Problema del empaquetamieto de texturas

Integrantes:
Alejandro García González **C411**
Carlos Mauricio Reyes Escudero **C411**

0. Introducción

1. Demostrando que el problema es NP-Duro

A continuación mencionamos la cadena de recucciones que seguiremos para demostrar que nuestro porblema es difícil. La notación $Y \leq_p X$ significa "el problema Y se reduce polinomialmente al problema X".

3-SAT $\leq_p$ 3D-Matching $\leq_p$ 4-Partition $\leq_p$ Rectangle Packing

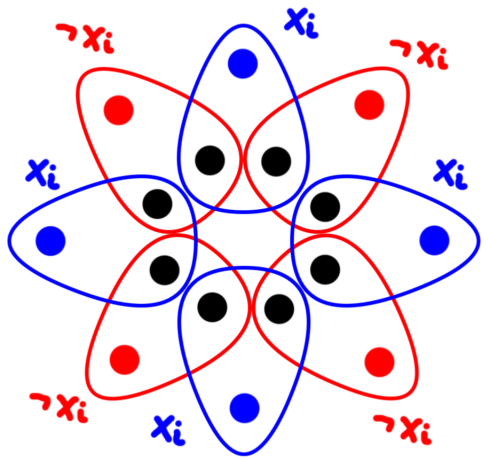
Definición (3D-Matching problem): Dado 3 conjuntos X, Y y Z disjuntos, cada uno de tamaño n, y un conjunto de triplas $T \subseteq X \times Y \times Z$, determinar si existe un $S \subseteq T$ tal que para todo $e \in X \cup Y \cup Z$, e aparece en exactamente una tripla $s \in S$.

Informalmente: Dado una especie de "grafo" donde las "aristas" conectan a 3 nodos entre sí en lugar de solo a 2, y que es tripartito, hallar un "emparejamiento" (entriplamiento?) perfecto.

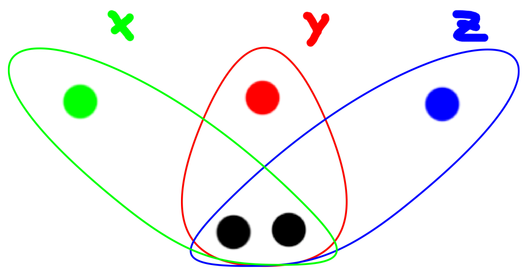
Teorema: 3D-Matching es NP-Completo.
Demostración:

Es fácil ver que dado un conjunto de triplas, se puede comprobar en tiempo polinomial que dicho conjunto da un 3D-Matching completo válido. O sea, 3D-Matching $\in$ NP.

Ahora, para demostrar que es NP-Duro, reduzcamos 3-SAT a 3D-Matching. Dada una formula booleana en forma normal conjuntiva, hagamos lo siguiente:

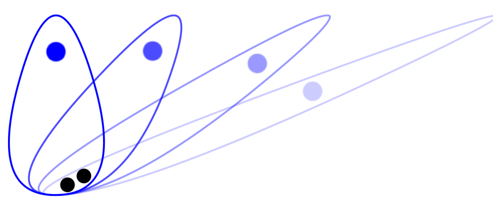


Por cada variable que aparece en la fórmula, construyamos un gadget como el que aparece en la figura. Este tiene el nombre de “gadget variable”. Que 3 nodos estén encerrados dentro de un óvalo significa que están conectados entre sí (o sea, los óvalos son las “aristas” de esta especie de “grafo”). Pero la cantidad de picos del gadget será $2n_{x_i}$, donde n_{x_i} es la cantidad de veces que aparece la variable x_i (o su negación) en la fórmula. Este gadget tiene la peculiaridad de que los nodos que se muestran en negro no están conectados con nada fuera del gadget (se verá que los rojos y azules sí). Esto implica que para lograr un 3D-Matching perfecto, es necesario matchearlos utilizando los óvalos del gadget. Se comprueba entonces que para poder seleccionar a todos los nodos negros, hay solo dos posibilidades: o se eligen solo los óvalos azules o solo los rojos para estar en el matching (cualquier combinación de rojos y azules poseería solapamiento). Esto corresponde a elegir verdadero o falso para la variable x_i. En particular: si se escogen solo los óvalos rojos, o sea que se dejan libres los azules, esto correspondería a elegir verdadero para la variable; y viceversa si se escogen los azules. Esto es así porque dejar nodos libres permite a otros gadgets usarlos, como se verá a continuación.



Por cada cláusula $(x \vee y \vee z)$, hagamos la construcción que se muestra en la figura. Los nodos negros son locales (no están conectados a más nada fuera del gadget), pero los de colores son los exteriores de uno del gadget correspondiente a las variables x, y, z respectivamente (nos aseguramos anteriormente de que cada gadget variable tuvuera suficientes picos para poder hacer esto sin solapamiento). Como en el gadget variable, si se quiere lograr un 3D-Matching perfecto, los nodos negros se tienen que matchear usando alguno de los óvalos del gadget (que por cierto, recibe el nombre de "gadget cláusula"). Esto corresponde a satisfacer esta cláusula usando el valor de una de sus variables. Entonces, satisfacer la fórmula sería equivalente a satisfacer todos los gadget cláusula.

Falta un detalle. Si la operación de dentro de una cláusula fuera un XOR en lugar de un OR, ya estaría la demostración terminada. Pero en una cláusula puede haber más de una variable satisfecha. Así que queda poner un "parche" para cuando esto pasa.



Creamos así un tercer y último gadget, que le llamaremos "gadget recolector de basura". Este consistirá en dos nodos locales (de nuevo los nodos negros de la figura), los que estarán conectados a todos los otros nodos pico de los gadgets variables. De esta forma, los nodos que "sobren" (o sea, que se hayan dejado libre en su gadget variable, y no se hayan usado para satisfacer algún gadget cláusula), se matchearán acá. La idea es usar $\sum n_{x_i} - \#clauses$ de estos gadgets, es decir, asumiendo que se hayan satisfecho todas las cláusulas, uno por cada nodo que se dejó sin matchear.

De esta forma ya tenemos una instancia (de tamaño polinomial) del problema de 3D-Matching que retorna verdadero ssi la fórmula que nos pasaron es satisfacible. ■

Definición (4-Partition problem): Dado un conjunto de n enteros $A = \{a_1, ..., a_n\}$, determinar si existe una partición del conjunto en $n/4$ subconjuntos, de 4 elementos cada uno, con la misma suma (que resulta tener que ser necesariamente $t = \frac{\sum A}{n/4}$).

Teorema: 4-Partition es NP-Completo.

Demostración:

Está claro que si se tiene una partición, comprobar que es válida es fácil: se suman los 4 números de cada subconjunto por separado, y se comprueba que todos den la misma suma. Luego, 4-Partition $\in$ NP.

Ahora, para demostrar que es NP-Duro, reduzcamos 3D-Matching a él. Entonces, dada una entrada al 3D-Matching, hagamos la siguiente construcción:

Vamos a llenar el conjunto que se quiere particionar para resolver el 3D-Matching con el 4-Partition. Procederemos a escribir los números en una especie de representación en base r, para un r lo suficientemente grande (para poder razonar mayormente dígito a dígito). En particular, usaremos $r = 100 \times 3n$, donde n es el tamaño de cada conjunto del 3D-Matching. En lo siguiente, la notación (a, b, c, d, e) representará al número $ar^4 + br^3 + cr^2 + dr + e$. En un momento usaremos "dígitos negativos", lo que justificaremos luego. Ahora, listemos por cada conjunto y tripla del 3D-Matching, qué números añadimos al conjunto a particionar, y cuál sería la suma objetivo (dictada por la estructura del conjunto, claro) y luego justifiquemos por qué resolver el 4-Partition en dicho conjunto es equivalente a resolver el 3D-Matching.

Por cada $x_i \in X \to (10, i, 0, 0, 1)$ y $(11, i, 0, 0, 1) \times (n_{x_i} - 1)$ copias
Por cada $y_j \in Y \to (10, 0, j, 0, 2)$ y $(11, 0, j, 0, 2) \times (n_{y_j} - 1)$ copias
Por cada $z_k \in Z \to (10, 0, 0, k, 4)$ y $(8, 0, 0, k, 4) \times (n_{z_k} - 1)$ copias
Por cada tripla $(x_i,\, y_j,\, z_k) \to (10, -i, -j, -k, 8)$
Suma objetivo (se demostrará en breves) $\to t = (40, 0, 0, 0, 15)$

Ahora a demostrar. Primero, demostremos que la suma es la suma objetivo es la que decimos allá arriba. La suma de cada subconjunto debe ser la suma total sobre un cuarto de la cantidad de elementos del conjunto. La catidad de elementos del conjunto es $4 \times \#triplas$, ya que por cada tripla hay un elemento que la representa, y por cada número de la tripla hay un elemento que lo representa; o sea, cada tripla se cuenta otras 3 veces. La suma total del conjunto se halla sumando dígito a dígito. Esta temina siendo $(40r^4 + 15) \times \#triplas$. Dividiendo, queda demostrado que la suma es t.

Es importante justificar el uso de esos “dígitos negativos”. Si bien en el contexto tienen sentido (no son más que coeficientes en una suma ponderada de potencias de r), en nuestro caso nos conviene que no interfieran con el dígito menos significativo, o sea, que independientemente de los valores de i, j y k, el valor del dígito menos significativo sigue siendo 8. Para esto, basta con que $10r^4 - ir^3 - jr^2 - kr \geq 0$ para los valores máximos de i, j y k, que son en los tres casos n (el tamaño de cada conjunto del 3D-Matching). Esto se logra sustituyendo el valor de r en función de n en esa expresión ($r = 3 \times 10^2 \times n$), sustituyendo i, j y k por su valor máximo n; luego de esto se reduce la expresión a una cuadrática con coeficiente principal positivo y ceros estrictamente menores a 1, lo que demuestra que la inecuación se cumple para todo valor de n, y por tanto para todo valor posible de i, j, k y r.

Ahora la pieza final de la demostración. Como la única forma de formar 15 sumando cuatro potencias de 2 es $8 + 4 + 2 + 1$, y el dígito menos significativo es independiente, entonces para formar un cuarteto de números que sumen a t es necesario tomar uno que haya sido generado por el conjunto X, uno por el conjunto Y, uno por el Z y uno por una tripla. Al sumar un elemento generado por cada uno de X, Y y Z, los dígitos del medio quedan (i, j, k), por lo que hay que sumar el elemento generado por la tipla hay que cancelarlos, lo que implica que hay que sumar el elemento de la tripla (i, j, k). Por último, hay dos maneras de llegar a 40: sumando los 3 elementos únicos generados por cada conjunto (aquellos de dígito más significativo 10) junto a su tripla correspondiente, o sumando los elementos de los que se generan varios por nodo (dígito menos significativo 11, 11, 8, que cumplen $11 + 11 + 8 = 30$) junto a la tripla correspondiente. La primera alternativa corresponde a tomar la tripla (i, j, k) dentro del Matching, la segunda a no tomarla. Entonces, habrá 3D-Matching Perfecto ssi hay una forma de agrupar todos los elementos únicos generados por X, Y y Z, con una tripla correspondiente. ■

Definición (Rectangle Packing problem): Dado un conjunto de rectángulos (con lados alineados a los ejes), y un rectángulo más grande (también con lados alineados a los ejes) cuya área es la suma de las áreas de dichos rectángulos, determinar si es posible cubrir completamente el rectángulo grande con los otros sin solapamiento. No se permite rotar ningún rectángulo.

Teorema: Rectangle Packing es NP-Completo.
Demostración:

Si se tiene la posición de cada rectángulo pequeño, es fácil comprobar en tiempo polinomial que no haya solapamiento (basta con comprobar intersección entre todo par de rectángulos), y si no hay solapamiento, como la suma de sus áreas es igual a la del rectángulo grande, entonces dicho rectángulo está completamente cubierto. Luego, Rectangle Packing $\in$ NP.

Para demostrar que es NP-Duro, reduzcamos 4-Partition a él. Esta reducción es relativamente sencilla: convertimos cada elemento a_i del conjunto que se quiere particionar en un rectangulito de $a_i \times 1$, y ponemos que nuestro rectángulo base sea de $\frac{\sum A}{n/4} \times n/4$. Así cada fila del rectángulo grande representará un subconjunto, habiendo $n/4$ filas. ■