Proceedings of the 45th IEEE Conference on Decision & Control
Manchester Grand Hyatt Hotel
San Diego, CA, USA, December 13-15, 2006

On Finding Globally Optimal Paths through Weighted Colored Graphs

David Wooden & Magnus Egerstedt

School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
{wooden,magnus}@ece.gatech.edu

***Abstract*— In this paper, we present a method for finding a globally optimal path through a colored graph. Optimal here means that, for a given path, the induced path coloring corresponds to an equivalent class. A total ordering is placed over these equivalent classes, and the edge weights are simply tie breakers within the classes. Optimality is achieved by mapping the class, or color, of each edge in combination with its weight to a real number. As a result, optimal paths can be computed using just the new weight function and standard edge relaxation methods (e.g. Dijkstra's Algorithm). The motivation for this research is the task of planning paths for mobile autonomous robots through outdoor environments with unknown and varied terrain.**

I. Introduction

An emerging approach to handling the complexity associated with robot navigation tasks in unstructured environments is to decompose the control task into basic building-blocks [2], [4], [6]. Each atomic building block corresponds to a particular control law (or mode), defined with respect to different tasks, sensory sources, or operating points [3]. The high-level control question then becomes that of concatenating these building-blocks together in order to meet the global objectives. The inherently unknown and unstructured nature of autonomous mobile robots' environments makes control particularly challenging, inasmuch as any precomputed optimal sequencing of the modes quickly becomes invalidated (suboptimal or even infeasible) as the environment or the robot's knowledge thereof changes.

The work in this paper is based on an exploration of the different modes in the sense that by trying different mode strings, the robot builds up a high-level description of how the selection of particular modes in particular situations affects the performance of the system, similar to the idea presented in [7]. We achieve this through so-called Visual Feature Graphs, where each edge in the graph corresponds to a particular control law, and each vertex corresponds to a distinctive feature or place. Such graphs thus describe how the application of a certain control law takes the robot from one distinctive feature to the next. As an example, consider Fig. 1. This way of structuring information about the environment is especially appealing in outdoor robotics applications, where the (distinctive) feature density can be expected to be fairly low.

This work was supported by DARPA through Grant number FA8650-04-C-7131.

Once such a Visual Feature Graph has been produced, one can plan optimal paths over them, which is the topic under investigation in this paper. However, since some distinctive places correspond to areas where the robot is likely to get stuck (such as mud, brushy vegetation or quicksand), these places (or vertices) should be avoided. Other vertices may define places where the robot could possibly traverse successfully but which should nonetheless be avoided if more traversable paths are available. We encode these initial observations in a directed, colored graph $G = (V, E, W_E, C_V, C_E)$ where

- V is the set of vertices (distinctive places/features).
- $E \subset V \times V$ is a set of ordered pairs of vertices.
- $W_E : E \to \mathbb{R}^+$ is a cost associated with each edge. This cost can for instance be interpreted as distance travelled or how long it would take to reach the edge's target vertex.
- $C_V : V \to \mathcal{K}$ designates a class (i.e. color) $C_V(v)$ to each vertex in the graph. The set of classes $\mathcal{K} = \{1, \ldots, K\}$ corresponds to the different levels of traversability (or some other encoding of how suitable that distinctive place is for navigation), as discussed above.
- $C_E : E \to \mathcal{K}$ designates a class $C_E(e)$ for each edge in the graph. In fact, we will let $C_E(e)$ be given by $C_V(v)$ where v is the target vertex of edge e.

The planning problem thus becomes that of finding the best path over a set of vertices, from the current vertex to the goal vertex, where "best" is loosely interpreted as minimizing the total edge-cost while avoiding vertices belonging to "bad" classes.

Some comments about the notation used in this paper should be made. We assume that we are given C_V, which assigns a class to a vertex, and the class corresponds to the traversability of the terrain associated with that distinctive place. Hence, our graph is a colored graph (following the notation of [5]), but because it is possibly *improperly colored*, and a vertex's color is dictated by terrain quality rather than, for example, its connectivity, we prefer the term "class"

over "color".

We moreover define C_E, the class of an edge, to be the class of the target vertex of the edge. The reason for this is that it is natural to assign a class to an edge matching where the edge terminates and the terrain that must be traversed to go there. (Note that C_E could instead be defined based on the source vertex of an edge with little effect for the purposes of this paper. It is important, however, that the definition be consistent.)

To make more concrete what we mean by "best path" and "bad classes", let us first establish the basic goal behind this research. We want to be able to use existing solutions to the `find-path` problem (e.g. Dijkstra's Algorithm), where, when given a colored graph G (as defined above), we are asked to plan a path over the edges between two specified vertices (for example, see [8]). Our goal here is to find a weight mapping $U_E : E \mapsto \mathbb{R}^+$, whereby global properties of the graph (including its constituent classes) are incorporated into U_E, with the result that the simpler new weighted graph (V, E, U_E) can be planned over without having to involve semantic symbols, i.e. the vertex classes.

This paper presents a method for determining U_E and an optimal path over the new graph that is in fact optimal also over the original colored graph. Here, "best" does not correspond to a simple minimum edge-cost path, but rather, as already pointed out, a path that is optimal within the set of paths of the lowest "path class". The next section causally motivates the problem we solve in this paper, and Section III presents a formal problem statement. The remainder of the paper is organized as follows: In Section IV, we present our solution, i.e. the derivation of an appropriate the mapping U_E. Sections V and VI provide theorems and proofs, and Section VIII finalizes the discussion with conclusions.

II. Motivation

In this section, we present a situation where we could naturally define distinctive terrain features. This example should be thought of as simply a motivating example, rather than a realistic, full-scale robotics application. Fig. 1 depicts an outdoor environment populated with six different terrain types (listed in decreasing traversability): roadway, pathway, field, forest, mud, and marsh. Marked in the figure are *start* and *goal* vertices. What is the best route over this environment?

What we know from experience is that wheeled robots drive efficiently over roadways and pathways, and reasonably well in fields, but have trouble in forests, and are practically stuck in mud or marsh. Three paths are overlaid on the figure, showing potential routes to the goal. Route 1 is the shortest, but passes through a forest region (for specifics, see Table 1). Route 2 is longer but avoids the forest regions, passing at worst through a field. Route 3 is similar to Route 2, but is in the field regions less. We intuitively come to the conclusion that Route 3 is the best of the three.

Consider Fig. 2, which is a graphical representation of Fig. 1. Routes 1-3 are marked on the graph. We will use the developments in the following sections to show that we can use standard edge relaxation methods on a simple graph (V, E, U_E), and in so doing, identify Route 3 as the best path among Routes 1-3.

It is our intention within this work that the difference amenest classes should represent something that could not be reflected as a well-defined or well-posed transform on edge cost. In fact, we will impose a total ordering on the path classes and simply use the edge weights to break intra-class ties. A good example might be to define class based on what's necessary for the robot's safe operation. In regions where the robot may totally fail its mission, such situations (may) need to be completely avoided if at all possible - if any alternative route is available. This work is targeted at tractable path planning for these types of scenarios.

III. Problem Definition

Define an edge $e_{ij} \in E$ as the ordered pair (v_i, v_j) with v_i, $v_j \in V$; the edge passes in the direction from $v_i \longmapsto v_j$. Let the path p be a string of edges over G which connects some v_0 to v_F, the first and last vertices of the string. The edges of p are ordered such that the n-th edge passes from vertex

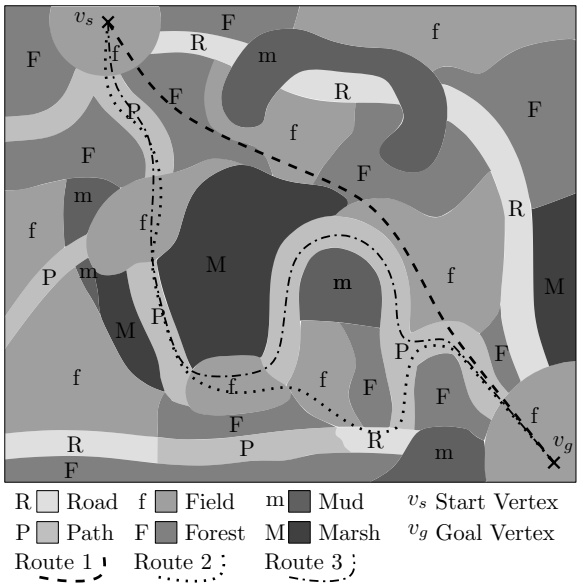

Fig. 1. An example environment with 6 terrain classes; roads are the most easily traversed, followed by paths, fields, forests, mud, and marsh. Three feasible paths between the start and goal vertices are shown.

Route Number	Road Edges	Path Edges	Field Edges	Forest Edges	Mud Edges	Marsh Edges
1	0	1	5	3	0	0
2	1	7	4	0	0	0
3	0	9	3	0	0	0

TABLE I

The number of edges per class for Paths 1-3 from Fig. 2.

Note how Path 1 has edges in the Forest class, while Paths 2 & 3 do not. Also, note how Path 3 has fewer Field edges than Path 2.

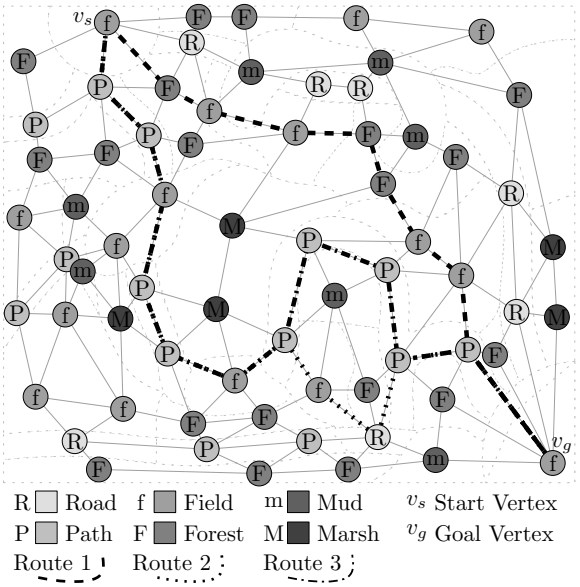

Fig. 2. A combinatorial representation $G = (V, E, W_E, C_V, C_E)$ of the environment from Fig. 1. The discs represent vertices, with labels identifying the class of the vertex. Edges are depicted as undirected for the sake of clarity. Note that Paths 2 and 3 partially overlap.

v_n to v_{n+1}, with $n \in \{1, \ldots, length(p)\}$ and $length(p)$ is the length of p.

Partition the set of vertices V into subsets based on class, such that

$$V = \bigcup_{k=1}^{K} V_k, \tag{1}$$

where

$$V_k = \Big\{ v \in V \;\Big|\; C_V(v) = k \Big\} \tag{2}$$

and where K is the number of classes. Similarly, partition the set of edges E, such that

$$E = \bigcup_{k=1}^{K} E_k, \tag{3}$$

where

$$E_k = \Big\{ e_{ij} = (v_i, v_j) \in E \;\Big|\; v_j \in V_k \Big\}. \tag{4}$$

The set of all acyclic paths connecting v_s to v_g (the start and goal vertices) is denoted by Π. (We could more precisely call this $\Pi(v_s, v_g)$, but omit the arguments for the sake of clarity.) Let Π_k be the set of paths where the highest vertex class included in each path within Π_k is exactly k. In other words,

$$\Pi_k = \Big\{ p \in \Pi \;\Big|\; N(p,k) > 0 \;\&\; N(p,l) = 0,\ \forall l > k \Big\}, \tag{5}$$

with

$$N(p,k) = \mathrm{card}(\{ e \in p \;\big|\; C_E(e) = k \}) \tag{6}$$

being the number of edges of the path p that are of class k. (Here, $\mathrm{card}()$ denotes cardinality.) Furthermore, let

$$\Pi_k^i = \Big\{ p \in \Pi_k \;\Big|\; N(p,k) = i \Big\} \tag{7}$$

be the set of k-class paths with exactly i edges of class k.

Now, when asked to plan a path between v_s and v_g, we define the (not necessarily unique) optimal path p^* as one satisfying the following criteria:

1) Let
$$\kappa = \min_{k \in \{1,\ldots,K\}} (\Pi_k \neq \emptyset), \tag{8}$$
and hence Π_κ is the set of all paths with the minimum maximum-vertex-class in Π. The optimal path p^* is in Π_κ.
2) Let
$$\mathcal{P}_\kappa = \arg\min_{p \in \Pi_\kappa} N(p, \kappa). \tag{9}$$
So, $\mathcal{P}_\kappa$ is the set of paths in Π_κ with the fewest number of κ-class vertices. The optimal path p^* is in $\mathcal{P}_\kappa$.
3) Let
$$\mathcal{P}_{\kappa-1} = \arg\min_{p \in \mathcal{P}_\kappa} N(p, \kappa - 1). \tag{10}$$
$\mathcal{P}_{\kappa-1}$ is the set of paths in $\mathcal{P}_\kappa$ with the fewest number of $(\kappa-1)$-class vertices.
4) Similarly,
$$\mathcal{P}_i = \arg\min_{p \in \mathcal{P}_{i+1}} N(p, i), \tag{11}$$
where $i = \{1, \ldots, \kappa - 1\}$. Path p^* is in all $\mathcal{P}_i$.
5) Finally, if $card(\mathcal{P}_1) > 1$, we wish that
$$p^* = \min_{p \in \mathcal{P}_1} \big(\sum_{e \in p} W_E(e) \big). \tag{12}$$

Consequently, we have a recursive problem definition, with the optimal path belonging to recursively dependant sets of paths, based on edge classes. What we desire is a formulation of a new edge-weight function U_E that encodes this semantic information about the edge classes and incorporates the original weight function W_E. In finding it, we will obtain a p^* satisfying Criteria 1-5, without having to worry about vertex classes. In fact, one can think of this construction as imposing a total order on the path classes, with $\Pi_k^i < \Pi_l^j$ for all $l > k$, and $\Pi_k^i < \Pi_k^j$ for all $j > i$, and so on.

Criteria 1-5 take into account all paths in the graph between the start and goal vertices. Hence, a selection of U_E finding p^* with respect to these specifications accounts for *all* such paths, and consequently, accounts for global properties of the graph's vertices and each vertex's class. Note that some modification of $W_E(e)$ based only on the immediate neighbors of $e \in E$ would be insufficient to guarantee the solution of p^* via an algorithm like Dijkstra's Algorithm. That is, locally modifying W_E does not incorporate the (global) information needed to satisfy criteria 1-5.

Now, we break the task of finding p^* into three subproblems:

Subproblem 1: Find $U_E : E \mapsto \mathbb{R}^+$ such that, corresponding to Criteria 1, the cost of paths belonging to class k (i.e. $p \in \Pi_k$) should always be greater than the cost of paths belonging to class $k-1$. By the cost of a path p, we mean

$$c(p) = \sum_{e \in p} U_E(e). \tag{13}$$

Subproblem 2: Find $U_E : E \mapsto \mathbb{R}^+$ such that, corresponding to Criteria 4, the cost should be higher for the path which has more edges in class k than other paths which have the same number of edges for all classes greater than k.

Subproblem 3: Find $U_E : E \mapsto \mathbb{R}^+$ such that, corresponding to Criteria 5, for paths that have the same number of edges in all classes, the optimal path minimizes $\sum_{e\in p} W_E(e)$. In other words, paths that are equivalent according to class are distinguished based on the original edge weighting.

This leads us to the main problem under consideration in this paper, namely:

Problem 1: Find $U_E : E \mapsto \mathbb{R}^+$ such that the optimal path p^* over the weighted graph (V, E, U_E) is also optimal over the original colored graph according to the optimality Criteria 1-5.

The overall implication of these subproblems is three-fold:

1) Subproblem 1 implies that the passage of a path through a k-class vertex makes that path less desirable than every path that visits vertices of class strictly less than k, without regard to how long or costly such paths may be or how many vertices they include; vertex classification bears strong meaning.
2) Subproblem 2 implies that the passage between adjacent vertices of the same class has more meaning than simply the weight of the edge joining them. The number of hops in a vertex class along a path has more impact than the length of the path.
3) The last subproblem implies that while vertex classification is the primary factor in determining an optimal path, the regular edge cost W_E is still required where vertex classification of two paths is identical.

IV. The new edge-weight function, U_E

Define the modified weight function

$$U_E(e) = W_E(e) + \sigma_i + \gamma_i, \tag{14}$$

with

$$\sigma_i = \sum_{e\in E_i} W_E(e), \tag{15}$$

$$\gamma_i = (1 + n_{i-1})(\sigma_{i-1} + \gamma_{i-1}), \tag{16}$$

and

$$\gamma_1 = 0, \tag{17}$$

where $n_i = card(E_i)$ denotes the number of edges in E_i, Finally, let C_{Π_k} be

$$C_{\Pi_k} = \Big\{ c(p) \;\Big|\; p \in \Pi_k \Big\}, \tag{18}$$

the set of costs for the paths in Π_k, where, as before,

$$c(p) = \sum_{e\in p} U_E(e). \tag{19}$$

We shall demonstrate in the following sections that this recursive definition of γ_i is just what we need to solve Problem 1.

V. Lemmas

Two lemmas will prove useful for the next section.

Lemma 5.1: The maximum cost of Π_k is less than γ_{k+1}.

Proof:

For an acyclic path which never visits any vertex v with $C_V(v) > k$, the highest possible cost of the path corresponds to one that includes every edge $e \in E$ where $C_E(e) \le k$ (though it may be that no such path exists). That is,

$$\begin{aligned}
&max(C_{\Pi_k}) \\
&\le \sum_{i=1}^{k} \sum_{e\in E_i} U_E(e) \\
&= \sum_{e\in E_1} U_E(e) + \sum_{e\in E_2} U_E(e) + \ldots + \sum_{e\in E_k} U_E(e) \\
&= (\sum_{e\in E_1} W_E(e) + n_1(\sigma_1 + \gamma_1)) + \\
&\quad +(\sum_{e\in E_2} W_E(e) + n_2(\sigma_2 + \gamma_2)) + \\
&\quad + \ldots + (\sum_{e\in E_k} W_E(e) + n_k(\sigma_k + \gamma_k)) \\
&= (\sigma_1 + n_1(\sigma_1 + \gamma_1)) + (\sigma_2 + n_2(\sigma_2 + \gamma_2)) + \\
&\quad + \ldots + (\sigma_k + n_k(\sigma_k + \gamma_k)) \\
&= (1 + n_1)(\sigma_1 + \gamma_1) + (\sigma_2 + n_2(\sigma_2 + \gamma_2)) + \\
&\quad + \ldots + (\sigma_k + n_k(\sigma_k + \gamma_k)) \\
&= (\gamma_2) + (\sigma_2 + n_2(\sigma_2 + \gamma_2)) + \\
&\quad + \ldots + (\sigma_k + n_k(\sigma_k + \gamma_k)) \\
&= (\gamma_3) + (\sigma_3 + n_3(\sigma_3 + \gamma_3)) + \\
&\quad + \ldots + (\sigma_k + n_k(\sigma_k + \gamma_k)) \\
&\cdots \\
&= (\gamma_{k-1}) + (\sigma_{k-1} + n_{k-1}(\sigma_{k-1} + \gamma_{k-1})) + \\
&\quad +(\sigma_k + n_k(\sigma_k + \gamma_k)) \\
&= (\gamma_k) + (\sigma_k + n_k(\sigma_k + \gamma_k)) \\
&= \gamma_{k+1}
\end{aligned}$$

■

Lemma 5.2: The minimum cost of Π_k is greater than $\sigma_k + \gamma_k$.

Proof:

Conceptually, the path in Π_k with minimum cost visits the fewest vertices possible and the weight of the edges on this path are minimal. So such a path would, in principle, visit exactly one k-class edge and a minimum of other edges. There are two possible cases:

Case 1: If $C_V(v_g) = k$, (where v_g is the goal vertex) then the minimal path would be just $v_s \to v_g$ (if such a path existed). So,

$$\begin{aligned}
&min(C_{\Pi_k}) \\
&\quad \ge U_E(e_{sg}) \\
&\quad = W_E(e_{sg}) + \sigma_k + \gamma_k \\
&\quad > \sigma_k + \gamma_k
\end{aligned}$$

Case 2: If $C_V(v_g) \neq k$, then the minimal path would be $v_s \rightarrow v_j \rightarrow v_g$, where $v_j \in V_k$. So,

$$\begin{aligned} & min(C_{\Pi_k}) \\ & \geq \; U_E(e_{sj}) + U_E(e_{jg}) \\ & = \; (W_E(e_{sj}) + \sigma_k + \gamma_k) + \\ & \quad +(W_E(e_{jg}) + \sigma_{C_V(v_g)} + \gamma_{C_V(v_g)}) \\ & > \; \sigma_k + \gamma_k \end{aligned}$$

In both cases, we find

$$min(C_{\Pi_k}) > \sigma_k + \gamma_k. \tag{20}$$

■

VI. Theorems

Theorem 6.1:

$$max(C_{\Pi_{k-1}}) < min(C_{\Pi_k}). \tag{21}$$

Proof:

Simply taking Lemmas 5.1 and 5.2 together, we find

$$max(C_{\Pi_{k-1}}) \leq \gamma_k < \sigma_k + \gamma_k < min(C_{\Pi_k}) \tag{22}$$

■

Theorem 6.2: The cost of traversing a k-class path with j k-class vertices is always less than a k-class path with $(j+1)$ k-class vertices, where k is the highest class on the two paths. In other words,

$$max(C_{\Pi_k^j}) < min(C_{\Pi_k^{j+1}}).$$

Proof: First, we assume $card(E_k) > j$.

$$\begin{aligned} & max(C_{\Pi_k^j}) \\ & \leq \; \sum_{i=1}^{k-1} \sum_{e \in E_i} U_E(e) + \text{worst } j\text{-length path in } E_k \\ & < \; \sum_{i=1}^{k-1} \sum_{e \in E_i} U_E(e) + j(\sigma_k + \gamma_k) + \sum_{e \in E_k} W_E(e) \\ & = \; \gamma_k + j(\sigma_k + \gamma_k) + \sigma_k \\ & = \; (j+1)(\sigma_k + \gamma_k) \end{aligned}$$

$$\begin{aligned} & min(C_{\Pi_k^{j+1}}) \\ & \geq \; \text{best } (j+1)\text{-length path in } E_k \\ & > \; (j+1)(\sigma_k + \gamma_k) \end{aligned}$$

Hence,

$$max(C_{\Pi_k^j}) < min(C_{\Pi_k^{j+1}}). \tag{23}$$

This proves the theorem.

■

Theorem 6.1 establishes that U_E solves Subproblem 1. In other words, a path in Π_k always has lower cost than a path in Π_{k+1}. Theorem 6.2 establishes that U_E solves Subproblem 2. U_E also solves Subproblem 3, by noting that for a path p

$$\begin{aligned} c(p) \; &= \; \sum_{e \in p} U_E(e) & (24) \\ &= \; \sum_{e \in p} (W_E(e) + \sigma_{C_E(e)} + \gamma_{C_E(E)}), & (25) \end{aligned}$$

and that for two paths p_1 and p_2 with the same numbers of edges over all the classes, the difference between the cost of the two paths reduces to the difference of the sum of their edge weights. That is,

$$c(p_1) - c(p_2) = \sum_{e \in p_1} W_E(e) - \sum_{e \in p_2} W_E(e). \tag{26}$$

By solving each of the three subproblems, U_E solves Problem 1.

VII. Implementation

Fig. 3 shows a screenshot taken from the GRITSlab [1] graph planning software library corresponding to the graph depicted in Fig. 2. Here, edge weights have been assigned according to U_E, and Dijkstra's algorithm has been run over the resulting graph. The dark line pointing out of each node indicates the next node to be taken on the optimal path to the goal (which is located in the lower right-hand corner).

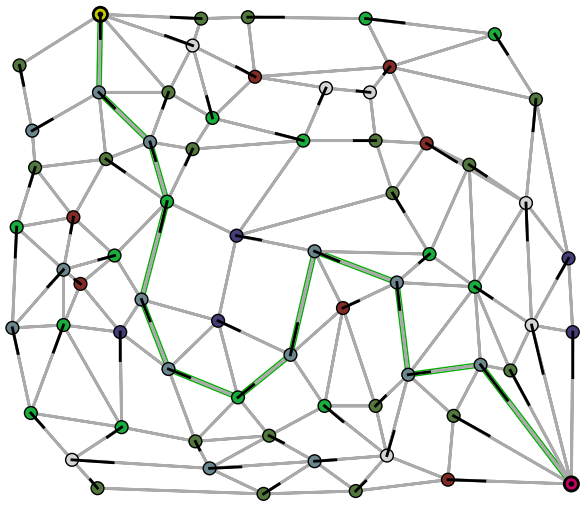

Fig. 3. Screenshot from the GRITSlab [1] graph planning software library of the example from Fig. 2.

The thick line starting from the robot (upper left-hand corner) and ending at the goal correctly identifies Route 3 as the optimal path over this weighted colored graph. The code needed to implement U_E is straight-forward and follows directly from equations 14-17.

VIII. Conclusions

We are able to find optimal paths over a directed colored graph $G = (V, E, W_E, C_V, C_E)$, by transforming it into (V, E, U_E), and applying standard edge relaxation algorithms. The resulting optimal path over (V, E, U_E) is also optimal over G in the sense that it matches our criteria

specified in Criteria III:1-5. This transformation has been motivated by the need to navigate a mobile robot through an outdoor environment populated with regions of relatively distinct and identifiable terrain.

REFERENCES

[1] Georgia robotics and intelligent systems laboratory. http://gritslab.ece.gatech.edu.
[2] R. C. Arkin. *Behavior-Based Robotics*. The MIT Press, 1998.
[3] R. Brockett. On the computer control of movement. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, volume 1, pages 534–40, 1988.
[4] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23, Mar. 1986.
[5] R. Diestel. *Graph Theory*. Springer-Verlag, third edition, 1997.
[6] M. Egerstedt, K. Johansson, J. Lygeros, and S. Sastry. Behavior based robotics using regularized hybrid automata. In *Proc. IEEE Conf. on Decision and Control*, volume 4, pages 3400–5, 1999.
[7] D. Hristu-Varsakelis and S. Andersson. Directed graphs and motion description languages for robot navigation. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, volume 3, pages 2689–94, 2002.
[8] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. Also available at http://msl.cs.uiuc.edu/planning/.