Received: xxx **Accepted:** xxx **Published:** xxx.
DOI: 10.30473/coam.2026.78574.1440
Volume xxx, Issue xxx, p.p. 1-30: xxx
Research Article

Open Access

Control and Optimization in Applied Mathematics - COAM

A Monotone Move-Based Heuristic for k-Partitioning Graphs with Fuzzy Edge and Vertex Weights

Fadhil Alsabbagh[1], **Mostafa Tavakoli**[1] ✉, **Ahmad Abouyee Mehrizi**[1], **Khatere Ghorbani-Moghadam**[2], **Omid Solaymani Fard**[1]

[1]Department of Applied Mathematics, Faculty of Mathematical Sciences, Ferdowsi University of Mashhad, P. O. Box 1159, Mashhad 91775, Iran,
[2]Mosaheb Institute of Mathematics, Kharazmi University, Tehran, Iran

✉ **Correspondence:**
Mostafa Tavakoli
E-mail:
m_tavakoli@um.ac.ir

How to Cite
Alsabbagh, F., Tavakoli, M., Abouyee Mehrizi, A., Ghorbani-Moghadam, Kh., Solaymani Fard, O. (2027). "A monotone move-based heuristic for k-partitioning graphs with fuzzy edge and vertex weights". *Control and Optimization in Applied Mathematics*, X(x), 1–30. https://doi.org/10.30473/coam.2026.78574.1440

Abstract. This paper considers a fuzzy graph k-partitioning problem in which both edge and vertex weights are modeled by trapezoidal fuzzy numbers. To avoid nonlinear operations that are not closed within the trapezoidal family, the fuzzy inputs are first mapped to scalar values using Yager's linear ranking function, and pairwise vertex-weight dissimilarities are then formed in the ranked space. On this basis, we develop a randomized exchange heuristic that preserves the prescribed partition sizes and evaluates each candidate move through an explicit gain formula. For a symmetric ranked matrix with a zero diagonal, the auxiliary matrix can be updated in closed form, which gives a direct monotonicity result and reduces the cost of an attempted exchange to $O(n)$ after $O(n^2)$ preprocessing. The method is evaluated over 15 independent runs on instances with $n = 20, 1000$, and 1500 vertices and is compared with a genetic algorithm (GA), Tabu search (TS), sequential least squares programming (SLSQP), and a Kernighan–Lin/Fiduccia–Mattheyses (KL/FM)-type baseline. On the two larger instances, the proposed method attains the highest mean objective value and the lowest mean runtime among the five methods; the paired comparisons against GA, TS, and SLSQP remain significant after Holm–Bonferroni correction and are supported by two-sided Wilcoxon signed-rank tests. On the smallest instance ($n = 20$), TS with tenure 15 and full cross-partition enumeration ($N_{\mathrm{TS}} = 190$ candidate pairs) attains a higher mean objective than the proposed method. A supplementary sensitivity study examines the main GA and TS parameters and the trade-off parameter θ.

Keywords. Graph partitioning, Balanced k-partitioning, Fuzzy optimization, Trapezoidal fuzzy numbers, Ranking function, Local search heuristic.

MSC. 90C27; 90C70.

https://mathco.journals.pnu.ac.ir

©2026 by the authors. Lisensee PNU, Tehran, Iran. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution 4.0 International (CC BY4.0) (http:/creativecommons.org/licenses/by/4.0)

Corrected Proofs

1 Introduction

Graph partitioning asks how the vertices of a graph can be divided into disjoint groups while optimizing a cut- or within-group criterion. In balanced formulations, the groups must also satisfy prescribed size or weight constraints. Variants of this problem arise in many fields, including circuit design, sparse matrix computations, parallel systems, image analysis, and clustering [2, 3, 11, 12, 29]. In general, graph partitioning is NP-complete [10], so practical algorithms are typically either exact or based on local search and metaheuristics [6, 14, 15].

Graph partitioning has an extensive algorithmic history, beginning with the Kernighan–Lin approach [18]. Subsequent research introduced genetic algorithms (GA), tabu search (TS), multilevel approaches [17], and even a continuous formulation [14, 15]. Recent studies still focus on improving local search and on multilevel approaches. Maas et al. [21], for example, applied a parallel local search to unconstrained irregular graph partitioning. Lu et al. [20] developed a multilevel iterated tabu search method for multi-constraint graph partitioning. Other studies include recursive partitioning of a multi-constraint graph [26] and a balanced partitioning method for big graphs based on coarsening, partitioning, and refinement steps [7]. Finally, in large-scale applications such as political districting, partitioning methods that scale with problem size become particularly important [5].

Two factors make the fuzzy k-partitioning problem difficult to handle in practice. First, partitioning instances are typically large, with up to millions of vertices, so even local searches costing $O(n^2)$ per pass may take an unacceptable amount of time when hundreds of such passes are needed for convergence. Second, many partitioning parameters, including vertex and edge weights, are uncertain. In communication networks, the strengths of connections between nodes are inherently noisy; similarity measures between points in a cluster may be estimated rather than known exactly; and the importance of a node (often represented by a PageRank score) may only be approximated. These issues have been examined individually. Techniques for scalable local-search and multilevel algorithms presume known parameters, whereas, to our knowledge, existing fuzzy-weight partitioning approaches [4] rely on population-based or continuous solvers whose per-iteration cost is not designed for large graphs. Addressing both issues calls for a combination of the two types of techniques: an approach that handles fuzzy inputs and has a per-move cost much lower than the square of the instance size. The present work addresses this requirement.

Most classical models assume that edge and vertex information is known precisely. In many networks, however, quantities such as connection strength, similarity, or vertex importance are uncertain or only approximately known. Fuzzy numbers [30] offer a natural way to represent this uncertainty. Trapezoidal fuzzy numbers are particularly convenient because they describe a core interval together with left and right spreads. Recent work in fuzzy optimization has continued to investigate ranking procedures, robustness, and numerical solution methods.

Corrected proofs

Inuiguchi et al. [16] developed a tolerance-approach analysis of the necessary optimality degree of non-degenerate basic feasible solutions in linear programs with fuzzy objective coefficients, yielding a robust-optimality criterion against coefficient fluctuations. Khan et al. [19] combined fuzzy neural networks with a penalty-function formulation to solve fuzzy quadratic programming problems arising in a mean-variance Markowitz portfolio model. Popa [23] proposed a new ranking function for trapezoidal intuitionistic fuzzy numbers, incorporating a parameter for the decision-maker's attitude, and applied it to multi-criteria decision making. Fuzzy-measure-based network models have also been used to improve community detection in directed graphs [13]. We note that the term "fuzzy graph" in this paper refers to a crisp graph equipped with fuzzy edge and vertex weights, which differs from Rosenfeld's fuzzy graph [25]; the distinction is made precise in Section 4.

The closest study to ours is that of Alsaeedi et al. [4], who considered fuzzy edge weights in a two-way partitioning problem and compared the GA, TS, and sequential least squares programming (SLSQP). Abouyee Mehrizi et al. [1] studied a related balanced partitioning problem with crisp edge and vertex weights using continuous quadratic programming and hybrid local-search strategies. The present work improves upon these studies in three ways. First, the general k-partition case is considered. Second, fuzzy weights are used for both edges and vertices. Third, an exchange update avoids a complete recomputation of the objective function after each move.

Fuzzy numbers in trapezoidal form do not generally remain in their original family after multiplication or exponentiation. Therefore, rather than defining the model on fuzzy numbers, we apply a ranking function to the fuzzy edge and vertex weights and build the squared vertex-weight dissimilarities in the ranked space. The optimization model is constructed from these ranked values. In this sense, the equivalence used here is an *objective-order equivalence induced by the ranking function*; it is not a claim that every fractional solution of a fuzzy continuous relaxation is equivalent to a binary one.

Starting from an initial k-partition that satisfies the assignment and cardinality constraints, the method repeatedly swaps two vertices assigned to different groups while preserving both constraints. We derive the exact change in the objective together with a closed-form update of the auxiliary matrix B (Section 5.1), which gives a direct monotonicity argument and makes move evaluation inexpensive after preprocessing.

The main contributions of the paper are as follows:

i. a ranking-based formulation for k-partitioning with fuzzy edge and vertex weights that avoids non-closed fuzzy products and squares (Sections 3 and 4);

ii. a clear separation between the discrete model, its diagonally perturbed continuous relaxation, and the ranking-induced scalar objective;

iii. a feasible exchange heuristic whose gain formula is the k-way analogue of the classical Kernighan–Lin gain update, extended to vertex weights through the quadratic dissimi-

Corrected proofs

larity term and to a ranked fuzzy model; the auxiliary matrix admits a closed-form two-column update, which yields a complete monotonicity proof (Section 5.1);

iv. a complexity analysis showing $O(n^2)$ preprocessing and $O(n)$ work per attempted exchange for dense instances (Section 5.2);

v. a 15-run computational study with fixed seeds, mean $\pm$ SD reporting, paired significance tests with Holm–Bonferroni correction, exact Wilcoxon signed-rank corroboration, SLSQP and Kernighan–Lin/Fiduccia–Mattheyses (KL/FM)-type baselines, and a parameter-sensitivity analysis (Section 6); on the two larger instances ($n = 1000$ and $n = 1500$), the proposed method delivers the best mean objective and the fastest mean runtime among all five methods, with speedups of up to $233.1\times$ over SLSQP and $136.1\times$ over TS on Example 2 and up to $120.4\times$ over SLSQP and $55.6\times$ over TS on Example 3;

vi. a timing comparison with a KL/FM-type baseline that selects the best cross-partition swap and recomputes gains from scratch: its wall-clock time is $2.35\times$ (Example 1), $11.7\times$ (Example 2), and $15.4\times$ (Example 3) larger than that of the proposed method on the same starting partitions, and the ratio reflects both move selection and the closed-form update (Table 11);

vii. an explicit statement of limitations (Section 7): Examples 2 and 3 use a single random instance, all instances are Erdős–Rényi graphs, and on the smallest instance a tuned TS configuration exceeds the mean objective of the proposed method.

The remainder of this paper is organized as follows. Section 2 examines the graph partitioning problem from a traditional perspective, covering the two-partition case (Section 2.1) and the k-partition case (Section 2.2). Section 3 describes edge and vertex weights and the diagonal perturbation. Section 4 discusses the fuzzy preliminaries and the ranking-based scalarization, including the ranking-based model (Section 4.1) and the assumptions used in the theoretical analysis (Section 4.2). Section 5 details the exchange algorithm, its objective-change formula and monotonicity (Section 5.1), its computational complexity (Section 5.2), and the TS benchmark (Section 5.3). Section 6 presents the numerical experiments: the experimental protocol (Section 6.1), the statistical analysis (Section 6.2), the hardware and software environment (Section 6.3), the parameter settings and sensitivity study (Section 6.4), the random instance generation (Section 6.5), the three benchmark examples (Section 6.6), and a discussion of the results (Section 6.7). Finally, Section 7 concludes the paper. The adjacency data of Example 1 are given in the Appendix.

Corrected proofs

2 Graph Partitioning Problem

The standard graph partitioning problem is first recalled for the two-set case (Section 2.1) and then generalized to the k-partition case (Section 2.2).

2.1 Graph Two-Partitioning Problem

An undirected graph $G = (V, E)$ is given, where $V = \{v_1, \ldots, v_n\}$ is the vertex set and E is the edge set, each edge having a non-negative weight. A partitioning criterion must be defined and optimized over all possible ways of partitioning the vertex set V. The vertex set is divided into two disjoint sets V_1 and V_2 such that

$$V = V_1 \cup V_2, \qquad V_1 \cap V_2 = \emptyset.$$

For example, a common criterion is to minimize the total weight of edges between different sets subject to balance or size constraints. Let $x = (x_1, x_2, \ldots, x_n)^\top$ be a binary vector associated with the vertices of G, where

$$V_1 = \{i : x_i = 1\}, \qquad V_2 = \{i : x_i = 0\}.$$

Let $e_n = (1, 1, \ldots, 1)^\top \in \mathbb{R}^n$. The condition $e_n^\top x = m$ means that exactly m vertices are assigned to V_1, while the remaining $n - m$ vertices are assigned to V_2. Let $A = (a_{ij})$ denote the weighted adjacency matrix of the graph, where a_{ij} is the weight of the edge connecting vertices v_i and v_j.

Following the continuous quadratic programming formulation, the two-partition problem can be relaxed as

$$\begin{aligned} \max \quad & (e_n - x)^\top (A + D)x \\ \text{s.t.} \quad & e_n^\top x = m, \\ & 0 \le x \le e_n, \end{aligned} \tag{1}$$

where D is a diagonal matrix specified in the next theorem. This theorem gives conditions under which the continuous formulation still admits a binary optimum.

Theorem 1 ([14]). If the diagonal matrix $D = \operatorname{diag}(d_{11}, d_{22}, \ldots, d_{nn})$ is chosen such that

$$d_{ii} + d_{jj} \ge 2a_{ij}, \qquad d_{ii} \ge 0, \qquad i, j = 1, 2, \ldots, n,$$

then problem (1) has a binary optimal solution. Furthermore, if the inequalities are strict, every local maximizer of problem (1) is binary.

Corrected proofs

2.2 Graph k-Partitioning Problem

The k-partitioning problem generalizes the two-way case by dividing V into k pairwise disjoint subsets $V_1, V_2, \ldots, V_k$. Let

$$V = \bigcup_{p=1}^{k} V_p, \qquad V_p \cap V_q = \emptyset \quad (p \neq q),$$

and suppose that $|V_p| = m_p$ for $p = 1, 2, \ldots, k$, where

$$m_1 + m_2 + \cdots + m_k = n.$$

The objective is to obtain a balanced partition with a small cross-partition edge weight. Equivalently, one may maximize the total weight of edges whose endpoints remain in the same subset.

Let A_0 denote the weighted adjacency matrix of G, and let

$$A = A_0 + D,$$

where D is a diagonal matrix satisfying

$$d_{ii} + d_{jj} \geq 2a_{ij}, \qquad 1 \leq i, j \leq n.$$

Hager and Krylyuk formulated the continuous version of the graph k-partitioning problem with weighted edges as follows [14]:

$$\begin{aligned} \max \quad & \operatorname{tr}(X^\top A X) \\ \text{s.t.} \quad & X e_k = e_n, \\ & X^\top e_n = m, \\ & X \geq 0. \end{aligned} \tag{2}$$

Here, $X \in \mathbb{R}^{n \times k}$ is the partition matrix, while $e_k \in \mathbb{R}^k$ and $e_n \in \mathbb{R}^n$ denote vectors of ones, and $m = (m_1, m_2, \ldots, m_k)^\top$. The constraint $X e_k = e_n$ ensures that each vertex is assigned to exactly one partition, while $X^\top e_n = m$ enforces the required size of each partition. In the discrete model, $X \in \{0, 1\}^{n \times k}$, with $x_{ip} = 1$ when vertex v_i is assigned to subset V_p and $x_{ip} = 0$ otherwise. The objective $\operatorname{tr}(X^\top A X)$ measures the total within-subset edge weight, so maximizing it is equivalent to minimizing the weight of edges crossing between subsets. Necessary and sufficient conditions for local optimality are given in [15]. In Section 3, we extend this formulation to include vertex weights as well as edge weights.

3 k-Partitioning Problem with Weighted Edges and Vertices

Let $G = (V, E)$ be an undirected graph with n vertices, and let $\Pi = \{V_1, \ldots, V_k\}$ denote a partition with prescribed cardinalities $|V_p| = m_p$, where $\sum_{p=1}^{k} m_p = n$. We extend the standard

Corrected proofs

multi-way partitioning model in Eq. (2) by combining the edge contribution with a term that reflects differences in vertex weights.

Application and role of θ. A representative application is document or entity clustering, in which the edge term rewards strong connections inside a cluster, while the vertex term rewards within-cluster differences in importance scores (e.g., PageRank or citation counts). The trade-off parameter $\theta \geq 0$ in Eq. (3) therefore controls how strongly the model balances these two criteria: larger values of θ place more emphasis on within-cluster heterogeneity in importance scores, whereas smaller values favor edge cohesion. In the experiments, $\theta = 0.5$ is used when the ranked edge and vertex contributions have comparable scales (Examples 1 and 2), while $\theta = 0.8$ is used in Example 3 because the vertex term is smaller relative to the edge term there. This is a scale-matching choice rather than a tuned hyperparameter, and its sensitivity is documented in Section 6.4.

Let $w_i = w(v_i)$ be the crisp weight of vertex v_i, and define the symmetric pairwise dissimilarity matrix $W = (w_{ij})$ by

$$w_{ij} = (w_i - w_j)^2, \qquad i, j = 1, \ldots, n.$$

The sum of all pairwise vertex-weight dissimilarities is fixed for a given graph. It follows that minimizing the dissimilarity across different subsets is equivalent to maximizing the dissimilarity retained within the subsets. For a binary partition matrix $X \in \{0,1\}^{n\times k}$, the corresponding within-partition term is $\mathrm{tr}(X^\top W X)$.

Let A be the symmetric weighted adjacency matrix, and let $\theta \geq 0$ control the relative importance of the vertex-weight dissimilarity term; larger θ gives greater emphasis to vertex-weight differences. Define

$$H = A + \theta W. \tag{3}$$

The discrete weighted k-partitioning model is

$$\begin{aligned} \max_{X\in\{0,1\}^{n\times k}} \quad & \mathrm{tr}(X^\top H X) \\ \text{s.t.} \quad & X^\top e_n = m, \\ & X e_k = e_n. \end{aligned} \tag{4}$$

To establish a connection between the discrete problem (4) and continuous quadratic programming, define the diagonal matrix $D = \mathrm{diag}(d_{11}, \ldots, d_{nn})$ and consider

$$\begin{aligned} \max_{X\in\mathbb{R}^{n\times k}} \quad & \mathrm{tr}\Big(X^\top (H + D) X\Big) \\ \text{s.t.} \quad & X^\top e_n = m, \\ & X e_k = e_n, \\ & X \geq 0. \end{aligned} \tag{5}$$

Corrected proofs

For every *binary* feasible X, each row contains exactly one unit entry, and hence

$$\operatorname{tr}(X^\top D X) = \sum_{i=1}^{n} d_{ii}. \tag{6}$$

Thus, on the binary feasible set, the diagonal perturbation adds the same constant to every feasible partition and therefore leaves the optimal partitions unchanged. The role of problem (5) is instead to provide a diagonally perturbed continuous relaxation from which structural properties can be derived.

Theorem 2. Assume that $H = (h_{ij})$ is symmetric. If D is chosen so that

$$d_{ii} + d_{jj} \geq 2h_{ij}, \qquad d_{ii} \geq 0, \qquad i, j = 1, \ldots, n, \tag{7}$$

then the continuous relaxation (5) has at least one binary global maximizer. If the inequalities are strict, then every local maximizer of (5) is binary.

Proof. Apply the continuous quadratic graph-partitioning result of Hager and Krylyuk (Theorem 1) [14] to the symmetric matrix $H = A + \theta W$. Condition (7) is exactly the required diagonal-perturbation condition with a_{ij} replaced by h_{ij}. The theorem ensures that, for the perturbed continuous problem, there exists a global maximizer that is binary. Likewise, all local maximizers are binary, provided that the inequalities are strict. Since Eq. (6) is constant for all feasible binary matrices, the binary maximizer of the perturbed discrete problem coincides with that of the unperturbed discrete problem. □

4 Fuzzy Graph Partitioning

This section discusses the fuzzy attributes used in the model and shows how they are converted into a single scalar objective that the algorithm can use.

Terminology note. In this work, a fuzzy graph is defined as a crisp graph with fuzzy weights on the vertices and edges; it is not a Rosenfeld fuzzy graph. This distinction is necessary because the procedure ranks weights rather than membership degrees; see also [9] for fuzzy number-valued fuzzy graphs.

Definition 1 (Fuzzy set and fuzzy number [22, 30])**.** Let $\mathcal{X}$ be a universe of discourse. A fuzzy set $\tilde{A}$ on $\mathcal{X}$ is characterized by a membership function $\mu_{\tilde{A}} : \mathcal{X} \to [0, 1]$. A fuzzy number is a normalized, convex fuzzy set on $\mathbb{R}$ with an appropriate piecewise-continuous membership function.

Corrected proofs

Definition 2 (Trapezoidal fuzzy number [8, 22])**.** A trapezoidal fuzzy number is represented by the quadruple $\tilde{a} = (a_L, a_U, \alpha_a, \beta_a)$, where $[a_L, a_U]$ is the core, $\alpha_a \geq 0$ is the left spread, and $\beta_a \geq 0$ is the right spread. The support is $[a_L - \alpha_a, a_U + \beta_a]$. The set of trapezoidal fuzzy numbers is denoted by $F(\mathbb{R})$.

Figure 1 illustrates a trapezoidal fuzzy number.

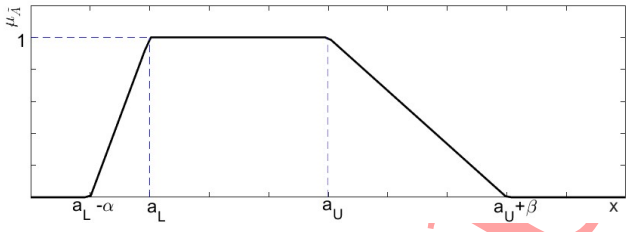


Figure 1: Visual representation of a trapezoidal fuzzy number $\tilde{a} = (a_L, a_U, \alpha_a, \beta_a)$.

Definition 3 (Fuzzy arithmetic [8, 22])**.** Let $\tilde{a} = (a_L, a_U, \alpha_a, \beta_a)$ and $\tilde{b} = (b_L, b_U, \alpha_b, \beta_b)$ be trapezoidal fuzzy numbers. Then $\tilde{a} + \tilde{b} = (a_L + b_L, a_U + b_U, \alpha_a + \alpha_b, \beta_a + \beta_b)$, and for $r \in \mathbb{R}$, $r\tilde{a} = (ra_L, ra_U, r\alpha_a, r\beta_a)$ if $r \geq 0$, and $r\tilde{a} = (ra_U, ra_L, -r\beta_a, -r\alpha_a)$ if $r < 0$.

Definition 4 (Ranking function [22, 28])**.** A ranking function $\mathcal{R} : F(\mathbb{R}) \to \mathbb{R}$ induces an order through $\tilde{a} \preceq_{\mathcal{R}} \tilde{b}$ if and only if $\mathcal{R}(\tilde{a}) \leq \mathcal{R}(\tilde{b})$.

Definition 5 (Yager's linear ranking function [27, 28])**.** For $\tilde{a} = (a_L, a_U, \alpha_a, \beta_a)$,

$$\mathcal{R}(\tilde{a}) = \frac{a_L + a_U}{2} + \frac{\beta_a - \alpha_a}{4}. \tag{8}$$

Accordingly, for $\tilde{a} = (a_L, a_U, \alpha_a, \beta_a)$ and $\tilde{b} = (b_L, b_U, \alpha_b, \beta_b)$,

$$\tilde{a} \preceq_{\mathcal{R}} \tilde{b} \iff 2(a_L + a_U) + (\beta_a - \alpha_a) \leq 2(b_L + b_U) + (\beta_b - \alpha_b).$$

4.1 Ranking-Based Fuzzy k-Partitioning Model

Let $\tilde{a}_{ij}$ be the trapezoidal fuzzy weight of edge (i, j), and let $\tilde{v}_i$ be the trapezoidal fuzzy weight associated with vertex i. Using the ranking function in Eq. (8), define the ranked edge weights and ranked vertex scores by

$$a_{ij}^{\mathcal{R}} = \mathcal{R}(\tilde{a}_{ij}), \qquad r_i = \mathcal{R}(\tilde{v}_i).$$

The ranked pairwise vertex dissimilarity is then

Corrected Proofs

$$w_{ij}^{\mathcal{R}} = (r_i - r_j)^2.$$

By forming the dissimilarity after ranking, we avoid terms such as $\tilde{v}_i^2$ and products of trapezoidal fuzzy numbers. Define the symmetric ranked matrix $H^{\mathcal{R}} = (h_{ij})$ by

$$h_{ij} = a_{ij}^{\mathcal{R}} + \theta w_{ij}^{\mathcal{R}}, \qquad h_{ii} = 0, \qquad \theta \geq 0. \tag{9}$$

The optimization criterion used in this paper is

$$\begin{aligned} \max_{X \in \{0,1\}^{n \times k}} \quad & F(X) = \operatorname{tr}\left(X^\top H^{\mathcal{R}} X\right) \\ \text{s.t.} \quad & X^\top e_n = m, \\ & X e_k = e_n. \end{aligned} \tag{10}$$

The fuzzy information therefore enters the optimization model only through scalar coefficients induced by the ranking function.

4.2 Assumptions Used in the Theoretical Analysis

The relaxation (Theorem 3) and monotonicity (Theorem 4) results rely on the following assumptions: (A1) the graph is undirected, so $H^{\mathcal{R}}$ is symmetric; (A2) all edge and vertex fuzzy weights are trapezoidal and are ranked by the same linear ranking function; (A3) $\theta \geq 0$ is fixed during a run; (A4) the prescribed partition sizes satisfy $m_p \in \mathbb{Z}_+$ and $\sum_p m_p = n$; (A5) the diagonal of $H^{\mathcal{R}}$ is set to zero; and (A6) when the simpler diagonal choice given below is used, the ranked weights satisfy $h_{ij} \geq 0$.

Theorem 3. Let $D = \operatorname{diag}(d_{11}, \ldots, d_{nn})$ be a *crisp* diagonal matrix satisfying

$$d_{ii} + d_{jj} \geq 2h_{ij}, \qquad d_{ii} \geq 0, \qquad i, j = 1, \ldots, n, \tag{11}$$

where h_{ij} is defined in Eq. (9). Then the diagonally perturbed continuous relaxation

$$\begin{aligned} \max_{X \in \mathbb{R}^{n \times k}} \quad & \operatorname{tr}\left(X^\top (H^{\mathcal{R}} + D) X\right) \\ \text{s.t.} \quad & X^\top e_n = m, \\ & X e_k = e_n, \\ & X \geq 0, \end{aligned}$$

has at least one binary global maximizer. If the inequalities in Eq. (11) are strict, every local maximizer of this relaxation is binary.

Corrected proofs

Proof. After applying the ranking function and constructing the pairwise vertex dissimilarities in the ranked space, the matrix $H^{\mathcal{R}}$ is an ordinary real symmetric matrix. Therefore, Theorem 2 applies directly with $H = H^{\mathcal{R}}$. Moreover, for binary feasible X, Eq. (6) gives $\mathrm{tr}(X^\top D X) = \sum_i d_{ii}$, so the diagonal perturbation does not change the ordering of binary partitions. □

A convenient sufficient choice that guarantees the non-negativity condition $d_{ii} \geq 0$, even when some ranked weights are negative, is

$$d_{ii} = \max\Big\{0,\ \max_{j \neq i} h_{ij}\Big\} + \varepsilon_D, \qquad \varepsilon_D > 0,$$

which makes the inequalities in Eq. (11) strict. Under (A6) ($h_{ij} \geq 0$), the simpler choice $d_{ii} = \max_{j \neq i} h_{ij} + \varepsilon_D$ also satisfies Eq. (11). A constant diagonal $d_{ii} = d_{\mathrm{val}} = \max_{p,q} h_{pq} + \varepsilon_D$ is also valid and is the specific choice used in the numerical implementation of Section 6. Because the corresponding contribution $n\, d_{\mathrm{val}}$ is identical for every method and every run on binary feasible matrices, it does not affect the rankings, the paired tests, or the speedup ratios, and all reported objective values equal $F(X) + n\, d_{\mathrm{val}}$.

5 Proposed Method

For notational simplicity, write $H = H^{\mathcal{R}}$ and let X be any binary feasible partition matrix for problem (10). Define

$$B = HX.$$

Suppose vertex i currently belongs to partition s and vertex j belongs to partition $t \neq s$. An exchange move replaces

$$x_s^N = x_s^O + e_j - e_i, \qquad x_t^N = x_t^O + e_i - e_j, \tag{12}$$

while all other columns of X remain unchanged. Because one vertex leaves and one enters each affected partition, both $Xe_k = e_n$ and $X^\top e_n = m$ are preserved.

5.1 Objective-Change Formula and Monotonicity

Since H is symmetric and $h_{ii} = h_{jj} = 0$, direct expansion of the two affected quadratic terms gives

$$\Delta F := F(X^N) - F(X^O) = 2\,(B_{js} - B_{is} - B_{jt} + B_{it} - 2h_{ij})\,. \tag{13}$$

Consequently, the exchange is nondecreasing whenever

$$h_{ij} \leq \frac{1}{2}\,(B_{js} - B_{jt} + B_{it} - B_{is})\,.$$

Corrected proofs

For a fixed pair (s, j), selecting

$$i \in \arg\max_{r:X_{rs}=1} \{B_{rt} - B_{rs} - 2h_{rj}\} \tag{14}$$

maximizes the move gain among the vertices currently assigned to partition s.

The same exchange also admits a simple update of B:

$$\begin{aligned} B^N_{:s} &= B^O_{:s} + H_{:j} - H_{:i}, \\ B^N_{:t} &= B^O_{:t} + H_{:i} - H_{:j}, \end{aligned} \tag{15}$$

and every other column of B is unchanged. Thus, after an accepted exchange, HX need not be recomputed from scratch.

Algorithm 1 Monotone move-based heuristic

Input: Symmetric ranked matrix H, feasible binary partition X, no-improvement limit $I_{\max}$, tolerance $\tau \geq 0$, and random seed

Output: Locally improved feasible partition X and objective value $F(X)$

Compute $B = HX$ and set $c \leftarrow 0$
while $c < I_{\max}$ **do**
  $c \leftarrow c + 1$
  Randomly choose a partition s
  Randomly choose a vertex j with $X_{js} = 0$, and let t be the unique partition with $X_{jt} = 1$
  Choose i by Eq. (14)
  Compute ΔF from Eq. (13)
  if $\Delta F \geq \tau$ **then**
    Exchange the assignments of vertices i and j between partitions s and t
    Update columns s and t of B using Eq. (15)
    if $\Delta F > \tau$ **then**
      $c \leftarrow 0$
    end if
  end if
end while
return X and $F(X)$

Positioning relative to Kernighan–Lin. The gain formula (13) and the two-column update (15) are the k-way analogue of the classical Kernighan–Lin D-value update [18]. Three features are new: (i) the update is carried out for the k-way partition matrix X; (ii) the auxiliary matrix $B = HX$ includes the vertex-weight dissimilarity term θW; and (iii) all coefficients of H are induced by a linear ranking function on trapezoidal fuzzy numbers. The resulting procedure is summarized in Algorithm 1.

Corrected proofs

Policy on ties and zero-gain moves: if multiple vertices attain the maximum in Eq. (14), one is selected uniformly at random. A zero-gain exchange is accepted only if $\tau = 0$ (weak improvement); with the experimental setting $\tau = 10^{-15}$, it is rejected.

Theorem 4. Assume (A1)–(A5). If the proposed algorithm accepts only exchanges with $\Delta F \geq \tau$, where $\tau \geq 0$, then the objective function values do not decrease. When $\tau > 0$, the algorithm accepts only exchanges that strictly improve the objective function value.

Proof. The change in the objective function due to a candidate exchange is given exactly by Eq. (13). By the acceptance criterion of the proposed algorithm, $\Delta F \geq \tau$ for every accepted exchange. Consequently, $F(X^N) \geq F(X^O)$, which proves monotonicity; if $\tau > 0$, then $F(X^N) > F(X^O)$. Feasibility follows from Eq. (12), and multiplying the two updated columns in Eq. (12) by H yields Eq. (15). □

5.2 Computational Complexity

For a dense $n \times n$ matrix H, $O(n^2)$ storage is needed just to form H explicitly. The initialization $B = HX$ takes $O(n^2)$ time. In each iteration, evaluating the selection rule (14) costs $O(n)$, the acceptance test costs $O(1)$, and the update (15) costs $O(n)$. Thus, after preprocessing, the proposed algorithm requires $O(n)$ work per attempted exchange and $O(n^2 + In)$ time over I iterations, with $O(n^2 + nk)$ storage.

A TS iteration that explicitly evaluates N_{TS} candidate neighbors performs at least $\Omega(N_{\mathrm{TS}})$ move evaluations. A GA generation evaluates a population of P candidate partitions. A KL/FM-type baseline that recomputes all gains from scratch after each accepted swap requires $O(n^2)$ work per accepted swap, whereas the proposed method requires $O(n)$ per attempted exchange. Actual wall-clock time depends, of course, on the implementation and the hardware, so the empirical comparison (Section 6) is reported separately from this structural complexity analysis.

5.3 Tabu Search

Tabu Search is a memory-based local search method that restricts recently used moves to reduce cycling, with aspiration criteria allowing a tabu move when it leads to a sufficiently good solution. TS has been used extensively in graph partitioning [24], and recent variants continue to combine tabu mechanisms with multilevel and specialized neighborhood strategies [20]. In our benchmark, TS follows the same feasibility-preserving exchange structure as the main problem, with a tabu tenure, a best-so-far aspiration rule, and a sampled neighborhood of size N_{TS} on the larger instances. The flowchart of the benchmark is shown in Figure 2.

Corrected proofs

Candidate pairs are sampled in the main experiments. In the $N_{\mathrm{TS}} = 190$ sensitivity setting on Example 1, all unordered pairs are enumerated. Feasible neighbors of the current partition X are generated as follows. For each of the N_{TS} candidate unordered pairs (p, q), let s and t be the partitions with $X_{ps} = X_{qt} = 1$. If $s = t$, the pair is skipped; otherwise, the neighbor $\bar{X}$ is obtained from X by exchanging the assignments of p and q, that is, $\bar{X}_{ps} = 0$, $\bar{X}_{pt} = 1$, $\bar{X}_{qt} = 0$, and $\bar{X}_{qs} = 1$, and it is stored as a candidate neighbor.

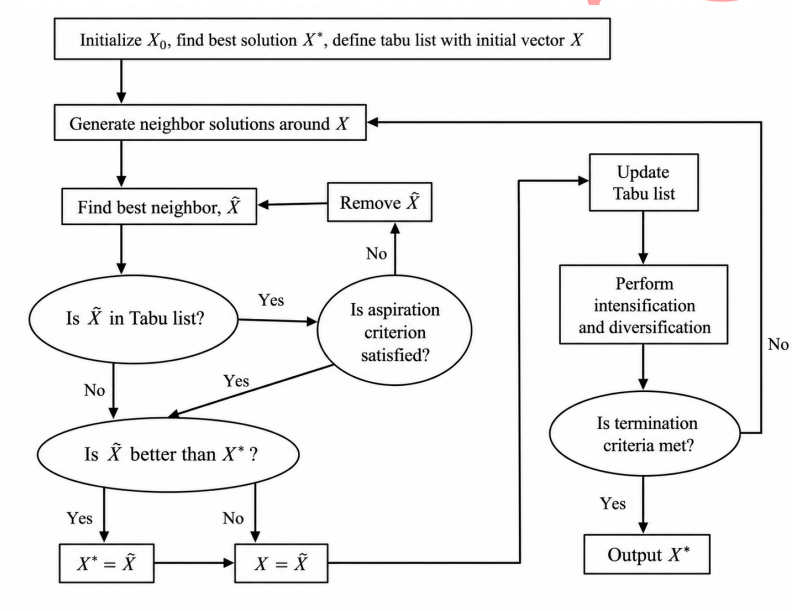


Figure 2: Tabu Search flowchart. The intensification/diversification step resets the tabu list whenever the best-so-far objective has not improved for $N_{\mathrm{div}} = 50$ iterations.

6 Numerical Experiments

This section evaluates the proposed heuristic on three benchmark instances: a small graph (Example 1) and two large random graphs (Examples 2 and 3); the examples are described in Section 6.6.

6.1 Experimental Protocol and Reproducibility

We evaluate the proposed heuristic against the GA, TS, SLSQP, and a Kernighan–Lin/Fiduccia–Mattheyses (KL/FM)-type swap local search on three benchmark instances. Each method is run $R = 15$ times on every benchmark using algorithm seeds $101, 102, \ldots, 115$. For the MATLAB-

based randomized methods, the random-number generator is initialized at the start of each run with `rng(seed,'twister')`. Within a repetition, the randomized methods use the same seed and the same feasible starting partition. The raw per-run outputs of the main experiments are collected in Supplementary Material S1 (see Code Availability in the Declarations). The means, sample standard deviations, best values, and paired-test p-values for the main experiments are computed from these records. The sensitivity-grid means are reported separately.

Reported objective and the diagonal perturbation. All objective values in Tables 8, 9, and 10 are computed from the diagonally perturbed matrix $A_{\text{obj}} = H^{\mathcal{R}} + D$ used in the implementation, with the constant diagonal $d_{ii} = d_{\text{val}} = \max_{p,q} h_{pq} + 1$. Because every feasible X is binary, the reported value is $F_{\text{rep}}(X) = F(X) + n\, d_{\text{val}}$. The same constant is added for every method and every run, so rankings, maximizer locations, paired objective differences, and their p-values are unaffected; absolute objective values and objective ratios, however, depend on this shift.

Termination rules per method. Let x_r denote the best objective value available at iteration r, and let

$$\left| \frac{x_r - x_{r-1}}{x_r} \right| < \epsilon, \qquad \epsilon = 10^{-5}, \tag{16}$$

be the relative-error stopping test. The termination rules are as follows:

- **GA:** terminate when Eq. (16) holds for five consecutive generations, or after at most 400 generations (Example 1) or 600 generations (Examples 2 and 3).
- **TS:** terminate when Eq. (16) holds for five consecutive iterations, or after at most 600 TS iterations.
- **SLSQP:** terminate when Eq. (16) holds for five consecutive iterations, or after at most 1000 SLSQP iterations.
- **KL/FM:** terminate when no cross-partition swap with strictly positive gain remains, or after at most 1000 accepted swaps.
- **Proposed method:** the outer loop of the proposed algorithm is stopped when the no-improvement counter reaches $I_{\max} = 1000$ consecutive attempts without a gain exceeding τ.

The choice $I_{\max} = 1000$ is conservative relative to the total number of attempted exchanges (on the order of 10^6 on Example 3), but the counter is reset to zero whenever $\Delta F > \tau$, as specified in the proposed algorithm. Table 1 reports, for each method and each example, the number of runs terminated by the primary rule versus the number that reached the safety cap. For the proposed method, the primary rule is the $I_{\max}$ counter itself, so the entry $0/15$ records the normal termination of every proposed run and does *not* indicate an abnormal cap hit. All

Corrected proofs

GA, TS, SLSQP, and KL/FM runs terminated on their primary rules, and the per-run termination flags are listed in Supplementary Material S1.

Table 1: Termination profile of each method on each example (number of runs terminated by the primary rule / number of runs that reached the safety cap).

Method	Example 1	Example 2	Example 3	Primary rule	Safety cap
GA	15/0	15/0	15/0	Eq. (16)	400/600/600 generations
TS	15/0	15/0	15/0	Eq. (16)	600 iterations
SLSQP	15/0	15/0	15/0	Eq. (16)	1000 iterations
KL/FM	15/0	15/0	15/0	No positive-gain swap	1000 swaps
Proposed	0/15	0/15	0/15	$I_{\max}$ counter	$I_{\max} = 1000$

For the GA, the population size is 60 for Example 1 and 100 for Examples 2 and 3, with crossover probability 0.9, mutation probability 0.25, tournament size 4, and elitism size 2. TS uses a tabu tenure of 7 and a sampled neighborhood of size 2000 in the main experiments. SLSQP is implemented through `scipy.optimize.minimize` with `method='SLSQP'`. The KL/FM-type baseline operates on the same ranked objective and starts from the same feasible initial partition; after each accepted swap, all gains are recomputed from scratch and the best cross-partition swap is selected. Table 11 reports the ratios of mean wall-clock times together with the mean and sample SD of the 15 paired KL/FM-to-proposed objective ratios, computed from Supplementary Material S1 using matched seeds. The KL/FM baseline is compared descriptively and is not included in the nine paired significance tests. The complete parameter settings are listed in Table 3.

6.2 Statistical Analysis

For all five methods, we report the mean and sample standard deviation of the objective value and wall-clock time, together with iteration counts where available. The proposed method is compared with GA, TS, and SLSQP using a two-sided paired t-test with 14 degrees of freedom at the 0.05 significance level. The Holm–Bonferroni correction is applied to the family of nine paired tests (three baselines across three examples). In the main experiments, the paired comparisons of the proposed method against GA, TS (with the reference configuration of Table 3), and SLSQP remain significant after this correction.

As a nonparametric robustness check, two-sided Wilcoxon signed-rank p-values were computed from the same 15 paired runs and are reported in Table 12. Exact permutation values are reported; the normal approximation with tie correction gives values between 4.0×10^{-4} and

Corrected proofs

9.8×10^{-4}, which are also below 0.05. All comparisons remain significant at the 0.05 level, and the ranking of methods is unchanged.

6.3 Hardware and Software Environment

All experiments were performed on the same computer (Table 2). The GA, TS, KL/FM, and the proposed heuristic were implemented in MATLAB R2023b, whereas SLSQP was implemented in Python 3.11.5 using SciPy 1.11.1 and NumPy 1.24.3. No GPU acceleration was used. MATLAB elapsed times were measured using `tic`/`toc`; the reported times are wall-clock times, not CPU times. The time ratios are computed from the mean recorded times without rounding.

Table 2: Hardware and software environment.

Item	Specification
Processor	Intel(R) Core(TM) i5-8250U CPU @ 1.60 GHz (1.80 GHz)
Installed RAM	12.0 GB (11.4 GB usable)
Operating system	Windows 10, 64-bit
MATLAB version	R2023b
Python version	3.11.5
SciPy version	1.11.1
NumPy version	1.24.3

6.4 Parameter Settings and Sensitivity

The main experiments use $\theta = 0.5$ in Examples 1 and 2 and $\theta = 0.8$ in Example 3; this is a scale-matching choice rather than a tuned hyperparameter. Table 3 summarizes the principal algorithm settings. The proposed method uses the acceptance tolerance $\tau = 10^{-15}$ in all three benchmark experiments; this is distinct from the relative-error tolerance $\epsilon = 10^{-5}$ used by the other methods. A sensitivity sweep over $\theta \in \{0.2, 0.5, 0.8, 1.0\}$ on Example 1 is reported in Table 7.

The GA mutation probability was tested at $\{0.05, 0.10, 0.25, 0.40, 0.50, 0.60\}$, with population sizes of $\{40, 60, 80, 100\}$. Similarly, the TS tenure was examined over $\{5, 7, 10, 15, 20, 25, 30\}$, while the neighborhood size was set to $\{50, 100, 190\}$. Tables 4 and 5 report the resulting mean objective values for the GA and TS, respectively, Table 7 reports the sensitivity of the proposed method to θ, and Table 6 lists the best grid setting and the reference setting used for each baseline on each instance.

Corrected proofs

Table 3: Main parameter settings.

Algorithm	Parameter	Value
GA	Population size	60 (Example 1); 100 (Examples 2 and 3)
GA	Safety maximum generations	400 (Example 1); 600 (Examples 2 and 3)
GA	Crossover probability	0.9
GA	Mutation probability	0.25
GA	Tournament size	4
GA	Elitism size	2
TS	Tabu tenure	7 (main); 5–30 (sensitivity)
TS	Candidate pairs N_{TS}	2000 (main); 50, 100, 190 (Example 1 sensitivity)
TS	Safety maximum iterations	600
TS	Diversification counter N_{div}	50 iterations
SLSQP	Safety maximum iterations	1000
KL/FM	Gain recomputation	Full, after each accepted swap
KL/FM	Safety maximum swaps	1000
KL/FM	Stopping	No positive-gain swap or 1000 accepted swaps
Proposed	No-improvement limit $I_{\max}$	1000 consecutive attempts
Proposed	Acceptance tolerance τ	10^{-15}
Proposed	Stopping	$I_{\max}$ counter (no relative-error test)
GA, TS, SLSQP	Relative-error tolerance ϵ	10^{-5}
All	Trade-off θ	0.5 (Examples 1 and 2); 0.8 (Example 3)

Table 4: GA sensitivity on Example 1 (mean objective over 15 runs).

	Population size			
Mutation probability	**40**	**60**	**80**	**100**
0.05	878.3	901.5	914.7	932.8
0.10	945.2	1030.4	1125.3	1195.3
0.25	1102.6	1255.8	1255.8	1255.8
0.40	983.4	1035.6	1132.9	1204.1
0.50	938.3	1021.5	1030.7	1083.3
0.60	935.3	1019.3	1025.6	1079.8

Corrected proofs

Table 5: TS sensitivity on Example 1 (mean objective over 15 runs).

	Tabu tenure						
Neighborhood size	**5**	**7**	**10**	**15**	**20**	**25**	**30**
50	968.4	983.2	981.7	997.0	994.2	992.8	990.2
100	1108.4	1130.2	1158.7	1183.0	1179.4	1177.2	1175.2
190	1191.4	1258.3	1261.2	**1277.8**	**1278.5**	**1276.9**	**1275.2**

The reference GA setting (population 60, mutation 0.25) attains the highest mean objective in the grid, tied with population sizes 80 and 100 at mutation 0.25. Mutation probabilities between 0.40 and 0.60 give lower means. The TS sensitivity grid includes tenures from 5 to 30; at $N_{\mathrm{TS}} = 190$, the highest reported mean is 1278.5 (tenure 20), compared with 1277.8 at tenure 15 and 1262.1 for the proposed method. This represents a limitation of the proposed method on Example 1. For $N_{\mathrm{TS}} = 190$, every unordered vertex pair is enumerated in each iteration: the 36 within-partition pairs are skipped, and all 154 distinct feasible cross-partition swaps are evaluated. Thus, N_{TS} counts candidate pairs, not feasible neighbors. The sensitivity grid was not run on Examples 2 and 3.

Numerical rationale for $N_{\mathrm{TS}} = 2000$ on Examples 2 and 3. The full cross-partition neighborhood has $\sum_{p<q} m_p m_q = (n^2 - \sum_p m_p^2)/2$ unordered pairs. With $n = 1000$ and $k = 50$ (average $m_p = 20$), this gives approximately 4.9×10^5 pairs, so $N_{\mathrm{TS}} = 2000$ samples about 0.41% of them. With $n = 1500$ and $k = 40$ (average $m_p = 37.5$), it gives approximately 1.1×10^6 pairs, so $N_{\mathrm{TS}} = 2000$ samples about 0.18%. The value 2000 was fixed to keep each TS run within a practical runtime budget.

Table 6: Best grid setting and reference setting of each baseline on each instance.

Method	**Instance**	**Best grid setting**	**Reference setting used**
GA	Example 1	population = 60, mutation = 0.25	population = 60, mutation = 0.25
GA	Example 2	grid not executed	population = 100, mutation = 0.25
GA	Example 3	grid not executed	population = 100, mutation = 0.25
TS	Example 1	tenure = 20, $N_{\mathrm{TS}} = 190$	tenure = 7, $N_{\mathrm{TS}} = 2000$
TS	Example 2	grid not executed	tenure = 7, $N_{\mathrm{TS}} = 2000$
TS	Example 3	grid not executed	tenure = 7, $N_{\mathrm{TS}} = 2000$
SLSQP	Examples 1–3	N/A	N/A

Corrected proofs

Table 7: Sensitivity of the proposed method to θ on Example 1.

θ	0.2	0.5	0.8	1.0
Mean objective	938.3	1262.1	1489.3	1725.1

6.5 Random Instance Generation

For Examples 2 and 3, we use a fixed instance seed, $s_{\text{inst}} = 2024$, so that every algorithm is evaluated on exactly the same graph and fuzzy data. The random graph is an undirected Erdős–Rényi graph in which each possible edge is included independently with probability equal to the required density (0.4 in Example 2 and 0.3 in Example 3). For each present edge, a trapezoidal fuzzy weight $\tilde{a}_{ij} = (a_L, a_U, \alpha_a, \beta_a)$ is generated, with a_L and a_U drawn uniformly from $[0, 10]$ and reordered if necessary so that $a_L \le a_U$, while α_a and β_a are drawn uniformly from $[0, 2]$. Nonedges receive $(0, 0, 0, 0)$. Vertex fuzzy weights $\tilde{v}_i = (v_L, v_U, \alpha_v, \beta_v)$ are generated independently by the same rule.

Limitations on instance variability. Examples 2 and 3 each use a single random instance, generated with the fixed seed $s_{\text{inst}} = 2024$. The 15 independent runs therefore capture algorithmic randomness only, not instance-to-instance variability. Furthermore, all instances in this study are Erdős–Rényi random graphs. Extending the evaluation to multiple random instances, different graph families, and real-world networks is left for future work (Section 7).

6.6 Benchmark Examples

The three benchmark instances are described below; the generation of the random instances is detailed in Section 6.5.

Example 1 (Small graph: $n = 20$, $k = 5$, $\theta = 0.5$)**.** For the small benchmark, the graph topology is given in the Appendix (Table 13). For each connected edge, the fuzzy edge weight is $\tilde{a}_{ij} = (0, 2, 1, 1)$; otherwise, $\tilde{a}_{ij} = (0, 0, 0, 0)$. If the degree of vertex v_i is d_i, its fuzzy vertex weight is $\tilde{v}_i = (d_i, d_i, 1, 1)$. The prescribed partition sizes are $m = [5, 1, 5, 5, 4]^\top$. Because $\tilde{a}_{ij}$ and $\tilde{v}_i$ have symmetric spreads ($\beta - \alpha = 0$), Yager's ranking function reduces to the core midpoint in this example; the asymmetric-spread case is exercised in Examples 2 and 3. The results are reported in Table 8.

The proposed method attains a mean objective of 1262.1 ± 1.1 and a mean wall-clock time of 0.03 ± 0.03 s. The paired comparisons are significant at the 0.05 level, and they remain significant after Holm–Bonferroni correction for GA and SLSQP; the comparison with TS also favors the proposed method ($p = 1.1 \times 10^{-2}$). The KL/FM baseline requires 0.07 ± 0.01 s per

Corrected proofs

Table 8: Results on Example 1 (mean ± sample SD over 15 runs).

Method	Objective	Time (s)	Iterations	Best objective	p-value
GA	1255.8 ± 4.2	0.18 ± 0.10	21.8 ± 1.9	1258.3	8.59×10^{-5}
TS	1258.3 ± 5.5	0.05 ± 0.03	14.7 ± 1.8	1262.0	1.11×10^{-2}
SLSQP	1248.0 ± 4.4	0.04 ± 0.03	10.3 ± 0.5	1250.5	1.06×10^{-8}
KL/FM	1249.47 ± 12.42	0.07 ± 0.01	—	1263.00	—
Proposed	$\mathbf{1262.1 \pm 1.1}$	$\mathbf{0.03 \pm 0.03}$	28.9 ± 2.1	**1263.0**	—

run, about $2.35\times$ longer than the proposed method (Table 11). The convergence of the proposed algorithm for one representative run is shown in Figure 3.

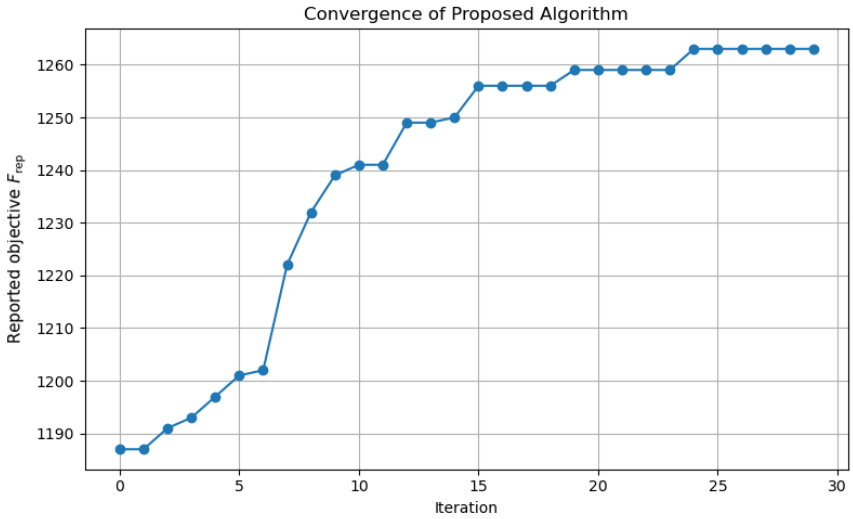


Figure 3: Convergence of the proposed algorithm for one representative run of Example 1 (reported objective F_{rep}).

Example 2 (Random graph: $n = 1000$, $k = 50$, edge density 0.4, $\theta = 0.5$)**.** The graph, fuzzy weights, and partition vector are generated as described in Section 6.5 with $s_{\text{inst}} = 2024$. The results are reported in Table 9.

For Example 2, the proposed method has the largest mean objective value among all five methods, 13801.3 ± 156.8, and the smallest mean wall-clock time, 16.40 ± 4.40 s. Relative to the baseline means, its mean objective is approximately 30.42% higher than that of GA, 2.80% higher than that of TS, and 43.44% higher than that of SLSQP. In terms of runtime, it is about $26.35\times$ faster than GA, $136.1\times$ faster than TS, and $233.1\times$ faster than SLSQP.

Corrected proofs

Table 9: Results on Example 2 (mean ± sample SD over 15 runs).

Method	Objective	Time (s)	Iterations	Best objective	p-value
GA	10582.4 ± 134.2	432.10 ± 13.20	407.2 ± 25.6	10832.3	5.19×10^{-20}
TS	13425.0 ± 150.6	2231.40 ± 157.20	532.3 ± 18.6	13580.2	6.69×10^{-5}
SLSQP	9621.4 ± 121.4	3821.20 ± 204.30	215.2 ± 11.6	9821.4	2.72×10^{-20}
KL/FM	13387.50 ± 221.29	192.00 ± 12.00	—	13643.17	—
Proposed	**13801.3 ± 156.8**	**16.40 ± 4.40**	1720.0 ± 113.5	**13921.6**	—

Example 3 (Large random graph: $n = 1500$, $k = 40$, edge density 0.3, $\theta = 0.8$)**.** The graph, fuzzy weights, and partition vector are generated as described in Section 6.5 with $s_{\text{inst}} = 2024$. The results are reported in Table 10.

Table 10: Results on Example 3 (mean ± sample SD over 15 runs).

Method	Objective	Time (s)	Iterations	Best objective	p-value
GA	32611.3 ± 274.2	683.20 ± 21.23	452.0 ± 31.6	33582.1	2.95×10^{-11}
TS	33629.4 ± 284.3	2400.93 ± 160.10	551.0 ± 21.2	33891.2	5.51×10^{-8}
SLSQP	28342.8 ± 172.1	5201.10 ± 211.31	211.0 ± 53.2	28421.0	2.70×10^{-19}
KL/FM	33474.80 ± 364.50	666.00 ± 30.00	—	33872.05	—
Proposed	**34511.0 ± 281.8**	**43.19 ± 18.20**	2529.2 ± 89.3	**34820.8**	—

Among the five methods, the proposed method attains the largest mean objective value, 34511.0 ± 281.8, compared with 32611.3 ± 274.2 for GA, 33629.4 ± 284.3 for TS, and 28342.8 ± 172.1 for SLSQP. The paired t-test p-values are 2.95×10^{-11}, 5.51×10^{-8}, and 2.70×10^{-19}, respectively, and the conclusions remain significant after Holm–Bonferroni correction and unchanged under the Wilcoxon check (Table 12). The proposed method also has a substantially lower mean wall-clock time: 43.19 ± 18.20 s versus 683.20 ± 21.23 s for GA, 2400.93 ± 160.10 s for TS, and 5201.10 ± 211.31 s for SLSQP.

6.7 Discussion of the Computational Results

Table 11 compares the proposed method with the KL/FM-type baseline on the same starting partitions, and Table 12 reports the Wilcoxon signed-rank p-values for the comparisons with GA, TS, and SLSQP.

Across the three benchmark instances, the proposed method attains the highest mean objective value and the lowest mean wall-clock time among all five methods on the two larger instances, and it is competitive on the smallest one. On Example 2, the proposed method im-

Corrected Proofs

Table 11: Proposed method versus KL/FM on the same starting partitions (runtime and descriptive objective-ratio comparison).

Metric	Example 1	Example 2	Example 3
Proposed time (s)	0.03 ± 0.03	16.40 ± 4.40	43.19 ± 18.20
KL/FM time (s)	0.07 ± 0.01	192.00 ± 12.00	666.00 ± 30.00
Time ratio (KL/FM / proposed)	$2.35\times$	$11.7\times$	$15.4\times$
Objective ratio (KL/FM / proposed)	0.99 ± 0.01	0.97 ± 0.01	0.97 ± 0.01

Table 12: Two-sided Wilcoxon signed-rank p-values of the proposed method versus each baseline over the 15 paired runs.

Baseline	Example 1	Example 2	Example 3
GA	1.2×10^{-4}	6.1×10^{-5}	6.1×10^{-5}
TS (reference)	6.1×10^{-5}	1.8×10^{-4}	6.1×10^{-5}
SLSQP	6.1×10^{-5}	6.1×10^{-5}	6.1×10^{-5}

proves the mean objective by 2.8% over TS while being $136.1\times$ faster than TS and $233.1\times$ faster than SLSQP (MATLAB versus Python/SciPy on the same machine; the latter ratio reflects both algorithmic and implementation differences). On Example 3, the improvement over TS is 2.6%, with runtime speedups of $55.6\times$ over TS and $120.4\times$ over SLSQP. The runtime reductions on these instances are consistent with the lower per-attempt cost of the proposed search; they are not a controlled measurement of that cost alone.

These findings are supported by the 15-run protocol, the paired significance tests with Holm–Bonferroni correction, the exact Wilcoxon corroboration (Table 12), the SLSQP and KL/FM-type baselines, the explicit reproducibility information, and the parameter-sensitivity study (Section 6.4). The wall-clock-time ratios between the KL/FM-type search and the proposed method are approximately $2.35\times$, $11.7\times$, and $15.4\times$ on Examples 1–3, respectively (Table 11). They reflect both the randomized move selection and the closed-form update of B, not the update alone. The KL/FM-to-proposed objective ratios are 0.99, 0.97, and 0.97 (Table 11), indicating close objective values on these instances. These descriptive ratios do not establish equivalence of local optima or statistical significance.

7 Conclusions

We have presented a ranking-based model and a monotone exchange heuristic for k-partitioning graphs whose edge and vertex weights are trapezoidal fuzzy numbers. The fuzzy inputs are

Corrected proofs

first mapped to scalar values through Yager's ranking function, and the squared vertex-weight dissimilarities are then constructed in that ranked space (Section 4). This keeps the model within a well-defined real-valued quadratic framework and avoids fuzzy products or squared fuzzy quantities that are not closed in the trapezoidal family.

The proposed search works directly with feasible partitions (Section 5). For each two-vertex exchange, the change in the objective is available in closed form, and only two columns of the auxiliary matrix B need to be updated. Under the stated symmetry and zero-diagonal assumptions, accepting only non-negative-gain moves leads to a non-decreasing objective sequence (Theorem 4). For dense instances, the cost is $O(n^2)$ for preprocessing and $O(n)$ per attempted exchange (Section 5.2).

The computational study (Section 6) uses 15 paired repetitions per instance. On the two larger synthetic instances, the proposed method achieves the highest mean objective among the five methods and the lowest mean runtime. Paired tests support the comparison with the reference GA, TS, and SLSQP settings. The sensitivity study examines algorithm parameters and the trade-off parameter θ; changing θ changes the scalar objective and is not a direct comparison of algorithm scores.

Limitations

(i) Examples 2 and 3 each use one random instance ($s_{\text{inst}} = 2024$); the 15 runs capture algorithmic randomness, not variability across instances.

(ii) The random instances are Erdős–Rényi graphs, and Example 1 has symmetric fuzzy spreads, so the ranking reduces to the core midpoint.

(iii) On Example 1, TS with a tuned tenure and the $N_{\text{TS}} = 190$ setting exceeds the mean objective of the proposed method; the baselines on Examples 2 and 3 use reference settings.

(iv) The proposed method is a local heuristic with no global optimality guarantee; termination depends on $I_{\max}$, and runtime comparisons depend on the implementations (MATLAB and Python/SciPy) and stopping rules.

(v) The KL/FM comparison reports descriptive paired objective ratios from Supplementary Material S1; no paired objective significance test against KL/FM is claimed.

Future Work

(i) Evaluate multiple random instances, other graph families, and real-world networks, and perform equal-time comparisons on larger instances.

Corrected proofs

(ii) Investigate alternative ranking functions and adaptive or multi-objective choices of θ.

(iii) Explore hybrid multilevel or learning-assisted searches, and perform exact or near-exact validation on small instances.

Appendix

The adjacency matrix of Example 1 is symmetric: $A(i, j) = A(j, i) = 1$ for every connection listed in Table 13 (83 undirected edges).

Table 13: Symmetrized vertex connections used in Example 1 (83 undirected edges).

Vertex	Adjacent vertices	Vertex	Adjacent vertices
1	2, 4, 5, 10, 15, 19	11	2, 4, 6, 7, 10, 12, 15, 18, 20
2	1, 3, 8, 10, 11, 12, 15, 18, 19, 20	12	2, 3, 4, 5, 6, 7, 8, 11, 13, 16, 18, 19
3	2, 7, 8, 9, 12, 13, 15, 16, 18, 20	13	3, 5, 6, 7, 12, 14, 15, 17, 18, 19
4	1, 5, 8, 11, 12, 14, 15, 18, 20	14	4, 13, 19, 20
5	1, 4, 6, 12, 13, 19	15	1, 2, 3, 4, 6, 8, 11, 13, 16, 17, 20
6	5, 8, 10, 11, 12, 13, 15, 16, 19, 20	16	3, 6, 12, 15, 17
7	3, 8, 10, 11, 12, 13, 17, 19, 20	17	7, 8, 9, 13, 15, 16, 18, 19
8	2, 3, 4, 6, 7, 12, 15, 17, 19	18	2, 3, 4, 9, 11, 12, 13, 17, 20
9	3, 17, 18, 19	19	1, 2, 5, 6, 7, 8, 9, 12, 13, 14, 17
10	1, 2, 6, 7, 11	20	2, 3, 4, 6, 7, 11, 14, 15, 18

Declarations

Code Availability

The code repository is publicly accessible at https://doi.org/10.5281/zenodo.23902741 under the CC BY 4.0 licence. Supplementary Material S1 contains the per-run results. The reproducibility file map is as follows:

- `proposed_move_based.m`: proposed rows of Tables 8–10, Table 7, and the proposed entries of Table 11.
- `ga_kpartition.m`: GA rows and Table 4.
- `tabu_kpartition.m`: TS rows and Table 5.

Corrected proofs

- `klfm_kpartition.m`: KL/FM rows and Table 11.
- `instance_generator.m`: Examples 2 and 3 with seed 2024.
- `build_adjacency_example1.m`: Example 1 adjacency (Table 13).
- `compute_tables.m`: Tables 8–12.
- `Supplementary_Material_S1.csv`: per-run results.
- `proposed_move_based_algorithm.py`: reference implementation.

Funding

The authors conducted this research without any funding, grants, or support.

Conflict of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

All authors contributed to the research and preparation of this manuscript, reviewed the revised text, and approved its final content. All authors accept responsibility for the work.

Artificial Intelligence Statement

ChatGPT (OpenAI) was used to assist with language editing, LaTeX revision, and checking calculations against the supplied supplementary data during manuscript revision. The original experiments were not rerun as part of this assistance. The authors reviewed and verified the revisions and take full responsibility for the manuscript.

Publisher's Note

The publisher remains neutral regarding jurisdictional claims in published maps and institutional affiliations.

References

[1] Abouyee Mehrizi, A., Ghanbari, R., Sadeghi, S., Ghorbani-Moghadam, K. (2024). "Solving continuous quadratic programming for the discrete balanced graph partitioning problem using simulated annealing algorithm hybridized local search and multi-start algorithms". *Applied Soft Computing*, 165, 112109. https://doi.org/10.1016/j.asoc.2024.112109

[2] Adoni, W. Y. H., Nahhal, T., Krichen, M., Aghezzaf, B., Elbyed, A. (2020). "A survey of current challenges in partitioning and processing of graph-structured data in parallel and

Corrected proofs

distributed systems". *Distributed and Parallel Databases*, 38, 495–530. https://doi.org/10.1007/s10619-020-07294-y

[3] Adoni, W. Y. H., Nahhal, T., Krichen, M., El Byed, A., Assayad, I. (2020). "DHPV: A distributed algorithm for large-scale graph partitioning". *Journal of Big Data*, 7, 76. https://doi.org/10.1186/s40537-020-00357-y

[4] Alsaeedi, M., Tavakoli, M., Abouyee Mehrizi, A., Ghorbani Moghadam, K., Ghanbari, R. (2025). "Solution techniques for fuzzy graph partitioning based on heuristic optimization". *Control and Optimization in Applied Mathematics*, 10(2), 273–293. https://doi.org/10.30473/coam.2025.74009.1296

[5] Belotti, P., Buchanan, A., Ezazipour, S. (2025). "Political districting to optimize the Polsby–Popper compactness score with application to voting rights". *Operations Research*, 73(5), 2330–2350. https://doi.org/10.1287/opre.2024.1078

[6] Bichot, C.-E., Siarry, P. (Eds.). (2011). "Graph partitioning". *ISTE Ltd./John Wiley & Sons*. ISBN: 978-1-84821-233-6. https://doi.org/10.1002/9781118601181

[7] Cheng, T., Zeng, G., Wang, S. (2025). "A balanced partitioning method for big graphs via coarsen-partition-refining steps with preserving atomic subgraphs". *Concurrency and Computation: Practice and Experience*, 37(25–26), e70304. https://doi.org/10.1002/cpe.70304

[8] Dubois, D., Prade, H. (1978). "Operations on fuzzy numbers". *International Journal of Systems Science*, 9(6), 613–626. https://doi.org/10.1080/00207727808941724

[9] Firouzian, S., Adabitabar Firozja, M. (2016). "Fuzzy number-valued fuzzy graph". *Control and Optimization in Applied Mathematics*, 1(2), 77–86. https://mathco.journals.pnu.ac.ir/article_3400.html

[10] Garey, M. R., Johnson, D. S., Stockmeyer, L. (1976). "Some simplified NP-complete graph problems". *Theoretical Computer Science*, 1(3), 237–267. https://doi.org/10.1016/0304-3975(76)90059-1

[11] George, A. (1973). "Nested dissection of a regular finite element mesh". *SIAM Journal on Numerical Analysis*, 10(2), 345–363. https://doi.org/10.1137/0710032

[12] Gilbert, J. R., Miller, G. L., Teng, S.-H. (1998). "Geometric mesh partitioning: Implementation and experiments". *SIAM Journal on Scientific Computing*, 19(6), 2091–2110. https://doi.org/10.1137/S106482759631031X

Corrected proofs

[13] Gutiérrez García-Pardo, I., Barroso Pérez, M., Gómez González, D., Castro Cantalejo, J. (2025). "Improving community detection algorithms in directed graphs with fuzzy measures: An application to mobility networks". *Expert Systems with Applications*, 269, 126305. https://doi.org/10.1016/j.eswa.2024.126305

[14] Hager, W. W., Krylyuk, Y. (1999). "Graph partitioning and continuous quadratic programming". *SIAM Journal on Discrete Mathematics*, 12(4), 500–523. https://doi.org/10.1137/S0895480199335829

[15] Hager, W. W., Krylyuk, Y. (2002). "Multiset graph partitioning". *Mathematical Methods of Operations Research*, 55(1), 1–10. https://doi.org/10.1007/s001860200173

[16] Inuiguchi, M., Gao, Z., Henriques, C. O. (2023). "Robust optimality analysis of non-degenerate basic feasible solutions in linear programming problems with fuzzy objective coefficients". *Fuzzy Optimization and Decision Making*, 22(1), 51–79. https://doi.org/10.1007/s10700-022-09383-2

[17] Karypis, G., Kumar, V. (1998). "A fast and high quality multilevel scheme for partitioning irregular graphs". *SIAM Journal on Scientific Computing*, 20(1), 359–392. https://doi.org/10.1137/S1064827595287997

[18] Kernighan, B. W., Lin, S. (1970). "An efficient heuristic procedure for partitioning graphs". *The Bell System Technical Journal*, 49(2), 291–307. https://doi.org/10.1002/j.1538-7305.1970.tb01770.x

[19] Khan, I. U., Aamir, M., Ullah, M., Shah, M. S. (2024). "Fuzzy neural network for fuzzy quadratic programming with penalty function and mean-variance Markowitz portfolio model". *Applied Computational Intelligence and Soft Computing*, 2024, 8694583. https://doi.org/10.1155/2024/8694583

[20] Lu, Z., Chen, L., Gao, J., Hao, J.-K. (2026). "Multilevel iterated tabu search for the multi-constraint graph partitioning problem". *Computers & Operations Research*, 189, 107389. https://doi.org/10.1016/j.cor.2026.107389

[21] Maas, N., Gottesbüren, L., Seemaier, D. (2024). "Parallel unconstrained local search for partitioning irregular graphs". *2024 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, 32–45. https://doi.org/10.1137/1.9781611977929.3

[22] Mahdavi-Amiri, N., Nasseri, S. H. (2007). "Duality results and a dual simplex method for linear programming problems with trapezoidal fuzzy variables". *Fuzzy Sets and Systems*, 158(17), 1961–1978. https://doi.org/10.1016/j.fss.2007.03.018

Corrected proofs

[23] Popa, L. (2023). "A new ranking method for trapezoidal intuitionistic fuzzy numbers and its application to multi-criteria decision making". *International Journal of Computers Communications & Control*, 18(2), 5118. https://doi.org/10.15837/ijccc.2023.2.5118

[24] Rolland, E., Pirkul, H., Glover, F. (1996). "Tabu search for graph partitioning". *Annals of Operations Research*, 63, 209–232. https://doi.org/10.1007/BF02125425

[25] Rosenfeld, A. (1975). "Fuzzy graphs". In L. A. Zadeh, K.-S. Fu, K. Tanaka, M. Shimura (Eds.), *Fuzzy Sets and Their Applications to Cognitive and Decision Processes*, 77–95. Academic Press. https://doi.org/10.1016/B978-0-12-775260-0.50008-6

[26] Sun, W., Liu, H., Wang, X. (2025). "Multi-constraint graph partitioning problems via recursive bipartition algorithm based on subspace minimization conjugate gradient method". *Journal of Applied Mathematics and Computing*, 71(Suppl 1), 1289–1315. https://doi.org/10.1007/s12190-025-02528-7

[27] Yager, R. R. (1981). "A procedure for ordering fuzzy subsets of the unit interval". *Information Sciences*, 24(2), 143–161. https://doi.org/10.1016/0020-0255(81)90017-7

[28] Yager, R. R., Filev, D. (1999). "On ranking fuzzy numbers using valuations". *International Journal of Intelligent Systems*, 14(12), 1249–1268. https://doi.org/10.1002/(SICI)1098-111X(199912)14:12<1249::AID-INT6>3.0.CO;2-#

[29] Yu, Q., Li, W., Yang, D., Zhang, H. (2021). "Partitioning urban road network based on travel speed correlation". *International Journal of Transportation Science and Technology*, 10(2), 97–101. https://doi.org/10.1016/j.ijtst.2020.07.001

[30] Zadeh, L. A. (1965). "Fuzzy sets". *Information and Control*, 8(3), 338–353. https://doi.org/10.1016/S0019-9958(65)90241-X

Authors' Bio-sketches

Fadhil Alsabbagh received his M.Sc. degree in Graph Theory from Ferdowsi University of Mashhad, Iran, under the supervision of Dr. Mostafa Tavakoli. He is currently a Ph.D. candidate in Applied Mathematics at the same university, conducting his doctoral research under the supervision of Prof. Mostafa Tavakoli and Dr. Omid Solaymani Fard. His research interests include graph theory and optimization.

Mostafa Tavakoli received his M.Sc. and Ph.D. degrees in Graph Theory under the supervision of Professor Hassan Yousefi-Azari at the University of Tehran and Professor

Corrected proofs

Freydoon Rahbarnia at Ferdowsi University of Mashhad, respectively. He is currently a Professor at Ferdowsi University of Mashhad, Iran. His research interests include graph theory, combinatorial optimization, and computer science. He also serves as the head of the Graph Theory and Its Applications Laboratory at Ferdowsi University of Mashhad. Corresponding author. Email: m_tavakoli@um.ac.ir.

Ahmad Abouyee Mehrizi received his Ph.D. degree from Ferdowsi University of Mashhad, Iran, under the supervision of Prof. Reza Ghanbari. His research interests include optimization and operations research.

Khatere Ghorbani-Moghadam received her M.Sc. degree from Ferdowsi University of Mashhad and her Ph.D. degree from Sharif University of Technology, Iran. She is currently an Assistant Professor at Kharazmi University, Iran. Her research interests include optimization and operations research.

Omid Solaymani Fard received his Ph.D. degree in Applied Mathematics from Ferdowsi University of Mashhad, Iran, in 2004. He is currently an Associate Professor of Optimization at the same institution. His primary research interests include optimization, fuzzy systems, and machine learning.

Corrected proofs