Multi-Objective Path Planning for Autonomous Robots Using Reconfigurable Analog VLSI

SCOTT KOZIOL, (Member, IEEE)
Department of Electrical and Computer Engineering, Baylor University, Waco, TX 76798, USA
e-mail: scott_koziol@baylor.edu

ABSTRACT This paper presents a Multi-Objective path planning approach using reconfigurable Analog-Very-Large-Scale-Integrated (AVLSI) circuits. It is significant because it is the first example of floating-gate based analog resistive grid circuits used for Multi-Objective path planning. The two path planning objectives are 1) minimizing path length and 2) minimizing path cost. Three hardware experimental results are presented that implement the approach using a Field Programmable Analog Array (FPAA) circuit. First, an example demonstrates a simple proof-of-concept. Second, an example shows how the FPAA solution compares to an entire solution set for a specific Start and Goal path planning problem. Third, an example shows how the FPAA solution compares to two edge-cases. The edge-cases are the two ideals: ideal lowest cost path, and ideal shortest distance path. Based on these foundational proof-of-concept hardware results, larger environment grids than are currently implementable on the FPAA hardware were simulated to predict performance if a custom FPAA application specific integrated circuit (ASIC) was built for this Multi-Objective path planning purpose. Finally, analysis is presented to address this method’s computational complexity.

INDEX TERMS Path planning, mobile robots, FPAA, analog signal processing, robots, multi-objective.

I. INTRODUCTION

This paper represents the first example of floating-gate-based analog resistive grid circuits used for Multi-Objective path planning. The goal is to plan a path from a mobile robot’s starting point to its goal point and balance the two objectives of 1) minimizing the path length and 2) minimizing the path cost, where cost can be correlated with environment features such as terrain and danger.

Multi-Objective (MO) decision problems are ubiquitous and encountered in diverse fields such as economics [1], health [2], medicine [3], education [4], transportation infrastructure maintenance [5], energy systems [6], and the topic of this paper, path planning for autonomous mobile robots [7]–[9].

Previous Multi-Objective path planning has been accomplished using techniques such as genetic algorithms [10], Pareto fronts [11], A* [12], Multi-Step A* [13], Multi-Objective D* lite [14], Rapidly Exploring Random Tree (RRT) based algorithms [15], [16], Neuromorphic systems [17], and Dijkstra’s algorithm [11], [18]. Multi-Objective algorithms have been applied to planning problems such as routing navy ships for fuel conservation [19], four dimensional Multi-Objective planning for Unmanned Aerial Vehicles (UAVs) [13], and for a multi-objective strategy for search and rescue mobile robots [20].

In this work and in other analog path planning papers, [21]–[26], the robot’s environment is modeled with nodes that represent locations in the physical environment, and edges that represent connecting paths between these nodes, Fig. 1. The connecting edges are modeled with a resistive circuit element [27]. In previous resistive grid path planning work, the resistive elements in the grid often took on one of two values: (1) low resistance to model passable paths or (2) high resistance to model impassable paths such as paths blocked by obstacles [21] or kinematically infeasible paths [28]. A significant aspect of this work is that these path’s resistive elements have been implemented with floating-gate transistors [29], as suggested by Marshall and Tarassenko [21].

This paper expands upon recent work in Single-Objective analog circuit based path planning [24], [29]. Prima facie, one might think that this work is very similar, however, this work is different for this important reason: This work focuses on the...
Key:  = Obstacle

= path cost shown by thickness

**FIGURE 1.** This figure shows an example robot environment where the nodes (black dots) represent locations in the environment. The resistors represent the passable paths between nodes, and the black lines represent obstacles. The varying path costs in this picture are represented with the resistors’ drawn thickness. This figure illustrates the problem addressed in this paper: To plan a path from the start node to the goal node that balances the path length and path cost such that it is close to a Pareto optimal solution.

second of two suggestions made by Marshall and Tarassenko in their early analog path planning paper [21]. Their suggestion is to extend the analog methods for applications with variable costs.

In this new work, variable resistance values represent non-uniform path costs between nodes. This allows one to model an edge path between nodes as traversable, however, penalize it if it represents, for example, dangerous or difficult terrain. This new model presents new system capability. This paper analyzes this capability for its applicability in solving Multi-Objective path planning problems.

The main contributions of this paper are:

- Adding edge resistance variability to the system configuration for representing non-uniform path costs.
- Developing an informed search strategy based on the results of the variable resistance grid.
- An analysis of the utility of non-uniform edge weights for Multi-Objective path planning using analog circuits.

These contributions are useful because it will be shown in Section V that the method may have a favorable time complexity metric when compared to Dijkstra’s digital method. There is a trade-off for this efficient time complexity, however. The trade-off, shown in Section IV, is that the solution may not be a Pareto optimal solution.

This introduction next presents a brief review of Multi-Objective problems and solutions, to include reviewing the concepts of Pareto optimal and Pareto front. It then introduces the topic of this paper, namely, the relevance and method of analog VLSI for Multi-Objective path planning solutions. The structure of the paper is also summarized at the end of the introduction.

**A. BRIEF REVIEW OF MULTI OBJECTIVE PROBLEMS**

At the heart of MO problems, an agent or computing system makes a choice when presented with different options, where each option leads to different, possibly conflicting, results in terms of the optimization criteria. For example: I need to eat calories. I find pleasure in eating sweets like chocolate cake, but I also want to maintain proper nutrition so eating vegetables is an alternate choice. In this example both choices provide the needed calories, however, for me they are conflicting in terms of the optimization criteria. Chocolate cake is optimal with respect to taste satisfaction, and carrots are optimal with respect to nutrition. (The astute reader may suggest carrot-cake as the global optimal choice!)

MO solution methods may be classified into two broad classes: One with a human in the loop, and one without. If there is a human in the decision loop, a good MO solver will present this human decision maker (DM) with a variety of options. Each option will be optimal with respect to one of the objective criteria. This can be visualized using a Pareto graph such as Fig. 2. The example in Fig. 2 presents a Pareto graph for a two-objective problem. Every possible decision option is plotted as a point on this graph (the black dots). For the two objective problem in Fig. 2, a particular decision results in a dot being placed on the graph with its score for objective one as its x-axis coordinate, and its score for objective two as its y-axis coordinate. For this plot, it is assumed that a lower

**FIGURE 2.** Example pareto graph that shows every possible decision (each represented by a dot). It also identifies a Pareto front which represents all of the multi-objective solutions that are presented to the decision maker. The different colored regions represent increasing regions of sub-optimality as the decisions are located inside the green, yellow, and red regions of this plot.
score is more desirable than a higher score. Two important observations can be made from this plot:

Observation 1: A “Pareto Front” (also known as “Pareto Frontier”) can be identified on the graph. This line or front is the MO solution which is presented to the decision maker. The front can be defined using the concept of “Pareto Dominance” [30]. A particular solution \( x_a \) in Fig. 2 dominates another solution \( x_b \) if two conditions hold [31]:

1) Solution \( x_a \) is no worse than \( x_b \) in all objectives
2) Solution \( x_a \) is better than \( x_b \) in at least one objective

As an example, in Fig. 2 decision \( x_1 \) dominates decision \( x_2 \). It is important to realize that a MO solution is a particular set of possible choices. Each black circle in the graph is a member of the entire solution set. A non-dominated solution set is the set of solutions that are not dominated by other members of the solution set [31]. The Pareto-optimal set is the non-dominated set of the entire possible solution set. In Fig. 2 the Pareto-optimal set is designated by the black dots with stars. The Pareto front is the boundary defined by the Pareto-optimal set [31]. In Fig. 2 this is shown by the dashed red line.

Observation 2: In practice, some sub-optimal decisions can be characterized as better than others. Based on a priori domain knowledge, one could potentially group “acceptable” decisions in green, “marginal” decisions in yellow, and “unacceptable” decisions in red. Such example regions are shown in Fig. 2.

When there is a human in the loop, this method can be further subdivided into three classes based on the role of the decision maker (DM) in the solution process [32].

- A priori methods: Human provides preferences before knowing the Pareto optimal solutions
- A posteriori: Human makes a decision at the end, after knowing the Pareto optimal solutions
- Interactive methods: Human iterates with the solution to arrive at the final result

Methods with no human in the loop are called No-preference methods [32]. The analog circuit based solution presented in this paper is a no-preference method. This paper’s no-preference solution will be compared with Pareto optimal solutions later in this paper.

B. MULTI OBJECTIVE ROBOT PATH PLANNING

Robot path planning can be viewed as a MO problem. For example two, possibly competing, objectives for a mobile robot going from a starting point to a goal point are the following:

1) Minimize path length
2) Minimize path cost

With regards to item two above, it is assumed that there are different path segments a mobile robot could take from its start location to its goal location. It is further assumed that the cost associated with each of these path segments is assigned based on aspects such as danger, energy use, or probability of successful traverse. Fig. 3 shows a simple example of such a MO robotics scenario. In this figure, the robot is planning a path from the green star to the red circle. The path labeled ‘B’ is the shortest, however it has rough terrain, so it may not be the best. The right route, path ‘C’, is smooth terrain, but it is the longest path. The left route, path ‘A’, represents a path which avoids both the rough terrain and also avoids the longest route. Path ‘A’ can be seen as the preferable option in some instances. A plot of the entire solution set for this example is trivial as it only has three points, one for each path. An example with a more complex Pareto plot is presented next.

Fig. 4a shows an example two dimensional path planning problem with varied edge weights, \( w_i \). For this example, the robot’s workspace has been discretized into a \( 6 \times 6 \) grid of nodes and connecting edges. The nodes are the dots. Black lines represent obstacles. It is assumed that the robot cannot move diagonally between nodes. The numbers represent the cost weights, \( w_i \), for traversing the edges. An \( n \)-length path from the start to the goal, \( P_i \) in (1), is composed of a set of connected edges.

\[
P_i(j) = \{ e_1, e_2, \ldots, e_n \}
\]

(1)

The total cost of a particular path, \( Cost_i \) in (2), is the sum of the edge weights in a path.

\[
Cost_i = \sum_{j=1}^{\text{length}(P_i)} w_{P_i(j)}
\]

(2)

Fig. 4b shows a Pareto plot for the entire solution set of the example in Fig. 4a. It was generated using a MATLAB program that identifies all forty-two unique, non-cyclical paths [33]. This Multi-Objective path planning problem results in a non-convex Pareto front [7].

If this was a human in the loop MO problem, the competing objectives of this problem would lead the DM to choose a “best” final choice in this problem differently depending on the specific situation. Sometimes, for example, the best route is the shortest route. Sometimes the best route is the longer, but safer, route. A safe route may be a path that

\[ \text{Robot Goal} \]

\[ \text{Robot Start} \]
avoids proximity to obstacles [17]. In another situation, as in Fig. 3, a robot may need to choose between a longer route with smooth, easily passable terrain, and a shorter route with difficult terrain such as sand, gravel, hills, time delays such as toll booths, or some other risk or objective like fuel efficiency [13], [14], [34].

The examples in Fig. 3 and Fig. 4 describe the nature of the Multi-Objective path planning problem addressed in this paper, namely, balancing the competing objectives of path length and path cost.

C. MOTIVATION FOR USING ELECTRONIC CIRCUITS FOR MULTI OBJECTIVE PLANNING PROBLEMS

The time needed to plan a path is a critical metric for autonomous mobile robots. Generating the complete Pareto front can be time consuming and computationally expensive [32]. Evolutionary Multi-Objective optimization algorithms have been used for MO solutions [35], however, these may not even reach the real Pareto optimal set [32].

The motivation for using analog circuits, or hybrid analog-digital systems, for path planning is the potential for better performance metrics in computation times, memory requirements, and lower power than when compared to all-digital solutions [29]. Better performance in these areas provides significant advantages for small mobile robots such as Micro Aerial Vehicles (MAVs), ocean gliders, Unmanned Undersea Vehicles (UUVs), and other robot applications where both the computing resources and power are limited [36], [37].

D. METHOD SUMMARY

The method presented in this paper is a no-preference method. As such, the hardware will not generate a Pareto front or identify the set of Pareto optimal solutions. Instead, it will generate one solution.

This paper’s method of generating this one solution can be described in three broad steps: First, the robot’s environment is mapped onto an analog resistive grid consisting of nodes and edges. Nodes represent discrete points in the environment and edges represent the paths between points. Next, current is input into the grid at the node representing the robot’s location, and current leaves the grid at the node representing the goal. After applying the current, the resistive grid’s nodes will settle to steady state voltages and form a three dimensional potential surface. Finally, node voltages on this surface are measured and these are used along with the a priori edge weights to find the path.

This last step, finding the path, is described as a search problem. Search algorithms are broadly categorized as either uninformed or informed methods [38]. This paper’s method is an informed search method because techniques are used for guiding the search. Specifically, a heuristic based on the measured analog grid circuit voltages guides this paper’s informed search strategy. Fig. 5 shows a taxonomy of the methods used in this research.

This paper is organized as follows: Section II describes the Field Programmable Analog Array (FPAA) electronic
hardware used for results in this paper. Section III describes the mathematical theory behind this paper’s method. Section IV presents results from both FPAA hardware experiments and also software simulations. Section V provides analysis of the method, and Section VI presents the overall conclusions of this work.

II. ELECTRONIC HARDWARE

A Field Programmable Analog Array (FPAA) Integrated Circuit (IC) [39] is used to implement this path planning algorithm. The key enabler to this work’s FPAA implementation is that paths between nodes are varied using non-volatile floating-gate pFET transistors. These adjustable transistors can represent varying degrees of path cost. Fig. 6 shows how the simple three-path example in Fig. 3 is implemented using floating-gate transistors. The bold circles in Fig. 6 identify two transistors that are programmed differently to model path segments with different terrain conditions (smooth and rough). In this example, where path B is rough terrain, the transistors connecting nodes 25 and 26 and all the other transistors on path B.

FIGURE 6. The edges between nodes are implemented using field programmable floating-gate transistors. The conductances of the floating-gate transistors are inversely proportional to the edge weights. The two circles in this figure highlight that in this example where path ‘B’ is rough terrain, the transistor connecting nodes 1 and 7, W(1,7), will be programmed to conduct a higher current than the transistors connecting nodes 25 and 26 and all the other transistors on path B.

This section next presents a brief review of relevant equations for transistors operating in subthreshold and above threshold, and it highlights how these are field programmable. This is followed with a description of how a robot’s environment like in Fig. 6 is mapped onto the FPAA’s architecture.

A. BACKGROUND TRANSISTOR EQUATIONS

The weight variability is accomplished by programming floating-gate voltages to particular values. This allows one to either increase or decrease the value of the resistive connection between two nodes. The current in a pFET transistor in subthreshold operation may be described by (3) [40], [41].

\[ I = I_s e^{\frac{V_{sd}(k-1)}{v_T}} e^{\frac{\kappa v_p + V_s}{v_T}} \left[ 1 - e^{\frac{-V_{sd}}{v_T}} \right] \]  

where the current \( I_s \) is programmable from femtoamps (fA) to microamps (uA) by adjusting the charge on the floating-gate [42]. \( I_s \) in (3) is a constant representing pre-exponential factors, \( \kappa \) is a constant representing the capacitive coupling ratio from gate to channel, and \( v_T \) is the thermal voltage.

Dropping all higher order terms of the Taylor expansion of (3) with respect to voltage \( V_{sd} \) allows one to find an expression for the pFET’s resistance. The current is approximately linear with respect to \( V_{sd} \) for \( V_{sd} \leq 4v_T \approx 100 \text{ millivolts} \), (4) [29], [41].

\[ R_{subVth} = \frac{U_T}{I_s} \]  

A pFET in above threshold operation but in the deep triode regime \( (v_{ds} \ll 2(v_{gs} - v_{th})) \) also can be shown to have a linearly proportional relationship between current, \( I \), and \( v_{sd} \) (5) [41], [43].

\[ I \approx -\beta \left( v_{gs} - v_{tp} \right) \frac{v_{ds}}{2} \approx -\beta \left( v_{gs} - v_{th} \right) v_{ds} \]  

The resistance can therefore be estimated as (6).

\[ R_{aboveVth} \approx \frac{1}{\beta \left( v_{gs} - v_{th} \right)} \]  

where \( \beta = \frac{W}{L}\mu p C_{ox} \), \( W \) and \( L \) are the transistor width and length respectively, \( C_{ox} \) is the capacitance of the gate oxide, \( v_{th} \) is the threshold voltage, and positive current is from drain to source.

The key insight of this transistor equation discussion is this: The resistance is adjustable because the transistor’s gate voltage, \( v_{gs} \), is adjustable by Fowler-Nordheim tunneling or hot electron injection. These tunneling and injection processes have been successfully implemented in both commercial and custom analog VLSI integrated circuits such as an FPAA [39], [44].

B. EXPRESSING THE ROBOT’s ENVIRONMENT WITH TRANSISTORS

The robot’s environment is assumed to be discretized into a map with \( N \) by \( M \) nodes. This map can be implemented on an FPAA using a bipartite graph configuration [29]. The bipartite configuration allows one to deterministically map any arbitrary two or three dimensional grid map onto the FPAA’s hardware [29], [45]. Fig. 7 illustrates how part of the map in Fig. 6 is implemented on the FPAA routing fabric. Floating-gate transistors make connections between the horizontal and vertical wires. Each horizontal or vertical wire represents a node in the graph. The floating-gate transistors represent the weighted edges in the \( N \) by \( M \) map.
An FPAA’s ability to modify the strengths of these connections between horizontal and vertical wires is the enabling factor of this paper. In contrast to other published references, this paper emphasizes the path planning benefits that are possible by modifying these edge resistances.

The FPAA used in this research has multiplexer circuitry that allows the user to sense any point in the routing fabric [39]. This special feature allows one to easily measure node voltages and quickly arrive at a path planning solution.

III. SOLUTION METHOD BASED ON RESISTIVE GRID SOLVING PDE

This section first describes the mathematical theory behind this paper’s method, and this is followed by a brief power analysis.

A. LAPLACIAN’S EQUATION THEORY

As in [29], the voltage at each node is a function of its x and y position, \( v(x, y) \) in the grid. This voltage can be described by a particular elliptic Partial Differential Equation (PDE) called the Laplacian (7).

\[
\Delta v = \nabla v^2 = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \tag{7}
\]

The steady-state solution we seek is found when the Laplacian is no longer changing (\( \Delta v = \nabla v^2 = 0 \)). This steady-state condition equation is called Laplace’s equation or the potential equation. The FPAA’s analog circuits solve the potential equation PDE, and the answer is obtained by measuring node voltages. This is in contrast to digital methods which use finite difference methods to iteratively solve this grid for the node voltages, numerical methods which solve it using a set of simultaneous equations, or SPICE circuit simulation programs [22], [25], [46]–[48].

As in [29], Gradient Descent is used to find the path from start to goal. However, unlike the method in [29] which followed the lowest voltages, this paper’s method follows the highest current [47]. The measured node voltages, \( v \), are combined with the a priori knowledge of the edge weights, \( w \), to estimate the direction of maximum current flow (8). This is selected as the path direction.

\[
I_{\text{max}} = \max_{i} \left( \frac{v(n_{\text{robot}}) - v(n_{\text{neighbor}})}{w(e_{n_{\text{robot}} \text{to} n_{\text{neighbor}}})} \right) \tag{8}
\]

In (8) \( n_{\text{robot}} \) is the robot’s present node in the path (i.e. the last established node in the path which is being created), \( n_{\text{neighbor}} \) is the \( i^{th} \) immediate neighbor of the robot’s present node (i.e. a possible next node in the path), and \( w(e_{n_{\text{robot}} \text{to} n_{\text{neighbor}}}) \) is the edge weight (resistance) connecting \( n_{\text{robot}} \) to \( n_{\text{neighbor}} \).

B. POWER ANALYSIS

The total system power is composed of the resistive grid power and its supporting embedded system power which includes the voltage read-out and current estimation process. This section addresses only the resistive grid power in this analysis.

Ideally, the power consumed by the computation phase of Laplace’s equation is a function of the current into the grid and the equivalent resistance seen by the grid input current. Theoretically, the worst case equivalent grid resistance occurs when the path to the goal is a “lawn-mower” pattern (i.e. snake pattern) through every row in the grid. A conservative upper-bound of the total grid resistance is estimated by (9) where \( R_{\text{max}} \) is the worst case path segment cost.

\[
R_{\text{grid}} \leq R_{\text{max}} [N - 1] M + (M - 1) = R_{\text{max}} [NM - 1] \tag{9}
\]

where \( R_{\text{max}} \) is estimated using (4) or (6) with the smallest programmed floating-gate current \( I_{\text{max}} \) (i.e. largest floating-gate voltage). If one assumes a square grid (\( M = N \)) then one can estimate the worst case power as (10) which is a function of the input current into the grid. This current is supplied by a pFET transistor connected to the node representing the robot’s location [29].

\[
P = I_{\text{input}}^2 R_{\text{max}} \tag{10}
\]
map with three non-equal paths is described. This demonstrates a simple proof-of-concept. Second, an example with forty-two different possible paths from start to goal is presented. This example is used to show how the FPAA solution compares to the entire solution set for a specific start and goal path planning problem. Third, an example of an environment with thirteen obstacles, one goal, and thirty-five starting points is described. This example is used to show how the FPAA solution compares to two edge-cases. The edge-cases are the two ideals: ideal lowest cost path, and ideal shortest distance path.

1) EXAMPLE 1: SIMPLE PROOF OF CONCEPT

Fig. 8 shows FPAA hardware measurement results from implementing the three optional paths example described in Fig. 3, Fig. 6 and Fig. 7. The middle path is shortest, but the weights are set to penalize this path because its surface has rough terrain, which is an undesirable characteristic. The right path is the longest path. The left path is the second shortest path. The assumption is that the right and left paths’ surfaces are smooth, so neither of these path weights are penalized because of the path environment. In this example the FPAA hardware directs the robot to take the second longest path which is to the left. This path balances the two objectives of path length and optimal terrain.

2) EXAMPLE 2: HARDWARE CHOICE WITH RESPECT TO AN ENTIRE SOLUTION SET

Fig. 9 shows the result of using the FPAA hardware to find a path for the previously introduced problem in Fig. 4. This problem has forty-two different unique path possibilities from the start location to the goal location. The FPAA hardware chose a path of length six and cost 112,400. A MATLAB simulator was written to model resistive grid circuits. The edges in the simulation are modeled by linear resistors and therefore represent a simplified version of the actual circuit where the resistors are actually implemented by floating-gate pFET transistors. This circuit simulator solves for the node voltages for a given start node, end node, obstacle, and edge weight configuration. The simulator for this scenario chose a path of length eight and cost 93,600. Two noteworthy observations from this hardware result are described next.

A first noteworthy observation is that the path solution calculated by the FPAA is within 12.83% of the Pareto front. This is shown in Fig. 9b. One might conclude that while not on the Pareto front, it is within a reasonable acceptable margin as described in Fig. 2. This result shows an example of how the FPAA’s No-preference method compares to the choice a human in the loop might make if presented with the Pareto front.

A second noteworthy observation is that the FPAA hardware solution takes a slightly different path than is predicted by the software simulation for this scenario. For this particular scenario, the FPAA hardware path is 25% shorter but 20% more costly than the simulation path. This variance between the simulator and actual hardware result is not unexpected for this particular start/goal/obstacle scenario. This is because the large voltage drops, that can occur in the FPAA hardware between the goal’s neighbor nodes and the goal node, effectively reduce the programmed costs of these near goal paths. The proposed explanation is as follows. The goal node in the hardware is connected to ground through a diode-connected transistor. Because of this, the voltage difference between the goal and its neighbor nodes may be large. It may be so...
FIGURE 9. Hardware Example 2: This is a continuation of the Multi-Objective path planning problem in Fig. 4.: a) This $6 \times 6$ grid shows nodes and edges. The nodes are the dots, and the black lines are obstacles through which the robot can not pass. The edge cost weights are listed in Fig. 4. There are forty-two unique paths from the start to the goal, and this figure highlights four important ones. b) This is the complete Pareto plot for this problem [33]. This plot shows how the hardware results compare to the simplified MATLAB simulation, and it also shows how these two compare to the points on the Pareto front (the set of MO solutions). The conclusion is that in this example the FPAA hardware results are within only 12.83% of being on the Pareto front.

3) EXAMPLE 3: HARDWARE CHOICE COMPARED TO EDGE-CASE CHOICES

This third example demonstrates how the FPAA hardware solution compares with two edge-cases. The edge-cases are the two ideals: ideal lowest cost path, and ideal shortest distance path.

Fig. 10a is the $6 \times 6$ grid used as the environment for this example. Black lines represent obstacles, and 21.7% of the edges are obstacles (i.e. 13 obstacles). The numbers represent costs of traveling between nodes. Two studies were performed on this grid. Study 1 describes finding a path from the green star to the red circle in Fig. 10a. Study 2 describes finding a path from any node on the grid to the red circle in Fig. 10a.

**Regarding Study 1.** Fig. 10b shows path length and path cost metrics for each of the 866 possible path solutions from the green star to the red circle in Fig. 10a. Each red asterisk describes the associated path length and path cost for each of these 866 possible solutions. The FPAA hardware, MATLAB simulator, and Dijkstra algorithm [49] were each used to find a solution.

The results of Study 1 are as follows: The FPAA hardware results shown in Fig. 10 found the same Pareto optimal result as the MATLAB simulation and the Dijkstra optimal lowest cost algorithm. This provides empirical data to have confidence that the hardware finds an acceptable path in this MO problem.

**Regarding Study 2.** Fig. 10a was also used in a more comprehensive path study to evaluate the path from every node of Fig. 10a to the red circle. This resulted in 35 cases of finding paths from these non-goal nodes to the goal node. Two figures are included to summarize the results: one which compares the FPAA results to the lowest cost path, Fig. 10b, and one figure which compares the FPAA results to the shortest path length, Fig. 10c. To find the shortest path in each case, Dijkstra’s algorithm was used while assuming equal path costs for all edges. This provides the shortest path length.

The results of Study 2 are as follows: The FPAA hardware results shown in Fig. 10 chose the lowest cost path in 82.86% of the cases (29 of 35 cases). This is shown in Fig. 10c. The FPAA chose the shortest path in 91.43% of the cases (32 of 35 cases). This is shown in Fig. 10d. The data in Fig. 10d represents the analysis for all 35 cases, however, because the points overlap it is difficult to see how many red asterisks are at each point. A histogram of the data is used to reveal this information. Fig. 11 is a companion to Fig. 10d as it uses the same data. It shows a histogram of the frequency of occurrence of path lengths for the thirty-five paths tested in Fig. 10a. Study 2’s empirical data provides further confidence that the hardware can find acceptable paths in an MO path planning problem.
FIGURE 10. Hardware Example 3: Comparing hardware results to edge-cases—
a) This 6 × 6 grid is used for Example 3. Black lines represent obstacles, and 21.7% of the edges are obstacles (i.e., 13 obstacles). The numbers represent the cost of traveling between nodes. b) Study 1: In Fig. 10a, there are 866 possible path solutions from the green star (Node 1) to the red circle (Node 36). This figure plots an red asterisk describing the associated path length and path cost for each of these 866 possible solutions. The FPAA hardware was used to find the solution, and it found the same Pareto optimal result as the MATLAB simulation and Dijkstra optimal lowest cost algorithm [49]. c) Study 2a: The FPAA hardware was used to evaluate the path from every node of Fig. 10a to the goal node. This resulted in 35 path cases. The FPAA chose the lowest cost path in 82.86% of the 35 cases. This is shown by all 29 of the test cases that fall on the blue dashed line. d) Study 2b: This is companion data to Fig. 10c. This plot further analyzes the 35 path cases from the non-goal nodes to the goal node. It compares the path lengths found using the FPAA to the shortest path lengths found using Dijkstra’s algorithm. The FPAA chose the lowest cost path in 91.43% of the 35 cases (i.e., 32 of 35 cases). This is shown by all the test cases which fall on the blue dashed line.

In these 35 cases presented in Fig. 10, the FPAA hardware and the simulator gave the same path 77% of the time (27 of 35 cases). This same map was re-programmed and the path-finding re-run to demonstrate repeatability. The FPAA was again able to find paths from all 35 starting nodes to the goal. In some of the repeat cases, the FPAA hardware and the simulator gave the same path a different percentage of the time. In a repeat scenario, for example, the FPAA and simulation gave the same result 91.4% of the time (32 of 35 cases). The FPAA hardware in those results chose the lowest cost path in 85.71% of the cases (30 of 35 cases). The FPAA chose the shortest path in 88.57% of those cases (31 of 35 cases). Some results variability can be expected because of system and measurement variability, however these results serve to provide a proof-of-concept of the method.

One of the ways hardware failure is mitigated is by setting a software flag when an anomaly condition occurs, and having conditional software tests in place to re-evaluate the measurement/calculation if this flag is set. For instance, after the node voltages settle, then they are used to follow the downward
current gradient for the solution. An anomaly might occur when the measured voltages indicate that the current gradients at a particular node are all “uphill”. This could occur due possibly to voltage measurement sensitivity or resolution issues. This technique was used to get the Fig. 10c and Fig. 10d data for Hardware Example 3’s Study 2.

V. ANALYSIS

Three different aspects of this method are further discussed in this section. First, an analysis of the method’s computational complexity is presented. Next, analysis regarding effectiveness of the method with respect to the size of the grid is discussed (i.e. the size of the robot’s workspace). Finally, the importance of Analog to Digital Converter (ADC) quantization is discussed.

A. COMPLEXITY ANALYSIS

The time complexity of an algorithm is used to refer to its computation time [38]. A big advantage of using analog
FIGURE 12. Results for 271 MATLAB Monte Carlo simulation experiments for 50 × 50 grids. a) Scatter plot comparing the simulated analog solution to the simulated digital solution. Each experiment iteration is compared to the iteration’s optimal path length and optimal lowest cost. Results in the lower left corner are preferred. For example, a red circle data point at (0,0) would indicate that the simulation iteration’s analog solution is both the ideal lowest cost path and shortest path. b) A cumulative distribution function (CDF) of the scatter plot data in Fig. 12a. The arrows point to \( F(100) \approx 0.87 \). This means that approximately 87% of the scatter plot data in Fig. 12a are located inside the 100% by 100% square. The conclusion is that for these Monte Carlo simulation experiments approximately 87% of the Monte Carlo iteration’s analog path solutions are less than twice as long and/or twice as costly as the optimal shortest paths and lowest costs. This empirical data provides insight into the quality of this paper’s no-preference MO solution method.

FIGURE 13. This is a companion figure to the data in Fig. 12a. It uses the same data, however, this histogram reveals the unobservable overlapping red circle data points in Fig. 12a. Each occurrence corresponds to one of the 271 MATLAB Monte Carlo iteration results.

FIGURE 14. Analog vs Digital complexity estimation comparisons for increasing \( N \times N \) grid size. This graph presents notional parameters for \( s_{meas} \) and \( bias_{prog} \) in order to demonstrate that theoretical estimates of the computational complexity “winner” depends on the factors of grid size and analog hardware computation parameters.

computation is the potential for a quicker computation time. Fig. 14 compares Dijkstra’s computation complexity to the FPAA’s complexity. Dijkstra’s complexity can be expressed as (14) where \( n \) is the number of vertices and \( m \) is the number of edges [50].

\[
O(n \log n + m) \quad (14)
\]

The number of edges, \( m \), is calculated using (15) where \( N \) is the size of an \( N \) by \( N \) grid (therefore \( n = N \times N \)).

\[
m = (N - 1)N + (N - 1)N \quad (15)
\]

This paper’s FPAA computation complexity estimation builds on previous analysis, [24], [29], and it uses scale factors to account for the analog system parameters of programming time, \( bias_{prog} \), and node voltage measurement read-out times, \( s_{meas} \). Regarding voltage read-out times, it may take the equivalent of multiple digital clock cycle times for each FPAA grid node’s voltage measurement. The scale factor parameter \( s_{meas} \) is intended to model this. The FPAA’s complexity can be estimated as (16) where \( d \) is the solution length.
In order to compare the complexity estimates of Dijkstra and FPAA methods one must make a correlation between the number of vertices, \( N \), and the solution path length, \( d \). This correlation estimate is found in (17) where \( d_{\text{max}} \) represents a “lawn-mower” pattern (i.e. snake pattern) path through the entire grid.

\[
O \left( d \cdot s_{\text{meas}} + \text{bias}_{\text{prog}} \right) \\
d = d_{\text{max}} = (N - 1)N + (N - 1)
\]

Fig. 14 compares Dijkstra’s complexity to the FPAA complexity estimate. This graph presents notional parameters for \( s_{\text{meas}} \) and \( \text{bias}_{\text{prog}} \) in order to demonstrate that theoretical estimates of the computational complexity “winner” depends on the factors of grid size and analog hardware computation parameters. An important take-away of Fig. 14, is that while increased performance of digital systems certainly provides improved solutions when paired with digital centric algorithms, even digital systems are not immune to the curse of dimensionality. Therefore, there is merit with continuing to explore and analyze alternate solutions such as this analog based solution.

**B. WORK SPACE SIZE ANALYSIS**

It is known that approaches using resistive grids to plan paths for mobile robots “become unattractive if long distances and high resolutions are required [46].” This has been explored in both numerically simulated methods of resistive grids, [22], [23], and in hardware versions [24]. In hardware versions, when the size of the resistive grid increases then the voltage drops between nodes can decrease (when all other factors are kept constant). This creates a potential voltage measurement challenge if the voltage drops are extremely small (i.e. a flat potential surface). In light of this, a couple of methods of using this planning method with large workspaces are:

- Plan over the entire workspace but lower the grid path resolution within the workspace or in parts of the workspace (variable node density) [26]
- Plan over the entire workspace, keeping the same grid path resolution, but use a finer resolution Analog to Digital Converter (ADC), i.e. one with more bits of precision.

Using longer voltage averaging times can also be helpful in both of the above methods, however this will also increase the planning time.

**C. QUANTIZATION EFFECTS ANALYSIS**

Quantization effects of numerically simulated methods of resistive grid path planning are discussed in [22]. They use a narrow corridor example to analyze the dynamic range of the surface. Their conclusion is that the dynamic range “can become arbitrarily large [22]” and “the number of bits of required precision is proportional to the length of the corridor divided by its width, its aspect-ratio [22].”

Regarding hardware quantization effects, an important take-away from this paper is that it is critically important to use an Analog to Digital Converter (ADC) with as many bits of precision as is practically possible. This helps to ensure that one may accurately ascertain the voltage difference between each pair of node neighbors in the grid. The ADC used for the hardware results in this paper is an integrated part of the Atmel ARM-based microcontroller on the embedded system board [39], [51]. It is a 10-bit ADC coupled to an external 2.5V high precision, low noise, low dropout voltage reference. This results in a 2.4 millivolt ADC step size (i.e. \( \text{step size} = \frac{V_{\text{ref}}}{2^{10}} \)) and therefore a possible quantization error of \( + - 1.2 \) millivolts (half of the step size) [52]. This quantization error is shown in the hardware results in Fig. 8. If one follows Path C from the robot start to the robot goal one sees that there are two pairs of nodes where the current appears to be going “up hill” when it is not. These are the pairs: (2.0855 Volts to 2.0863 Volts) and (2.0688 Volts to 2.0703 Volts). This corresponds to node differences of 0.7473 millivolts and 1.5 millivolts respectively. Both measurement noise and quantization error can cause results such as these. Fig. 15 shows all thirty-three measured node voltage differences. The voltage differential of the two “up hill” pairs in this discussion can be seen below the dashed line representing the voltage difference measurement floor. Thus, the quantization error may be contributing to the erroneous apparent “up hill” currents.

**VI. CONCLUSION**

This paper presents the first example of floating-gate based analog resistive grid circuits used for Multi-Objective
path planning. In this work, variable resistance values represent non-uniform costs between nodes. This paper analyzed the capability for its applicability in solving path planning problems with multiple objectives. The Multi-Objective method presented in this paper is characterized as a non-preference method, and the hardware supplied solution is a mixture of the multi-objectives, Fig. 12.

The hardware experiments showed that the analog MO method can find a solution that is close to the Pareto front. For example, in Fig. 9b the FPAA solution was 12.83% from the front and in Fig. 10b the FPAA solution is on the Pareto front. The 35 hardware cases in Fig. 10c and Fig. 10d, and the simulation data in Fig. 12 provide further insight into the quality of the analog MO solutions. For example, Fig. 10c and Fig. 10d show that in most of these example cases the FPAA solution was equal to the ideal lowest path cost and the ideal shortest path length. Fig. 12 and Fig. 13 show that approximately 87% of the 271 simulated analog solutions were less than a threshold of 100% error in path length and/or path cost.

This work used uniform and static obstacles, however unstructured environments are also important planning environments. Some examples of these include parking lots to be traversed [53], minefields which are to be cleared [54], or robots performing rescue in dangerous environments such as a building fire or a building damaged by an earthquake [55].

The key question being addressed in this paper is to address the analog method’s use in multi-objective problems for cases of known paths with clear pre-known weights for each path segment. In order for this algorithm to be effective in an unstructured environment, it is assumed that the robot’s sensors have ascertained any path change (from the assumed path), and have updated the a priori pre-existing map. For example, the robot may have a pre-existing architectural plan for a building, and it will be run-time updated based on any sensor feedback that may indicate path changes (for example paths blocked by debris or fire). After this update, the algorithm in this paper could be run.

In this paper the simulation studies used Dijkstra’s algorithm to provide the two optimal conditions (optimally shortest, and optimally lowest cost) to provide a clear comparison of results. Dijkstra’s algorithm, or even A*, however do not make a judgment call as to how to choose a final answer based on the multi-objective question of which path to take based on these two criteria. This paper sought to provide a solution for that problem and show how it compares to the Dijkstra results.

The overall conclusion is that analog can offer a potential computation complexity advantage for Multi-Objective problems which can tolerate a trade-off of sub-optimal solutions in the metrics. This is a reasonable and desired capability for some situations because sometimes computation time is critically limited, and a good enough decision made quickly is better than an optimal solution delivered too late.

Future work can include experiments with larger grids on FPAA hardware.

ACKNOWLEDGMENT
The author wishes to thank J. Hasler at Georgia Institute of Technology for FPAA hardware access and G. Ledford at Baylor University for MATLAB circuit simulation software. The author also thanks the anonymous reviewers whose comments helped improve the paper.

REFERENCES

[1] M. G. Castillo Tapia and C. A. Coello Coello, “Applications of multi-objective evolutionary algorithms in economics and finance: A survey,” in Proc. IEEE Congr. Evol. Comput., Sep. 2007, pp. 532–539.
[2] M. Janidarmian, A. Roshan Fekr, K. Radecka, and Z. Zilic, “Multi-objective hierarchical classification using wearable sensors in a health application,” IEEE Sensors J., vol. 17, no. 5, pp. 1421–1433, Mar. 2017.
[3] S. Topaloglu, “A multi-objective programming model for scheduling emergency medicine residents,” Comput. Ind. Eng., vol. 51, no. 3, pp. 375–388, Nov. 2006.
[4] M. A. Badri, D. L. Davis, D. F. Davis, and J. Hollingsworth, “A multi-objective course scheduling model: Combining faculty preferences for courses and times,” Comput. Oper. Res., vol. 25, no. 4, pp. 303–316, Apr. 1998.
[5] A. Nunez, A. Jamshidi, and H. Wang, “Pareto-based maintenance decisions for regional railways with uncertain weld conditions using the Hilbert spectrum of axle box acceleration,” IEEE Trans. Ind. Inform., vol. 15, no. 3, pp. 1496–1507, Mar. 2019.
[6] A. Raza, Z. Yousaf, M. Jamli, S. O. Gilani, G. Abbas, M. Uzair, S. Shaheen, A. Benrabah, and F. Li, “Multi-objective optimization of VSC stations in multi-terminal VSC-HVdc grids, based on PSO,” IEEE Access, vol. 6, pp. 62995–63004, 2018.
[7] H. Jan and Z. Qiangbo, “Multi-objective mobile robot path planning based on improved genetic algorithm,” in Proc. Int. Conf. Intell. Comput. Technol. Autom., May 2010, pp. 752–756.
[8] Q. Yang and S.-J. Yoo, “Optimal UAV path planning: Sensing data acquisition over IoT sensor networks using multi-objective bio-inspired algorithms,” IEEE Access, vol. 6, pp. 13671–13684, 2018.
[9] K. Jeddilasvari, R. I. Altayoghe, L. C. A. Pimenta, and F. G. Guimaraes, “Multi-objective approach for robot motion planning in search tasks,” Int. J. Speech Technol., vol. 45, no. 2, pp. 305–321, Sep. 2016, doi: 10.1007/s10498-015-0754-y.
[10] F. Ahmed and K. Deb, “Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms,” Soft Comput., vol. 17, no. 7, pp. 1283–1299, Jul. 2013.
[11] L. Ferariu and C. Ciamparu, “Pareto genetic path planning hybridized with multi-objective Dijkstra’s algorithm,” in Proc. 18th Int. Conf. Syst. Theory, Control Comput. (ICSTCC), Oct. 2014, pp. 341–346.
[12] K. Jeddilasvari, R. J. Altayoghe, and F. G. Guimaraes, “Multi-objective mobile robot path planning based on A* search,” in Proc. 6th Int. Conf. Comput. Knowl. Eng. (ICCKE), Oct. 2016, pp. 7–12.
[13] P. P.-Y. Wu, D. Campbell, and T. Merz, “Multi-objective four-dimensional vehicle motion planning in large dynamic environments,” IEEE Trans. Syst., Man, Cybern., B (Cybern.), vol. 41, no. 3, pp. 621–634, Jun. 2011.
[14] T. Oral and F. Polat, “MOD*: lite: An incremental path planning algorithm taking care of multiple objectives,” IEEE Trans. Cybern., vol. 46, no. 1, pp. 245–257, Jan. 2016.
[15] U. C. Kaya, A. Dogan, and M. Huber, “A utility-based path planning for safe UAS operations with a task-level decision-making capability,” in Proc. IEEE Int. Conf. Syst., Man Cybern. (SMC), Oct. 2019, pp. 1227–1233.
[16] T. Shan and B. Englot, “Sampling-based minimum risk path planning in multiobjective configuration spaces,” in Proc. 54th IEEE Int. Conf. Decis. Control (CDC), Dec. 2015, pp. 814–821.
[17] S. Koziol, S. Brink, and J. Hasler, “A neuromorphic approach to path planning using a reconfigurable neuron array IC,” IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 22, no. 12, pp. 2724–2737, Dec. 2014.
[18] J. Kim, R. A. Pearce, and N. M. Amato, “Extracting optimal paths from roadmaps for motion planning,” in Proc. IEEE Int. Conf. Robot. Autom., Sep. 2003, pp. 2424–2429.
