tiempo polinomial y da una solución entera. Recordar que el problema de asignación es un problema de distribución o transporte con orígenes y destinos de 1.

**C2 De los dos planteos de modelización más conocidos para el viajante (plantear todas las restricciones que**

**evitan subtours, también llamado MZT y el que agrega las restricciones de Ui) ¿cuál es el mejor en términos de**

**resolución del problema? ¿por qué piensa que ese es el mejor?**

El número exponencial de restricciones hace impráctico resolverlo directamente por el método de Ui. Una opción es agregar únicamente las restricciones para evitar subtours en los casos en los cuales arma subtour y no en todos los casos. Este último es mejor cuando el problema del viajante tiene muchas ciudades, es decir, muchas ecuaciones del problema en sí. En cambio el método de las Ui es muy práctico cuando hay pocas ciudades ya que no aumenta considerablemente el número de ecuaciones y resulta más práctico la resolución.

\*\* Si bien la formulación de subtours tiene demasiadas restricciones (si queremos evitar TODOS los subtours) y por eso se agregan únicamente las restricciones de los subtours que se hayan formado (luego de resolverlo sin esas restricciones), la formulación con UI (también llamado modelo MZT del viajante) tiene un poliedro que incluye al de la formulación por subtours, por eso está más alejado de la cápsula convexa entera y desde el punto de vista de resolución es mejor el de subtours.

**C1 Para resolver un problema de Programación Lineal Entera, uno de los procedimientos que se pueden utilizar**

**es Branch & Bound ¿Cómo se puede acelerar la resolución por Branch & Bound para que termine antes?**

La eficiencia de este método depende fundamentalmente del procedimiento de expansión de nodos, o de la estimación de los nodos padres e hijos. Es mejor elegir un método de expansión que provea que no se solapen los subconjuntos para ahorrarnos problemas de duplicación de ramas.

También se pueden agregar restricciones redundantes del modelo original para acelerar el proceso y que el método de Branch & Bound elimine de la evaluación ramas redundantes que no llegarán a ningún resultado.

**C2 El problema de conjuntos a cubrir ¿es un problema difícil?. Definir brevemente el problema e indicar por qué es un problema difícil o por qué no lo es.**

Es un problema NP-Completo (está en la lista de los 21 problemas NP-Completos de Karp).

Dado un conjunto de elementos \{1,2,...,m\} (llamado *universo*) y n conjuntos cuya unión comprende el universo, el set cover problem consiste en identificar el menor número de conjuntos cuya unión aun contiene todos los elementos del universo. Por ejemplo, sea U = \{1, 2, 3, 4, 5\} y los conjuntos S = \{\{1, 2, 3\}, \{2, 4\}, \{3, 4\}, \{4, 5\}\}. Claramente, la unión de todos los conjuntos de S contiene todos los elementos de U. Sin embargo, podemos cubrir todos los elementos con el siguiente conjunto de elementos, con menor número de elementos: \{\{1, 2, 3\}, \{4, 5\}\}.

**C1 ¿Es el problema de Distribución o Transporte un problema difícil? ¿Por qué?.**

**C2 Si en un problema de coloreo de grafos coloreáramos cada nodo con un color diferente la solución es muy**

**fácil de obtener. Entonces, ¿por qué el problema de coloreo de grafos es un problema difícil?**

Porque el problema de coloreo de grafos busca colorear al grafo con la mínima cantidad de colores. Si vamos pintando cada nodo de un color teniendo en cuenta de que no haya adyacentes con el mismo color, estaríamos utilizando una heurística, la cual no necesariamente nos puede llevar al mínimo de colores. Esto termina siendo un problema difícil pues dependiendo del grafo puede haber varias combinaciones posibles y no hay forma de obtener este mínimo de colores con algún método de programación lineal.

**C1 ¿Para qué utiliza el método Branch and Bound la mejor solución entera encontrada hasta el momento?**

**Relacionada: C1 Para resolver un problema de Programación Lineal Entera, uno de los procedimientos que se pueden utilizar es Branch & Bound ¿Cómo se puede acelerar la resolución por Branch & Bound para que termine antes?**

Se la utiliza como cota inferior como condición de corte para otros branches que quedan por explorar. Por ejemplo si encontramos un Z óptimo entero que es 4 y nos queda algún branch por explorar, si este tiene un Z menor al mejor encontrado hasta el momento y la restricción del branch de la variable correspondiente hace que el funcional disminuya, es imposible que de mejor que el que ya encontramos, por ende ese branch se corta. En caso de poder mejorar o si ya en sí el Z es mayor al mejor encontrado, seguimos explorando.

**C2 Si en el problema de la mochila se permitiera colocar fracciones de objetos en lugar de exigir que los objetos se coloquen enteros ¿sería el problema igualmente difícil? ¿por qué?**

El problema es difícil porque la solución continua no me sirve para determinar qué objetos deben entrar o no, en especial con el objeto crítico (el primero que no entra y convendría que entrara). Si el problema fuera continuo la complejidad sería mucho menor, porque la solución continua me sirve (sirve que entre una fracción de un objeto)

**C1 ¿Cuándo se considera que un problema es “difícil”?. Justifique en base a lo visto en las clases teórico prácticas acerca de complejidad.**

A la vez es un problema complejo, si por complejidad nos referimos a la computacional. Un problema se cataloga como inherentemente difícil si su solución requiere de una cantidad significativa de recursos computacionales, sin importar el algoritmo utilizado

**C2 Para resolver un modelo de Programación Lineal Entera ¿podemos usar el método simplex? ¿qué diferencias hay entre resolver un modelo de Programación Lineal Entera y un modelo de Programación Lineal Continua?**

El método simplex no sirve para resolver problemas de programación lineal entera, solo sirve para continua. Se pueden utilizar métodos los cuales utilizan dentro de si al método de simplex, como es el Branch & Bound, que resuelve un problema con restricciones agregadas enteras varias veces hasta obtener una solución óptima entera.

La principal diferencia es el tipo de variable.

**C) Estamos resolviendo un problema de Programación Lineal Entera y no sabemos si es fácil o difícil. Indique cómo encararía el problema aprovechando los diferentes temas vistos en la materia**

**Resolveria por simplex y despues haria branch and bound??**

Primero hay que modelizar el problema en papel, clasificando las variables en continuas, enteras y binarias (subconjunto de enteras). El segundo paso es pasar el modelo a Simplex por ejemplo que es un método de resolución de PLC. Si en PLC el problema da entero, quiere decir que acabamos de resolver la relajación lineal del problema y nos dio entero. Si no da entero, se aplica el método branch and bound o cualquier otro que lo que hacen es buscar la cápsula convexa entera del problema contínuo, hasta dar con una solución. Hallar la cápsula convexa puede tomar mucho tiempo y si así pasa, se puede optar por utilizar una heurística para encontrar una solución aproximada del problema en vez de la solución exacta óptima.

**C1 Si existen procedimientos como Branch and Bound, Branch and Cut y otros, que resuelven de manera exacta un problema de programación lineal entera ¿por qué algunos problemas se resuelven de manera aproximada con un algoritmo heurístico?**

Porque a veces es muy costoso llegar a la solución entera exacta, y realmente bastaría con una buena aproximación de la misma.

**C2 El problema de la mochila ¿es un problema difícil?. Definir brevemente el problema e indicar por qué es un problema difícil o por qué no lo es.**

A la vez es un problema complejo, si por complejidad nos referimos a la computacional. Un problema se cataloga como inherentemente difícil si su solución requiere de una cantidad significativa de recursos computacionales, sin importar el algoritmo utilizado . El problema de la mochila forma parte de una lista histórica de problemas NP Completos elaborada por Richard Karp en 1972 2 . En el caso del problema de la mochila, si contáramos con 4 productos, para saber cuál es la mejor solución podríamos probar las 24=16 posibilidades. El 2 se desprende del hecho de que cada decisión es incluir o no al producto y el 4 de la cantidad de productos. 16 posibilidades es un número manejable, sin embargo, si la cantidad de elementos por ejemplo ascendiera a 20, tendríamos que analizar nada más y nada menos que 220=1048576 posibilidades.

**C1 Los problemas que están en NP ¿son siempre más difíciles que los que están en P?. Responder en términos de complejidad**

Primero que nada, P está incluido en NP, así que algunos problemas que están en P también están en NP. Los problemas que están en NP y no en P (por el momento se cree que es así) son problemas de orden de complejidad exponencial, es decir, la naturaleza (usualmente combinatoria) del problema hace que al agregar un elemento, la cantidad de entradas o restricciones del problema crezca exponencialmente.

**C2 Dentro de los problemas de cobertura de conjuntos hay tres tipos de problema: conjuntos a cubrir, partición de conjuntos y packing. Los problemas de packing siempre tienen solución factible, en cambio los problemas de conjuntos a cubrir y de partición pueden dar incompatible ¿por qué?.**

El problema de packing es buscar cubrir elementos tratando de maximizar la cantidad de elementos sin importar si están cubiertos por más de un conjunto y este problema siempre tiene solución compatible. En cambio el problema de conjuntos a cubrir tiene que cumplirse que si o si se cubran todos los elementos. Este problema se realiza incompatible si existe un elemento que no pertenece a ningún conjunto, entonces será imposible cubrir todos los elementos con conjuntos, porque no existe ningún conjunto que contenga al elemento. El problema de partición es inclusive más restrictivo que la cobertura de conjuntos, porque no sólo pide que deben estar cubiertos todos los elementos, sino que además exige que los elementos deben ser cubiertos únicamente por un conjunto. Por ejemplo si tengo:

, en cobertura es un sistema compatible y la respuesta es c1 y c2, en cambio en partición es incompatible porque para cubrir A, B y C tengo que incluir c1 y c2, pero B se solapa en la solución.

**C1 Además del método simplex, hay varios métodos para resolver problemas de programación lineal continua. Uno de ellos es el método del elipsoide, que no se aplica en la práctica pero tiene un gran valor teórico ¿por qué tiene un gran valor teórico?**

El valor que tiene el método del elipsoide es que demuestra que el problema de la programación lineal continua es un problema de orden polinomial (orden n^100, horrible, pero sigue siendo polinomial). Por lo tanto, se demuestra que PLC pertenece al conjunto de problemas en P.

**C1 En la prueba de heurísticas se suelen utilizar ejemplos de problemas extraídos de bibliotecas de problemas, en muchos casos ya resueltos exactamente. Un ejemplo es el de las bibliotecas TSPLIB en el caso de los problemas del viajante ¿qué ventajas tiene probar con esos casos en lugar de hacerlo con ejemplos elaborados por el autor de la heurística?**

La ventaja que tiene es que uno puede construir una heurística que funciona muy bien para un caso particular, pero no funciona bien en muchas otras instancias del problema. Por eso es que se utilizan bibliotecas de instancias, para que uno pueda probar una heurística contra diversas instancias y no una única. (El clásico problema de overfitting).

**C2 Si tenemos dos ejemplos de dos problemas diferentes ¿cómo podríamos determinar si alguno de los dos es un problema difícil y si uno es más difícil que el otro? ¿qué definición utilizarías para decir que un problema es difícil?**

Para definir si un problema es más difícil que otro se puede utilizar el teorema de la reducción. Supongamos que tenemos dos problemas distintos, uno en NP y otro en P. Si uno está en P, quiere decir que se conoce un algoritmo que resuelve el problema en tiempo polinomial. Si de alguna forma se consigue modelar el problema de NP como un problema de P, quiere decir que lo hemos reducido, es decir, hemos expresado el problema de NP como un problema de P, entonces se dice que el problema de NP no es “difícil” y también pertenece a P.

Si tenemos dos problemas, P1 y P2 ambos de complejidad NP, y logramos reducir el problema P1 y convertirlo en P2 EN TIEMPO POLINOMIAL, se dice que P2 no es más fácil que P1.

**C1 Supongamos que tenemos un problema de coloreo de grafos con 5 nodos. Al resolverlo de manera exacta obtenemos que podemos usar un mínimo de 3 colores para colorearlo. Sin embargo, se obtienen varias soluciones alternativas (una con los colores 1, 2 y 3; otra con los colores 2, 3 y 4, etc.). ¿Qué condiciones agregaríamos al modelo para evitar esas soluciones alternativas?**

Se me ocurren dos opciones:

1. El funcional de coloreo suele ser: siendo Wi cada uno de los colores posibles. Reemplazamos ese funcional por: en este caso estamos ponderando los colores. El color de menor costo será el 1, luego el 2, etc, entonces el modelo utilizará primero W1, luego W2 y así sucesivamente.
2. Agregar restricciones del tipo: entonces W2 no podrá valer 1 hasta que W1 valga 1.

**C1 En la resolución heurística del problema de la mochila, ¿para qué se usan las cotas?.**

Se utilizan cotas para medir que tan buena es nuestra heurística. Si sabemos que el óptimo es X y nuestra heurística nos está dando un resultado de X^4, se sospecha que la heurística no es para nada buena.

**C1 Respecto a complejidad ¿Puede un problema estar en P y en NP simultáneamente? ¿Por qué sí o por qué no?**

**C2 ¿Cómo influye en la resolución de problemas que no se pueda determinar si P es distinto o igual que NP (en términos de complejidad de problemas)?**

El problema es que existen muchos problemas en NP de los cuales no se conoce si existe un mejor algoritmo que solucione el problema en tiempo polinomial. Es decir, no hay certeza si los problemas se pueden resolver en tiempo polinomial y por ello es que existe un premio de 1 millón de dólares para la persona que demuestre que si P = NP o P != NP.

1. Si se encuentra un algoritmo que resuelve, digamos, el problema del viajante en tiempo polinomial, esto demuestra que P = NP, porque lo único que habría que hacer es reducir otros problemas de NP a un viajante y resolverlos en tiempo polinomial.
2. Si se demuestra que no existe ningún algoritmo que resuelva algún problema en NP en tiempo polinomial, habría certeza de que es imposible encontrar una solución en tiempo polinomial y se sabría que la solución exacta es difícil de hallar o que la utilización de heurísticas es útil.

**C2 En el problema de coloreo de grafos, si se resuelve suponiendo que las variables pueden tomar valores no enteros (es decir, se lo resuelve como un problema continuo) ¿se puede usar esa solución como base para resolver el problema en el cual las variables son enteras? Indique las limitaciones de la solución continua (si las tiene).**

No, no tiene ningún sentido la solución obtenida. Tendría algún sentido si la solución dijera que algunos nodos hay que pintarlos de un color específico, pero el problema es que la solución indicará que a los nodos hay que pintarlos un 20% con un color, un 30% con otro color y un 50% con un tercer color. La solución continua no tiene ningún sentido en el problema real.

**C2 El problema de la mochila ¿es un problema difícil?. Definir brevemente el problema e indicar por qué es un problema difícil o por qué no lo es.**

El problema de la mochila consiste de elegir una cantidad de elementos que tienen una capacidad y un beneficio. La mochila tiene una capacidad máxima que pone límite a los elementos a cargar. El problema consiste determinar qué objetos ingresar en la mochila con el objetivo de maximizar el beneficio que otorgan los elementos. Es un problema NP-Completo debido a que para obtener la solución óptima es necesario analizar todas las combinaciones de elementos. Este problema es de programación lineal entera, su relajación lineal carece de sentido ya que uno cargaría la mochila con todos los elementos enteros posibles y el elemento crítico, que es el que entra sólo una fracción, se ingresa esa fracción y listo.