1 Applications

These techniques were developed mainly with pathfinding for circular units in mind, however, they afford benefits to several different aspects of pathfinding and also other applications that warrant discussion. These are discussed in the sections below.

Section 1.1 describes the benefits of using the arc method to avoid "growing" obstacles to deal with non-point units. Section 1.2 discusses several implications of the abstracted graph for pathfinding for multiple units. Section 1.3 explains how searching through the graph has been modified to be an anytime algorithm that is guaranteed to find the optimal path to the goal.

Section 1.4 looks at ways in which the abstracted graph could be used in terrain analysis both in generating and making decisions based on RTS game maps. Finally, Section 1.5 explores how the abstract graph affords improvements in both search time and graph representation size and complexity.

1.1 Elimination of the Minkowski Sum

One advantage obtained by using the arc method of determining the radius of the largest circular unit that can traverse a triangle, it is unnecessary to compute things like the Minkowski Sum of non-point units around the obstacles. This results in several benefits.

First, there are obvious savings by not having to do this computation. Second, especially in the case of circular units which have to be approximated by a many-sided regular polygon, this complicates the triangulation immensely. The number of triangles in the triangulation grows by a factor of approximately the number of sides in such a polygon, which slows down later computations on this triangulation, such as point location and the abstraction algorithm.

Furthermore, the triangles that are added to the triangulation by this operation - particularly in the case of circular units - are often very thin and sliver-like, which add little to the overall description of the space and are generally difficult to deal with. Finally, if there was more than one unit footprint, a separate version of the triangulation would have to be maintained for each possible size and shape of unit. This would be a large overhead, especially as the number of different units increased.

This technique allows application to any radius of circular unit without any overhead, and extends easily to units of other shapes and sizes as well.

1.2 Reduction to Network Flow

Another benefit realized by both the maximum radius determination and the abstraction of the graph is that the problem of moving multiple units can be reduced to a network flow problem. Once the problem is raised up to the

abstract level with only degree-3 nodes between each of which is known the approximate distance and narrowest point, it can be viewed from a higher level.

With this information, a number of possibilities become apparent for sending multiple units to a single destination. One can determine if it is advantageous to send all units along the shortest path to the goal, or if that route is too narrow, perhaps it would be better to select a longer but wider path for all the units. Another possibility would be to split the units between available paths, possibly sending the faster units along longer paths so all the units arrive at roughly the same time, or assigning the faster units to the shortest path in order to get some units to the goal as soon as possible.

Given this extra information of the paths' lengths and narrowest points coupled with the units' speed and size attributes, further dimensions to the group pathfinding problem can be realized. One could send units simply based on the capacities of the edges in the abstract graph as in a network flow problem, or could take the egdes' lengths into account to further benefit. Finally, incorporating the different units' sizes and speeds would add more information to the problem than was possible before.

It would be possible to tailor the path combinations for particular applications such as the speed/cohesion tradeoff. Perhaps early in a game where the locations of enemy forces are largely unknown, it may be beneficial to send all units on the same path, even if it is long or the narrowest point may bottleneck the units. This is because if the units are attacked, they would stand a better chance at survival as a group than if they were split up. Later when the enemy units' positions are known, it may be possible to split up the group into different paths and allow them to reach the goal sooner without such risk of ambush.

There are many uses for the abstract graph and the information it carries, that many possibilities for tuning paths for multiple units should be achievable.

1.3 Anytime Algorithm and Optimal Paths

The method of searching for a path by estimating one moving between midpoints of edges can lead to suboptimal solutions once the shortest path is determined within the triangles of this canonical path. By erring on the side of caution, we can both find the globally optimal solution, and return some solution quickly as part of an anytime algorithm.

When a unit is travelling between two edges of a triangle, its shortest path through that triangle is an arc of radius equal to the radius of the unit, between those edges from the vertex joining them. Thus by recording this angle as the cost of traversing a triangle, the length of the arc can be determined simply by multiplying this value by the radius of the unit.

The search for a path is done using the Euclidean distance between the goal and the closest point to it on the current edge for that unit - at least

that unit's radius from either incident vertex - to form an admissible and consistent heuristic. Because the cost incurred at any point along the search will not be known until a path is determined and the funnel algorithm run, we must estimate it. This is where the underestimation becomes useful.

The cost associated with reaching any state is calculated as the cost associated with reaching its parent plus the larger of: the arc length for that unit through the next triangle, or the difference between the heuristic values of that state and its parent. This means the cost incurred at any state is at most that which would actually be incurred once the entire path has been calculated.

A path is found using these costs, and then the triangles traversed in this path are entered into the funnel algorithm in order to determine actual length of the shortest path within these triangles. The search can now return a (possibly suboptimal) solution. However, search can continue, using branch-and-bound in order to prune paths whose minimum accumulated costs exceed the actual cost of the best path found so far. When more paths are found, the funnel algorithm can be run on them and if their actual cost is less than the best path found so far, this new path replaces it.

Because the accumulated costs are minimums, the optimal path cannot be pruned in this way and the search algorithm will eventually find it. Not only is it beneficial that the algorithm find the optimal solution but the ability to return a solution quickly and improve upon it with time would be especially useful when working within the real-time constraints of a game such as an RTS.

1.4 Terrain Analysis

Another main advantage of the triangulation of an environment and the abstraction thereof with information of the capacities of the different paths is that the information can be used for terrain analysis. This representation allows quick determination of open areas, narrow "choke" points, centers with access to a number of places on the map, others with limited entrances, etc.

Such information is especially useful in an RTS game where one might want to build a base in a large area with limited access points to guard. One might also want to guard structures at the narrowest points along corridors connecting it to other regions, or station units in a place where they will readily have access to several points on the map.

This would also be useful when generating fair and useful maps on which to play games such as RTS. This could be used to ensure that there are enough places on the map suitable for bases, none of which are significantly better than others, and have these places connected somewhat evenly, with a few large or interesting regions in between.

1.5 Graph Simplification

An obvious advantage of the abstraction of the triangulation is that the graph induced by triangles adjacent by an unconstrained edge is significantly simplified. This leads to a number of improvements.

One such improvement is that searching through this abtracted graph requires accessing fewer nodes. If a path does not exist between the start and goal, this is determined instantly instead of by exhausting all triangles in the connected component of either the start or goal node. Then, if there are no "floating" obstacles near the path between the start and goal triangles, this can be determined instantly and the actual path trivially thereafter, whereas a regular search would have to be performed in the absence of this abstraction.

With the abstraction in place, the search between any two points consists of locating their triangles, jumping from those triangles (if they are degree-1) to the closest connected degree-2 node, then (having reached or starting off in degree-2 triangles,) jumping to adjacent degree-3 nodes and searching between degree-3 nodes.

Without such an abstraction, search must traverse individual triangles. This is detrimental because it visits many more search nodes, on the order of the number of triangles in the graph. In contrast, the abstract search visits only decision points (degree-3 abstract nodes), which are linear in the number of "floating" obstacles, irrespective of the number of vertices in the graph.

Another advantage of this abstraction is that the abstract graph is much more representative of the actual decisions faced when finding a path, just deciding to which side of each obstacle to travel, and trusting the best path given this decision can be found. Without the abstraction, one must search for a path when logically, there is really only one. One must also follow long corridors - where really the only choice should be to exit out the other side - or worse, following dead-end branches off such corridors that do not lead to the goal.

Thus the abstraction leads to much faster searches as described above and also at the highest level only includes the actual decision points on the graph, producing a much more compact and meaningful representation.