José Alejandro Pérez Márquez 310109800
Tania Michelle Rubí Rojas 315121719
Erick Bernal Márquez 317042522

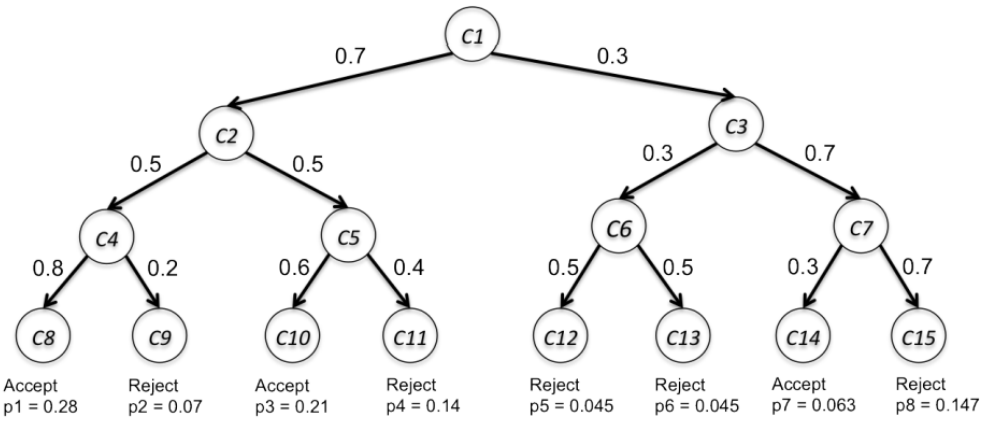
Complejidad en Máquinas de Turing Probabilistas

Introducción

Han ocurrido varios intentos por acercanos a la respuesta a una de las grandes preguntas del milenio, **¿P=NP?** sin embargo, aun no hemos podido afirmar o negar una conclusión. Tales intentos suponen ciertas afirmaciones teoricas para lograr su cometido, en este caso exploraremos una de ellas que utliza Maquina de Turing probabilísticas, cabe aclarar que si bien no resuelve el problema, es un acercamiento a las solciones de los problemas que están en **NP** acercandono, pero no del todo, a la igualdad. Podríamos pensar en ello como un punto medio que emula el no determinismo, conservando en cierta parte el determinismo.

Una Máquina de Turing Probabilística (MTP) no es mas que una máquina de Turing no-determinista modificada para que tenga dos funciones de transición, digamos δ_1 y δ_2, de tal forma que en cada paso durante el computo de una entrada w la MTP elige de manera aleatoria (con una probabilidad de $\frac{1}{2}$) entre aplicar la función δ_1 o δ_2, donde dicha elección es realizada independientemente de las elecciones anteriores. Esto implica que la ejecución de la MTP con una entrada w genera resultados estocásticos, es decir, dicha ejecución puede llevar a que la MTP acepte a w en algunas ejecuciones y la rechace en otras, por lo que podemos tratar a la ejecución de la MTP sobre una cadena w como una variable aleatoria en el conjunto $\{\text{aceptar}, \text{rechazar}\}$. Debido a esta aleatoriedad existen multiples definiciones para la aceptación, en este caso usaremos que tenga un porcentaje de certeza de al menos el 66.6%.

Por ejemplo, el siguiente árbol muestra las rutas de cálculo de una MTP en la que cada rama tiene una probabilidad asociada, (veremos más adelante que no importa la ponderación de estas) dada por alguna de las funciones de transición δ_1 o δ_2 y con algunos caminos de aceptación: p_1, p_3 y p_7 y otros de rechazo: p_2, p_4, p_5, p_6 y p_8.



La configuración inicial C_1 representa la máquina en su estado inicial leyendo el primer símbolo de la cadena de entrada w. La configuración C_2 representa una configuración *posible* después de un paso computacional el cual es elegido con una probabilidad de 0.7. Con probabilidad 0.3 se eligirá la otra elección computacional posible de la configuración inicial y el resultado sería C_3, y así sucesivamente. La probabilidad de que la ruta p_1 acepte a la cadena w es $p_1 = 0.7 \times 0.5 \times 0.8 = 0.28$. Por otro lado, la probabilidad de que la MTP acepte la cadena w es la suma de las probabilidades de los caminos de aceptación, es decir,

$$P(w) = p_1 + p_3 + p_7 = 0.28 + 0.21 + 0.063 = 0.553$$

pero como esto no es igual o superior a $\frac{2}{3}$, entonces la cadena w no será aceptada por nuestra MTP. No obstante hay manera de *"darle la vuelta"* a esto jugando con ciertas propiedades y por lo tanto hacer que acepte a w.

Así, a lo largo de este reporte veremos que la belleza de las MTP's radica en que es posible definir diferentes clases de complejidad de acuerdo a cómo definimos la decibilidad de un lenguaje para una MTP, su relación con **P**, **NP**, y las clases que estan maquina inducen, las cuales son **BPP, RP, cRP** y **ZPP**.

Desarrollo

Aceptación en MTP y la clase BPP

Primero debemos tener en cuenta a los ejemplares del problema que realmente están en el lenguaje ya que de ahí partimos para escribir nuestra defínción de aceptación.

Una vez que sepamos cuales ejemplares en efecto están o no en el lenguaje y pongamos a funcionar nustra maquina con algún ejemplar, cabe preguntarnos *¿cúales son las posibles combinaciones que puede arrojar la máquina?*. Consideramos 4

1. La MTP dice que **SÍ** está en el lenguaje y sabemos que sí.
2. La MTP dice que **NO** cuando en realidad sí está en el lenguaje.
3. La MTP dice que **NO** está en el lenguaje y sabemos que no.
4. La MTP dice que **SÍ** cuando en realidad no está en el lenguaje.

Los incisos 2 y 4 claramente son un error debido a la probabilidad de transiciones que este tipo de maquinas poseen, pudo ser el caso en que una transición nos condujo a una respuesta incorrecta. Así es su naturaleza.

Con esto decimos que una MTP decide un lenguaje si acierta en 2/3 partes de las veces en las que ejecutamos la maquina, es decir, tiene un porcentaje de asertividad del 66 por ciento. Y definimos a la clase **BPP** como la unión de los tiempos $O(T(n))$ en que se tarda una MTP en producir un resultado. Es analogo a la defínción de la clase **P** añadiendo ciertas restricciones como la certeza de la máquina. O formalmente.

Para $T : \mathbb{N} \to \mathbb{N}$ y $L \subseteq \{0,1\}^*$ decimos que una MTP M decide L en tiempo $T(n)$ si para cada $x \in \{0,1\}^*, M$ se detiene en $T(n)$ pasos independientemente de sus elecciones aleatorias, y $p(M(x) = L(x)) \geq \frac{2}{3}$, donde denotamos $L(x) = 1$ si $x \in L$ y $L(x) = 0$ si $x \notin L$.
Así **BPTIME($T(n)$)** denota la clase de lenguajes decicidos por una MTP en tiempo $O(T(n))$ y **BPP** $= \cup_c$ **BPTIME**(n^c).

Tenemos entonces un cuadro como el siguiente

Correct answer \ Answer produced	Yes	No
Yes	≥ 2/3	≤ 1/3
No	≤ 1/3	≥ 2/3

Probabilidades de la clase BPP

¿Por que elegimos $\frac{2}{3}$?
En realidad es más un acuerdo ya que también la podríamos definir con que acierte igual o más al 50% de las veces, siguiendo la idea de cachar el peor caso. Esto suguiere la idea de poder disminuir o aumentar la probabilidad de una MTP y de hecho así es, se logra mediante reducciones de error las cuales nos brindan el siguiente resultado: podemos transformar una máquina de este tipo en una máquina con una probabilidad de éxito muy cercana a uno.

Es aquí donde surge nuestro primer resultado importante, podemos pensar a las Maquinas de Turing clásicas como un caso especial de las MTP, donde ambas transiciones son iguales y por lo tanto tiene un 100% de probabilidad de dar la respuesta correcta, de esta manera es claro que **P** $\in$ **BPP** y bajo ciertas suposiciones tenemos que **P** $=$ **BPP**. Notemos también que **BPP** $\in$ **EXP** ya que podemos enumerar a las distintas posibles combinaciones, las ramas, de la MTP las cuales nos dan justo un número exponencial.

RP, coRP y ZPP

La clase **BPP** captura lo que se conoce como error bilateral, si sabemos que un ejemplar x está en el lenguaje la maquina puede decir que no está, o bien que si nosotros sabemos que x no está en el lenguaje la maquina puede decir que sí está.

Las clases RP y coRP capturan error unilateral, esto es, si sabemos que x no está en el lenguaje no hay manera en que la maquina diga que sí, tenemos la certeza al 100% cuando el ejemplar x no está.

RTIME($t(n)$) contiene cada lenguaje L para los cuales hay una MTP M corriendo en tiempo $t(n)$ tal que

$$x \in L \Rightarrow p(M \text{ acepte } x) \geq \frac{2}{3}$$

$$x \notin L \Rightarrow p(M \text{ acepte } x) = 0$$

Definimos $\mathbf{RP} = \cup_{c>0}\mathbf{RTIME}(n^c)$

La clase **coRP** es análoga, solo que en "la otra dirección", puede regresar 1 cuando $x \notin L$, pero nunca regresará 0 si $x \in L$. Cabe señalar que **RP** $\subseteq$ **NP** ya que cada rama de aceptación es un certificado para los problemas en **NP**, en ese mismo sentido **coRP** $\subseteq$ **RP**. Aunque desconocemos si **BPP** $\subseteq$ **NP**

Tenemos entonces el las siguiente tablas

Correct answer \ Answer produced	Yes	No
Yes	≥ 1/2	≤ 1/2
No	0	1

Probabilidades de la clase RP

Correct answer \ Answer produced	Yes	No
Yes	1	0
No	≤ 1/2	≥ 1/2

Probabilidades de la clase coRP

Las probabilidades son diferentes a la definición, sin embargo, recordemos que podemos aumentar o disminuir estas a través de reducciones de error por lo cual no le daremos mucha importancia a la desigualdad.

Así como tenemos máquinas con error bilateral que definen la clase **BPP** y máquinas con error unilateral que definen las clases **RP** y **coRP** también existen máquinas con "error 0", esa es la idea de la clase **ZPP**

La clase **ZTIME(T(n))** contiene todos los lenguajes L para los cuales hay una máquina en tiempo esperado $O(T(n))$. Esto es

$$x \in L \Rightarrow p(M \text{ acepte } x) = 1$$

$$x \notin L \Rightarrow p(M \text{ se detiene sin aceptar } x) = 1$$

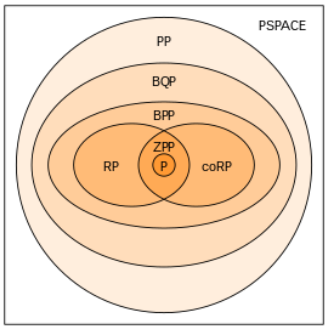
Definimos $\mathbf{ZPP} = \cap_{c>0} \mathbf{ZTIME}(n^c)$.

El siguiente resultado es impresionante debido a que aún no conocemos si $\mathbf{P} = \mathbf{NP} \cap \mathbf{coNP}$, inclusive si $\mathbf{P} = \mathbf{NP}$.

Sabemos que $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$! Y tiene sentido, si unimos una máquina que no tenga error cuando un ejemplar no está en el lenguaje con otra que no lo tenga cuando ejemplar sí está en el lenguaje obtenemos otra máquina capaz de no tener error de algún tipo. Comprendemos las siguientes relaciones

$$\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$$
$$\mathbf{RP} \subseteq \mathbf{BPP}$$
$$\mathbf{coRP} \subseteq \mathbf{BPP}$$

Para una mejor perspectiva de cómo se relacionan estas clases tenemos la siguiente diagrama



Jugando con las Probabilidades

Ambas funciones de transición de una MTP tienen probabilidad del 50% de ser elegidas, si la aumentamos en una de estas como si fueran monedas cargadas ¿Obtenemos un nuevo poder de cómputo? La desencantadora respuesta es que no.

Una moneda con probabilidad $p(Cara) = \pi$ puede ser simulada por una MTP, cuyas monedas tienen igual probabilidad del 50%, en tiempo esperado $O(1)$ siempre y cuando el i-ésimo bit de π sea computable en tiempo polinomial.

La demostración es a nivel de bits por lo cual no nos centraremos mucho en ello. Existe un resultado análogo

Una moneda con $p(Cara) = \frac{1}{2}$ *puede ser simulada por una PTM con transiciones con probabilidad* π *en tiempo* $O(\frac{1}{\pi(1-\pi)})$

Construimos dicha maquina y la ejecutamos, la probabilidad de obtener 2 veces *cara* es π^2, la probabilidad de obtener dos veces *cruz* es $(1-\pi)^2$, obtener *cara* y luego *cruz* tiene una probabilidad de $\pi(1-\pi)$, de obtener *cruz* y luego *cara* es $(1-\pi)\pi$, siguiendo esta idea, la probabilidad de detenernos y provocar una salida en cada paso es de $2\rho(1-\rho)$, condicionado a esto la probabilidad de que caiga cara o cruz es de hecho la misma.

De esta manera concluimos que ambas máquinas poseen el mismo poder cómputo.

Problemas completos para BPP

De manera análoga a cómo definimos los problemas **NP**-completos podemos definir problemas **BPP**-completos, el detalle aquí es que no sabemos en qué lugar dentro de las clases se encuentra. Por ejemplo tratemos con el siguiente lenguaje

$$L = \{\langle M, x\rangle : p(M(x) = 1) \geq \frac{2}{3}\}$$

Este lenguaje es **BPP**-duro, pero no sabemos si está en **BPP**, de hecho no sabemos si está en algún otro conjunto, en alguna otra clase. La dificulta radica en la propia definición de las clases, para la clase **NP** esta está definida *sintácticamente*, esto es dada una cadena w es relativamente sencillo decidir si la cadena pertenece al lenguaje o no. Mientras que para las clases probabilistas la definición es *semántica*, es decir para cadena aceptan al menos dos terceras partes y la rechazan un tercera, por lo cual probar si pertenece al lenguaje como tal es indecidible.

Esto dificulta la defínción de completud en Maquinas de Turing Probabilistas, pues no resulta claro como la podríamos definir sin tener en cuenta la pertenencia al lenguaje.

Teorema de Jerarquía

Si no sabemos de la existencia de problemas **BPP**-completos entonces ¿existe algún teorema de jerarquía?

Por ejemplo $\mathbf{BPTIME}(n^c) \subseteq \mathbf{BPTIME}(n)$ con $c > 1$. Tampoco sabemos la respuesta y ni si quiera hemos podido mostrar si $\mathbf{BPTIME}(n^{log^2 n}) \subseteq \mathbf{BPTIME}(n)$, las razones son parecidas a la anterior.

Conclusiones

Como podemos notar la clase **BPP** tiene mucho que ofrecer y hacer un analisis detallado de todo aquello que contempla junto con las demostraciones requeriría un capítulo completo de un libro dedicado a la Complejidad Compuacional. Mostraremos los más importantes.

1. Una máquina de Turing probabilística es entonces una Máquina de Turing con dos funciones de transición que escoge de manera aleatoria cuál de sus dos funciones de transición aplicar.

2. M decide un lenguaje si para toda x la probabilidad de que M nos dé una respuesta acertada de si x está en L es mayor o igual a $\frac{2}{3}$, aunque este número puede cambiar siempre y cuando sea mayor o igual a $\frac{1}{2}$

3. **BPP** es la clase de problemas de decisión que se deciden por una Máquina de Turing probabilística en tiempo polinomial.

4. $\mathbf{P} \subseteq \mathbf{BPP} \subseteq \mathbf{EXP}$, esto se debe a que podemos enumerar todas las posibilidades que la máquina de turing probabilística puede dar.

5. Si $\mathbf{P} = \mathbf{NP}$ la jerarquía polinomial colapsa a **P** y tenemos que $\mathbf{PH} = \mathbf{P}$ (donde **PH** es la jerarquía polinomial) y dado que **BPP** está en **PH** (que es igual a **P**) entonces ya tendríamos que $\mathbf{BPP} = \mathbf{P}$.

6. **RP, coRP** y **ZPP** son todas sublclases de **BPP**, Ya que **RP** y **coRP** son las subclases correspondientes a los algoritmos probabilísticos que capturan error unilateral, mientras que **ZPP** capturar error 0. (Recordar que **BPP** captura error bilateral).

7. No conocemos ningún problema **BPP**-completo

Existen multiples resultados entre las diversas clases de complejidad tanto de máquina determinas como no deterministas, probabilistas, en espacio y jerarquía, etc. Sin embargo hacer un ánalisis de todas estas relaciones exigiría una enciclopedia entera, por ello decidimos abarcar solo las cuestiones más importantes y una de ellas es el famoso **P** *vs* **NP**. ¿Cómo es que nos ayudan las MTP con este debate? Exploramos ya la parte teórica, ahora sigue la parte práctica.

En realidad las MTP son un intento por acercarse a la implementación del no determinismo, es por así decirlo un punto intermedio entre **P** y **NP**, recordemos $\mathbf{P} \subseteq \mathbf{BPP}$, por lo tanto $\mathbf{P} \subseteq \mathbf{RP}$ y $\mathbf{P} \subseteq \mathbf{coRP}$, a su vez $\mathbf{RP} \subseteq \mathbf{NP}$ y $\mathbf{coRP} \subseteq \mathbf{coNP}$, por lo cual $\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$ mostrando así ese punto intermedio, aunque no sabemos si $\mathbf{NP} = \mathbf{BPP}$.

La máquinas deterministas nos conducen por una sola rama de posibilidades, mientras que las no deterministas tiene varias ramas conduciendonos por todas a la vez, ¡al mismo tiempo!, por eso decimos que una máquina no determinista "adivina" la rama correcta. Las MTP

mantienen la propiedad de tener múltiples ramas reparando en el hecho de elegir un solo camino a la vez, es elegir una sola rama de las múltiples que existen en el no determinismo pudiendo elegir el camino correcto o no, con tal de acertar al menos 2/3 partes de las veces.

Es un acercamiento bastante bueno debido a que conservamos las características más importantes de ambas clases, y de manera práctica nos lleva a la idea de que podemos lograr **P=NP**, una cuestión que contradice lo que planteamos en el principio. Recordemos existen heurísticas y algoritmos de aproximación para poder llevar a cabo los problemas en **NP**, pero esto no quiere decir que las clases sean iguales.

Algunos conjeturan que sí y otras personas que no. Un punto medio parece ser, si bien no la respuesta, aquello que une a estas dos comunidades tan dispares. Haciendo de esto un gran paso hacia la verdadera respuesta aunque no debemos caer en el error de que estamos cerca porque en realidad no sabemos qué tan lejos estamos y puede ser que algún día inventemos otro tipo de máquinas digamos *continuas*, por poner un ejemplo, de hecho ya ha pasado con las computadoras cuánticas e incluso así seguimos sin poder con la pregunta de si **P=NP**. Lo cual nos pone a pensar *¿Estamos cerca?*

Referencias

- Arora, Sanjeev, and Boaz Barak
  Computational complexity: a modern approach
  Cambridge University Press, 2009
- tok.wiki. (s. f.). Máquina probabilística de Turing: Descripción y Definicion formal. `https://hmong.es/wiki/Probabilistic_Turing_machine`
- (s. f). Probabilistic Turing Machines. `https://axon.cs.byu.edu/Dan/252/misc/252-ProbTMs.pdf`
- Imagenes: Tomadas de Google