Search Graph Formation for Minimizing the Complexity of Planning

Alberto Lacaze

Computational Intelligence Laboratory, ECE
University of Maryland, College Park

Stephen Balakirsky

Intelligent Systems Division
National Institute of Standards and Technology

Abstract

A large number of path planning problems are solved by the use of graph based search algorithms. There are a variety of techniques available to optimize the search within these graphs as well as thorough studies of the complexity involved in searching through them. However, little effort has been dedicated to constructing the graphs so that the results of searching will be optimized.

The commonly used approach for the evaluation of complexity assumes that the complexity of a path planner can be evaluated by the number of nodes in the graph. However, in many path planning problems (especially in complex, dynamic environments) the evaluation of the cost of traversing edges is the major culprit of computational complexity. In this paper we will assume that the complexity associated with the computation of cost of traversing an edge is significantly larger than the overhead of searching through the graph. This assumption creates non-trivial complexity results that allows to optimize the creation of the graph based on the computational power available.

We will present a numerical evaluation of several graph creation algorithms including the commonly used four and eight connected grid. Different scenarios for which ground truth is available are explored. Comparison among the graph creation algorithms reveals serious downfalls that are common practice throughout the literature.

1 Introduction

Planning can be defined as the process of finding the steps necessary to bring a system from an initial (current) state to a final (desired) state. Most planning techniques represent the planning problem in a graph $G(V, E)$. Where V is a set of vertices, and E is a binary relation on V [6, 7, 9]. The elements of the set V are called vertices and represent states. The elements of the set E are called edges and represent the ability of the system to move from one state to another. In planning graphs, the edges are ordered or unordered pairs of vertices, (v_i, v_j) where $v_i \in V$ and $v_j \in V$. A walk is an alternating sequence of vertices and edges, a trail is a walk with distinct edges, and a path is a trail with distinct vertices.

When solving a planning problem, we must find a path or plan from a starting vertex v_s to an ending vertex v_e while minimizing a cost function $C = \sum_s^e w_{ij}$ where w_{ij} is the cost of traversing the edge (v_i, v_j). Some planning problems can be solved by algorithms with polynomial complexity. Unfortunately, these tractable set of problems covers only a few of the relevant problems encountered in path planning. Most problems, however, can only be solved by polynomial algorithms on non deterministic machines, ie NP. For a thorough study on the problem of tractability and its taxonomy see [8].

One very useful tool when fighting the computational complexity of planning is the creation of hierarchies of planners. The Real-time Control System (RCS) reference model architecture is one such architecture and it has been successfully applied to multiple diverse systems [1, 3]. The target systems for RCS are in general, complex control problems. Although it has been shown [2, 10] that the complexity of a control problem is reduced by the use of a hierarchical control system, the reduction of error as a function of complexity at one level of the hierarchy has been mostly overlooked.

The complexity of search algorithms inside a graph has been thoroughly studied [11, 13, 14]. However, with few exceptions [4, 12], little attention has been paid on how the graph should be built with some exceptions [4, 12]. In most cases, it is recommended that the graph for search on "empty space" should be built using grids, Voronoi diagrams, or visibility graphs. It

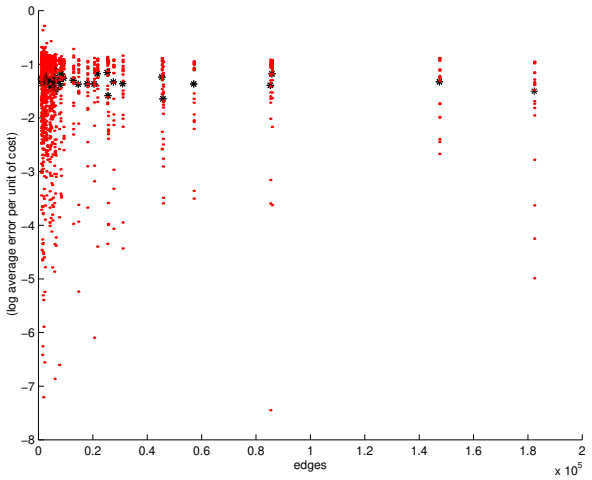

Figure 1: Average error for a 4 connected grid.

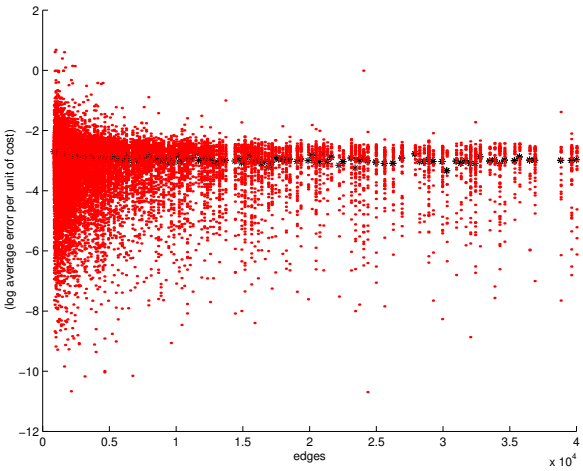

Figure 2: Average error for a 8 connected grid.

is not clear from the literature which of these methods should be used and when. Moreover, in most cases the complexity of algorithms is calculated solely based on the number of vertices in the graph. In most path planning problems, the computational complexity of calculating the cost of the edges is orders of magnitude higher than the actual time spent searching through the graph once these values have been calculated.

2 Numerical Exploration of Graph Creation

In order to compare the different graph formation algorithms, we started by defining a simple test scenario. The analytical closed form evaluation of the complexity of finding the optimum path taking under consideration the placement of the vertices in the solution space becomes easily intractable. Therefore, we decided to study the problem numerically. In the experiments presented in this paper, simple Euclidean distances were used to calculate the cost of traversing the edges. The advantage of using this measure is that we have ground truth. We assumed that the Euclidean distance is calculated with an accuracy of five significant figures.

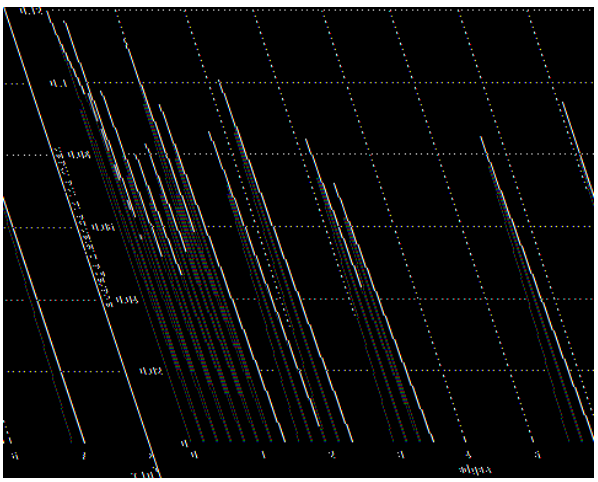

Figure 3: Average distance to the mean.

2.1 Grid Based Graphs

By far, the most commonly used graph for search in planning algorithms is the four-connected square grid. In this kind of graph, the vertices are placed at regular intervals and it is assumed that each vertex is connected to four (or eight) of its closest neighbors.

We built a two dimensional four-connected square grid with a random number of vertices. We repeated this experiment several times. Figure 1 shows $log(error)$ where $error$ is defined as

$$error = abs(d_{s,e} - ((\sum_{vs}^{ve} d_{i,i+1}) + d_{s,v_s} + d_{e,v_e})) \quad (1)$$

s is a randomly selected starting point, e is a randomly selected ending point, v_e is the closest vertex in the graph to e, v_s is the closest vertex in the graph to s, $d(i,j)$ is the Euclidean distance between two points. Please note that this cost function may underestimate the real error of traversing the planned graph as it is assuming that d_{s,v_s} and d_{e,v_e} are Euclidean. This is a best case scenario.

The summation in the equation represents the added cost of the optimal path through the graph. The average error (marked with a black star in the Figure) is kept constant as the number of edges is changed. The different values at a particular number of edges correspond to the different number of times that the experiment was performed using different e and s.

Figure 2 shows the error function shown in 1 applied to a eight-connected grid. As expected, the error function settles at a lower error. By comparing the 4-connected grid to the 8-connected grid we can appreciate that the average error decreases with the higher connectivity, however in both cases, the error quickly settles to a constant value.

Please note that in both cases, increasing the number of edges, and therefore increasing the computational complexity gives us very modest improvements of the final cost. Another problem found experimentally with the 4 and 8 connected grids using this cost function is that there are many paths that have exactly the optimal cost. This has the effect that the optimal path that the algorithm will choose, may wander off the "expected" straight path line from e to s. In other words, many paths within the parallelogram defined by v_s and v_e have exactly the same "optimal" cost. Another effect that results from square grids is that the error varies significantly depending on the direction of travel. A numerical evaluation of this deviation can be appreciated by examining Figure 3. The large average distance to the mean is due to the fact that some s and e happened to be horizontal or vertical, therefore giving small error, while some created a very costly stair-step paths through the graph.

2.2 Shaking the Grid

Some of the pitfalls of the grid based graphs can be avoided by:

1. Shaking the vertices within the grid. In other words, building a square grid, adding a random displacement to the vertices, and finally connecting all the vertices that are within a neighborhood. The size of the neighborhood dictates the vertices to edges ratio. This has two effects:
   (a) Break the ties among optimal paths so that only one path is found to be optimal. This is very helpful in re-planning systems as it forces to commit instead of randomly flipping among the set of "optimal" paths.
   (b) Create a more uniformly distributed set of vertices where all " directionalities" are represented.
2. Create higher connectivity rates (higher than in the 8-connected grid).

Figure 4 through Figure 7 shows the results of a set of experiments run using the above principles. To compute these figures, the vertices of the grid are placed first in a grid pattern where each point is l apart from its closest neighbor. Next, a random vector is added to each vertex of maximum amplitude $3l$. All vertices within a distance threshold are then connected. By varying the connection threshold, different ratios between the number of nodes and the number of edges are achieved. We can see from Figure 4 that the error decreases as the number of edges increases, approaching the 10e-5 mark set by the 5 significant figures used to calculate the Euclidean distances. Figure 5 shows a top view of the same numerically found error. We can see that even a simple Euclidean cost function creates ripple effects in the final cost.

If we take the assumption that the computational complexity is directly proportional to the number of edges (as it is in most cases), we can see in Figure 8 the error function as a function of the number of nodes. The almost counter-intuitive results can be explained from the fact that by increasing the number of vertices the average cost of an edge decreases. In Figure 9 we assumed that we could only calculate the cost of 40000 edges. By visual inspection of Figure 9 we can determine that the least error is given by about 2000 vertices, and therefore creating a graph where each vertex has 20 connected neighbors.

3 Vehicle Planner Example

In order to validate the above rules of thumb, several experiments were conducted using the Demo III

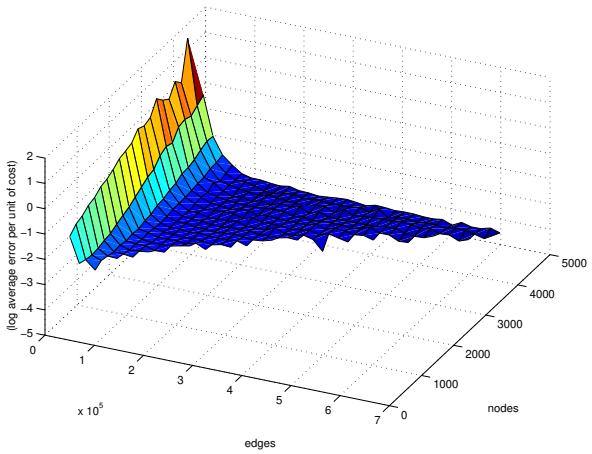

Figure 4: Average error in a shaken grid.

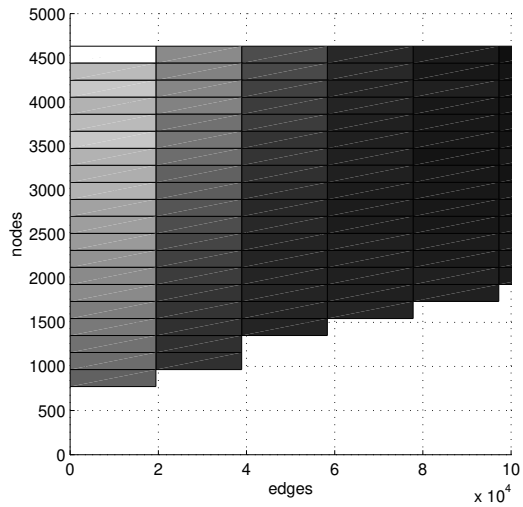

Figure 5: Average error in a shaken grid:top view.

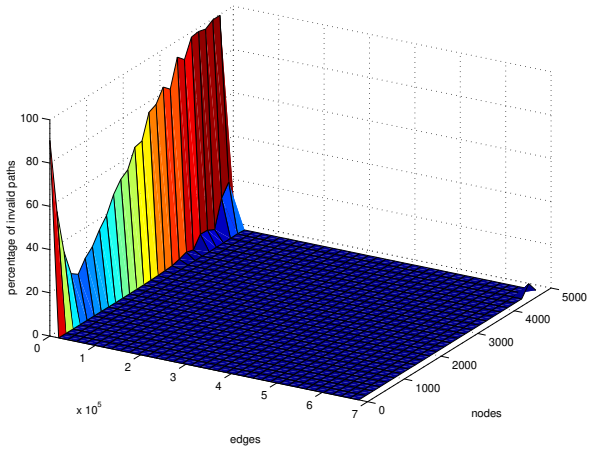

Figure 6: Percentage of failed planning processes in shaken grid.

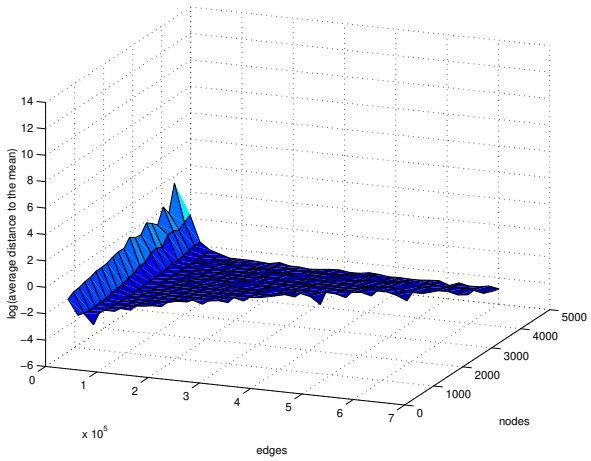

Figure 7: Average distance to the mean in shaken grid.

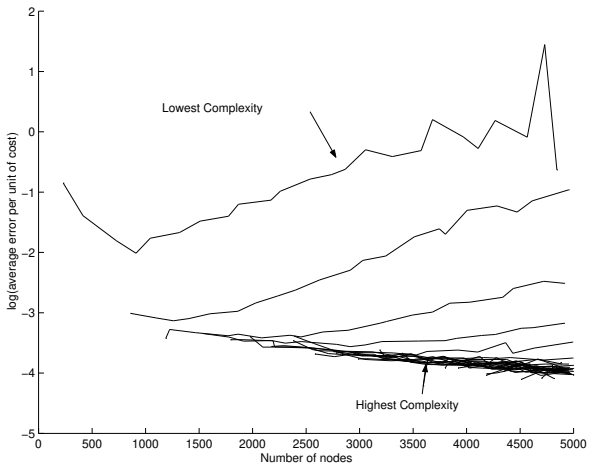

Figure 8: Error for different complexities and varying number of vertices

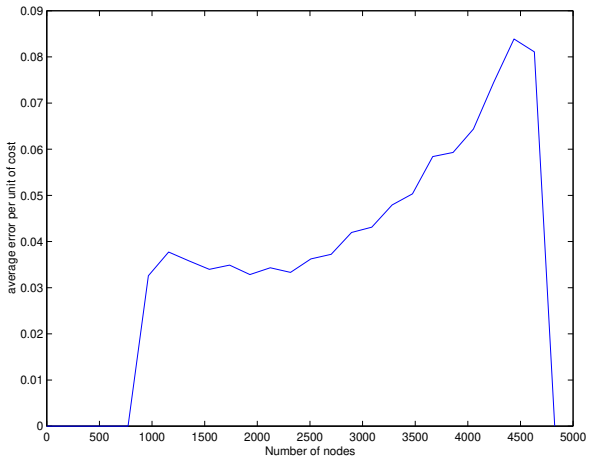

Figure 9: Error for fixed complexity and varying number of vertices

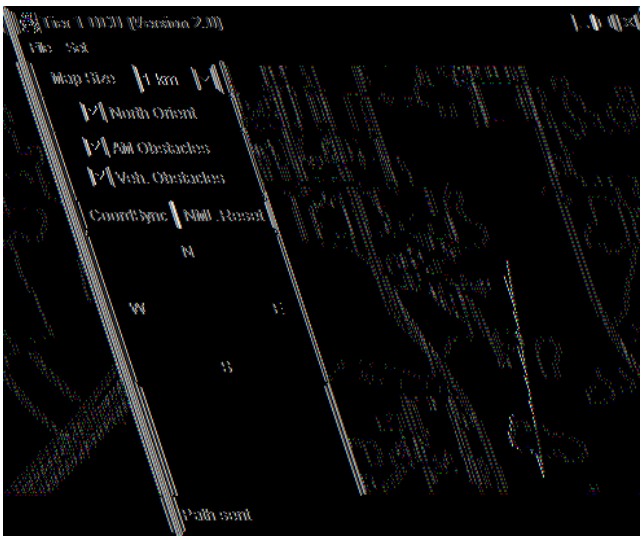

Figure 10: Planning result for complex cost map with four-connected graph.

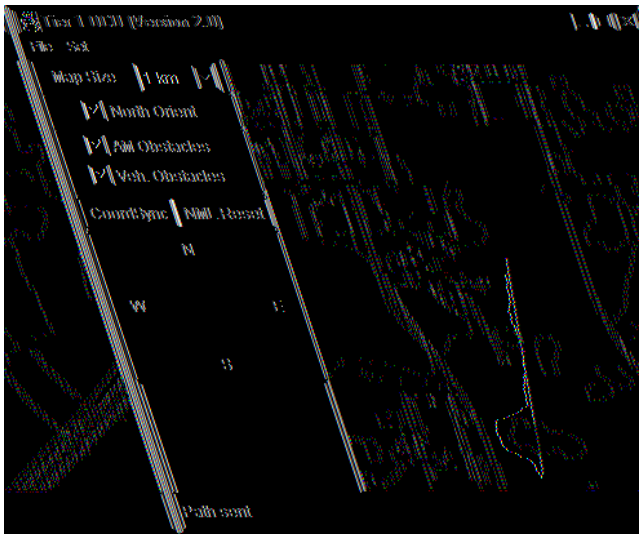

Figure 11: Planning result for complex cost map with many-connected graph.

Vehicle Level Planner [5]. In these experiments, a four-connected graph and a shaken graph of the form of section 2.2 were run using a complex world model and cost function. The four-connected graph had a grid size of 8 meters with 61012 connections and the shaken graph had a grid size of 11 meters with 45086 connections (26% fewer connections) and was shaken ± 5.5 meters. The world model contained a priori information on the NIST grounds at 4 meter resolution including the locations of wooded areas, buildings, roads, and fences. It should be noted that the world model resolution is twice that of the four-connected graph and almost three times that of the highly-connected graph.

In the Demo III Vehicle Level Planner, the planning module passes path segment endpoints (the vertices of the planning graph) to the world model for evaluation. The world model simulates driving a straight line path (the edges of the planning graph) between these end points and returns the cost of traversal to the planner. The planner then conducts an optimal search algorithm to find the cheapest path (in reference to the cost function used by the world model). The cost function used by the world model favored paths that avoided roads and buildings, and drove next to, but not in wooded areas combined with the time of traversal of the route (assumed uniform vehicle velocity over the route segment).

The straight line segments used by the world model may cause plan failures when the resolution of the planning graph is less then that of the world model.

This occurs when a very narrow low-cost corridor is surrounded by a very high cost area. It may occur that there are no straight line segments at the graph resolution that traverse this low-cost corridor. This phenomenon can be avoided in the highly-connected graph by adding additional vertices in these high payoff areas. This approach was not taken in the experiments described below.

Using this planning system, we found that the highly-connected graph performed as much as 27% better then the four-connected graph, even though it used 26% fewer connections. Sample output paths may be seen in Figure 10 for the four-connected graph and Figure 11 for the highly-connected graph. A snap-shot of the world model may be seen as the background of these images. As one would expect, the benefit of using the highly-connected graph is directly tied to the shape of the optimal path. For straight paths, the two graphs performed on par with each other. For paths which required many turns, the highly-connected graph significantly outperformed the four-connected graph.

4 Conclusion

- "Optimal" paths found using the four-connected grid based graph are in general, directionally biased, favoring the traversal of the space in certain directions and not in others. They also create symmetries that result in noncommittal paths. Shaken grids and high connectivity between vertices was shown numerically to improve these pitfalls.
- The number of edges in the graph and their cost evaluation are in most cases, the major culprit for computational complexity. Therefore, it is recommended that the graph design process starts by determining the number of edges that can be evaluated, and then selecting the number of vertices that give the least error.
- Numerical evaluation of the error are in most cases the only way to select parameters for the formation of search graphs in complex environments. Most analytical evaluations of the complexity in the literature make the assumption that the burden of computational complexity is in the "opening" of the vertices in the search graph, and are not readily applicable to planning problems.

References

[1] J. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:473–509, 1991.

[2] J. Albus, A. Meystel, and A. Lacaze. Multiresolutional planning with minimum complexity. *Intelligent System and Semiotics*, 97.

[3] J.S. Albus. *Brain, Behavior, and Robotics.* McGraw-Hill, 1981.

[4] R.S. Alexander and N.C. Rowe. Path planning by optimal-path-map construction for homogenous-cost two-dimensional regions. In *IEEE International Conference on Robotics and Automation - 1990*, 1990.

[5] Stephen Balakirsky and Alberto Lacaze. World modeling and behavior generation for autonomous ground vehicles. In *Proceedings IEEE International Conference on Robotics and Automation*, 2000.

[6] J. Bondy and U. Murty. *Graph Theory with Applications.* North Holland, 1976.

[7] N. Deo. *Graph Theory with Applications to Engineering and Computer Science.* Prentice Hall, 1974.

[8] M. Garey and D. Johnson. *Computers and Intractability.* Freeman, 1979.

[9] F. Harary. *Graph Theory.* Addison Wesley, 1972.

[10] Y. Maximov and A. Meystel. Optimum design of multiresolutional hierarchical control systems. In *Proceedings of IEEE Int'l Symposium on Intelligent Control*, pages 514–520, Glasgow, U.K., 1992.

[11] C.H. Papadimitriou. *Computational Complexity.* Addison Wesley, 1994.

[12] F.P. Preparata and M.I. Shamos. *Computational Geometry, An Introduction.* Springer Verlag, 1988.

[13] J.H. Reif. Complexity of the generalized moving problem. In et al Schwartz, editor, *Planning Geometry and Complexity of Robot Motion.* Ablex Publishing Corporation, 1987.

[14] J.T. Schwartz, M. Sharir, and J. Hopcroft, editors. *Complexity of the Generalized Moving Problem.* Ablex Publishing Corporation, 1987.