mathematics

MDPI

Article

Gromov–Wasserstein Meets Combinatorial Optimization: A Scalable Solver for the Capacitated Quadratic Assignment Problem

Iman Seyedi [1,*], Antonio Candelieri [2], Enza Messina [1] and Francesco Archetti [1,*]

[1] Department of Computer Science Systems and Communication, University of Milano-Bicocca, 20126 Milan, Italy; enza.messina@unimib.it
[2] Department of Economics Management and Statistics, University of Milano-Bicocca, 20126 Milan, Italy; antonio.candelieri@unimib.it
* Correspondence: seyediman.seyedi@unimib.it (I.S.); francesco.archetti@unimib.it (F.A.)

Abstract

The Capacitated Quadratic Assignment Problem (CQAP) arises in logistics and network design, requiring the allocation of tasks to agents under quadratic interaction costs and capacity constraints. Classical exact solvers become computationally infeasible for large-scale instances, while heuristic methods such as Genetic Algorithms suffer from scalability limitations and sensitivity to local optima, leaving a gap for principled scalable approximations. In this paper, we address CQAP using the Gromov–Wasserstein (GW) framework, derived from Optimal Transport (OT) theory. In particular, we propose a multi-initialization GW strategy (GW_MultiInit) that mitigates the local optima problem inherent to non-convex GW optimization and scales efficiently to large problem sizes. Computational experiments on synthetic CQAP instances show that GW_MultiInit consistently achieves solutions close to the exact optimum for small- and medium-scale problems, and outperforms heuristic baselines such as the genetic algorithm at large scale in both runtime and solution quality across the benchmarks tested. To validate generalizability, we further evaluate GW_MultiInit On 17 QAPLIB benchmark instances adapted to the CQAP setting, GW_MultiInit achieves the best approximate result on 15 out of 17 instances with an average optimality gap of 0.34%, demonstrating strong generalizability beyond synthetic data. Additional comparisons with Entropic GW and Fused GW highlight practical trade-offs between accuracy, speed, and parameter sensitivity, offering guidelines for real-world deployment. Our results suggest that GW-based methods, and GW_MultiInit in particular, offer a promising and scalable approach for CQAP and related large-scale assignment problems within the problem scales examined.

Keywords: optimal transport; Capacitated Quadratic Assignment Problem; Gromov-Wasserstein; approximate optimization

MSC: 90C27; 90C59; 60B05; 90C26; 49Q20

Academic Editor: Janez Žerovnik

Received: 21 April 2026
Revised: 20 May 2026
Accepted: 29 May 2026
Published: 3 June 2026

1. Introduction

The assignment problem is a cornerstone of operations research, with applications in workforce allocation (e.g., assigning nurses to hospital shifts), supply chain management (e.g., matching vehicles to delivery routes), and computer vision (e.g., aligning keypoints between images). For further examples and detailed surveys, see [1–3]. At its core, it

seeks the best one-to-one matching between two sets, typically agents and tasks, such that the total cost is minimized. Classical solution approaches to assignment problems include combinatorial optimization and exact solvers. Traditionally, this has been addressed through combinatorial optimization techniques, with exact algorithms such as the Hungarian method providing efficient solutions for structured cases [4]. However, as real-world scenarios grow in complexity, these classical approaches reveal significant limitations. Many real-world applications involve more complex interactions than simple one-to-one assignments. For example, pairwise dependencies between tasks or facilities can arise and give rise to the Quadratic Assignment Problem, a well-known generalization of the assignment problem [5,6]. Building further on this, the Capacitated Quadratic Assignment Problem introduces capacity constraints on facilities and demand requirements on tasks. This richer formulation allows each facility to serve multiple tasks subject to its capacity, while each task must satisfy its demand. Like QAP, CQAP is NP-hard [7], with the consequence that exact algorithms quickly become computationally infeasible as problem sizes increase [8]. For this reason, many heuristics and metaheuristics, such as Genetic Algorithms (GA) [9] have been developed. This motivates the development of alternative formulations and approximation methods that balance scalability with solution quality.

Optimal Transport (OT) provides a modern mathematical framework to generalize assignment problems. The Wasserstein distance has become a powerful tool for comparing probability distributions with broad applications across machine learning and computer vision [10]. However, standard OT assumes that source and target distributions lie in the same metric space, restricting its application when relational structure differs across domains. The Gromov–Wasserstein (GW) distance [11] overcomes this limitation by comparing distributions across different metric spaces through their internal relational structures, making it particularly well-suited for QAP and CQAP: the quadratic cost term in QAP can be rewritten as a GW discrepancy, and capacity constraints map naturally onto the transport marginals. This is especially well-suited for the structural matching problems as posed by QAP and CQAP. In particular, the quadratic cost term in QAP can be rewritten as a GW discrepancy, so that CQAP is directly related to the GW framework. Alternative relaxation approaches for QAP-class problems lose the quadratic relational structure of the objective upon linearization, require a common ambient metric space that may not exist for general facility-location data, or scale poorly due to the large number of lifted variables required. The GW framework, by contrast, preserves the quadratic relational cost structure natively through its pairwise distance comparison mechanism, encodes capacity and demand constraints directly via the marginals of the transport polytope, and admits efficient entropic approximations that scale polynomially with problem size [12]. This combination of structural fidelity, constraint compatibility, and computational tractability makes GW a particularly principled choice for CQAP. Despite this natural connection, GW optimization is inherently non-convex and NP-hard, often converging to poor local minima [12] which is a limitation that existing GW variants do not fully address.

In this paper, we propose a multi-initialization GW strategy, called GW_MultiInit, for mitigating the non-convexity of GW optimization in solving CQAP by exploring multiple initial couplings and selecting the best solution. Unlike relying on a single initialization, GW_MultiInit explores multiple random initial couplings and selects the best solution, hence mitigating the risk of poor local optima. We benchmark GW_MultiInit against exact solvers, GA, and other GW variants (EGW and FGW) across CQAP instances. Our results reveal that GW_MultiInit consistently achieves near-optimal solutions, significantly outperforms GA in scalability, and remains computationally efficient compared to exact methods.

The remainder of this paper is structured as follows. Section 2 introduces Related Work and Background on Assignment Problems and solution methods in detail. Section 3 es-

tablishes their connection to OT. Section 4 presents the Gromov Wasserstein distance, focusing on GW optimization and its variants, entropic, fused GW and GW_MultiInit. Section 5 provides computational experiments across synthetic CQAP instances of varying scales (Section 5.1) and standard QAPLIB benchmark instances adapted to the CQAP setting (Section 5.2). Section 6 concludes with limitations and future research directions.

2. Related Work and Background

2.1. Assignment Problems: From AP to CQAP

The classical assignment problem (AP) is a combinatorial optimization problem that seeks to assign a set of agents (or facilities) to a set of tasks (or locations) in a way that minimizes the total assignment cost [6]. To define the AP formally, we introduce the following notation:

n: number of facilities and locations.

x_i as the i-th facility, y_j as the j-th location.

$c_{ij} = c(x_i, y_j)$ as the cost of assigning facility i to *location* j.

$$x_{ij} = \begin{cases} 1 \; if \; facility \; i \; is \; assigned \; to \; location \; j \\ 0 \qquad\qquad\qquad\qquad otherwise \end{cases}$$

The AP can then be formulated mathematically as:

$$\min_{x_{ij}} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

subject to:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i \quad \text{(each facility is assigned to one location)} \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j \quad \text{(each location hosts one facility)} \tag{3}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \tag{4}$$

This linear integer programming formulation ensures a one-to-one assignment of agents to tasks while minimizing the total cost. The AP is widely used in logistics, scheduling, and resource allocation problems and serves as the foundation for more complex assignment models, such as the Quadratic Assignment Problem (QAP). The QAP is a classic combinatorial optimization problem. It generalizes the linear assignment problem by adding pairwise interaction costs between assignments [5,6]. Koopmans and Beckmann first formulated it for facility layout [5]. The QAP models situations where the cost of assigning an agent to a task depends not only on that assignment but also on the assignments of other agents. This quadratic dependency makes QAP a natural framework for facility location, electronic circuit design, graph matching, and structural pattern recognition [13]. We consider two matrices:

$F \in \mathbb{R}^{n \times n}$: flow matrix, where F_{ik} denotes the interaction (flow) between facility i and facility k.

$D \in \mathbb{R}^{n \times n}$: distance (or dissimilarity) matrix, where D_{jl} represents the distance between *location* j and location l.

The objective of QAP is to find the bijective assignment $\sigma : \{1, \ldots, n\} \to \{1, \ldots, n\}$ that minimizes the total interaction cost:

$$\min_{\sigma \in S_n} \sum_{i=1}^{n} \sum_{k=1}^{n} F_{ik} \cdot D_{\sigma(i)\sigma(k)} + \sum_{i=1}^{n} C_{i\sigma(i)} \tag{5}$$

where:

The first term captures quadratic costs due to interactions between pairs of facilities and the corresponding distances between their assigned locations.

The optional second term $C_{i\sigma(i)}$ represents linear assignment costs that may be present in hybrid formulations.

The binary integer programming form of QAP uses decision variables $x_{ij} \in \{0,1\}$ indicating whether facility i is assigned to *location* j:

$$\min_x \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n} F_{ik} \cdot D_{jl} \cdot x_{ij} \cdot x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{n} C_{ij} \cdot x_{ij} \tag{6}$$

Subject to (2)–(4).

This quadratic objective makes the problem NP-hard [7], rendering it intractable for $n \gtrsim 30$ in most cases.

The CQAP extends the classical QAP by introducing capacity constraints on facilities (agents) and demand requirements on locations (tasks). Unlike the standard QAP, where each facility is assigned to exactly one location, the CQAP allows facilities to serve multiple locations, subject to capacity limits, while locations must satisfy their demand requirements. This formulation is especially relevant in several real-world applications that involve pairwise interaction costs and heterogeneous resource constraints, such as vehicle assignment to service regions, supply chain network design, and communication resource allocation [13,14]. To formalize the CQAP, we introduce the following parameters:

u_i: capacity of facility i.

d_j: demand at *location* j.

m: number of locations (not necessarily equal to n).

The CQAP objective is:

$$\min_x \sum_{i=1}^{n}\sum_{k=1}^{n}\sum_{j=1}^{m}\sum_{l=1}^{m} F_{ik} \cdot D_{jl} \cdot x_{ij} \cdot x_{kl} + \sum_{i=1}^{n}\sum_{j=1}^{m} C_{ij} \cdot x_{ij} \tag{7}$$

subject to capacity and demand constraints:

$$\sum_{j=1}^{m} d_j x_{ij} \leq u_i, \forall i = 1, \ldots, n \qquad \text{(Each facility cannot exceed its capacity)} \tag{8}$$

$$\sum_{i=1}^{n} d_j x_{ij} = d_j, \forall j = 1, \ldots, m \qquad \text{(Each location's demand must be met)} \tag{9}$$

$$x_{ij} \in \{0,1\}, \forall i, j \tag{10}$$

This capacity constraint extension makes the problem even more computationally challenging than standard QAP, belonging to the class of NP-hard problems. Nevertheless, it provides a realistic modeling tool for problems involving both interaction costs and heterogeneous resource constraints.

2.2. Solution Methods for QAP and CQAP

Exact methods. Classical exact solution approaches to QAP include the Hungarian algorithm [6,15], branch-and-bound [16], and mixed-integer quadratic programming (MIQP) [17]. While these methods guarantee global optimality, their exponential worst-case complexity makes them infeasible as problem sizes increase [8]. For CQAP, exact MIQP formulations can solve small instances by globally optimizing the nonconvex quadratic objective via branch-and-bound methods. However, the combinatorial explosion inherent to NP-hard quadratic programs causes runtimes to grow super-exponentially with problem size, rendering exact methods computationally infeasible beyond small scales [18]. The

additional capacity constraints of CQAP further tighten the feasible region and increase the number of branching nodes required, compounding this scalability challenge and limiting the reach of exact approaches even more severely than in the uncapacitated QAP [13,19].

Heuristic and metaheuristic methods. Given the NP-hardness of QAP and CQAP, a rich body of heuristic and metaheuristic methods has been developed. Genetic algorithms (GA) [20,21], are among the most widely applied, exploiting evolutionary operators to search the permutation space. Simulated annealing [22], tabu search [23], and ant colony optimization [24] have also been applied to QAP with competitive results on medium-scale instances. We note that quantum and hybrid quantum-classical heuristics have also been recently explored for combinatorial optimization problems [25,26]. However, these methods degrade in solution quality and runtime as problem size increases, and are sensitive to hyperparameter tuning. For CQAP specifically, the additional capacity constraints complicate the feasibility-preserving mutation and crossover operations that standard metaheuristics rely on, limiting their direct applicability [27].

Relaxation-based methods. Semidefinite programming (SDP) relaxations of QAP provide tighter lower bounds than linear relaxations [28], but require solving a large-scale SDP whose size grows as $O(n^4)$, making them impractical for large cases. Lagrangian decomposition approaches [29] decompose the QAP into tractable subproblems but rely on dual bound tightness that degrades for dense flow matrices. Linear programming relaxations of the assignment polytope lose the quadratic relational structure of the objective upon relaxation, yielding weak bounds [18]. These limitations collectively motivate the development of structure-preserving approximation methods that maintain the quadratic relational cost structure while scaling to large instances.

Optimal transport-based methods. OT provides a modern mathematical framework to generalize assignment problems. Originating from Monge's classical problem [30], OT was reformulated by Kantorovich [31] into a convex optimization problem, enabling both probabilistic couplings and capacity constraints [32]. The resulting optimal transport cost naturally induces the Wasserstein distance when the ground cost is chosen as a metric, which has since become a powerful tool for comparing probability distributions, with broad applications in statistics, machine learning, and computer vision [33–37]. However, standard OT assumes that source and target distributions lie in the same metric space, restricting its application to structured data like graphs or facilities with relational constraints. To overcome this limitation, the GW distance [11,38] extends Wasserstein, comparing distributions across different metric spaces by aligning them through their internal relational structures. A key observation is that the GW problem is itself a relaxation of QAP. The GW objective minimizes a quadratic cost over soft assignments, making it structurally equivalent to a continuous relaxation of QAP [39,40]. This connection is not merely formal; ref. [12] proves that computing the exact GW distance is NP-hard, confirming that GW inherits the computational intractability of QAP. Consequently, all practical applications of GW rely on local heuristics or approximate solvers rather than exact computation [41]. The most widely used approach solves a sequence of nested entropy-regularized OT problems (Entropic GW, EGW [42]), which improves tractability through Sinkhorn iterations at the cost of solution accuracy. Fused GW (FGW) [38] further extends GW by incorporating node-feature information alongside structural costs, enabling richer comparisons when auxiliary data is available, but introducing an additional hyperparameter that requires tuning. Low-rank GW approximations [41] reduce the cubic complexity of EGW to near-linear, at the cost of restricting the coupling to a low-rank factorization. Despite these advances, GW's inherent non-convexity means that all these methods are sensitive to initialization and prone to poor local minima [12,13].

2.3. Research Gap

The preceding review reveals three specific gaps that motivate this work. First, exact methods for CQAP are computationally infeasible beyond small scale, and existing heuristics do not scale well or handle capacity constraints naturally. Second, while GW optimal transport provides a principled connection to QAP through the quadratic relational cost structure, this connection has not been formally extended to CQAP with explicit treatment of capacity constraints as transport marginals. Third, GW optimization is inherently non-convex and sensitive to initialization, yet no systematic multi-initialization strategy has been proposed or evaluated for CQAP. GW_MultiInit directly addresses all three gaps.

This paper addresses the following research questions: (RQ1) Can the Gromov–Wasserstein optimal transport framework provide a scalable and accurate approximation for the CQAP? (RQ2) Does a multi-initialization strategy for GW optimization meaningfully reduce the risk of poor local optima compared to single-start alternatives? (RQ3) How do GW-based methods compare to established heuristics (GA) and GW variants (EGW, FGW) across different problem scales in terms of solution quality and runtime?

3. From Assignment to Optimal Transport

While QAP and CQAP arise naturally in discrete optimization, their structure aligns closely with Optimal Transport (OT). OT originates from Monge's classical problem [30] and was later reformulated by Kantorovich [31], enabling mass splitting and convex optimization methods [32]. In the discrete case, which is the one most relevant here, the assignment problem appears as a special case of OT when transport plans are restricted to permutation matrices. Moreover, many studies have employed OT for optimization in various contexts [43].

3.1. From Monge's Problem to the Assignment Problem

The origins of optimal transport can be traced back to the problem of moving piles of earth to fill holes while minimizing the total transport cost [44]. Formally, consider two discrete distributions of "mass" [30]:

$$\mu = \sum_{i=1}^{n} \mu_i \delta_{x_i},\ \nu = \sum_{j=1}^{n} \nu_j \delta_{y_j} \tag{11}$$

where:

- $x_i \in \mathbb{R}^d$ represents the i-th location point in the source distribution;
- $y_j \in \mathbb{R}^d$ represents the j-th location point in the target distribution;
- $\mu_i > 0$ and $\nu_j > 0$ are the amounts of mass at each source and target point;
- δ_{x_i} and δ_{y_j} are Dirac delta measures at x_i and y_j, respectively.

The Monge problem seeks a transport map $T : \{x_1, \ldots, x_n\} \to \{y_1, \ldots, y_m\}$ that minimizes the total transport cost:

$$\min_{T} \sum_{i=1}^{n} c(x_i, T(x_i)) \quad subject\ to \quad T\#\mu = \nu \tag{12}$$

where:

$c(x_i, y_j)$ is the cost of moving a unit of mass from x_i to y_j;

$T\#\mu = \nu$ means that pushing forward the distribution μ through T exactly reproduces ν, i.e., the total mass assigned to each target location equals its prescribed mass.

In the discrete uniform case, where ($a_i = b_j = \frac{1}{n}$, $n = m$), the transport map T reduces to a bijection. In this setting, Monge's problem is equivalent to the classical AP, where each source is assigned to exactly one target while minimizing the total assignment cost, Equations (1)–(4).

3.2. Beyond Monge: Kantorovich Relaxation

While elegant, Monge's formulation is restrictive. It forbids splitting mass and may not admit a solution. These limitations motivated Kantorovich's 1942 relaxation [31], which replaces transport maps with transport plans. Instead of assigning each source to exactly one target, Kantorovich allows fractions of mass to be distributed across multiple targets. This both guarantees the existence of solutions and recasts the problem as a convex linear program.

For discrete distributions with source masses $\mu = (\mu_1, \ldots, \mu_m)$ and target masses $\nu = (\nu_1, \ldots, \nu_n)$, $\mu_i, v_j \geq 0$, $\sum_i \mu_i = \sum_j v_j = 1$ let the transport plan become a matrix where $T_{i,j}$ represents the mass transported from source i to target j.

$$\min_{T \in \mathbb{R}_+^{\{m \times n\}}} \sum_{i=1}^{m} \sum_{j=1}^{n} C_{i,j} \cdot T_{i,j} \tag{13}$$

Subject to:

$$\sum_{j=1}^{n} T_{i,j} = \mu_i \quad \text{for all } i = 1, ..., m \text{ (source constraints)} \tag{14}$$

$$\sum_{i=1}^{m} T_{i,j} = v_j \quad \text{for all } j = 1, \ldots, n \text{ (target constraints)} \tag{15}$$

$$T_{i,j} \geq 0 \quad \text{for all } i, j \quad \text{(non-negativity)} \tag{16}$$

The assignment problem arises as a special case when all masses are unitary ($\mu_i = v_j = 1$) and the feasible set is restricted to permutation matrices. In this sense, AP corresponds to the extreme points of the Kantorovich polytope.

3.3. Optimal Transport and CQAP

While the Kantorovich formulation overcomes Monge's limitations by allowing mass splitting, ensuring solution existence, and admitting a convex optimization structure, it remains linear; it only models costs between individual source–target pairs (x_i, y_j) and does not model interactions between multiple pairs simultaneously. In contrast, the QAP and its continuous analogue (CQAP) depend on pairwise relational costs, where the objective reflects interactions between assignments of pairs. Such relational dependence cannot be captured by Kantorovich's framework, which lacks a way to compare internal geometries of the two domains. The Gromov–Wasserstein (GW) distance addresses this gap by replacing pointwise costs with comparisons of intra-domain relations [11,38]. Specifically, the quadratic term in Equation (6) is equivalent in form to the GW cost function when F and D are interpreted as intra-domain distances (or similarities) in the source and target spaces.

The capacity constraints of CQAP are incorporated into the GW framework through the choice of marginal distributions. Specifically, we set $\mu_i = u_i/U$ and $v_j = d_j/D$, where $U = \Sigma u_i$ and $D = \Sigma d_j$ are the total facility capacity and total task demand, respectively (assumed equal by feasibility). Under this construction, the marginal constraint $\Sigma_j T_{i,j} = \mu_i$ in the transport polytope $\Pi(\mu, \nu)$ directly encodes the capacity restriction of facility i, and $\Sigma_i T_{i,j} = v_j$ encodes the demand of task j. The quadratic objective of CQAP maps to the GW discrepancy by identifying the flow matrix F as the intra-facility structural matrix C_X and the distance matrix D as the intra-location structural matrix C_Y.

4. Gromov–Wasserstein (GW) Distance

4.1. Wasserstein Distances as Transport Costs

The Wasserstein distance is a key link between optimal transport theory and modern statistics. It provides a solid way to compare probability distributions and has impacted machine learning, statistics, and computational optimization [45–47]. Unlike traditional

measures like Kullback–Leibler divergence or total variation distance, Wasserstein distances follow metric rules and give meaningful geometric measures of similarity that respect the structure of the space. The p-Wasserstein distance between probability measures μ and ν supported on the same ambient metric space (Ω, d) is defined through the Kantorovich formulation [45]. Let $\{x_i\}_{i=1}^{n} \subset X$ and $\{y_j\}_{j=1}^{m} \subset Y$ be the supports of two discrete probability measures $\mu = \sum_{i=1}^{n} \mu_i \delta_{x_i}$, $\nu = \sum_{j=1}^{n} \nu_j \delta_{y_j}$ with weights $\mu_i \geq 0$, $\nu_j \geq 0$ and $\sum_i \mu_i = \sum_j \nu_j = 1$ (or equal total mass if not normalized).

Define the cost $c(x_i, y_j) = d(x_i, y_j)^p$ for some metric d and exponent $p \geq 1$. Then the p-Wasserstein distance satisfies:

$$W_p^p(\mu, \nu) = \min_{T \in \Pi(\mu,\nu)} \sum_{i=1}^{n} \sum_{j=1}^{m} c(x_i, y_j) T_{ij} \quad = \min_{T \in \Pi(\mu,\nu)} \sum_{i=1}^{n} \sum_{j=1}^{m} d(x_i, y_j)^p T_{ij} \tag{17}$$

where the set of couplings (transport plans) is $\Pi(\mu, \nu) = \left\{ T \in R_{\geq 0}^{n \times m} : T1_m = \mu, T^\top 1_n = \nu \right\}$. Where $\Pi(\mu, \nu)$ is the set of all couplings T with marginals μ and ν. This formulation directly connects Wasserstein distances to optimal transport by interpreting the distance as the minimum total cost of transporting mass from sources x_i to targets y_j under the cost function $c(x_i, y_j) = d(x_i, y_j)^p$.

The 2-Wasserstein distance ($p = 2$) admits an equivalent Monge formulation, which interprets the distance as the minimum "effort" to transport one distribution to another deterministically. This geometric perspective underlies many applications, including interpolation between distributions and computation of Wasserstein Barycenters [48]. For Gaussian distributions, the 2-Wasserstein distance has a closed-form solution via the Bures metric, which reduces to simpler expressions in diagonal or commutative cases. For general distributions, exact computation is costly, motivating approximate solutions such as entropic regularization and Sinkhorn-based algorithms.

4.2. Gromov Wasserstein

The classical Wasserstein distance is powerful for comparing probability measures when both are defined on the same metric space. Its cost relies on computing distances $c(x_i, y_j) = d(x_i, y_j)^p$ between points across the two measures. However, this becomes limiting when the distributions live in different or unrelated spaces, where no natural cross-domain distance is available. In such cases, the Wasserstein distance cannot be directly applied. The GW distance is a strong generalization of the classic Wasserstein distance. It solves a key problem: comparing distributions that are not in the same space. GW compares distributions using their internal structural relationships. It extends optimal transport to cases where distributions cannot be compared point by point [49]. Instead of comparing points $x_i \in X$ and $y_j \in Y$ directly, GW compares the internal structures of the two metric spaces by aligning their pairwise distance matrices, where the intrinsic geometry of the objects is more important than any embedding into a common space. This perspective makes GW well suited for comparing metric–measure (mm) spaces, where the geometry of the objects is more important than any embedding into a common ambient domain. An mm space is defined as a triple $\mathcal{X} = (X, d_x, \mu)$ where X is a set d_x is a metric on X, and $\mu \in M^+(X)$ is a finite nonnegative measure over X (typically a probability measure). Many data structures naturally fit this framework. For instance, a point cloud $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ equipped with the Euclidean metric $d_X(x_i, x_j) = \|x_i - x_j\|_2$ and the uniform measure $\mu = \frac{1}{N} \sum_i \delta_{x_i}$ constitutes an mm space; similarly, a graph becomes an mm space by letting X be the set of nodes, d_x the shortest-path metric, and μ the uniform distribution on nodes [50]. GW aims to define a meaningful distance between two mm spaces $\mathcal{X} = (X, d_x, \mu)$ and $\mathcal{Y} = (Y, d_y, \nu)$, each represented as a probability measure over

their elements and attributes. To formalize this idea, we define the structural distance matrices:

$C_X \in \mathbb{R}^{n\times n}$, $C_Y \in \mathbb{R}^{m\times m}$ denote the pairwise structural dissimilarities within each graph, $C_X(i,j) = d_X(x_i, x_j)$ and $C_Y(k,l) = d_Y(y_k, y_l)$. Where $X = \{x_1, \ldots, x_n\}$ with measure $\mu \in \mathbb{R}^n$, and $Y = \{y_1, \ldots, y_m\}$ with measure $\nu \in \mathbb{R}^m$.

$$GW_p^p(\mu,\nu) = \min_{T\in\Pi(\mu,\nu)} \sum_{i,j=1}^{n} \sum_{k,l=1}^{m} L\Big(C_X(i,j), C_Y(k,l)\Big) T_{ik} T_{jl} \tag{18}$$

where $\Pi(\mu,\nu) = \Big\{T \in \mathbb{R}_+^{n\times m} : T1_m = \mu,\ T^\top 1_n = \nu\Big\}$ is the transportation polytope. A common choice is the squared difference loss:

$$L(C_X(i,j), C_Y(k,l)) = |\, C_X(i,j) - C_Y(k,l)\,|^p, \tag{19}$$

Intuitively, the GW objective penalizes discrepancies between the pairwise distances $C_X(i,j)$ within the source domain and $C_Y(k,l)$ within the target domain, weighted by the transport plan T (Figure 1). A perfect structural match—an isometry between the two spaces—would yield a GW cost of zero. In practice, since no exact isometry typically exists between F and D in CQAP instances, the optimizer seeks a soft coupling T that minimizes the total structural distortion. The quadratic dependence on T (through the $T_{ik}.T_{jl}$ product) is what makes this a non-convex problem and directly links it to QAP, where $x_{ij}\cdot x_{kl}$ encodes the same pairwise assignment interaction.

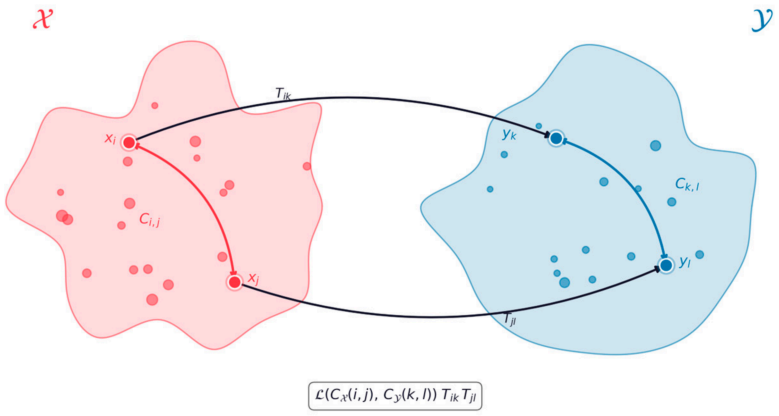

Figure 1. Illustration of GW related to Equation (18). Arrows represent transport couplings (T_{ik} and T_{jl}) between points in the source space $\mathcal{X}$ and target space $\mathcal{Y}$.

The assignment between two graphs (graph isomorphism) is known to be NP-hard [7]. To properly address such cases, one must consider mm-spaces, which take into account both the measure and the metric jointly. The Gromov–Wasserstein problem is a non-convex quadratic program over the transport plan $T \in \Pi(\mu,\nu)$, with a quadratic dependence $T \otimes T$ that compares the pairwise structural distances $C_X(i,j)$ and $C_Y(k,l)$, as in Equation (18). This quadratic term is what characterizes GW as a Quadratic Assignment Problem (QAP) and makes it NP-hard [12,13]. Since GW is non-convex, standard algorithms only guarantee a stationary point of the objective. One common approach is to linearize the quadratic program, generating a sequence of updates where T is optimized iteratively [11]. Variants of GW exploit additional information, such as computationally efficient lower bounds derived from global histograms of distances [51], which can be computed via classical OT and used as initialization for the GW optimization. The non-convexity of GW optimization and the resulting sensitivity to initialization motivate the development of

more robust solution strategies. A broader review of computational GW solvers from the literature is provided in Appendix A.

4.3. Gromov Wasserstein Extensions

While the standard GW distance is effective for comparing structured data, many real-world problems require additional flexibility. To address this, several extensions of GW have been proposed. These include the Fused Gromov–Wasserstein distance, which incorporates both structural and feature information and Entropic Gromov Wasserstein. This section introduces each of these extensions and highlights their motivations and use cases.

4.3.1. Entropic Gromov Wasserstein

As we mentioned in the previous section, the GW distance involves solving a non-convex quadratic assignment problem, known to be NP-hard [52]. Traditional exact optimization techniques and convex relaxations (e.g., semidefinite programming or eigenvalue relaxations) often suffer from scalability issues, as they may require up to $O(n^4)$ variables (e.g., $n^2 \times n^2$ for convex formulations) [11,42]. To address these computational limitations, Entropic Gromov–Wasserstein (EGW) was introduced as a scalable, differentiable approximation to the original GW problem by adding an entropy regularization term to the optimization objective [42]. Unlike classical methods that rely on rigid constraints and non-convex optimization schemes, EGW maintains the coupling constraints while providing global convergence guarantees and improved numerical stability. One of the key strengths of the entropic formulation is its flexibility: it naturally extends to structured transport problems, enabling efficient solutions for Fused Gromov Wasserstein (FGW) and Unbalanced Gromov Wasserstein (UGW). The algorithm operates in an iterative framework, where each iteration consists of two main steps [38,52]:

- Gradient computation of the GW objective via tensor contractions, which has a computational cost of $O(n^3)$;
- Sinkhorn updates to solve the resulting entropic OT subproblem have a per-iteration cost of $O(n^2)$.

But since these updates are performed repeatedly within the GW optimization loop, the overall computational complexity of the full GW algorithm remains $O(n^3)$. This separation of the GW structure-matching step and the entropy-regularized transport update makes EGW particularly effective for medium-scale graphs. However, the cubic complexity in the number of nodes remains a bottleneck in large-scale settings. Entropic regularization addresses the GW objective with a negative entropy term, yielding the EGW formulation. Specifically, let $L(T)$ denote the unregularized GW loss function defined in (18), and let $\varepsilon > 0$ be the entropic regularization parameter. The entropy of the coupling T is $H(T) = -\sum_{i,k} T_{i,k}\, log T_{i,k}$: entropy of the coupling.

The general entropic OT problem augments the OT objective with a Kullback–Leibler (KL) divergence penalty:

$$EGW_\varepsilon(\mu,\nu) = \min_{T \in \Pi(\mu,\nu)} L(T) + \varepsilon\, KL(T \parallel \mu \otimes \nu) \tag{20}$$

where $\Pi(\mu,\nu)$ is the transportation polytope defined in (18) and $KL(T \parallel \mu \otimes \nu)$ is the standard KL divergence between T and the independent measure $\mu \otimes \nu$.

Specializing this formulation to the GW setting in Equation (18) yields the entropic GW objective:

Specializing this to the Gromov–Wasserstein setting, where the quadratic loss compares the structural distance matrices $C_X(i,j)$ and $C_Y(k,l)$, yields the entropic GW objective:

$$EGW_p(C_X, C_Y; \mu, \nu) = \min_{T \in \Pi(\mu,\nu)} \sum_{i,j=1}^{n} \sum_{k,l=1}^{m} L\Big(C_X(i,j), C_Y(k,l)\Big) T_{ik} T_{jl} \quad - \varepsilon H(\pi) \tag{21}$$

where $L(C_X(i,j), C_Y(k,l))$ is defined in (19). In practice, replacing the KL divergence with the negative entropy term enables efficient Sinkhorn-type algorithms, which iteratively scale the transport matrix T while preserving the marginal constraints μ and ν. These algorithms achieve linear convergence rates, are amenable to GPU acceleration, and reduce memory requirements compared to unregularized GW solvers. Entropic GW is particularly valuable in large-scale settings, soft-matching scenarios, and unbalanced OT problems, where approximate transport solutions and relaxed mass constraints can be naturally incorporated through the entropic penalty [42,53].

4.3.2. Fused Gromov Wasserstein

While GW captures structural similarity, many practical graphs also come with node features (e.g., text, categories, vectors). The FGW distance extends GW by combining both feature-level and structure-level comparisons in a unified framework [54].

Figure 2 illustrates the key idea behind the FGW distance. It visually represents how FGW aligns nodes between two graphs by considering both feature similarity (e.g., node attributes or embeddings) and structural similarity (relationships between nodes captured by the adjacency or structure matrices).

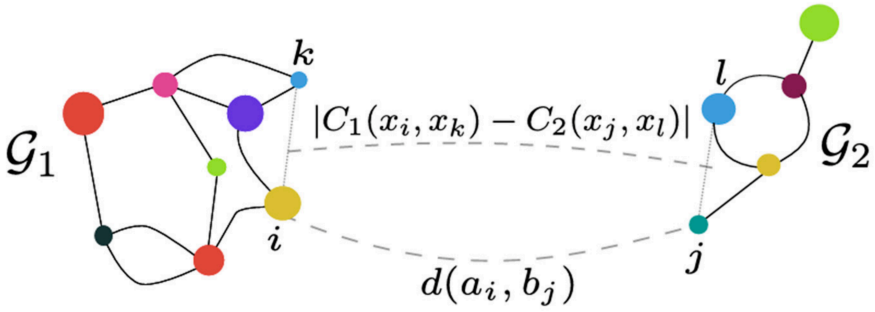

Figure 2. Feature and Structural Alignment in Fused Gromov–Wasserstein Coupling from [38]. Colored nodes represent node features/attributes, while the gray dashed lines indicate the feature-matching term $d(a_i, b_j)$ and the structure-preserving term $| C_1(x_i, x_k) - C_2(x_j, x_l) |$ in the FGW objective.

Let $M_{ij} = d\big(a_i, b_j\big)$ be a feature dissimilarity cost (e.g., Euclidean or cosine distance),

Equation (19) can be used in terms of the loss function to formulate the FGW distance as a trade-off between node feature similarity and graph structure similarity [38]:

$$FGW_p^{\alpha}(\mu, \nu) = \min_{T \in \Pi(\mu,\nu)} \sum_{i,\acute{i}=1}^{n} \sum_{j,\acute{j}=1}^{m} \varepsilon_p^{\alpha}(T; M, C_X, C_Y) \tag{22}$$

where:

$$\varepsilon_p^{\alpha}(\pi; M, C_X, C_Y) = (1-\alpha) d\big(a_i, b_j\big)^p + \alpha \, L\Big(C_X(i,j), C_Y(k,l)\Big) T_{ik} T_{jl} \tag{23}$$

where $\alpha \in [0,1]$ is a trade-off parameter:

- $\alpha = 0$: pure feature-based comparison (Wasserstein);
- $\alpha = 1$: pure structure-based comparison (Gromov–Wasserstein);
- $\alpha \in (0,1)$: fused distance accounting for both aspects.

4.3.3. Multiple Initialization Strategy for Gromov–Wasserstein Optimization

The Gromov–Wasserstein problem presents significant computational challenges because its objective function is non-convex. This creates many local optima that can trap optimization algorithms. Standard GW solvers usually start with a single initialization, often using the product of marginal distributions. But this often leads to suboptimal solutions, especially with complex cost matrix structures. To improve the exploration of the optimization space and mitigate the risk of getting stuck in local minima, we adapted an approach borrowed from non-convex optimization, called multi-initialization (GW_MultiInit), which runs the standard GW algorithm from multiple randomly generated starting points and selects the solution with the lowest objective value [55]. The specifics of the algorithm developed are provided in the following pseudo-code Figure 3:

Input: *Cost matrices* $C_X \in \mathbb{R}^{n\times n}$, $C_Y \in \mathbb{R}^{m\times m}$, *marginals* $\mu \in \mathbb{R}^n$, $\nu \in \mathbb{R}^m$, *Multi start (T)*

Output: *Optimal coupling* $T^* \in \mathbb{R}^{n\times m}$

Initialize: $f^* \leftarrow +\infty$, $T^* \leftarrow \emptyset$

1. Try default GW: $T_0 \leftarrow GW(C_X, C_Y, \mu, \nu)$, $f_0 \leftarrow \mathcal{L}_{GW}(T_0)$ *if* $f_0 < f^*$ *then* $f^* \leftarrow f_0, T^* \leftarrow T_0$

2. *for* $t = 1, \ldots, T$ *do*

- *Generate random coupling:* $G_t \leftarrow Uniform(0,1)^{n\times m} + \epsilon$
- *Repeat*:

$$G_t \leftarrow diag\left(\frac{\mu}{G_t 1_m}\right) G_t$$

$$G_t \leftarrow G_t diag\left(\frac{\nu}{G_t^T 1_n}\right) until \ \| G_t 1_m - \mu \|_\infty < \delta \ and \ \| G_t^T 1_n - \nu \|_\infty < \delta$$

- *Solve GW with random init:* $P_t \leftarrow GW(C_1, C_2, \mu, \nu, G_t)$
- *Evaluate:* $f_t \leftarrow L_{GW}(T_t)$ *if* $f_t < f^*$ *then* $f^* \leftarrow f_t, T^* \leftarrow T_t$

3. return T^*

Where: $\mathcal{L}_{GW}(T) = \sum_{i,j,k,\ell}(C_X^{ij} - C_Y^{k\ell})^2 T_{ik} T_{j\ell}$, $\delta = 10^{-12}$

Figure 3. Pseudo-code for Gromov–Wasserstein with Multiple Random Initializations (GW-MultiInit).

In each call to $GW(C_X, C_Y, \mu, \nu, G_t)$, the underlying solver (the conditional gradient method implemented in the Python Optimal Transport library [56]) iterates until the Frank–Wolfe gap falls below a tolerance of $tol = 10^{-9}$ or a maximum of 1000 iterations is reached, whichever occurs first. No additional stagnation criterion is applied at the multi-initialization level. The outer loop runs a fixed number of T trials regardless of intermediate objective values. The GW_MultiInit algorithm was run with T = 10 random restarts in all main experiments. This value was selected based on a sensitivity analysis over $T \in \{1, 3, 5, 7, 10, 15, 20\}$ using 20 independent runs per configuration across all benchmark instances (see

Appendix B). Solution stability plateaus at $T = 10$ with no meaningful variance reduction for larger T, while runtime scales linearly, making $T = 10$ the best trade-off between robustness and computational cost.

The algorithm begins by solving the standard GW problem with a default initialization, then performs T additional trials, each starting from a randomly generated valid coupling matrix. For each random initialization, we first generate a uniform random matrix and project it onto the coupling polytope $\Pi(\mu, \nu)$ using Sinkhorn iterations to ensure marginal constraints are satisfied. The GW algorithm is then executed from this random starting point. Upon completion of all $T + 1$ trials (the default initialization plus T random restarts), the coupling achieves the lowest GW objective value $\mathcal{L}_{GW}(T) = \sum_{i,j,k,\updownarrow} \left(C_X^{ij} - C_Y^{kl}\right)^2 T_{ik} T_{jl}$ across all trials is returned the final solution T^*. This multi-initialization strategy leverages the fact that different starting points can lead the iterative GW solver to different local optima, thereby increasing the likelihood of finding a globally optimal or near-optimal solution to the challenging non-convex Gromov–Wasserstein matching problem. A systematic review of related multi-initialization and multi-start strategies proposed in the GW literature is provided in Appendix A.

5. Computational Results

In our computational experiments, we address the CQAP through the lens of GW optimization, a framework originally developed for comparing structured distributions in different metric spaces. Our evaluation is organized into two complementary parts. First, in Section 5.1, we assess all methods on a set of synthetic CQAP instances divided by problem size. Second, in Section 5.2, we evaluate all approximate methods on 17 real-world instances drawn from the QAPLIB benchmark library [57], adapted to the CQAP setting by augmenting the standard flow and distance matrices with randomly generated capacity and demand constraints, providing an external validity check on the conclusions drawn from synthetic data.

Across both benchmarks, we conduct a systematic evaluation using several methods: exact quadratic solvers (where feasible), standard GW, GW_MultiInit, Entropic GW (EGW), Fused GW (FGW), and the Genetic Algorithm (GA). The GA is included as a widely used heuristic for solving QAP-type problems [9,58,59], allowing us to benchmark the performance of our optimal transport-based methods against established, general-purpose optimization approaches. Each approach provides a different balance between computational efficiency and objective quality. The exact MIQP solver (Gurobi with NonConvex = 2) serves as the optimality reference for small and medium instances but is excluded for large instances, where it becomes computationally infeasible. EGW and FGW are included as GW variants offering complementary tradeoffs. EGW trades accuracy for speed through entropic regularization, while FGW incorporates node-feature information not present in standard CQAP instances and thus serves as a structural comparison rather than a direct competitor. Our goal is to assess how well these methods approximate the CQAP solution under scalability constraints, and to identify which variants are most suitable for practical deployment depending on the problem size and resource availability.

Solution quality is measured as the percentage gap from the best-known solution, where the reference is the exact optimum for small and medium instances and the best result across all approximate methods for large instances. All results are averaged over 30 independent runs. Methods were not constrained to a common time budget. Each ran to its natural convergence under its own stopping criterion, so runtime figures reflect typical deployment conditions rather than a controlled budget comparison. Additionally, runtime comparisons should be interpreted with the awareness that different methods operate under different internal computational settings for instance, GW-based methods

leverage the POT library's optimized C backend, while the GA relies on Python-level evolutionary operators, so observed runtime differences reflect both algorithmic efficiency and implementation characteristics.

All experiments were run on a laptop with a 13th Gen Intel Core i7-1355U processor (1.70 GHz) and 16 GB RAM, using a 64-bit Windows operating system. For Gromov–Wasserstein distance estimation, we employed the Python Optimal Transport (POT) library (version 0.9.5) [56], together with supporting libraries such as NumPy (version 1.24.4), SciPy (version 1.10.1) for numerical routines, and Matplotlib (version 3.7.5) for visualization. The full implementation of all algorithms and the experimental setup is available on GitHub at https://github.com/iman-ie/GW_CQAP for reproducibility and further research.

5.1. Synthetic CQAP Instances

We evaluate all methods on a set of synthetic CQAP instances spanning three problem scales, Small ($n \leq 6$), Medium ($10 \leq n \leq 20$), and Large ($30 \leq n \leq 100$), as described in Table 1. Each instance specifies the number of agents (facilities) and tasks (locations); capacities and demands are assigned as random integers. For instance, with $n \leq 20$, a Gurobi MIQP solver with the NonConvex = 2 flag provides exact solutions against which all approximate methods are evaluated. For large instances ($n \geq 30$), exact computation becomes infeasible, and we report relative comparisons among approximate methods only.

Table 1. Test Problem Instances.

Size	Test ID	# Agents	# Tasks	Total Mass
Small	S1	3	3	9
	S2	4	4	10
	S3	5	6	17
	S4	6	5	21
Medium	M1	10	10	31
	M2	12	14	36
	M3	15	12	43
	M4	20	20	62
Large	L1	30	30	86
	L2	40	50	118
	L3	50	40	146
	L4	60	60	176
	L5	100	100	294

Solution quality on small and medium instances. Table 2 reports the percentage gap from the exact optimal solution for each method. GW_MultiInit achieves the lowest average gap of 1.78%, consistently matching or closely approximating the exact solution across all problem sizes. EGW with $\varepsilon = 0.8$ attains an average gap of 7.08% and provides the best accuracy among the EGW variants based on the sensitivity analysis in Appendix B, which identifies this value as the most effective across these scales. FGW with $\alpha = 0.7$ achieves an average gap of 13.59%, confirming that structural information (high α) is the dominant factor for CQAP quality. GA performs comparably to GW_MultiInit on the smallest instances ($n \leq 6$), where differences between methods are often modest (0.05–0.2%); however, GA's gap increases noticeably at medium scale, where GW_MultiInit maintains a more consistent advantage. We note that on individual small instances, the numerical differences between methods should be interpreted cautiously, as the margins are narrow; the advantage of GW_MultiInit is more clearly established at medium scale and above. Extended per-method objective values are reported in Appendix B (Table A2), and full computation times in Appendix B (Table A3).

Table 2. Gap from Exact Solution (%). All values report the mean percentage gap averaged over 30 independent runs.

Problem Size	GW Default	GW MultiInit	EGW 0.3	EGW 0.5	EGW 0.8	FGW 0.3	FGW 0.5	FGW 0.7	GA
3 × 3	1.45	0.00	1.45	0.00	-	51.69	0.00	0.00	0.00
4 × 4	0.00	0.00	0.03	0.00	0.00	43.76	43.76	9.67	0.00
5 × 6	0.73	0.73	0.73	0.75	1.06	89.70	14.22	14.22	0.89
6 × 5	0.32	0.00	0.32	0.32	0.37	33.88	33.88	33.88	0.32
10 × 10	70.78	9.93	24.91	25.29	16.08	29.92	18.53	15.95	70.78
12 × 14	35.40	0.00	54.29	43.49	17.97	53.33	32.28	7.79	35.40
Average	18.12	1.78	13.62	11.64	7.08	50.38	23.78	13.59	17.89

The 30-run variability analysis in Appendix B (Figure A5) reveals that GW_MultiInit exhibits consistently lower median objective values and tighter interquartile ranges than GW Default and GA across all tested instance sizes, confirming that the multi-initialization strategy improves not only average performance but also run-to-run stability. GA shows the highest variability, particularly at medium scale (n = 12–20), where its stochastic search becomes increasingly sensitive to initialization. Among GW variants, GW Default shows moderate variability that GW_MultiInit reliably reduces. The runtime growth analysis in Figure A6 further confirms that all GW-based methods scale polynomially while the exact solver grows superexponentially, becoming impractical beyond n = 14. The gap is computed as a relative error:

$$Gap\ (\%) = \frac{Approx.\ Objective - Exact\ Objective}{Exact\ Objective} \times 100 \tag{24}$$

In large instances (Table 3), GW_MultiInit consistently achieves the best objective values across all sizes from 15 × 12 to 100 × 100. The GA is excluded for the two largest instances (60 × 60, 100 × 100) due to prohibitive runtimes exceeding available computation time. FGW's relative performance degrades at larger scales, likely because feature–structure blending becomes less effective when no explicit node features are present. Table 4 documents that all GW-based methods complete within under 5 min, even at n = 100, while the GA requires hours.

Table 3. Large Problems Performance (No Exact Solution Available).

Problem Size	Mass	GW Default	GW MultiInit	EGW 0.3	EGW 0.8	FGW 0.3	FGW 0.7	GA
15 × 12	49	9657.404	8349.234	9411.960	9347.94	13,574.982	14,434.627	9657.404
20 × 20	62	8405.15	8405.15	12,115.71	12,737.47	8428.60	8471.25	8405.15
30 × 30	86	13,304.89	13,205.94	13,572.06	15,004.71	21,060.28	19,424.06	13,304.89
40 × 50	118	22,767.37	21,181.44	23,504.53	33,807.78	34,714.27	33,806.34	22,767.37
50 × 40	146	52,157.45	27,032.48	39,587.11	49,224.38	45,991.30	45,560.61	39,587.11
60 × 60	176	28,256.44	28,256.44	30,728.57	38,284.84	41,041.32	40,649.58	NA
100 × 100	294	95,867.89	88,468.94	97,709.38	128,374.23	92,070.45	89,252.85	NA

Figure 4 shows the scalability of different methods across problem sizes. GW_MultiInit and GW Default maintain solution quality close to the exact baseline up to n = 20, while EGW and FGW diverge more noticeably at medium scale. The curves begin to separate visibly at approximately n = 12–15, where entropic regularization in EGW and the feature–structure blending in FGW start introducing approximation error that single-start and multi-start GW avoid. The gap between GW_MultiInit and the next-best approximate method widens from approximately 5% at n = 10 to over 10% at n = 20, indicating that the advantage of multi-initialization becomes more pronounced as problem size

grows. Figure 5 jointly visualizes solution quality and runtime trade-offs at a large scale. GW_MultiInit achieves the best objective values across all large instances but requires substantially more computation time than single-start alternatives: at $n = 100$, GW_MultiInit takes 846 s compared to 58 s for GW Default and under 51 s for EGW and FGW variants. EGW and FGW offer practical compromises, achieving objective values within 30–50% above GW_MultiInit at a fraction of the runtime. The observed runtime growth is broadly consistent with the theoretical $O(n^3)$ complexity of the GW solver: from $n = 30$ to $n = 100$, GW_MultiInit's runtime increases by approximately 15×, close to the expected 8× for pure cubic scaling with the additional constant factor attributable to the 10 random restarts. This confirms that the empirical scaling behavior aligns with the theoretical prediction, and extrapolating from Figure 5 suggests that GW_MultiInit will exceed practical time thresholds at approximately $n = 250$–300 on comparable hardware. The accuracy–speed trade-off across all methods and problem sizes is summarized in Figure 6, where GW_MultiInit occupies the most favorable position in terms of solution quality, with an average gap of 1.78% on small and medium instances compared to 7.08% for EGW and 13.59% for FGW, while GA, despite comparable quality on small instances, becomes computationally prohibitive at large scale, requiring hours where GW_MultiInit completes within minutes. Table 5 provides a consolidated best-method summary across all instances.

Table 4. Computation Time Analysis for Large Problems (seconds).

Problem Size	GW Default	GW MultiInit	EGW 0.3	EGW 0.8	FGW 0.3	FGW 0.7	GA	Fastest Method
15 × 12	**0.013**	0.238	0.319	2.55	**0.013**	0.013	293.332	**GW Default/FGW**
20 × 20	0.059	1.024	0.480	1.18	**0.058**	0.059	1283.793	**FGW 0.3**
30 × 30	0.294	5.114	1.173	1.39	**0.289**	0.292	4650.805	**FGW 0.3**
40 × 50	1.469	24.381	5.086	1.91	**1.412**	1.420	19,168.123	**FGW 0.3**
50 × 40	1.442	24.349	2.164	2.07	**1.418**	1.429	25,276.4331	**FGW 0.3**
60 × 60	4.653	78.373	5.358	5.29	**4.590**	4.603	NA	**FGW 0.3**
100 × 100	57.83	846.45	50.21	41.48	38.29	**37.49**	NA	**FGW 0.7**

Note: Boldface indicates the best-performing value in each problem.

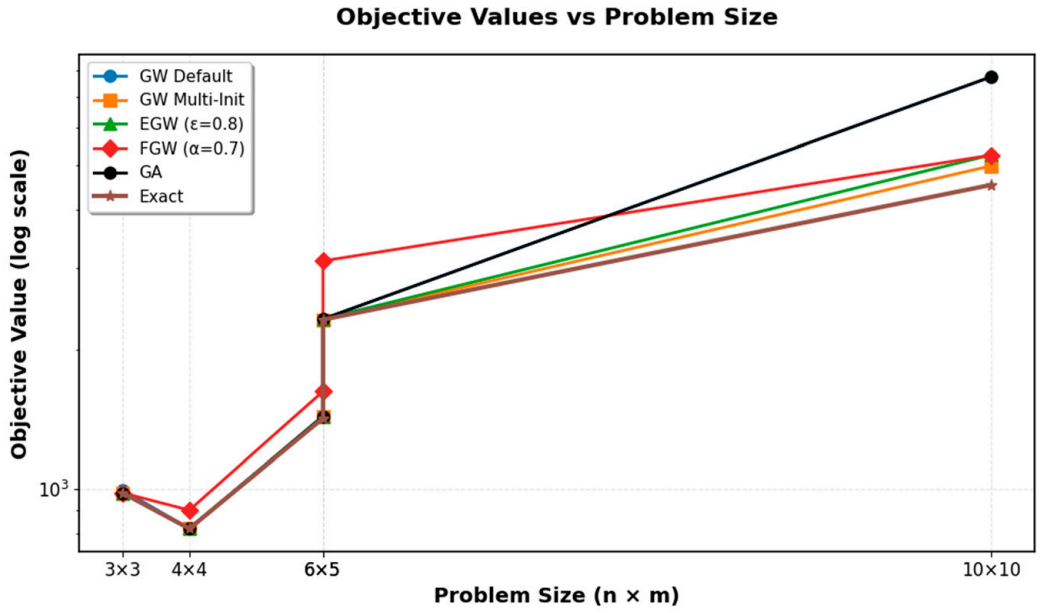

Figure 4. Runtime Scalability Analysis of GW Optimization Methods.

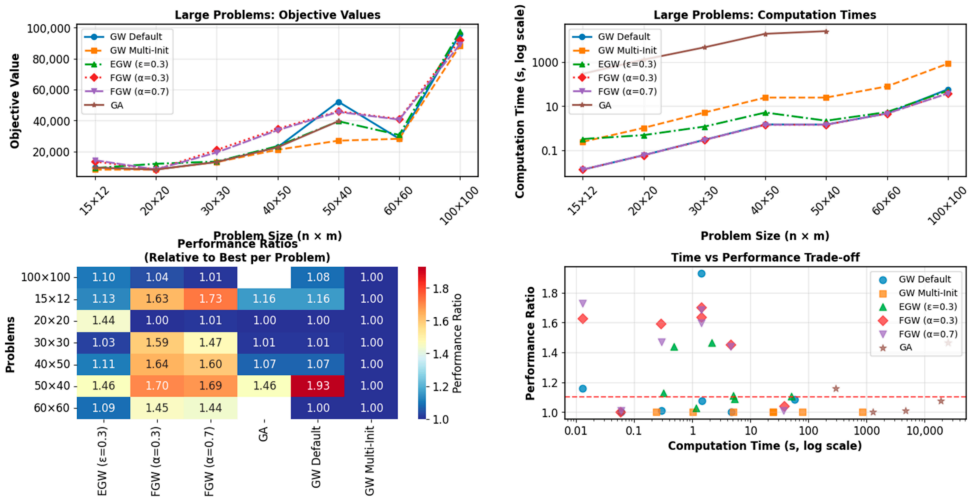

Figure 5. Solution Quality and Runtime Trade-offs for Large-Scale Problems.

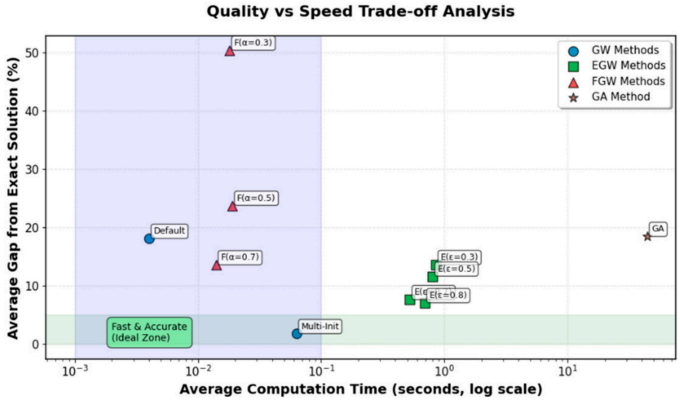

Figure 6. Trade-off between average solution quality and average computational time across methods. The vertical purple band marks the fast-computation zone (average time < 0.1s); the horizontal green band marks the high-quality zone (average gap < 5% from the exact solution); their intersection defines the ideal operating region.

Table 5. Performance Summary of CQAP Instances.

Instance (Agents × Tasks)	Best Method	Objective
3 × 3	Exact, GW_MultiInit	981.6324
4 × 4	Exact, GW_MultiInit, EGW	821.4323
5 × 6	Exact	1420.1807
6 × 5	Exact, GW_MultiInit	2317.2933
10 × 10	Exact	4525.7574
12 × 14	GW_MultiInit	5405.2937
15 × 12	GW_MultiInit	11,089.4670
20 × 20	GW_Default	8405.1519
30 × 30	GW_MultiInit	13,205.9421
40 × 50	GW_MultiInit	21,181.4378
50 × 40	GW_MultiInit	27,032.4820
60 × 60	GW_MultiInit, GW_Default	28,256.4427
100 × 100	GW_MultiInit	88,468.94

Extended results covering the full EGW ε and FGW α parameter sensitivity analyses, detailed runtime growth charts, and per-instance boxplots over 30 runs are provided in Appendix B.

5.2. Evaluation on QAPLIB Instances (Varied CQAP Mode)

To assess generalizability beyond synthetic data, we evaluate all approximate methods on 17 instances drawn from the QAPLIB benchmark library [57], transformed into CQAP instances by augmenting the standard flow and distance matrices with randomly generated integer capacities and demands. Facility capacities u_i were drawn uniformly from [2,5] and raw task demands from [1,3], then scaled so that total demand equals total capacity $(\Sigma d_j = \Sigma u_i)$, producing a balanced instance. A fixed random seed (seed = 42) was used throughout for reproducibility. The full generation procedure is available in the public repository at https://github.com/iman-ie/GW_CQAP. The flow matrix F and distance matrix D are sourced directly from QAPLIB, spanning instances from n = 12 (nug12, chr12a, tai12a) to n = 50 (tai50a) and covering three structural families: nugent (regular structure), chr (asymmetric), and tai (random dense). Since the capacity constraints produce fractional transport plans, we use the GW objective directly rather than rounding to permutations, and measure performance gaps relative to the best result found by any approximate method. The exact Gurobi solver is run with a two-hour per-instance limit; the first instance exceeding this limit and all larger instances receive NA and are excluded from the exact baseline. Complete per-instance results are provided in Appendix C.

Table 6 summarizes average performance across all 17 instances. GW MultiInit achieves the lowest average gap of 0.34% and attains the best approximate result on 15 out of 17 instances, at an average runtime of 0.085 s per instance. GW Default yields a gap of 3.26% with near-instantaneous runtime (0.003 s), and FGW with α = 0.7 reaches 6.97% while remaining fast (0.008 s). The GA matches GW Default in average gap (3.26%) but is three orders of magnitude slower, requiring up to 142 s per instance at n = 40. EGW variants show substantially larger gaps (≈80%), reflecting sensitivity to fixed ε values relative to the scale of QAPLIB cost matrices; this issue and its mitigation are discussed in Appendix C.

Table 6. Summary of QAPLIB Varied CQAP benchmark results (17 instances, n = 12–50). Gap values report single-run outcomes; GW_MultiInit's internal T= 10 trials per execution mitigate run-to-run variability by construction.

Method	Avg. Gap from Best (%)	Avg. Runtime (s)	Best on # Instances
GW MultiInit	0.34	0.085	15/17
GW Default	3.26	0.003	0/17
FGW	6.97	0.008	2/17
EGW	79.87	0.007	0/17
GA	3.26	49.9	0/17

The accuracy–runtime trade-off of GW_MultiInit warrants explicit discussion. Relative to GW Default, GW_MultiInit requires approximately 28× more computation time (0.085 s vs. 0.003 s per instance) in exchange for a gap reduction from 3.26% to 0.34%, which is a roughly 9.6× improvement in solution quality. Whether this trade-off is justified depends on the application. For time-critical deployment, GW Default offers near-instantaneous results at acceptable quality, while GW_MultiInit is preferable when solution quality is the primary criterion and runtimes below one second are sufficient. Compared to GA, GW_MultiInit delivers the same quality improvement while being three orders of magnitude faster (0.085 s vs. 49.9 s), making it the dominant choice in both dimensions against the evolutionary baseline.

Figure 7 shows the per-instance gap distribution. GW_MultiInit's gap is zero on all but two instances where GW_MultiInit does not achieve the best result are tai15a and tai30a. Both tai15a and tai30a belong to the tai (random dense) family, characterized by

large, uniformly random flow and distance matrices with no exploitable regularity. In tai15a, FGW with $\alpha = 0.5$ finds a lower objective (gap = 5.01% for GW_MultiInit), likely because the feature–structure blending in FGW aligns fortuitously with the dense random cost structure. In tai30a, FGW with $\alpha = 0.3$ marginally dominates with a difference of only 0.74%. The shared structural characteristic, dense, unstructured matrices, suggests that the GW relational matching is slightly less effective when no geometric regularity exists to exploit, which is consistent with GW's design as a structure-preserving method. Crucially, in both cases, GW_MultiInit remains the second-best method and the gaps are small in absolute magnitude, confirming that these exceptions do not undermine the overall reliability of the approach across structural families. Full per-instance results are provided in Appendix C (Table A5 and Figure A7). Figure 8 places all methods in the accuracy–runtime plane: GW_MultiInit occupies the Pareto-optimal region, combining the smallest gap with runtimes well below those of GA.

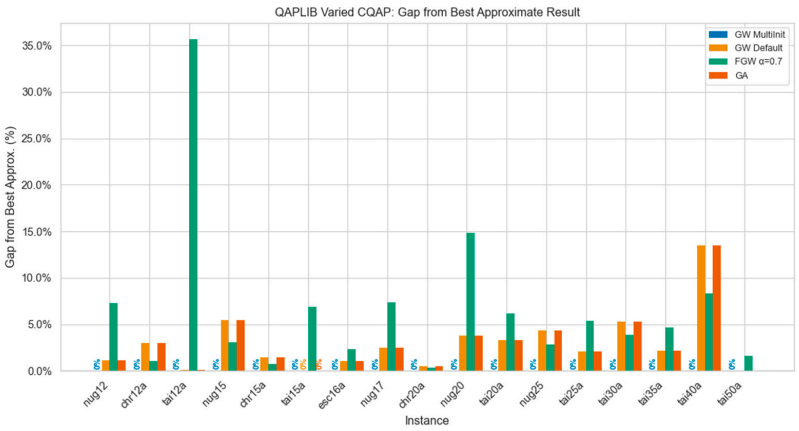

Figure 7. Percentage gap from the best approximate result per QAPLIB instance (n = 12–50), for GW_MultiInit, GW Default, FGW $\alpha = 0.7$, and GA. GW_MultiInit achieves zero gap on 15 of 17 instances. EGW is omitted due to scale mismatch (avg. gap $\approx$ 80%); see Appendix C.

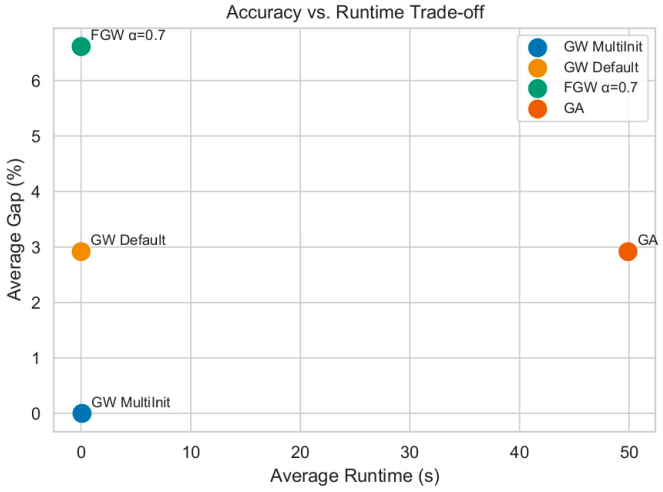

Figure 8. Average gap from best approximate result vs. average runtime (seconds) across 17 QAPLIB Varied CQAP instances. Bottom-left indicates the optimal accuracy–speed trade-off. GW_MultiInit achieves the lowest gap at runtimes three orders of magnitude faster than GA.

These findings corroborate the synthetic benchmark results of Section 5.1. GW_MultiInit generalizes robustly to established combinatorial benchmark instances with diverse structural properties, and its advantage over both GA and single-initialization GW is consistent across all tested problem families.

6. Conclusions, Limitations, and Perspectives

GW optimal transport provides a powerful framework for comparing metric-measure spaces while preserving geometric structure, directly linking to the non-convex CQAP, which is NP-hard. In this paper, we established a formal connection between CQAP and the GW framework by showing how capacity and demand constraints are encoded in the transport marginals, allowing the CQAP objective to be expressed directly as a GW discrepancy. Building on this, we introduced GW_MultiInit, a multi-initialization strategy that mitigates the non-convexity of GW optimization by exploring multiple random starting couplings and selecting the solution with the lowest objective value. This strategy is simple to implement, adds no hyperparameters beyond the number of restarts, and integrates naturally with existing GW solvers.

We investigated several GW variants and solvers, including entropy-regularized GW, FGW, and GA approaches, and compared them systematically against GW_MultiInit across both synthetic and benchmark instances. Our results show that GW_MultiInit consistently achieves higher alignment accuracy and is more robust across synthetic benchmarks and real-world datasets. It outperforms EGW, FGW, and GA while remaining computationally manageable through Sinkhorn-based regularization. Across benchmarks, approximate and regularized solvers can substantially reduce computational cost while maintaining high-quality solutions, particularly for large CQAP instances. Evaluation on 17 QAPLIB instances in the adapted CQAP setting confirms that GW_MultiInit generalizes to established combinatorial benchmarks with diverse structural properties, achieving an average gap of 0.34% from the best-known solution and the best approximate result on 15 out of 17 instances. These results strengthen the case for GW-based methods as a practical, general-purpose solver for large-scale assignment problems beyond the synthetic setting. We note that at small scale ($n \leq 6$), GA achieves a lower average gap than GW_MultiInit, as the compact feasible space allows evolutionary search to converge reliably near the optimum; this advantage reverses at medium and large scales where GW's continuous relaxation enables scalability that evolutionary methods cannot match within comparable runtime.

Limitations. Several limitations should be acknowledged. First, the cubic runtime complexity $O(n^3)$ of the GW solver becomes prohibitive for very large instances; based on the runtime trends observed in Section 5.1, computational cost is expected to exceed practical thresholds at approximately n = 200–250 on comparable hardware. Second, GW_MultiInit does not provide optimality guarantees: the returned solution is a stationary point of the non-convex GW objective, and global optimality is not assured even with multiple restarts. Furthermore, solution quality relative to the global optimum may degrade for very large or structurally complex instances where the non-convex GW landscape contains many competing local optima that multi-initialization can mitigate but not eliminate. In such cases, the gap from the global optimum may be substantially larger than the 0.34% observed on the QAPLIB benchmark, and users should treat reported gaps as lower bounds on the true optimality gap when the exact solution is unavailable. Third, capacity constraints are encoded via normalized marginals, which is exact only when total capacity equals total demand; instances with imbalanced mass require the Unbalanced GW framework, which was not explored here. Fourth, comparisons across methods were conducted without a shared time budget, so runtime figures reflect natural convergence behavior rather than a controlled efficiency comparison.

Future directions. Looking ahead, several directions could make GW methods more scalable and flexible. Some promising options are multi-marginal GW for aligning multiple metric-measure spaces simultaneously, unbalanced GW for measures with unequal mass, partial GW for missing or noisy correspondences, sliced GW for computation using fewer dimensions, and scalable GW for billion-scale graph and point cloud matching. Algorithm

improvements such as convex relaxations, spectral correspondence techniques, scalable graph-based GW learning, sampled GW, importance sparsification, and low-rank GW can also lower complexity without sacrificing accuracy. Bayesian optimization in permutation spaces—with Gaussian process surrogates and kernels such as Mallows or Merge—offers a principled way to address non-convexity in GW-related QAPs, providing globally convergent strategies that complement existing gradient methods. These developments could extend GW-based methods to larger and more complex domains, from structured data analysis and machine learning to network science and computational biology. More immediately, warm-starting GW_MultiInit with GA or greedy solutions could combine the complementary strengths of evolutionary and transport-based methods, and applying the framework to real-world logistics and telecommunication instances would provide a direct test of its operational value.

Author Contributions: Conceptualization, E.M. and F.A.; Methodology, I.S., A.C. and F.A.; Software, I.S.; Validation, I.S. and A.C.; Writing—original draft, I.S. and F.A.; Writing—review & editing, I.S., A.C., E.M. and F.A.; Visualization, I.S.; Supervision, F.A. All authors have read and agreed to the published version of the manuscript.

Funding: The work has been supported by "ULTRA OPTYMAL—Urban Logistics and sustainable TRAnsportation: OPtimization under uncertainTY and MAchine Learning", a PRIN2020 project funded by the Italian University and Research Ministry (grant number 20207C8T9M).

Data Availability Statement: The original data presented in the study are openly available in GitHub at https://github.com/iman-ie/GW_CQAP (accessed on 21 April 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Review of the Computational Gromov–Wasserstein and Multi-Initialization Strategies Literature

This section provides an overview of existing GW solvers, both unregularized and entropic approach algorithms, and is designed for GW extensions such as Fused [38] and Unbalanced [55]. Since GW is closely related to the quadratic assignment problem (QAP), which is known to be NP-hard [1], finding a global minimum is computationally expensive and theoretically intractable given the non-convexity of the objective function. Recent works have proposed relaxations to make this global search tractable: ref. [60] formulate GW as a generalized moment problem that can be truncated at the desired degree of precision, while ref. [61] propose a semi-definite relaxation solvable in polynomial time. Alternatively, ref. [39] exploit the structure of squared Euclidean costs to design a cutting plane algorithm that efficiently explores the set of admissible solutions. Despite these advances, global methods remain slow and are limited to inputs containing at most a few hundred points.

Many practical applications only require a quantification of similarity between α and β rather than an exact matching. In such cases, cheaper alternatives to the full GW problem may suffice. This category includes SLICED GW [62], QUANTIZED-GW [63], MINIBATCH-GW [64] and LINEAR-GW [65], as well as GW-inspired distances for Gaussian mixture models [66]. While these heuristics can be computed in quadratic or even linear time, they rely on heavy approximations and are less suitable for tasks requiring precise point-wise correspondences, such as shape matching.

Exact Entropic Solvers. Entropic regularization, introduced by [42,67], allows GW to be solved using the efficient Sinkhorn algorithm. The standard solver, ENTROPIC-GW, is a Projected Gradient Descent (PGD) in the KL-divergence geometry. When costs are squared norms, [41] further reduces this computation to $\mathbf{O}(\mathbf{NMD})$. Similarly, the dual solver DUAL-GW of [53] achieves quadratic complexity and is guaranteed to converge.

Accelerated Approximations. Apart from the squared norm case, solving EGW remains expensive. The algorithm complexity can still be improved thanks to several approximation strategies introduced in the last few years. SAMPLED-GW [68], SPARSE-GW [69] applying a sparsity mask to fasten both cost matrix computations and Sinkhorn iterations. LOWRANK-GW [41] approximates the input cost matrices via low-rank factorization. These three algorithms reach a quadratic running time, improving significantly the scalability of the original EGW solvers. With LINEARLOWRANK-GW [41] combine the low-rank cost reduction with a low-rank constraint on the transport plan, further reducing computation time to $\mathbf{O}(\mathbf{N}+\mathbf{M})$. Finally, ULOT (Unsupervised Learning of Optimal Transport) [70] trains neural networks to predict optimal couplings, removing the optimization burden at inference time—although its training phase keeps a cubic complexity.

Multi-Solution Discovery and Initialization Strategies

Given the non-convexity of GW and related optimization landscapes, multi-start or multi-initialization strategies have been suggested. In this context, [71] introduced Rank-Ordered Bayesian Optimization with Trust Regions (ROBOT), which seeks multiple high-performing, diverse solutions within a single optimization run. ROBOT efficiently balances exploration and diversity, discovering diverse local optima with minimal additional function evaluations. Further contributions by [72] emphasized the importance of informed initialization in BO, particularly under limited data scenarios. Their HIPE (Hyperparameter Informative Predictive Exploration) strategy improves sample efficiency by prioritizing uncertainty reduction in both prediction and hyperparameter learning.

Appendix B

Appendix B.1. Sensitivity Analysis: Number of Random Restarts (T)

To justify the choice of $T = 10$ restarts for GW_MultiInit, we conducted a sensitivity analysis over $T \in \{1, 3, 5, 7, 10, 15, 20\}$. For each value of T and each benchmark instance ($n = 4, 6, 10, 15, 20$), we performed 20 independent runs and recorded the mean objective value, its standard deviation, and wall-clock runtime.

Figure A1 shows that runtime scales approximately linearly with T across all instance sizes, increasing from an average of 0.071 s at $T = 1$ to 0.658 s at $T = 20$. Figure A2 shows that solution stability, measured by the standard deviation of the objective value across runs, improves consistently up to $T = 10$ and plateaus thereafter, with no meaningful reduction in variance observed for $T > 10$.

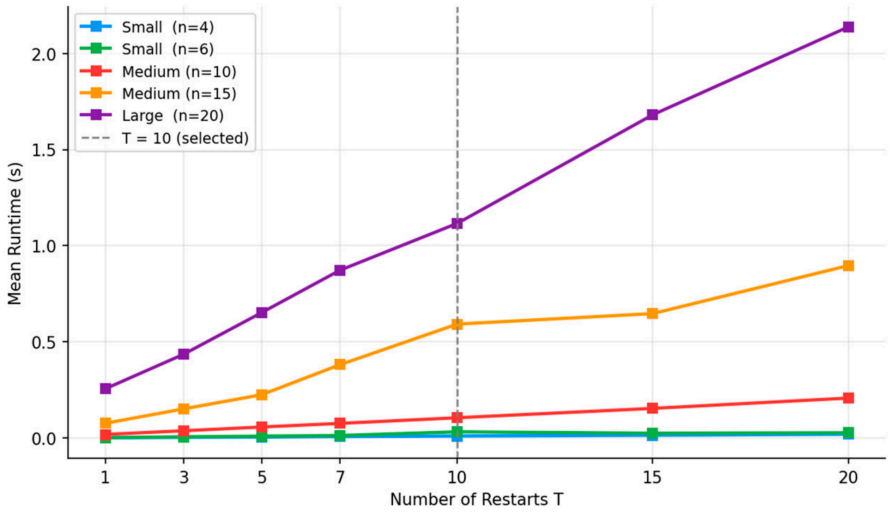

Figure A1. Sensitivity of Runtime to Number of Restarts.

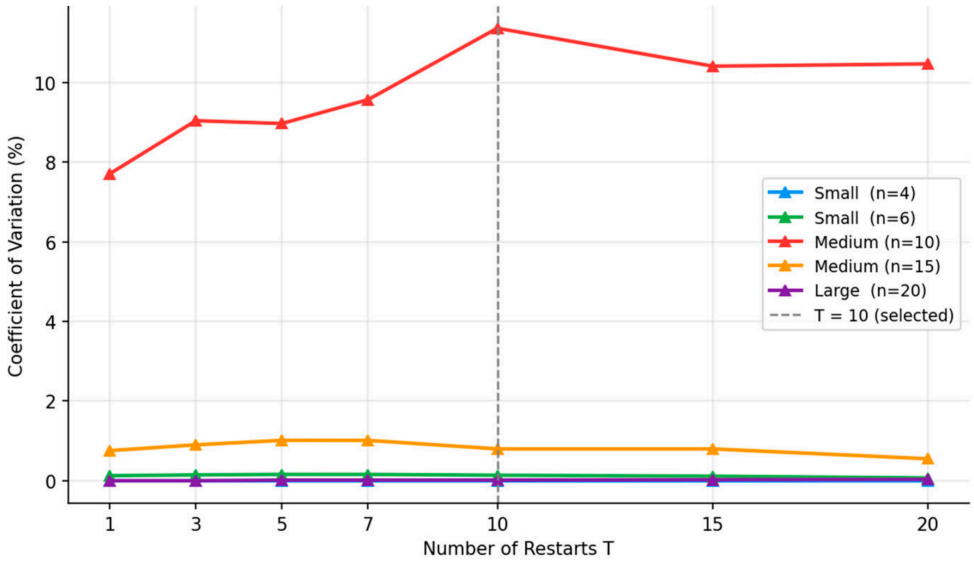

Figure A2. Run-to-Run Stability vs. Number of Restarts.

These results indicate that $T = 10$ achieves stable solution quality without incurring unnecessary computational overhead, and is therefore used in all main experiments.

Appendix B.2. Extended Synthetic CQAP Instances Results

This section of the appendix presents the full numerical results for the synthetic benchmark described in Section 5.1.

Full objective values and runtimes (small/medium instances). Table A1 reports the raw GW objective values for all methods on instances with $n \leq 20$, where exact Gurobi solutions are available. Table A2 gives the corresponding per-instance runtimes in seconds. All GW-based methods complete in under 2 s on every instance; the Exact solver requires up to 4368 s (interrupted) for the 12×14 instance, and the GA requires up to 134 s at the same scale.

Table A1. Summary of Methods Performance (Problems with Exact Solutions).

Problem Size	Mass	GW Default	GW MultiInit	EGW 0.3	EGW 0.5	EGW 0.7	FGW 0.3	FGW 0.5	FGW 0.7	GA	Exact
3×3	9	995.89	**981.63**	995.89	**981.6324**	981.6325	1489.06	**981.63**	**981.63**	**981.63**	**981.63**
4×4	10	**821.43**	**821.43**	**821.43**	821.441	821.445	1180.88	1180.88	900.83	**821.43**	**821.43**
5×6	17	1430.52	1430.52	1430.53	1430.86	1432.81	2694.07	1622.19	1622.19	1432.81	**1420.18**
6×5	21	2324.62	**2317.29**	2324.62	2324.65	2325.10	3102.40	3102.40	3102.40	2324.62	**2317.29**
10×10	31	7729.06	4975.22	5653.20	5670.36	5752.01	5879.82	5364.28	5247.65	7729.06	**4525.76**
12×14	36	7319.04	**5405.29**	8339.81	7755.81	6347.12	8287.72	7149.91	5826.33	7319.04	**5951.45** *

Note: Boldface indicates the best-performing value in each problem. * The best result until 4367.95 s after the run was interrupted.

Table A2. Computation Times (seconds) for Exact-Solvable Problems.

Problem Size	GW Default	GW MultiInit	EGW 0.3	EGW 0.5	EGW 0.7	FGW 0.3	FGW 0.5	FGW 0.7	GA	Exact
3×3	**0.001**	0.010	0.083	0.058	0.041	**0.000**	**0.001**	**0.000**	2.265	0.034
4×4	**0.001**	0.012	0.151	0.148	0.149	**0.000**	**0.001**	**0.000**	3.350	0.065
5×6	**0.001**	0.022	0.146	0.224	0.280	**0.001**	**0.001**	**0.001**	11.239	0.801
6×5	**0.001**	0.021	0.065	0.056	0.053	**0.000**	**0.001**	**0.000**	40.445	1.254
10×10	0.005	0.084	0.304	0.311	1.179	**0.004**	**0.004**	**0.004**	77.509	732.42
12×14	0.018	0.229	2.028	2.036	0.510	0.037	0.039	**0.015**	134.691	4367.95

Note: Boldface indicates the best-performing value in each problem.

EGW ε parameter sensitivity. Table A3 reports the gap from exact and average runtime as a function of the entropic regularization parameter $\varepsilon \in \{0.0, 0.3, 0.5, 0.7, 0.8, 0.9, 1.0\}$. Larger ε values reduce the average gap but increase runtime marginally; $\varepsilon = 0.8$ achieves the best accuracy–speed balance (avg. gap 7.08%, avg. runtime 0.69 s) and is recommended as a default. Figure A3 visualizes this trade-off curve.

Table A3. EGW Epsilon Parameter Analysis.

Problem Size	$\varepsilon = 1.0$	$\varepsilon = 0.9$	$\varepsilon = 0.8$	$\varepsilon = 0.7$	$\varepsilon = 0.6$	$\varepsilon = 0.5$	$\varepsilon = 0.3$
Gap from Exact (%)							
3×3	-	-	-	0.00	0.00	0.00	1.45
4×4	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5×6	0.22	0.14	1.06	0.89	0.80	0.75	0.73
6×5	0.49	0.42	0.37	0.34	0.32	0.32	0.32
10×10	16.81	16.45	16.08	27.09	26.45	25.29	24.91
12×14	19.60	18.73	17.97	17.42	44.28	43.49	54.29
Average Gap	7.43	7.12	7.08	7.62	11.97	11.64	13.62
Computation Time (s)							
Average	0.63	0.64	0.69	0.52	0.79	0.80	0.85

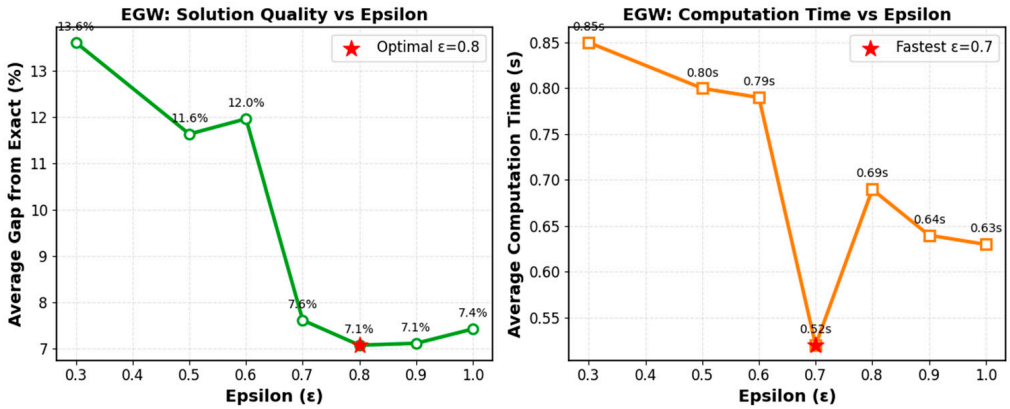

Figure A3. EGW Epsilon Parameter Optimization: Balancing Accuracy and Efficiency.

FGW α parameter sensitivity. Table A4 reports the analogous analysis for the α blending parameter in FGW $\in \{0.0, 0.3, 0.5, 0.7\}$. As shown in Figure A4, $\alpha = 0.7$ achieves the lowest average gap (13.59%) with negligible runtime overhead (avg. < 0.02 s), confirming that structural information dominates for CQAP. Values near $\alpha = 0$ produce large gaps (>50%), underscoring the inadequacy of pure feature-based transport for this problem class.

Table A4. FGW Alpha Parameter Analysis.

Problem Size	$\alpha = 0.0$	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$
Gap from Exact (%)				
3×3	51.69	51.69	0.00	0.00
4×4	43.76	43.76	43.76	9.67
5×6	89.42	89.70	14.22	14.22
6×5	32.14	33.88	33.88	33.88
10×10	33.97	29.92	18.53	15.95
12×14	66.80	53.33	32.28	7.79
Average Gap	52.96	50.38	23.78	13.59
Computation Time (s)				
Average	0.014	0.018	0.019	0.014

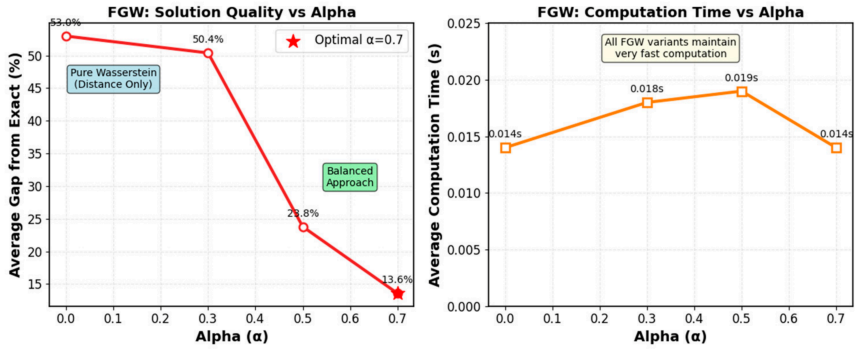

Figure A4. FGW Performance Analysis: Optimal Alpha Parameter Selection.

Variability across runs. Figure A5 presents boxplots over 30 independent runs for each method on instances ranging from 15 × 12 to 40 × 50. GW MultiInit shows consistently lower median objective values and tighter interquartile ranges than GW Default and GA, demonstrating both accuracy and reliability of the multi-initialization strategy.

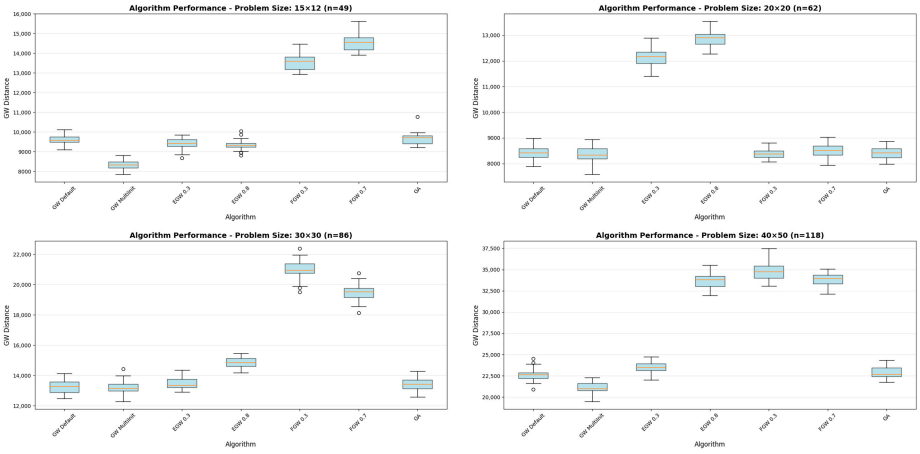

Figure A5. Algorithm Performance Comparison Across Problem Sizes.

Exact vs. approximate runtime growth. Figure A6 plots computational time as a function of problem size for all methods including the Exact solver. The exact solver's runtime grows superexponentially (from 0.034 s at $n = 3$ to >4000 s at $n = 14$), while all GW variants scale polynomially, with GW Default and FGW remaining the fastest and GW_MultiInit incurring a constant factor overhead proportional to the number of restarts.

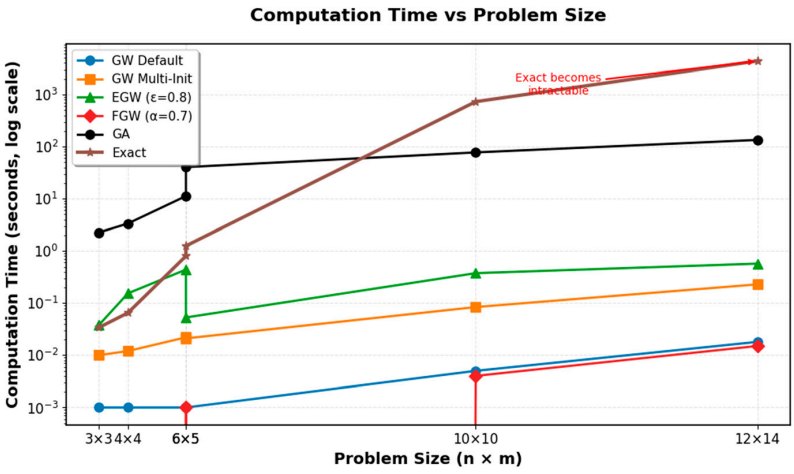

Figure A6. Computational Time Growth Comparison: Exact vs. Approximate Methods.

Appendix C. Extended QAPLIB Varied CQAP Benchmark Discussion

This appendix provides extended numerical results and analysis for the QAPLIB Varied CQAP benchmark evaluation described in Section 5.2.

Per-instance objective values. Table A5 reports the full objective value for each method on all 17 QAPLIB instances. The column "Best Approx." records the minimum objective across all approximate methods and serves as the gap reference. Bold entries indicate the best result for each instance.

Table A5. Per-instance results on QAPLIB Varied CQAP benchmark.

Instance	n	GW MultiInit	GW Default	FGW α=0.7	GA	Best Approx.	MI Gap (%)
nug12	12	8476.00	8573.00	9092.10	8573.00	8476.00	0
chr12a	12	3,662,833	3,770,811	3,701,336	3,770,811	3,662,833	0
tai12a	12	1,339,575	1,340,668	1,816,923	1,340,668	1,339,575	0
nug15	15	16,284.20	17,173.00	16,781.60	17,173.00	16,284.20	0
chr15a	15	7,202,157	7,305,004	7,254,557	7,305,004	7,202,157	0
tai15a	15	2,190,497	2,190,497	2,341,117	2,190,497	2,086,017	5.01
esc16a	16	5393.60	5449.50	5520.20	5449.50	5393.60	0
nug17	17	23,624.20	24,212.80	25,372.00	24,212.80	23,624.20	0
chr20a	20	1,716,165	1,724,623	1,722,316	1,724,623	1,716,165	0
nug20	20	32,618.20	33,837.60	37,469.30	33,837.60	32,618.20	0
tai20a	20	4,521,811	4,669,769	4,799,767	4,669,769	4,521,811	0
nug25	25	50,907.70	53,112.40	52,361.00	53,112.40	50,907.70	0
tai25a	25	7,503,553	7,662,467	7,905,838	7,662,467	7,503,553	0
tai30a	30	11,061,937	11,649,887	11,487,442	11,649,887	10,980,196	0.74
tai35a	35	15,653,117	15,998,512	16,386,739	15,998,512	15,653,117	0
tai40a	40	19,359,212	21,976,129	20,970,619	21,976,129	19,359,212	0
tai50a	50	34,921,914	34,931,901	35,486,759	34,931,901	34,921,914	0

Scaling behavior. Figure A7 plots the normalized objective (method value/best approx.) as a function of problem size n, averaged over instances of equal size. GW_MultiInit remains at 1.00 for all sizes except tai15a and tai30a, where the deviation is negligible. GW Default and GA diverge from 1.00 by up to 14% at n = 17 (nug17) but recover at larger sizes, suggesting instance-specific sensitivity rather than a systematic scaling problem.

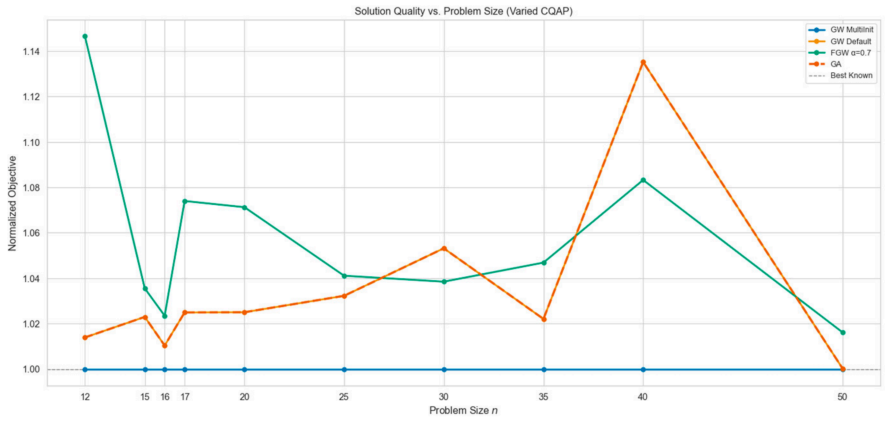

Figure A7. Normalized CQAP objective (method value/best approximate) vs. problem size n, averaged over instances of equal size. A value of 1.00 (dashed line) denotes the best known result. GW_MultiInit remains at or below 1.01 across all sizes (n = 12–50).

Figure A8 is the most natural addition—a per-instance runtime chart split into two panels: GW-based methods on the left, GA on the right (separate scales due to the orders-of-

magnitude difference). This directly parallels Table A3 in Appendix B and gives readers a visual companion to the per-instance data in Table A5.

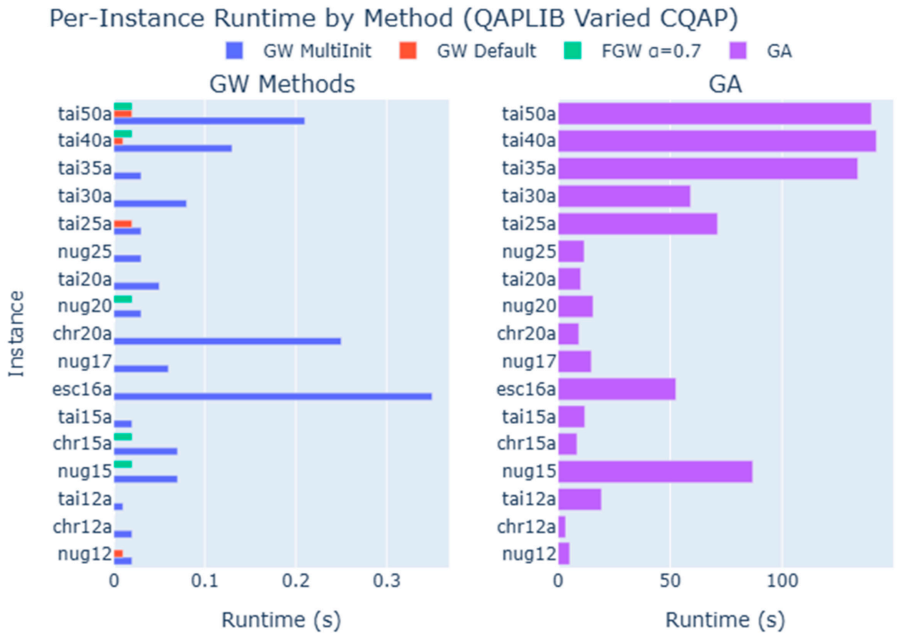

Figure A8. Per-instance runtime (seconds) for GW MultiInit, GW Default, and FGW $\alpha = 0.7$ (**left** panel) and GA (**right** panel) across 17 QAPLIB Varied CQAP instances, sorted by problem size n. Panels use separate linear scales due to the large runtime disparity between GW-based methods ($\leq$0.35 s) and GA (up to 142 s). GW MultiInit's overhead relative to GW Default reflects the cost of 10 random restarts.

Note on EGW performance. EGW variants produce objective values 1.7×–7.5× larger than GW MultiInit across all instances, with the largest gaps observed for tai-family instances (e.g., EGW $\varepsilon = 0.3$ yields 10,064,591 vs. 4,521,811 for GW MultiInit on tai20a). This behavior arises from scale mismatch: QAPLIB cost matrices in the tai-family have entries of order 10^5–10^6, causing a fixed $\varepsilon = 0.3$ to impose excessive entropic smoothing and prevent the solver from resolving sharp assignment boundaries. Normalizing ε by the Frobenius norm of the cost matrices—i.e., setting $\widetilde{\varepsilon} = \varepsilon \cdot \frac{\|C_X\|_F}{n}$—is expected to substantially reduce EGW's gap and is a practical recommendation for practitioners applying EGW to instances outside the synthetic range explored in Section 5.1.

Benchmark configuration. All QAPLIB experiments use the default configuration: 10 restarts for GW MultiInit, $\varepsilon \in \{0.3, 0.5, 0.8\}$ for EGW, $\alpha \in \{0.3, 0.5, 0.7\}$ for FGW. The tai10a instance was unavailable for automated download and is excluded from the evaluation. The benchmark runner is included in the public repository at https://github.com/iman-ie/GW_CQAP.

References

1. Burkard, R.; Dell'Amico, M.; Martello, S. Assignment problems: Revised reprint. In *Assignment Problems*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2012; pp. 327–393. [CrossRef]
2. Pentico, D.W. Assignment problems: A golden anniversary survey. *Eur. J. Oper. Res.* **2007**, *176*, 774–793. [CrossRef]
3. Seyedi, I.; Mirzazadeh, S.; Malekidaronkolaei, A.R.E.F.; Mukhtar, M.; Sahran, S. An inventory model with reworking and setup time to consider effect of inflation and time value of money. *J. Eng. Sci. Technol.* **2016**, *11*, 416–430.
4. Munkres, J. Algorithms for the Assignment and Transportation Problems. *J. Soc. Ind. Appl. Math.* **1957**, *5*, 32–38. [CrossRef]
5. Koopmans, T.C.; Beckmann, M. Assignment problems and the location of economic activities. *Econom. J. Econom. Soc.* **1957**, *25*, 53–76. [CrossRef]
6. Sahni, S.; Gonzalez, T. P-Complete Approximation Problems. *J. ACM* **1976**, *23*, 555–565. [CrossRef]

7. Chaovalitwongse, W.A.; Androulakis, I.P.; Pardalos, P.M. Quadratic Integer Programming: Complexity and Equivalent Forms. In *Encyclopedia of Optimization*; Pardalos, P.M., Prokopyev, O.A., Eds.; Springer International Publishing: Cham, Switzerland, 2024; pp. 1–8. [CrossRef]
8. Bertsekas, D. *Network Optimization: Continuous and Discrete Models*; Athena Scientific: Belmont, MA, USA, 1998; Volume 8.
9. Ahmed, Z.H. A hybrid algorithm combining lexisearch and genetic algorithms for the quadratic assignment problem. *Cogent Eng.* **2018**, *5*, 1423743. [CrossRef]
10. Villani, C. The Wasserstein distances. In *Optimal Transport*; in Grundlehren der mathematischen Wissenschaften; Springer: Berlin/Heidelberg, Germany, 2009; Volume 338, pp. 93–111. [CrossRef]
11. Mémoli, F. Gromov–Wasserstein Distances and the Metric Approach to Object Matching. *Found. Comput. Math.* **2011**, *11*, 417–487. [CrossRef]
12. Kravtsova, N. The NP-hardness of the Gromov-Wasserstein distance. *arXiv* **2025**, arXiv:2408.06525. [CrossRef]
13. Burkard, R.E.; Çela, E.; Pardalos, P.M.; Pitsoulis, L.S. The Quadratic Assignment Problem. In *Handbook of Combinatorial Optimization*; Du, D.-Z., Pardalos, P.M., Eds.; Springer: Boston, MA, USA, 1998; pp. 1713–1809. [CrossRef]
14. Çela, E. The Quadratic Assignment Problem. In *Combinatorial Optimization*; Springer: Boston, MA, USA, 1998; Volume 1. [CrossRef]
15. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
16. Hahn, P.; Grant, T.; Hall, N. A branch-and-bound algorithm for the quadratic assignment problem based on the Hungarian method. *Eur. J. Oper. Res.* **1998**, *108*, 629–640. [CrossRef]
17. Zhang, H.; Beltran-Royo, C.; Ma, L. Solving the quadratic assignment problem by means of general purpose mixed integer linear programming solvers. *Ann. Oper. Res.* **2013**, *207*, 261–278. [CrossRef]
18. Loiola, E.M.; de Abreu, N.M.M.; Boaventura-Netto, P.O.; Hahn, P.; Querido, T. A survey for the quadratic assignment problem. *Eur. J. Oper. Res.* **2007**, *176*, 657–690. [CrossRef]
19. Pardalos, P.M.; Vavasis, S.A. Quadratic programming with one negative eigenvalue is NP-hard. *J. Glob. Optim.* **1991**, *1*, 15–22. [CrossRef]
20. Ahuja, R.K.; Orlin, J.B.; Tiwari, A. A greedy genetic algorithm for the quadratic assignment problem. *Comput. Oper. Res.* **2000**, *27*, 917–934. [CrossRef]
21. Said, G.A.E.N.A.; Mahmoud, A.M.; El-Horbaty, E.S.M. A Comparative Study of Meta-heuristic Algorithms for Solving Quadratic Assignment Problem. *Int. J. Adv. Comput. Sci. Appl.* **2014**, *5*, 4863. [CrossRef]
22. Hussin, M.S.; Stützle, T. Tabu search vs. simulated annealing as a function of the size of quadratic assignment problem instances. *Comput. Oper. Res.* **2014**, *43*, 286–291. [CrossRef]
23. James, T.; Rego, C.; Glover, F. A cooperative parallel tabu search algorithm for the quadratic assignment problem. *Eur. J. Oper. Res.* **2009**, *195*, 810–826. [CrossRef]
24. Puris, A.; Bello, R.; Herrera, F. Analysis of the efficacy of a Two-Stage methodology for ant colony optimization: Case of study with TSP and QAP. *Expert Syst. Appl.* **2010**, *37*, 5443–5453. [CrossRef]
25. Lu, Y.; Chen, S.; Zhang, X.; Pan, X.; Gang, Y.; Wang, C. A quantum-enhanced heuristic algorithm for optimizing aircraft landing problems in low-altitude intelligent transportation systems. *Sci. Rep.* **2025**, *15*, 21606. [CrossRef]
26. Wang, Z.; Wang, X. Quantum-Classical Hybrid Genetic Evolutionary Algorithm for Topology Optimization of Continuum Structures. *Int. J. Numer. Methods Eng.* **2025**, *126*, e70073. [CrossRef]
27. Coello, C.A.C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Comput. Methods Appl. Mech. Eng.* **2002**, *191*, 1245–1287. [CrossRef]
28. de Klerk, E.; Sotirov, R.; Truetsch, U. A New Semidefinite Programming Relaxation for the Quadratic Assignment Problem and Its Computational Perspectives. *Inf. J. Comput.* **2015**, *27*, 378–391. [CrossRef]
29. Zhang, H.; Beltran-Royo, C.; Wang, B.; Ma, L.; Zhang, Z. Solution to the quadratic assignment problem using semi-Lagrangian relaxation. *J. Syst. Eng. Electron.* **2016**, *27*, 1063–1072. [CrossRef]
30. Monge, G. *Mémoire sur le Calcul Intégral des Équations aux Différences Partielles*; Imprimerie Royale: Paris, France, 1784.
31. Kantorovich, L. On the transfer of masses. In *Doklady Akademii Nauk*; Russian Academy of Sciences: Moscow, Russia, 1942; Available online: https://cir.nii.ac.jp/crid/1370565168575910170 (accessed on 17 July 2025). (In Russian)
32. Rachev, S.T.; Rüschendorf, L. Duality Theory for Mass Transfer Problems. In *Probability and Its Applications*; Springer: New York, NY, USA, 2012; pp. 161–273. [CrossRef]
33. Montesuma, E.F.; Mboula, F.M.N.; Souloumiac, A. Recent advances in optimal transport for machine learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *47*, 1161–1180. [CrossRef]
34. Khamis, A.; Tsuchida, R.; Tarek, M.; Rolland, V.; Petersson, L. Scalable optimal transport methods in machine learning: A contemporary survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**. [CrossRef]
35. Candelieri, A.; Ponti, A.; Archetti, F. Wasserstein enabled Bayesian optimization of composite functions. *J. Ambient Intell. Humaniz. Comput.* **2023**, *14*, 11263–11271. [CrossRef]

36. Kaňková, V. Stochastic optimization problems with nonlinear dependence on a probability measure via the Wasserstein metric. *J. Glob. Optim.* **2024**, *90*, 593–617. [CrossRef]
37. Tao, Y.; Chen, C.; Jiang, S.; Wang, Q. Strong Group Fair Classification via Optimal Transport and Mixed-Integer Linear Programming. *J. Glob. Optim.* **2025**, 1–22. [CrossRef]
38. Vayer, T.; Chapel, L.; Flamary, R.; Tavenard, R.; Courty, N. Fused Gromov-Wasserstein Distance for Structured Objects. *Algorithms* **2020**, *13*, 212. [CrossRef]
39. Karlsson, J.; Kronqvist, J.; Ryner, M. Globally solving the Gromov-Wasserstein problem for point clouds in low dimensional Euclidean spaces. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 7930–7946.
40. Takeda, S.; Akagi, Y. Gromov-Wasserstein Problem with Cyclic Symmetry. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, TN, USA, 11–15 June 2025; pp. 21011–21020. Available online: http://openaccess.thecvf.com/content/CVPR2025/html/Takeda_Gromov-Wasserstein_Problem_with_Cyclic_Symmetry_CVPR_2025_paper.html (accessed on 7 May 2026).
41. Cetbon, M.; Peyré, G.; Cuturi, M. Linear-time gromov wasserstein distances using low rank couplings and costs. In *International Conference on Machine Learning*; PMLR: London, UK, 2022; pp. 19347–19365. Available online: https://proceedings.mlr.press/v162/scetbon22b (accessed on 21 August 2025).
42. Peyré, G.; Cuturi, M.; Solomon, J. Gromov-wasserstein averaging of kernel and distance matrices. In *International Conference on Machine Learning*; PMLR: London, UK, 2016; pp. 2664–2672.
43. Seyedi, I.; Candelieri, A.; Messina, E.; Archetti, F. Wasserstein Distributionally Robust Optimization for Chance Constrained Facility Location Under Uncertain Demand. *Mathematics* **2025**, *13*, 2144. [CrossRef]
44. Peyré, G.; Cuturi, M. Computational optimal transport: With applications to data science. *Found. Trends® Mach. Learn.* **2019**, *11*, 355–607. [CrossRef]
45. Cuturi, M.; Doucet, A. Fast computation of Wasserstein barycenters. In *International Conference on Machine Learning*; PMLR: London, UK, 2014; pp. 685–693. Available online: https://proceedings.mlr.press/v32/cuturi14.html (accessed on 9 June 2025).
46. Candelieri, A.; Ponti, A.; Giordani, I.; Archetti, F. On the use of Wasserstein distance in the distributional analysis of human decision making under uncertainty. *Ann. Math. Artif. Intell.* **2023**, *91*, 217–238. [CrossRef]
47. Shan, Z.; Lu, X.; Yang, J. Distributionally robust resource allocation using Wasserstein distance. *J. Glob. Optim.* **2025**, 1–16. [CrossRef]
48. Seyedi, I.; Candelieri, A.; Archetti, F. Distributionally Robust Bayesian Optimization via Sinkhorn-Based Wasserstein Barycenter. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 90. [CrossRef]
49. Candelieri, A.; Ponti, A.; Archetti, F. Gaussian Process regression over discrete probability measures: On the non-stationarity relation between Euclidean and Wasserstein Squared Exponential Kernels. *J. Glob. Optim.* **2025**, *92*, 253–278. [CrossRef]
50. Séjourné, T.; Peyré, G.; Vialard, F.X. Unbalanced optimal transport, from theory to numerics. *Handb. Numer. Anal.* **2023**, *24*, 407–471. [CrossRef]
51. Mémoli, F.; Needham, T. Distance distributions and inverse problems for metric measure spaces. *Stud. Appl. Math.* **2022**, *149*, 943–1001. [CrossRef]
52. Zhang, W.; Wang, Z.; Fan, J.; Wu, H.; Zhang, Y. Fast Gradient Computation for Gromov-Wasserstein Distance. *arXiv* **2024**, arXiv:2404.08970. [CrossRef]
53. Rioux, G.; Goldfeld, Z.; Kato, K. Entropic gromov-wasserstein distances: Stability and algorithms. *J. Mach. Learn. Res.* **2024**, *25*, 1–52.
54. Seyedi, I.; Candelieri, A.; Archetti, F. Fused Unbalanced Gromov–Wasserstein-Based Network Distributional Resilience Analysis for Critical Infrastructure Assessment. *Mathematics* **2026**, *14*, 417. [CrossRef]
55. Séjourné, T.; Vialard, F.X.; Peyré, G. The unbalanced gromov wasserstein distance: Conic formulation and relaxation. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 8766–8779.
56. Flamary, R.; Courty, N.; Gramfort, A.; Alaya, M.Z.; Redko, I.; Chapelle, O.; De Lara, N.; Ferreira, S.; Valentin, B.; Seguy, B.; et al. Pot: Python optimal transport. *J. Mach. Learn. Res.* **2021**, *22*, 1–8.
57. Burkard, R.E.; Çela, E.; Karisch, S.E.; Rendl, F.; Anjos, M.; Hahn, P. *QAPLIB—A Quadratic Assignment Problem Library—Problem Instances and Solutions, [Dataset]*; University of Edinburgh: Edinburgh, UK; Computational Optimization Research At Lehigh: Bethlehem, PA, USA, 2022. [CrossRef]
58. Seyedi, I.; Maleki-Daronkolaei, A. Solving a two-stage assembly flowshop scheduling problem to minimize the mean tardiness and earliness penalties by three meta-heuristics. *Casp. J. Appl. Sci. Res.* **2013**, *2*, 67–78.
59. Seyedi, I.; Maleki-Daronkolaei, A.; Kalashi, F. Tabu search and simulated annealing for new three-stage assembly flow shop scheduling with blocking. *Interdiscip. J. Contemp. Res. Usiness* **2012**, *4*, 394–402.
60. Mula, O.; Nouy, A. Moment-SoS methods for optimal transport problems. *Numer. Math.* **2024**, *156*, 1541–1578. [CrossRef]
61. Chen, J.; Koh, S.; Nguyen, B.; Soh, Y. Semidefinite relaxations of the Gromov-Wasserstein distance. *Adv. Neural Inf. Process. Syst.* **2024**, *37*, 69814–69839.

62. Titouan, V.; Flamary, R.; Courty, N.; Tavenard, R.; Chapel, L. Sliced gromov-wasserstein. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 14726–14736.
63. Chowdhury, S.; Miller, D.; Needham, T. Quantized Gromov-Wasserstein. In *Machine Learning and Knowledge Discovery in Databases. Research Track*; Oliver, N., Pérez-Cruz, F., Kramer, S., Read, J., Lozano, J.A., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2021; Volume 12977, pp. 811–827. [CrossRef]
64. Fatras, K.; Zine, Y.; Majewski, S.; Flamary, R.; Gribonval, R.; Courty, N. Minibatch optimal transport distances; analysis and applications. *arXiv* **2021**, arXiv:2101.01792. [CrossRef]
65. Beier, F.; Beinert, R.; Steidl, G. On a Linear Gromov–Wasserstein Distance. *IEEE Trans. Image Process.* **2022**, *31*, 7292–7305. [CrossRef]
66. Salmona, A.; Delon, J.; Desolneux, A. Gromov-Wasserstein-like Distances in the Gaussian Mixture Models Space. *arXiv* **2024**, arXiv:2310.11256.
67. Olomon, J.; Peyré, G.; Kim, V.G.; Sra, S. Entropic metric alignment for correspondence problems. *ACM Trans. Graph.* **2016**, *35*, 1–13. [CrossRef]
68. Kerdoncuff, T.; Emonet, R.; Sebban, M. Sampled Gromov Wasserstein. *Mach. Learn.* **2021**, *110*, 2151–2186. [CrossRef]
69. Li, M.; Yu, J.; Xu, H.; Meng, C. Efficient Approximation of Gromov-Wasserstein Distance Using Importance Sparsification. *J. Comput. Graph. Stat.* **2023**, *32*, 1512–1523. [CrossRef]
70. Mazelet, S.; Flamary, R.; Thirion, B. Unsupervised Learning for Optimal Transport plan prediction between unbalanced graphs. *arXiv* **2025**, arXiv:2506.12025. [CrossRef]
71. Maus, N.; Wu, K.; Eriksson, D.; Gardner, J. Discovering Many Diverse Solutions with Bayesian Optimization. *arXiv* **2023**, arXiv:2210.10953. [CrossRef]
72. Hvarfner, C.; Eriksson, D.; Bakshy, E.; Balandat, M. Informed Initialization for Bayesian Optimization and Active Learning. *arXiv* **2025**, arXiv:2510.23681. [CrossRef]