Solving a Special Type of Optimal Transport Problem by a Modified Hungarian Algorithm

author names withheld

Under Review for OPT 2022

Abstract

We observe that computing empirical Wasserstein distance in the independence test is an optimal transport (OT) problem with a special structure. This observation inspires us to study a special type of OT problem and propose a modified Hungarian algorithm to solve it exactly. For an OT problem between marginals with m and n atoms ($m \geq n$), the computational complexity of the proposed algorithm is $\mathcal{O}(m^2 n)$. Computing the empirical Wasserstein distance in the independence test requires solving this special type of OT problem, where we have $m = n^2$. The associate computational complexity of our algorithm is $\mathcal{O}(n^5)$, while the order of applying the classic Hungarian algorithm is $\mathcal{O}(n^6)$. Numerical experiments validate our theoretical analysis. Broader applications of the proposed algorithm are discussed at the end.

1. Introduction

One appealing application of optimal transport (OT) and Wasserstein distance [17, 19] is independence test. The Wasserstein distance between two distributions $\mu_1, \mu_2 \in Z$ is defined as:

$$W(\mu_1, \mu_2) := \inf \left\{ \int_{Z^2} d(z, z') d\gamma(z, z') : \ \gamma(\mu_1, \mu_2) \text{ is a joint distribution of } \mu_1 \text{ and } \mu_2 \right\},$$

where (Z, d) is a metric space (w.l.o.g., we consider 1-Wasserstein distance in this paper). Wasserstein distance is a metric on probability measures [19]. Hence, to test the independence between variables $Y \sim \nu_1$ and $Z \sim \nu_2$, people utilize the Wasserstein distance between the joint distribution of ν_1, ν_2 and the product distribution of ν_1, ν_2, i.e., $W_p(\gamma(\nu_1, \nu_2), \nu_1 \otimes \nu_2)$. While the statistical properties of this approach have been intensely investigated [12, 15, 20], there is not much discussion on the computational aspects. In this paper, we will talk about '*How to calculate the empirical Wasserstein distance in the independence test efficiently?*'

In practice, given n i.i.d. samples $\{(y_1, z_1), ..., (y_n, z_n)\}$ generated from $Y \times Z$, we build the statistic—$W_p(\gamma(\widehat{\nu}_1, \widehat{\nu}_2), \widehat{\nu}_1 \otimes \widehat{\nu}_2)$, where $\widehat{\nu}$ means the associate empirical distribution—to test the independence. We will show later that computing this statistic is equivalent to solving the following optimization problem:

$$\min_{X^\circ \in \Pi^\circ} \sum_{i,j,k=1}^{n} d((x_i, y_j), (x_k, y_k))^p X^\circ_{ij;k}, \quad \Pi^\circ = \left\{ X^\circ_{ij;k} \geq 0 \middle| \sum_{k=1}^{n} X^\circ_{ij;k} = \frac{1}{n^2}, \sum_{i,j=1}^{n} X^\circ_{ij;k} = \frac{1}{n} \right\}. \tag{1}$$

© .

The problem (1) is an OT problem between two marginals where one is an n dimension uniform distribution and the other is an n^2 dimension uniform distribution. Inspired by this structure and Birkhoff's theorem [3], we study the following special type of OT problem:

$$\min_{X' \in \mathcal{U}'} \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{1}{m} X'_{ij} C_{ij}, \quad \mathcal{U}' = \left\{ X'_{ij} = \{0,1\} \middle| \sum_{j=1}^{n} X'_{ij} = 1, \sum_{i=1}^{m} X'_{ij} = m_j, \sum_{j=1}^{n} m_j = m \right\}. \tag{2}$$

where $0 < n \leq m$, $i = 1, ..., m$, $j = 1, ..., n$, and m_j are positive integers. In essence, problem (1) is a special case of problem (2) where $m_j = n, m = n^2$, seeing more details in Section 3.

Then our question is *'How to solve problem* (2) *efficiently?'* One may observe that problem (2) is embedded with a similar structure to the assignment problem. However, different from the permutation matrix, $X' \in \mathcal{U}'$ is an $m \times n$ matrix instead of a square matrix and has multiple entries of 1 in each column instead of only one entry (we call this property as 'one-to-many'). In this way, we are not able to directly apply algorithms for the assignment problem like the Hungarian algorithm [13] to solve the problem (2). An approach to obtain the precise solution to (2) is first to duplicate the columns of C and X', and then apply the Hungarian algorithm. The computational complexity of this approach is $\mathcal{O}(m^3)$. In this paper, we will propose a modified Hungarian algorithm. The algorithm utilizes the 'one-to-many' property and specializes in solving the special type of OT problem (2) with a provable lower order—$\mathcal{O}(m^2 n)$.

Back to the Wasserstein distance-based independence test problem (1), the resulting computational complexity of applying the proposed algorithm is $\mathcal{O}(n^5)$ while the order of applying the classic Hungarian algorithm is $\mathcal{O}(n^6)$.

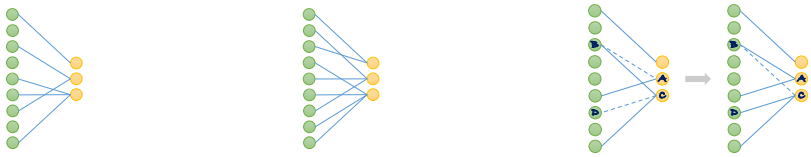
Related work: Notice that our modified Hungarian algorithm is an exact OT solver. There are also a bunch of approximation algorithms [4, 5, 8, 10, 21]. Note that some scenarios may require the precise solution of OT problems, and paper [6] demonstrates the favorable numerical performance of the exact algorithms over the approximation algorithms. Therefore, developing efficient exact algorithms is meaningful.

Our contributions: Firstly, we find out that computing the Wasserstein distance in the independence test (1) belongs to a special type of OT problem (2). Secondly, we propose a modified Hungarian algorithm to solve problem (2). The algorithm modifies the Hungarian algorithm to deal with the scenario where two marginals are of different dimensions and atoms in one marginal have multiple assignments ('one-to-many' property). Thirdly, we adopt our algorithm to the independence test problem (1). We also talk about other applications of the proposed algorithm, seeing Section 5.

2. Modified Hungarian algorithm

In this section, we propose a modified Hungarian algorithm to solve the problem (2).

We first review the basics of Hungarian algorithm [13, 18]. Suppose that a bipartite graph $G = (V, E)$, where $V = V_1 \cup V_2, E \subset V_1 \times V_2$, has weight $w(\cdot)$ for each edge $e \in E$. $M \subset E$ is a matching if every node of G coincides with at most one edge of M. Kuhn-Munkres theorem [13] illustrates that solving the assignment problem is equivalent to finding a perfect matching on the equality graph associated with some feasible labeling. (The feasible labeling is defined as a function $l : V \to \mathbb{R}$ satisfying $l(v_1) + l(v_2) \geq w(v_1, v_2), \forall v_1 \in V_1, v_2 \in V_2$, and the equality graph w.r.t. labeling l is $G' = (V, E_l)$, where $E_l = \{(v_1, v_2) : l(v_1) + l(v_2) = w(v_1, v_2)\}$.) Hence, Hungarian algorithm solves the assignment problem by looking for a perfect matching on some equality graph.



pseudo-matching perfect pseudo-matching pseudo-augmenting process

Figure 1: Examples when $n = 3, m = 9, m_1 = 2, m_2 = 3, m_3 = 4$. The solid line means that the edge belongs to the pseudo-matching. The dashed line indicates that the edge belongs to the equality graph but does not belong to the pseudo-matching.

In problem (2), $X' \in \mathcal{U}'$ has one entry of 1 in each row, multiple entries of 1 in each column and $0'$s elsewhere. Since a permutation matrix corresponds to a (perfect) matching in the bipartite graph, we define 'pseudo-matching' in the bipartite graph to describe X': In the bipartite graph G, where $|V_1| = m, |V_2| = n$. $PM \subset E$ is a **pseudo-matching** if every node of V_1 coincides with at most one edge of PM and jth node of V_2 coincides with at most m_j edges of PM, where $\sum_{j=1}^{n} m_j = m$. Further, if every node of V_1 coincides with one edge of PM and jth node of V_2 coincides with m_j edges of PM, PM is called a **perfect pseudo-matching**. See examples in Figure 1.

We develop a modified Kuhn-Munkres theorem (the theorem and its proof can be found in the Appendix). It demonstrates that we only need to find a perfect pseudo-matching on some equality graph to solve the problem (2). In this way, we design a modified Hungarian algorithm (Algorithm 1). It improves either the feasible labeling (adding edges to the associate equality graph) or the pseudo-matching until the pseudo-matching is perfect on some equality graph w.r.t. some feasible labeling. The algorithm improves the pseudo-matching by generating pseudo-augmenting paths and then exchanging the edge status along the paths. This process is called the pseudo-augmenting process. Also, we force the pseudo-augmenting paths emanating from V_2, which have a lower order of nodes. See the relevant definitions in Definition 1 and examples in Figure 1 .

Definition 1 (Free, matched, pseudo-matched, pseudo-alternating path, pseudo-augmenting path) *Let PM be a pseudo-matching of $G = (V, E)$. • If the node v is in V_1, it is pseudo-matched if it is an endpoint of some edge in PM; if the node v is the jth node in V_2, it is pseudo-matched if it is an endpoint of m_j edges in PM. Otherwise, the node is free. • If the node $v \in V$, we say it is matched if it is an endpoint of some edge in PM. • A path is pseudo-alternating if its edge alternates between PM and $E - PM$. A pseudo-alternating path is pseudo-augmenting if both its endpoints are free.*

We now analyze the computational complexity of Algorithm 1. Similar to the Hungarian algorithm [18], we keep track of $slack_{v_1} = \min_{v_2 \in S}\{l(v_1) + l(v_2) - w(v_1, v_2)\}$, $\forall v_1 \notin T$. The number of edges of the pseudo-matching increases by 1 after one loop, so we need $\mathcal{O}(m)$ loops to form a perfect pseudo-matching. There are two subroutines in each loop: (1): updating feasible labeling (Step 2); (2): improving pseudo-matching (Step 3). In procedure (1): Since there are n nodes in V_2, improving the labeling occurs $\mathcal{O}(n)$ times to build a pseudo-alternating tree. Computing α_l, updating the *slack*s and calculating the labeling cost $\mathcal{O}(m)$ each time. In procedure (2): When a new node has been added to S, it costs $\mathcal{O}(m)$ to update *slack*s, and $\mathcal{O}(n)$ nodes could be added. On the other hand, when a node has been added to T, we just remove the corresponding $slack_{v_1}$.

From the above we conclude that each loop costs $\mathcal{O}(mn)$, so the total computational complexity of Algorithm 1 to solve problem (2) is $\mathcal{O}(m^2 n)$. Note that the adoption of the Hungarian algorithm

Algorithm 1: Modified Hungarian Algorithm

Generate an initial feasible labeling l: $\forall v_2 \in V_2, l(v_2) = 0$;
$\forall v_1 \in V_1, l(v_1) = \max_{v_2 \in V_2}\{w(v_1, v_2)\}$ and initialize a pseudo-matching M in E_l;

1 **if** M *is a perfect pseudo-matching* **then**
  Stop
else
  Pick up a free node $v_{\text{free}} \in V_2$. Set $S = \{v_{\text{free}}\}, T = \emptyset$;
  for $\mathbf{v_1} \in V_1$ *is matched to* v_{free} **do**
    $T = T \cup \mathbf{v_1}$
  end
end

2 **if** $N_l(S) - T = \emptyset$ **then**
  update labeling such that forcing $N_l(S) - T \neq \emptyset : \alpha_l = \min_{v_1 \notin T, v_2 \in S}\{l(v_1) + l(v_2) - w(v_1, v_2)\}; l(v) = l(v) - \alpha_l, v \in S; l(v) = l(v) + \alpha_l, v \in T; l(v) = l(v)$, otherwise.
end

3 **if** $N_l(S) - T \neq \emptyset$ **then**
  pick $v_1 \in N_l(S) - T$;
  if v_1 *is free* **then**
    $v_{\text{free}} \to v_1$ is a pseudo-augmenting path. Pseudo-augment the pseudo-matching M. Go to Step 1;
  end
  if v_1 *is pseudo-matched to* z **then**
    extend the pseudo-matching tree: $S = S \cup \{z\}, T = T \cup \{v_1\}$;
    for $\mathbf{v_1} \in V_1$ *is matched to* z **do**
      $T = T \cup \mathbf{v_1}$;
    end
  end
  Go to Step 2.
end

has an order of $\mathcal{O}(m^3)$. Hence, the proposed modified Hungarian algorithm will outperform, especially when $m \gg n$.

3. Independence test using the Wasserstein distance

In this section, we focus on the special case of the OT problem (2)—independence test using the Wasserstein distance, and apply the modified Hungarian algorithm to solve it.

Suppose that we have n i.i.d. samples $\{(y_1, z_1), ..., (y_n, z_n)\}$, where $(y_i, z_j) \sim Y \times Z$. If we utilize the statistic $W_p(\gamma(\widehat{\nu}_1, \widehat{\nu}_2), \widehat{\nu}_1 \otimes \widehat{\nu}_2)$ to test the independence between Y and Z, we need to solve the following optimization problem:

$$\min_{X \in \Pi} \sum_{i,j,k,l=1}^{n} d((x_i, y_j), (x_k, y_l)) X_{ij;kl}, \tag{3}$$

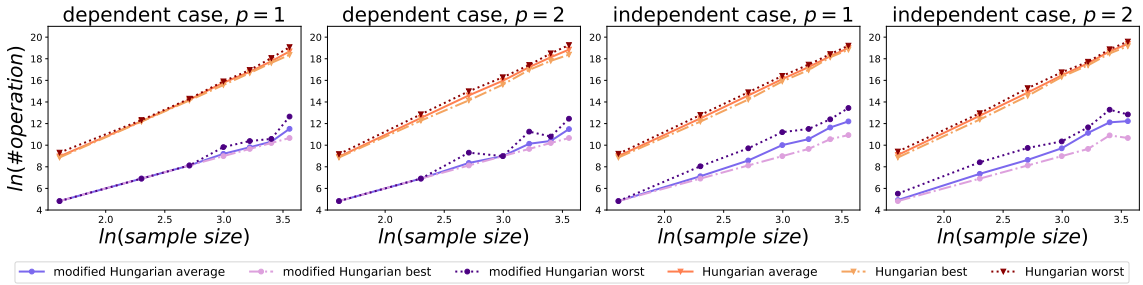


Figure 2: Results of CIFAR10

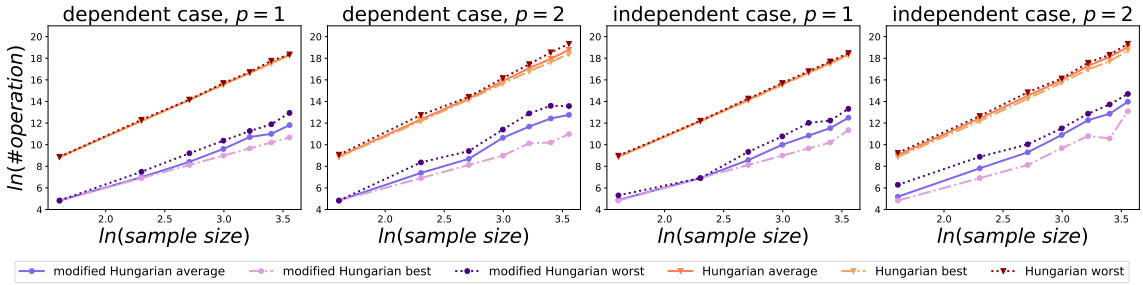


Figure 3: Results of Wisconsin breast cancer data

where

$$\Pi = \left\{ X_{ij;kl} \geq 0 \middle| \sum_{k,l=1}^{n} X_{ij;kl} = \frac{1}{n^2}, \sum_{i,j=1}^{n} X_{ij;kl} = \begin{cases} \frac{1}{n} & k = l \\ 0 & k \neq l \end{cases}, \forall i,j,k,l = 1,...,n. \right\}.$$

It is worth noting that $X_{ij;kl} = 0,\ \forall k \neq l$. If we further let $X^{\circ}_{ij;k} := \sum_{l=1}^{n} X_{ij;kl}$, problem (3) could be reformulated as problem (1) and is equivalent to the following optimization problem: (The proof and relevant discussions are relegated to the Appendix.)

$$\min_{X' \in \Pi'} \sum_{i,j,k=1}^{n} \frac{d((x_i, y_j), (x_k, y_k))}{n^2} X'_{ij;k}, \quad \Pi' = \left\{ X'_{ij;k} = \{0,1\} \middle| \sum_{k=1}^{n} X'_{ij;k} = 1, \sum_{i,j=1}^{n} X'_{ij;k} = n \right\}. \tag{4}$$

Problem (4) belongs to the special type of OT problem where $m_j = n, m = n^2$. Adopting the Hungarian algorithm to problem (4) costs $\mathcal{O}(n^6)$ while adopting the proposed Hungarian algorithm directly costs $\mathcal{O}(n^5)$.

4. Numerical experiments

In this section, we carry out experiments on the independence test problem (1) via CIFAR10[1] [11] and Wisconsin breast cancer[2] [7] datasets to validate the computational complexity of the modified Hungarian algorithm and the favorability of applying the proposed algorithm over applying the

1. https://www.cs.toronto.edu/~kriz/cifar.html
2. https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

Hungarian algorithm. The performance is evaluated by the number of numerical operations here. We relegate the results of the running time to the Appendix.

Experiments details: The metric in problem (1) is chosen as $d((x_i, y_j), (x_k, y_l)) = \|x_i - x_k\|_p + \|y_j - y_l\|_p$, $p = 1, p = 2$. We create one dependent case and one independent case with different sample sizes: **CIFAR10**: Each image is a 3072-dim vector. Let $X \in \mathbb{R}^{3072}$ be generated uniformly from the first 5 classes; $Y \in \mathbb{R}^{3072}$ be generated uniformly from other classes. We calculate the Wasserstein distance (1) *independent case*: between X and Y_1 (where Y_1 is the first 1536 coordinates of Y); (2) *dependent case*: between X and Z (where $Z = X_2/2 + Y_1/2$, X_2 is the last 1536 coordinates of X, Y_1 is the first 1536 coordinates of Y). **Breast cancer data:** Each instance is a 30-dim vector. Let $X \in \mathbb{R}^{30}$ be generated uniformly from the benign class, and $Y \in \mathbb{R}^{30}$ be generated uniformly from the malignant class. We calculate the Wasserstein distance (1) *independent case*: between X_1 and Y_2 (where X_1 is the first 5 coordinates of X, Y_2 the last 25 coordinates of Y); (2) *dependent case*: between X and Z (where $Z = X_1 * Y_1$, X_1 is the first 5 coordinates of X, Y_1 is the first 5 coordinates of Y, $*$ means the coordinate-wise product). We run the algorithms on each case 10 times and plot the worst, best and average number of numerical operations.

Experiments results: The results are presented in Figure 2, 3. The figures illustrate that the proposed algorithm gains a factor n in computational complexity when solving problem (1). To be more specific, notice that the slope of $\ln$(number of numerical operations) over $\ln$(sample size) indicates the order of the associate algorithm, and the slope of our algorithm is around 5 while the slope of the Hungarian algorithm is around 6. This observation implies that the order of applying our algorithm is $\mathcal{O}(n^5)$ while the order of applying the Hungarian algorithm is $\mathcal{O}(n^6)$. It is consistent with our theoretical analysis.

5. Discussion

We propose a modified Hungarian algorithm to solve a special type of OT problem (2). The proposed algorithm could also help solve the one-to-many assignment [22] and many-to-many assignment [23] problems. In this way, our algorithm could be applied to solve related practical problems in service assignment problems [14], sensor networks [2], access control [1], etc. In addition, we could also apply our algorithm to some modern machine learning models or approaches including clustering [9] and representation learning [16].

References

[1] Gail-Joon Ahn and Hongxin Hu. Towards realizing a formal RBAC model in real systems. In *Proceedings of the 12th ACM symposium on Access control models and technologies*, pages 215–224, 2007.

[2] Manish Bhardwaj and Anantha P Chandrakasan. Bounding the lifetime of sensor networks via optimal role assignments. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1587–1596. IEEE, 2002.

[3] Garrett Birkhoff. Three observations on linear algebra. *Univ. Nac. Tacuman, Rev. Ser. A*, 5: 147–151, 1946.

[4] Antonin Chambolle and Juan Pablo Contreras. Accelerated Bregman primal-dual methods applied to optimal transport and Wasserstein Barycenter problems. *arXiv preprint arXiv:2203.00802*, 2022.

[5] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

[6] Yihe Dong, Yu Gao, Richard Peng, Ilya Razenshteyn, and Saurabh Sawlani. A study of performance of optimal transport. *arXiv preprint arXiv:2005.01182*, 2020.

[7] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[8] Pavel Dvurechensky, Alexander Gasnikov, and Alexey Kroshnin. Computational optimal transport: Complexity by accelerated gradient descent is better than by Sinkhorn's algorithm. In *International conference on machine learning*, pages 1367–1376. PMLR, 2018.

[9] Aude Genevay, Gabriel Dulac-Arnold, and Jean-Philippe Vert. Differentiable deep clustering with cluster size constraints. *arXiv preprint arXiv:1910.09036*, 2019.

[10] Arun Jambulapati, Aaron Sidford, and Kevin Tian. A direct $\tilde{O}$(1/epsilon) iteration parallel algorithm for optimal transport. *Advances in Neural Information Processing Systems*, 32, 2019.

[11] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[12] Gilles Mordant and Johan Segers. Measuring dependence between random vectors via optimal transport. *Journal of Multivariate Analysis*, 189:104912, 2022.

[13] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957.

[14] Vincent TY Ng, Boris Chan, Louis LY Shun, and Ringo Tsang. Quality service assignments for role-based web services. In *2008 IEEE International Conference on Systems, Man and Cybernetics*, pages 2219–2224. IEEE, 2008.

[15] Thomas Giacomo Nies, Thomas Staudt, and Axel Munk. Transport dependency: Optimal transport based dependency measures. *arXiv preprint arXiv:2105.02073*, 2021.

[16] Sherjil Ozair, Corey Lynch, Yoshua Bengio, Aaron Van den Oord, Sergey Levine, and Pierre Sermanet. Wasserstein dependency measure for representation learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[17] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[18] Subhash Suri. Bipartite matching & the Hungarian method. *Notes, Department of Computer Science, University of California, Santa Barbara*, 8, 2006.

[19] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer, 2009.

[20] Johannes CW Wiesel. Measuring association with Wasserstein distances. *Bernoulli*, 28(4): 2816–2832, 2022.

[21] Yiling Xie, Yiling Luo, and Xiaoming Huo. An accelerated stochastic algorithm for solving the optimal transport problem. *arXiv preprint arXiv:2203.00813*, 2022.

[22] Haibin Zhu, MengChu Zhou, and Rob Alkins. Group role assignment via a Kuhn–Munkres algorithm-based solution. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 42(3):739–750, 2011.

[23] Haibin Zhu, Dongning Liu, Siqin Zhang, Yu Zhu, Luyao Teng, and Shaohua Teng. Solving the many to many assignment problem by improving the Kuhn–Munkres algorithm with backtracking. *Theoretical Computer Science*, 618:30–41, 2016.

Appendix A. Modified Kuhn-Munkres theorem

Theorem 2 (Modified Kuhn-Munkres theorem) *If l is a feasible labeling on the weighted bipartite graph $G = (V, E)$ and $PM \subset E_l$ is a perfect pseudo-matching on the corresponding equality graph $G' = (V, E_l)$, then PM is a maximum weighted pseudo-matching.*

Proof Denote the edge $e \in E$ by $e = (e_{v_1}, e_{v_2})$. Let PM' be any perfect pseudo-matching in G (not necessarily in the equality graph E_l). And $v_1^i, i = 1, ..., m$; $v_2^j, j = 1, ..., n$ are nodes from V_1 and V_2, respectively. Since $v_1^i \in V_1$ is covered exactly once by PM' and $v_2^j \in V_2$ is covered exactly m_j times by PM', we have

$$w(PM') = \sum_{e \in PM'} w(e) \leq \sum_{e \in PM'} (l(e_{v_1}) + l(e_{v_2})) = \sum_{i=1}^{m} l(v_1^i) + \sum_{j=1}^{n} m_j l(v_2^j),$$

where the first inequality comes from the definition of the feasible labeling.

Thus, $\sum_{i=1}^{m} l(v_1^i) + \sum_{j=1}^{n} m_j l(v_2^j)$ is upper bound of the weight of any perfect pseudo-matching. Then let PM be a perfect pseudo-matching in the equality graph E_l, we have

$$w(PM) = \sum_{e \in PM} w(e) = \sum_{i=1}^{m} l(v_1^i) + \sum_{j=1}^{n} m_j l(v_2^j).$$

Hence $w(PM') \leq w(PM)$ and PM is the maximum weighted pseudo-matching. ■

A.1. Proof and relevant discussion for Section 3

Instead of direct proving (3) $\Longleftrightarrow$ (4), we prove the more general scenario first.

Proposition 3 *The optimization problem (2) is equivalent to the optimization problem (5):*

$$\min_{X \in \mathcal{U}} \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij} C_{ij}, \quad \mathcal{U} = \left\{ X_{ij} \geq 0 \middle| \sum_{j=1}^{n} X_{ij} = \frac{1}{m}, \sum_{i=1}^{m} X_{ij} = \frac{m_j}{m}, \sum_{j=1}^{n} m_j = m \right\}. \quad (5)$$

Proof First consider the following two optimization problems (6)-(7):

$$\min_{X^\dagger \in \mathcal{U}^\dagger} \sum_{i=1}^{m}\sum_{j=1}^{m} \frac{1}{m} X_{ij}^\dagger C_{ij}^\dagger, \quad \mathcal{U}^\dagger = \left\{ X_{ij}^\dagger = \{0,1\} \middle| \sum_{j=1}^{m} X_{ij}^\dagger = 1, \sum_{i=1}^{m} X_{ij}^\dagger = 1 \right\}, \tag{6}$$

where $C^\dagger$ is an $m \times m$ matrix generated by duplicating the jth column of C m_j times:

$$C_{it}^\dagger = C_{ij}, \quad 1 \le t \le m_j, j = 1;\ m_1 + ... + m_{j-1} + 1 \le t \le m_1 + ... + m_j, j \ge 2.$$

$$\min_{X^1 \in \mathcal{U}^1} \sum_{i=1}^{m}\sum_{t=1}^{m} X_{it}^1 C_{it}^1, \quad \mathcal{U}^1 = \left\{ X_{it}^1 \ge 0 \middle| \sum_{t=1}^{m} X_{it}^1 = \frac{1}{m}, \sum_{i=1}^{m} X_{it}^1 = \frac{1}{m} \right\}, \tag{7}$$

where C^1 is an $m \times m$ matrix generated by duplicating the jth column of C m_j times:

$$C_{it}^1 = C_{ij},\ 1 \le t \le m_j, j = 1;\ m_1 + ... + m_{j-1} + 1 \le t \le m_1 + ... + m_j, j \ge 2.$$

Then we denote the objective functions of the problems (2), (5), (6) and (7) by $f(X)$, $f^\dagger(X^\dagger)$, $f'(X')$ and $f^1(X^1)$, respectively.

Firstly, we prove (5) $\iff$ (7).

On the one hand, for any $X^1 \in \mathcal{U}^1$, if we let

$$X_{ij} = \sum_{t=1}^{m_1} X_{it}^1, \quad j = 1,$$

$$X_{ij} = \sum_{t=m_1+...+m_{j-1}+1}^{m_1+...+m_j} X_{it}^1, \quad j \ge 2,$$

then we have

$$X_{ij} \ge 0,$$

$$\sum_{j=1}^{n} X_{ij} = \sum_{t=1}^{m_1} X_{it}^1 + \sum_{t=m_1+...+m_{j-1}+1}^{m_1+...+m_j} X_{it}^1 = \sum_{t=1}^{m} X_{it}^1 = \frac{1}{m},$$

$$\sum_{i=1}^{m} X_{ij} = \sum_{i=1}^{m}\sum_{t=1}^{m_1} X_{it}^1 = \frac{m_1}{m}, \quad j = 1,$$

$$\sum_{i=1}^{m} X_{ij} = \sum_{i=1}^{m} \sum_{t=m_1+...+m_{j-1}+1}^{m_1+...+m_j} X_{it}^1 = \frac{m_j}{m}, \quad j \ge 2.$$

Thus, $X \in \mathcal{U}$. For the objective functions, we have the following:

$$f(X) = \sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij} C_{ij} = \sum_{i=1}^{m} \left(\sum_{t=1}^{m_1} X_{it}^1 C_{it}^1 + \sum_{t=m_1+...+m_{j-1}+1}^{m_1+...+m_j} X_{it}^1 C_{it}^1 \right) = \sum_{i=1}^{m}\sum_{t=1}^{m} X_{it}^1 C_{it}^1 = f^1(X^1).$$

On the other hand, for any $X \in \mathcal{U}$, if we let

$$X^1_{it} = \frac{X_{ij}}{m_j}, \quad 1 \le t \le m_j, j = 1;\ m_1 + ... + m_{j-1} + 1 \le t \le m_1 + ... + m_j, j \ge 2,$$

then we have

$$X^1_{it} \ge 0,$$
$$\sum_{t=1}^{m} X^1_{it} = \sum_{j=1}^{n} \frac{X_{ij}}{m_j} m_j = \sum_{j=1}^{n} X_{ij} = \frac{1}{m},$$
$$\sum_{i=1}^{n} X^1_{it} = \sum_{i=1}^{n} \frac{X_{ij}}{m_j} = \frac{1}{m_j} \sum_{i=1}^{n} X_{ij} = \frac{1}{m}.$$

Thus, $X^1 \in \mathcal{U}^1$. For the objective functions, we have the following:

$$f^1(X^1) = \sum_{i=1}^{m} \sum_{t=1}^{m} X^1_{it} C^1_{it} = \sum_{i=1}^{m} \sum_{j=1}^{n} \frac{X_{ij}}{m_j} C_{it} m_j = f(X).$$

Hence, (5) $\iff$ (7). By Birkhoff's theorem, we know (6) $\iff$ (7). Therefore, we have (5) $\iff$ (6).

Similarly, for any $X^\dagger \in \mathcal{U}^\dagger$, if we let

$$X'_{ij} = \sum_{t=1}^{m_1} X^\dagger_{it}, \quad j = 1,$$
$$X'_{ij} = \sum_{t=m_1+...+m_{j-1}+1}^{m_1+...+m_j} X^\dagger_{it}, \quad j \ge 2,$$

then $X' \in \mathcal{U}'$ and $f^\dagger(X^\dagger) = f'(X')$.

For any $X' \in \mathcal{U}'$, if we let

$$X^\dagger_{it} = \frac{X'_{ij}}{m_j}, \quad 1 \le t \le m_j, j = 1;\ m_1 + ... + m_{j-1} + 1 \le t \le m_1 + ... + m_j, j \ge 2.$$

then $X^\dagger \in \mathcal{U}^\dagger$ and $f^\dagger(X^\dagger) = f'(X')$. Therefore, (6) $\iff$ (2).

In conclusion, we have (5) $\iff$ (6) $\iff$ (2). ■

Next, we apply Proposition 3 to get our main result.

Proposition 4 *The optimization problem* (1) *is equivalent to the optimization problem* (4).

Proof Applying Proposition 3 to the optimization problem (1), the optimization problem (1) is equivalent to problem (8):

$$\min_{X' \in \Pi'} \sum_{h=1}^{n^2} \sum_{k=1}^{n} \frac{1}{n^2} C^d_{hk} X'_{hk},\ \Pi' = \left\{ X'_{hk} = \{0,1\} \middle| \sum_{k=1}^{n} X'_{hk} = 1, \sum_{h=1}^{n^2} X'_{hk} = n \right\}. \tag{8}$$

If we express the index h by two indices i, j and express t by k, l, one could check that (8) $\iff$ (4). In conclusion, (1) $\iff$ (4). ■

Appendix B. Experimental results w.r.t. running time

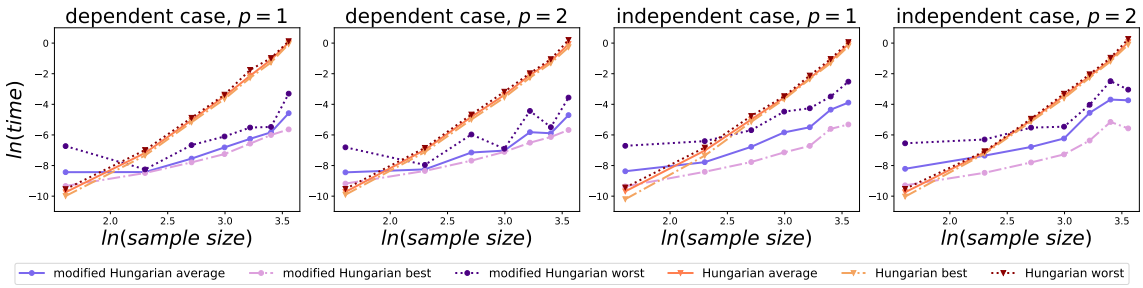


Figure 4: Results of CIFAR10 w.r.t. time

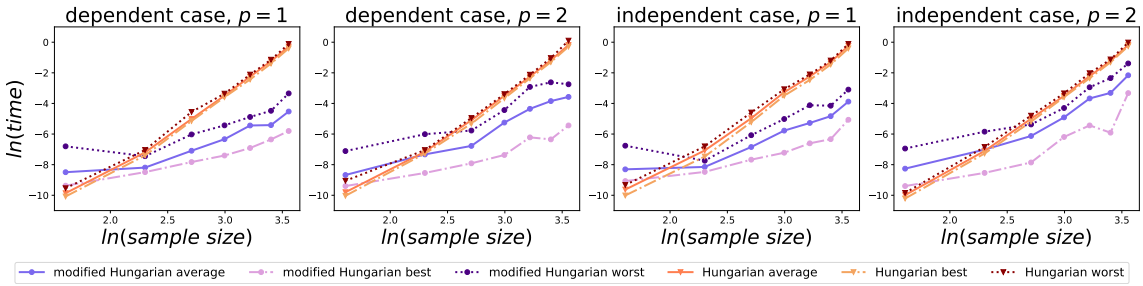


Figure 5: Results of Wisconsin cancer data w.r.t. time

We run the Hungarian algorithm code from package 'scipy' in Python and implement the modified Hungarian algorithm. We record the running time. Figure 4, 5 are experiment results w.r.t. the running time. When the sample size is small, our algorithm may be slower than the Hungarian algorithm. However, as the sample size increases, our algorithm becomes more efficient than the Hungarian algorithm. In theory, our proposed algorithm will compare more favorably if both algorithms are implemented in a low-level language, such as C++.

Appendix C. Resources

All experiments in this paper are implemented using Python 3.8 on Macbook Pro 2019 with 16 GB memory.