Min Nim

Eric Brandwein

El archivo `min_nim.py` implementa el algoritmo que calcula, para cada posición posible de un juego dado, si es una N-posición o no. Para esto, la función `calcular_posiciones(elementos, opciones)` devuelve un diccionario con las posiciones posibles en un juego con `elementos = [n1, n2, n3]` y `opciones = [r1, r2, r3]`, y con `True` o `False` según si la posición es una N-posición o no.

Las jugadas posibles son sacar r1, r2, o r3 objetos de la pila 1, 2, o 3, si es que la pila tiene suficientes objetos. Como r1 $<$ r2 $<$ r3, las posiciones terminales, en las que no hay jugadas posibles, serán las que tengan menos de r1 objetos en cada pila.

La función recorre el árbol de posiciones posibles, y para cada posición, calcula si es una N-posición o no. Para esto, se fija si existe una jugada que lleve a una P-posición. Si existe, la posición es una N-posición. Si no existe, la posición es una P-posición.

Notar que esto hace que las posiciones terminales sean P-posiciones. Si quisiésemos modificar el programa para que calcule los valores de las posiciones para la versión en que pierde quien juega último, deberíamos inicializar las posiciones terminales como N-posiciones.

La función también calcula la distancia de cada posición a una posición terminal. Esto es útil para calcular la jugada óptima, y será explicado más adelante.

La complejidad temporal de este algoritmo es $\Theta((n1+1)\cdot(n2+1)\cdot(n3+1))$, ya que la cantidad de posiciones devueltas en el diccionario es $(n1+1)\cdot(n2+1)\cdot(n3+1)$, y el costo de calcular si una posición es una N-posición o no, y calcular la distancia a un terminal, son constantes. Para poder correrlo en un tiempo razonable, se recomienda que $n1, n2, n3$ sean menores a 100.

El archivo también implementa la función `jugada_optima(elementos, opciones)`, que calcula la posición óptima a la que debería llegar el jugador que juegue primero. Si el jugador que empieza tiene estrategia ganadora, es decir, la posición es una N-posición, se devuelve cualquier posición alcanzable que sea una P-posición. Sino, se devuelve la posición que dificulte más la decisión del siguiente jugador.

Para dificultar la decisión lo máximo posible al otro jugador cuando L está en una posición perdedora, se hace la jugada que lleve a una posición más alejada posible de una posición terminal. La distancia de una posición a una posición terminal se calcula de la siguiente manera:

- Si la posición es una N-posición, el jugador actual desea hacer una jugada que lleve a una P-posición. Además, desea llegar lo más rápido posible a una posición terminal, porque así debe hacer la menor cantidad de

cálculos a futuro. Por lo tanto, la distancia es la mínima distancia de las P-posiciones alcanzables más 1.

- Si, en cambio, la posición es una P-posición, ocurre lo contrario. El jugador actual desea que el juego dure lo máximo posible, y entonces la distancia es la máxima distancia de las posiciones alcanzables más 1.

Corriendo el algoritmo

El cálculo de las posiciones se puede ejecutar con el comando `python3 min_nim.py n1 n2 n3 r1 r2 r3`, donde `n1, n2, n3` y `r1, r2, r3` son los parámetros del juego.