UNIVERSIDAD DE GRANADA

Práctica 3: mancala

Juan Ocaña Valenzuela

3 de junio de 2018

Índice

1. Diario de trabajo

Jueves, 31 de mayo de 2018

Me hubiese gustado empezar esta práctica antes, pero quien me conozca sabe que Java y yo no nos llevamos bien, y estuvo prohibiéndome usar la aplicación del mancala, pese a tener Java 8 instalado.

Además, Ubuntu 18.04 también se unió a la fiesta, actuando de mago y haciendo desaparecer ventanas con la ayuda de su ayudante GTK. Por si fuera poco, la distribución que elegí para reemplazar Ubuntu, Antergos, decidió romper sus repositorios media hora antes de que yo empezase a instalarla, quedándome sin Linux hasta el lunes 28 por la mañana.

Una vez toda esta serie de catastróficas desdichas se solucionó, instalé todo lo que tenía que instalar de nuevo, y me puse a... estudiar AC (tenemos exámenes, es lo que hay). Cuando me cansé de la optimización de código con gcc, decidí que era un buen momento para retar al no tan temible GreedyBot, y ya de paso, ponerme a pensar en cómo desarrollar una heurística curiosa.

Del lunes hasta el miércoles, el mancala se ha adueñado de mi tiempo, de mis ideas, y hasta de mis sueños (esto último no es una exageración), y ha dado como resultado un pseudo-código que describe más o menos lo que quiero que haga mi bot. Bot que, hasta la fecha, no tiene nombre.

Aquí adjunto el resultado de mis cavilaciones:

```
#-------------------------------------------------------------#
#   Ideas heuristica: Mancala                                 #
#-------------------------------------------------------------#

#Idea 1
#-------------------------------------------------------------

func h(pos_actual):
    var h = metidas_granero() - dadas_rival()
    var ultima_pos = pos_actual + pos_actual.semillas

    return h

func metidas_granero():
    if pos_actual.semillas >= pos_actual:
        return 1
    else:
        return 0Las optimizaciones hacen lo que tienen que hacer.

func dadas_rival():
```

```
    if pos_actual.semillas > pos_actual:
        return pos_actual.semillas - pos_actual
    else:
        return 0

#-----------------------------------------------------------------

#Idea 1.1: unificar metidas_granero() y dadas_rival() en valor_jugada()
#-----------------------------------------------------------------

func h(pos_actual):
    var h = valor_jugada()
    var ultima_pos = pos_actual + pos_actual.semillas

    return h

func valor_jugada():
    if pos_actual.semillas >= pos_actual:
        return pos_actual.semillas - pos_actual
    else:
        return 0

#-----------------------------------------------------------------

#Idea 2: anadida comprobacion de poder robar
#-----------------------------------------------------------------

func h(pos_actual):
    var h = valor_jugada()
    var ultima_pos = pos_actual + pos_actual.semillas

    if ultima_pos.es_mia() and not ultimaPos.es_granero():
        if ultima_pos.semillas == 0 and ultima_pos.enfrente().semillas > 0:
            h = h + ultima_pos.enfrente().semillas + 1

    return h

func valor_jugada():
    if pos_actual.semillas >= pos_actual:
        return pos_actual.semillas - pos_actual
    else:
        return 0
```

```
func enfrente():
    return pos(oponente, 7 - pos_actual)

#-----------------------------------------------------------

#Idea 3: anadida comprobacion de si una casilla tiene semillas
#-----------------------------------------------------------

func h(pos_actual):

    var h
    var ultima_pos

    if pos_actual.semillas > 0:
        h = valor_jugada()
        ultima_pos = pos_actual + pos_actual.semillas

        if ultima_pos.es_mia() and not ultimaPos.es_granero():
            if ultima_pos.semillas == 0 and
            ultima_pos.enfrente().semillas > 0:
                h = h + ultima_pos.enfrente().semillas + 1


    else:
        #Valor muy bajo para que nunca lo tome.
        h = -70

    return h

func valor_jugada(pos_actual):
    if pos_actual.semillas >= pos_actual:
        return pos_actual.semillas - pos_actual
    else:
        return 0

func enfrente(pos_actual):
    return pos(oponente, 7 - pos_actual)
#-----------------------------------------------------------

#Idea 4: tiene en cuenta que cuando das mas de seis semillas al rival,
#vuelven a ti (mas vueltas).
#Corregido calculo de valor_jugada(). Se debe restar
#pos_actual - pos_actual.semillas, no al contrario.
```

```
#-------------------------------------------------------------------

const CASILLAS_VUELTA_COMPLETA = 13

func h(pos_actual):

    var h
    var ultima_pos

    #Para que una ficha "salte" del lado contrario al nuestro dando
    #la vuelta, necesitamos tener un numero de semillas mayor al que
    #tiene que recorrer contando el granero,
    #es decir, 6 + pos. Como pueden dar mas de una vuelta, calculamos
    #modulo(semillas, pos+6), y asi vemos cuantas se introducen en el
    #granero, ademas de ver cuantas semillas damos al rival en caso de
    #dar vueltas.
    var vueltas = pos_actual.semillas%(pos_actual+6)


    if pos_actual.semillas > 0:
        h = valor_jugada(pos_actual, vueltas)

        #Calculamos ultima posicion:
        if pos_actual > semillas:
            ultima_pos = posicion_actual - posicion_actual.semillas

        elif pos_actual == pos_relativa(pos_actual, vueltas):
            ultima_pos = tu_granero

        elif (pos_relativa(pos_actual, vueltas) > pos_actual and
        pos_relativa(pos_actual, vueltas) < pos_actual+6:
        #se queda en el otro lado
            ultima_pos = enfrente(semillas -
            vueltas*CASILLAS_VUELTA_COMPLETA)

        else: #se queda en nuestro lado
            ultima_pos = pos_actual - pos_relativa(pos_actual, vueltas)

        #Robo:
        if ultima_pos.es_mia() and not ultimaPos.es_granero():
            if ultima_pos.semillas == 0 and
            ultima_pos.enfrente().semillas > 0:
                h = h + ultima_pos.enfrente().semillas + 1
```

```
    else:
        #Valor muy bajo para que nunca lo tome.
        h = -70

    return h

func valor_jugada(pos_actual, vueltas):
    if pos_actual.semillas >= pos_actual:
        if vueltas > 1:
            return pos_actual - 6*vueltas + vueltas
        else:
            return pos_actual - pos_actual.semillas + 1
    else:
        return 0

func enfrente(pos_actual):
    return pos(oponente, 7 - pos_actual)

#Esta funcion devuelve la posicion relativa a la actual.
#Valores positivos: a la derecha.
#Valores negativos: a la izquierda.
func pos_relativa(pos_actual, vueltas):
    return semillas - vueltas*CASILLAS_VUELTA_COMPLETA)

#-------------------------------------------------------------------
```

Con este puñado de ideas, me dispongo a encajarlas tanto en el sistema proporcionado, como en un algoritmo minimax con poda alfa-beta. Espero salir bien parado.

Sábado, 2 de mayo de 2018

Entre ayer y hoy he conseguido adaptar la heurística al simulador proporcionado, pero el proceso me ha generado alguna que otra duda. La profesora de prácticas ya me comentó en una tutoría que no había contemplado la opción de poder simular movimientos, y al principio lo rechacé, pensando que sería echar por tierra todo el trabajo que había hecho hasta ahora, pero pensándolo en frío, quizás no sea tan mala idea.

De momento, voy a seguir implementando el minimax con poda alfa-beta, manteniendo la heurística intacta. Si funciona, perfecto; si no, consideraré una opción más simple, como la de simular un movimiento y quedarme con el que más semillas meta en mi granero. He oído que ha sido la elección de bastante gente (quién sabe, igual es mejor que mi código, y es mucho menos laborioso...), pero no quiero dejar de intentar mi opción hasta que no vea que sólo estoy perdiendo el tiempo. Ya tuve que dejar de lado mi primera opción en la anterior práctica, y no quiero que vuelva a suceder.
Intentaré dejarlo todo funcionando para esta noche, y dedicar el día de mañana a corregir cosas puntuales, como hacer ganar al bot. Espero de verdad que todo marche bien.

Actualización nocturna: he implementado tanto la heurística como el algoritmo minimax con poda alfa-beta, corregido varios errores, mayormente de *casting* y sintaxis, y únicamente falta implementar el método nextMove para poner el bot en marcha.
Además, ya tiene nombre: **Patrick Dotimas**.