Runtime-Switchable Heuristics in A* for Autonomous Robots

Ignacio Pastore Benaim, David Padilla Orenga

***Abstract*— This document presents a practical exploration of integrating a global planner based on the A* algorithm with runtime-switchable heuristics for ROS-based autonomous navigation. Traditional A* implementations utilize a fixed heuristic, which can limit adaptability across diverse environments. Our approach introduces a flexible framework that allows dynamic heuristic selection (e.g., Manhattan, Euclidean, Chebyshev) during execution without requiring navigation stack restarts.**

We demonstrate the methodology, implementation, and experimental results obtained across various map scenarios. Experiments highlight how adaptive heuristic switching can improve planning times and path quality by tailoring the heuristic to specific map topologies. This innovation ensures robust performance in heterogeneous environments, making it a valuable tool for autonomous robotic navigation.

I. Introduction

In autonomous robotics, global path planning is critical for navigating complex environments. Algorithms such as Dijkstra, A*, and RRT* have become foundational tools, with A* standing out due to its balance between optimality and computational efficiency. Introduced by Hart, Nilsson, and Raphael in their seminal work [1], the A* algorithm combines the advantages of Dijkstra's algorithm and greedy best-first search by using a heuristic function to estimate the cost to the goal, ensuring both optimality and completeness when the heuristic is admissible.

Despite its effectiveness, traditional A* implementations rely on a fixed heuristic function, such as Manhattan (L1 norm) or Euclidean (L2 norm), which can constrain performance in dynamic or heterogeneous environments. Different heuristics have varying strengths; for example, Manhattan is well-suited for grid-based maps with axis-aligned obstacles, while Euclidean provides better results in open spaces. However, a static heuristic limits the algorithm's ability to adapt to changing conditions.

This project addresses these limitations by introducing a runtime-switchable A* planner that enables dynamic heuristic selection during navigation. This feature enhances adaptability and efficiency without requiring the navigation stack to restart, making it particularly suitable for ROS-based systems.

A. Project Motivation and Scope

The primary motivation is to demonstrate the impact of heuristic selection on planning speed and path quality in grid-based navigation. The scope of this work includes developing a ROS plugin that integrates with the 'move_base' framework and supports dynamic heuristic switching. Key components include:

- The ROS Navigation Stack for costmap generation and path execution,
- A custom global planner plugin implementing A*,
- Dynamic Reconfigure for runtime heuristic adjustments.

B. Literature Review

Recent studies underscore the importance of heuristic selection in grid-based path planning. For instance, Manhattan (L1 norm) excels in structured, corridor-like environments, while Euclidean (L2 norm) is more effective in open spaces [2], [3]. Advanced methods such as RRT* [4] provide smooth paths but at a higher computational cost. This work builds on the foundational insights of Hart et al. [1], emphasizing the flexibility of A* through heuristic switching to address these challenges.

C. Outline of Approach

The remainder of this paper is organized as follows:

- Section II details the classical A* algorithm and our modifications for heuristic switching.
- Section III describes the ROS plugin architecture and implementation.
- Section IV presents experimental results on various map types.
- Section V concludes with key findings and future research directions.

II. Method Description

A* [1] is a graph-based path planning algorithm that expands nodes from a start position until reaching the goal. The priority of exploration is determined by:

$$f(n) = g(n) + h(n), \tag{1}$$

where $g(n)$ is the cost from the start to node n and $h(n)$ is a heuristic function estimating the cost from n to the goal.

In a typical 2D grid map:

- **Manhattan:** $h(n) = |x_{goal} - x_n| + |y_{goal} - y_n|$,
- **Euclidean:** $h(n) = \sqrt{(x_{goal} - x_n)^2 + (y_{goal} - y_n)^2}$,
- **Chebyshev:** $h(n) = \max(|x_{goal} - x_n|,\ |y_{goal} - y_n|)$.

A. Proposed Improvement: Runtime-Switchable Heuristic

Rather than fix one heuristic, we introduce an interface that can *toggle* the heuristic type (Manhattan, Euclidean, Chebyshev) at runtime. This is accomplished through a Dynamic Reconfigure server in ROS, enabling quick adaptation to different map characteristics.

III. Implementation

A. System Overview

Our system consists of:

- **Global Planner Node (Custom):** Implements A* with a dynamic parameter for the heuristic.
- **Costmap2DROS:** Provides obstacle/costmap data from sensor inputs.
- **Dynamic Reconfigure Server:** Exposes the parameter `/GlobalPlanner/heuristic_type` for at-runtime switching.

Figure 1 shows a conceptual node/topic diagram (placeholder).

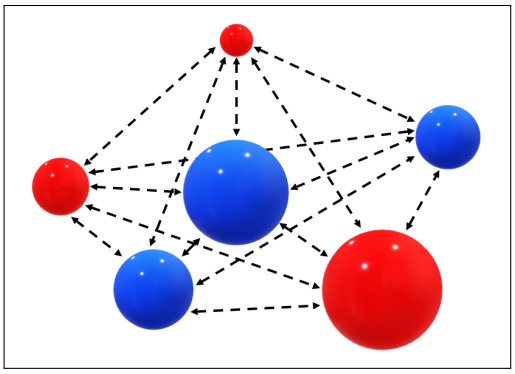

Fig. 1. Conceptual Node Diagram: the custom global planner node with connections to costmap and move_base.

B. Key Modifications

Motivation: Different map structures benefit from different heuristics. **Deviation from Standard A*:** We add a callback in the A* plugin that updates the "heuristic_type" string (or integer) at runtime. **Performance Impact:** If a user chooses a suboptimal heuristic for a large open map, planning might be slower. But if they switch to a more direct Euclidean measure, path length and timing might improve.

Algorithm 1 Pseudo-code for Switchable A* Heuristic

Require: heuristic_type in {Manhattan, Euclidean, Chebyshev}

if heuristic_type == Manhattan **then**
  $h(n) \leftarrow |x_{goal} - x_n| + |y_{goal} - y_n|$
else if heuristic_type == Euclidean **then**
  $h(n) \leftarrow \sqrt{(x_{goal} - x_n)^2 + \dots}$
else if heuristic_type == Chebyshev **then**
  $h(n) \leftarrow \max(|x_{goal} - x_n|, |y_{goal} - y_n|)$
end if
return $f(n) = g(n) + h(n)$

IV. Experimental Results

A. Setup

We evaluated the planner in ROS Noetic, using:

- **Simulator:** Gazebo or Stage
- **Maps:**
  - **Maze-like map** with corridors
  - **Open map** with wide free space
- **Robot model:** Differential drive (e.g., TurtleBot)
- **Metrics:** Planning time, path length, smoothness

We set multiple start-goal pairs and toggled the heuristic at runtime.

B. Results and Analysis

TABLE I
Comparison of Heuristics in Maze Map

Heuristic	Planning Time (s)	Path Length (m)	Smoothness
Manhattan	0.05	5.4	2.1
Euclidean	0.07	5.1	2.2
Chebyshev	0.06	5.2	2.3

Table I shows sample results in a corridor environment. Manhattan has slightly lower planning time, potentially due to alignment with grid directions.

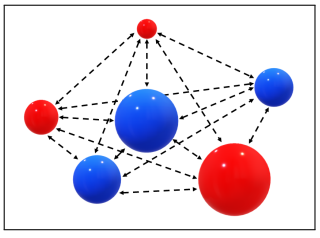

Fig. 2. Visualization of paths in a corridor map. (Top) Manhattan-based route. (Bottom) Euclidean-based route.

C. Open Map

In an open, large arena, Euclidean performed better for overall path length and has comparable planning time to Manhattan. See Table II.

TABLE II

COMPARISON OF HEURISTICS IN OPEN MAP

Heuristic	Time (s)	Path Length (m)	Smoothness
Manhattan	0.08	12.4	3.0
Euclidean	0.08	11.3	2.9
Chebyshev	0.09	11.6	2.8

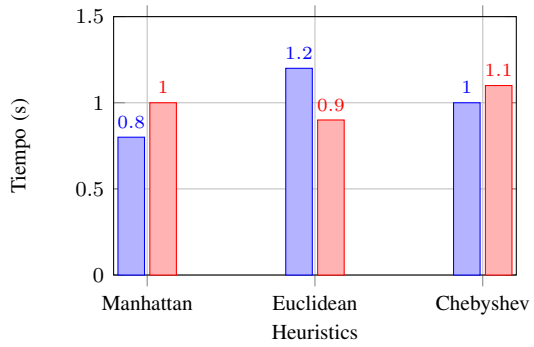

Fig. 3. Comparison of times between heuristics in different maps..

D. Discussion

Changing the heuristic **at runtime** allowed for:

- Adapting to map layout quickly,
- Observing noticeable differences in path shapes/time,
- Overcoming potential suboptimal expansions in corridor-like vs. open spaces.

V. CONCLUSION AND FUTURE WORK

We introduced a **runtime-switchable heuristic A*** planner within the ROS Navigation Stack. Our experiments suggest that no single heuristic dominates across all environments. Instead, toggling heuristics can yield improved performance in specific scenarios (like mazes vs. open maps).

Limitations:

- Only tested in simulation with static obstacles,
- No direct integration with advanced local planners for final path smoothing.

Future Work: We aim to:

- Evaluate real-world experiments on a physical robot,
- Explore automatic heuristic selection based on map analysis,
- Integrate smooth path post-processing or TEB local planner.

REFERENCES

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, MIT Press, 2005.

[3] S. M. LaValle, *Planning Algorithms*, Cambridge University Press, 2006.

[4] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.