ICRA 2000, to appear

Path Planning Using Lazy PRM

Robert Bohlin
Department of Mathematics
Chalmers University of Technology
SE–412 96 Göteborg, Sweden

Lydia E. Kavraki
Department of Computer Science
Rice University
Houston, TX 77005, USA

Abstract

This paper describes a new approach to probabilistic roadmap planners (PRMs). The overall theme of the algorithm, called Lazy PRM, is to minimise the number of collision checks performed during planning. Our algorithm builds a roadmap in the configuration space, whose nodes are the user-defined initial and goal configurations and a number of randomly generated nodes. Neighbouring nodes are connected by edges representing the straight line path between the nodes. In contrast with PRMs, our planner initially assumes that all nodes and edges in the roadmap are collision-free, and searches the roadmap at hand for a shortest path between the initial and the goal node. The nodes and edges along the path are then checked for collision. If a collision with the obstacles occurs, the corresponding nodes and edges are removed from the roadmap. Our planner either finds a new shortest path, or first updates the roadmap with new nodes and edges, and then searches for a shortest path. The above process is repeated until a collision-free path is returned.

Lazy PRM is tailored to efficiently answer single planning queries, but can also be used for multiple queries. Experimental results presented in this paper show that our lazy method is very efficient in practice.

1 Introduction and Motivation

Autonomous path planning addresses the problem of finding collision-free paths for moving objects – robots – among obstacles. In this paper we consider robots operating in workspaces occupied by stationary, completely known obstacles.

The robot in this context may be any moving object, or collection of objects, associated with a certain configuration space. A *configuration* is a set of independent parameters such that the position of every point of the robot can be determined relative to a fixed frame in the workspace. The *configuration space* $\mathcal{C}$, of dimension equal to the number of degrees of freedom (dof) of the robot, is the set of all configurations. The open subset $\mathcal{F} \subset \mathcal{C}$ is the set of collision-free configurations.

The basic path planning problem can be formulated as follows: Given an initial configuration $\boldsymbol{q}_{init}$ and a goal configuration $\boldsymbol{q}_{goal}$ in $\mathcal{F}$, find a continuous curve in $\mathcal{F}$ connecting these points, or determine that none exists [20]. This formulation of the problem is usually favourable, since it is stated in terms of navigating a point, rather than objects in the workspace.

The path planning problem has been extensively studied in the last two decades, and a number of different approaches are proposed. See [10, 14, 20] for overviews. The complexity of certain versions of the problem is proven to be very high. In the case of a robot consisting of polyhedral bodies among polyhedral obstacles, the problem is PSPACE-hard, see [27]. Hence, there is strong evidence that a solution requires time exponential in the number of dof of the robot.

A *complete* algorithm will always find a solution, or determine that none exists. Most complete methods, however, are only applicable to problems in low dimensional configuration spaces, say of dimension three, or less. A complete algorithm, working for arbitrary dimension, was given in [8]. Although the algorithm is exponential in the number of dof, it has the lowest time complexity of all complete algorithms known so far. It is mostly used in theoretical analysis as an upper bound on the complexity of the path planning problem, see [10, 20].

1.1 Single Query Path Planning

Of particular interest in industrial and real-time applications, are planners that without preprocessing can answer single queries very quickly. Such planners are important in applications where the configuration space obstacles sometimes change. This occurs, for instance, when the robot changes tools, grasps an object, or a new obstacle enters the workspace. Ideally, the time required for planning should relate to the difficulty of the planning task, i.e., a simple path in an uncluttered environment should be found quickly, while a more complicated path may require more time.

In a similar way, the planning time should relate to the desired quality of the returned path. The quality of a path is difficult to quantify (see also Section 3.2), but in general we prefer short paths in $\mathcal{C}$, with respect to some metric. The planner should allow the user to intuitively tune short planning time versus high quality.

We would also like the planner to learn to some extent, i.e., to use information from previous queries in order to speed up subsequent queries. For example, if the planner finds a path through a narrow passage in $\mathcal{F}$, it should be able to use that information when searching for a new path back through the passage.

Due to the complexity of the path planning problem, complete planners are far too slow to be useful in practice. Trading completeness for speed, probabilistic techniques have been successfully applied to many problems