Agente JJA para Geometry Friends - Pista Rectángulo

1st Javier Mahana Palomer
Estudiante de pregrado - Clase: Inteligencia Aritificial
Universidad de Talca
Talca, Chile

2st Joaquín Herrera Briones
Estudiante de pregrado - Clase: Inteligencia Aritificial
Universidad de Talca
Talca, Chile

***Abstract*—Este documento trata sobre el desarrollo de un agente para competir en Geometry Friends - pista rectángulo. Nuestra propuesta se basará en data analizada por un paper que revisa y compara todos los mejores agentes de las competencias hasta el año 2020, en este se indica que hasta ahora el mejor agente para el circuito del rectángulo es "The MARL-GF". Nuestra implementación al igual que el agente de referencia funciona a base de algoritmos A* sub-goal y Q-learning. También explicarán las decisiones y el funcionamiento general de nuestro agente además de posibles modificaciones con el fin de mejorarlo y para finalizar se comparará de forma general con los otros competidores para la sección en que participamos.**

I. Introduction

El equipo participará en la pista de rectángulo de Geometry Friends, el rectángulo tiene las acciones de arrastrarse en el piso y cambiar sus proporciones(ancho y alto) con la condición de que siempre debe mantener su área total. Para la competencia se desarrollará un agente (inteligencia artificial) que tendrá que ser capaz de reunir la mayor cantidad de diamantes bajo el tiempo indicado en cada nivel, por lo que tendrá que ser capaz de reconocer que acciones utilizar y cuando. La competencia consiste de 10 niveles, los primeros 5 son públicos, cuya función es tener un espacio de prueba en la cual podamos corroborar la función, implementación y optimización de nuestro agente, los otros 5 niveles es en donde el agente será realmente puesto a prueba, pues estos niveles son ocultos y ahí se probará su capacidad de solucionar ambientes nuevos.

Para empezar, debemos comprender bien el funcionamiento de la figura, y la creación de funciones y algoritmos que logren completar los primeros 5 niveles. La mayor dificultad se encuentra en el hecho de que hay 5 niveles cuyas estructuras o puzles no conocemos, por lo que el agente debe estar desarrollado hasta un nivel en que sea capaz de adaptarse en tiempo real a cualquier desafío o problema que se presente.

La planificación de este informe comenzará por indicar el estado del arte para la competencia de Rectangle de Geometry Friends, en este espacio se describirán los mejores algoritmos y que es lo que rescatamos de cada uno de ellos. Luego se introducirá nuestra propuesta de agente y su implementación, esta será explicada en grandes rasgos, pero detallando los aspectos fundamentales de su funcionamiento y acompañada por código del agente. A continuación explicaremos errores encontrados en el agente en su actual estado. Finalmente se realizará una conclusión sobre el proceso, el resultado y lo aprendido.

II. Estado del Arte

Destacaremos el uso general de conjunto de algoritmos que han presentado los mejores resultados para nuestro interés, la pista del rectángulo:

- Subgoal A* 2015 [1] : Este acercamiento fue utilizado para crear a un agente que pueda solucionar la pista del rectángulo, ésta aproximación resultó ser muy eficiente pese a lo relativamente sencillo que es. Para este método lo primero que se hace es generar una capa de abstracción del nivel, la cual sea fácil de leer e interpretar por el agente de IA, para crear la capa de abstracción primero se crea una matriz establecido con información que el mismo framework del juego entrega, con esta matriz se crean puntos(que funcionan como nodos), con estos puntos se forma un grafo de nodos conectados del nivel, en donde los nodos son las partes críticas del nivel (esquinas de obstáculos o caídas) y las conexiones se especifican mediante varias matrices de adyacencia, las cuales entregan información valiosa entre dos nodos, como por ejemplo, la acción que se debe realizar para llegar de un nodo a otro,la distancia que existe entre estos y la dirección del movimiento que se debe realizar. Con la capa de abstracción lista, se utiliza el algoritmo A* para crear una ruta que permita llegar a todos los diamantes. Finalmente, con la abstracción del nivel lista y la ruta planeada, el agente es manejado por un sistema de 13 reglas que especifican como debe comportarse. Este método logró ganar la competencia del año 2015 en la pista del rectángulo. Si bien esta es de implementación tiene varios años, es utilizada por la mayoría de agentes que ganaron la competencia en los años siguientes, por lo que sus mejores aspectos serán considerados.
- Reinforcement Learning utilizando Q-learning [2]: Es una de las mejores soluciones hasta ahora para circuito del rectángulo y esta solución adoptó en gran medida lo que el método Subgoal A* presentó, solo que cambia el método que maneja el agente; en vez de utilizar un

sistema de reglas, utiliza tipo de reinforcement learning, Q-learning, este algoritmo funciona en base que va aprendiendo sobre el mundo en el que funciona, para esto utiliza la funcion Q que tiene dos parametros, “s”(estado del mundo) y “alpha”(tasa de aprendizaje), el algoritmo inicia desde s y luego escoge una acción en base a lo que observa del mundo, tras tomar la acción observa un factor “r” recomensa al igual que el nuevo estado del mundo, luego actualiza el valor Q comparando la recompensa máxima para el siguiente estado de mundo con la recompensa obtenida, luego actualiza el estado de mundo y sigue repitiendose hasta llegar a la meta o hasta una determinada cantidad de iteraciones. Por la complejidad de este método, decidimos no adentrarnos mucho en su investigación. Si bien se implementó en el 2015 los mejores agentes hasta el día de hoy lo utilizan como base.

- A* Sub-goal y Q-Learning (The MARL-GF) [4]: Este multi-agente hasta ahora ha sido el que mejor rendimiento ha tenido en la pista de rectángulo y la pista cooperativa, tanto en tiempo de completación como en puntajes, por lo indicado en el análisis de data de todas las competencias realizadas desde el inicio de estas hasta el año 2020 en un paper [3]. Sin embargo, nos enfocaremos en usar sólo lo que esté relacionado y sea necesario para el control del rectángulo. Este agente utiliza como base el Subgoal A* que fue descrito anteriormente, con distancia Euclediana. El sistema que guía al agente tiene tres fases, *entrenamiento*, *mapeo* y *control*. *Entrenamiento* resuelve subproblemas para alcanzar una posición específica en la pista, para esto utiliza Q-Learning. *Mapeo* crea un grafo con nodos establecidos por las esquinas de las plataformas. *Control* utiliza el algoritmo Subgoal A* en el grafo de mapeo para obtener el próximo movimiento y dependiendo de este escoge una acción. El lado negativo de esta aproximación a la competencia es que hay ciertos bugs en el juego que deben ser tomados en consideración al momento de implementar el agente, para alcanzar un rendimiento óptimo.

III. Propuesta e Implementación

Nuestra propuesta se basará en el agente “The MARL-GF”, ya que hasta ahora es el mejor agente en la pista de rectángulo, pero le haremos modificaciones con el fin de mejorarlo. Por esto en nuestro acercamiento al problema presentado por “Geometry Friends” utilizaremos los principales métodos y algoritmos que este agente presenta; tales como el A* subgoal para encontrar la mejor secuencia de movimientos para alcanzar el estado Objetivo, y también el sistema que controla la aceleración y desaceleración vertical del agente, entregada por una Q-table, la cual fue generada anteriormente por un algoritmo de Q-Learning que hizo simulaciones del juego de manera previa al tiempo de ejecución de la IA. De esta manera el grueso de nuestra implementación se basa en el agente “MARL-GF” pero haciendo cambios a detalles de implementación planeamos mejorar el rendimiento de este agente.

Actualmente ya reconocemos varios aspectos que se pueden mejorar de nuestro agente base, estos se encuentran principalmente en el algoritmo de subgoal A*; por ejemplo, en este algoritmo existe un método que evita la ejecución infinita del algoritmo, en el cual si se demora mucho en encontrar una ruta para recolectar todos los diamantes del nivel en un solo recorrido, utiliza un método el cual elimina al diamante con la posición en Y más alta, para así buscar una ruta, pero con un diamante menos. Esto lo cambiaremos para que elimine a el diamante que se encuentra más lejano de la posición inicial, ya que creemos que generará mejores resultados. También hay algunas secciones del código que son rebuscadas y pueden ser simplificadas, tanto funciones como variables con nombres poco representativos, nosotros vamos a corregir esto agregando comentarios que ayuden a entender a cabalidad el código.

A. Implementación:

Para comenzar, este método requiere de muchos parámetros, estos son entregados por el framework del videojuego, hay información para el rectángulo, el círculo y sobre el nivel, usaremos toda la información menos la del círculo al ser este agente sólo para el rectángulo en la pista individual.

Primero se asignan todos los parámetros a sus correspondientes atributos de la clase, entre estos hay una instancia “levelArray” de la clase “LevelArray” cuya función es crear la representación del nivel en una matriz de números enteros, esto se hace con la función “CreateLevelArray” de la clase ’“levelArray”, en esta clase también hay métodos que transforman la posición del array del nivel a puntos y viceversa, ésta transformación a puntos se utiliza mucho después para recorrer el nivel. Cada una de las posiciones del array se utiliza para indicar si en ella hay un obstáculo, un diamante o un espacio abierto.

Segundo se utiliza una instancia de la clase “Platform”, ésta es para establecer todas las plataformas del nivel, las cuales están definidas como una superficie en donde el agente no debe cambiar sus proporciones para desplazarse sobre esta. Hay tres tipos de plataformas: escalera, hueco y horizontales. Estos son de tipo structs y guardan información importante, como su ancho, para saber si es que la plataforma tiene la distancia necesaria para que el rectángulo alcance ciertas velocidades requeridas en la ruta planeada.

Una vez finalizado el SetUp, el código sigue en el ciclo Update del agente. De éste método solo se mostrará el comienzo pues es muy largo.

Dentro del Update, primero se revisa si es que se puede realizar una nueva acción/movimiento pues hay un tiempo de descanso entre acciones que es muy breve, este se revisa en

el primer “if”. Luego se revisa si es que hubo un cambio de plataforma o un cambio en la cantidad de diamantes del nivel, que de ser verdadero significa que el agente esta en un nuevo estado y debe encontrar la mejor manera de proseguir en el nivel. Para esto se utiliza el Sub-Goal A*, el cual se encarga de encontrar la “nextMove” la que especifica que tipo de movimiento debe realizar para alcanzar la próxima plataforma.

Luego se comienzan a revisar casos en los que el rectángulo está sobre una plataforma o si es que está cayendo, para esto se utilizan varios “if/else if/else” revisando distintos parámetros constantes, estos parámetros constantes se obtienen de una instancia de la clase “GameInfo”, entre estas constantes está por ejemplo, el área total del rectángulo, su altura, su velocidad máxima entre otras cosas. Para cada uno de los casos de condiciones del “Update” se asigna una acción correspondiente, que permita seguir recorriendo el path actual. En esta parte del código se utiliza la Q-table en caso de que el movimiento que se desea lograr es uno que se puede definir solo con las acciones para moverse a la izquierda o la derecha.

El proceso se repite en el Update, que va actualizando la ruta cada vez que el agente cambia de plataforma u obtiene un diamante, hasta completar el nivel.

B. Errores Actuales e Hipótesis de Causas:

Actualmente la implementación tiene bastantes errores, como por ejemplo el algoritmo en identifica de manera errónea partes del nivel y entrega acciones incorrectas en algunas plataformas.

Otro problema recurrente de nuestro agente, es que cuando intenta extenderse hacia arriba (morph up) y luego debe extenderse hacia abajo (morph down), el agente se queda enfrascado en un bucle con estas acciones, de manera errática, del cual no puede salir. Aún no sabemos que causa este error, pero utilizaremos break points en distintos niveles que producen este error para encontrar la causa del problema.

Finalmente el último problema recurrente que nos encontramos, es que al principio de algunos niveles el agente simplemente no encuentra un plan de acción y se queda quieto. Esto sucede en un nivel en donde debe escalar una escalera para alcanzar a los diamantes y en otro en donde debe hacer un morph down para pasar por un hueco en una plataforma. Para resolver esto tendremos que revisar bien las lista de movimientos que se preparan para recorrer las rutas en cuyos casos la plataforma inicial sea de tipo escalera(stair) o hueco(gap), creemos que este tipo de plataforma genera problemas debido a que actualmente ambos comparten la misma categoría.

Actualmente el agente es capaz de completar sólo el nivel 3 del juego. En otros niveles logra conseguir tan sólo uno o dos de los diamantes, en los peores casos ni siquiera se mueve.

IV. Conclusiones:

El proceso fue un tanto complejo, a pesar de la simplicidad de las reglas del juego, debido a que fue necesario un buen entendimiento de la forma en la que se leen las pistas y el como debíamos crear una abstracción de esta que le permitiese al agente determinar las acciones más convenientes dependiendo de su posición, estructura de nivel y orden en el que conviene coleccionar los diamantes. Revisar las implementaciones de los agentes que se presentaron en el estado del arte, facilitaron el encontrar una aproximación para representar el mundo y comenzar a implementar el agente.

Si bien el desarrollo de nuestro agente se basó en el mejor agente hasta la fecha para la competencia del rectángulo, el desempeño fue extremadamente bueno o extremadamente malo, Los niveles que se lograron completaron se realizaron en muy buen tiempo y con la cantidad mínima de movimientos necesarias y obteniendo todos los diamantes, por otro lado, en el resto de pistas el agente simplemente era incapaz de decidir que acción tomar o no lograba encontrar una secuencia de acciones que le permitiese obtener todos los diamantes del nivel.

En retrospectiva, para mejorar el agente se tuvo que haber dedicado más tiempo y tener una estructura más clara al momento de implementarlo, comenzar programando para niveles de poca complejidad y luego programar o tomar en cuenta aquellos más difíciles. Por otro lado, nos basamos demasiado en el agente “Marl-f”, el más complejo de todos, tal vez hubiese convenido comenzar con un agente más sencillo e ir mejorándolo con el tiempo. Sin embargo, en los niveles que nuestro agente si logró completar el desempeño fue óptimo, se tomaron todos los diamantes, con buen tiempo y con la cantidad mínima de acciones (morphs y moves) necesarias.

References

[1] Daniel Fischer. Workflow of subgoal a* agent. 2015.

[2] João Quitério, Rui Prada, and Francisco S Melo. A reinforcement learning approach for the circle agent of geometry friends. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 423–430. IEEE, 2015.

[3] Ana Salta, Rui Prada, and Francisco S Melo. A game ai competition to foster collaborative ai research and development. *IEEE Transactions on Games*, 13(4):398–409, 2020.

[4] Ricardo Ari Sequeira. Building a multi-agent learning system for geometry frien, 2019. https://fenix.tecnico.ulisboa.pt/downloadFile/1689244997259768/extended$_abstract.pdf$.