Geometry Friends Competition (CIG 2014) OPU-SCOM

Vallade Benoît and David Alexandre and Tomoharu Nakashima
Department of Computer Science and Intelligent Systems
Osaka Prefecture University
Osaka, Japan
Email: valladeben@cs.osakafu-u.ac.jp
alexandre.david@cs.osakafu-u.ac.jp
tomoharu.nakashima@kis.osakafu-u.ac.jp

I. Introduction

This document describes the particularity of our team's (OPU-SCOM) submission to the geometry friends competition held during the CIG conference. Our submission is targeting the "Rectangle Track" of the competition. Its purpose is to control the behavior of a rectangular shaped agent within a 2 dimensional game. The agent has to collect the objective points spread on the map. To reach its objective it will be able to move on the left and the right as well as modify its height and width (while its area remain the same). Our solution submits an artificial intelligence (AI) composed of two layers. The first one converts the map into a graph, finds the best path between each couple of objective points and finally find the best order between the points. The second layer uses the data calculated previously to generate a sequence of actions usable by the agent.

First, the next section talks about the global concept of our solution. The division in layers and subdivision in modules, the purpose of each one and finally the way they interact with the agent and the game. Then the first layer algorithm will be explained in more detail . Followed by the second layer's description in the fourth section. Finally we conclude by summarizing our explanation.

II. OPU-SCOM's artificial intelligence global concept

This section discusses about the concept of our solution. OPU-SCOM's AI is composed of two layers. The first layer's purpose is to search for a global strategy while the second layer searches for the sequence of actions allowing to complete this strategy.

The first layer's objective is to generate a global strategy to win the game. This global strategy is based on a way to move between the obstacles to travel across the map and reach all the objective points. To accomplish its task, the layer is divided into two modules. The first module is a path finding module based on the possibility search which convert the map given by the game into a graph. Then the Dijkstra algorithm is used to find the best way to join two objective points. This system is used to find all the paths between each couple of objective points and their cost. Then the second module based on an improved version of the particle swarm optimization algorithm (PSO) searches for the best order between the points. The computation of the first layer is entirely executed during the setup step of the level.

The second layer receives the strategy determined by the first layer. Then it computes the list of orders required in order to complete this strategy. These orders are determined by the sequence of nodes to be visited using pre-determined schemes. Then during the game it generates the elementary actions (move and morph) to send to the game according to the previously computed orders

III. First Layer

The first layer is explained in more details in this section. We begin by describing the concept and operation of the first module. Then the procedure followed by the second module are detailed.

A. First module

The first module's purpose is to find the best possibility to reach each couple of points (objectives points and the starting point). A possibility is not a specific path composed of a sequence of tiles as the traditional path finding used to generate but is a global guideline to reach the objective. It does not describe any specific path in particular. However it gives indications on how get around the obstacles. These global guidelines can be considered as an area within which the agent is free to follow any path. The difference between these two notions is represented below in the Fig 2. The specific path is represented in green while the possibilities are the red and blue arrows. We can see that the sequence of tiles, given by the black arrows, describes the displacement step by step whereas the possibilities just suggest to go above or below the obstacle.

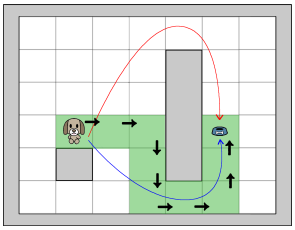

Fig. 1. Representation of the differences between possibility and specific path.

To be able to generate a such possibility, the first step is to simplify the map given by the game. To do so, the search possibility approach use the obstacle which compose the map to draw a set of cells. The cells composing the simplified map are obtained using the two following rules:

- For each objects edge (special grounds and obstacles), a line following the edges direction is added to the grid.
- A line is ended when it hits the maps edge or another objects edge

Then the second step converts the simplified map in a weighted graph. In this graph each node represents a maps cell's edge. And the graphs edges connecting the nodes represent the possibility to travel between two edges (of the map) across one cell. After that, we create two more nodes, one representing the starting position and another the objective position. These nodes are merged to the graph. They are linked to the nodes which represent the edges (maps edges) of the cell where they are positioned.

Fig 3 shows the map after processing the simplification step. The obstacles are represented in gray. It also represents the graph obtained from this simplified map. Each node represents the cell's edge of the same name. And each graph's edge represents the cell used to travel from an edge to another. The corresponding cell's name is written on the graph's edge. The starting and objective nodes are made up of the gray nodes and are merged to the graph by the dotted edges.

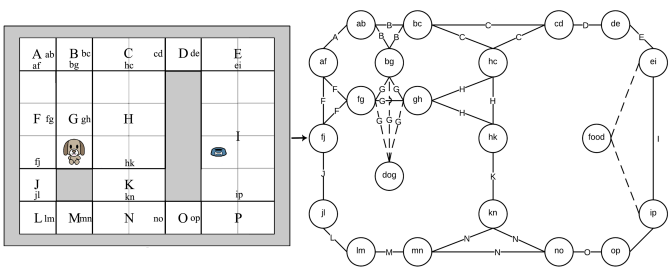

Fig. 2. Simplified map and its graph representation.

The weights between the nodes are calculated using a user defined fitness function.For example, if we search for the shortest path, the fitness function will depend on criterion such as the distance between the cell' s edges (of the map). However, if we want to prioritize a possibility where a smooth path is possible, we will use a fitness function which takes into account the area of the cell and other geometrical criteria. In addition, if the map also include objects which are not obstacles but special areas such as harmful areas or bonus areas we will be able to prioritize or avoid the travel across them by modulating the weight of the corresponding graph's edge.

Finally the last step's objective is to determinate the best possibility. For this we apply the Dijkstra algorithm on the graph. As a result of this operation, we obtain a sequence of nodes which corresponds to a sequence of map's cell's edges and so describes the cells representing the possibility and the order to visit them. This possibility is considered the best based on the fitness function previously defined. The sequence of cells is describing a limited area of the map. which we call the reduced map. Fig 4 represents this reduce map in the situation where the possibility is going above the obstacle. The reduced map is composed of the white areas. The arrows show the sequence of cells composing the possibility.

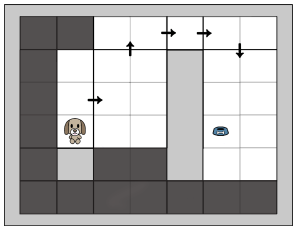

Fig. 3. Reduced map

This step is repeated for each couple of points. At the end of the process we obtain two tables of data. One containing the sequences of nodes for each couple of points. And one for the corresponding fitness values.

B. Second module

The second module uses the data calculated by the first module to choose the best order between the points and to generate the final path. This process is executed using an improved version of the PSO. The PSO is a research algorithm. Its purpose is to tune a set of parameters, which usually take their values in static search spaces. However in our cases, the search space will change during the optimization. Each point can only be visited once so when a point is selected as the value of a variable, the PSO can't give this value to the other variables anymore. That is why we need a PSO algorithm which is able to function on a dynamic search space.

The PSO will find the best combination of values for a set of variables. The available values for the set of variables is composed of the objective points' id and the starting point's id. The first variable is always fixed at the starting point and as soon as the PSO algorithm gives a value to a variable, this one is deleted from the search space. A solution to this problem corresponds to a sequence of objective points to visit. The first parameter's value corresponds to the first objective point to visit, and so on. To calculate the quality of a solution ,the PSO uses the fitness values given by the first module. It will add the value corresponding to the chained couple of points composing the solution.

Finally the global path is generated by appending the possibility given by the first module and corresponding to the chained couple of points composing the solution.

IV. Second layer

This section describes the operating process of OPU-SCOM's AI's second layer. In the next subsection the order generation process will be explained. Then how these orders are translated into elementary tasks. And in the last subsection achievement control will be explained.

A. Order generation

Orders are meta-tasks, they are like sub-objectives to be achieved in order to complete the global strategy. The different types of orders are as follow :

- MoveTo, for simple horizontal movements to a targeted point
- DropAt, for falling through a hole
- Catch, for catching a collectible
- FallOver, for going upstair
- MorphUp and MorphDown, for morphing to a target absolute Y
- Morph, for morphing to a targeted height
- SpeedTo, for simple horizontal movements to a targeted point without stopping
- NoAction, for when the agent should wait

These orders are generated using the strategy given by the first layer before the start of the level. Types are determined by specific sequences of nodes in the global strategy and then the orders are added with their needed parameters in a queue and will be executed once the level starts.

B. Orders processing and tasks generation

Tasks are elementary actions with durations and goals. The four elementary actions are described in Fig 4. Once generated they are put in a queue. Each order correspond to a specific succession of tasks. When a task is finished and the queue is empty an order is dequeued and executed to generate the tasks to be queued. Once a task is finished, its achievement is tested and correction tasks are generated if necessary.

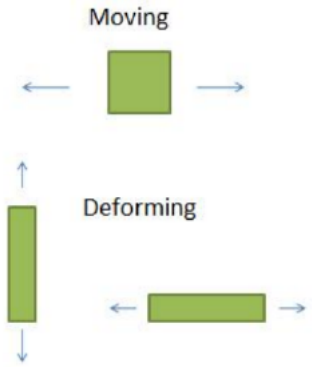

Fig. 4. Representation of the rectangle's actions.

C. Achievement control

When a task is finished its achievement is tested before going to the next task. Has the targeted position been reached ? Has the collectible been caught ? Has the targeted height been reached ? etc... If the achievements criterion are fulfilled we simply go to the next task, else a corrective order is executed and the tasks it generates are inserted in front of the queue in order to be sure that they are executed before the ones of another order. In this achievement control we also check if the agent is not blocked in which case we cancel the current task and go to the next one.

V. Conclusion

To conclude, our solution proposes an agent implementation for the "Rectangle Track" of the Geometry Friends' competition which is composed of two layers, the first one is only executed during the setup process at the beginning of a level. It generates the global strategy by first converting the map to a graph, finds the best path between these points using Dijkstra's algorithm and then it determines the order to capture the points using particle swarm optimization . The second layer then generates, still during the setup, orders to perform in order to achieve the previously computed strategy. And then during the game phase it manages the execution of these orders and the corresponding tasks generated in order to complete the level.