Algoritmos y Estructuras de Datos III
Primer Cuatrimestre 2019
Trabajo Práctico 3

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

Darwin en línea

Enunciado

Introducción

En la historia de las ciencias de la computación muchas veces se ha buscado llegar al hito en donde una computadora realiza una tarea con mejor desempeño que un humano. En particular, muchos juegos de mesa han sido protagonistas en estas búsquedas.

En 1996 *Deep Blue*[1] gana su primera partida de ajedrez contra un campeón mundial, y en 1997 gana un match completo. En 2007 el juego estandar de damas en tablero de 8x8 es débilmente resuelto (resultado mucho más fuerte que ganarle a un humano)[2] por el programa *Chinook*. En 2016 *AlphaGo*[3] logra vencer en un match al campeón mundial de go.

En Algoritmos III hemos aprendido sobre técnicas de optimización y no queremos quedarnos atras. En nuestro caso nos interesa hacer un buen jugador del famoso juego *cuatro en línea*. Como es de esperar, el juego *cuatro en línea* ya se encuentra resuelto, por lo que dedicaremos nuestros esfuerzos a una versión más general, el *C en línea*.

Reglas del juego *C en línea*

El juego *C en línea* consiste en un tablero de N columnas y M filas y dos jugadores que cuentan con p fichas cada uno, cada jugador tiene fichas de un mismo color y las fichas de un jugador son de distinto color a las fichas del otro jugador. Los jugadores se alternan en poner las fichas en el tablero. En cada turno un jugador elige una columna para colocar la ficha y esta se ubica en la última casilla desocupada (la menor fila posible). El objetivo del juego es conseguir una línea de fichas del mismo color de tamaño igual a c, el primer jugador en conseguirlo gana. La línea puede ser tanto horizontal, vertical o diagonal, pero debe ser siempre una línea recta. **Nota:** *Cuatro en línea* es una instancia particular de *C en línea* donde $N = 7$, $M = 6$, $p = 21$ y $c = 4$.

Consignas a realizar

1. Soluciones heurísticas:

   a) Implementar un algoritmo de heurística golosa, para esto se pide realizar los siguientes pasos:

1) Implementar una función parametrizable que dado un estado cualquiera de un tablero, la misma determine la próxima jugada a realizar. La complejidad de esta función debe ser como máximo $\mathcal{O}(NM)$.
2) Implementar un jugador que utilice la función del punto $1a1$ como estrategia para elegir en cada paso la jugada a realizar.

2. Optimización de los parámetros:

   a) Utilizar la técnica de *grid-search* de forma de conseguir buenos valores para los parámetros de la heurística propuesta en el punto 1.

   b) Utilizar la técnica de algoritmos genéticos para optimizar los parámetros de la función desarrollada en 1. Para esto definir a los cromosomas como la parametrización anterior o una codificación de la misma e implementar lo siguiente:

      1) Una población (conjunto de cromosomas).
      2) Al menos dos funciones de *fitness*
      3) Al menos una operación de *crossover*
      4) Al menos una operación de mutación
      5) Al menos dos métodos de selección distintos

3. Utilizar los parámetros óptimos obtenidos en el punto 2 y dar dos jugadores (uno con los parámetros de *grid-search* y otro con los del algoritmo genético) que sepan jugar en un tablero de *Cuatro en linea*. Estos jugadores deben ser capaces de vencer a los jugadores de la cátedra de forma consistente (ganarle más del 75 % de los partidos jugando 1000 o más partidos).

4. El día 28 de Junio se llevará a cabo una competencia de *c en linea* para la cual cada grupo compite con jugadores que deberán subministrar para los valores de N, M, c y p mostrados a continuación, la modalidad de la competencia será revelada el mismo día junto con los premios.

   - $N \in [4, \ldots, 20]$.
   - $M \in [N-1, N, N+1]$
   - $p = 2 * N * M$.
   - $c \in [3, \ldots, 8]$ siempre que cumpla $c \leq \text{mín}\{N, M\}$.

Especificaciones de los programas a desarrollar y del entorno a utilizar

Cada jugador es representado por un programa que lee la configuración inicial y las jugadas de su rival por la entrada estándar y escribe sus propias jugadas en la salida estándar.

La cátedra provee un programa que hace de juez (c_linea.py). Este programa puede ser provisto de hasta dos binarios y se encarga de ejecutarlos, comunicarles las jugadas del otro jugador, recibir sus jugadas y coordinar el partido, además, cuenta con la opción de mostrar mediante una interfaz gráfica el estado del juego y mediante archivos de logs. También provee algunos

jugadores sin una estrategia inteligente al solo efecto de facilitar las pruebas y la posibilidad de jugar manualmente contra otra persona, contra uno de los jugadores provistos o contra un binario pasado por parámetro.

La primera línea de la entrada estándar leída por un jugador en cada partido será la configuración inicial en forma de cuatro enteros N, M, p y c separados por espacios. N es la cantidad de columnas, M es la cantidad de filas, p es la cantidad de fichas de cada jugador y c es el tamaño de la línea a formar. Luego sigue una línea que le indica al jugador si le toca empezar o no (*vos*, *el* respectivamente).

A partir de ahí, cada línea leída de la entrada estándar será una jugada del rival y cada línea escrita en la salida estándar será una jugada propia. Cuando el partido termine, el juez enviará algunos de los mensajes *perdiste*, *ganaste* o *empataron* indicando que el partido terminó y el resultado del mismo.

Luego de recibir el resultado de un partido, automaticamente empieza otro partido, esto se repite tantas veces como iteraciones fueron especificadas por parámetro al juez (si este parámetro no está especificado, se juegan infinitos partidos). En caso de que al menos uno de los jugadores sea un humano, cuando un partido termina, hay que hacer click en la pantalla para pasar al siguiente.

Cuando el juez decide terminar, ya sea por que se cumplieron las iteraciones a realizar o porque manualmente se cerró el programa mediante la cruz con el mouse o mediante la tecla *Esc* (en caso de que al menos un jugador sea humano), se envía el mensaje *salir* a los jugadores para que los mismos puedan correctamente terminar sus ejecuciones y cerrar sus recursos (en caso de tenerlos). Una vez realizado esto y antes de terminar, los jugadores deben responder *'listo'* al juez indicando que él también puede terminar.

Nota: Tener cuidado con los mensajes ya que los mismos son *case-sensitive*

Los parámetros del programa c_linea.py están explicados en el mismo programa utilizando el flag –help para visualizar la ayuda donde está detallado como utilizar los parámetros disponibles junto con ejemplos ilustrativos de como correrlo.

Ejemplos de como correr el juez

Dos jugadores humanos jugando al c_linea con la configuraciones de cuatro en linea:

```
python c_linea.py --first azul --ui True --columns 7 --rows 6 --p 21 --c 4
```

Jugador rojo humano y jugador azul bot random de la cátedra jugando al c_linea con la configuraciones de cuatro en linea:

```
python c_linea.py --blue_player ./random_player --first azul
                  --ui True --columns 7 --rows 6 --p 21 --c 4
```

Ejemplo de input/output

A continuación se muestra un ejemplo de una entrada y su posible salida. Dado que la entrada y salida corresponde a un protocolo de comunicación entre el juez y un jugador, los mismos se muestran de forma intercalada para expresar la temporalidad en que cada proceso envía sus mensajes.

Entrada de ejemplo	**Salida esperada de ejemplo**
rojo azul	
1 2	
4 4 3 21	
vos	
	2
0	
	3
0	
	1
ganaste	
rojo azul	
1 2	
4 4 3 21	
vos	
	3
1	
	3
3	
	0
2	
	3
perdiste	
salir	
	listo

Fechas de entrega

- *Formato Electrónico:* Jueves 27 de Junio de 2019, hasta las **23:59 hs**, enviando el trabajo (informe + código) a la dirección `algo3.dc@gmail.com`. El subject del email debe comenzar con el texto `[TP3]` seguido de la lista de apellidos de los alumnos.
- *Formato físico y competencia:* Viernes 28 de Junio de 2019, a las 18 hs.

Importante: El horario es estricto. No se considerarán los correos recibidos después de la hora indicada.

Referencias

[1] Murray Campbell, A.Joseph Hoane, and Feng hsiung Hsu. Deep blue. *Artificial Intelligence*, 134(1):57 – 83, 2002.

[2] Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. Checkers is solved. *Science*, 317(5844):1518–1522, 2007.

[3] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017.