Graph Coarsening using Game Theoretic Approach

Anonymous authors
Paper under double-blind review

Abstract

Graph coarsening is a method for reducing the size of an original graph while preserving its structural and feature-related properties. In graph machine learning, it is often employed as a preprocessing step to improve efficiency and scalability when handling large graph datasets. In this study, we address the challenge of coarsening an original graph into a coarsened graph that retains these characteristics. We propose a Cooperative-Based Graph Coarsening (CGC) algorithm, which leverages cooperative game theory as a framework for combinatorial optimization, aiming to minimize the total Dirichlet energy of the graph through localized optimizations. We prove that the proposed coarsening game is a potential game that guarantees convergence to a stable coarsened graph. Tests on real-world datasets demonstrate that CGC algorithm surpasses prior state-of-the-art techniques in terms of coarsened graph accuracy and achieves reduced time complexity. These results highlight the potential of game-theoretic approaches in the advancement of graph coarsening techniques.

1 INTRODUCTION

Graphs are powerful data structures used to represent complex relationships and interactions among entities. Graphs offer a natural and flexible way to model real-world systems, which are widely employed across varied domains such as social networks, biological systems, transportation infrastructures, and communication networks Fan et al. (2019); Li et al. (2021); Rahmani et al. (2023); Wu et al. (2022); Shi & Weninger (2017). In these representations, entities are modeled as nodes, while the relationships or interactions among them are captured through edges. This framework enables systematic analysis of both local and global connectivity patterns, as well as the structural characteristics of complex systems Neville & Jensen (2007); Angles & Gutierrez (2008).

With the growing need to extract knowledge from graph-structured data, Graph Neural Networks (GNNs) have emerged as a class of deep learning models specifically designed for learning on graphs. By leveraging both the feature information of nodes and the structure of their neighborhoods, GNNs propagate information through the graph via message passing and aggregation mechanisms. This allows GNNs to capture rich relational patterns and contextual dependencies, making them highly effective for tasks such as node classification, link prediction, and graph classification Corso et al. (2024).

However, as real-world data becomes increasingly voluminous and interconnected, large-scale graphs have become more common, driven by advances in scientific research, online platforms, and the proliferation of connected devices in the Internet of Things. The application of GNNs to such massive graphs introduces significant computational and memory bottlenecks. The recursive neighborhood expansion in message passing leads to rapidly growing computation, while deep GNN architectures suffer from over-smoothing, wherein node representations become indistinct. Additionally, training GNNs often requires extensive hyperparameter tuning, further increasing time and resource demands. These challenges highlight the need for efficient preprocessing strategies to reduce graph size without compromising structural and semantic integrity.

There are various techniques for graph reduction, including graph sparsification Fung et al. (2011); Bravo Hermsdorff & Gunderson (2019); Li et al. (2022), graph coarsening Dickens et al. (2024); Kumar et al. (2023); Loukas (2019); Huang et al. (2021); Loukas & Vandergheynst (2018); Wang et al. (2019), and graph condensation Jin et al. (2021); Xiao et al. (2024); Gao et al. (2025).

Graph sparsification reduces the number of edges while preserving key structural properties such as cuts, pairwise distances, or spectral signatures. However, it retains the full set of nodes, limiting its potential for computational speedup in node-centric operations. Moreover, many sparsification algorithms depend on costly spectral computations (e.g., effective resistance or eigenvalue approximations) and do not always align with downstream task objectives, such as those in GNN-based learning Li et al. (2022).

Graph condensation, in contrast, synthesizes a small, learnable graph from scratch such that a GNN trained on this synthetic graph approximates the performance of one trained on the original graph. While effective at reducing memory usage during GNN training, condensation methods rely on nested or bi-level optimization (e.g., gradient or distribution matching), which incurs substantial preprocessing cost and memory overhead. Furthermore, they often require label information and offer limited interpretability, which hampers generalization across models and datasets.

Graph coarsening offers a balanced alternative by reducing both nodes and edges through the aggregation of structurally or feature-similar nodes into supernodes, while maintaining the overall topology and semantics of the graph Chen et al. (2022). This approach retains a clear mapping between the original and reduced graph, allowing interpretability and compatibility with a variety of downstream GNN tasks. As a result, coarsening enables more scalable and efficient graph learning without sacrificing representational fidelity.

Recent graph coarsening approaches fall into three main categories: heuristic-based methods Loukas (2019); Huang et al. (2021); Loukas & Vandergheynst (2018), optimization-based methods Kumar et al. (2023; 2024), and deep learning-based methods Dickens et al. (2024); Jin et al. (2021). Heuristic methods often operate solely on the adjacency matrix and disregard node attributes, limiting their applicability to feature-rich graphs. For instance, spectral coarsening methods such as those in Loukas (2019); Loukas & Vandergheynst (2018) rely on eigenvalue approximations and may not generalize well to tasks requiring attribute preservation. Some methods, like SCAL Huang et al. (2021), discard supernodes with mixed labels, potentially omitting critical information.

Deep learning-based methods utilizes the graph neural network (GNN) to learn coarsening operations as part of graph representation learning. The coarsened graph is the byproduct of training a GNN for a downstream task or creating synthetic datasets. Ying et al. (2018) used the hierarchical pooling method to group nodes and edges for high-level feature extraction in downstream tasks. Jin et al. (2021) used the gradient matching scheme for graph condensation (GCond) Zhao et al. (2020) where the matching loss between the original set and the synthetic sets is minimized. Zheng et al. (2023) proposed the structure-free graph condensation (SFGC), which combines the gradient matching method and the GNN feature score matrix method. These methods are computationally intensive and do not suit well for generalized GNN as trained on a particular GNN model does not perform well on other models.

Optimization-based methods improve upon these by integrating structural and feature information; e.g., the framework in Kumar et al. (2023) consider both adjacency and feature matrices, while Kumar et al. (2024) further incorporates label information into the coarsening objective. Nonetheless, these methods typically introduce multiple hyperparameters, often four to five, that require thorough dataset-specific tuning. Changes in coarsening ratios or downstream GNN architectures may also necessitate re-optimization, limiting their practicality and generalization.

Overall, while current coarsening methods demonstrate promising performance, there remains a clear need for robust, architecture-agnostic, and computationally efficient coarsening algorithms that can generalize across datasets and models with minimal tuning overhead.

To address the limitations of prior methods, we propose a game-theoretic graph coarsening approach that eliminates the need for extensive hyperparameter tuning. Our method adaptively forms local node coalitions based on their marginal contributions to Dirichlet energy, enabling a data-driven coarsening strategy that maintains consistent performance across diverse datasets and coarsening ratios without manual parameter adjustments. By minimizing Dirichlet energy during coalition formation, the approach preserves structural properties, which can be interpreted as graph cuts, while significantly enhancing computational efficiency and node classification accuracy. The method performs one-hop coarsening and produces a coarsened graph with an approximate coarsening ratio of 0.2 to 0.3, effectively reducing 70–80% of the original graph. For greater reduction, the framework can be extended to n-hop coarsening. We also provide a theoretical guarantee demonstrating that the proposed game constitutes a potential game, ensuring convergence to a Nash equilibrium that defines the coarsened graph, i.e., formed groups are stable, as no node has an incentive to deviate unilaterally from its local neighborhood coalition.

The main contributions of this paper are summarized as follows:

- Introduced a novel graph coarsening framework using game theory.
- Introduced a cost function that utilizes marginal contributions to Dirichlet energy, ensuring structural preservation.
- Empirical validation of the approach on real-world datasets, demonstrating its effectiveness in node classification tasks.
- Presented a theoretical proof demonstrating that the proposed game satisfies the conditions of a potential game.

The remaining paper is structured as follows. Section 2 presents the necessary background on graphs and graph coarsening, followed by the proposed problem formulation and the game-theoretic framework. Section 3 introduces the algorithm along with its convergence guarantee. Section 4 provides experimental results on real-world datasets for node classification, structural property preservation, and comparisons to state-of-the-art methods.

2 Background

This section presents the fundamental concepts of graphs and graph coarsening, including a toy example, followed by the formulation of the proposed problem and its game-theoretic formulation.

2.1 Graph

A simple undirected graph with features and labels, denoted as $G = (V, E, W, X, L)$, is defined by a node set $V = \{V_1, V_2, \dots, V_p\}$ and an edge set $E \subseteq [V]^2$. The topology of the graph is represented using a weighted adjacency matrix $W \in \mathbb{R}^{p\times p}$, where each entry w_{ij} indicates the strength of the connection between the nodes V_i and V_j, and $p = |V|$ indicates the total count of nodes. The matrix $X \in \mathbb{R}^{p\times n}$ encodes node features, with each row x_i representing an n-dimensional attribute vector corresponding to V_i node, and the label is denoted as $L \in \mathbb{R}^{p\times m}$, assigns each node a one-hot encoded categorical label, representing its membership in a predefined set of m classes. To analyze structural properties such as connectivity or spectral characteristics of the graph, we employ the combinatorial Laplacian matrix $\Theta = D - W$, where D is the degree matrix. The Laplacian is symmetric, positive semidefinite, and has zero row sums, making it useful for computational tasks such as graph compression, embedding, manifold learning and spectral preservation. It satisfies $\Theta_{ij} = -W_{ij}$ for $i \neq j$, while diagonal elements represent node degrees. The set of all feasible combinatorial graph Laplacians $\mathcal{L}$ is defined as:

$$\mathcal{L} = \{\Theta \in \mathbb{R}^{m\times m} \mid \Theta = \Theta^T, \Theta_{ij} \leq 0 \text{ for } i \neq j, \Theta_{ii} = -\sum_{j\neq i} \Theta_{ij}\} \tag{1}$$

However, directly analyzing large-scale graphs is computationally expensive, which requires reducing their size while retaining structural properties. In the next subsection, we discuss graph coarsening in detail.

2.2 Graph Coarsening

Graph coarsening aims to construct a smaller and tractable graph $G_c = (V_c, E_c, W_c, X_c, Y_c)$ with k nodes from a larger graph $G = (V, E, W, X, Y)$ with p nodes, while preserving structural and featural properties of the original graph, where $k \ll p$. Each node in G_c, referred to as a supernode, represents a merging of nodes from G grouped based on shared attributes or structural similarities. The coarsening process aims to construct a mapping matrix $C \in \mathbb{R}^{p \times k}$, where $C_{ij} > 0$ signifies that the i-th node of the original graph G is mapped to the j-th supernode of the coarsened graph G_c. This mapping adheres to a many-to-one relationship, as each node in G is assigned to exactly one supernode in G_c, while preserving an orthogonal structure among the supernodes. The matrix C lies in the following set S_c:

$$S_c = \left\{ C \in \mathbb{R}_{+}^{p \times k} \;\middle|\; \langle C_j, C_{j'} \rangle = 0 \ \forall j \neq j';\ \langle C_j, C_j \rangle = d_j,\ \|C_j\|_0 \geq 1,\ C_j \in \mathbb{R}^p;\ \|[C^\top]_i\|_0 = 1,\ [C^\top]_i \in \mathbb{R}^k \right\} \tag{2}$$

Here, $\langle C_j, C_{j'} \rangle$ denotes the standard inner product of the vectors C_j and $C_{j'}$, $\|C_j\|_0$ denotes the number of non-zero entries of a vector (the ℓ_0-norm), C_j and $C_{j'}$ denote the j-th and j'-th columns, and $[C^\top]_j$ represents the j-th row of C of the loading matrix C. The orthogonality condition $\langle C_j, C_{j'} \rangle = 0$ ensures that the columns of C are mutually orthogonal. This condition leads to two constraints: first, the identity $\langle C_j, C_j \rangle = d_j$ captures the aggregated contribution of the nodes assigned to the j-th supernode; second, the constraint $\|[C^\top]_i\|_0 = 1$ enforces that each row of C contains exactly one non-zero entry, meaning that each node in the original graph is associated with a single supernode in the coarsened graph, thereby implementing a hard assignment.

Let $P \in \mathbb{R}^{k \times p}$ denote the coarsening matrix, which serves as the (pseudo-)inverse of the loading matrix C. That is, we define $P = C^\dagger$, where $C^\dagger$ is the Moore-Penrose pseudoinverse of C. This matrix enables the reconstruction (or projection) of coarse-level representations back to the original graph domain. The Laplacian matrix of the original graph Θ, the Laplacian matrix of the coarsened graph Θ_c, the feature matrix of the original graph X and the feature matrix of the coarsened graph X_c, together satisfy the following relations such as:

$$\Theta_c = C^\top \Theta C, \quad X_c = PX, \quad C = P^\dagger, \quad Y_c = \arg\max(PY) \tag{3}$$

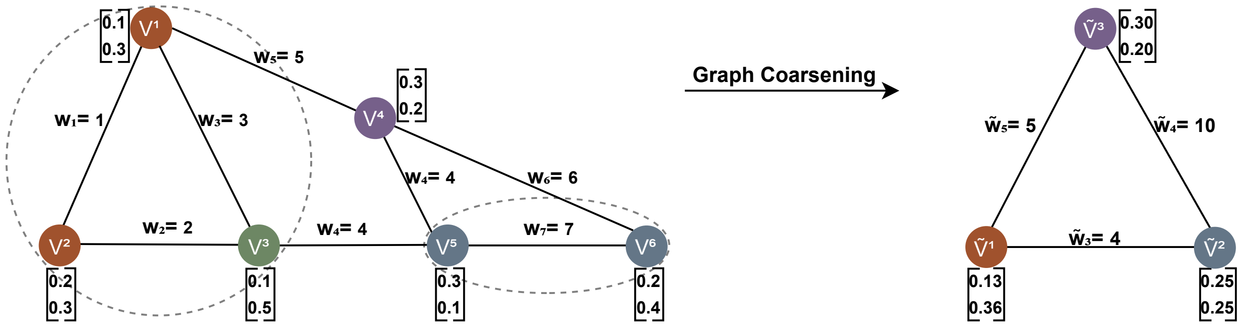


Figure 1: Illustration of the graph coarsening process on a small example.

We demonstrates how node grouping of a simple graph G with six nodes $V = \{V_1, V_2, V_3, V_4, V_5, V_6\}$ through the assignment matrix $\Theta \in \mathbb{R}^{3 \times 6}$ leads to reduced-dimensional graph G_c with nodes $V_c = \{\tilde{V}_1, \tilde{V}_2, \tilde{V}_3\}$ and where each node is assigned a two-dimensional feature vector. The coarsening matrix P is defined below and its pseudoinverse is calculated using $C = P^\dagger$. The feature matrix $X \in \mathbb{R}^{6 \times 2}$ and its coarsened feature matrix $X_c \in \mathbb{R}^{3 \times 2}$ is calculated using $X_c = PX$, as shown in equation (1). Now the original and coarsened Laplacian matrices $\Theta \in \mathbb{R}^{6 \times 6}$ and $\Theta_c \in \mathbb{R}^{3 \times 3}$, computed as $\Theta_c = C^\top \Theta C$:

$$P = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad \Theta = \begin{bmatrix} 9 & -1 & -3 & -5 & 0 & 0 \\ -1 & 3 & -2 & 0 & 0 & 0 \\ -3 & -2 & 9 & 0 & -4 & 0 \\ -5 & 0 & 0 & 15 & -4 & -6 \\ 0 & 0 & -4 & -4 & 15 & -7 \\ 0 & 0 & 0 & -6 & -7 & 13 \end{bmatrix} \quad \text{and} \quad \Theta_c = C^\top \Theta C = \begin{bmatrix} 9 & -4 & -5 \\ -4 & 14 & -10 \\ -5 & -10 & 15 \end{bmatrix} \tag{4}$$

To ensure meaningful coarsening, modern methods optimize the mapping matrix C under constraints such as sparsity, orthogonality, and balance among supernodes. These techniques incorporate the characteristic matrix X, and in some cases, both X and the label information Y, together with the adjacency matrix, to generate coarser graphs more informative. The resulting graph facilitates downstream tasks while maintaining the structural and spectral fidelity of the original graph. There exist various graph coarsening techniques, such as optimization-based, deep learning-based, and heuristic-based methods as follows:

2.3 Proposed Problem Formulation

Given a weighted adjacency matrix $W \in \mathbb{R}^{p \times p}$ and a feature matrix $X \in \mathbb{R}^{p \times n}$ corresponding to the node set $V = \{V_1, V_2, \ldots, V_p\}$ of the original graph, we model the graph coarsening problem as a cooperative game. In this formulation, graph nodes are treated as players, and the edges associated with each node define the possible actions available to that player. The game aims to study how groups of players (nodes) form coalitions (supernodes) to achieve a common goal and fairly distribute outcomes among the players. Throughout this paper, we use the terms player and node interchangeably. The goal of each coalition in this cooperative game is to minimize the cost of each player. The objective is to determine a set of coalitions $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$, where each coalition $S_i \subseteq V$, that collectively minimizes the overall Dirichlet energy of the coarsened graph and is formulated as

$$\min_{\mathcal{S} \in \mathcal{S}_T} \sum_{\substack{S_i, S_j \in \mathcal{S} \\ i<j}} \tilde{w}_{ij} \|\tilde{x}_i - \tilde{x}_j\|^2 \quad \text{s.t.} \quad \mathcal{S}_T = \left\{ \mathcal{S} \mid \bigcup_{i=1}^{K} S_i = V,\ S_i \cap S_j = \emptyset\ \forall i \neq j \right\} \tag{5a}$$

$$\tilde{w}_{ij} = \sum_{u \in S_i, v \in S_j} w_{uv}, \quad \tilde{x}_i = \frac{1}{|S_i|} \sum_{v \in S_i} x_v, \quad \forall i \neq j,\ \forall i,\ \forall u, v \in S_i,\ \text{path}(u, v) \leq 2 \tag{5b}$$

where each $S_i \in \mathcal{S}$ denotes a subset of nodes mapped to a corresponding supernode $\tilde{V}_i$ and here i represents the ith supernode. The term w_{ij} refers to the (i, j)-th entry of the original graph weight matrix W, while $\tilde{w}_{ij}$ denotes the (i, j)-th entry of the coarsened weight matrix W_c, representing the weight of the edge between supernodes $\tilde{V}_i$ and $\tilde{V}_j$. Similarly, x_i denotes the i-th row of the original feature matrix X, and $\tilde{x}_i$ denotes the i-th row of the coarsened feature matrix X_c, representing the feature vector of the supernode $\tilde{V}_i$. The term $\|\tilde{x}_i - \tilde{x}_j\|^2$ quantifies the local variation in the feature signal between the supernodes, and the summation aggregates this variation between all pairs of connected supernodes. The constraint S_T defines the coalition structure, ensuring a complete and disjoint partition of the node set V into K subsets. $|S_i|$ is the cardinality (size) of the nodes in set S_i. The path(u, v) defines the length of the shortest path between two nodes u and v within the same subset S_i and constraint by most two in the original graph. This constraint ensures that each coalition forms a connected subgraph with mutual reachability. By minimizing the objective function defined in equation (5a), the resulting supernodes exhibit smooth variations and preserve the structure of the original graph while ensuring energy-efficient coarsening. Minimizing this problem is NP-hard, as it generalizes diameter-constrained graph partitioning as descibed in Zhang et al.. We formulate the graph coarsening task as a cooperative game such that a set of coalitions $\mathcal{S} = \{S_1, S_2, \ldots, S_K\}$ by incorporating structural constraints. This new interpretation enables game-theoretic reasoning over graph structures. The formal definition of this game is presented in the next subsection.

2.4 Game setup

In this work, we apply cooperative game theory to address the problem of graph coarsening. Cooperative game theory, as defined by v Neumann & Morgenstern (1953), is a mathematical framework in which a group of players interacts to achieve a common goal by forming coalitions based on binding agreements. We adopt this framework to model graph coarsening as a game, defined as CGame $= (N, v)$, where $N = \{1, 2, \ldots, p\}$ denotes the set of players (graph nodes), and v is the characteristic function that assigns a transferable total cost value to each feasible coalition. Each player $i \in N$ has a feature vector x_i and a set of neighboring nodes $A_i = \{a_1, a_2, \ldots, a_m\}$, where m denotes the number of neighbors of node i. The members of the set A_i will be the potential collaborators of player i for coalition formation, from which a feasible subset can be selected based on cost and structural constraints.

To model coalition dynamics and cost distribution in the CGame framework, we define the mapping profile, characteristic function, and player cost. The mapping profile is defined as a tuple $m = (m_1, m_2, \dots, m_p)$, where each m_i indicates the coalition $S_j \in \mathcal{S}$ to which node i is assigned based on the merging decision. This mapping induces a set of supernodes $V_c = \{\tilde{V}_1, \tilde{V}_2, \dots, \tilde{V}_k\}$, where k is the number of distinct coalitions formed. The profile m thus specifies the complete node-to-supernode assignment. For a particular node i, the mapping can be expressed as (m_i, m_{-i}), where m_{-i} represents the mapping decisions of all nodes except i. The set of nodes assigned to coalition S_i under mapping m is denoted by $R^m_{S_i}$, and the coalition S_i excluding node j under mapping m is represented by $R^m_{S_i} \setminus \{j\}$. The characteristic function v of $R^m_{S_i}$ is defined as:

$$v(R^m_{S_i}) = \begin{cases} \mathrm{DE}(R^m_{S_i}), & \text{if } |R^m_{S_i}| = 1 \\ \sum_{j \in R^m_{S_i}} c^m_{S_i}(j), & \text{if } |R^m_{S_i}| > 1 \end{cases} \tag{6}$$

$$\text{where, } \mathrm{DE}(R^m_{S_i}) = \frac{1}{2} \sum_{(u,v) \in E_x} w_{uv} \|x_u - x_v\|^2, \quad c^m_{S_i}(j) = \mathrm{DE}(R^m_{S_i}) - \mathrm{DE}(R^m_{S_i} \setminus \{j\}) \tag{7}$$

Where, $DE(R^m_{S_i})$ represents the contribution of Dirichlet energy of the i th supernode or coalition S_i, formed under the mapping profile m and $c^m_{S_i}(j)$ denotes the cost of node $j \in R^m_{S_i}$, defined as the marginal increase in Dirichlet energy due to its presence in coaliton S_i. E_x denotes the set of edges connected to the ith supernode on the coarsened graph.

Given the characteristic function v and the individual player cost $c(j)$ in terms of cost, we now analyze the strategic behavior of players to reach the convergence state of the game, or a stable coalition structure. A coalition structure is considered stable if it satisfies two conditions: internal stability and external stability. A coalition is internally stable if no subset of its members can decrease their total allocated cost by deviating to form a new coalition, as expressed in equation (8).

$$\sum_{i \in T} c_i \le v(T), \quad \forall T \subseteq S \tag{8}$$

A coalition is externally stable if no player has an incentive to deviate from their current coalition to join another feasible coalition. Let Γ_i represent the set of feasible coalitions available to the player i, that is, coalitions for which the player's inclusion does not increase their cost or violate coalition constraints. Then, the optimal strategy for the player i, denoted by m_i^*, is defined as the best response that minimizes their individual cost, as shown in equation (9).

$$m_i^* = \arg\min_{m_i \in \Gamma_i} c_i(m_i, m_{-i}) \tag{9}$$

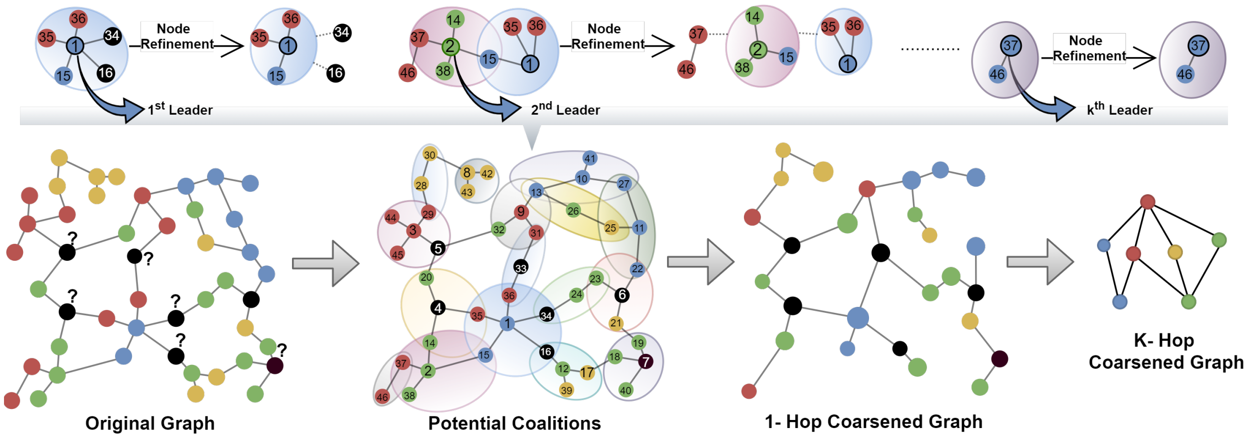


Figure 2: The figure illustrates the step-by-step process of cooperative game-based graph coarsening. It depicts coalition formation based on degree (as leaders), followed by refinement using the cost of each node within a coalition to achieve stability. The input to the CGC algorithm is the original graph $G(V, E, X, L)$, where some nodes may not have label information.

where c_i denotes the cost allocated to player i, which can be expressed as a function $c_i(m_i, m_{-i})$ representing the cost when player i selects the mapping m_i while others choose a fixed profile m_{-i}. Here, $v(T)$ is the value of subset T given by the characteristic function. In the next subsection, we present the CGC algorithm that ensures convergence to this equilibrium state.

3 ALGORITHM DEVELOPMENT

In CGame, each player joins a coalition to minimize individual cost with respect to the current coalition structure. The contribution of each player within a coalition is then computed. Players follow best-response decisions, leading the system to converge toward a stable coalition structure where no player has an incentive to deviate. To implement this process, we introduce the cooperative game-based graph coarsening (CGC) algorithm.

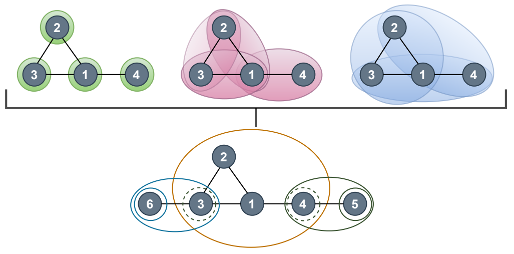
Algorithm 1 Cooperative game-based graph coarsening (CGC) using one-hop neighbor exploration

```
1: Input: Graph $G(V, E, X)$
2: Initialize: $c_{cost} \leftarrow \{i : Dn[i],\ \forall i \in V\}$, $S \leftarrow \{\{i\}, \forall i \in V\}$, $S_{coalition}, leaderSet \leftarrow \emptyset$
3: Sort $V$ by degree in descending order
4: repeat
5:   $S_{\text{prev}} \leftarrow S$
6:   if $leaderSet \neq \emptyset$ then $V \leftarrow leaderSet$
7:   for each node $i \in V$ do
8:     if $i \notin S_{\text{coalition}}$ and $i$ has edges then
9:       $S_i \leftarrow \{i\} \cup \text{neighbors}(i)$
10:      Compute each node cost $c_{S_i}(j)$ for all $j \in S_i$ (equation (7)) and
11:      set coalition cost $v(S_i) = \sum_{j \in S_i} c_{S_i}(j)$ (equation (6)).
12:      if $v(S_i) < \sum_{j \in S_i} c_{\text{cost}}(j)$ then
13:        for each node $j \in S_i$ do
14:          if $c_{S_i}(j) > c_{cost}(j)$ then Remove $j$ from $S_i$
15:        end for
16:      else $S_i \leftarrow \emptyset$
17:      end if
18:      if $|S_i| > 1$ then $leaderSet \leftarrow leaderSet \cup \{i\}$, $S_{\text{coalition}} \leftarrow S_i \cup S_i$, update $S$, $c_{\text{cost}}$
19:    end if
20:  end for
21: until $S = S_{\text{prev}}$
22: Output: Loading matrix C (constructed based on S)
```

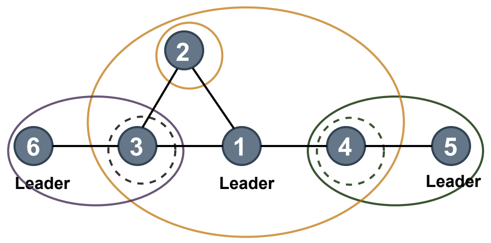
Given a graph $G = (V, E, X)$ with a weighted adjacency matrix $A \in \mathbb{R}^{|V| \times |V|}$ and node features $X \in \mathbb{R}^{|V| \times d}$, CGC algorithm employs an iterative procedure to minimize the cost of each player and converge to a stable coalition structure described in algorithm 1. The algorithm begins by initializing each node as a singleton coalition, and in this configuration, its cost is calculated from equation (7) (line 2).

In the coalition formation step, we begin by forming a local coalition with a node and its neighbors. To check the stability of this coalition, we evaluate whether any individual node or group of nodes within the coalition has an incentive to deviate and form a separate coalition that would reduce their cost as defined in equation (7). This requires checking whether any subset of the current coalition can deviate as a best response to obtain a lower cost. However, exhaustively evaluating all subsets is computationally expensive for large coalitions. To improve convergence speed and emphasize structurally important nodes, we incorporate a leader-based coalition formation strategy. In this strategy, a coalition is only valid if it includes a designated leader; otherwise, it is discarded. The nodes with the highest degree of centrality are prioritized as leaders, ensuring that highly connected nodes initiate the formation of coalitions.

The formation process commences by sorting the nodes in descending order of degree (line 3), since high-degree nodes are more influential in the network, and their early consideration ensures that major connectivity



(a) All Possible Deviating Subcoalitions in the 1-Hop Neighborhoods of Nodes 1, 5, and 6 (Without Leader-Based Formation)



(b) All Possible Deviating Subcoalitions in the 1-Hop Neighborhoods of Nodes 1, 5, and 6 (Leader-Based Formation)

Figure 3: Comparison of possible internal deviations in coalitions formed using leaderless and leader-based methods. Circles indicate all possible subgroups that can break the coalition, while dotted circles highlight nodes that may deviate into alternate coalitions.

hubs are prioritized during coalition formation. Then each node is sequentially evaluated as a potential leader. A potential leader attempts to form a preliminary coalition by selecting its immediate one-hop neighbors as S_i. The feasibility of a potential coalition S_i is evaluated by checking whether $v(S_i) \leq \sum_{i \in S_i} v(\{i\})$, i.e., its total cost is less than or equal to the sum of the costs of its members when acting individually. If S_i is not feasible, the potential leader is retained as a singleton coalition, and the algorithm proceeds to the next leader. Otherwise, if S_i is feasible, the tentative coalition undergoes a refinement phase (lines 17–21), where each node's individual cost is compared against its previous configuration, whether as a singleton or as part of another coalition. Only nodes whose cost is reduced by joining the coalition are retained. If the refined coalition contains more than one member, it is accepted, and the initiating node is confirmed as the coalition leader (lines 17). The next leader is selected from the remaining nodes in the sorted list, skipping those already assigned to an existing multinode coalition (line 8).

In non-leader-based coalition refinement, the problem involves exploring all possible subsets of a coalition to check for deviations. We consider the Stable Coalition problem, which asks whether there exists a coalition satisfying the internal stability condition, as defined in equation (8). We prove in Theorem 1 that this problem is NP-complete. To illustrate the efficiency of our approach, consider the example shown in Figure 3a. In the leader-based coalition refinement, we adopt a heuristic approach to address the Stable Coalition problem, with a complexity of $O(\Delta)$ for checking the stability of a single coalition and $O(m \cdot \Delta)$ for verifying stability across all m coalitions, where Δ denotes the average degree of the graph and m represents the number of coalitions. This improvement is illustrated in Figure 3b, with further details provided in Appendix A.2. The performance in leader-based coalition formation will not be affected. Once the coalition cost is distributed among players, each player can unilaterally deviate under best-response dynamics whenever it is beneficial. Moreover, each such deviation decreases the overall system cost, as demonstrated in the next subsection, where we prove that this game is a potential game.

Theorem 1. *The* Stable Coalition *problem is NP-complete.*

Proof. The proof of Theorem 1 has been given in Appendix A.1

The cost allocated to individual nodes within the coalition is determined using the Dirichlet energy-based marginal contribution method defined in equation (7), which measures the incremental impact of a node on the total value of the coalition (lines 10). After computing the costs, each node finds its best mapping decision according to equation (8). If its cost within the coalition exceeds the cost from its previous configuration, it is removed from the coalition. This ensures that participation is beneficial and rational for all members of the final coalition structure. As per algorithm 1, the entire process is repeated iteratively until the coalition structure S reaches equilibrium, where no further beneficial changes in the mapping decision are possible. At this point, the configuration is considered stable as no node has an incentive to switch coalitions, leave, or form new coalitions.

The proposed method dynamically determines the final size of the coarsened graph based on the number of coalitions formed at the stable state. To enable deeper reductions while preserving structural locality, we extend our approach via a recursive multi-level strategy. In this setting, the coarsened graph from one iteration of the CGC algorithm 1 becomes the input for the next. The features of the coarsened graph at each level is calculated using $X_c = PX$. Although each level performs only one-hop neighborhood exploration, the overall effect of k such iterations achieves an effective k-hop coarsening of the original graph. The algorithm for multi-level coarsening is provided in Appendix A.3.

Table 1: Performance comparison across different GNN models (GCN, GAT, APPNP) on datasets including Cora, DBLP, CoCS, Pubmed, Co-Physics, Genius, and OGBN-Arxiv, under various coarsening ratios using the proposed CGC algorithm. Each value represents the mean accuracy ± standard deviation over 10 runs.

Dataset	**r=k/p**	**GCN**	**GAT**	**APPNP**
Cora	0.21	87.96 ± 0.05	86.74 ± 0.02	89.07 ± 0.03
	0.05	80.57 ± 0.07	80.95 ± 0.04	82.00 ± 0.02
DBLP	0.23	84.75 ± 0.12	84.00 ± 0.06	84.97 ± 0.08
	0.05	80.74 ± 0.06	83.29 ± 0.05	83.43 ± 0.04
	0.01	76.77 ± 0.08	78.45 ± 0.06	80.24 ± 0.05
CoCS	0.14	92.53 ± 0.09	89.36 ± 0.07	93.47 ± 0.05
	0.03	91.13 ± 0.03	88.34 ± 0.04	91.15 ± 0.02
	0.01	85.66 ± 0.05	83.82 ± 0.06	88.27 ± 0.03
Pubmed	0.235	86.84 ± 0.04	78.63 ± 0.05	85.22 ± 0.06
	0.07	86.02 ± 0.10	76.20 ± 0.08	85.74 ± 0.07
	0.02	83.41 ± 0.09	78.31 ± 0.07	83.99 ± 0.05
Co-Physics	0.09	94.79 ± 0.11	92.11 ± 0.10	95.85 ± 0.08
	0.01	94.15 ± 0.02	93.51 ± 0.03	95.14 ± 0.01
	0.005	92.43 ± 0.07	92.03 ± 0.05	93.94 ± 0.04
Genius	0.06	80.00 ± 0.03	79.84 ± 0.34	79.78 ± 0.51
	0.02	79.97 ± 0.08	79.98 ± 0.04	79.96 ± 0.09
	0.009	79.91 ± 0.14	80.00 ± 0.00	79.15 ± 1.34
	0.007	78.63 ± 0.03	80.00 ± 0.00	79.60 ± 0.73
OGBN-Arxiv	0.15	53.83 ± 0.23	51.14 ± 0.13	51.07 ± 0.20
	0.04	52.81 ± 0.33	49.87 ± 0.25	49.92 ± 0.28
	0.01	50.91 ± 0.42	48.06 ± 0.28	47.93 ± 0.58
	0.004	46.86 ± 0.50	45.21 ± 0.28	45.13 ± 0.58
	0.001	42.01 ± 0.56	40.44 ± 0.58	39.25 ± 1.29

4 ANALYSIS OF THE ALGORITHM

4.1 Theoretical Analysis

This section analyzes the convergence of the CGC algorithm to an equilibrium state (where no coalition can reduce the cost by unilateral deviation) within CGame. To assess the existence of an equilibrium, it is essential to examine the convergence behavior of the CGC algorithm. This can be achieved by demonstrating that the CGC algorithm of CGame constitutes a potential game. Before presenting the proof, we first introduce the concept of a potential game and define the potential function used in our analysis.

Definition 1. Potential Game *A cooperative game can be regarded as a potential game if there exists a global scalar function Φ, called the potential function, that captures the alignment of individual cost changes with a collective objective. Formally, for any player $i \in N$, and any two mapping decisions $m_i, m_i' \in A_i$, with a fixed mapping configuration of other players $m_{-i} \in \prod_{j\neq i} A_j$, the following implication holds:*

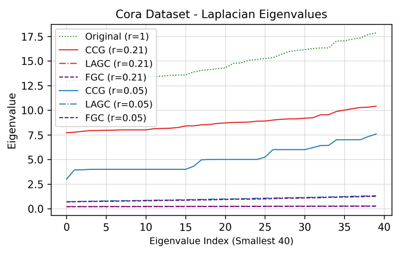
$$c_i(m_i', m_{-i}) - c_i(m_i, m_{-i}) \leq 0 \quad \Rightarrow \quad \Phi(m_i', m_{-i}) - \Phi(m_i, m_{-i}) \leq 0 \tag{10}$$

where, c_i represents the cost incurred by node i. This condition ensures that a reduction in an individual node's cost corresponds to a non-increasing global potential, thereby aligning local choices with a global optimization goal.

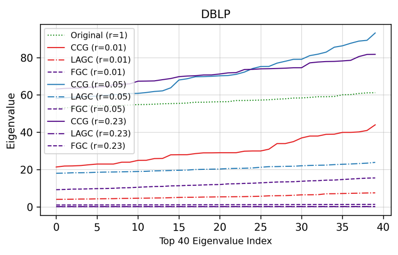
The total potential of the CGame can be evaluated by aggregating the individual potentials of each coalition. When a node is mapped to a coalition, it incurs a cost c, which contributes to the coalition's potential. Accordingly, we compute the potential of each coalition. The potential of coalition S_i under mapping profile m, denoted by $\phi(R^m_{S_i})$, is defined as follows:

Calculation of Coalition-Level Potential: Given a coalition S_i under a specific mapping profile m, its potential contribution $\phi(R^m_{S_i})$ is defined as:

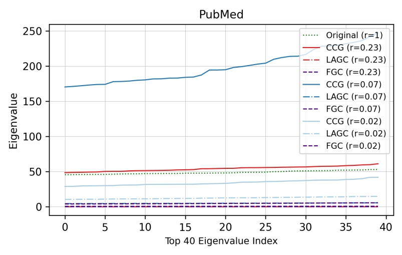
$$\phi(R^m_{S_i}) = \mathrm{DE}(R^m_{S_i}) = \frac{1}{2} \sum_{(u,v)\in E_{S_i}} w_{uv} \|x_u - x_v\|^2, \tag{11}$$



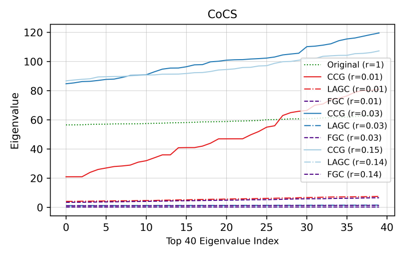
(a) Cora



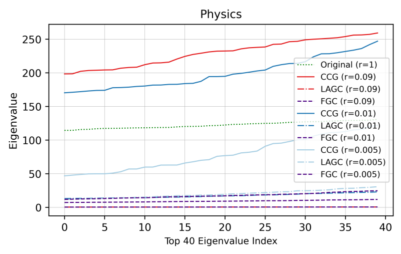
(b) DBLP



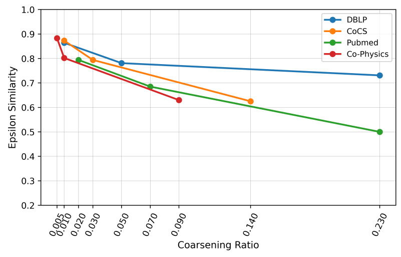
(c) Pubmed



(d) CoCS



(e) Co-Physics



(f) ϵ-similarity

Figure 4: This figure plots the top 40 eigenvalues of the Laplacian matrix for both the original and coarsened graphs across various datasets: (a) Cora, (b) DBLP, (c) PubMed, (d) CoCS, and (e) Co-Physics, under different coarsening ratios. Figure (f) shows the corresponding ϵ-similarity values for the same datasets. The values of ϵ, which range from 0 to 1, shows the similarity between the original graph G and the coarse graph G_c generated by the proposed CGC algorithm, with lower values implying a higher structural similarity.

where E_{S_i} denotes the set of edges connected to the ith supernode corresponding to S_i th coalition on the coarsened graph, w_{uv} is the weight on edge (u, v), and x_u, x_v are feature vector of nodes u and v under m.

Global Potential $\Phi(m)$: Let $\mathcal{S} = \{S_1, S_2, \ldots, S_k\}$ represent the set of all coalitions. This global potential aggregates the contributions of all coalitions, providing a holistic measure of the graph reduction configuration. The total potential of CGame under a mapping profile m is given by:

$$\Phi(m) = \sum_{S_i \in \mathcal{S}} \phi(R^m_{S_i}). \tag{12}$$

Where, $\phi(R^m_{S_i})$ is the Coalition-Level Potential as defined in equation (11)

Theorem 2. *The function $\Phi(m)$ satisfies the conditions of a potential function. Therefore, CGame constitutes a potential game.*

Proof. The proof of Theorem 2 has been given in Appendix A.5

Theorem 3. *At least one Pure Nash Equilibrium (PNE) is ensured in the proposed CGame framework when operating under the CGC algorithm.*

Proof. The proof of Theorem 3 has been given in Appendix A.5

4.2 Numerical Analysis

In this section, we first describe the experimental setup, a detailed overview of the datasets, followed by analyzing the proposed algorithm's experimental results on real graph data sets which evaluate its performance across various graph neural network models, and then compare it with state-of-the-art methods to showcase the effectiveness of the CGC algorithm.

Table 2: Node classification accuracy on various real-world datasets using the proposed CGC algorithm, compared against state-of-the-art methods SCAL, FGC, and LAGC. Experiments are conducted under different coarsening ratios ($r = k/p$). Each value represents the mean accuracy ± standard deviation computed over 10 runs. The results demonstrate that CGC consistently achieves superior performance across a wide range of datasets and coarsening levels. OOM indicates cases where execution exceeded memory.

Dataset	$r = k/p$	SCAL	FGC	LAGC	CGC	Whole Data
CORA	0.21	80.08 ± 1.02	83.33 ± 0.02	84.01 ± 0.13	87.96 ± 0.32	89.50 ± 1.20
	0.05	54.07 ± 0.94	67.98 ± 1.77	71.73 ± 0.13	80.57 ± 0.21	
DBLP	0.23	75.09 ± 1.59	81.47 ± 0.13	81.31 ± 0.75	84.75 ± 0.54	85.35 ± 0.86
	0.05	76.38 ± 1.83	74.75 ± 0.10	75.53 ± 0.65	80.74 ± 0.43	
	0.01	68.70 ± 4.50	69.28 ± 0.06	69.90 ± 0.69	76.77 ± 0.12	
CO-CS	0.14	81.41 ± 3.15	91.37 ± 0.00	90.01 ± 0.82	92.53 ± 0.67	93.32 ± 0.62
	0.03	50.46 ± 5.43	78.74 ± 0.04	84.71 ± 0.25	91.13 ± 0.31	
	0.01	25.20 ± 7.32	83.13 ± 0.01	77.29 ± 0.13	85.66 ± 0.45	
PUBMED	0.235	78.02 ± 0.48	81.71 ± 0.09	83.21 ± 0.71	86.84 ± 0.11	88.89 ± 0.57
	0.07	73.73 ± 0.90	69.37 ± 0.06	77.24 ± 0.81	86.02 ± 0.28	
	0.02	64.71 ± 4.68	71.10 ± 1.93	76.37 ± 0.05	83.41 ± 0.72	
CO-PHYSICS	0.09	90.68 ± 1.03	87.25 ± 0.40	87.11 ± 0.03	94.79 ± 0.19	96.22 ± 0.74
	0.01	31.08 ± 10.02	91.38 ± 0.12	91.11 ± 0.19	94.15 ± 0.37	
	0.005	24.92 ± 2.97	88.01 ± 0.29	89.26 ± 0.03	92.43 ± 0.58	
Genius	0.06	OOM	OOM	OOM	80.00 ± 0.03	86.71 ± 0.12
	0.02	OOM	OOM	OOM	79.97 ± 0.08	
	0.009	OOM	OOM	OOM	79.91 ± 0.14	
	0.007	OOM	OOM	OOM	78.63 ± 0.03	
Obgn arxiv	0.15	OOM	OOM	OOM	53.83 ± 0.23	56.24 ± 0.22
	0.04	OOM	OOM	OOM	52.81 ± 0.33	
	0.01	OOM	OOM	OOM	50.91 ± 0.42	
	0.004	OOM	OOM	OOM	46.86 ± 0.50	
	0.001	OOM	OOM	OOM	42.01 ± 0.56	

Experimental Setup: Experiments were conducted on a system running `Ubuntu 18.04.6 LTS` (x86_64) with dual `Intel Xeon E5-2630 v4` CPUs (40 logical cores, 2.20 GHz) and 247 GB RAM.

Dataset We have evaluated our method on real-world datasets, including Cora, DBLP, CoCS, PubMed, Co-Physics, OGBN-Arxiv, and Genius. Detailed data set statistics are provided in Table A.4 in the appendix.

Baseline models: We evaluate the performance of the CGC algorithm (proposed) by its output as a coarsened graph as the input for the GNN model for node classification task. To ensure consistency, we adopt GNN architectures similar to those in the baseline study, including GCN Kipf & Welling (2016), GAT Veličković et al. (2017), and APPNP Gasteiger et al. (2018). We compare the performance of the proposed CGC algorithm with state-of-the-art methods, including SCAL Huang et al. (2021), FGC Kumar et al. (2023), and LAGC Kumar et al. (2024), in terms of node classification accuracy. Experiments conducted on real-world datasets demonstrate that our model surpasses existing state-of-the-art methods in both node classification accuracy and computational efficiency.

Node Classification For the node classification task on a coarsened graph, we implement the proposed CGC algorithm to generate a coarsened version of the original graph $G(\Theta, X, Y)$ without utilizing its node labels. After obtaining the coarsened graph $G_c(\Theta_c, X_c)$, we infer its node labels using $Y_c = \arg\max(PY)$, where P represents the pseudoinverse of the mapping matrix C. We then train Graph neural Network (GNN) models on $G_c(\Theta_c, X_c, Y_c)$ and evaluate its performance on the remaining 20% of nodes, excluding their labels during the coarsening process. Furthermore, we perform node classification on the original graph and compare the results with those obtained from the coarsened graph. To ensure an equitable comparison, we use the same data split, allocating 80% of the node labels for training and the remaining 20% for testing. It is evident from Table 1 that the node classification performance remains consistent across different GNN models (GCN, GAT, APPNP) on datasets including Cora, DBLP, CoCS, PubMed, Co-Physics, Genius, and OGBN-Arxiv under various coarsening ratios, demonstrating the robustness and effectiveness of the proposed CGC algorithm. Furthermore, we compare the node classification performance of the GCN model using our proposed CGC method against state-of-the-art approaches including SCAL, FGC, and LAGC. The results, summarized in Table 2, demonstrate that CGC consistently outperforms these baseline methods across various datasets and coarsening ratios.

Structural Properties: To assess the structural preservation of the coarsened graph, we evaluate spectral and smoothness properties using the Relative Eigenvalue Error (REE) and ϵ-similarity metrics. **REE** is defined as $\frac{1}{q}\sum_{i=1}^{q}\frac{|\tilde{\lambda}_i-\lambda_i|}{\lambda_i}$, measures the spectral deviation between the original and coarsened Laplacian matrices Θ and Θ_c over the smallest q eigenvalues, where λ_i and $\tilde{\lambda}_i$ denote the i-th eigenvalues of Θ and Θ_c, respectively. Lower REE indicates better spectral preservation. It is evident from Figure 4 that the plots of the top 40 eigenvalues of the Laplacian matrix for both the original and coarsened graphs across various datasets (a) Cora, (b) DBLP, (c) PubMed, (d) CoCS, and (e) Co-Physics under different coarsening ratios, closely approximate the original spectra and demonstrate improved alignment compared to state-of-the-art algorithms. For smoothness, **ϵ-similarity** requires that $(1-\epsilon)\,\mathrm{tr}(X^\top\Theta X) \le \mathrm{tr}(\tilde{X}^\top\Theta_c\tilde{X}) \le (1+\epsilon)\,\mathrm{tr}(X^\top\Theta X)$, where $\mathrm{tr}(X^\top\Theta X)$ and $\mathrm{tr}(\tilde{X}^\top\Theta_c\tilde{X})$ denote the Dirichlet energy of the original and coarsened graphs, respectively. This ensures that the Dirichlet energy of the coarsened graph remains close to that of the original, indicating smoothness preservation. It is evident in Figure 4(f) that the ϵ-similarity for the DNLP, CoCS, PubMed, and CoPhysics datasets across different coarsening ratios r remains within valid range of 0 to 1, indicating reliable smoothness preservation.

Table 3: Runtime comparison of the proposed CGC algorithm with baseline algorithms SCAL, FGC, and LAGC across various real-world datasets and coarsening ratios ($r = k/p$). Time(in seconds) includes only the graph coarsening step. The results demonstrate that CGC is computationally efficient and scales well across different datasets.

Dataset	**r**	**SCAL**	**FGC**	**LAGC**	**CGC**
Cora	0.21	2.28	18.90	9	0.35
	0.05	6.58	7.50	9	0.35
DBLP	0.23	85.02	2818.14	1980	14
	0.05	65.96	581.31	309	15
	0.01	64.27	169.28	84	15
CoCS	0.14	23.13	3398.42	1758	13
	0.03	27.38	704.16	357	15
	0.01	45.33	261.51	126	15
Pubmed	0.235	53.37	2609.87	4380	23
	0.07	59.75	640.52	569	24
	0.02	61.34	167.48	202	24
Co-Physics	0.09	58.98	6647.98	10078	58
	0.01	110.38	914.43	585	67
	0.005	155.63	634.21	422	67
Genius	0.06	-	-	-	7019
	0.02	-	-	-	9011
	0.009	-	-	-	9613
	0.007	-	-	-	9720
OGBN-Arxiv	0.15	-	-	-	1503
	0.04	-	-	-	2579
	0.01	-	-	-	2877
	0.004	-	-	-	2912
	0.001	-	-	-	2924

Time complexity: Given a graph with p nodes and an average degree Δ, the time complexity of the proposed coarsening algorithm is $\mathcal{O}(p\cdot\Delta\cdot k)$, where k denotes the number of nodes of the coarsened graph. This computational efficiency arises from the fact that each node interacts only with its neighbors and that the coarsening process significantly reduces the size of the problem. As demonstrated in Table 3, the proposed CGC algorithm achieves faster execution times (in seconds) compared to the baseline methods such as SCAL, FGC, and LAGC in most coarsening ratios.

5 CONCLUSION

This paper introduced the CGC algorithm, a cooperative game-theoretic approach to graph coarsening that utilizes node features and the weighted adjacency matrix. We defined the coalition cost using the Dirichlet energy. We measured each player's cost as its marginal contribution, which is the change in energy that results from including or excluding the player from the coalition. We formally proved that the CGC algorithm operates within the CGame framework as a potential game, guaranteeing convergence to a stable solution. Since the potential function is defined as the sum of Dirichlet energies of all coalitions, the final stable state corresponds to a minimum total energy configuration. Empirical results on node classification tasks show that our method achieves superior accuracy and significantly lower coarsening time than state-of-the-art baselines. Overall, the CGC algorithm provides a computationally efficient and effective solution for graph coarsening, demonstrating both theoretical soundness and practical advantages.

References

Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM computing surveys (CSUR)*, 40(1):1–39, 2008.

Gecia Bravo Hermsdorff and Lee Gunderson. A unifying framework for spectrum-preserving graph sparsification and coarsening. *Advances in Neural Information Processing Systems*, 32, 2019.

Jie Chen, Yousef Saad, and Zechen Zhang. Graph coarsening: from scientific computing to machine learning. *SeMA Journal*, 79(1):187–223, 2022.

Gabriele Corso, Hannes Stark, Stefanie Jegelka, Tommi Jaakkola, and Regina Barzilay. Graph neural networks. *Nature Reviews Methods Primers*, 4(1):17, 2024.

Charles Dickens, Edward Huang, Aishwarya Reganti, Jiong Zhu, Karthik Subbian, and Danai Koutra. Graph coarsening via convolution matching for scalable graph neural network training. In *Companion Proceedings of the ACM on Web Conference 2024*, pp. 1502–1510, 2024.

Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation. In *The world wide web conference*, pp. 417–426, 2019.

Wai Shing Fung, Ramesh Hariharan, Nicholas JA Harvey, and Debmalya Panigrahi. A general framework for graph sparsification. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pp. 71–80, 2011.

Xinyi Gao, Junliang Yu, Tong Chen, Guanhua Ye, Wentao Zhang, and Hongzhi Yin. Graph condensation: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Johannes Gasteiger, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Zengfeng Huang, Shengzhong Zhang, Chong Xi, Tang Liu, and Min Zhou. Scaling up graph neural networks via graph coarsening. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pp. 675–684, 2021.

Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. *arXiv preprint arXiv:2110.07580*, 2021.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Manoj Kumar, Anurag Sharma, and Sandeep Kumar. A unified framework for optimization-based graph coarsening. *Journal of Machine Learning Research*, 24(118):1–50, 2023.

Manoj Kumar, Subhanu Halder, Archit Kane, Ruchir Gupta, and Sandeep Kumar. Optimization framework for semi-supervised attributed graph coarsening. In *The 40th Conference on Uncertainty in Artificial Intelligence*, 2024.

Jiayu Li, Tianyun Zhang, Hao Tian, Shengmin Jin, Makan Fardad, and Reza Zafarani. Graph sparsification with graph convolutional networks. *International journal of data science and analytics*, pp. 1–14, 2022.

Rui Li, Xin Yuan, Mohsen Radfar, Peter Marendy, Wei Ni, Terrence J O'Brien, and Pablo M Casillas-Espinosa. Graph signal processing, graph neural network and graph learning on biological data: a systematic review. *IEEE Reviews in Biomedical Engineering*, 16:109–135, 2021.

Andreas Loukas. Graph reduction with spectral and cut guarantees. *Journal of Machine Learning Research*, 20(116):1–42, 2019.

Andreas Loukas and Pierre Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *International conference on machine learning*, pp. 3237–3246. PMLR, 2018.

Dov Monderer and Lloyd S Shapley. Potential games. *Games and economic behavior*, 14(1):124–143, 1996.

Jennifer Neville and David Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8(3), 2007.

David Pisinger and Paolo Toth. Knapsack problems. In *Handbook of Combinatorial Optimization: Volume1–3*, pp. 299–428. Springer, 1998.

Saeed Rahmani, Asiye Baghbani, Nizar Bouguila, and Zachary Patterson. Graph neural networks for intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 24(8): 8846–8885, 2023.

Baoxu Shi and Tim Weninger. Proje: Embedding projection for knowledge graph completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Johann v Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton University Press, 1953.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pp. 2022–2032, 2019.

Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: a survey. *ACM Computing Surveys*, 55(5):1–37, 2022.

Zhenbang Xiao, Yu Wang, Shunyu Liu, Huiqiong Wang, Mingli Song, and Tongya Zheng. Simple graph condensation. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 53–71. Springer, 2024.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. *Advances in neural information processing systems*, 31, 2018.

Jack Zhang, Lucas Silveira, Hamidreza Validi, Logan Smith, Austin Buchanan, and Illya V Hicks. Partitioning a graph into low-diameter clusters.

Bo Zhao, Konda Reddy Mopuri, and Hakan Bilen. Dataset condensation with gradient matching. *arXiv preprint arXiv:2006.05929*, 2020.

Xin Zheng, Miao Zhang, Chunyang Chen, Quoc Viet Hung Nguyen, Xingquan Zhu, and Shirui Pan. Structure-free graph condensation: From large-scale graphs to condensed graph-free data. *Advances in Neural Information Processing Systems*, 36:6026–6047, 2023.

A Appendix

A.1 NP-Completeness of Stable Coalition problem in Non-Leader-Based Refinement

Proof of Theorem 1. To prove that Stable Coalition problem Q' is NP-complete, we must show two things: (1) $Q' \in$ NP and (2) a known NP-complete problem Q can be reduced to Q' in polynomial time.

Given a finite set of real numbers S and a polynomial-time computable function f, the Stable Coalition decision problem is defined as determining whether there exists a non-empty subset $S_i \subseteq S$ such that $\text{sum}(S_i) \leq f(S_i)$. Here, $\text{sum}(S_i)$ represents the total cost of the players in the subset S_i, obtained by summing the individual costs assigned to players within the coalition S, while $f(S_i)$ denotes the worth value of the coalition S_i, as defined by the internal stability condition in equation (8). We now proceed to show that it satisfies the two required conditions for NP-completeness.

Step 1: Stable Coalition problem is NP. A problem belongs to the class NP if there exists a polynomial-time verification algorithm such that, for every instance of the problem, the algorithm accepts whenever a valid certificate is provided. Let a certificate be a subset $S_i \subseteq S$. We can verify this certificate using the verification algorithm $A_{Q'}$, which operates as follows:

1. **Computation of** $\mathrm{sum}(S_i)$: The total cost of coalition S_i is defined as
$$\mathrm{sum}(S_i) = \sum_{j \in S_i} c_S(j), \quad \text{where } c_S(j) = \mathrm{DE}(S) - \mathrm{DE}(S \setminus \{j\}),$$
and the Dirichlet energy DE() of a coalition S is given by
$$\mathrm{DE}(S) = \frac{1}{2} \sum_{(u,v) \in E_S} w_{uv} \|x_u - x_v\|^2,$$
where E_S denotes the set of internal edges within coalition S. Each term $\|x_u - x_v\|^2$ can be computed in $O(p)$ time, where p is the embedding dimension of the node features. Since each edge $(u, v) \in E_S$ is processed once, the total time to compute $\mathrm{DE}(S)$ is $O(|E_S| \cdot p)$. Therefore, computing $\mathrm{sum}(S_i)$ for a subset S_i takes $O(|S_i| \cdot |E_S| \cdot p)$ time. Hence, the computation of $v_1 = \mathrm{sum}(S_i)$ can be performed in polynomial time.

2. **Computation of** $f(S_i)$: The function $f(\cdot)$ corresponds to the characteristic function $v(\cdot)$ defined in equations (6) and (7), which determines the worth value of a coalition S_i as
$$v(S_i) = \begin{cases} \mathrm{DE}(S_i), & \text{if } |S_i| = 1, \\ \sum\limits_{j \in S_i} c_{S_i}(j), & \text{if } |S_i| > 1, \end{cases}$$
where each player's cost contribution $c_{S_i}(j)$ is given by
$$c_{S_i}(j) = \mathrm{DE}(S_i) - \mathrm{DE}(S_i \setminus \{j\}).$$
From Step (1), the computation of $\mathrm{DE}(\cdot)$ for a coalition S_i requires $O(|E_{S_i}| \cdot p)$ time, where $|E_{S_i}|$ denotes the number of internal edges within S_i and p is the dimension of the node features. Hence, the total cost of computing all $c_{S_i}(j)$ for $j \in S_i$ is $O(|S_i| \cdot |E_{S_i}| \cdot p)$.

   Combining these results, the overall computational complexity of evaluating the characteristic function $v(S_i)$ remains polynomial in both the size of the subset S_i and the number of edges within it. Therefore, $f(S_i) = v(S_i)$ can be computed in polynomial time.

3. **Verification:** Compare $sum(S_i)$ and $f(S_i)$. Checking whether $sum(S_i) < f(S_i)$ takes constant time.

Since all steps of the verification process (computing v_1, computing v_2, and performing the comparison) are executable in polynomial time with respect to the input size, the Coalition_Deviation problem belongs to the class NP.

Step 2: Reduction from a Known NP-Complete Problem to Problem Q'. It is well known that the 0–1 Knapsack Decision Problem is NP-complete. In the special case where the weight of each item is proportional to its profit value (that is, $v_i = w_i$), the 0–1 Knapsack Problem reduces to the Subset Sum problem. Since Subset Sum is also NP-complete, this special case of the Knapsack Problem remains NP-complete, as discussed by Pisinger & Toth (1998).

Special Case of 0–1 Knapsack Problem. Given a set of n items A, where each item $i \in A$ has a weight $w_i \in \mathbb{R}^+$ and a value $v_i \in \mathbb{R}^+$, and a maximum knapsack capacity W, does there exist a non-empty subset $A' \subseteq A$ such that the total weight satisfies $\sum_{i \in A'} w_i \leq W$ and the total value satisfies $\sum_{i \in A'} v_i \geq t$? In this special case, the value of each item is proportional to its weight, i.e., $v_i = w_i$.

We demonstrate a polynomial-time reduction from the NP-complete problem SPECIAL CASE OF 0-1 KNAPSACK PROBLEM Q to the problem Q'. Let $R_{Q \to Q'}$ denote a polynomial-time reduction algorithm that generates a mapping function g from an instance S_i of Q to a corresponding instance $g(S_i)$ of Q'. Assume $A_{Q'}$ is a decision algorithm for Q' and let A_o denote a composite procedure that applies $R_{Q \to Q'}$ followed by $A_{Q'}$. Figure 5 illustrates this overall reduction and verification process.

The Reduction: Let the reduction algorithm $R_{Q \to Q'}$ be defined as follows:

1. Given an instance of the special 0–1 Knapsack problem: a set of items $A = \{1, 2, \ldots, n\}$ with weights $w_i \in \mathbb{R}^+$ (and values $v_i = w_i$), a maximum capacity $W \in \mathbb{R}^+$ and target value $t \in \mathbb{R}^+$, construct a corresponding instance of Stable Coalition(S,f):

$$S = A \quad \text{and assign costs } c_S(i) = w_i \text{ for each } i \in S.$$

2. Define the coalition worth function f for any subset $S_i \subseteq S$ as

$$f(S_i) = \begin{cases} W, & \text{if } \sum_{i \in S_i} w_i \geq t, \\ \sum_{i \in S_i} w_i - 1, & \text{otherwise.} \end{cases}$$

This reduction is clearly polynomial-time. Constructing S from A requires $O(|A|)$ operations. Evaluating $\sum_{i \in S_i} c_S(i)$ for any subset S_i requires $O(|S_i|)$ time, and computing $f(S_i)$ is $O(|S_i|)$ time. Hence, the overall transformation is polynomial-time.

Proof of Equivalence: We now show that the special Knapsack instance has a feasible solution if and only if the constructed Stable Coalition instance has a subset satisfying the stability condition $\sum_{i \in S_i} c_i \leq f(S_i)$.

- $\Rightarrow$ Suppose the Knapsack instance has a subset $A' \subseteq A$ such that $t \leq \sum_{i \in A'} w_i \leq W$. Let $S_i = A'$. Then $\sum_{i \in S_i} w_i \geq t$, so by construction $f(S_i) = W$. The Stable Coalition condition is $\sum_{i \in S_i} c_S(i) = \sum_{i \in A'} w_i \leq W = f(S_i)$, which holds.

- $\Leftarrow$ Suppose there exists a subset $S_i \subseteq S$ such that $\sum_{i \in S_i} c_S(i) \leq f(S_i)$. Two cases arise:

  1. If $\sum_{i \in S_i} w_i \geq t$, then $f(S_i) = W$ and $\sum_{i \in S_i} w_i \leq W$, so $t \leq \sum_{i \in S_i} w_i \leq W$, which is a feasible Knapsack solution.
  2. If $\sum_{i \in S_i} w_i < t$, then $f(S_i) = \sum_{i \in S_i} w_i - 1 < \sum_{i \in S_i} w_i$. In this case, the condition $\sum_{i \in S_i} c_S(i) \leq f(S_i)$ is false, so this subset cannot satisfy the Stable Coalition condition.

Hence, the Stable Coalition instance has a solution if and only if the special Knapsack instance has a feasible subset.

Conclusion Since Stable Coalition is in NP (Step 1), and a known NP-complete problem reduces to it in polynomial time (Step 2), it follows that Stable Coalition is NP-complete.

□

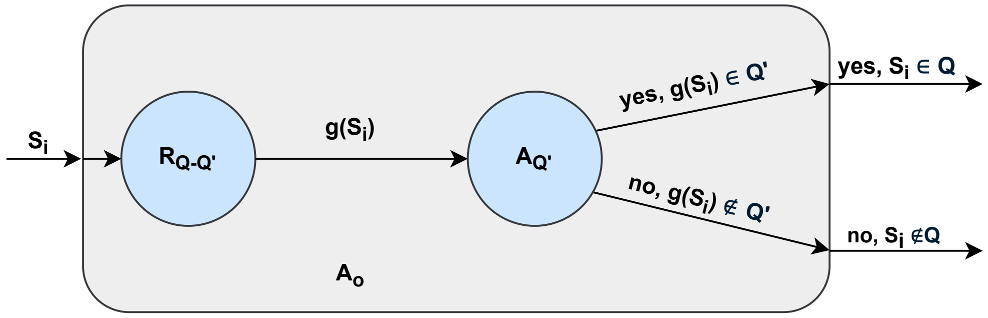


Figure 5: The proof of Theorem 1 is outlined as follows. Assume that $R_{Q\to Q'}$ denotes a polynomial-time reduction algorithm that generates the mapping function g from problem Q to Q', and that $A_{Q'}$ represents a polynomial-time decision algorithm for Q'. To determine whether $S_i \in Q$, algorithm A_o applies $R_{Q\to Q'}$ to transform the input instance S_i into $f(S_i)$, and then employs $A_{Q'}$ to verify whether $f(S_i) \in Q'$.

A.2 Illustrative Example of Coalition Stability

As illustrated in Figure 3, consider a graph in which three coalitions $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ are formed based on the 1-hop neighborhoods of nodes 1, 5, and 6, respectively. Specifically, for node 1, its local neighborhood includes nodes 2, 3, and 4, resulting in the coalition $\mathcal{C}_1 = \{1, 2, 3, 4\}$. To analyze the stability of these coalitions, we consider two approaches: first, the case without leader-based refinement, and then the case with leader-based refinement.

In Non-Leader-Based Coalition Refinement, the stability of a coalition is evaluated by checking whether any individual node or group of nodes has an incentive to deviate. Evaluating stability in this way requires examining all possible non-empty subsets of the coalition. For $\mathcal{C}_1$, which contains four nodes, the maximum number of non-empty subsets to consider is $2^4 - 1 = 15$. However, as shown in Figure 3a, only 11 subsets are observed to deviate since $\mathcal{C}_1$ is not a fully connected clique. When the nodes in a coalition are fully connected, the deviation-checking process corresponds to the Stable Coalition problem, which is NP-complete, as established in Theorem 1.

In the leader-based coalition refinement, nodes $1, 5, 6$ are selected as leaders, as shown in Figure 3b, and form coalitions with all their neighbors. In the refinement step, each neighbor a_i is individually checked to decide whether it should (i) remain in the leader's coalition, (ii) leave to join another coalition led by a different leader, or (iii) remain unassigned if no better option is available. This requires only 6 checks, one per neighbor, resulting in linear time complexity. This localized refinement procedure generalizes efficiently across the entire graph. Let m denote the number of leaders (at most n), and d_i the number of neighbors of leader i. The total refinement cost across all leader-based coalitions is $O\left(\sum_{i=1}^{m} d_i\right)$. Since $\sum_{i=1}^{m} d_i \leq n \cdot \Delta$, where Δ is the average degree in the graph, the total refinement time is bounded by $\boxed{O(n \cdot \Delta)}$.

A.3 Multi-Level CGC Algorithm

Algorithm 2 Multi-Level CGC Algortihm based Graph Coarsening

```
Require: Original graph $G = (V, E, X)$, number of coarsening levels $k$
Ensure: Coarsened graph $G^{(k)}$
1: $G^{(0)} \leftarrow G$
2: for $i = 1$ to $k$ do
3:     Apply CGC algorithm 1 on $G^{(i-1)}$
4:     Obtain coarsened graph $G^{(i)}$
5: end for
6: return $G^{(k)}$
```

A.4 Additional Dataset Details

Dataset	number of nodes	features	levels
Cora	2,704	1,433	7
DBLP	17,716	1,639	4
CoCS	18,333	2,000	5
Pubmed	19,717	500	3
Co-Physics	34,493	8,415	6
OGBN-Arxiv	169343	128	40
Genius	421961	12	2

Table 4: Details of the real datasets, including the number of nodes p, features n , and levels l.

A.5 Theoretical Convergence Guarantee

Proof of Theorem 2. Recall that for each coalition S_i, the coalition-level potential $\phi(R^m_{S_i})$ is defined in equation (11). The global potential is then given by $\Phi(m) = \sum_{S_i \in \mathcal{S}} \phi(R^m_{S_i})$.

If the potential function Φ meets the cost-alignment condition defined in equation 10, then CGame confirms the properties of a potential game, where local cost improvements support the minimization of global potential. Under CGC algorithm, node i updates its mapping decision from a coalition S_i to S'_i at a mapping profile $m = (m_i, m_{-i})$, only if its cost decreases. Suppose $R^m_{S_i}$ and $R^m_{S'_i}$ are the nodes in coalitions S_i and S'_i, respectively, before changing the mapping decision. The change in node i's cost due to the transition from coalition S_i to S'_i is calculated as:

$$\Delta c_i = c_i(m'_i, m_{-i}) - c_i(m_i, m_{-i}) = DE(R^m_{S'_i} \cup \{i\}) - DE(R^m_{S'_i}) - \big(DE(R^m_{S_i}) - DE(R^m_{S_i} \setminus \{i\})\big) \tag{13}$$

where $\Delta c_i < 0$. This inequality holds because a node deviates to S'_i only when such a move strictly reduces its cost. Formally,

$$c_i(m'_i, m_{-i}) < c_i(m_i, m_{-i}) \quad \Rightarrow \quad \Delta c_i = c_i(m'_i, m_{-i}) - c_i(m_i, m_{-i}) < 0. \tag{14}$$

Therefore, the update rule ensures that a node's cost always decreases after deviation.

Now, when a node updates its mapping, the value of the potential function changes as follows:

$$\Delta \phi = \phi_{\text{inc}} - \phi_{\text{dec}}, \tag{15}$$

where ϕ_{inc} represents the increase in ϕ, and ϕ_{dec} represents the decrease in ϕ.

Specifically, when node i is remapped from coalition S_i to S'_i, the potential value ϕ of coalition S'_i increases, while the potential value ϕ of coalition S_i decreases. These changes in ϕ are calculated as follows:

The increase in ϕ for the new coalition S'_i is:

$$\begin{aligned}
\phi_{\text{inc}} &= \phi(R^m_{S'_i} \cup \{i\}) - \phi(R^m_{S'_i}) \\
&= DE(R^m_{S'_i} \cup \{i\}) - DE(R^m_{S'_i}) && \text{(by equation (11))} \\
&= c_i(m'_i, m_{-i}) && \text{(by equation (7)).}
\end{aligned} \tag{16}$$

The decrease in ϕ for the original coalition S_i is:

$$\begin{aligned}
\phi_{\text{dec}} &= \phi(R^m_{S_i}) - \phi(R^m_{S_i} \setminus \{i\}) \\
&= DE(R^m_{S_i}) - DE(R^m_{S_i} \setminus \{i\}) && \text{(by equation (11))} \\
&= c_i(m_i, m_{-i}) && \text{(by equation (7)).}
\end{aligned} \tag{17}$$

If the condition $\Delta c_i < 0$ is satisfied, then from equations (16) and (17), the change in potential $\Delta\phi$ is equal to Δc_i. Therefore,$\Delta\phi < 0$. This proves that CGame is a potential game. This also indicates that the value of $\Delta\phi$ decreases as nodes update their mapping decisions according to the CGC algorithm.

□

Proof of Theorem 3. The presence of a PNE[1] is justified using the arguments outlined below.

1. In the proposed `CGame`, the potential function $\phi(m)$ is defined over the set of all possible mapping profiles, where each node selects a coalition from its neighbors. Since both the number of nodes and their neighbor sets are finite, the number of mapping profiles and thus the range of $\phi(m)$ is finite.
2. The game progresses through updates to these profiles, with nodes updating their choices based on the CGC algorithm. As shown in Theorem 2 and above point that this game is a finite-potential game where each update results in a strictly decreasing value of $\phi(m)$, preventing the game from revisiting previous states and ensuring convergence. Once the potential function reaches its minimum value, any unilateral deviation by a player would increase their own cost and thus increase $\phi(m)$. Therefore, a player will not unilaterally deviate, confirming that the converged state corresponds to a Pure Nash Equilibrium.[2]

□

[1]In CGame, a PNE represents a mapping profile where no player can further reduce its cost by unilaterally changing its mapping choice.

[2]Every finite potential game admits a pure-strategy equilibrium, as proved by Monderer & Shapley (1996).