Solving graph compression via optimal transport

Vikas K. Garg
CSAIL, MIT
v garg@csail.mit.edu

Tommi Jaakkola
CSAIL, MIT
tommi@csail.mit.edu

Abstract

We propose a new approach to graph compression by appeal to optimal transport. The transport problem is seeded with prior information about node importance, attributes, and edges in the graph. The transport formulation can be setup for either directed or undirected graphs, and its dual characterization is cast in terms of distributions over the nodes. The compression pertains to the support of node distributions and makes the problem challenging to solve directly. To this end, we introduce Boolean relaxations and specify conditions under which these relaxations are exact. The relaxations admit algorithms with provably fast convergence. Moreover, we provide an exact $O(d \log d)$ algorithm for the subproblem of projecting a $d$-dimensional vector to transformed simplex constraints. Our method outperforms state-of-the-art compression methods on graph classification.

1 Introduction

Graphs are widely used to capture complex relational objects, from social interactions to molecular structures. Large, richly connected graphs can be, however, computationally unwieldy if used as-is, and spurious features present in the graphs that are unrelated to the task can be statistically distracting. A significant effort thus has been spent on developing methods for compressing or summarizing graphs towards graph sketches [1]. Beyond computational gains, these sketches take center stage in numerous tasks pertaining to graphs such as partitioning [2, 3], unraveling complex or multi-resolution structures [4, 5, 6, 7], obtaining coarse-grained diffusion maps [8], including neural convolutions [9, 10, 11].

State-of-the-art compression methods broadly fall into two categories: (a) sparsification (removing edges) and (b) coarsening (merging vertices). These methods measure spectral similarity between the original graph and a compressed representation in terms of a (inverse) Laplacian quadratic form [12, 13, 14, 15, 16]. Thus, albeit some of these methods approximately preserve the graph spectrum (see, e.g., [1]), they are oblivious to, and thus less effective for, downstream tasks such as classification that rely on labels or attributes of the nodes. Also, the key compression steps in most of these methods are, typically, either heuristic or detached from their original objective [17].

We address these issues by taking a novel perspective that appeals to the theory of optimal transport [18], and develops its connections to minimum cost flow on graphs [19, 20]. Specifically, we interpret graph compression as minimizing the transport cost from a fixed initial distribution supported on all vertices to an unknown target distribution whose size of support is limited by the amount of compression desired. Thus, the compressed graph in our case is a subgraph of the original graph, restricted to a subset of the vertices selected via the associated transport problem. The transport cost depends on the specified prior information such as importance of the nodes and their labels or attributes, and thus can be informed by the downstream task. Moreover, we take into account the underlying geometry toward the transport cost, unlike agnostic measures such as KL-divergence [21].

There are several technical challenges that we must address. First, the standard notion of optimal transport on graphs is tailored to directed graphs where the transport cost decomposes as a directed
flow along the edge orientations [22]. To circumvent this limitation, we extend optimal transport on graphs to handle both directed and undirected edges, and derive a dual that directly measures the discrepancy between distributions on the vertices. As a result, we can also compress mixed graphs that contain both directed and undirected edges [23, 24, 25, 26].

The second challenge comes from the compression itself, enforced in our approach as sparse support of the target distribution. Optimal transport (OT) is known to be computationally intensive, and almost all recent applications of OT in machine learning, e.g., [27, 28, 29, 30, 31, 32] rely on entropy regularization [33, 34, 35, 36, 37] for tractability. However, entropy is not conducive to sparse solutions since it discourages the variables from ever becoming zero [22]. In principle, one could consider convex alternatives such as enforcing ℓ₁ penalty [38]. However, such methods require iterative tuning to find a solution that matches the desired support, and require strong assumptions such as restricted eigenvalue, isometry, or nullspace conditions for recovery. Some of these issues were previously averted by introducing binary selection variables [39, 40] and using Boolean relaxations in the context of unconstrained real spaces (regression). However, they do not apply to our setting since the target distribution must reside in the simplex. We introduce constrained Boolean relaxations that are not only efficient, but also provide exactness certificates.

Our graph compression formulation also introduces new algorithmic challenges. For example, solving our sparsity controlled dual transport problem involves a new subproblem of projecting on the probability simplex Δ under a diagonal transformation. Specifically, let \( D(\epsilon) \) be a diagonal matrix with the diagonal \( \epsilon \in [0, 1]^d \setminus \{0\} \). Then, for a given \( \epsilon \), the problem is to find the projection \( x \in \mathbb{R}^d \) of a given vector \( y \in \mathbb{R}^d \) such that \( D(\epsilon)x \in \Delta \). This generalizes well-studied problem of Euclidean projection on the probability simplex [41, 42, 43], recovered if each \( \epsilon_i \) is set to 1. We provide an exact \( O(d \log d) \) algorithm for solving this generalized projection. Our approach leads to convex-concave saddle point problems with fast convergence via methods such as Mirror Prox [44].

To summarize, we make the following contributions. We propose an approach for graph compression based on optimal transport (OT). Specifically, we (a) extend OT to undirected and mixed graphs (section 2), (b) introduce constrained Boolean relaxations for our dual OT problem, and provide exactness guarantees (section 3), (c) generalize Euclidean projection onto simplex, and provide an efficient algorithm (section 3), and (d) demonstrate that our algorithm outperforms state-of-the-art exactness guarantees (section 3), and (d) demonstrate that our algorithm outperforms state-of-the-art compression methods, both in accuracy and compression time, on classifying graphs from standard real datasets. We also provide qualitative results that our approach provides meaningful compression in synthetic and real graphs (section 4).

2 Optimal transport for general edges

Let \( \hat{G} = (V, \hat{E}) \) be a directed graph on nodes (or vertices) \( V \) and edges \( \hat{E} \). We define the signed incidence matrix \( \hat{F} \): \( \hat{F}(\hat{e}, v) = 1 \) if \( \hat{e} = (w, v) \in \hat{E} \) for some \( w \in V \), \( -1 \) if \( \hat{e} = (v, w) \in \hat{E} \) for some \( w \in V \), and 0 otherwise. Let \( c(\hat{e}) \in \mathbb{R}_+ \) be the positive cost to transport unit mass along edge \( \hat{e} \in \hat{E} \), and \( \Delta(V) \) the probability simplex on \( V \). The shorthand \( a \preceq b \) denotes that \( a(i) \leq b(i) \) for each component \( i \). Let \( \mathbf{0} \) be a vector of all zeros and \( \mathbf{1} \) a vector of all ones. Let \( \rho_0, \rho_1 \in \Delta(V) \) be distributions over the vertices in \( V \). The optimal transport distance \( \hat{W}(\rho_0, \rho_1) \) from \( \rho_0 \) to \( \rho_1 \) is [22]:

\[
\hat{W}(\rho_0, \rho_1) = \min_{J \in \mathbb{R}^{|V|]} \sum_{\hat{e} \in \hat{E}} c(\hat{e})J(\hat{e}) \quad \text{s.t.} \quad \hat{F}^\top J = \rho_1 - \rho_0,
\]

where \( J(\hat{e}) \) is the non-negative mass transfer from tail to head on edge \( \hat{e} \). Intuitively, \( \hat{W}(\rho_0, \rho_1) \) is the minimum cost of a directed flow from \( \rho_0 \) to \( \rho_1 \). In order to extend this intuition to the undirected graphs, we need to refine the notion of incidence, and let the mass flow in either direction. Specifically, let \( G = (V, E) \) be a connected undirected graph. We define the incidence matrix pertaining to \( G \) as \( F(e, v) = 1 \) if edge \( e \) is incident on \( v \), and 0 otherwise. With each undirected edge \( e \in E \), having cost \( c(e) \in \mathbb{R}_+ \), we associate two directed edges \( e^+ \) and \( e^- \), each with cost \( c(e) \), and flow variables \( J^+(e), J^-(e) \geq 0 \). Then, the total undirected flow pertaining to \( e \) is \( J^+(e) + J^-(e) \). Since we incur cost for flow in either direction, we define the optimal transport cost \( W(\rho_0, \rho_1) \) from \( \rho_0 \) to \( \rho_1 \) as

\[
\min_{J^+, J^- \in \mathbb{R}^{|E|}, 0 \preceq J^+ - J^-} \sum_{e \in E} c(e)(J^+(e) + J^-(e)) \quad \text{s.t.} \quad F^\top (J^- - J^+) = \rho_1 - \rho_0.
\]
We view graph compression as the problem of minimizing the optimal transport distance from an initial distribution $\rho_0$ having full support on the vertices $V$ to a target distribution $\rho_1$ that is supported only on a subset $S_V(\rho_1)$ of $V$. The compressed subgraph is obtained by restricting the original graph to vertices in $S_V(\rho_1)$ and the incident edges. The initial distribution $\rho_0$ encodes any prior information. For instance, it might be taken as a stationary distribution of random walk on the graph. Likewise, the cost function $c$ encodes the preference for different edges. In particular, a high value of $c(e)$ would inhibit edge $e$ from being active. This flexibility allows our framework to inform compression based on the specifics of different downstream applications by getting to define $\rho_0$ and $c$ appropriately.

### 3.1 Dual characterization of the transport distance

Note that $W(\rho_0, \rho_1)$ defines an optimization problem over edges. However, our perspective requires quantifying $W(\rho_0, \rho_1)$ as an optimization over the vertices. Fortunately, strong duality comes to our rescue. Let $c = (c(e), e \in E)$ be the column vector obtained by stacking the costs. The dual of (1) is

$$\max_{0 \leq y \leq \rho_0} \min_{\rho \in \Delta(V)} (y - z)^\top (\rho_1 - \rho_0), \quad \text{or equivalently,} \quad \max_{t \in \mathbb{R}^{|V|}} \min_{c \in \mathbb{R}^{|E|}} t^\top (\rho_1 - \rho_0).$$

This alternative formulation of $W(\rho_0, \rho_1)$ in (2) lets us define compression solely in terms of variables over vertices. Specifically, for a budget of at most $k$ vertices, we solve

$$\min_{\rho_1 \in \Delta(V)} \frac{1}{\|\rho_1\|_0 \leq k} \max_{t \in \mathbb{R}^{|V|}} \min_{c \in \mathbb{R}^{|E|}} t^\top (\rho_1 - \rho_0) + \frac{\lambda}{2} \|\rho_1\|^2,$$

where $\lambda > 0$ is a regularization hyperparameter, and $\|\rho_1\|_0$ is the number of vertices with positive mass under $\rho_1$, i.e., the cardinality of support set $S_V(\rho_1)$. The quadratic penalty is strongly convex in $\rho_1$, so as we shall see shortly, would help us leverage fast algorithms for a saddle point problem. Note that a high value of $\lambda$ would encourage $\rho_1$ toward a uniform distribution. We favor this penalty over entropy, which is not conducive to sparse solutions since entropy would forbid $\rho_1$ from having zero mass at any vertex in the graph.

Our next result reveals the structure of optimal solution $\rho_1^*$ in (3). Specifically, $\rho_1^*$ must be expressible as an affine function of $\rho_0$ and $F$. Moreover, the constraints on active edges are tight. This reaffirms our intuition that $\rho_1^*$ is obtained from $\rho_0^*$ by transporting mass along a subset of the edges, i.e., the active edges. The remaining edges do not participate in the flow.

**Theorem 2.** The optimal $\rho_1^*$ in (3) is of the form $\rho_1^* = \rho_0 + F^\top \eta$, where $\eta \in \mathbb{R}^{|E|}$. Furthermore, for any active edge $e \in E$, we must have $F^\top(e) \in \{c(e), -c(e)\}$.

### 3.2 Constrained Boolean relaxations

The formulation (3) is non-convex due to the support constraint on $\rho_1$. Since recovery under $\ell_1$ based methods such as Lasso often requires extensive tuning, we resort to the method of Boolean
relaxations that affords an explicit control much like the $\ell_0$ penalty. However, prior literature on Boolean relaxations is limited to variables that have no additional constraints beyond sparsity. Thus, in order to deal with the simplex constraints $\rho_1 \in \Delta(V)$, we introduce constrained Boolean relaxations. Specifically, we define the characteristic function $g_V(x) = 0$ if $x \in \Delta(V)$ and $\infty$ otherwise, and move the non-sparse constraints inside the objective. This lets us delegate the sparsity constraints to binary variables, which can be relaxed to $[0, 1]$. Using the definition of $L$, we can write (3) as

$$\min_{\rho_1 \in [0, 1]} \max_{t \in [0, 1]} L_\lambda(\rho_1, t; \rho_0) + g_V(\rho_1).$$

Denoting by $\odot$ the Hadamard (elementwise) product, and introducing variables $\epsilon \in [0, 1]^{|V|}$, we get

$$\min_{\epsilon \in [0, 1]^{|V|}} \min_{\rho_1 \in [0, 1]} \max_{t \in [0, 1]} L_\lambda(\rho_1 \odot \epsilon, t; \rho_0) + g_V(\rho_1 \odot \epsilon).$$

Adjusting the characteristic term as a constraint, we have the following equivalent problem

$$\min_{\epsilon \in [0, 1]^{|V|}} \min_{\rho_1 \in [0, 1]} \max_{t \in [0, 1]} L_\lambda(\rho_1 \odot \epsilon, t; \rho_0).$$

**Algorithm 1** Algorithm to compute Euclidean projection on the $d$-simplex $\Delta$ under a diagonal transformation.

**Input:** $y, \epsilon$

Define $I_\geq \{ j \in [d] \mid \epsilon_j > 0 \}$
Define $I_\leq \{ j \in [d] \mid \epsilon_j \leq 0 \}$

$$y_{\geq} \triangleq \{ y_j \mid j \in I_\geq \}; \epsilon_{\geq} \triangleq \{ \epsilon_j \mid j \in I_\geq \}$$
$$\text{Sort } y_{\geq} \text{ into } y_{\geq}, \text{ and } \epsilon_{\geq} \text{ into } \epsilon_{\geq}, \text{ in non-increasing order, based on } y_j, \epsilon_j, j \in I_\geq.$$

Define $I_{\geq} = I_{\geq}^T = I_{\geq} \odot y_{\geq}$, $I_{\leq} = I_{\leq}^T = I_{\leq} \odot \epsilon_{\leq}$.

Let $\pi$ map $j \in I_{\geq}$ to $\pi(j) \in [y_{\geq}]$. Thus

$$\hat{y}_j / \epsilon_j = \hat{y}_j / \epsilon_j \geq \ldots \geq \hat{y}_j / \epsilon_j \geq 0.$$

Define $T_{\pi, \epsilon} = \max \{ j \in [y_{\geq}] \mid b_j > 0 \}$

$$\ell = \max \{ j \in [y_{\geq}] \mid b_j > 0 \}$$

$$\alpha = \frac{1}{\max_{\ell \in y_{\geq}} \{ \hat{y}_j \}}$$

$$x_j = \max \{ \hat{y}_j / \epsilon_j, \hat{y}_j \}, \forall j \in I_{\geq}, x_j = y_j, \forall j \in I_{\leq}.$$

Our formulation in (4) requires solving a new subproblem, namely, Euclidean projection on the $d$-simplex $\Delta$ under a diagonal transformation. Specifically, let $D(\epsilon)$ be a diagonal matrix with the diagonal $\epsilon \in [0, 1]^d \setminus \{0\}$. Then, for a given $\epsilon$, the problem is to find the projection $x \in \mathbb{R}^d$ of a given vector $y \in \mathbb{R}^d$ such that $D(\epsilon)x = x \odot \epsilon \in \Delta$. This problem generalizes Euclidean projection on the probability simplex $\mathbb{R}_{+}^{d}$, which is recovered when we set $\epsilon$ to 1, i.e., an all-ones vector. Our next result shows that Algorithm 1 solves this problem exactly in $O(d \log d)$ time.

**Theorem 3.** Let $\epsilon \in [0, 1]^{|V|}$ be a given vector of weights, and $y \in \mathbb{R}^d$ be a given vector of values. Algorithm 1 solves the following problem in $O(d \log d)$ time

$$\min_{x \in \mathbb{R}^d : x \odot \epsilon \in \Delta} \frac{1}{2} ||x - y||^2.$$
where we denote the constraints for $\epsilon$ and $t$ respectively by

$$
\mathcal{E}_k \triangleq \{ \epsilon \in [0, 1]^{|V|} \mid |\epsilon^T 1| \leq k \} , \quad \mathcal{T}_{F,c} \triangleq \{ t \in \mathbb{R}^{|V|} \mid -c \leq Ft \leq c \} .
$$

We can thus eliminate $\epsilon$ from the regularization term via a change of variable $\tilde{\rho}_1 \odot \epsilon \rightarrow \tilde{\rho}_1$

$$
\min_{\epsilon \in \mathcal{E}_k} \min_{\tilde{\rho}_1 \in \mathbb{R}^{|V|}} \max_{t \in \mathcal{T}_{F,c}} \mathcal{L}_0(\tilde{\rho}_1 \odot \epsilon, t; \rho_0) + \frac{\lambda}{2} ||\tilde{\rho}_1||^2 . \tag{6}
$$

We note that (5) is a mixed-integer program due to constraints $\mathcal{E}_k$, and thus hard to solve. Nonetheless, we can relax the hard binary constraints on the coordinates of $\epsilon$ to $[0, 1]$ intervals to obtain a saddle point formulation with a strongly convex term, and solve the relaxation efficiently, e.g., via customized versions of methods such as Mirror Prox [44], Accelerated Gradient Descent [47], or Primal-Dual Hybrid Gradient [48]. An attractive property of our relaxation is that if the solution $\hat{\epsilon}$ from the relaxed problem is integral then $\hat{\epsilon}$ must be optimal for the non-relaxed hard problem (6), and so the original formulation (3). We now pin down the necessary and sufficient conditions for optimality of $\hat{\epsilon}$.

**Theorem 4.** Let $S_V(\rho_1^*) = \{ v \in V \mid \rho_1^*(v) > 0 \}$ be the support of optimal $\rho_1$ in the original formulation (3). Let the indicator $I_{S_V}^* \in \{0, 1\}^{|V|}$ be such that $I_{S_V}^*(v) = 1$ if $v \in S_V(\rho_1^*)$ and 0 otherwise. The relaxation of (6) is guaranteed to recover $S_V(\rho_1^*)$ if and only if there exists a tuple $(\gamma, \hat{t}, \hat{\nu}, \hat{\zeta}) \in \mathbb{R}_+ \times \mathbb{R}^{|V|} \times \mathbb{R}_+ \times \mathbb{R}$ such that the following holds for all vertices $v \in V$,

$$
|\hat{t}(v) - \hat{\nu}(v) + \hat{\zeta}| \begin{cases} > \gamma & \text{if } v \in S_V(\rho_1^*) \\ < \gamma & \text{if } v \notin S_V(\rho_1^*) \end{cases} , \tag{7}
$$

$$
(\hat{t}, \hat{\nu}, \hat{\zeta}) \in \arg \max_{t \in \mathcal{T}_{F,c}} \max_{\nu \in \mathbb{R}^{|V|}} \max_{\zeta \in \mathbb{R}} -\left( \frac{1}{2\lambda} \| (t - \nu + \zeta 1) \odot I_{S_V}^* \| ^2 + t^T \rho_0 + \zeta \right) . \tag{8}
$$

The quantity $|\hat{t}(v) - \hat{\nu}(v) + \hat{\zeta}|$ in (7) may be viewed as the strength of a signal. In that sense, we require the vertices in support of optimal $\rho_1$ to have a strictly higher signal that the vertices not in the support. Such signal detection conditions appear in various contexts and often have information theoretic implications, e.g., Ising models [49]. (7) is also reminiscent of the $\beta$-min condition on regression coefficients for variable selection with Lasso in high-dimensional linear models [50].

For some applications projecting on the simplex, as required by (6), may be an expensive operation. We can invoke the minimax theorem to swap the order of $\tilde{\rho}_1$ and $t$, and proceed with a Lagrangian dual to eliminate $\tilde{\rho}_1$ at the expense of introducing a scalar variable. Thus, effectively, we can replace the projection on simplex by a one-dimensional search. We state this equivalent formulation below.

**Theorem 5.** Problem (6), and thus the original formulation (3), is equivalent to

$$
\min_{\epsilon \in \mathcal{E}_k} \max_{\zeta \in \mathbb{R}} \frac{1}{2\lambda} \sum_{v \in \mathbb{V}} (\epsilon(v)(\hat{t}(v) + \zeta)^2 + 2\lambda t(v)\rho_0(v)) - \sum_{v \in \mathbb{V}} t(v)\rho_0(v) - \zeta . \tag{9}
$$

We present a customized Mirror Prox procedure in Algorithm 2. The projections $\text{Proj}_{\mathcal{E}_k}$ and $\text{Proj}_{\mathcal{T}_{F,c}}$ can be computed efficiently [45,46]. We round the solution $\hat{\epsilon} \in [0, 1]^{|V|}$ returned by the algorithm to have at most $k$ vertices as the estimated support for the target distribution $\rho_1$ if $\hat{\epsilon}$ is not integral. The compressed graph is taken to be the subgraph spanned by these vertices.

### 3.3 Specifying the cost function

In our experiments, we fixed the cost of each edge, computed based on the agreement between the associated vertex labels. Here we illustrate briefly how to parameterize the cost and how the parameters could be learned. Define $\ell(i, j) = 1$ for edge $(i, j)$ if vertices $i$ and $j$ have the same label, and $-1$ otherwise. Let the cost function be parameterized by $\theta = (\theta_s, \theta_d), \theta_s > 0, \theta_d > 0$ such that

$$
e^\theta(i, j) = 0.5(\theta_d(1 - \ell(i, j)) + \theta_s(1 + \ell(i, j))) .$$
We conducted several experiments to demonstrate the merits of our method. We start by describing the experimental setup. We fixed the value of hyperparameters in Algorithm 2 for all our experiments. Specifically, we set the regularization coefficient $\lambda = 1$, and the gradient rates $\alpha = 0.1$, $\beta = \ldots$.

### 3.4 Relation to other compression techniques

Most compression algorithms try to preserve the graph spectrum via a (multi-level) coarsening procedure: at each level they compute a matching of vertices and merge the matched vertices, e.g., Heavy Edge [1] contracts those edges $(i,j)$ that are incident on low degree vertices. Likewise REC [1] follows a randomized greedy procedure for generating maximal matching incrementally. Let $d_i$ be the degree of node $i$. Setting the cost $c(i, j) = \max(d_i, d_j)$ in our framework will incentivize flow on edges with low degree vertices, and in turn, compression of one of their end points. The vertices not in support of target distribution may then be viewed as being matched to (a subset of) adjacent vertices that they transfer flow to. Unlike other methods, our approach is flexible in terms of defining $c(i, j)$.

### 4 Experiments

We provide the statistics on the number of graphs, number of classes, average number of nodes, and average number of edges in each dataset. The classification test accuracy (along with standard deviation) when each graph was (roughly) compressed to half is shown for each method for each training fraction in $\{0.2, \ldots, 0.8\}$. The algorithm having the best performance is indicated with bold font in each case.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>method</th>
<th>acc@0.2</th>
<th>acc@0.3</th>
<th>acc@0.4</th>
<th>acc@0.5</th>
<th>acc@0.6</th>
<th>acc@0.7</th>
<th>acc@0.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSRC-21C</td>
<td>REC</td>
<td>.485±.016</td>
<td>.543±.010</td>
<td>.595±.010</td>
<td>.625±.013</td>
<td>.641±.008</td>
<td>.696±.013</td>
<td>.738±.016</td>
</tr>
<tr>
<td>classes: 20</td>
<td>Affinity</td>
<td>.413±.021</td>
<td>.489±.008</td>
<td>.516±.011</td>
<td>.549±.010</td>
<td>.560±.016</td>
<td>.607±.019</td>
<td>.654±.021</td>
</tr>
<tr>
<td>nodes: 40.2</td>
<td>Alg. Dist.</td>
<td>.452±.036</td>
<td>.498±.035</td>
<td>.524±.021</td>
<td>.535±.027</td>
<td>.531±.029</td>
<td>.590±.032</td>
<td>.652±.044</td>
</tr>
<tr>
<td>edges: 96.6</td>
<td>OTC</td>
<td>.548±.004</td>
<td>.605±.003</td>
<td>.639±.006</td>
<td>.679±.003</td>
<td>.696±.002</td>
<td>.742±.007</td>
<td>.778±.005</td>
</tr>
<tr>
<td>DHFR</td>
<td>REC</td>
<td>.681±.011</td>
<td>.704±.014</td>
<td>.724±.007</td>
<td>.738±.009</td>
<td>.749±.008</td>
<td>.756±.011</td>
<td>.771±.011</td>
</tr>
<tr>
<td>graphs: 467</td>
<td>Heavy</td>
<td>.719±.010</td>
<td>.751±.010</td>
<td>.776±.012</td>
<td>.782±.008</td>
<td>.777±.009</td>
<td>.786±.014</td>
<td>.799±.013</td>
</tr>
<tr>
<td>classes: 2</td>
<td>Affinity</td>
<td>.717±.013</td>
<td>.733±.011</td>
<td>.745±.014</td>
<td>.761±.014</td>
<td>.771±.019</td>
<td>.767±.015</td>
<td>.785±.013</td>
</tr>
<tr>
<td>nodes: 44.5</td>
<td>Alg. Dist.</td>
<td>.743±.011</td>
<td>.761±.012</td>
<td>.768±.022</td>
<td>.786±.019</td>
<td>.810±.025</td>
<td>.817±.033</td>
<td>.809±.030</td>
</tr>
<tr>
<td>edges: 44.5</td>
<td>OTC</td>
<td>.757±.004</td>
<td>.784±.003</td>
<td>.797±.005</td>
<td>.799±.003</td>
<td>.811±.007</td>
<td>.814±.006</td>
<td>.823±.004</td>
</tr>
<tr>
<td>MSRC-9</td>
<td>REC</td>
<td>.738±.011</td>
<td>.782±.010</td>
<td>.817±.009</td>
<td>.818±.013</td>
<td>.835±.020</td>
<td>.833±.018</td>
<td>.840±.013</td>
</tr>
<tr>
<td>graphs: 221</td>
<td>Heavy</td>
<td>.648±.019</td>
<td>.710±.024</td>
<td>.766±.014</td>
<td>.773±.010</td>
<td>.786±.009</td>
<td>.796±.010</td>
<td>.813±.009</td>
</tr>
<tr>
<td>classes: 8</td>
<td>Affinity</td>
<td>.745±.015</td>
<td>.729±.026</td>
<td>.762±.010</td>
<td>.774±.014</td>
<td>.789±.026</td>
<td>.796±.019</td>
<td>.801±.017</td>
</tr>
<tr>
<td>nodes: 40.6</td>
<td>Alg. Dist.</td>
<td>.666±.048</td>
<td>.717±.051</td>
<td>.756±.029</td>
<td>.771±.039</td>
<td>.798±.032</td>
<td>.803±.030</td>
<td>.809±.046</td>
</tr>
<tr>
<td>edges: 97.9</td>
<td>OTC</td>
<td>.784±.005</td>
<td>.808±.005</td>
<td>.826±.007</td>
<td>.846±.003</td>
<td>.839±.006</td>
<td>.842±.007</td>
<td>.854±.003</td>
</tr>
<tr>
<td>BZR-MD</td>
<td>REC</td>
<td>.525±.011</td>
<td>.548±.015</td>
<td>.563±.020</td>
<td>.553±.021</td>
<td>.563±.012</td>
<td>.569±.012</td>
<td>.587±.020</td>
</tr>
<tr>
<td>graphs: 306</td>
<td>Heavy</td>
<td>.497±.000</td>
<td>.546±.000</td>
<td>.555±.000</td>
<td>.522±.000</td>
<td>.550±.000</td>
<td>.572±.000</td>
<td>.558±.000</td>
</tr>
<tr>
<td>classes: 2</td>
<td>Affinity</td>
<td>.508±.006</td>
<td>.534±.012</td>
<td>.534±.017</td>
<td>.532±.015</td>
<td>.549±.020</td>
<td>.567±.033</td>
<td>.562±.029</td>
</tr>
<tr>
<td>edges: 225.06</td>
<td>OTC</td>
<td>.534±.000</td>
<td>.569±.000</td>
<td>.547±.000</td>
<td>.579±.000</td>
<td>.572±.000</td>
<td>.607±.000</td>
<td>.603±.000</td>
</tr>
<tr>
<td>Mutagenicity</td>
<td>REC</td>
<td>.713±.006</td>
<td>.730±.006</td>
<td>.742±.005</td>
<td>.752±.004</td>
<td>.758±.005</td>
<td>.765±.007</td>
<td>.769±.007</td>
</tr>
<tr>
<td>graphs: 437</td>
<td>Heavy</td>
<td>.718±.006</td>
<td>.738±.004</td>
<td>.753±.004</td>
<td>.763±.004</td>
<td>.771±.003</td>
<td>.779±.004</td>
<td>.783±.004</td>
</tr>
<tr>
<td>classes: 2</td>
<td>Affinity</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nodes: 30.3</td>
<td>Alg. Dist.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>edges: 30.8</td>
<td>OTC</td>
<td>.749±.002</td>
<td>.768±.003</td>
<td>.779±.003</td>
<td>.787±.004</td>
<td>.792±.004</td>
<td>.795±.003</td>
<td>.799±.003</td>
</tr>
</tbody>
</table>
We observe that on some datasets the discrepancy between the average test accuracy of two algorithms is massive for every fraction. Note that though OTC performs the best on DHFR, the performance of most methods is similar (except REC, which lags behind). In contrast, REC performs better than all methods except OTC on MSRC-9. This seems to suggest that REC performs well on graphs with a wide spectrum in terms of the number of graphs, number of classes, average number of nodes per graph, and average number of edges per graph (see Table 1 for details). All these datasets have a class label for each graph, and additionally, labels for each node in every graph.

We compare the test accuracy of our algorithm, OTC (short for Optimal Transport based Compression), to several state-of-the-art methods: REC [1], Heavy edge matching (Heavy) [2], Affinity vertex proximity (Affinity) [14], and Algebraic distance (Algebraic) [12, 13]. Amongst these, the REC algorithm is a randomized method that iteratively contracts edges of the graph and thereby coarsens the graph. Since the method is randomized, REC yields a compressed graph that achieves a specified compression factor, i.e., the ratio of the number of nodes in the reduced graph to that in the original uncompressed graph, only in expectation. Therefore, in order to allow for a fair comparison, we first run REC on each graph with the compression factor set to 0.5, and then execute other baselines and our algorithm, i.e. Algorithm[2] with k set to the number of nodes in the compressed graph produced by REC. Also to mitigate the effects of randomness on the number of nodes returned by REC, we performed 5 independent runs for REC, and subsequently all other methods.

Each collection of compressed graphs was then divided into train and test sets, and used for our classification task. Since our datasets do not provide separate train and test sets, we employed the following procedure for each dataset. We partitioned each dataset into multiple train and test sets of varying sizes. Specifically, for each p ∈ {0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8}, we divided each dataset randomly into a train set containing a fraction p of the graphs in the dataset, and a test set containing the remaining 1 − p fraction. To mitigate the effects of chance, we formed 5 such independent train-test partitions for each fraction p for each dataset. We averaged the results over these multiple splits to get one reading per collection, and thus 5 readings in total for all collections, for each fraction for each method. We averaged the test accuracy across collections for each method and fraction.

We now specify the cost function c for our algorithm. As described in section 3, we can leverage the cost function to encode preference for different edges in the compressed graph. For each graph, we set c = 0.01 for each edge e incident on the nodes with same label, and c = 0.02 for each e incident on the nodes that have different label. Thus, in effect, we slightly biased the graphs to prefer retaining the edges that have separate labels at their end-points. In our experiments, we employed support vector machines (SVMs), with Weisfeiler-Leman subtree kernel [55] to quantify the similarity between graphs [53, 56, 57]. This kernel is based on the Weisfeiler-Leman test of isomorphism [58, 59], and thus naturally takes into account the labels of the nodes in the two graphs. We fixed the number of kernel iterations to 5. We also fixed T = 25 for our algorithm. For each method and each train-test split, we used a separate 5-fold cross-validation procedure to tune the coefficient of error term C over the set {0.1, 1, 10} for training an independent SVM model on the training portion.

Table 1 summarizes the performance of different methods for each fraction of the data. As the numbers in bold indicate, our method generally outperformed the other methods across the datasets. We observe that on some datasets the discrepancy between the average test accuracy of two algorithms is massive for every fraction. Note that though OTC performs the best on DHFR, the performance of most methods is similar (except REC, which lags behind). In contrast, REC performs better than all methods except OTC on MSRC-9. This seems to suggest that REC performs well on graphs with non-bipartite graphs, it has a nice physical interpretation in that any random walk on the graph always converges to this distribution irrespective of the graph structure.

The objective of our experiments is three-fold. Since compression is often employed as a preprocessing step for further tasks, we first show that our method compares favorably, in terms of test accuracy, to the state-of-the-art compression methods on graph classification. We then demonstrate that our method performed the best in terms of the compression time. We finally show that our approach provides qualitatively meaningful compression on synthetic and real examples.

### 4.1 Classifying standard graph data

We used several standard graph datasets for our experiments, namely, DHFR [51], BZR-MD [52], MSRC-9, MSRC-21C [53], and Mutagenicity [54]. We focused on these datasets since they represent a wide spectrum in terms of the number of graphs, number of classes, average number of nodes per graph, and average number of edges per graph (see Table 1 for details). All these datasets have a class label for each graph, and additionally, labels for each node in every graph.

We used several standard graph datasets for our experiments, namely, DHFR [51], BZR-MD [52], MSRC-9, MSRC-21C [53], and Mutagenicity [54]. We focused on these datasets since they represent a wide spectrum in terms of the number of graphs, number of classes, average number of nodes per graph, and average number of edges per graph (see Table 1 for details). All these datasets have a class label for each graph, and additionally, labels for each node in every graph.

Each collection of compressed graphs was then divided into train and test sets, and used for our classification task. Since our datasets do not provide separate train and test sets, we employed the following procedure for each dataset. We partitioned each dataset into multiple train and test sets of varying sizes. Specifically, for each p ∈ {0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8}, we divided each dataset randomly into a train set containing a fraction p of the graphs in the dataset, and a test set containing the remaining 1 − p fraction. To mitigate the effects of chance, we formed 5 such independent train-test partitions for each fraction p for each dataset. We averaged the results over these multiple splits to get one reading per collection, and thus 5 readings in total for all collections, for each fraction for each method. We averaged the test accuracy across collections for each method and fraction.

We now specify the cost function c for our algorithm. As described in section 3, we can leverage the cost function to encode preference for different edges in the compressed graph. For each graph, we set c = 0.01 for each edge e incident on the nodes with same label, and c = 0.02 for each e incident on the nodes that have different label. Thus, in effect, we slightly biased the graphs to prefer retaining the edges that have separate labels at their end-points. In our experiments, we employed support vector machines (SVMs), with Weisfeiler-Leman subtree kernel [55] to quantify the similarity between graphs [53, 56, 57]. This kernel is based on the Weisfeiler-Leman test of isomorphism [58, 59], and thus naturally takes into account the labels of the nodes in the two graphs. We fixed the number of kernel iterations to 5. We also fixed T = 25 for our algorithm. For each method and each train-test split, we used a separate 5-fold cross-validation procedure to tune the coefficient of error term C over the set {0.1, 1, 10} for training an independent SVM model on the training portion.

Table 1 summarizes the performance of different methods for each fraction of the data. As the numbers in bold indicate, our method generally outperformed the other methods across the datasets. We observe that on some datasets the discrepancy between the average test accuracy of two algorithms is massive for every fraction. Note that though OTC performs the best on DHFR, the performance of most methods is similar (except REC, which lags behind). In contrast, REC performs better than all methods except OTC on MSRC-9. This seems to suggest that REC performs well on graphs with non-bipartite graphs, it has a nice physical interpretation in that any random walk on the graph always converges to this distribution irrespective of the graph structure.
Figure 1: **Comparison on standard graph datasets.** The top row shows the average test accuracy and corresponding standard deviation for our method (OTC) and state-of-the-art baselines for different fractions of training data. The bottom row compares the corresponding compression times. Our method outperforms the other methods in terms of both accuracy and compression time.

strong connectivity, while others might be better on data with a long backbone besides these ring structures. We believe robust performance of OTC across these datasets comprising graphs with vastly different topologies underscores the promise of our approach.

Further, as Fig. 1 shows, OTC performed best in terms of compression time as well. We emphasize that the discrepancy in compression times became quite stark for larger datasets (i.e., DHFR and Mutagenicity). To provide more evidence on the scalability of our method, we also experimented with the larger Tox21AR-LBD data,[1] which consists of about 8600 graphs. Both our method and Algebraic distance performed very well in terms of classification accuracy (∼97%) on this data. Our approach took in total about 39 seconds to compress graphs in this dataset to 90% (low compression), and about 41 seconds in total to compress to 10% (high compression). In contrast, the Algebraic distance method took about 48 seconds to compress to 90% and a significantly longer time, i.e., 3.5 minutes to compress to 10%. The other baselines failed to compress this data.

### 4.2 Compressing synthetic and real examples

We now describe our second set of experiments with both synthetic and real data to show that our method can be seeded with useful prior information toward preserving interesting patterns in the compressed structures. This flexibility in specifying the prior information makes our approach especially well-suited for downstream tasks, where domain knowledge is often available.

Fig. 2 demonstrates the effect of compressing a synthetic tree-structured graph. The penultimate level consists of four nodes, each of which has four child nodes as leaves. We introduce asymmetry with respect to the different nodes by specifying different combinations of $c(e)$ for edges $e$ between the leaf nodes and their parents: there are respectively one, two, three and four heavy edges (i.e. with $c(e) = 0.5$) from the internal nodes 1, 2, 3, and 4 to their children, i.e., the leaves in their subtree.

As shown in Fig. 2, our method adapts to the hierarchical structure with a change in the amount of compression from just one node to about three-fourths of the entire graph. We also show mean-
Figure 2: **Visualizing compression on synthetic and real examples.** (a) A synthetic graph structured as a 4-ary tree of depth 2. The root 0 is connected to its neighbors by edges having $c(e) = 0.3$. All the other edges have either $c(e) = 0.5$ (thickest) or $c(e) = 0.1$ (lightest). The left out portions are shown in gray. (b) Leaf node 13, which is in the same subtree as three nodes with heavy edges 14-16, is the first to go. (c) Proceeding further, 9 and 10 are left out followed by the remaining nodes (i.e. 5, 6, 7) connected by light edges. (d) When the graph is compressed to 5 vertices, only the root and its neighbors remain despite bulkier subtrees, e.g. the one with node 4 and its neighbors, that are discarded. Thus, our method yields meaningful compression on this synthetic example. (e-f) Compressed structures pertaining to some sample graphs from real datasets that have some other interesting motifs. In each case, the compressed graph consists of red edges and the incident vertices, while the discarded parts are shown in gray. All the figures here are best viewed in color.

Figure 2 indicates meaningful compression on some examples from real datasets. The bottom row of Fig. 2 shows two such examples, one each from Mutagenicity and MSRC-21C. For these graphs, we used the same specification for $c$ as in section 4.1. The example from Mutagenicity contains patterns such as rings and backbone structures that are ubiquitous in molecules and proteins. Likewise, the other example is a good representative of the MSRC-21C dataset from computer vision.

Thus, our method encodes prior information, and provides fast and effective graph compression for downstream applications.

**Acknowledgments**

We thank the anonymous reviewers for their thoughtful questions that led to sections 3.3 and 3.4 and experiments on the Tox21 data. We are grateful to Andreas Loukas for the code of their algorithm [1]. VG and TJ were partially supported by a grant from the MIT-IBM collaboration.
References


A Supplementary Material

We provide here proofs of all the results from the main text.

Proof of Theorem 1

Proof. We will prove the result by contradiction.

Suppose there is some \( \hat{e} \in E \) such that the optimal solution assigns \( J^+(\hat{e}) > 0 \) and \( J^-(\hat{e}) > 0 \). Then we note that

\[
a \triangleq \min\{J^+(\hat{e}), J^-(\hat{e})\} > 0 .
\]

Consider an alternate solution \((\hat{J}^+, \hat{J}^-)\) such that \( \forall e \in E \),

\[
\hat{J}^+(e) = \begin{cases} J^+(e) - a & \text{if } e = \hat{e} \\ J^+(e) & \text{if } e \in E \setminus \{\hat{e}\} \end{cases}
\]

\[
\hat{J}^-(e) = \begin{cases} J^-(e) - a & \text{if } e = \hat{e} \\ J^-(e) & \text{if } e \in E \setminus \{\hat{e}\} \end{cases}
\]

Clearly, \((\hat{J}^+, \hat{J}^-)\) is feasible for (1). Moreover, it achieves a lower value of the objective than the optimal solution \((J^+, J^-)\). Therefore, \((J^+, J^-)\) cannot be optimal.

\[\square\]

Proof of Theorem 2

Proof. We introduce non-negative Lagrangian vectors \( \alpha \) and \( \beta \), respectively, for the constraints \( Ft \leq c \) and \(-c \leq Ft\). We consider the terms in the objective that depend on \( t \)

\[
g(t) = t^\top(\rho_1 - \rho_0) + \alpha^\top(Ft - c) - \beta^\top(Ft + c) .
\]

The gradient \( \nabla g(t) \) must vanish at optimality, so

\[
\rho_1^* - \rho_0 + F^\top(\alpha - \beta) = 0 .
\]

The first part of the theorem follows immediately by defining \( \eta = \beta - \alpha \). A closer look at (1) reveals that \( \eta = J^- - J^+ \) is, in fact, the net flow along the edges \( e^- \) from (1).

Now, we prove the second part. By definition, in order for an edge \( e \) to be active, at least one of \( e^+ \) and \( e^- \) must be active, i.e., we must have \( J^+(e) + J^-(e) > 0 \). On the other hand, Theorem 1 implies that at least one of \( J^+(e) \) and \( J^-(e) \) is 0 for each \( e \in E \) in the optimal solution. Combining these facts, we have that for any active edge \( e \), exactly one of \( e^+ \) and \( e^- \) is active, i.e., exactly one of the inequalities \( J^+(e) > 0 \) and \( J^-(e) > 0 \) must hold. This immediately implies, by complementary slackness, that exactly one of \( \alpha(e) \) or \( \beta(e) \) is 0. Thus, for any active edge \( e \), either the lower bound or the upper bound on \( Ft(e) \) in the constraints \(-c(e) \leq Ft(e) \leq c(e)\) must become tight. Therefore, we must have \( Ft(e) \in \{\pm c(e)\} \).

\[\square\]

Proof of Theorem 3

Proof. Note that since at least one coordinate of \( \epsilon \) is strictly greater than 0, the feasible region is non-empty, and consequently, a unique projection exists. We introduce variables \( \alpha \in \mathbb{R} \) and \( \beta \in \mathbb{R}^d_+ \), and form the Lagrangian

\[
L(x, \alpha, \beta) = \frac{1}{2}||x - y||^2 - \alpha((x \circ \epsilon)^\top 1 - 1) - \beta^\top(x \circ \epsilon) .
\]

We now write the KKT conditions for the optimal solution \( x \). For each \( j \in [d] \), we must have

\[
x_j - y_j - \alpha \epsilon_j - \beta_j \epsilon_j = 0
\]

\[
\epsilon_j x_j \geq 0
\]

\[
\beta_j \geq 0
\]

\[
\epsilon_j x_j \beta_j = 0 .
\]

Additionally,

\[
\sum_{j=1}^d \epsilon_j x_j = 1 .
\]
Clearly, for $j \in [d] \triangleq \{1, 2, \ldots, d\}$, $\epsilon_j = 0 \implies x_j = y_j$. Therefore, without loss of generality we assume in the rest of the proof that $\epsilon_j > 0$ for all $j$. Then we can immediately simplify the KKT conditions to

$$x_j - y_j - \alpha \epsilon_j - \beta_j \epsilon_j = 0$$  \hphantom{\text{(10)} \atop \text{x}_j \geq 0 } \hphantom{\beta_j \geq 0} \hphantom{\text{x}_j \beta_j = 0} \hphantom{\sum_{j=1}^d \epsilon_j x_j = 1} \tag{10}$$

$$x_j \geq 0$$  \hphantom{\beta_j \geq 0} \hphantom{\text{x}_j \beta_j = 0} \hphantom{\sum_{j=1}^d \epsilon_j x_j = 1} \tag{11}$$

$$\beta_j \geq 0$$  \hphantom{\text{x}_j \beta_j = 0} \hphantom{\sum_{j=1}^d \epsilon_j x_j = 1} \tag{12}$$

$$x_j \beta_j = 0$$  \hphantom{\sum_{j=1}^d \epsilon_j x_j = 1} \tag{13}$$

$$\sum_{j=1}^d \epsilon_j x_j = 1$$ \hphantom{\text{x}_j \beta_j = 0} \hphantom{\sum_{j=1}^d \epsilon_j x_j = 1} \tag{14}$$

We note that $x_j > 0$ \hphantom{\beta_j = 0} \hphantom{y_j + \alpha \epsilon_j > 0} \hphantom{(\epsilon_j > 0) \implies y_j / \epsilon_j > -\alpha}$, whereas

$$x_j = 0 \implies y_j + \alpha \epsilon_j = -\beta_j \epsilon_j \implies y_j + \alpha \epsilon_j \leq 0 \implies y_j / \epsilon_j \leq -\alpha.$$ \hphantom{(10) \atop \text{y}_j \geq 0 \implies y_j / \epsilon_j \leq -\alpha} \tag{15}$$

This shows that the zero coordinates $x_j$ correspond to smaller values of $y_j / \epsilon_j$. Thus, we can sort the indices $j$ in non-increasing order based on the ratio $y_j / \epsilon_j$, reorder $x$ according to the sorted indices, and find an index $\ell \in [d]$ such that $x_j > 0$ for $j \in [\ell]$ and 0 for $\ell < j \leq d$. Without loss of generality, we therefore assume that

$$x_1 \geq x_2 \ldots \geq x_\ell > 0 = x_{\ell+1} \ldots = x_d,$$

and

$$y_1 / \epsilon_1 \geq y_2 / \epsilon_2 \ldots \geq y_\ell / \epsilon_\ell.$$ \hphantom{(15) \atop \text{y}_1 / \epsilon_1 \geq y_2 / \epsilon_2 \ldots \geq y_\ell / \epsilon_\ell} \tag{16}$$

We then have from (14) that

$$1 = \sum_{j=1}^d \epsilon_j x_j = \sum_{j=1}^\ell \epsilon_j x_j = \sum_{j=1}^\ell \epsilon_j (y_j + \alpha \epsilon_j)$$

$$\implies \alpha = \frac{1 - \sum_{j=1}^\ell \epsilon_j y_j}{\sum_{j=1}^\ell \epsilon_j^2}.$$ \hphantom{(15) \atop \text{y}_1 / \epsilon_1 \geq y_2 / \epsilon_2 \ldots \geq y_\ell / \epsilon_\ell} \tag{16}$$

Thus, our task essentially boils down to finding the number of positive coordinates $\ell$. We now show that

$$\ell = \max \left\{ j \in [d] \mid y_j + \epsilon_j \left(1 - \frac{\sum_{i=1}^j \epsilon_i y_i}{\sum_{i=1}^j \epsilon_i^2}\right) > 0 \right\}.$$ \hphantom{(16) \atop \text{y}_1 / \epsilon_1 \geq y_2 / \epsilon_2 \ldots \geq y_\ell / \epsilon_\ell} \tag{16}$$

First consider $j < \ell$. Then $y_j / \epsilon_j > -\alpha$ for $j \in [\ell]$. Noting that $\epsilon_j > 0$ for all $j$ and using (16), we must have

$$y_j + \epsilon_j \left(1 - \frac{\sum_{i=1}^j \epsilon_i y_i}{\sum_{i=1}^j \epsilon_i^2}\right)$$

$$= \frac{\epsilon_j}{\sum_{i=1}^j \epsilon_i^2} \left(y_j \sum_{i=1}^j \epsilon_i^2 - \sum_{i=1}^j \epsilon_i y_i\right)$$

which has the same sign as

$$y_j \sum_{i=1}^j \epsilon_i^2 - \sum_{i=1}^j \epsilon_i y_i$$

$$= y_j \sum_{i=1}^j \epsilon_i^2 - \sum_{i=1}^j \epsilon_i y_i + \sum_{i=1}^{\ell} \epsilon_i y_i - \sum_{i=1}^{\ell} \epsilon_i y_i$$

$$= y_j \sum_{i=1}^j \epsilon_i^2 - \sum_{i=1}^{\ell} \epsilon_i y_i + \alpha \sum_{i=1}^\ell \epsilon_i^2$$

$$= \left(\frac{y_j}{\epsilon_j} + \alpha\right) \sum_{i=1}^j \epsilon_i^2 + \sum_{i=1}^{\ell} \epsilon_i^2 \left(\frac{y_i}{\epsilon_i} + \alpha\right)$$

$$> 0.$$
Now consider \( j = \ell \). Since \( y_\ell / \epsilon_\ell > -\alpha \) and \( \epsilon_\ell > 0 \), we have \( y_\ell + \alpha \epsilon_\ell > 0 \). Thus
\[
y_j + \epsilon_j \frac{1 - \sum_{i=1}^{j} \epsilon_i y_i}{\sum_{i=1}^{j} \epsilon_i^2} = y_\ell + \epsilon_\ell \frac{1 - \sum_{i=1}^{\ell} \epsilon_i y_i}{\sum_{i=1}^{\ell} \epsilon_i^2} = y_\ell + \alpha \epsilon_\ell > 0.
\]

Finally, we consider \( \ell < j \leq d \). We note that
\[
y_j + \epsilon_j \frac{1 - \sum_{i=1}^{j} \epsilon_i y_i}{\sum_{i=1}^{j} \epsilon_i^2} = \frac{\epsilon_j}{\sum_{i=1}^{j} \epsilon_i^2} \left( y_j \frac{\sum_{i=1}^{j} \epsilon_i^2}{\epsilon_j} + 1 - \sum_{i=1}^{j} \epsilon_i y_i \right),
\]
which has the same sign as
\[
y_j \frac{\sum_{i=1}^{j} \epsilon_i^2}{\epsilon_j} + 1 - \sum_{i=1}^{j} \epsilon_i y_i = y_j \frac{\sum_{i=1}^{j} \epsilon_i^2}{\epsilon_j} + 1 - \sum_{i=1}^{\ell} \epsilon_i y_i - \sum_{i=\ell+1}^{j} \epsilon_i y_i = y_j \frac{\sum_{i=1}^{j} \epsilon_i^2}{\epsilon_j} + \alpha \sum_{i=1}^{\ell} \epsilon_i^2 - \sum_{i=\ell+1}^{j} \epsilon_i y_i = \left( \frac{y_j}{\epsilon_j} + \alpha \right) \sum_{i=1}^{\ell} \epsilon_i^2 + \sum_{i=\ell+1}^{j} \epsilon_i^2 \left( \frac{y_j}{\epsilon_j} - \frac{y_i}{\epsilon_i} \right),
\]
by leveraging the sorted property in (15) and the fact that \( y_j / \epsilon_j \leq -\alpha \) for \( j \in [\ell] \).

Therefore, we have shown that \( y_j + \epsilon_j \frac{1 - \sum_{i=1}^{j} \epsilon_i y_i}{\sum_{i=1}^{j} \epsilon_i^2} > 0 \) for all \( j \in [\ell] \), and at most 0 for \( \ell < j \leq d \). Algorithm [1] implements this procedure, and that proves its correctness. The \( O(d \log d) \) time complexity is due to the cost of sorting the indices \( j \in [d] \) based on \( y_j / \epsilon_j \). \( \square \)

**Proof of Theorem 4**

**Proof:** Recall the formulation (6):
\[
\min_{\epsilon \in \mathcal{E}_k} \max_{\tilde{\rho}_1 \circ \epsilon \in \Delta(V)} \max_{t \in T_{F,c}} \phi(t, \epsilon, \tilde{\rho}_1) = \lambda I_2(\epsilon, \tilde{\rho}_1) + \epsilon \sum_{i=1}^{\ell} \epsilon_i^2 + \sum_{i=\ell+1}^{j} \epsilon_i^2 \left( \frac{y_j}{\epsilon_j} - \frac{y_i}{\epsilon_i} \right).
\]

Making the constraints \( \mathcal{E}_k \) explicit, we get
\[
\min_{\epsilon \in \{0, 1\}^{|V|}} \min_{1 \leq k} \left( \min_{\tilde{\rho}_1 \in \Delta(V)} \max_{t \in T_{F,c}} \phi(t, \epsilon, \tilde{\rho}_1) \right).
\]

Note that for any fixed \( \epsilon \) (a) \( \{ \tilde{\rho}_1 \in \mathbb{R}^{|V|} \mid (\tilde{\rho}_1 \circ \epsilon) \in \Delta(V) \} \) is convex, and \( T_{F,c} \) is convex and compact, (b) \( \phi(t, \tilde{\rho}_1) \) is continuous, and (c) for every fixed \( t, \phi(t, \epsilon, \cdot) \) is convex in \( \tilde{\rho}_1 \); while for every fixed \( \tilde{\rho}_1 \), \( \phi(\epsilon, \cdot, \tilde{\rho}_1) \) is linear (thus concave) in \( t \). Therefore, invoking the Sion’s minimax theorem [60], we can swap the order of min and max within the parentheses in (17), and obtain
\[
\min_{\epsilon \in \{0, 1\}^{|V|}} \min_{1 \leq k} \left( \max_{t \in T_{F,c}} \min_{\tilde{\rho}_1 \in \Delta(V)} \phi(t, \epsilon, \tilde{\rho}_1) \right).
\]
We introduce Lagrangian variables $\nu \in \mathbb{R}^{|V|}$ and $\zeta \in \mathbb{R}$, respectively, for the simplex constraints (a) $0 \leq \tilde{\rho}_1 \odot \epsilon$ and (b) $(\tilde{\rho}_1 \odot \epsilon)^\top 1 = 1$ to get

$$\Phi(\epsilon, t, \tilde{\rho}_1, \nu, \zeta) \triangleq \phi(\epsilon, t, \tilde{\rho}_1) + \zeta((\tilde{\rho}_1 \odot \epsilon)^\top 1 - 1) - \nu^\top (\tilde{\rho}_1 \odot \epsilon).$$

Applying the optimality conditions, we note for any fixed pair $(\epsilon, t)$, the corresponding optimal $\tilde{\rho}_1$ must satisfy

$$\partial_{\tilde{\rho}_1} \Phi(\epsilon, t, \tilde{\rho}_1, \nu, \zeta) = 0,$$

whereby

$$\tilde{\rho}_1 = -\left(\frac{(t - \nu) \odot \epsilon + \zeta \epsilon}{\lambda}\right).$$

Plugging in $\tilde{\rho}_1$ from (20) into (19), we thus have the following equivalent dual formulation for (18)

$$\min_{\epsilon \in \{0, 1\}^{|V|}} \max_{t \in T_F} \max_{\nu \in \mathbb{R}^{|V|}, \zeta \in \mathbb{R}} \mathcal{M}(\epsilon, t, \nu, \zeta),$$

where

$$\mathcal{M}(\epsilon, t, \nu, \zeta) = -\frac{1}{2\lambda}||((t - \nu + \zeta 1) \odot \epsilon)^1 - t^\top \rho_0 - \zeta||.$$

Invoking the first order convex optimality condition for constrained optimization, the $\hat{\epsilon}$ obtained from relaxation of (6) is optimal if and only if

$$0 \in \left\{ \partial_\epsilon \max_{t \in T_F} \max_{\nu \in \mathbb{R}^{|V|}, \zeta \in \mathbb{R}} \mathcal{M}(\epsilon, t, \nu, \zeta) + N \right\},$$

where $N$ is the normal cone of the relaxed constraints

$$\tilde{E}_k = \left\{ \epsilon \in \{0, 1\}^{|V|} \mid \epsilon^\top 1 \leq k \right\}.$$

Now we note that for any vector $x$, we can write $x \odot \epsilon = D(\epsilon)x$, where $D(\epsilon)$ is the diagonal matrix corresponding to $\epsilon$. Also, since $\epsilon \in \{0, 1\}^{|V|}$, we get $D(\epsilon)^\top D(\epsilon) = D(\epsilon)$. Thus, for any $x$, we have

$$||x \odot \epsilon||^2 = ||D(\epsilon)x||^2 = (D(\epsilon)x)^\top D(\epsilon)x = x^\top D(\epsilon)^\top D(\epsilon)x = x^\top D(\epsilon)x.$$

In particular, we can simplify $||(t - \nu + \zeta 1) \odot \epsilon||^2$ in (21) when we set $x$ to $t - \nu + \zeta 1$. The theorem statement then follows immediately from (22) by representing $N$ at the integral point $\epsilon^*$ and leveraging the non-negative dual parameter associated with the constraint $\epsilon^\top 1 \leq k$.

**Proof of Theorem**

**Proof.** We will use the shorthand $\tilde{\rho}_{1v}$ for $\tilde{\rho}_1(v)$, and likewise for indexing $t, \nu, \epsilon$. Using (20),

$$\tilde{\rho}_1 = -\left(\frac{(t - \nu) \odot \epsilon + \zeta \epsilon}{\lambda}\right) = -\left(\frac{t - \nu + \zeta 1}{\lambda}\right) \odot \epsilon.$$

Therefore,

$$\tilde{\rho}_1 \odot \epsilon = -\left(\frac{t - \nu + \zeta 1}{\lambda}\right) \odot \epsilon \odot \epsilon = \left(\frac{t - \nu + \zeta 1}{\lambda}\right) \odot \epsilon = \tilde{\rho}_1,$$

since $\epsilon \in \{0, 1\}^{|V|}$. We write one of the KKT conditions for optimality

$$\tilde{\rho}_1 \odot \epsilon \odot \nu = \tilde{\rho}_1 \odot \nu = 0.$$

We consider the different cases. Note that for $v \in V$, using (23), we have

$$\tilde{\rho}_{1v} > 0 \implies \nu_v = 0 \implies \tilde{\rho}_{1v} = -\left(\frac{t_v + \zeta}{\lambda}\right) \epsilon_v,$$

(24)
\[ \tilde{\rho}_v = 0 \implies (t_v - \nu_v + \zeta)\epsilon_v = 0 \implies \nu_v \epsilon_v = (t_v + \zeta)\epsilon_v. \tag{25} \]

But since \( \nu_v \geq 0 \), and \( \epsilon_v \in \{0, 1\} \), we note that \( \nu_v \epsilon_v \geq 0 \). Then, by (25), we have
\[ \nu_v \epsilon_v \geq 0 \implies \frac{(t_v + \zeta)\epsilon_v}{\lambda} \leq 0 = \tilde{\rho}_v. \tag{26} \]

Combining (24) and (26), we can write
\[ \tilde{\rho}_v = \max \left\{ -\frac{(t_v + \zeta)\epsilon_v}{\lambda}, 0 \right\} = \epsilon_v \max \left\{ -\frac{(t_v + \zeta)}{\lambda}, 0 \right\} \tag{27} \]

since \( \epsilon_v \geq 0 \) for all \( v \in V \) and \( \lambda > 0 \). Therefore, we get \( \tilde{\rho}_1 = \frac{\epsilon}{\lambda} \odot r_+ \), where \( r = -(t + \zeta 1) \), and \( r_+ \) is computed by setting the negative coordinates of \( r \) to 0.

Moreover, since \( \tilde{\rho}_1 \odot \nu = 0 \), we can eliminate \( \nu \) from (19) and write (17) as
\[
\min_{\epsilon \in E_k} \max_{t \in T_{F,c}} \max_{\zeta \in \mathbb{R}} t^\top (\tilde{\rho}_1 - \rho_0) + \frac{\lambda}{2} ||\tilde{\rho}_1||^2 + \zeta (\tilde{\rho}_1 \top 1 - 1)
\]

Substituting for \( \tilde{\rho}_1 \) from (27), we obtain the following equivalent problem
\[
\min_{\epsilon \in E_k} \max_{t \in T_{F,c}} \max_{\zeta \in \mathbb{R}} -\frac{1}{\lambda}\epsilon^\top (\epsilon \odot (2r - r_+)) - t^\top \rho_0 - \zeta,
\]

which can be written as
\[
\min_{\epsilon \in E_k} \max_{t \in T_{F,c}} \max_{\zeta \in \mathbb{R}} -\frac{1}{2\lambda} \sum_{v : t_v \leq -\zeta} \epsilon_v (t_v + \zeta)^2 - t^\top \rho_0 - \zeta
\]

\[= \min_{\epsilon \in E_k} \max_{t \in T_{F,c}} \max_{\zeta \in \mathbb{R}} -\frac{1}{2\lambda} \sum_{v : t_v \leq -\zeta} \epsilon_v (t_v + \zeta)^2 - t^\top \rho_0 - \zeta
\]

\[= \min_{\epsilon \in E_k} \max_{t \in T_{F,c}} \max_{\zeta \in \mathbb{R}} -\frac{1}{2\lambda} \sum_{v : t_v \leq -\zeta} (\epsilon_v (t_v + \zeta))^2 + 2\lambda \epsilon_v \rho_0,v - t^\top \rho_0 - \zeta.
\]

\[\square\]