Exploring Vertex and Edge Weighted Graphs

By Kenny Yu

1 Introduction

In graph theory, the traditional weighted graph consists of weights on edges only. Whereas weighing edges has many practical applications, weighing vertices as well also serve many purposes. In this paper, we will explore properties of a *doubly-weighted graph*—a graph in which both edges and vertices are weighted—and how they differ from an edge-only-weighted graph. Using these properties, we will explore and solve a problem by modeling it with a doubly-weighted graph.

2 Definitions

To begin our discussion of weighted graphs, we will define the notion of a *doubly-weighted graph*, a *vertex-weighted graph*, and an *edge-weighted graph*.

Doubly-Weighted Graph. Define a *doubly-weighted graph* $G = (V(G), \omega_V, E(G), \omega_E)$. The set $V(G)$ is called the *vertex set* of G, and elements of this set are called *vertices*. The *vertex weight function* $\omega_V : V(G) \rightarrow \mathbb{R}$ maps all vertices onto the set of real numbers. We denote the *weight* of a vertex v as $\omega_V(v)$. The set $E(G)$ is called the *edge set* of G, and elements of this set are called *edges*. Each edge e_i is defined by two vertices, v_j and v_k, such that $e_i = v_j v_k$. We say that this edge e_i is *incident* to vertices v_i and v_j, and we denote this by $e_i \,\alpha\, v_j$ and $e_i \,\alpha\, v_k$ respectively. Let $\delta(v)$ be the *degree* of vertex v—that is the number of edges incident to v. The *edge weight function* $\omega_E : E(G) \rightarrow \mathbb{R}$ maps all edges onto the set of real numbers. We denote the *weight* of an edge e as $\omega_E(e)$.

Vertex-Weighted Graph. Let graph $G = (V(G), \omega_V, E(G), \omega_E)$, and let ω_E be defined such that for all edges e in $E(G)$, $\omega_E(e) = 0$. Since all the weights of the edges are zero, it is as if the edges are not weighted at all. Thus we say that graph G is a *vertex-weighted graph* and its edge weights are not counted when considering weights.

Edge-Weighted Graph. Let graph $G = (V(G), \omega_V, E(G), \omega_E)$, and let ω_V be defined such that for all vertices v in $V(G)$, $vw(v) = 0$. Since all the weights of the vertices are zero, it is as if the vertices are not weighted at all. Thus we say that graph G is an *edge-weighted graph* and its vertex weights are not counted when considering weights.

3 Reducible and Irreducible Problems

3.1 Transforming a Doubly-Weighted Graph

Before we explore the aforementioned problem, we will examine properties of a doubly-weighted graph. Namely, we will show how all doubly-weighted graphs can be transformed into an equivalent directed edge-weighted graph. Different problems that optimize different sums or products, however, necessitate the use of doubly-weighted graphs, and we will call these problems *irreducible.*

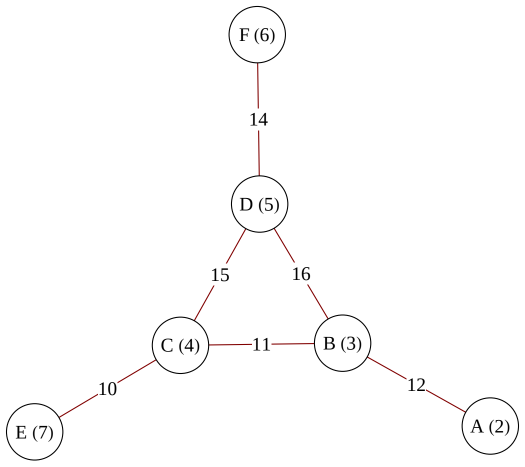

Figure 1: Graph G

To see that every doubly-weighted is equivalent to a directed edge-weighted graph, consider vertex B on the graph G in Figure 1. To remove the weight on vertex B, we construct a new graph G'. Let G' be identical to G, except vertex B is now replaced with two new vertices B' and B''. Construct a directed edge from B' to B'' with weight $\omega_V(B)$. This edge will be analogous to the vertex weight in G. Change all edges incident that were incident to B into directed edges leading into B', and let weights of these edges match their counterparts in the original graph G. Now create edges between B'' and the corresponding vertices adjacent to B in the original graph G. Thus all incoming edges that used to enter B enter B' and all outgoing edges that used to leave B leave through B''. All other edges are now directed edges going in both ways. Let the weights of these edges match their counterpart in G. Figure 2 shows this construction. This transformed graph is

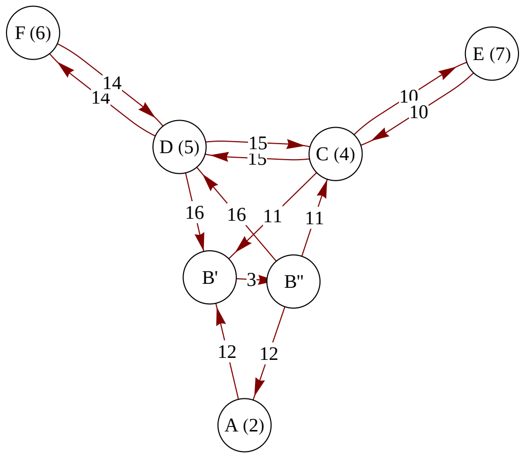

Figure 2: Graph G'

equivalent to the original graph in that the weights of paths and walks are preserved. All walks that enter B' must contain edge $B'B''$ and leave through B'', and thus must have $\omega_E(B'B'') = \omega_V(B)$ in the final weight. For example, consider the walk C, B, A, B, D on graph G. The weight of this walk can be found through the sum

$$\omega_V(C) + \omega_E(CB) + \omega_V(B) + \omega_E(BA) + \omega_V(A) + \omega_E(AB) + \omega_V(B) + \omega_E(BD) + \omega_V(D) = 68. \quad (1)$$

On graph G', the equivalent walk would be $C, B', B'', A, B', B'', D$. The weight of this walk can be found by replacing in equation (1) each $\omega_E(BX)$ with $\omega_E(B''X)$, $\omega_E(XB)$ with $\omega_E(XB')$, and $\omega_V(B)$ with $ew(B'B'')$ for all vertices X. Thus the weight of the equivalent walk in G' is

$$\omega_V(C)+\omega_E(CB')+\omega_E(B'B'')+\omega_E(B''A)+\omega_V(A)+\omega_E(AB')+\omega_E(B'B'')+\omega_E(B''D)+\omega_V(D) = 68.$$

If we use this algorithm on each of the remaining vertices, we will eventually get an equivalent directed edge-weighted graph with no weighted vertices. Thus all doubly-weighted graphs can be transformed into an equivalent directed edge-weighted graph.

3.2 Reducible and Irreducible Problems.

In the previous section, when we transformed the doubly-weighted graph into a directed edge-weighted graph, we noted that this transformation preserves the weights of walks and paths. Thus the problem of finding the shortest weighted path from a vertex v to a vertex u can be reduced to a simple directed edge-weighted problem that can be easily solved using Dijkstra's Algorithm. From this observation, we wonder what other problems can be reduced to an edge-weighted problem and do not need to be modeled using doubly-weighted graphs. From equation (1), we see that the total weight of the walk is

$$\begin{aligned}\omega_V(C)+\omega_E(CB)+\omega_V(B)+\omega_E(BA)+\omega_V(A)+\omega_E(AB)+\omega_V(B)+\omega_E(BD)+\omega_V(D)\\ =\omega_V(C)+\omega_E(CB)+2\omega_V(B)+2\omega_E(BA)+\omega_V(A)+\omega_E(BD)+\omega_V(D),\end{aligned}$$

which is a linear combination of the vertex weights and edge weights. From this observation, we can conjecture what form a *reducible problem* takes.

Definition: Problem. Given a graph G, a *problem* is an expression involving the weights of vertices and edges. To *solve* a problem, one must optimize this expression.

Definition: Reducible. Given a graph G, a problem for doubly-weighted graphs is *reducible* if it can be solved using algorithms strictly based on directed or non-directed edge-weighted graphs. A problem that is not reducible is said to be *irreducible*.

Conjecture 3.1. *A problem for doubly-weighted graph G is reducible if and only if the problem has an expression of the form*

$$c_1\omega_V(v_1)+c_2\omega_V(v_2)+\cdots+c_n\omega_V(v_n)+c_{n+1}\omega_E(e_1)+c_{n+1}\omega_E(e_2)+\cdots+c_{n+m}\omega_E(e_m)$$

where n and m are the number of vertices and edges in $V(G)$ and $E(G)$ respectively, $c_k\in\mathbb{Z}$ for all integers $1\le k\le n+m$, $v_i\in V(G)$ for all integers $1\le i\le n$, and $e_j\in E(G)$ for all integers $1\le j\le m$.

4 Traffic Grid Problem

Consider the following problem that we will model using doubly-weighted graphs:

Traffic Grid Problem In a town with m intersections spread across n roads that each connect two intersections, there are no traffic lights. Each intersection has an associated *danger number* that quantifies the danger level of an intersection, and each road has an associated *congestion number* that quantifies the amount of traffic on that road. The city planners wish to place l traffic lights at intersections across the city (l is less than or equal to the number of intersections). Placing a traffic light will make the intersection safer and thus decrease the intersection's danger number by a real number a that is less than the minimum danger number. However, the traffic light will increase the amount of traffic leading into that intersection and thus raise the congestion number of each road leading into that intersection by a real number b. How can the city minimize the sum of the products formed by multiplying each road's congestion number with the sum of the danger numbers of the intersections the road connects?

To model this problem, we create a graph G where the vertex set is the set of intersections and the edge set is the set of roads connecting these intersections. The vertex weight function maps these vertices to the corresponding intersection's danger number, which is a real number. The edge weight function maps the edges to the corresponding intersection's edge number, which is also a real number. The question thus becomes:

Problem Given graph G as constructed above, minimize the following expression:

$$S = \sum_{v_i v_j \in V(G)} \omega_E(v_i v_j) \left[\omega_V(v_i) + \omega_V(v_j)\right].$$

This expression clearly does not have the form

$$c_1\omega_V(v_1) + c_2\omega_V(v_2) + \cdots + c_n\omega_V(v_n) + c_{n+1}\omega_E(e_1) + c_{n+1}\omega_E(e_2) + \cdots + c_{n+m}\omega_E(e_m)$$

because we have products of edge weights and vertex weights. Thus from our conjecture, we can conclude that this problem is irreducible and cannot be solved with algorithms strictly based on edge-weighted graphs. To find an algorithm to solve this problem, we will solve an equivalent problem instead. Consider graph G in Figure 3.

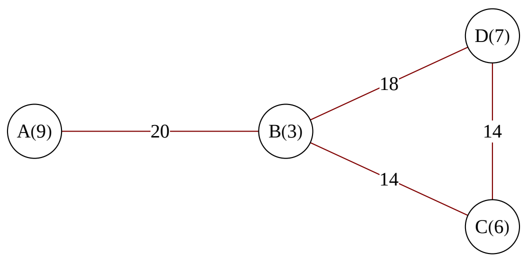

Figure 3: Graph G

If we examine the sum S, we note an interesting observation:

$$\begin{aligned} S &= \sum_{v_i v_j \in V(G)} \omega_E(v_i v_j)\left[\omega_V(v_i) + \omega_V(v_j)\right] \\ &= \omega_E(AB)[\omega_V(A) + \omega_V(B)] + \omega_E(BD)[\omega_V(B) + \omega_V(D)] + \omega_E(CD)[\omega_V(C) + \omega_V(D)] \\ &\quad +\omega_E(BC)[\omega_V(B) + \omega_V(C)] \\ &= \omega_V(A)[\omega_E(AB)] + \omega_V(B)[\omega_E(BC) + \omega_E(BD)] + \omega_V(C)[\omega_E(CB) + \omega_E(CD)] \\ &\quad +\omega_V(D)[\omega_E(DB) + \omega_E(DC)] \\ &= \sum_{v \in V(G)} \left(\omega_V(v) \sum_{\substack{e \,\alpha\, v \\ e \in E(G)}} \omega_E(e) \right). \end{aligned}$$

Thus the sum of the products formed by multiplying each edge weight with the sum of the incident vertex weights is equal to the sum of the products formed by multiply each vertex weight with the sum of the incident edge weights. Therefore we have the following theorem:

Theorem 4.1. *Given a doubly-weighted graph G,*

$$\sum_{v_i v_j \in V(G)} \omega_E(v_i v_j)\left[\omega_V(v_i) + \omega_V(v_j)\right] = \sum_{v \in V(G)} \left(\omega_V(v) \sum_{\substack{e \,\alpha\, v \\ e \in E(G)}} \omega_E(e) \right).$$

Proof. Consider all ordered pairs $(e_i, v_k) \in G$ formed by edges $e_i \in E(G)$ and vertices $v_k \in$

$V(G)$ where $e_i \,\alpha\, v_k$. Define the *weight product* of such an ordered pair (e_i, v_k) to be the product $\omega_E(e_i)\omega_V(v_k)$.

Consider each edge e_i. Each e_i is incident to exactly two other vertices, call them u_{i1} and u_{i2}. Thus $e_i = u_{i1}u_{i2}$ and is in exactly two ordered pairs and thus two weight products. Consider the sum of all weight products Q, and let m be the number of edges in $E(G)$. Then:

$$\begin{aligned}
Q &= \sum_{(e_i,v_k)\in G} \omega_E(e_i)\omega_V(v_k) \\
&= \omega_E(e_1)\omega_V(u_{11}) + \omega_E(e_1)\omega_V(u_{12}) + \omega_E(e_2)\omega_V(u_{21}) + \omega_E(e_2)\omega_V(u_{22}) \\
&\quad + \cdots + \omega_E(e_m)\omega_V(u_{m1}) + \omega_E(e_m)\omega_V(u_{m2}) \\
&= \omega_E(e_1)[\omega_V(u_{11}) + \omega_V(u_{12})] + \cdots + \omega_E(e_m)[\omega_V(u_{m1}) + \omega_V(u_{m2})] \\
&= \omega_E(u_{11}u_{12})[\omega_V(u_{11}) + \omega_V(u_{12})] + \cdots + \omega_E(u_{m1}u_{m2})[\omega_V(u_{m1}) + \omega_V(u_{m2})] \\
&= \sum_{v_iv_j\in V(G)} \omega_E(v_iv_j)\left[\omega_V(v_i) + \omega_V(v_j)\right].
\end{aligned}$$

Now consider each vertex v_k. Let $\delta(v_k)$ denote the degree of vertex v_k. Each vertex v_k is incident to exactly $\delta(v_k)$ edges, call them $s_{i1}, s_{i2}, ..., s_{i\delta(v_k)}$. Therefore v_k is in exactly $\delta(v_k)$ ordered pairs and thus $\delta(v_k)$ weight products. Consider the sum of all the weight products, and let n be the number of vertices in $V(G)$. Then:

$$\begin{aligned}
Q &= \sum_{(e_i,v_k)\in G} \omega_E(e_i)\omega_V(v_k) \\
&= \omega_E(s_{11})\omega_V(v_1) + \omega_E(s_{12})\omega_V(v_1) + \cdots + \omega_E(s_{1\delta(v_1)})\omega_V(v_1) \\
&\quad + \cdots + \omega_E(s_{n1})\omega_V(v_n) + \omega_E(s_{n2})\omega_V(v_n) + \cdots + \omega_E(s_{n\delta(v_n)})\omega_V(v_n) \\
&= \omega_V(v_1)[\omega_E(s_{11}) + \omega_E(s_{12}) + \cdots + \omega_E(s_{1\delta(v_1)})] \\
&\quad + \cdots + \omega_V(v_n)[\omega_E(s_{n1}) + \omega_E(s_{n2}) + \cdots + \omega_E(s_{n\delta(v_n)})] \\
&= \sum_{v\in V(G)} \left(\omega_V(v) \sum_{\substack{e\,\alpha\, v \\ e\in E(G)}} \omega_E(e) \right).
\end{aligned}$$

Combining the two equations we just derived, we have that

$$Q = \textstyle\sum_{v_iv_j\in V(G)} \omega_E(v_iv_j)\left[\omega_V(v_i) + \omega_V(v_j)\right] = \sum_{v\in V(G)} \left(\omega_V(v) \sum_{\substack{e\,\alpha\, v \\ e\in E(G)}} \omega_E(e) \right)$$

as desired. $\square$

Now that we have this equality, in order to solve the Traffic Grid Problem, we need to only minimize the value of the expression

$$\sum_{v\in V(G)} \left(\omega_V(v) \sum_{\substack{e\,\alpha\,v \\ e\in E(G)}} \omega_E(e) \right).$$

To place traffic lights into the city, we need to define a transformation L on the graph G that represents the addition of a traffic light.

Definition: Transformation *L(v)* Given a graph G, define the transformation L on the graph G given a parameter vertex v as follows:

- $\omega_V(v)$ is set to the difference of its old value and a.
- For all $e \in E(G)$ and $e\,\alpha\,v$, $\omega_E(e)$ is set to the sum of its old value and b.

Consider the following product $P = \omega_V(v) \sum_{\substack{e\,\alpha\,v \\ e\in E(G)}} \omega_E(e)$ and suppose we applied the transformation $L(v)$. Let $T = \sum_{\substack{e\,\alpha\,v \\ e\in E(G)}} \omega_E(e)$. Then the updated product would be:

$$\begin{aligned}
(\omega_V(v) - a) \sum_{\substack{e\,\alpha\,v \\ e\in E(G)}} (\omega_E(e) - b) &= (\omega_V(v) - a)(T + b\delta(v)) \\
&= \omega_V(v)T + b\delta(v)\omega_V(v) - aT - ab\delta(v) \\
&= P - aT + b\delta(v)[\omega_V(v) - a].
\end{aligned}$$

Since a and b are constants with respect to v, and since the values $P, -aT$ are invariants in the sum, then these values do not matter when optimizing this product. Thus, the only expression that we need to minimize is

$$\delta(v)[\omega_V(v) - a].$$

We can easily find the vertex v in graph G that minimizes this product at every step with an $O(n)$ operation where n is the number of vertices in G. Thus, a *greedy algorithm* that repeatedly applies the transformation L to the vertex that minimizes the product each time will minimize the expression S.

Definition: Greedy Algorithm A *greedy algorithm* is an algorithm that takes the best choice at every possible step. In the case of this problem, the best choice at every step would be to apply the transformation L to vertex v that minimizes the expression S.

Solution to Problem: Greedy Algorithm The algorithm to solve the Traffic Grid Problem is as follows:

1. Let the set $R = \{\}$.
2. Find a vertex $v \notin R$ that minimizes the expression $\delta(v)[\omega_V(v) - a]$. Apply the transformation $L(v)$. Add v to R.
3. While $|R| < l$ where l is the number of traffic lights alotted, repeat step (2).

Theorem 4.2. *The algorithm described above solves the Traffic Grid Problem.*

Proof. We will use induction the prove that this algorithm works.

Base Case. If $l = 1$, then applying this algorithm finds the vertex v that minimizes $\delta(v)[\omega_V(v) - a]$ which then minimizes $(\omega_V(v) - a) \sum_{\substack{e \,\alpha\, v \\ e \in E(G)}} (\omega_E(e) - b)$, and thus minimizes the resulting quantity $\sum_{v \in V(G)} \left(\omega_V(v) \sum_{\substack{e \,\alpha\, v \\ e \in E(G)}} \omega_E(e) \right)$ as desired.

Inductive Hypothesis. Assume that this algorithm works for $l = r$ for some positive integer $r < n$ where n is the number of vertices. Then after applying this algorithm r times, S is minimized. Call the resulting graph G'. Now view G' as some arbitrary doubly-weighted graph. If we apply this transformation one more time, this is equivalent to the base case, which we already show minimizes S. Since S is still minimized after $r + 1$ transformations, then the greedy algorithm minimizes S and thus solves the Traffic Grid Problem. $\square$

5 Conclusion

In this paper, we examined properties of doubly-weighted graphs and used these properties to help us solve the Traffic Grid Problem. We constructed and analyzed the transformation of doubly-weighted graphs into directed edge-weighted graphs, and we conjectured the necessary and sufficient criteria in order for a problem to be reducible. We determined that the Traffic Grid Problem was irreducible and thus could not use well-known algorithms to solve this problem. We created a greedy algorithm after modelling the problem with a doubly-weighted graph. The fact that this relatively complex problem was solved with a simple polynomial-time algorithm suggests that many more difficult problems can be easily solved by using doubly-weighted graphs.