1^{er} Proyecto de Programación Declarativa

Hive

Integrantes:

Richard García De la Osa C412.

Andy A. Castañeda Guerra C412

Estructura:

El proyecto está estructurado en varios scripts. Los scripts con nombre de bichos contienen el comportamiento de cada bicho respectivamente. Luego están **utils.pl** y **dfs.pl** que contienen la mayoría de los metodos auxiliares que se utilizan en los demás asi como los algoritmos desarrollados para moverse por la colmena entre otros. El **board.pl** es el que se encarga de convertir de un formato de celdas a un formato bidimensional y escribir en consola el tablero finalmente en forma de colmena. Tenemos **ia.pl** que es el que contiene el algoritmo desarrollado para la IA. **main.pl** se encarga de manejar la interface del juego y es con la cual deberemos interactuar para utilizar la aplicación.

Funcionamiento:

Las celdas están representadas por un predicado denominado **hex** el cual tiene 7 parametros, Type, Row, Col, Color, Height, OnGame y Blocked, estos representan las propiedades que utilizamos para representar una celda en nuestro tablero. De estos Type es de tipo string y todos los demás son de tipo entero. OnGame alterna entre 0(no está en juego) y 1(está en juego). Blocked representa si la ficha está bloqueada(si acaba de ser movida/puesta o trasladada por el efecto especial del pillbug). Tenemos dos listas que representan a los jugadores, cada una de las listas tienen a 14 bichos los cuales pueden estar o no en juego(existe un predicado para conocer los que están en juego). En cada turno se decide que jugada se va a realizar, si poner una ficha, mover o usar algun movimiento especial. Para la regla *freedom of movement* sobre una pieza en específico, hallamos un adyacente a la misma en la colmena, la quitamos del tablero, y realizamos un dfs sobre el nuevo tablero contando la cantidad de fichas en juego que son alcanzables; al final el resultado de esto debe corresponderse con la cantidad de fichas totales al empezar el algoritmo, menos 1(la que se quitó).

Al intentar mover una ficha, se comprueba de los caminos posibles hacia esa posición, si hay alguno válido(que se cumplan todas las reglas durante el recorrido de el mismo). Además antes de ver si se puede mover una pieza verificamos que la reina de ese color esté en juego. Para colocar una nueva ficha se verifica su validez viendo que ningún adyacente sea del color contrario, que exista algún bicho libre del tipo que se quiere colocar, que la posición en donde se va a colocar no esté ocupada y que sea adyacente a la colmena.

El visual del juego es desarrollado en la consola, se escriben pequeños octágonos para que la información brindada sea mejor, pero el comportamiento de las celdas es de hexágonos como se podrá comprobar durante el desarrollo del juego. Cada celda tiene en su tope dos letras, la primera corresponde a su color(**B** para las negras y **W** las blancas) y la segunda corresponde a la primera letra del nombre en ingles del bicho que representan. Luego en el centro de la celda en los extremos derecho e izquierdo se escriben la posición x, y de la casilla en el tablero. Esto es para mayor claridad a la hora de realizar movimientos; además, se escriben las celdas vacías alrededor de la colmena para que sea más fácil identificar las casillas adyacentes. Dado que el espacio para escribir la información relativa a una ficha es limitado, cuando se trabaja con coordenadas de más de un dígito, escribimos el valor del número módulo 10.

Después de que cada jugador culmine su turno se revisan los adyacentes a cada reina para verificar el estado del juego. Si la reina contraria fue rodeada gana este jugador, si la suya fue rodeada pierde y si ambas son rodeadas a la vez se considera un empate, luego de esto se ejecuta el predicado **abort().** y se asume concluido el juego.

Modos de juego:

El proyecto posee 3 modos de juego, player vs player, player vs ia, ia vs ia. En la interface se explica como acceder a cada uno de estos modos.

Player vs Player: como el nombre indica es para 2 jugadores humanos y se esperaran entradas de ambos para cada jugada.

Player vs IA: un jugador en contra del algoritmo desarrollado, deberá escribir la jugada y luego se le pide confirmar la jugada.

IA vs IA: en este modo solo deberá confirmar cada turno para ver como se desarrolla el juego entre dos instancias del algoritmo.

Las jugadas deberán ser de la forma:

<bug_type> x y Para colocar una ficha de tipo *bug_type* en las coordenadas x y.

x_1 y_1 x_2 y_2 Para mover la ficha que se encuentre en x_1, y_1, hacia x_2, y_2.

P_x P_y x_1 y_1 x_2 y_2 Para utilizar el pillbug que se encuentre en P_x, P_y, para mover la ficha x_1, y_1, hacia el espacio libre x_2, y_2.

pass Para pasar el turno, por ejemplo, cuando no existe ninguna jugada válida.

ia Durante el turno de un jugador humano, para que sea llevado a cabo por el algoritmo minimax.

Consideraciones:

Tomamos como consideraciones:

-Cada ficha al moverse siempre debe estar en contacto con alguna ficha de la colmena: cada paso del camino recorrido debe estar tocando la colmena.

-El hive nunca debe desconectarse o dividirse en dos: cada paso del camino recorrido debe cumplir esta condición, ningún paso dado puede desconectar la colmena.

-El caso de empate solamente consideramos cuando las dos reinas son rodeadas al mismo tiempo. El segundo caso no lo consideramos un empate, aquí queda en manos de los jugadores declarar el empate o no, dado que consideramos que esta situación solo se dará si los jugadores no quieren cometer un mal movimiento.

IA:

La Inteligencia artificial del juego fue hecha utilizando la estrategia de MiniMax. La idea básica es ver todos los movimientos posibles del jugador actual, y quedarse con el mejor. Sin embargo, esto que a simple vista parece buena idea, puede ser un arma de doble filo, porque quizás existe una jugada muy buena para el jugador actual, que en el turno inmediato del oponente le permita hacer una jugada que le de la victoria. De modo que una mejor idea sería hacer Minimax a mayor profundidad, es decir, dados todos los movimientos posibles del jugador actual, calcular todos los movimientos subsiguientes del oponente, y de todos esos pares de movimientos, llevar a cabo el que maximiza el "valor" de nuestra jugada y minimiza el del oponente. Una manera de hacer esto computable, es asignar a cada jugada un valor numérico dados varios parámetros calculables sobre el estado del tablero. Los parámetros que usamos en aras de evaluar un movimiento son la cantidad de fichas que rodean a la reina de ambos jugadores antes y después del movimiento, así como la cantidad de fichas bloqueadas para ambos jugadores. Esto, sin embargo, trajo una dificultad. Poner una ficha, especialmente al inicio del juego, implica bloquear una ficha propia, de modo que bajo esta estrategia la IA preferiría moverse, indefinidamente, antes de poner una segunda ficha. Para corregir esto simplemente le dimos más peso(aunque no mucho más, de hecho, le dimos la menor cantidad efectiva posible) al valor de la evaluación cuando se pone una ficha. Esto conlleva que en muchas ocasiones, a la hora de decidir entre poner una nueva ficha o mover una existente, a menos que el movimiento sea realmente bueno(no bloquee piezas propias, bloquee piezas enemigas, haga al oponente perder su turno, o bloquee más a la reina contraria), se preferirá colocar una nueva pieza. Recalcar además que para estados avanzados del juego, especialmente cuando se tienen muchas hormigas y/o mosquitos adyacentes a hormigas en juego, la cantidad de movimientos posibles crece de manera considerable, y generar todos los pares de movimientos es sumamente costoso.

Finalmente, concluir que la IA hace todo lo que un jugador normal puede hacer, y jugará en su propio favor de ser posible, o al menos en detrimento del jugador oponente. Puede colocar todas las fichas, puede bloquear fichas con el beetle, cargar fichas con el pillbug, y utilizar todo el arsenal del mosquito. La única limitación autoimpuesta por nosotros, por simplicidad, es que su primer movimiento hábil consistirá en poner a la reina en el tablero.