Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)
Special Track on AI for Good

Group Sparse Optimal Transport for Sparse Process Flexibility Design

Dixin Luo[1] , **Tingting Yu**[1] , **Hongteng Xu**[2,3*]

[1]School of Computer Science and Technology, Beijing Institue of Technology
[2]Gaoling School of Artificial Intelligence, Renmin University of China
[3]Beijing Key Laboratory of Big Data Management and Analysis Methods
{dixin.luo, 1120192166}@bit.edu.cn, hongtengxu@ruc.edu.cn

Abstract

As a fundamental problem in Operations Research, sparse process flexibility design (SPFD) aims to design a manufacturing network across industries that achieves a trade-off between the efficiency and robustness of supply chains. In this study, we propose a novel solution to this problem with the help of computational optimal transport techniques. Given a set of supply-demand pairs, we formulate the SPFD task approximately as a group sparse optimal transport (GSOT) problem, in which a group of couplings between the supplies and demands is optimized with a group sparse regularizer. We solve this optimization problem via an algorithmic framework of alternating direction method of multipliers (ADMM), in which the target network topology is updated by soft-thresholding shrinkage, and the couplings of the OT problems are updated via a smooth OT algorithm in parallel. This optimization algorithm has guaranteed convergence and provides a generalized framework for the SPFD task, which is applicable regardless of whether the supplies and demands are balanced. Experiments show that our GSOT-based method can outperform representative heuristic methods in various SPFD tasks. Additionally, when implementing the GSOT method, the proposed ADMM-based optimization algorithm is comparable or superior to the commercial software Gurobi. The code is available at https://github.com/Dixin-s-Lab/GSOT.

1 Introduction

Sparse process flexibility design (SPFD) is a fundamental Operations Research (OR) problem that aims to design a sparse bipartite manufacturing network to make plants shift production portfolios according to the dynamics of demands with little penalty in terms of time and cost. This problem originates in automotive manufacturing [Jordan and Graves, 1995] and commonly appears in various industrial scenarios, e.g., supply chain design [Graves and Tomlin, 2003; Simchi-Levi *et al.*, 2018], medical resource management [Chan *et al.*, 2022], express delivery system optimization [Chou *et al.*, 2008], and so on. Because of its significance in industry, the study of the SPFD problem has both theoretical values and social impacts, which is highly correlated with the 8th and 9th United Nations Sustainable Development Goals, i.e., promoting sustained, inclusive, and sustainable economic growth, building resilient infrastructure, promoting inclusive and sustainable industrialization, and fostering innovation. Especially for those developing countries, the flexibility of their supply chains is critical for making their economy robust to the uncertainty in the global market. Solving the SPFD problem helps optimize their existing supply chains and provides valuable guidance for establishing their future industrial infrastructures.

In general, the SPFD problem corresponds to a multi-stage optimization problem. In particular, we need to estimate the significance of edges in the current manufacturing network and update the network topology via adding or deleting edges based on the edge significance [Chou *et al.*, 2008]. Repeating the two steps above leads to a heuristic search process. Most existing methods implement the above heuristic strategy in different ways. When estimating the edge significance, they often solve the maximum flow (max-flow) problem [Harris and Ross, 1956] or learn a neural network-based surrogate of the max-flow solver in the reinforcement learning framework [Wei *et al.*, 2021; Chan *et al.*, 2022]. Given the edge significance, the network topology is updated in a deterministic [Chou *et al.*, 2010] or stochastic [Chen *et al.*, 2015] way. These methods have been applied in many scenarios, whose performance even has theoretical guarantees in some settings. However, the methods and their theory are often developed under questionable assumptions, e.g., the same cost/profit per edge, same-sized supply and demand sets, specified topology type (like long chain or k-chain), and so on, which limits their practical applications. Moreover, none of the methods consider jointly optimizing the edge significance and the network topology in a scalable framework. As a result, they often lead to local optimum for large-scale applications [Feng *et al.*, 2017].

To overcome the drawbacks mentioned above, we propose a novel approximate solution, called *group sparse optimal transport (GSOT)*, to the SPFD problem based on computational optimal transport techniques [Peyré *et al.*, 2019]. As illustrated in Figure 1, given predefined supplies and a

*Correspondence author.

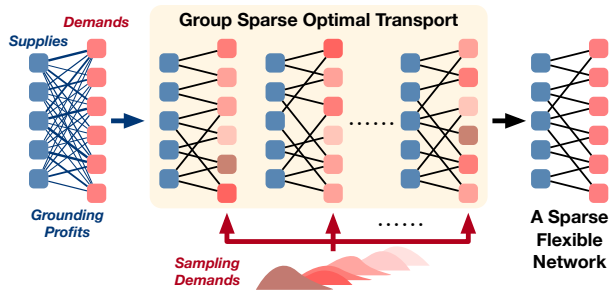


Figure 1: An illustration of the proposed method.

set of demands sampled from some distributions (as existing work [Feng *et al.*, 2017; Wei *et al.*, 2021; Chan *et al.*, 2022] did), we jointly approximate the SPFD problem by solving a set of optimal transport (OT) problems. Each OT problem corresponds to maximizing the expected profit of a manufacturing network given a supply-demand pair. All OT problems are optimized jointly with a group sparse regularizer [Bengio *et al.*, 2009; Deng *et al.*, 2013], leading to a set of couplings (or called optimal transport plans) with a consistent sparse structure. The aggregation of the group sparse couplings indicates the flexible network topology that is robust to the demand uncertainty.

Instead of applying heuristic multi-stage optimization strategies, our GSOT approximates the SPFD problem by solving a convex optimization problem, whose convergence to the global optimum is theoretically-guaranteed. In particular, we design an algorithmic framework based on the alternating direction method of multipliers (ADMM) [Boyd *et al.*, 2011], solving the optimization problem efficiently. The ADMM framework updates the target network topology iteratively, in which a group-lasso operation [Simon *et al.*, 2013] ensures the group sparsity of the couplings. Each coupling is updated by solving a quadratically-regularized optimal transport problem, leading to the smooth OT algorithm [Blondel *et al.*, 2018]. Our GSOT method is applicable for both balanced and unbalanced supply-demand settings, and the ADMM framework is feasible for large-scale SPFD problems. The comparisons with existing heuristic methods on synthetic and real-world datasets demonstrate the effectiveness of our method.

2 Related Work

2.1 Sparse Process Flexibility Design

Focusing on the SPFD problem, many solutions have been proposed. The early work in [Jordan and Graves, 1995] proposed a concept called "long chain", which designed a flexible process empirically by constructing a long circular chain visiting as many nodes as possible. Taking the long chain as the design principle, many heuristic methods have been proposed [Graves and Tomlin, 2003]. These methods design a sparse manufacturing network by adding edges to a null graph [Chou *et al.*, 2011; Feng *et al.*, 2017] or deleting edges from a complete bipartite graph [Yan *et al.*, 2018; Simchi-Levi *et al.*, 2018] step by step.

In each step, different methods select one or multiple candidate edges to add or delete by various criteria, which can be coarsely categorized into three classes. The optimization-based methods select the edges by solving a max-flow problem [Harris and Ross, 1956] or its variants (e.g., its dual problem [Yan *et al.*, 2018] and its stochastic version [Feng *et al.*, 2017]) given the current network topology. The node expansion methods adapt the concept of graph expanders in graph theory, adding edges iteratively to improve upon the node expansion ratio in a greedy manner [Chou *et al.*, 2008]. Theoretically, the "probabilistic graph expander" introduced in [Chen *et al.*, 2015] provides an upper bound for the number of edges that guarantees $(1-\epsilon)$-optimality with a high probability in balanced and symmetric systems. The work in [Chen *et al.*, 2019] further generalizes the theoretical result to unbalanced and asymmetric systems with $O(n\ln(1/\epsilon))$ edges. Recently, some learning-based SPFD methods have been developed to solve combinatorial problems [Khalil *et al.*, 2017; Sultana *et al.*, 2022; Barrett *et al.*, 2020]. These methods formulate the steps of adding edges and their influences on the cost of the whole supply chain as an action-state sequence in the reinforcement learning (RL) framework. Instead of solving the max-flow problem per step, they train a policy model to imitate the optimizer [Wei *et al.*, 2021; Chan *et al.*, 2022]. The RL-based methods are more efficient than the optimization-based methods in the deploying phase because of avoiding iterative optimization. However, they often have poor scalability and limited generalization power.

Although the above methods achieve encouraging performance in some cases, their nature of heuristic search often leads to undesired local optimum in practical applications. In theory, the optimality of the long chain is only held in some ideal situations, as discussed in [Désir *et al.*, 2016]. As aforementioned, most existing methods ignore the grounding cost/profit associated with each edge. Moreover, none consider reformulating the two-stage strategy (i.e., the network update and evaluation) in a joint optimization framework.

2.2 Computational Optimal Transport

Optimal transport (OT) theory [Villani, 2008] has proven to be useful in machine learning tasks, e.g., distribution matching [Frogner *et al.*, 2015], graph matching [Maretic *et al.*, 2022], data clustering [Agueh and Carlier, 2011; Cuturi and Doucet, 2014], information fusion [Xu and Cheng, 2022], and generative modeling [Arjovsky *et al.*, 2017]. Given the samples of two distributions, the discrete OT problem corresponds to a linear programming problem [Kusner *et al.*, 2015], which optimizes a joint distribution of the samples (a.k.a., the coupling or the optimal transport plan) to minimize the expected pairwise distance between the two distributions' samples. Typically, this problem is solved by the conditional gradient algorithm [Titouan *et al.*, 2019] or the Bregman ADMM algorithm [Wang and Banerjee, 2014; Ye *et al.*, 2017]. To accelerate the optimization step, the work in [Cuturi, 2013] proposed a surrogate called Sinkhorn distance, which makes the OT problem strictly convex by adding an entropic regularizer to the coupling. The entropic OT problem can be solved efficiently by the Sinkhorn-scaling algorithm [Sinkhorn and Knopp, 1967; Benamou *et al.*,

2015]. To further improve the numerical stability and scalability of the Sinkhorn-scaling algorithm, some variants have been proposed, including the proximal point method [Xie *et al.*, 2020] and the stochastic Sinkhorn algorithm (called Greenkhorn) [Altschuler *et al.*, 2017]. Besides the entropic OT problem, the quadratically-regularized OT problem is also considered in many applications, which can be solved by the smoothed semi-dual algorithm [Blondel *et al.*, 2018] or the Bregman ADMM algorithm [Xu and Cheng, 2022].

Note that although the above methods are designed for the balanced OT problem (i.e., the marginal distributions of the coupling are known and with equal measures), they can be extended to the cases where the marginal distributions of the coupling are unreliable or unavailable, e.g., the unbalanced OT problem [Chizat *et al.*, 2018] and the partial OT problem [Benamou *et al.*, 2015]. The optimal coupling indicates the correspondence between the two distributions' samples, which has the potential to various resource assignment problems [Zhang and Zhu, 2019; Hughes and Chen, 2021]. To our surprise, however, applying the optimal transport methods is seldom considered for the SPFD problem. In the following content, we fill this blank and propose a group sparse optimal transport method to solve the SPFD problem approximately.

3 Proposed Method

3.1 An Optimal Transport Perspective on SPFD

Given M plants and N products, we would like to design a flexible manufacturing network to match the plants' supplies with the products' random demands in a robust and efficient way. Following existing SPFD work [Chou *et al.*, 2008], we assume that the supplies are predefined and formulated as a vector $\boldsymbol{\mu} \in [0,\infty)^M$, and the demands are unknown but can be sampled from a known distribution, i.e., $\boldsymbol{\nu} \in [0,\infty)^N \sim \nu_{\mathcal{D}}$, where $\nu_{\mathcal{D}}$ is the distribution of the demands defined in the sample space $\mathcal{D}$. As illustrated in Figure 1, the proposed SPFD problem corresponds to constructing a bipartite graph connecting the plants with the products. Mathematically, this problem can be formulated as the following bi-level optimization problem:

$$\begin{aligned} \boldsymbol{A}^* = \arg \underbrace{\max\nolimits_{\boldsymbol{A}\in\{0,1\}^{M\times N}} \mathbb{E}_{\boldsymbol{\nu}\sim\nu_{\mathcal{D}}}[p_{\boldsymbol{\nu}}(\boldsymbol{A})]}_{\text{Expected profit maximization}} \\ s.t.\ p_{\boldsymbol{\nu}}(\boldsymbol{A}) = \underbrace{\max\nolimits_{\boldsymbol{A}\odot\boldsymbol{X}\in\Pi(\boldsymbol{\mu},\boldsymbol{\nu})} \langle \boldsymbol{P}, \boldsymbol{A}\odot\boldsymbol{X}\rangle}_{\text{Max-flow problem}}, \\ \forall \boldsymbol{\nu}\sim\nu_{\mathcal{D}},\ \text{and}\ \underbrace{\|\boldsymbol{A}\|_0 \le L}_{\text{Sparse constraint}}\ . \end{aligned} \tag{1}$$

Here, $\boldsymbol{A} = [a_{mn}] \in \{0,1\}^{M\times N}$ is a binary matrix indicating the network topology. $a_{mn} = 1$ means that there exists an edge (m,n) between the plant m and the product n. $p_{\boldsymbol{\nu}}(\boldsymbol{A})$ represents the maximum profit obtained when the network topology is $\boldsymbol{A}$ and the demand vector is $\boldsymbol{\nu}$. The upper-level optimization problem corresponds to optimizing the network topology to maximize the expected profit. In the lower level, a set of max-flow problems are solved to determine the profits given the current network topology. To avoid a trivial solution (i.e., $\boldsymbol{A} = \boldsymbol{1}_{M\times N}$), we further consider the sparsity of $\boldsymbol{A}$, restricting its number of nonzero elements to be equal to or less than a threshold L, i.e., $\|\boldsymbol{A}\|_0 \le L$, where $\|\cdot\|_0$ is the ℓ_0-norm of the matrix.

Following the work in [Chan *et al.*, 2022], we decouple the variables of each max-flow problem as the network topology $\boldsymbol{A}$ and the transport matrix $\boldsymbol{X} = [x_{mn}] \in \mathbb{R}^{M\times N}$. Accordingly, $\boldsymbol{A}\odot\boldsymbol{X}$ represents the Hadamard product of the two matrices. Its (nonzero) element $a_{mn}x_{mn}$ indicates the number of products passing through the edge (m,n). Obviously, $\boldsymbol{A}\odot\boldsymbol{X}$ should meet the constraints provided by the supplies and the demands, i.e., $\boldsymbol{A}\odot\boldsymbol{X} \in \Pi(\boldsymbol{\mu},\boldsymbol{\nu}) = \{\boldsymbol{T} \ge \boldsymbol{0} | \boldsymbol{T}\boldsymbol{1}_N \le \boldsymbol{\mu}, \boldsymbol{T}^T\boldsymbol{1}_M \le \boldsymbol{\nu}\}$. The objective $\langle \boldsymbol{P}, \boldsymbol{A}\odot\boldsymbol{X}\rangle = \sum_{m,n} p_{mn}a_{mn}x_{mn}$ is the whole profit created by the network, and $\boldsymbol{P} = [p_{mn}]$ is the grounding profit matrix recording the profit per product created by a plant (i.e., p_{mn} represents the profit of making the product n from the plant m).

In this study, we consider simplifying the SPFD problem in (1) based on computational optimal transport techniques. In particular, we replace $\boldsymbol{A}\odot\boldsymbol{X}$ with a single functional variable $\boldsymbol{T}(\cdot)$. Accordingly, we can reformulate (1) as follows.

$$\begin{aligned} &\max\nolimits_{\boldsymbol{T}(\cdot)} \mathbb{E}_{\boldsymbol{\nu}\in\nu_{\mathcal{D}}}[\underbrace{\max\nolimits_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu},\boldsymbol{\nu})} \langle \boldsymbol{P}, \boldsymbol{T}\rangle}_{\boldsymbol{T}(\boldsymbol{\nu})}] \\ &s.t.\ \mathbb{E}_{\boldsymbol{\nu}\in\mathcal{D}}[\boldsymbol{T}(\boldsymbol{\nu})]\|_0 \le L. \end{aligned} \tag{2}$$

Here, the functional variable $\boldsymbol{T}(\cdot) : \mathcal{D} \mapsto \mathbb{R}^{M\times N}$ takes a demand vector as its input and outputs a transport matrix. Accordingly, $\boldsymbol{T}(\boldsymbol{\nu}) := \arg\max_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu},\boldsymbol{\nu})} \langle \boldsymbol{P}, \boldsymbol{T}\rangle$ corresponds to the output determined by the input $\boldsymbol{\nu}$, which is a typical optimal transport problem.[1] The expectation of the output, i.e., $\mathbb{E}_{\boldsymbol{\nu}\in\mathcal{D}}[\boldsymbol{T}(\boldsymbol{\nu})]$, indicates the network topology, whose element represents the expected amount of products per edge. It works as a surrogate of the $\boldsymbol{A}$ in (1), and we impose the sparse constraint on it, as shown in (2).

Introducing the functional variable, we reformulate the bi-level optimization problem (1) to a nested functional optimization problem (2). The inner problem optimizes a slice $\boldsymbol{T}(\boldsymbol{\nu})$ given a specific $\boldsymbol{\nu}$, and the outer problem optimizes the whole function $\boldsymbol{T}(\cdot)$ with a sparse constraint. Suppose that we have a set of demands sampled from $\nu_{\mathcal{D}}$, denoted as $\{\boldsymbol{\nu}_k\}_{k=1}^K$, as the work in [Wei *et al.*, 2021; Chan *et al.*, 2022] did. We can approximate the functional variable with the help of the kernel trick [Schölkopf *et al.*, 2001], i.e., $\boldsymbol{T}(\boldsymbol{\nu}) = \sum_{k=1}^K \boldsymbol{T}_k \kappa(\boldsymbol{\nu}, \boldsymbol{\nu}_k)$, where $\boldsymbol{T}_k := \boldsymbol{T}(\boldsymbol{\nu}_k)$ and κ can be a predefined kernel function. Then, we can rewrite (2) as

$$\begin{aligned} &\max\nolimits_{\{\boldsymbol{T}_k\in\Pi(\boldsymbol{\mu},\boldsymbol{\nu}_k)\}_{k=1}^K} \frac{1}{K}\sum\nolimits_{k=1}^K \langle \boldsymbol{P}, \boldsymbol{T}_k\rangle, \\ &s.t.\ \Big\| \sum\nolimits_{k=1}^K \boldsymbol{T}_k \Big\|_0 \le L, \end{aligned} \tag{3}$$

which corresponds to solving a group of K optimal transport problems with a consistency regularizer on the sparsity of the couplings. Denote the coupling group as a tensor $\mathcal{T} = [\boldsymbol{T}_k] \in \mathbb{R}^{M\times N\times K}$. Following the work intensor [Bengio *et al.*, 2009;

[1] When $\Pi(\boldsymbol{\mu},\boldsymbol{\nu}) = \{\boldsymbol{T} \ge \boldsymbol{0} | \boldsymbol{T}\boldsymbol{1}_N = \boldsymbol{\mu}, \boldsymbol{T}^T\boldsymbol{1}_M = \boldsymbol{\nu}\}$, the problem is a balanced OT problem [Villani, 2008]. When $\Pi(\boldsymbol{\mu},\boldsymbol{\nu}) = \{\boldsymbol{T} \ge \boldsymbol{0} | \boldsymbol{T}\boldsymbol{1}_N \le \boldsymbol{\mu}, \boldsymbol{T}^T\boldsymbol{1}_M \le \boldsymbol{\nu}\}$, the problem is an unbalanced/partial OT problem [Benamou *et al.*, 2015].

Deng *et al.*, 2013], we relax the strict sparse constraint in (3) to a group sparse regularizer and derive the final group sparse optimal transport (GSOT) problem as follows.

$$\min_{\{\boldsymbol{T}_k\in\Pi(\boldsymbol{\mu},\boldsymbol{\nu}_k)\}_{k=1}^K} \frac{1}{K}\sum_{k=1}^K \langle -\boldsymbol{P},\boldsymbol{T}_k\rangle + \alpha\|\mathcal{T}\|_{1,2}, \quad (4)$$

where $\|\mathcal{T}\|_{1,2}=\sum_{m=1}^M\sum_{n=1}^N\sqrt{\sum_{k=1}^K t_{mnk}^2}$ represents the group sparse regularizer. This regularizer encourages the couplings to share the same sparse structure, whose significance is controlled by the hyperparameter $\alpha\geq 0$.

We achieve sparse process flexibility design approximately by solving the GSOT problem in (4). In particular, after obtaining the optimal couplings $\{\boldsymbol{T}_k^*\}_{k=1}^K$, we can determine the network topology as $I(\sum_{k=1}^K \boldsymbol{T}_k^*)$, where $I(\cdot)$ is an indicator returning 1 when the input is nonzero and returning 0 otherwise. Compared to the original SPFD problem in (1), the GSOT problem is convex, which can be solved efficiently and have a theoretical guarantee of convergence. In the next subsection, we develop an efficient ADMM algorithm to solve this problem iteratively.

3.2 An ADMM-Based Optimization Algorithm

For the GSOT problem in (4), we introduce an auxiliary variable $\mathcal{Z}=[\boldsymbol{Z}_k]\in\mathbb{R}^{M\times N\times K}$ and a dual variable $\mathcal{U}=[\boldsymbol{U}_k]\in\mathbb{R}^{M\times N\times K}$, respectively, and reformulate it in the following augmented Lagrangian form:

$$\begin{aligned}&\min_{\{\boldsymbol{T}_k\in\Pi(\boldsymbol{\mu},\boldsymbol{\nu}_k),\boldsymbol{Z}_k,\boldsymbol{U}_k\}_{k=1}^K} \frac{1}{K}\sum_{k=1}^K\langle -\boldsymbol{P},\boldsymbol{T}_k\rangle+\\ &\alpha\|\mathcal{Z}\|_{1,2}+\rho\sum_{k=1}^K\langle\boldsymbol{U}_k,\boldsymbol{T}_k-\boldsymbol{Z}_k\rangle+\frac{\rho}{2}\|\mathcal{T}-\mathcal{Z}\|_F^2,\end{aligned} \quad (5)$$

where $\rho>0$ and $\|\cdot\|_F$ represents the Frobenius norm of matrix. This problem can be solved in an ADMM framework. In particular, we update $\mathcal{T}$, $\mathcal{Z}$, and $\mathcal{U}$ iteratively. In the l-th iteration, we solve the following three subproblems.

1) Update $\mathcal{T}$: Fixing current $\mathcal{Z}^{(l)}$ and $\mathcal{U}^{(l)}$, we achieve this step via solving K independent optimal transport problems with ℓ_2 regularizers: for $k=1,...,K$, we have

$$\begin{aligned}\boldsymbol{T}_k^{(l+1)}=&\arg\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu},\boldsymbol{\nu}_k)}\langle -\boldsymbol{P},\boldsymbol{T}\rangle\\ &+\frac{\rho}{2}\|\boldsymbol{T}-\boldsymbol{Z}_k^{(l)}+\boldsymbol{U}_k^{(l)}\|_F^2.\end{aligned} \quad (6)$$

Ignoring the term irrelevant to $\boldsymbol{T}$, we can equivalently formulate (6) as a quadratically-regularized optimal transport (QROT) problem, i.e.,

$$\min_{\boldsymbol{T}\in\Pi(\boldsymbol{\mu},\boldsymbol{\nu}_k)}\langle\boldsymbol{C},\boldsymbol{T}\rangle+\frac{\rho}{2}\langle\boldsymbol{T},\boldsymbol{T}\rangle, \quad (7)$$

where $\boldsymbol{C}=-\boldsymbol{P}+\boldsymbol{U}_k^{(l)}-\boldsymbol{Z}_k^{(l)}$. We solve the problem based on the smooth OT algorithm proposed in [Blondel *et al.*, 2018]. Here, we consider the following two situations.

- **Balanced supply-demand cases:** When $\|\boldsymbol{\mu}\|_1=\|\boldsymbol{\nu}_k\|_1$, we impose the doubly-stochastic constraint on the coupling $\boldsymbol{T}$, i.e., the feasible domain is $\Pi(\boldsymbol{\mu},\boldsymbol{\nu}_k)=\{\boldsymbol{T}\geq\boldsymbol{0}|\boldsymbol{T}\boldsymbol{1}_N=\boldsymbol{\mu},\boldsymbol{T}^T\boldsymbol{1}_M=\boldsymbol{\nu}_k\}$. In this situation, we can rewrite the QROT problem in (7) in its smooth relaxed dual formulation:

$$\begin{aligned}&\max_{\boldsymbol{a}\in\mathbb{R}^M,\boldsymbol{b}\in\mathbb{R}^N}\boldsymbol{\mu}^T\boldsymbol{a}+\boldsymbol{\nu}_k^T\boldsymbol{b}\\ &-\frac{1}{2\rho}\|[\boldsymbol{a}\boldsymbol{1}_N^T+\boldsymbol{1}_M\boldsymbol{b}^T-\boldsymbol{C}]_+\|_F^2,\end{aligned} \quad (8)$$

where $[\cdot]_+$ means setting negative elements to be zeros, and $\boldsymbol{a}$ and $\boldsymbol{b}$ are dual variables. This dual problem is unconstrained and can be solved efficiently by the L-BFGS algorithm [Liu and Nocedal, 1989]. Given the optimal solution of (8), denoted as $\boldsymbol{a}^*$ and $\boldsymbol{b}^*$, we can derive the optimal coupling $\boldsymbol{T}_k^{(l+1)}=\frac{1}{\rho}[\boldsymbol{a}^*\boldsymbol{1}_N^T+\boldsymbol{1}_M(\boldsymbol{b}^*)^T-\boldsymbol{C}]_+$.

- **Unbalanced supply-demand cases:** When $\|\boldsymbol{\mu}\|_1\neq\|\boldsymbol{\nu}_k\|_1$, the feasible domain of the coupling becomes $\Pi(\boldsymbol{\mu},\boldsymbol{\nu}_k)=\{\boldsymbol{T}\geq\boldsymbol{0}|\boldsymbol{T}\boldsymbol{1}_N\leq\boldsymbol{\mu},\boldsymbol{T}^T\boldsymbol{1}_M\leq\boldsymbol{\nu}_k$, and $\boldsymbol{1}_M^T\boldsymbol{T}\boldsymbol{1}_N=\min\{\|\boldsymbol{\mu}\|_1,\|\boldsymbol{\nu}_k\|_1\}\}$. In this situation, we relax the problem in (7) to a semi-relaxed smooth form: without the loss of generality, we assume that $\|\boldsymbol{\mu}\|_1<\|\boldsymbol{\nu}_k\|_1$, and the problem becomes

$$\begin{aligned}&\min_{\boldsymbol{T}\geq\boldsymbol{0},\boldsymbol{T}\boldsymbol{1}_N=\boldsymbol{\mu}}\langle\boldsymbol{C},\boldsymbol{T}\rangle+\frac{\rho}{2}\langle\boldsymbol{T},\boldsymbol{T}\rangle\\ &+\frac{1}{2\rho}\|\boldsymbol{T}^T\boldsymbol{1}_M-\boldsymbol{\nu}_k\|_2^2.\end{aligned} \quad (9)$$

This problem can be solved by the FISTA algorithm [Beck and Teboulle, 2009] or conditional gradient.

2) Update $\mathcal{Z}$: Given fixed $\mathcal{T}^{(l+1)}$ and $\mathcal{U}^{(l)}$, we achieve this step via solving a group sparse problem.

$$\mathcal{Z}^{(l+1)}=\arg\min_{\mathcal{Z}}\|\mathcal{Z}\|_{1,2}+\frac{\rho}{2\alpha}\|\mathcal{T}^{(l+1)}-\mathcal{Z}+\mathcal{U}^{(l)}\|_F^2. \quad (10)$$

Applying the soft-thresholding method [Bengio *et al.*, 2009; Deng *et al.*, 2013], the solution of this problem corresponds to $M\times N$ independent updates:

$$\begin{aligned}&\boldsymbol{r}_{mn}=\boldsymbol{t}_{mn}^{(l+1)}+\boldsymbol{u}_{mn}^{(l)},\ \forall\, m=1,...,M,\ n=1,...,N\\ &\boldsymbol{z}_{mn}^{(l+1)}=\max\Big\{1-\frac{\alpha}{\rho\|\boldsymbol{r}_{mn}\|_2},0\Big\}\boldsymbol{r}_{mn},\end{aligned} \quad (11)$$

where $\boldsymbol{t}_{mn}=[t_{mnk}]\in\mathbb{R}^K$, $\boldsymbol{u}_{mn}=[u_{mnk}]\in\mathbb{R}^K$ and $\boldsymbol{z}_{mn}=[z_{mnk}]\in\mathbb{R}^K$ are vectors in $\mathcal{T},\mathcal{U}$ and $\mathcal{Z}$.

3) Update $\mathcal{U}$: In the framework of ADMM, the update of $\mathcal{U}$ is simple:

$$\mathcal{U}^{(l+1)}=\mathcal{U}^{(l)}+\mathcal{T}^{(l+1)}-\mathcal{Z}^{(l+1)}. \quad (12)$$

Convergence and efficiency. As aforementioned, the GSOT problem is a convex optimization problem. Applying our ADMM framework, each subproblem is convex as well. Therefore, the convergence of our optimization algorithm is guaranteed in theory. More specifically, the convergence rate is $\mathcal{O}(\epsilon^{-1})$ in general because we apply first-order optimization algorithms to solve the subproblems [Nishihara *et al.*, 2015].[2] Regarding time complexity, the main bottleneck is solving K QROT problems that involve iterative updates. Fortunately, these problems are independent. We can solve them in parallel and accelerate the whole process.

[2]The L-BFGS is a quasi-Newton method, which applies a first-order strategy to approximate the second-order strategy.

Algorithm 1 Oue GSOT method for the SPFD problem

Input: A supply vector $\boldsymbol{\mu}$ and a set of demand vectors $\{\boldsymbol{\nu}_k\}_{k=1}^K$. The profit matrix $\boldsymbol{P}$. The hyperparameters α and ρ. The maximum number of edges L.
Output: Top-L edges $\{(m_\ell, n_\ell)\}_{\ell=1}^L$.
Initialize $\mathcal{T} = \{\boldsymbol{T}_k = \boldsymbol{\mu}\boldsymbol{\nu}_k^T\}$, $\mathcal{Z} = \mathcal{T}, \mathcal{U} = \mathbf{0}_{M\times N\times K}$.
while Not converge **do**
  for $k = 1 : K$ **do**
    $\boldsymbol{C} = -\boldsymbol{P} + \boldsymbol{U}_k - \boldsymbol{Z}_k$.
    if $\|\boldsymbol{\mu}\|_1 = \|\boldsymbol{\nu}_k\|_1$ (Balanced case) **then**
      Solve (8) via L-BFGS and get $\boldsymbol{a}^*$ and $\boldsymbol{b}^*$.
      Update $\boldsymbol{T}_k$ as $\frac{1}{\rho}[\boldsymbol{a}^*\mathbf{1}_N^T + \mathbf{1}_M(\boldsymbol{b}^*)^T - \boldsymbol{C}]_+$.
    else
      Solve (9) via FISTA or conditional gradient and get $\boldsymbol{T}_k$ accordingly.
    end if
  end for
  Update $\mathcal{Z}$ via (11).
  Update $\mathcal{U}$ via (12).
end while
$\mathcal{T}^* \leftarrow \mathcal{T}$, and derive $\{(m_\ell, n_\ell)\}_{\ell=1}^L$ via (13).

3.3 Sparse Network Topology Design

Repeating the above optimization steps till the objective function converges, we obtain the optimal variable, denoted as $\mathcal{T}^* = [\boldsymbol{T}_k^*]$. As aforementioned, the aggregation of the $\boldsymbol{T}_k^*$'s (i.e., $\boldsymbol{T}^* := \frac{1}{K}\sum_{k=1}^K \boldsymbol{T}_k^*$) is sparse and indicates the proposed network topology.

Besides automatically designing a sparse manufacturing network, our GSOT method can also design a network with a predefined number of edges (e.g., the L in (1)). In particular, from a statistical viewpoint, each element of $\boldsymbol{T}^*$, i.e., t_{mn}^*, approximates the expected amount of product n made by plant m. Accordingly, the expected profit created by the network is approximated as $\boldsymbol{P} \odot \boldsymbol{T}^*$, where each element $p_{mn}t_{mn}^*$ corresponds to the expected profit generated by the edge (m, n). We can take $\boldsymbol{P} \odot \boldsymbol{T}^*$ as the significance of the edges. As a result, we can sort the edges according to their significance and select the top-L significant edges, i.e.,

$$\{(m_\ell, n_\ell)\}_{\ell=1}^L = \text{Top}_L(\boldsymbol{P} \odot \boldsymbol{T}^*). \tag{13}$$

The scheme of our GSOT method is shown in Algorithm 1.

4 Experiments

4.1 Implementation Details

Methods. To demonstrate the feasibility and effectiveness of our **GSOT** method, we compare it with representative SPFD methods on both synthetic and real-world datasets. According to the methodological categories proposed by [Chou *et al.*, 2008], we consider the following three methods as our baselines, which represent three different strategies.

- **Sampling method** in [Lien *et al.*, 2011]: This method samples edges and updates the significance of the remaining edges in an iterative and alternating manner. In each step, the sampling probability of each edge is estimated based on the solution of a max-flow problem.
- **Expander method** in [Chou *et al.*, 2011]: This method applies the theory of graph expander, designing a sparse process structure by constructing a graph with a good node expansion ratio. It adds edges iteratively to improve the node expansion ratio in a greedy manner.
- **Chaining method** in [Jordan and Graves, 1995]: When the number of plants is equal to that of products and the supplies and demands are balanced, we further consider the classic long chain as a baseline.

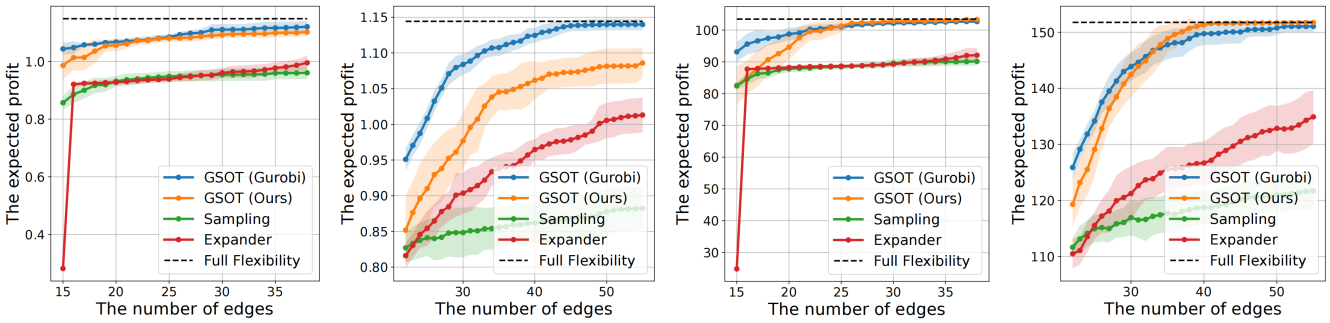
All the methods, including the proposed method and the baselines, are implemented in Python. The Python Optimal Transport (POT) toolbox [Flamary *et al.*, 2021] is used to solve the QROT problems in our method. Additionally, to verify the usefulness of our ADMM-based optimization algorithm (i.e., **GSOT (Ours)**), we apply the commercial optimization software Gurobi [Gurobi Optimization, 2021] to solve the proposed GSOT problem in (4) (i.e., **GSOT (Gurobi)**). For our GSOT method, the comparison between our ADMM-based algorithm and the Gurobi-based implementation is provided.

Datasets. We consider the **symmetry** between plants and products (i.e., whether $M = N$ or not) and the **balance** between supplies and demands (i.e., whether $\|\boldsymbol{\mu}\|_1 = \|\boldsymbol{\nu}_k\|_1$ or not), constructing four synthetic datasets accordingly. The symmetric and balanced case corresponds to an ideal scenario used in many existing work [Simchi-Levi and Wei, 2012; Chou *et al.*, 2008]: We set $M = N$; The supplies provided by the M plants are fixed and formulated as a uniform vector, while the demands of each product are sampled from independently a uniform distribution, leading to K N-dimensional vectors; we then normalize the supply vector and the demand vectors (i.e., $\|\boldsymbol{\mu}\|_1 = \|\boldsymbol{\nu}_k\|_1 = 1$). In the asymmetric cases, we set $N = \text{round}(1.5M)$. In the unbalanced cases, we use unnormalized supply and demand vectors. For each synthetic dataset, we construct a grounding profit matrix randomly. We set $M \in \{5, 10, 15, 20\}$ and $K \in \{10, 20, 50\}$, testing the scalability of different methods accordingly.

Besides the synthetic datasets, we consider the real-world fashion manufacturing dataset [Chou *et al.*, 2014], which contains ten plants and ten products in the fashion industry. There exists a profit matrix with size 10×10, whose element indicates the profit per product generated by a plant.[3] The supplies and the mean and variance of demands are predefined, so we set one normalized supply vector and simulate 100 unnormalized demand vectors. Fifty demand vectors are used for training, while the remaining vectors are used for testing.

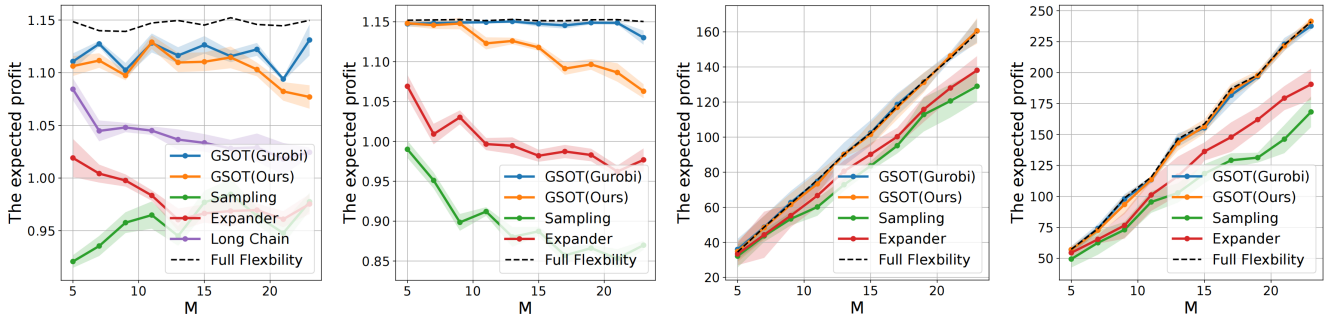
Evaluation criteria. For each dataset, we split the demand vectors into two sets. A training set is used to design the manufacturing network, and a testing set is used to simulate and evaluate the performance of the designed network. Given a dataset, we evaluate the performance of various methods in two ways: i) setting the number of edges from $\max\{M, N\}$ to $\text{round}(2.5\max\{M, N\})$ and comparing the profits achieved by the networks designed by different methods; ii) designing a network with $2\max\{M, N\}$ edges by

[3] In [Chou *et al.*, 2014], all the plants have the same profit per product, which is impractical in general. We add random noise to the rank-1 profit matrix and get a full-rank profit matrix.



(a) Balanced symmetric case (b) Balanced asymmetric case (c) Unbalanced symmetric case (d) Unbalanced asymmetric case

Figure 2: The comparisons for various methods on their expected profits achieved under different network topologies. In this experiment, we set $M = 15$ and $K = 20$. The fully-flexible scenario corresponds to a complete bipartite manufacturing network.



(a) Balanced symmetric case (b) Balanced asymmetric case (c) Unbalanced symmetric case (d) Unbalanced asymmetric case

Figure 3: The comparisons for various methods on their scalability. In each figure, we change the number of plants (M) and design a manufacturing network with $2\max\{M, N\}$ edges.

different methods and evaluating the performance with the increase of N (or M). In each experiment, we test each method in 10 trials and record the average and standard deviation of its performance. For our GSOT method and its ADMM-based algorithm, we i) analyze the robustness of the hyperparameters α and ρ, ii) analyze its empirical sample complexity, and iii) compare it with the Gurobi-based implementation.

4.2 Evaluation on Synthetic Data

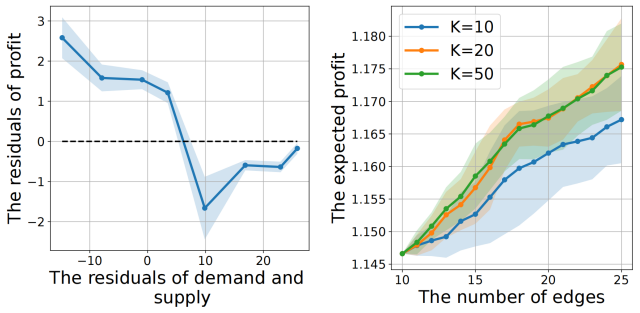
Figure 2 shows the performance of different methods on four synthetic datasets. We can find that with the increase of edge numbers, the networks designed by different methods can always increase expected profits. However, our GSOT method consistently works better than the baselines in all four datasets. In particular, when designing a network with L edges, our GSOT method obtains higher expected profits than the baselines, which quickly approaches the complete-bipartite network's performance (with full flexibility).

Figure 3 further shows the comparisons for various methods on their scalability. In this experiment, we increase M from 5 to 20 and design a network with $2\max\{M, N\}$ edges in each scenario. Similar to Figure 2, we can find that our GSOT method consistently outperforms the baselines. Note that our GSOT method works better than the long chain in the balanced symmetric case. A potential reason for this phenomenon is that we introduce a random price matrix to assign the edges with different significance. This leads to a complicated scenario ignored by theoretical analysis [Désir *et al.*, 2016] — even if the supplies and demands are balanced and the plants and the products have the same amount, the long chain may not be optimal when the edges are weighted.

4.3 Feasibility of ADMM-Based Algorithm

According to the results in Figures 2 and 3, we can find that compared to the Gurobi-based implementation, our ADMM-based algorithm does not perform so well in the balanced cases. However, in the challenging unbalanced cases, it is comparable to even superior to the Gurobi-based implementation — as shown in Figures 2(c, d), our GSOT method converges to the full flexibility situation more quickly when it is implemented based on our ADMM framework.

Additionally, we find that in the unbalanced cases, the performance of Gurobi and our algorithm is different. Denote the expected profits achieved by Gurobi and our ADMM algorithm as p_{G} and p_{A}, respectively. Figure 4(a) shows that when supplies are insufficient (i.e., $\|\boldsymbol{\mu}\|_1 < \|\boldsymbol{\nu}_k\|_1$) Gurobi often works better (i.e., $p_{\mathrm{G}} \geq p_{\mathrm{A}}$). On the contrary, when supplies are sufficient (i.e., $\|\boldsymbol{\mu}\|_1 > \|\boldsymbol{\nu}_k\|_1$), our ADMM algorithm often outperforms Gurobi. This result gives us useful insight into the selection of algorithms in practice.



(a) $\|\boldsymbol{\mu}\|_1 - \|\boldsymbol{\nu}_k\|_1$ v.s. $p_{\mathrm{G}} - p_{\mathrm{A}}$ (b) The influence of K

Figure 4: (a) The influence of unbalanced supplies and demands on our algorithm. (b) The performance of our GSOT method with respect to K (in the balanced and symmetric case $M = N = 10$).

$\alpha\,(\rho = 1)$		0.01	0.1	1	10
Balanced Symmetric	$M = 5$	1.12	1.14	1.11	1.13
	$M = 10$	1.10	1.10	1.10	1.06
	$M = 15$	1.11	1.11	1.11	0.41
	$M = 20$	1.09	1.10	1.08	0.34
$\rho\,(\alpha = 1)$		0.01	0.1	1	10
Balanced Asymmetric	$M = 5$	0.97	1.13	1.13	1.12
	$M = 10$	0.78	1.13	1.13	1.11
	$M = 15$	0.68	0.47	1.06	1.10
	$M = 20$	0.63	0.56	1.09	1.07

Table 1: The influences of hyperparameters on expected profit

Remark. Gurobi is a sophisticated commercial optimization software, while our ADMM algorithm is implemented based on Python. Refactoring our code with professional acceleration and optimization techniques can further improve the performance of our algorithm, which is left as our future work. Compared with Gurobi, our work is more friendly to developing countries because we will provide open-source code and make it free for commercial applications.

Additionally, our ADMM-based algorithm is robust to the data size and hyperparameter settings. As shown in Figure 4(b), the performance of our algorithm becomes stable given $K = 20$ supply-demand pairs. In other words, applying our algorithm, we merely need to sample a small set of supply-demand pairs when designing manufacturing networks. Table 1 shows the influences of two key hyperparameters (α and ρ) on the performance of our algorithm. We can find that when $\alpha \in [0.01, 1]$ and $\rho \in [1, 10]$, our algorithm works well, whose performance is with good stability. It means that our algorithm is robust to the hyperparameters, and we can set them in a wide range. Empirically, we set $\alpha = \rho = 1$ in our experiments.

4.4 Evaluation on Real-World Data

The real-world data is generated by a fashion manufacturing company with $M = 10$ manufacturing plants. The company aims to design a flexible and sparse manufacturing network to produce $N = 10$ different styles of parkas. For the plant m, the profit of making one product n is denoted as p_{mn}, and all the profits are stored in a profit matrix. Here, the supplies of the plants are fixed, and the demands of the products obey different independent distributions. This problem corresponds to an unbalanced symmetric SPFD problem.

Method	Expected Profit		
	20 Edges	23 Edges	25 Edges
Sampling	$0.501_{\pm 0.009}$	$0.529_{\pm 0.001}$	$0.520_{\pm 0.001}$
Expander	$0.522_{\pm 0.010}$	$0.532_{\pm 0.008}$	$0.535_{\pm 0.007}$
GSOT (Ours)	$0.502_{\pm 0.008}$	$0.535_{\pm 0.011}$	$0.541_{\pm 0.005}$
GSOT (Gurobi)	$0.528_{\pm 0.004}$	$0.543_{\pm 0.000}$	$0.544_{\pm 0.002}$
Full Flexibility	0.548		

Table 2: Comparison for various methods on real-world data

Table 2 shows the expected profits obtained by different methods. Following the experiments on synthetic data, for each method, we set the number of the corresponding network in the range $[2M, \mathrm{round}(2.5M)]$, which corresponds to the networks with edges equal to and more than a long chain, respectively. The results show that when the network is as sparse as a long chain, our GSOT method is comparable to the baselines. When further increasing the number of edges, our GSOT method outperforms the baselines consistently and approaches the performance of a complete bipartite network. Our ADMM-based algorithm is comparable to Gurobi — by naïve Python implementation, its performance is just slightly worse than Gurobi, demonstrating its potential for practical scenarios. In the future, we plan to improve our algorithm by optimizing and adjusting its hyperparameter setting.

5 Conclusion

In this study, we have proposed a GSOT method for sparse process flexibility design, which provides a new solution to this significant OR problem. Instead of applying the traditional two-stage design strategy, our method constructs and evaluates the designed network in a joint optimization framework, which applies to unbalanced and asymmetric SPFD problems. We develop an effective optimization algorithm to achieve our GSOT method, leading to encouraging experimental results. In the future, we will study the theory of our method and consider improving its computational efficiency further. Moreover, we would like to find partners in the industry and test our method in real-world cases.

Acknowledgements

Dr. Dixin Luo was supported by the National Natural Science Foundation of China (No. 62102031) and the Young Scholar Program (No. XSQD-202107001) from Beijing Institute of Technology. Dr. Hongteng Xu was supported by the National Natural Science Foundation of China (No. 62106271, 92270110), CAAI-Huawei MindSpore Open Fund, the Fundamental Research Funds for the Central Universities, and the Research Funds of Renmin University of China. He thanks the support from the Beijing Key Laboratory of Big Data Management and Analysis Methods, the Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative.

References

[Agueh and Carlier, 2011] Martial Agueh and Guillaume Carlier. Barycenters in the Wasserstein space. *SIAM Journal on Mathematical Analysis*, 43(2):904–924, 2011.

[Altschuler *et al.*, 2017] Jason Altschuler, Jonathan Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In *Advances in Neural Information Processing Systems*, pages 1964–1974, 2017.

[Arjovsky *et al.*, 2017] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223, 2017.

[Barrett *et al.*, 2020] Thomas Barrett, William Clements, Jakob Foerster, and Alex Lvovsky. Exploratory combinatorial optimization with reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3243–3250, 2020.

[Beck and Teboulle, 2009] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[Benamou *et al.*, 2015] Jean-David Benamou, Guillaume Carlier, Marco Cuturi, Luca Nenna, and Gabriel Peyré. Iterative Bregman projections for regularized transportation problems. *SIAM Journal on Scientific Computing*, 37(2):A1111–A1138, 2015.

[Bengio *et al.*, 2009] Samy Bengio, Fernando Pereira, Yoram Singer, and Dennis Strelow. Group sparse coding. *Advances in neural information processing systems*, 22, 2009.

[Blondel *et al.*, 2018] Mathieu Blondel, Vivien Seguy, and Antoine Rolet. Smooth and sparse optimal transport. In *International conference on artificial intelligence and statistics*, pages 880–889. PMLR, 2018.

[Boyd *et al.*, 2011] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011.

[Chan *et al.*, 2022] Timothy CY Chan, Daniel Letourneau, and Benjamin G Potter. Sparse flexible design: a machine learning approach. *Flexible Services and Manufacturing Journal*, pages 1–51, 2022.

[Chen *et al.*, 2015] Xi Chen, Jiawei Zhang, and Yuan Zhou. Optimal sparse designs for process flexibility via probabilistic expanders. *Operations Research*, 63(5):1159–1176, 2015.

[Chen *et al.*, 2019] Xi Chen, Tengyu Ma, Jiawei Zhang, and Yuan Zhou. Optimal design of process flexibility for general production systems. *Operations Research*, 67(2):516–531, 2019.

[Chizat *et al.*, 2018] Lenaic Chizat, Gabriel Peyré, Bernhard Schmitzer, and François-Xavier Vialard. Scaling algorithms for unbalanced optimal transport problems. *Mathematics of Computation*, 87(314):2563–2609, 2018.

[Chou *et al.*, 2008] Mabel C Chou, Chung-Piaw Teo, and Huan Zheng. Process flexibility: design, evaluation, and applications. *Flexible Services and Manufacturing Journal*, 20:59–94, 2008.

[Chou *et al.*, 2010] Mabel C Chou, Geoffrey A Chua, Chung-Piaw Teo, and Huan Zheng. Design for process flexibility: Efficiency of the long chain and sparse structure. *Operations research*, 58(1):43–58, 2010.

[Chou *et al.*, 2011] Mabel C Chou, Geoffrey A Chua, Chung-Piaw Teo, and Huan Zheng. Process flexibility revisited: The graph expander and its applications. *Operations research*, 59(5):1090–1105, 2011.

[Chou *et al.*, 2014] Mabel C Chou, Geoffrey A Chua, and Huan Zheng. On the performance of sparse process structures in partial postponement production systems. *Operations research*, 62(2):348–365, 2014.

[Cuturi and Doucet, 2014] Marco Cuturi and Arnaud Doucet. Fast computation of Wasserstein barycenters. In *International Conference on Machine Learning*, pages 685–693, 2014.

[Cuturi, 2013] Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pages 2292–2300, 2013.

[Deng *et al.*, 2013] Wei Deng, Wotao Yin, and Yin Zhang. Group sparse optimization by alternating direction method. In *Wavelets and Sparsity XV*, volume 8858, pages 242–256. SPIE, 2013.

[Désir *et al.*, 2016] Antoine Désir, Vineet Goyal, Yehua Wei, and Jiawei Zhang. Sparse process flexibility designs: Is the long chain really optimal? *Operations Research*, 64(2):416–431, 2016.

[Feng *et al.*, 2017] Wancheng Feng, Chen Wang, and Zuo-Jun Max Shen. Process flexibility design in heterogeneous and unbalanced networks: A stochastic programming approach. *IISE Transactions*, 49(8):781–799, 2017.

[Flamary *et al.*, 2021] Rémi Flamary, Nicolas Courty, Alexandre Gramfort, Mokhtar Z. Alaya, Aurélie Boisbunon, Stanislas Chambon, Laetitia Chapel, Adrien Corenflos, Kilian Fatras, Nemo Fournier, Léo Gautheron, Nathalie T.H. Gayraud, Hicham Janati, Alain Rakotomamonjy, Ievgen Redko, Antoine Rolet, Antony Schutz, Vivien Seguy, Danica J. Sutherland, Romain Tavenard, Alexander Tong, and Titouan Vayer. Pot: Python optimal transport. *Journal of Machine Learning Research*, 22(78):1–8, 2021.

[Frogner *et al.*, 2015] Charlie Frogner, Chiyuan Zhang, Hossein Mobahi, Mauricio Araya, and Tomaso A Poggio. Learning with a wasserstein loss. In *Advances in Neural Information Processing Systems*, pages 2053–2061, 2015.

[Graves and Tomlin, 2003] Stephen C Graves and Brian T Tomlin. Process flexibility in supply chains. *Management Science*, 49(7):907–919, 2003.

[Gurobi Optimization, 2021] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2021.

[Harris and Ross, 1956] TE Harris and FS Ross. Fundamentals of a method for evaluating rail net capacities. *Research Memorandum*, 1573, 1956.

[Hughes and Chen, 2021] Jason Hughes and Juntao Chen. Fair and distributed dynamic optimal transport for resource allocation over networks. In *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2021.

[Jordan and Graves, 1995] William C Jordan and Stephen C Graves. Principles on the benefits of manufacturing process flexibility. *Management science*, 41(4):577–594, 1995.

[Khalil *et al.*, 2017] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

[Kusner *et al.*, 2015] Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In *International conference on machine learning*, pages 957–966. PMLR, 2015.

[Lien *et al.*, 2011] Robert W Lien, Seyed MR Iravani, Karen Smilowitz, and Michal Tzur. An efficient and robust design for transshipment networks. *Production and Operations Management*, 20(5):699–713, 2011.

[Liu and Nocedal, 1989] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[Maretic *et al.*, 2022] Hermina Petric Maretic, Mireille El Gheche, Matthias Minder, Giovanni Chierchia, and Pascal Frossard. Wasserstein-based graph alignment. *IEEE Transactions on Signal and Information Processing over Networks*, 8:353–363, 2022.

[Nishihara *et al.*, 2015] Robert Nishihara, Laurent Lessard, Ben Recht, Andrew Packard, and Michael Jordan. A general analysis of the convergence of admm. In *International conference on machine learning*, pages 343–352. PMLR, 2015.

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

[Schölkopf *et al.*, 2001] Bernhard Schölkopf, Ralf Herbrich, and Alex J Smola. A generalized representer theorem. In *Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory*, pages 416–426, 2001.

[Simchi-Levi and Wei, 2012] David Simchi-Levi and Yehua Wei. Understanding the performance of the long chain and sparse designs in process flexibility. *Operations research*, 60(5):1125–1141, 2012.

[Simchi-Levi *et al.*, 2018] David Simchi-Levi, He Wang, and Yehua Wei. Increasing supply chain robustness through process flexibility and inventory. *Production and Operations Management*, 27(8):1476–1491, 2018.

[Simon *et al.*, 2013] Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of computational and graphical statistics*, 22(2):231–245, 2013.

[Sinkhorn and Knopp, 1967] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21(2):343–348, 1967.

[Sultana *et al.*, 2022] Nasrin Sultana, Jeffrey Chan, Tabinda Sarwar, and AK Qin. Learning to optimise general tsp instances. *International Journal of Machine Learning and Cybernetics*, 13(8):2213–2228, 2022.

[Titouan *et al.*, 2019] Vayer Titouan, Nicolas Courty, Romain Tavenard, and Rémi Flamary. Optimal transport for structured data with application on graphs. In *International Conference on Machine Learning*, pages 6275–6284. PMLR, 2019.

[Villani, 2008] Cédric Villani. *Optimal transport: Old and new*, volume 338. Springer Science & Business Media, 2008.

[Wang and Banerjee, 2014] Huahua Wang and Arindam Banerjee. Bregman alternating direction method of multipliers. In *Advances in Neural Information Processing Systems*, pages 2816–2824, 2014.

[Wei *et al.*, 2021] Yehua Wei, Lei Zhang, Ruiyi Zhang, Shijing Si, Hao Zhang, and Lawrence Carin. Reinforcement learning for flexibility design problems. *arXiv preprint arXiv:2101.00355*, 2021.

[Xie *et al.*, 2020] Yujia Xie, Xiangfeng Wang, Ruijia Wang, and Hongyuan Zha. A fast proximal point method for computing exact wasserstein distance. In *Uncertainty in artificial intelligence*, pages 433–453. PMLR, 2020.

[Xu and Cheng, 2022] Hongteng Xu and Minjie Cheng. Regularized optimal transport layers for generalized global pooling operations. *arXiv preprint arXiv:2212.06339*, 2022.

[Yan *et al.*, 2018] Zhenzhen Yan, Sarah Yini Gao, and Chung Piaw Teo. On the design of sparse but efficient structures in operations. *Management Science*, 64(7):3421–3445, 2018.

[Ye *et al.*, 2017] Jianbo Ye, Panruo Wu, James Z Wang, and Jia Li. Fast discrete distribution clustering using wasserstein barycenter with sparse support. *IEEE Transactions on Signal Processing*, 65(9):2317–2332, 2017.

[Zhang and Zhu, 2019] Rui Zhang and Quanyan Zhu. Consensus-based distributed discrete optimal transport for decentralized resource matching. *IEEE Transactions on Signal and Information Processing over Networks*, 5(3):511–524, 2019.