Orienting an Edge-Bi-Weighted Graph to Minimize the Heaviest Path

1 Introduction

A *edge bi-weighted graph* is an undirected graph $G = (V, E)$, such that each edge $\{u, v\} \in E$ has a pair of (possibly different) weights $w(u, v)$ and $w(v, u)$ associated with its two possible orientations (u, v) and (v, u), respectively. An *orientation* of G creates a directed graph $\vec{G}$ by selecting for each undirected edge $\{u, v\} \in E$ exactly one of its two possible orientations. Denote by $\mathcal{O}(G)$ the set of all orientations of G.

The problem we address is the following.
Input: An edge bi-weighted graph G.
Output: An orientation of G that minimizes the weight of the heaviest of the resulting simple directed paths.

Actually, there remains an ambiguity in the specification of the cost function because it fails to specify whether the minimum is taken only over maximal simple paths or over all simple subpaths. To distinguish between these two possibilities we define two measures.

Definition 1.1. *Given a directed path* $\vec{P} =< v_0, \ldots, v_n >$ *let*

$$h_m(\vec{P}) = \sum_{k=0}^{n-1} w(v_k, v_{k+1}),\ h_s(\vec{P}) = \max_{0 \le i < j \le n-1} \sum_{k=i}^{j} w(v_k, v_{k+1}). \quad (1)$$

We extend these measures to an oriented graph $\vec{G}$

$$h_m(\vec{G}) = \max\{h_m(\vec{P}) \mid \vec{P} \text{ is a maximal simple path in } \vec{G}\}, \quad (2)$$
$$h_s(\vec{G}) = \max\{h_s(\vec{P}) \mid \vec{P} \text{ is a simple path in } \vec{G}\}. \quad (3)$$

The corresponding cost functions for orienting an undirected graph are

$$H_m(G) = \min\{h_m(\vec{G}) \mid \vec{G} \in \mathcal{O}(G)\}, H_s(G) = \min\{h_s(\vec{G}) \mid \vec{G} \in \mathcal{O}(G)\}. \quad (4)$$

To illustrate the differences between the two cost functions consider $\vec{P} =< v_0, v_1, v_2, v_3 >$, with $w(v_0, v_1) = 2, w(v_1, v_2) = -3, w(v_2, v_3) = 6$. Then $h_m(\vec{P}) = 5, h_s(\vec{P}) = 6$.

A very useful property of h_s is that it is monotone: the cost of a graph is never less than the cost of a subgraph.

Lemma 1.1. *Given any edge bi-weighted graph* G, $H_s(G) \ge H_s(G')$ *for any subgraph* G' *of* G.

Proof: It is clear from the definition (1) of h_s that $h_s(\vec{P}) \ge h_s(\vec{P}')$for any subpath $\vec{P}'$ of $\vec{P}$. □

This property sets h_m apart from h_s, as the above example shows: $h_m(\vec{P}) = 5 < h_m(\vec{P'}) = 6$ for the subgraph $\vec{P'} =< v_2, v_3 >$ of $\vec{P}$.

However, if all weights are non-negative h_m, too, is monotonic. In fact, in that case the two definitions coincide.

Lemma 1.2. *Suppose that G is an edge bi-weighted graph all of whose weights are non-negative. Then $h_s(\vec{G}) = h_m(\vec{G})$ for any orientation $\vec{G}$. In particular, $H_s(G) = H_m(G)$, and $\vec{G}$ is an optimal orientation of G with respect to h_s if and only if it is optimal with respect to h_m.*

Proof: From the definitions of h_m and h_s it is clear that for any path graph P $h_m(P) \leq h_s(P)$, and that the inequality is in fact an equality if all the weights are non-negative. This implies $H_s(G) = H_m(G)$.

Clearly any orientation of G that is optimal with respect to h_m is also optimal with respect to h_s. □

Conjecture: *Given an edge bi-weighted graph G, any orientation that is optimal with respect to h_m is also optimal with respect to h_s.*

In the following sections we will consider the following two problems for various classes of graphs.

Problem 1.1 (HS)**.** *Given a class of edge bi-weighted graphs find an algorithm to compute $H_s(G)$ for any G in the class.*

Problem 1.2 (HM)**.** *Given a class of edge bi-weighted graphs find an algorithm to compute $H_m(G)$ for any G in the class.*

2 An Algorithm for Linear Graphs

Given a bi-weighted linear graph P of length n we assume that its vertices are numbered from *0* to n, and denote the weights of edges $(i, i+1)$ and $(i+1, i)$ by $w(i, i+1)$ and $w(i+1, 1)$, respectively.

We describe next an algorithm for finding an optimal orientation of a linear graph. The high level version of the algorithm makes no use of the details of the cost function, be it h_m or h_s. The description will therefore omit these subscripts. However, upon implementing the high level version, the difference between the two cost functions leads to a surprising difference in running times: it is linear under h_s but quadratic under h_m.

Notation:

- $P_{i,j}$ is the sub-graph of P induced by the vertices $i, \ldots, j$, and $P = P_{0,n}$.
- $\vec{P_{i,j}}$ denotes the oriented version of $P_{i,j}$ in which all edges are directed to the right (i.e. of the form $(i, i+1)$), and similarly $\overleftarrow{P_{i,j}}$ denotes the oriented version of $P_{i,j}$ in which all edges are directed to the left.
- $H^r[i]$ is the value of an optimal orientaton of $P_{0,i}$ under the constraint that edge $\{i, i+1\}$ is directed towards $i+1$, and $H^\ell[i]$ is the value of an optimal orientaton of $P_{0,i}$ under the constraint that edge $\{i, i+1\}$ is directed towards i.

The algorithm uses dynamic programming, and its basic step is to locate the last breakpoint of an optimal solution, in the sense that its two edges are either both ingoing or both outgoing.

Algorithm BestCostPath (P):

1. $H^r[0] = H^\ell[0] = 0$;
2. for $j = 1$ to n do
   (a) $H^r[j] = \min_{0 \le i < j} \max\{H^\ell[i], h(\vec{P_{i,j}})\}$;
   (b) $H^\ell[j] = \min_{0 \le i < j} \max\{H^r[i], h(\overleftarrow{P_{i,j}})\}$;
3. return $\max\{H^r[n],\ H^\ell[n]\}$.

Turning now to the running time analysis, we note first of all that the values $h(\vec{P_{i,j}}), h(\overleftarrow{P_{i,j}})$ used in statements (2a) and (2b) can be computed in constant time if the cost function is h_m. Namely, if the $2n$ values $h_m(\vec{P_{0,j}})$, $h_m(\overleftarrow{P_{0,j}})$, $1 \le j \le n$ are precomputed in $O(n)$ time, then $h_m(\vec{P_{i,j}}) = h_m(\vec{P_{0,j}}) - h_m(\vec{P_{0,i}})$.

To prove that this is true also for the cost function h_s takes more doing.