Computing Close to Optimal Weighted Shortest Paths in Practice

Nguyet Tran*, Michael J. Dinneen, Simone Linz
School of Computer Science, University of Auckland, New Zealand

*Email: ntra770@aucklanduni.ac.nz

Abstract

This paper proposes a new practical method for the weighted region problem (WRP). The objective of WRP is to find a minimum cost path between two vertices among different regions where each region incurs a traversal cost per unit distance.

Currently, there is no practical algorithm that solves this problem exactly. Among the approximation methods that solve instances of WRP, there is a limited number of algorithms that compute paths whose lengths are close to optimal, which we call very-close optimum paths. However, they are considered as theoretical methods. On the other hand, algorithms for solving WRP that can be applied to practical data sets (using decomposition ideas or heuristics) are not guaranteed to find a very-close optimum path within an acceptable amount of time.

In this paper, we consider an alternative method for solving WRP that exploits Snell's law of physical refraction. We compare the performance of our new algorithm with that of two existing algorithms, using at least 500 test cases for each such comparison. The experimental results show that our algorithm returns a very-close optimum weighted shortest path in reasonable time.

Introduction

- $W = (T, E, V)$: a continuous two-dimensional workspace, where T is a set of non-overlapping *regions*, E is the set of *edges* and V is the set of *vertices* of the regions in T.
- Each region $t_i \in T$ is a triangle, and assigned a unit *weight* (or cost) $w_i > 0$.
- For each edge $e_i \in T$, if e_i has two adjacent triangles t_a and t_b, the unit weight of e_i is $w(e_i) = \min(w_a, w_b)$, where w_a and w_b are the unit weights of t_a and t_b, respectively. Otherwise, if e_i is only on one triangle t_a, e_i is a *border edge*, and $w(e_i) = w_a$.
- Let p and q be two points on a region $t_i \in T$, $D(p, q) = w \cdot d(p, q)$ is the *weighted length* (or cost) between p and q, where w is the unit weight of t_i or the edge that the segment (p, q) is on, and $d(p, q)$ is the Euclidean distance between p and q.

For a pair of two vertices $u, v \in V$, **the weighted region problem (WRP)** asks for the minimum cost (or the weighted shortest) path $P^*(u, v) = (u = o_0, o_1, \ldots, o_k, o_{k+1} = v)$ such that the weighted length $\sum_{i=0}^{k} D(o_i, o_{i+1})$ is minimum, where every o_i, $i \in \{1, \ldots, k\}$, called a *crossing point*, can be a point on an edge in E or a vertex in V.

Applications

The regions in T can be smooth flat, desert, rocks, water, forest, grassland, etc. The energy-consuming levels of a robot can be different depending on the moving regions. Finding a weighted shortest path turns into finding an optimal energy path (or a minimum time path) for the robot. WRP is a classical path planning problem, with many applications in robotics, geographical planning and manufacturing.

Difficulties

The problem is unknown NP-hard or not. Currently, there is no known polynomial or exponential time algorithm for finding the exact weighted shortest path. The exiting algorithms to solve WRP are all approximations.

Existing approaches

- Exploiting Snell's law (impractical solutions)
- Using heuristic methods (unpredictable results)
- Applying decomposition ideas, with a grid of cells or a graph of Steiner-points (time-consuming for a very-close optimal result)

Our approach

Our target is a very-close optimal solution, thus we create the first practical solution exploiting Snell's law.

Experimental Results

- *Scenario 1: Compare against Quadratic Programming*
- *Scenario 2: Compare against the Steiner-Point method* (Placing from $m = 6$ to $m = 400$ discrete points on each edges in E, in 500 test cases)

Scenario 2:

Number of regions		5	10	15	20	25	30
Our method's average times		**0.02**	**0.11**	**0.37**	**0.75**	**1.82**	**3.03**
$m = 400$	average times	1.13	3.33	5.80	8.94	12.64	17.32
	$\%D$	0.00015%	0.00029%	0.00056%	0.0010%	0.0013%	0.0018%

- Our results are always shorter in weighted length.
- Our running times are faster in case a close to an optimal path is needed.

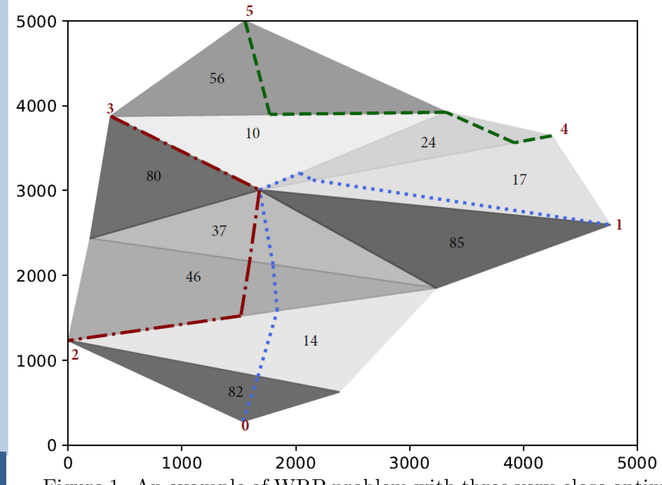


Figure 1. An example of WRP problem with three very-close optimum paths between vertices (0 and 1), (2 and 3) and (4 and 5).

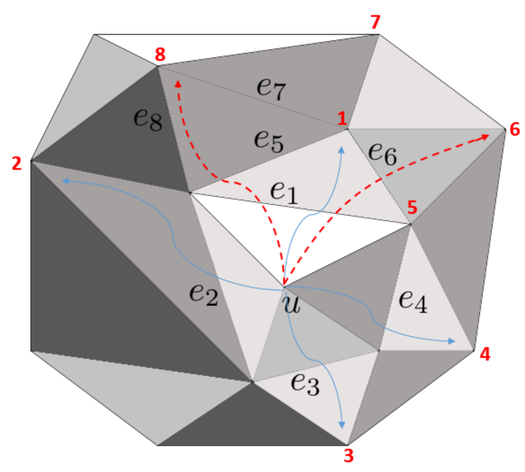
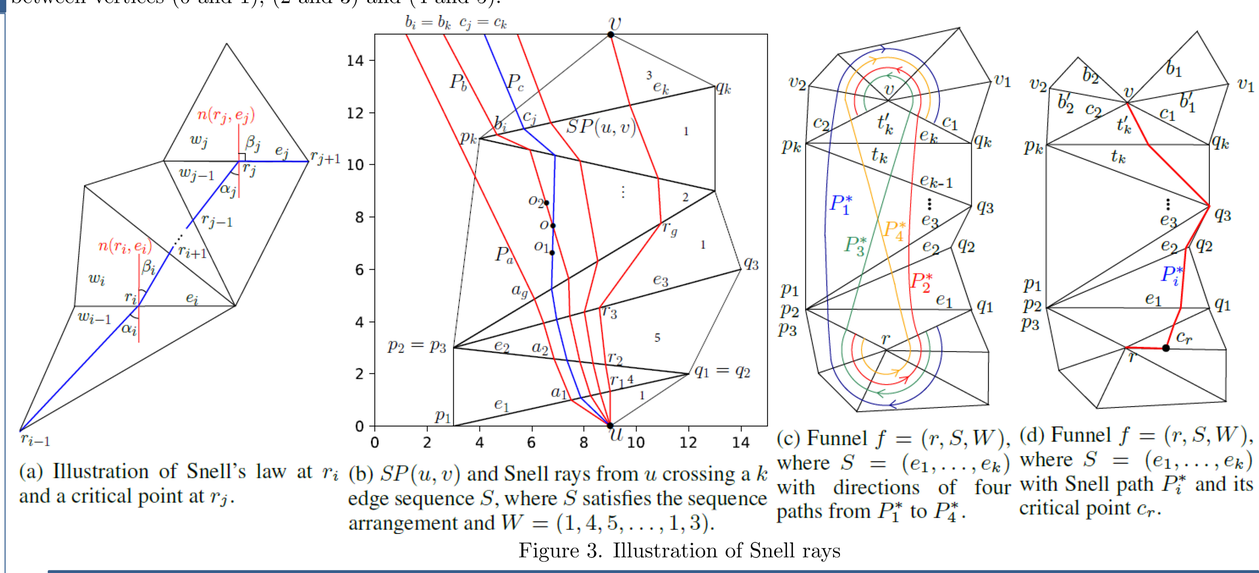


Figure 2. Illustration of building the D-graph.



(a) Illustration of Snell's law at r_i and a critical point at r_j.

(b) $SP(u, v)$ and Snell rays from u crossing a k edge sequence S, where S satisfies the sequence arrangement and $W = (1, 4, 5, \ldots, 1, 3)$.

(c) Funnel $f = (r, S, W)$, where $S = (e_1, \ldots, e_k)$ with directions of four paths from P_1^* to P_4^*.

(d) Funnel $f = (r, S, W)$, where $S = (e_1, \ldots, e_k)$ with Snell path P_i^* and its critical point c_r.

Figure 3. Illustration of Snell rays

Proposed method

1. Weighted shortest path crossing an edge sequence *S*

$S = (e_1, \ldots, e_k)$: an ordered sequence of k edges, where three consecutive edges in S cannot be in the same triangles. Otherwise, we present how to process it in the paper. $W = (w_0, \ldots, w_k)$: the weight list of S (see Figure 3b).

Snell's law: $P(u, v) = (u = r_0, r_1, \ldots, r_k, r_{k+1} = v)$ has the minimum weighted length crossing S if and only if at every crossing point r_i on e_i, for which r_i is not an endpoint of e_i, the following condition holds: $w_{i-1} \sin \alpha_i = w_i \sin \beta_i$ (see Figure 3a).

Snell ray: $P_a = (u, a_1, a_2, \ldots, a_g, R_g^a)$, where every a_i is a point on e_i, obeying Snell's law, and R_g^a is the *out-ray* of the path at $e_g \in S$ (see Figure 3b).

Snell path: $P(u, v) = (u = r_0, r_1, \ldots, r_k, r_{k+1} = v)$ is a Snell path if Snell's law is obeyed at each r_i, and r_i must be on the interior of e_i, which cannot be one of the two endpoints of e_i (see Figure 3b).

Two Snell rays $P_b = (u, b_1, \ldots, b_i, R_i^b)$ and $P_c = (u, c_1, \ldots, c_j, R_j^c)$, where $b_1 \neq c_1$, cannot intersect each other (the case in Figure 3b will never happen).

Finding the Snell path $P(u, v)$ crossing S (approximately):

From the middle point m_1 of e_1, create a Snell ray $P_m = (u, m_1, \ldots, R_g^m)$. If e_{g+1}, where $e_{k+1} = (v, v)$, is on the left (resp. right) of R_g^m, then the Snell ray that hits v must cross only the parts from p_i to m_i (resp. from m_i to q_i). Thus, we trim $e_i = (p_i, q_i)$ to (p_i, m_i) (resp. (m_i, q_i)). This process is iterated until P_m hits v, or all edges in S are trimmed such that $d(p_i, q_i) < \delta$, where δ be an extremely small value. However, if the Snell path crosses an endpoint of any original e_i in S, this process will be stopped.

2. Main algorithm

D-graph: an undirected graph (V_D, E_D), where $V_D = V \cup V_c$ with V_c being the set of critical points. An edge in E_D between two points u and v in V_D is created if there exists a Snell path between u and v, **which only cross the interiors of the edges in** E. The weight of every edge between u and v in E_D is the minimum weighted length among all possible Snell paths between u and v.

Funnel: $f = (r, S, W)$, where $S = (e_1, \ldots, e_k)$, $W = (w_0, \ldots, w_{k-1})$, $r \in V$ is the root of f, and the last edge $e_k \in S$ is the bottom of f (see Figure 3c).

- In a funnel f, the Snell path from r to v crossing S can go around the adjacent edges at r and v with critical points (at most four possible paths, P_1^* to P_4^*) (see Figure 3c and 3d). We present in the paper how to avoid finding all of these four Snell paths.
- After finding the Snell path from r to v, let $S_1 = S \circ (c_1)$, $S_2 = S \circ (c_2)$, and W_1 and W_2 be two weight lists with respect to S_1 and S_2, respectively. One of the following three conditions holds: (i) two new funnels $f_1 = (r, S_1, W_1)$ and $f_2 = (r, S_2, W_2)$ are created, (ii) only one new funnel $f_1 = (r, S_1, W_1)$ or $f_2 = (r, S_2, W_2)$ is created, (iii) no new funnel is created.

Main idea:

- Building a D-graph for $W = (T, E, V)$.
- Applying a shortest path graph algorithm on the D-graph to find the weighted shortest path between any pair of vertices.

Building D-graph:

Using a queue Q. For each vertex $u \in V$, initializing funnels $f = (r, S, W)$, where $r = u$ and S contains only one edge opposite to u. Pushing the funnels into Q. Popping one funnel $f = (r, S, W)$ out Q, we then find the Snell path from the root r to the vertex v, which is opposite to the last edge e_k in S, crossing S. If the Snell path between r and v crossing S exists, updating the D-graph. Then, the new funnels (at most two) corresponding to f are created and pushed into Q. The process will be stopped when Q is empty (see Figure 2).

Note: In practice, finding a Snell path will easily cross a vertex in V and stop. Thus, not too many funnels will be created.