Abstraction Search: A Novel Approach to Higher-Dimensional Cost Minimization and Pathfinding

Yonis Mohamed

December 1, 2025

***Abstract*—The curse of dimensionality is a phenomenon that largely affects cost minimization and pathfinding algorithms. For algorithms such as A* and RRT to work properly, their runtimes increase exponentially as the number of dimensions increases. To address this flaw in current procedures, I propose a novel algorithm with a runtime that scales linearly with the increase of dimensions. Abstraction Search abstracts parts of the environment into individual objects that can be defined by a scalar field. Then, it creates a differential equation for each dimension, in which iterative methods are used to solve for a near-optimal solution. (Do data collection in late December to mid January, use numerical values to describe comparisons between AS and A*, use a .cpp file if possible).**

I. Introduction

Developments in optimization are useful in a majority of fields. Fields such as finance, engineering, medicine, and computer science all benefit from breakthroughs in cost minimization and pathfinding. However, the curse of dimensionality leads to super-polynomial runtimes in many modern algorithms, even those specifically designed for higher-dimensional searches. This massively hinders cost minimization for systems with many degrees of freedom, as the runtime of current algorithms lead to acceptable paths not being produced in a reasonable amount of time. To alleviate these issues, I developed Abstraction Search—an algorithm that runs in polynomial time for higher dimensions.

II. Algorithm Instructions

Abstraction Search is an algorithm with five steps.

The first instruction is to abstract segments of the environment into their own scalar fields. The simplest way to do this is to imagine the environment as a sum of n-dimensional superellipses. The equation for simulating these scalar fields, which will be referred to as **objects**, will be discussed later. Let $\mathcal{S}(x_1, \dots, x_n)$ be the sum of all objects for an n-dimensional scalar field $\mathcal{S}$.

The second instruction is to calculate the Lagrangian of the line integral of $\mathcal{S}$ from $(p_1, \dots, p_n)$ to $(q_1, \dots, q_n)$. This can be given by the expression

$$L(x_1(t), \dots) = \mathcal{S}(x_1(t), \dots)\sqrt{\dot{x}_1^2 + \cdots + \dot{x}_n^2} \quad (1)$$

where $x_i(0) = p_i$ and $x_i(1) = q_i$.

The third instriuction is to create a Euler-Lagrange equation for each dimension to solve for the minimal path $x_i(t)$ for $i \in \{\mathbb{Z} \cap [1, n]\}$. This creates n differential equations. The equation for a given i is

$$\frac{d}{dt}\left(\frac{dL}{d\dot{x}_i}\right) = \frac{dL}{dx_i}.$$

Symbolic differentiation libraries should be utilized to find the symbolic expression of each side of the equation.

The fourth instruction is to replace every instance of a higher-order derivative of x_i with the alternative given by Finite Difference Method, creating $n \cdot k$ system of equations, where k is the number of discretized grid points for each path. k should be dependent on the maximum path length in a way that

$$k = \max(|p_i - q_i|) \cdot \omega \quad (2)$$

where ω is the number of grid points per unit and $\omega \in \mathbb{Z}^+$. The fact that k is solely dependent on the distance between two given dots relative to a single axis is crucial to maintaining polynomial runtime.

The fifth instruction is to choose an initial guess for the optimal path. Iteration of the Newton-Raphson method repeatedly applies a re-correction vector to the path until the path approaches a critical point of the line integral functional, whether it be a local minimum, maximum, or a saddle point. Let δ be the number of times the Newton-Raphson method is iterated. For δ to be acceptable, the optimal path or a near-optimal path must be found in a reasonable amount of time. Tuning δ is necessary depending on the complexity of the environment or the accuracy needed.

At this point in the algorithm, a near-optimal/optimal path is found with some confidence. The confidence function can be estimated via

$$\mathcal{C}(\mathcal{S}(x_1, \ldots, x_n)) \approx \frac{\mathcal{A}(\delta, \beta)}{2^{(\beta-\xi)} - (\beta - \xi) - 1} \qquad (3)$$

where β is the number of objects, ξ is the number of overlapping objects, and $\mathcal{A}(\delta, \beta)$ is the average number of objects discovered from Newton-Raphson method out of β total objects after δ iterations. This confidence score is not guaranteed to work for all possible values of $\mathcal{S}$, however it is a decent heuristic for estimating the chances that the optimal path is found. The estimated bounds of $\mathcal{C}$ is $\mathcal{C} \in [0, 1)$ for $\beta \in \mathbb{Z}^+$ and $\delta \geq 0, \delta \in \mathbb{Z}$.

III. Objects

The abstraction of objects is at the heart of the algorithm.