A Factor-Two Approximation Algorithm for Two-Dimensional Phase-Unwrapping

Reuven Bar-Yehuda and Irad Yavneh
Department of Computer Science
Technion—Israel Institute of Technology
Haifa 32000, Israel

October 19, 2004

Abstract

Two-dimensional phase unwrapping is the problem of deducing unambiguous "phase" from values known only modulo 2π. Many authors agree that the objective of phase unwrapping should be to find a (weighted) minimum of the number of places where adjacent discretized phase values differ by more than π. This problem, which is known to be NP-hard, is of considerable practical interest, largely due to its importance in interpreting data acquired with synthetic aperture radar (SAR) interferometry. Consequently, many heuristic algorithms for its approximate solution have been proposed. Here we present a novel approach to this problem, based on the local-ratio principle, which guarantees a solution whose cost is at most twice the minimum sought.

1 Introduction

Phase unwrapping is a computational process whereby a surface, Φ, is reconstructed from its so-called "wrapped" form, Ψ. Here, $\Phi = \Phi(\mathbf{x})$ is a real-valued function, commonly with $\mathbf{x} \in \mathbb{R}^2$. In the absence of noise, $\Psi(\mathbf{x})$ is equal to $\Phi(\mathbf{x}) + 2\pi\mathbf{k}(\mathbf{x})$, where $\mathbf{k}(\mathbf{x})$ is an integer-valued function such that $-\pi \leq \Psi < \pi$. The problem of phase unwrapping has drawn considerable interest. It is an essential part of many coherent signal processing applications, for example, Interferometric Synthetic Aperture Radar (SAR) and optical interferometers. Coherent processing is based on a signal property known as "phase", which often relates to some physical quantity such as surface topography. However, the actual phase values cannot be extracted directly from the physical signal, since phase influences the signal through its principal values that lie between $\pm\pi$ radians. All we are given is the wrapped phase, that is, the phase values forced into the interval $[-\pi, \pi)$ by a modulo 2π operation.

The unwrapping process is aimed at providing an estimation of the actual phase function, Φ, given the wrapped function, Ψ. This turns out to be a difficult problem. Many phase unwrapping algorithms were developed during the last twenty years. These algorithms may be divided into two general classes: path-following and minimum-norm methods [5]. The path-following, also called residue-cut "tree" algorithms, unwrap by integrating the gradients of the wrapped phase (approximated by wrapped differences) along paths that avoid problematic areas where these differences are inconsistent. The minimum-norm algorithms, on the other hand, minimize some norm of the distance between the local differences of the estimate of Φ and the corresponding wrapped values computed from Ψ. Often, the L^p norm is used with $0 < p \leq 2$. The problem generally becomes more difficult as p is decreased, but there is an agreement amongst many authors that reconstructions generally improve as p tends to zero [3, 4, 5, 6, 8]. For $p \to 0$, the minimum norm and residue-cut approaches coincide. This is the case we are considering in this paper. We follow the approach of previous authors who developed heuristic algorithms for this problem, especially [3, 4, 5, 6, 8]. Our aim in this paper is to formalize and somewhat generalize this approach and, mainly, to present an algorithm which yields a solution whose cost is at worst twice that of the optimum.

2 Problem Formulation

Suppose that the wrapped phase values are given at a set of discrete points, which we consider as vertices of a graph, $G = (V, E)$. (In practical applications, the graph is usually a simple planar grid). Each vertex, $v \in V$, is associated with a phase value, $\tilde{\phi}[v]$. For each vertex $v \in V$, we are given the "wrapped" values of $\tilde{\phi}[v]$, denoted by $\psi[v]$. For convenience, we transform the problem such that the wrapping yields values in the range $[0, 1)$, rather than $[-\pi, \pi)$. Hence, "wrapping" now means taking the fractional part, while "unwrapping" means adding an integer. Given the wrapped vector, $\psi \in [0, 1)^{|V|}$, we would like to find a phase vector, $\phi \in \mathbb{R}^{|V|}$, which is our best estimate of the original unknown phase, $\tilde{\phi}$. Clearly, some assumptions are required in order to evaluate different solutions. Unwrapping algorithms are invariably based on some smoothness assumptions regarding the original surface. Typically, the smoothness is evaluated by some measure of distance between the phase values at adjacent vertices. Additionally, some given edge-based non-negative weight function, $w : E \to \mathbb{R}^+$, is often introduced, which represents the degree of our confidence in the smoothness assumption at the corresponding location. This general problem can formally be written as follows.

$$\begin{array}{rll} \text{Minimize over } \phi\text{:} & \sum_{\{u,v\}\in E} & w(\{u,v\})\ |distance(\phi[u], \phi[v])|, \\ & \text{subject to} & \phi - \psi \in \mathbb{Z}^{|V|}, \end{array} \qquad (1)$$

where $w(\{u, v\})$ is the given weight associated with the edge $\{u, v\}$. Unwrapping algorithms differ mainly by the choice of distance measure. Many authors agree that the best choice is the so-called L^0-norm measure, corresponding to

$$distance(a, b) = \begin{cases} 1, & b - a \geq \frac{1}{2} \\ -1, & b - a \leq -\frac{1}{2} \\ 0, & \text{otherwise.} \end{cases} \qquad (2)$$

An edge, $\{u, v\}$, for which $|distance(\phi[u], \phi[v])| = 1$, is said to constitute a "violation" (often referred to as a "discontinuity" in the literature). Henceforth, we refer to the problem defined

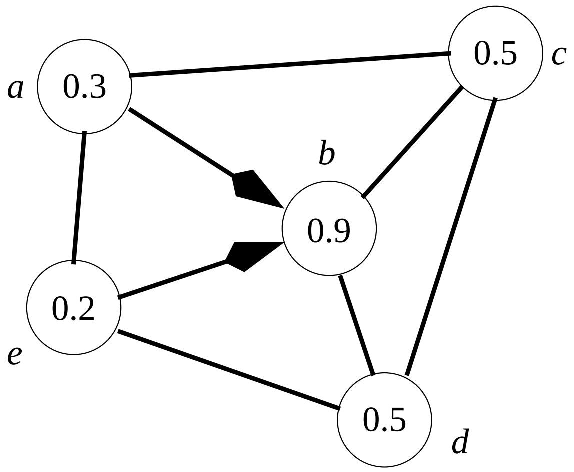

Figure 1: *An example of the graph corresponding to a wrapped image. The numbers represent ψ values, and an arrow from vertex u to vertex v indicates that $d(u,v) = -d(v,u) = 1$.*

by (1,2), aimed at minimizing the weight of violations, as the Phase-Unwrapping Problem (PUP).

Remark 1 *Edges, $\{u,v\}$, satisfying $|\psi[u] - \psi[v]| = 1/2$, will evidently constitute a violation in* any *solution, and therefore need not be taken into account when attempting to solve the PUP. Thus, we assume that all such edges are omitted from the input.*

In the sequel we transform the PUP, first into an edge-deletion problem, and then, for planar graphs, into a forest-satisfaction problem in the dual graph (cf. [7]).

2.1 From PUP to an edge-deletion problem

For adjacent vertices, $u, v \in V$, denote

$$d(u,v) \equiv distance(\psi[u], \psi[v])\,.$$

Denoting $\Delta = \phi - \psi$, one can readily see that the following problem is equivalent to PUP.

PUP′ : Given a graph, $G = (V, E)$, with, for every edge, $e \in E$, a weight, $w(e) \in \mathbb{R}^+$, and, for every pair of adjacent vertices, $u, v \in V$, a distance, $d(u,v) = -d(v,u) \in \{-1, 0, 1\}$,

$$\text{Minimize over } \Delta \in \mathbb{Z}^{|V|}\text{:} \quad \sum_{d(u,v) \neq \Delta[u] - \Delta[v]} w(\{u,v\})\,. \tag{3}$$

This simplifies the formulation, allowing us to work with integers instead of reals.

Consider for example Fig. 1. The numbers indicate the wrapped phase values, ψ. An arrow from vertex u to vertex v indicates that $d(u,v) (= -d(v,u)) = 1$. Let $P = v_0, v_1, \ldots, v_{\ell-1}$ be a path in the graph. Denote by $d(P)$ the total distance from v_0 to $v_{\ell-1}$ via the path, i.e.,

$$d(P) = d(v_0, v_1, \ldots, v_{\ell-1}) = d(v_0, v_1) + d(v_1, v_2) + \ldots + d(v_{\ell-2}, v_{\ell-1})\,.$$

A cycle is called *d-balanced* if its total distance (in any direction) is zero, for example, in Fig. 1,

$$d(abcdbea) = d(ab) + d(bc) + d(cd) + d(db) + d(be) + d(ea) = 1 + 0 + 0 + 0 + (-1) + 0 = 0\,.$$

A graph is said to be d-balanced if all its cycles are d-balanced. The graph in Fig. 1 is *not* d-balanced, as, for example, the cycle $(abca)$ is not d-balanced.

Proposition 1 *A connected graph G is d-balanced if and only if, for every two paths, P and Q, sharing the same start vertex and the same end vertex, $d(P) = d(Q)$.*

Proof. This is an immediate result of the fact that the two paths can be combined into a d-balanced cycle. □

For a connected d-balanced graph, G, we can now define $d(u,v)$ for *any* two vertices to be the length of an arbitrary path, whose start vertex is u and end vertex is v, since, by Proposition 1, all such paths have the same length. For d-balanced graphs, a zero-cost solution to PUP$'$ exists and can easily be obtained by the following algorithm. (The algorithm assumes a connected graph. For graphs that are not connected, it can be applied independently to each connected component).

Flood-Fill Algorithm: FF(G, w, d). Input: a graph, $G = (V, E)$, with, for every edge $e \in E$, a weight $w(e) \in \mathbb{R}^+$, and, for every pair of adjacent vertices $u, v \in V$, a distance, $d(u,v) = -d(v,u) \in \{-1, 0, 1\}$. Assumption: G is connected and d-balanced. Output: a solution, Δ, to PUP$'$.

1. Select an arbitrary vertex, r.
2. For each vertex v, define $\Delta[v] = d(v, r)$.

Proposition 2 *After the execution of the Flood-Fill algorithm, $\Delta[u] - \Delta[v] = d(u,v)$ for every pair of vertices, u and v.*

Proof. Since G is connected, there exists a path P_{uv} from u to v, a path P_{vr} from v to r, and a path P_{ru} from r to u. These three paths can be concatenated into a cycle, which is d-balanced by assumption, and therefore $d(u,v) + d(v,r) + d(r,u) = 0$. By the algorithm, $d(u,v) + \Delta[v] - \Delta[u] = 0$. □

Corollary *The Δ vector produced by the Flood-Fill algorithm yields*

$$\sum_{d(u,v) \neq \Delta[u] - \Delta[v]} w(\{u,v\}) = 0\,,$$

and is therefore a zero-weight solution to (3).

We next define the Minimal Balancing Edge Deletion (BED) problem, and show it to be equivalent to PUP$'$:

Min BED Problem: Given a graph, $G = (V, E)$, with, for every edge, $e \in E$, a weight, $w(e) \in \mathbb{R}^+$, and, for every pair of adjacent vertices, $u, v \in V$, a distance, $d(u, v) = -d(v, u) \in \{-1, 0, 1\}$,

$$\begin{aligned} \text{Minimize over } F \subseteq E\text{:} \quad & \textstyle\sum_{e\in F} \; w(e), \\ \text{subject to:} \quad & \tilde{G} = (V, E - F) \quad \text{is } d\text{-balanced}\,. \end{aligned} \tag{4}$$

Proposition 3 *Problems* PUP′ *and* Min BED *are equivalent up to a linear-time reduction.*

Proof. Let F be a feasible solution to Min BED (i.e., $\tilde{G} = (V, E - F)$ is a d-balanced graph). Apply the FF algorithm on each connected component of $\tilde{G}$, obtaining a zero-cost solution on $\tilde{G}$. This yields a solution to PUP′ on G whose cost is at most $\sum_{e\in F} \; w(e)$. For the other direction, let Δ be a feasible solution to PUP′ (i.e., $\Delta \in \mathbb{Z}^V$). Define $F = \{u, v\} : \Delta[u] - \Delta[v] \neq d(u, v)$. The graph $\tilde{G} = (V, E - F)$ is evidently d-balanced, because the distance of every cycle consists of the sums of $\Delta[v]$ and $-\Delta[v]$ for every vertex, v, in the cycle. □

Problem Min BED is known to be hard. In fact, it has been shown to be NP-hard even for the restricted case of rectilinear planar graphs [4]. We are therefore interested in approximation algorithms to this problem. However, we are unaware of an approach that will yield a constant-factor approximation to the Min BED problem on general graphs. Since the motivating problem, PUP, deals in practice with planar graphs, we concentrate on this case. In Section 3 we develop a factor-two approximation algorithm for the planar-graph case.

2.2 From edge deletion to cycle covering

For later convenience, we reformulate problem Min BED from a deletion problem into a covering problem. $F \subseteq E$ is called a d-Unbalanced-Cycle Covering (d-UBCC) if every cycle that is not d-balanced contains at least one edge that is in the set F.

Min d-UBCC Problem: Given a graph, $G = (V, E)$, with, for every edge, $e \in E$, a weight, $w(e) \in \mathbb{R}^+$, and, for every pair of adjacent vertices, $u, v \in V$, a distance, $d(u, v) = -d(v, u) \in \{-1, 0, 1\}$,

$$\begin{aligned} \text{Minimize over } F \subseteq E\text{:} \quad & \textstyle\sum_{e\in F} \; w(e), \\ \text{subject to:} \quad & F \text{ is } d\text{-UBCC}\,. \end{aligned} \tag{5}$$

Obviously, problems Min d-UBCC and Min BED are equivalent, as we have simply replaced deletion by covering.

The dual graph. Let $G = (V, E)$ be a planar graph, with, for every $e \in E$ a weight $w(e) \in \mathbb{R}^+$, and, for every pair of adjacent vertices, $u, v \in V$, a distance, $d(u, v) = -d(v, u) \in \{-1, 0, 1\}$. It is convenient to pursue our problem in the framework of the *dual graph*, similarly to the approach for PUP originally proposed in [8], which is the basis for residue-cut algorithms. Let $G' = (V', E')$ be the dual graph of a planar embedding of G, where

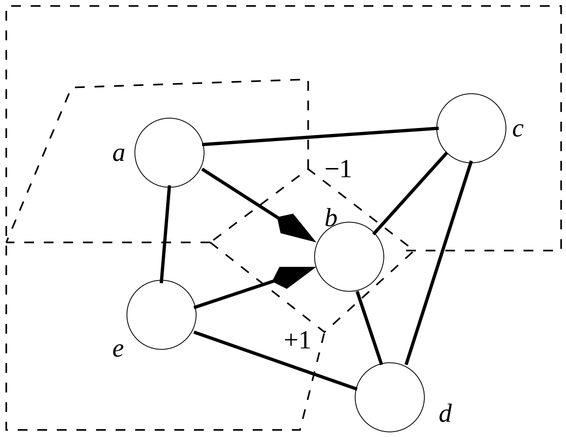

Figure 2: *An example of the primal (solid edges) and dual (dashed edges) graphs. The numbers at the vertices of the dual graph denote charges.*

V' is the set of faces in the planar embedding, and E' is the set of adjacent faces in the embedding. For each $e' \in E'$, let $e \in E$ be the edge in G which separates the adjacent faces that comprise e'. Define $w(e') = w(e)$. With every vertex $v' \in V'$, we associate a *charge*, $c(v')$, which is the distance of the clockwise cycle around the face in G which corresponds to the vertex v' in G'. (This charge is often referred to as "residue" in the literature). Fig. 2 shows an example of a graph and its dual. The numbers at the vertices of the dual graph are the charges. For example, the clockwise cycle in the primal graph, $acba$, is of length -1. Therefore, the charge at the corresponding dual vertex is -1.

A vertex set $U' \subseteq V'$ is called c-balanced if the sum of all its charges is zero. A graph is called c-balanced if each of its connected components is c-balanced. $F' \subseteq E'$ is called a c-Unbalanced-Cut Covering (c-UBCC) if, for every non-c-balanced set, $U' \subseteq V'$, F' contains at least one edge of the cut of U'. This leads to the following problem:

Min c-UBCC Problem: Given a graph, $G' = (V', E')$, with, for every edge, $e' \in E'$, a weight, $w(e') \in \mathbb{R}^+$, and, for every vertex, $v' \in V'$, a charge, $c(v') \in \mathbb{Z}$,

$$\begin{aligned} \text{Minimize over } F' \subseteq E'\text{:} \quad & \textstyle\sum_{e' \in F'} w(e'), \\ \text{subject to:} \quad & F' \text{ is } c\text{-UBCC}\,. \end{aligned} \tag{6}$$

The two problems, Min d-UBCC and Min c-UBCC, are equivalent. To show this, we prove the following proposition.

Proposition 4 *Let $G = (V, E)$ be a planar graph and let $G' = (V', E')$ be the dual graph of a planar embedding of G. Let $F \subseteq E$, and let $F' \subseteq E'$ be the dual of F. Then, F is d-UBCC in G if and only if F' is c-UBCC in G'.*

Proof. Consider a simple non-d-balanced cycle, and let $U' \subseteq V'$ be the set of vertices enclosed by this cycle in the planar embedding. The sum of charges of the vertices in U' is equal to the sum of the clockwise distances of the cycles surrounding each of the vertices. However, the latter sum is equal to the clockwise distance of the surrounding cycle, because all other (internal) edges cancel pairwise. Therefore, the total charge is nonzero, and if the cut is covered by F', then the non-d-balanced cycle in G is covered by F. For the other direction, let U' be a non-c-balanced set of vertices whose cut is not covered by F'. We need to show that there exists a non-d-balanced cycle in G, which is not covered by F. Construct a directed graph, $\vec{G} = (V, \vec{E})$, where $\vec{E}$ is the union of all the edges obtained from clockwise cycles around the faces corresponding to U'. Delete from this graph all anti-parallel edges. Clearly, the remaining set of edges can be partitioned into simple directed cycles. Furthermore, since the sum of distances of the cycles comprising $\vec{G}$ is non-zero by assumption, one of these simple cycles must have non-zero distance in G. However, the dual of any edge in this cycle is in the cut, and therefore it is not covered by F'. Hence, the edge itself is not covered by F. □

3 A 2-approximation algorithm

Since the Min c-UBCC Problem is NP-hard even for planar graphs, we develop an approximation algorithm. Our algorithm does not expolit the planarity property, and therefore it is applicable to general graphs. Hence, and also for simplicity of notation, we restate the Min c-UBCC Problem for general graphs, dropping the primes (that were associated with the planar dual graph):

Min c-UBCC Problem: Given a graph, $G = (V, E)$, with, for every edge, $e \in E$, a weight, $w(e) \in \mathbb{R}^+$, and, for every vertex, $v \in V$, a charge, $c(v) \in \mathbb{Z}$,

$$\begin{aligned} \text{Minimize over } F \subseteq E: &\quad w(F)\,, \\ \text{subject to:} &\quad F \text{ is } c\text{-UBCC}\,, \end{aligned} \tag{7}$$

where $w(F) \equiv \sum_{e \in F} w(e)$.

All discussion in the remainder of this paper is assumed to be in the context of this problem and its input and notation. We will use the term "feasible solution" (or simply "solution") to indicate a set of edges which is c-UBCC, and "optimal solution" to indicate a minimal-weight feasible solution, i.e., one which solves the Min c-UBCC problem. Let $F^* \subseteq E$ denote an optimal solution to the problem. Given a constant $r \geq 1$, a set $F \subseteq E$ is called an r-approximation if it is a feasible solution and satisfies $w(F) \leq r\, w(F^*)$. An algorithm is called an r-approximation algorithm for the problem if it produces an r-approximation. Evidently, an r-approximation algorithm for this problem implies an r-approximation algorithm for the original problem, PUP, with the same time complexity.

We say that a feasible solution, $F \subseteq E$, is "clean" if no proper subset of F is feasible. Clearly, it suffices to consider clean solutions, because eliminating unnecessary edges from a non-clean solution cannot increase its weight. Clean solutions evidently satisfy the following properties.

Proposition 5 *Let F be clean, and let G_F be the subgraph of G induced by F. Then,* *1. G_F is cycle-free (i.e., a forest).*

2. Every leaf in G_F is charged (i.e., has nonzero charge).

We next turn our attention to the weight function. Our approach will require us to construct weight functions with the following special property. A weight function is called r-effective if *every* clean solution is an r-approximation with respect to it. The idea is to decompose the actual weight function, w, into a sum of r-effective weight functions. This process is based on the local-ratio theorem (see, e.g., [2, 1]):

Theorem 1 *Let w_1, w_2, w be weight functions satisfying $w = w_1 + w_2$. Then, if F is an r-approximation both with respect to w_1 and with respect to w_2, then it is also an r-approximation with respect to w.*

Proof. Let F_1^*, F_2^*, and F^* be optimal solutions to the problem with resepect to w_1, w_2, and w, respectively. Then,

$$\begin{aligned} w(F) &= w_1(F) + w_2(F) && [w = w_1 + w_2] \\ &\le r\,w_1(F_1^*) + r\,w_2(F_2^*) && [\text{definition of } r\text{-approximation}] \\ &\le r\,w_1(F^*) + r\,w_2(F^*) && [\text{definition of optimality}] \\ &= r\,w(F^*) && [w = w_1 + w_2] \end{aligned}$$

□

The general idea in applying this theorem is to select an appropriate r-effective w_1, and to use it to reduce w by recursively finding an r-approximate solution with respect to $w_2 = w - w_1$. Thus, the main task is to construct a nontrivial (i.e., not identically zero) r-effective weight function, w_1. In our case, the best (smallest) r achieved is $r = 2$.

3.1 A 2-effective weight function

A weight function, $\hat{w}$, is said to be *homogeneous* if, for each edge, $\{u, v\}$, it satisfies:

$$\hat{w}(\{u,v\}) \in \begin{cases} \{0\}, & \text{if } c(u) = c(v) = 0\,, \\ \{1\}, & \text{if } c(u) = 0,\ c(v) \neq 0\,,\ \text{or } c(u) \neq 0,\ c(v) = 0\,, \\ [1,2], & \text{if } c(u) > 0,\ c(v) > 0\,,\ \text{or } c(u) < 0,\ c(v) < 0\,, \\ \{2\}, & \text{if } c(u)c(v) < 0\,. \end{cases} \tag{8}$$

Theorem 2 *A homogeneous weight function, $\hat{w}$, is 2-effective.*

Proof. Let F be a clean solution, and denote by $n(F)$ the number of vertices with non-zero charge in the graph induced by F. To prove the theorem, it is sufficient to show

$$n(F) \le \hat{w}(F) \le 2n(F)\,. \tag{9}$$

Let $T \subseteq F$ be any tree in the forest induced by F. Clearly, to prove (9), it suffices to show that

$$n(T) \le \hat{w}(T) \le 2n(T)\,, \tag{10}$$

and then sum up over all the trees in the forest induced by F. Since F is clean, it is sufficient to show (10) for any tree satsifying: (1) All leaves of T are charged; (2) There exists at least one positive-charged vertex and at least one with negative charge. We call trees satisfying the latter property *heterogeneous*. We prove (10) by induction on $n(T)$.

Base: $n(T) = 2$. In this case, T must be a tree with exactly two leaves with opposite-signed charges. Case 1: The two leaves are adjacent via an edge, e, and therefore, for the homogeneous weight function, $\hat{w}(T) = \hat{w}(e) = 2$, which satisfies (10). Case 2: The two leaves are not adjacent, therefore each of the two edges that are adjacent to the leaves have weight 1, and all other edges have weight 0. Therefore, $\hat{w}(T) = 1 + 1 = 2$, which again satisfies (10).

Step: $n(T) > 2$. Let v_0 be a leaf whose removal leaves the tree heterogeneous. (It is easy to show that such a leaf always exists). Let $P = v_0, v_1, \ldots, v_t$ be the longest path such that all vertices but the endpoints are uncharged and have degree 2. For the homogeneous weight function, it is easy to show (case by case) that the sum of all weights in this path is in the interval $[1, 2]$. Let T' be the tree that is obtained from T by deleting the path (but not v_t), that is, deleting v_0 and all the vertices leading from v_0 to v_t, not including v_t itself. Clearly, T' is heterogeneous and all its leaves are charged. Thus, by the induction hypothesis,

$$n(T') \leq \hat{w}(T') \leq 2n(T').$$

Summing this with $1 \leq \hat{w}(P) \leq 2$ yields

$$n(T') + 1 \leq \hat{w}(T') + \hat{w}(P) \leq 2n(T') + 2.$$

Now, since $n(T) = n(T') + 1$ and $T = T' \cup \{\text{edges of } P\}$, we obtain (10). □

3.2 The local-ratio algorithm

The construction of the 2-effective weight function, $\hat{w}$, is the essential step in the development of the 2-approximation algorithm, which is defined as follows.

Algorithm $\mathbf{LR}(G, w, c)$

If $n(G) = 0$, return $\emptyset$.

If there exists an edge e such that $w(e) = 0$, do:
  Let (G', w', c') be the instance obtained by contracting e.

  $F' \leftarrow \mathbf{LR}(G', w', c')$.
  If F' is a feasible solution with respect to (G, w, c):
    Return the solution $F = F'$.
  Else: Return the solution $F = F' \cup \{e\}$.

Else: Let $\hat{w}$ be any homogeneous weight function w.r.t. (G, c).
  Let $\epsilon = \min\{w(e)/\hat{w}(e) \mid \hat{w}(e) \geq 1\}$.
  Define the weight functions $w_1(e) = \epsilon\, \hat{w}(e)$, $w_2 = w - w_1$.

  Return $\mathbf{LR}(G, w_2, c)$.

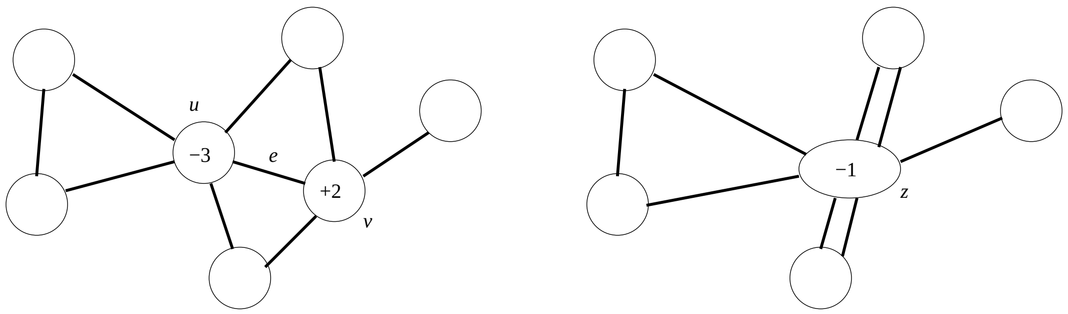

Figure 3: *An example of contraction of an edge, e. The numbers at the vertices represent charges, with $c(z) = c(u) + c(v)$.*

The contraction of an edge, e, in step 3 is performed by deleting e, "fusing" its two end vertices, u and v, into a single new vertex, z, and defining the charge of z to be the sum of the charges of u and v. Every edge incident on u or v becomes incident on z instead. See Fig. 3 for an example. The algorithm exploits the following observation.

Proposition 6 *Let (G', w', c') be the instance obtained from (G, w, c) by the contraction of an edge, e. Then*

1. *The weight of an optimal solution for (G, w, c) is not less than the weight of an optimal solution for (G', w', c').*
2. *If F' is an optimal solution for (G', w', c'), then either F' is an optimal solution for (G, w, c), or else it is infeasible for (G, w, c), but $F' \cup \{e\}$ is an optimal solution for (G, w, c).*

3.3 Tightness of $\hat{w}$ criteria

In this subsection we show that 2-effectiveness is the best that can be achieved with the present techniques if the r-effective weight function is *locally-defined.* By "locally-defined" we mean that the weight function for an edge, e, depends only on the charges of the vertices at the endpoints of e. Examples are constructed in which all the non-zero charges at vertices are equal to each other. We show then that the homogeneous weight-function (8) is in fact the most general choice, up to a multiplicative constant of course.
Consider a locally-defined weight function, w, assumed to be 2-effective. By a series of counter-examples, we show that, up to a multiplicative constant, w must be homogeneous, i.e., satisfy (8). Let w be given by

$$\hat{w}(\{u,v\}) = \begin{cases} z, & \text{if } c(u) = c(v) = 0\,, \\ a, & \text{if } c(u) = 0,\ c(v) \neq 0\,, \text{ or } c(u) \neq 0,\ c(v) = 0\,, \\ b, & \text{if } c(u) > 0,\ c(v) > 0\,, \text{ or } c(u) < 0,\ c(v) < 0\,, \\ c, & \text{if } c(u)c(v) < 0\,, \end{cases}$$

where z, a, b, and c are fixed independently of the graph. For brevity, we treat here positive and negative charges symmetrically, but the derivation is equally valid for the

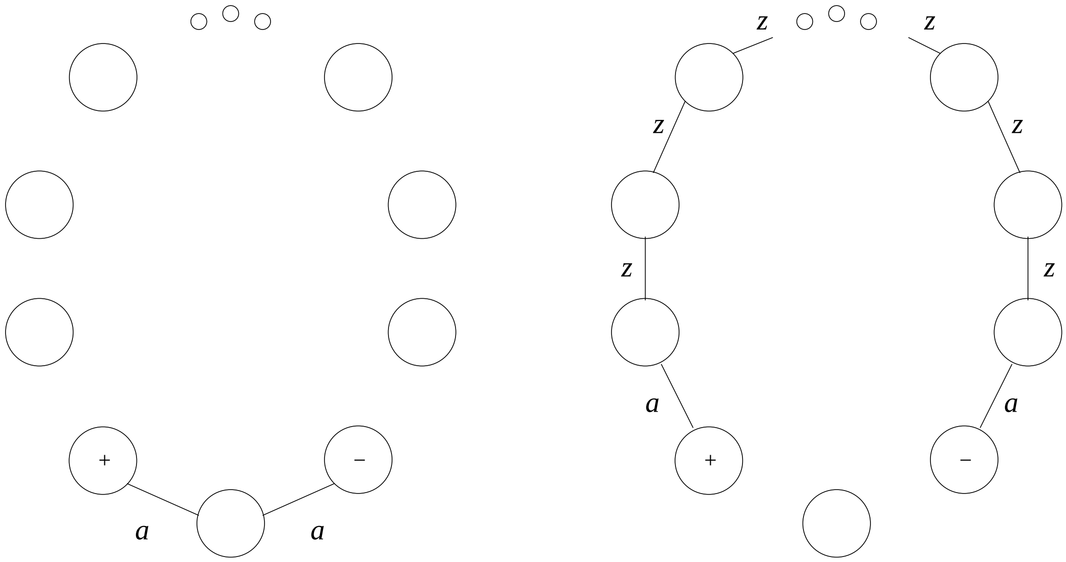

Figure 4: *The left and right figures correspond to two different feasible solutions on the same single graph, with only the relevant edges shown for each. The non-zero charges at the vertices are marked. If $a = 0$, then w is trivial (hence useless) for the left-hand graph. Therefore, a must be strictly positive, and, without loss of generality, we choose $a = 1$. Therefore, the cost of the solution on the left is $2a$, while the cost of the right-hand solution is $2a$ plus an arbitrarily large number times z. 2-effectiveness requires that the ratio between these costs be between $1/2$ and 2. Therefore, we must have $z = 0$.*

general case. (This only requires considering some of our examples also with the signs of the charges reversed).
In each of the figures we show two possible solutions for the same graph. Edges not participating in the solution are omitted for clarity. In each example we apply the definition of 2-effectiveness to derive a constraint on the parameters, $\{z, a, b, c\}$. Fig. 4 (left) shows that a must be strictly positive, otherwise, the total weight of the solution is zero in the case shown, and the algorithm gets stuck. Without loss of generality, we set $a = 1$. Next, when we compare the cost of the left solution to that of the right solution, we find that z must equal zero, else we could always construct a large enough example such that the right solution will cost more than twice as much as the left solution. In a similar fashion, Figs. 5 and 6 show that c must be not lesser than 2, and no greater than 2, respectively. Finally, Figs. 7 and 8 show that b must be not greater than 2, but not lesser than 1, respectively. Together, these constraints show that w must indeed be homogeneous, and also that it is not possible to achieve better than 2-effectiveness by a locally-defined weight function.

4 Acknowledgment

We wish to thank Moshe Israeli for introducing us to this problem and for his advice, and Guy Flysher for his help.

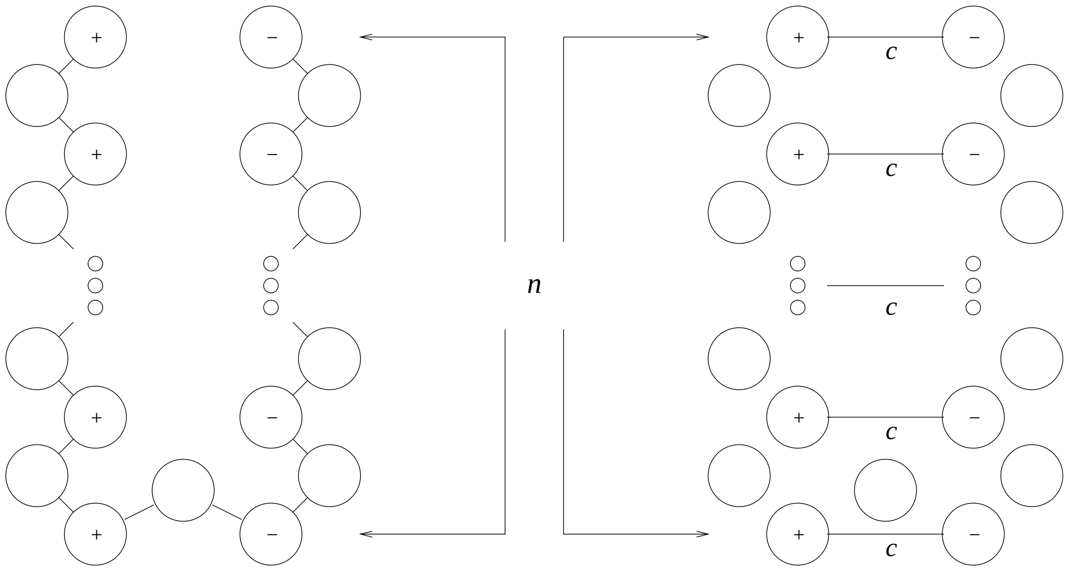

Figure 5: *The left and right figures correspond to two different feasible solutions on the same single graph, with only the relevant edges shown for each. The nonzero charges at the vertices are marked. Unmarked edges have weight* $a = 1$. *The cost of the solution depicted on the left is* $4n - 2$, *while the cost of the right-hand solution is* cn. 2-*effectiveness requires that the ratio between these costs be between* $1/2$ *and* 2. *This implies* $4n - 2 \leq 2cn$, *hence,* $c \geq 2 - 1/n$. *Since* n *may be arbitrarily large, we obtain the constraint,* $c \geq 2$.

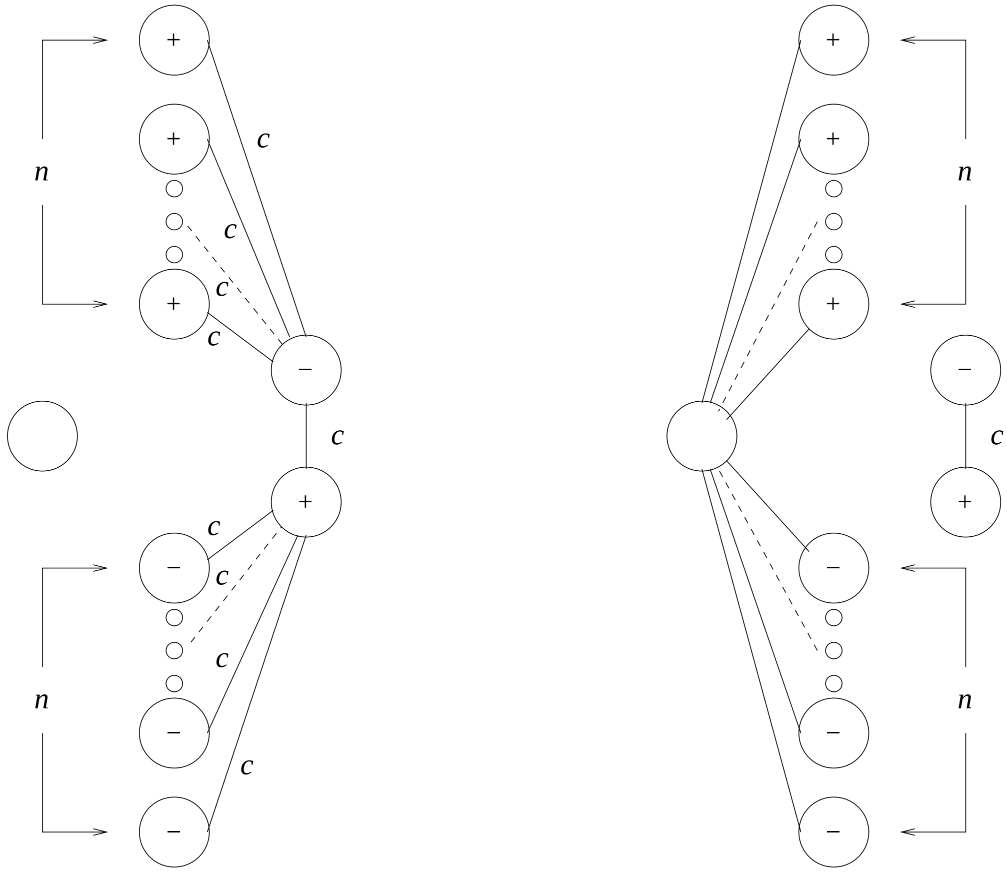

Figure 6: *The left and right figures correspond to two different solutions on the same single graph, with only the relevant edges shown for each. The nonzero charges at the vertices are marked. Unmarked edges have weight $a = 1$. The cost of the solution depicted on the left is $c(2n+1)$, while the cost of the right-hand solution is $2n + c$. 2-effectiveness requires that the ratio between these costs be between 1/2 and 2. This implies $c(2n+1) \leq 2(2n+c)$, hence, $c \leq 2/(1 - 1/2n)$. Since n may be arbitrarily large, we obtain the constraint, $c \leq 2$. Together with the previous example, we obtain $c = 2$.*

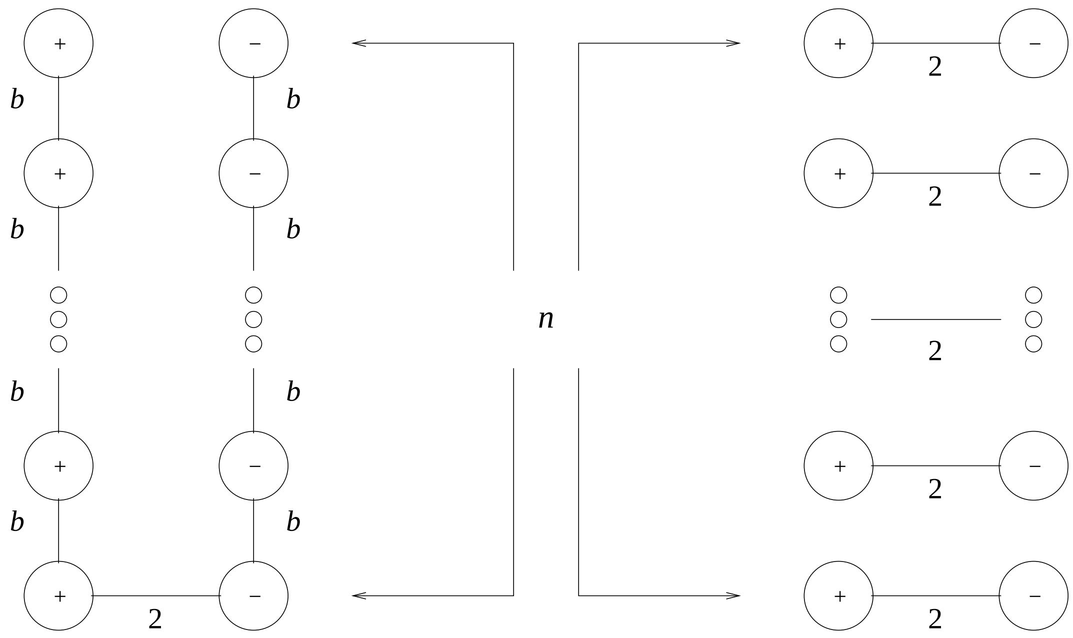

Figure 7: *The left and right figures correspond to two different solutions on the same single graph, with only the relevant edges shown for each. The non-zero charges at the vertices are marked. Edges connecting opposite-sign charged vertices have weight 2, due to the previous examples. The cost of the solution depicted on the left is $2b(n-1)+2$, while the cost of the right-hand solution is $2n$. 2-effectiveness requires that the ratio between these costs be between $1/2$ and 2. This implies $2b(n-1)+2 \leq 4n$, hence, $b \leq (2n-1)/(n-1)$. Since n may be arbitrarily large, we obtain the constraint, $b \leq 2$.*

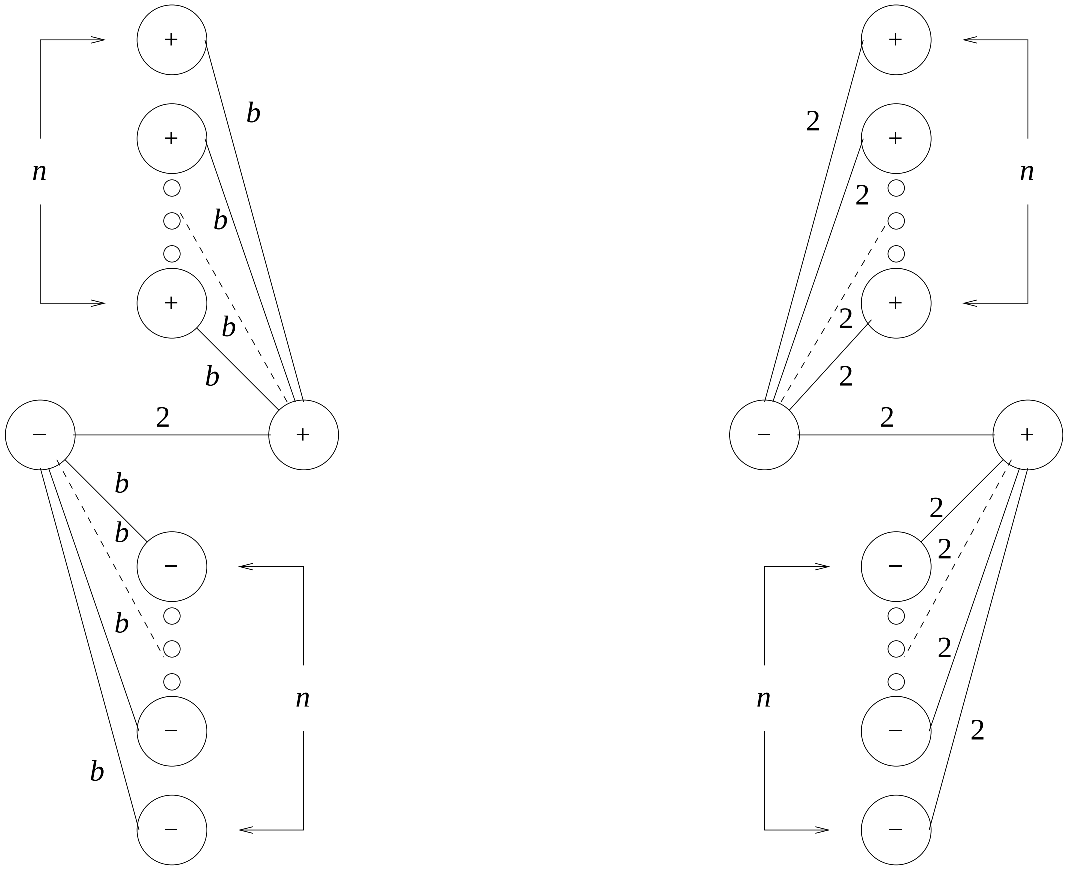

Figure 8: *The left and right figures correspond to two different solutions on the same single graph, with only the relevant edges shown for each. The non-zero charges at the vertices are marked. The cost of the solution depicted on the left is $2nb + 2$, while the cost of the right-hand solution is $4n + 2$. 2-effectiveness requires that the ratio between these costs be between $1/2$ and 2. This implies $2nb + 2 \geq \frac{1}{2}(4n + 2)$, hence, $b \geq 1 - 1/2n$. Since n may be arbitrarily large, we obtain the constraint, $b \geq 1$. Together with the previous example, we obtain $1 \leq b \leq 2$.*

References

[1] R. Bar-Yehuda, *One for the price of two: A unified approach for approximating covering problems*, Algorithmica, 27 (2000), pp. 131–144.

[2] R. Bar-Yehuda and S. Even, *A local ratio theorem for approximating the weighted vertex cover problem*, Annals of Discrete Mathematics, 25 (1986), pp. 27–46.

[3] J. R. Buckland, J. M. Huntley, and S. R. E. Turner, *Unwrapping noisy phase maps by use of a minimum-cost-matching algorithm*, Appl. Opt., 34 (1995), pp. 5100–5108.

[4] C. W. Chen and H. A. Zebker, *Network approaches to two-dimensional phase unwrapping: Intractability and two new algorithms*, J. Opt. Soc. Am. A, 17 (2000), pp. 401–414.

[5] D. C. Ghiglia and M. D. Pritt, *Two-Dimensional Phase Unwrapping: Theory, Algorithms, and Software*, John Wiley & Sons, Inc., New York, first ed., 1998.

[6] D. C. Ghiglia and L. A. Romero, *Minimum L^p-Norm two-dimensional phase unwrapping*, J. Opt. Soc. Am. A, 13 (1996), pp. 1999–2013.

[7] M. X. Goemans and D. P. Williamson, *A general approximation technique for constrained forest problems*, SIAM J. Comput., 24 (1995), pp. 296–317.

[8] R. M. Goldstein, H. A. Zebker, and C. L. Werner, *Satellite radar interferometry: Two-dimensional phase unwrapping*, Radio Sci., 23 (1988), pp. 713–720.