Domain Adaptation in Physical Systems via Graph Kernel

Haoran Li
lhaoran@asu.edu
Arizona State University
Tempe, Arizona, USA

Hanghang Tong
htong@illinois.edu
University of Illinois at Urbana-Champaign
Champaign, Illinois, USA

Yang Weng
yang.weng@asu.edu
Arizona State University
Tempe, Arizona, USA

ABSTRACT

Physical systems are extending their monitoring capacities to edge areas with low-cost, low-power sensors and advanced data mining and machine learning techniques. However, new systems often have limited data for training the model, calling for effective knowledge transfer from other relevant grids. Specifically, *Domain Adaptation* (DA) seeks domain-invariant features to boost the model performance in the target domain. Nonetheless, existing DA techniques face significant challenges due to the unique characteristics of physical datasets: (1) complex spatial-temporal correlations, (2) diverse data sources including node/edge measurements and labels, and (3) large-scale data sizes. In this paper, we propose a novel cross-graph DA based on two core designs of graph kernels and graph coarsening. The former design handles spatial-temporal correlations and can incorporate networked measurements and labels conveniently. The spatial structures, temporal trends, measurement similarity, and label information together determine the similarity of two graphs, guiding the DA to find domain-invariant features. Mathematically, we construct a Graph kerNel-based distribution Adaptation (GNA) with a specifically-designed graph kernel. Then, we prove the proposed kernel is positive definite and universal, which strictly guarantees the feasibility of the used DA measure. However, the computation cost of the kernel is prohibitive for large systems. In response, we propose a novel coarsening process to obtain much smaller graphs for GNA. Finally, we report the superiority of GNA in diversified systems, including power systems, mass-damper systems, and human-activity sensing systems.

CCS CONCEPTS

• **Computing methodologies** → **Kernel methods**.

KEYWORDS

Physical System, Domain Adaptation, Networked Data, Graph Kernels, Graph Coarsening

ACM Reference Format:
Haoran Li, Hanghang Tong, and Yang Weng. 2022. Domain Adaptation in Physical Systems via Graph Kernel. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534678.3539380

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539380

1 INTRODUCTION

Modern physical systems are experiencing tremendous changes. First, the system territory is expanding to better serve society. Second, to increase the efficiency and decrease the cost of the system, new components are continuously introduced [20]. Some of them bring a lot of uncertainties due to intrinsic uncertain sources, imperfect component monitoring ability, etc. While conventional system analysis may fail for large-scale edge areas with high uncertainties, Data Mining (DM) and Machine Learning (ML) methods could play a game-changing role with accurate and cost-efficient implementations for new technologies of system monitoring, control, and protection. Especially, DM/ML models have been developed for physical system state estimation, cyber-attack detection, and event identification [21, 22], etc. Such wide applications are due to DM/ML's flexible modeling capability, robust performance, and high inference speed.

However, most of the existing DM/ML models on physical systems require a certain amount of training data. If the data are limited, the model training is likely to fail due to the curse of dimensionality. Unfortunately, data-limited scenarios often occur for a completely new grid or an old grid with increased metering, not to mention the common upgrading process with new nodes/lines. Thus, it is urgent to employ new methods to transfer knowledge from the source grid with rich data to the target grid with limited data [19, 23].

For this goal, Transfer Learning (TL) is defined conceptually as an efficient procedure to extract common knowledge from two different domains and boost the performance of the data-limited domain. Specifically, an efficient approach is Domain Adaptation (DA) [24–26, 30, 35, 37, 42] which minimizes the data distribution discrepancy of two domains, usually in a low-dimensional space for domain invariant features. Therefore, DA promises that joint training using common knowledge can significantly boost the learning process. While many efficient DA models have been applied to computer vision [12, 25] and natural language process [39], relatively few work has been done for graph data [31].

For physical systems, the widely placed sensors bring numerous networked measurements for nodes and edges with complex spatial-temporal correlations. Can we fully explore the correlations to improve DA methods? Further, nodes or edges can have different physical characteristics, i.e., different labels. Is the label information beneficial to the distribution adaptation? Intuitively, the answer shall be yes for both questions as the above information represents different levels of graph similarity that further guides the extraction of common knowledge in DA.

To elaborate more on the above statements, we present the following observations. Spatially, neighborhoods of nodes usually

have similar measurement patterns. For example, one residential area often reflects similar energy consumption patterns for local households. Further, the connectivity provides the separation of physical flow (e.g., water or gas) patterns. Temporally, an interval of temporal measurements represents a continuous change of system states, containing the invariant knowledge of cross-system dynamics. Finally, the node/edge labels help to further divide the spatial groups with commonality. For instance, in a power system, nodes for household electricity consumption may be close to nodes with solar panels that can generate electricity. However, they have completely different patterns: the former is based on human behaviors, while the latter is based on the strength of the sunlight.

Thus, it is critical to integrate the structures, temporal changes, and label information to identify graph similarity. In this paper, we show the integration can be conveniently implemented via a specially-designed graph kernel. Further, the proposed kernel can be embedded into a DA model [25] with theoretical guarantees. Then, the common knowledge is represented in a low-dimensional feature space converted from graph kernel features. We denote the final model as Graph kerNel-based distribution Adaptation (GNA). Nevertheless, the computational cost for GNA is high for large systems. Thus, we develop a pre-processing approach of graph coarsening to aggregate original graphs into coarser graphs based on structural and label information. On the other hand, the knowledge related to the graph similarity is preserved. Then, the proposed DA model can be easily and quickly implemented on the coarser graphs. Finally, we summarize our contributions as follows.

- We develop novel graph coarsening and graph kernel techniques to construct our GNA model.
- We theoretically analyze the model validity and efficiency.
- We conduct experiments to demonstrate the superiority of our proposed methods on diverse physical systems.

2 RELATED WORK

2.1 Domain Adaptation

2.1.1 General Categorization. DA methods try to match the conditional and/or marginal distributions of the source and the target domains by minimizing the distribution discrepancy measure like Maximum Mean Discrepancy (MMD) [14], Jensen-Shannon Divergence [32], and Wasserstein distance [41], etc. Traditional models usually employ linear or kernalized non-linear transformation matrices to extract common features with a minimal distribution difference [25, 30]. Recently, with the ability of high-level feature extractions, deep learning models are introduced to DA to obtain more transferable features [13, 24, 26, 35–37, 42]. Specifically, [24, 26, 35, 37] introduce the domain shift measure as another loss to Convolutional Neural Networks (CNNs) to simultaneously train the classifier and match domains. [13, 36, 42] employ adversarial learning with domain discriminator(s) to ensure the minimization of the domain discrepancy.

2.1.2 DA with Data Structural Information. Under the above taxonomy, studies of DA to accommodate data structures are heavily investigated. They typically utilize graphs as containers for the input data or features. Thus, ML technologies for graphs can be conveniently utilized. In general, we categorize them by the different types of graphs. (1) Instance graph. The instance graph treats each node as a sample from either source or target domains [1, 5, 7, 16, 29]. Then, the graph similarity can indicate the distribution difference and provide graph-based domain alignment. Specifically, [5, 7] build an optimization model to minimize MMD measure, which considers the connections among samples to construct a better distribution difference measure. [1, 16, 29] employ deep learning models, e.g., CNNs and Graph Neural Networks (GNNs), for high-level feature extractions. They all utilize the adversarial learning scheme to learn domain invariant features. These methods, however, focus on the relationship between instances but not the spatial-temporal correlations within one instance in networked data.

Thus, other work investigates (2) networked data graph that treats one or several data features as a node for the networked dataset [6, 10, 11, 18, 31]. Specifically, [10] projects networked data into a latent space that captures the common sub-structure. However, finding common sub-structures will cause the non-ignorable loss by removing non-common parts. [11] can measure the similarity of different graphs with an index based on the node degree similarity. However, the index is not sufficient to tackle physical systems with measurements and labels for nodes and edges. In this paper, we propose a graph kernel to make use of all data to measure the graph similarity for two different graphs. [6, 18, 31] build CNNs and GNNs to process networked data to learn transferable features. However, their projected feature space, though containing network information, is restricted with the same dimensionality and causes information losses for systems of different sizes. Further, they generally suffer high computational costs for large-scale physical systems and lack theoretical guarantees. We tackle these issues by proposing an efficient optimization using graph kernels and MMD measures with rigorous proofs.

2.2 Graph Kernel

Graph kernel is a measure of similarity between graphs [38]. There are many types of graph kernels with different formulations under different kinds of data. The R-convolution kernel calculates kernels as the product of R-convolutions between components of graphs. Specifically, one decomposes graphs into substructures (e.g., subgraphs) and evaluates a kernel between them [17]. Usually, nodal labels are used for evaluation. However, the computational cost is extremely high as there are various divisions for the substructures. The Optimal Assignment (OA) kernel uses a base kernel to compare nodal labels and forms a similarity measure for a pair of graphs. Further, OA searches for the optimal mapping between nodes to save computational costs. However, [38] shows that OA is usually not positive semi-definite, which is not suitable for the MMD measure of DA.

A more efficient and popular measure is the so-called random walk graph kernel that implements random walks on two graphs and counts the matched walks [38, 44]. Traditional random walk kernel usually utilizes edge labels to calculate the kernel. [2] proposes a modified version that can incorporate both labels and measurements for nodes and edges. However, this modification is still not suitable for our spatial-temporal graphs. Especially, the temporal correlations are not considered but are essential for the following domain adaptation and feature learning. To capture the temporal correlations, we propose a new graph kernel and prove that it is suitable to construct kernel-based MMD measures.

2.3 Graph Coarsening

For large-scale graphs, the computation of graph kernels is still inefficient. Thus, an effective way is to pre-process graphs and obtain coarser graphs that preserve the main information of original graphs. For example, [27, 28] keep the spectral similarity. [3] preserves the structure of the inverse Laplacian. [4] maintains the classification accuracy of GNNs. These goals, however, do not match the purpose of distribution adaptation. Thus, we propose a simple but efficient coarsening scheme that captures the local similarity using local structures and labels.

3 PROPOSED METHOD

3.1 Notation and Problem Definition

Physical systems can be modeled as spatial-temporal graphs [40]. Specifically, we denote $\{G^n = \{V, E, D^n\}\}_{n=1}^N$ as the source graph, where V and E are the node and edge sets, respectively. D^n is the corresponding data at the time slot n and N is the number of time slots for the graphs. Similarly, the target graph is denoted as $\{\widetilde{G}^n = \{\widetilde{V}, \widetilde{E}, \widetilde{D}^n\}\}_{n=1}^{\widetilde{N}}$. In the following derivations, if there are no special notices, quantities with tilde are by default linked to the target graph.

The physical graph contains data of different modality, which together can boosts the knowledge transfer and feature extraction for the final machine learning model. In this paper, we assume the source graph contains node measurements $\boldsymbol{X} \in \mathbb{R}^{|V|\times N}$, node labels $\boldsymbol{L} \in \mathbb{Z}^{|V|\times N}$, edge measurements $\boldsymbol{Y} \in \mathbb{R}^{|E|\times N}$, and edge labels $\boldsymbol{M} \in \mathbb{Z}^{|E|\times N}$, where $|\cdot|$ for a set represents the operation to obtain the set cardinality. Secondly, we focus on the graph-level classification and assume the task label vector is $\boldsymbol{h} \in \mathbb{Z}^N$. Thirdly, we assume the node/edge/task label lies in the range of $\{0, 1, \cdots, K_1\}$, $\{0, 1, \cdots, K_2\}$, and $\{0, 1, \cdots, K_3\}$, respectively. Finally, the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{|V|\times|V|}$ is assumed to be known. Similarly, we denote the corresponding quantities for the target graph as $\widetilde{\boldsymbol{X}} \in \mathbb{R}^{|\widetilde{V}|\times\widetilde{N}}$, $\widetilde{\boldsymbol{L}} \in \mathbb{Z}^{|\widetilde{V}|\times\widetilde{N}}$, $\widetilde{\boldsymbol{Y}} \in \mathbb{R}^{|\widetilde{E}|\times\widetilde{N}}$, $\widetilde{\boldsymbol{M}} \in \mathbb{Z}^{|\widetilde{E}|\times\widetilde{N}}$, $\widetilde{\boldsymbol{h}} \in \mathbb{Z}^{\widetilde{N}}$, and $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{|\widetilde{V}|\times|\widetilde{V}|}$.

Then, we can find that $\{D^n\}_{n=1}^N = \{\boldsymbol{X}, \boldsymbol{L}, \boldsymbol{Y}, \boldsymbol{M}, \boldsymbol{h}\}$ and $\{\widetilde{D}^n\}_{n=1}^{\widetilde{N}} = \{\widetilde{\boldsymbol{X}}, \widetilde{\boldsymbol{L}}, \widetilde{\boldsymbol{Y}}, \widetilde{\boldsymbol{M}}, \widetilde{\boldsymbol{h}}\}$. For some realistic scenarios when there is no label information, one can utilize matrices with all ones to indicate that nodes (edges) share the same label. Further, our model can be easily generalized to the case when there exist multiple measurements/labels for one node/edge at each time slot.

In the transfer learning setting, we address the problem when distributions of $\{D^n\}_{n=1}^N$ and $\{\widetilde{D}^n\}_{n=1}^{\widetilde{N}}$ have a certain distance, which causes troubles for utilizing knowledge from both graphs for ML model training. Secondly, we assume there are much more data in the source graph, i.e., $N \gg \widetilde{N}$. Therefore, it is essential to convert data or features from the source graph to the target graph to enhance the ML models.

3.2 Graph Coarsening to Improve Efficiency

In this section, we propose a pre-processing method to obtain coarser graphs which are much smaller than source and target graphs, thus improving the model efficiency for the following computations. Mathematically, we aim to find spatial-temporal coarser graphs $\{G_C^n = \{V_C, E_C, D_C^n\}\}_{n=1}^N$ and $\{\widetilde{G}_C^n = \{\widetilde{V}_C, \widetilde{E}_C, \widetilde{D}_C^n\}\}_{n=1}^{\widetilde{N}}$ such that $|V_C| < |V|$ and $|\widetilde{V}_C| < |\widetilde{V}|$. To obtain coarser graphs, we have the following objectives.

- Propose surjective maps $\pi : V \to V_C$ and $\widetilde{\pi} : \widetilde{V} \to \widetilde{V}_C$ to aggregate nodes. Then, design the corresponding maps that can project nodal measurements/labels from the source/target graphs to the coarser graphs. Namely, we need to find $\pi^M : \mathbb{R}^{|V|} \to \mathbb{R}^{|V_C|}$, $\pi^L : \mathbb{Z}^{|V|} \to \mathbb{Z}^{|V_C|}$, $\widetilde{\pi}^M : \mathbb{R}^{|\widetilde{V}|} \to \mathbb{R}^{|\widetilde{V}_C|}$, and $\widetilde{\pi}^L : \mathbb{Z}^{|\widetilde{V}|} \to \mathbb{Z}^{|\widetilde{V}_C|}$.
- Figure out the connections E_C and $\widetilde{E}_C$ with the corresponding maps $\omega : E \to E_C$ and $\widetilde{\omega} : \widetilde{E} \to \widetilde{E}_C$. Then, find the maps of the edge measurements/labels from source/target graphs to the coarser graphs. Namely, we need to find $\omega^M : \mathbb{R}^{|E|} \to \mathbb{R}^{|E_C|}$, $\omega^L : \mathbb{Z}^{|E|} \to \mathbb{Z}^{|E_C|}$, $\widetilde{\omega}^M : \mathbb{R}^{|\widetilde{E}|} \to \mathbb{R}^{|\widetilde{E}_C|}$, and $\widetilde{\omega}^L : \mathbb{Z}^{|\widetilde{E}|} \to \mathbb{Z}^{|\widetilde{E}_C|}$.

3.2.1 Structural Node Aggregation with Label Information. The neighborhood of nodes for a physical system usually indicates a group with similar physical behaviors. Thus, we can utilize one node to represent them in the coarser graph. Secondly, the label information of nodes can further improve the node grouping for the local areas. Thus, node maps π and $\widetilde{\pi}$ should take in both factors for a reasonable coarsening process. For the convenience of derivations, we only consider the coarsening of the source graph. Subsequently, we first initialize a set of neighborhoods based on the connectivity. Each neighborhood contains a center $i \in V$ and the neighbouring nodes $\text{Neigh}(i)$ such that $j \in \text{Neigh}(i)$ if and only if $d(i, j) \le d_{max}$, where $d(i, j)$ represent the pre-defined distance between node i and j, and d_{max} represents the maximum threshold. For example, we can define $d(i, j)$ to be the number of edges along the shortest path between i and j. As shown in Fig. 1, we consider $d_{max} = 1$ to aggregate nodes that are 1-hop away from the center nodes marked with a red star. Further, if there is no path between i and j, we define $d(i, j) = \infty$.

Subsequently, based on the previous discussion, the grouping of nodes in a neighborhood is too coarse and needs to be divided by the node labels. Thus, we divide $\text{Neigh}(i) = \{\text{Neigh}_k(i)\}_k^{K_1}$ such that $\forall j_1, j_2 \in \text{Neigh}_k(i)$, $\boldsymbol{L}(j_1, n) = \boldsymbol{L}(j_2, n) = k$ (recall that n is the index for the time slot for the discussed graph and data). For example, in Fig. 1, the blue and purple nodes have different labels. Thus, even though they share one neighborhood, they should be divided into two groups. Then, we have

$$\pi\big(\text{Neigh}_k(i)\big) = h \in V_C. \tag{1}$$

Based on the above node aggregation rules, since nodes in $\text{Neigh}_k(i)$ share one label k, the projected node $h \in V_C$ also has a label k. Namely,

$$\pi^L\Big(\boldsymbol{L}\big(\text{Neigh}_k(i), n\big)\Big) = k. \tag{2}$$

For the measurements, as we assume there is no prior on the importance of nodes, we directly average the measurements in the related nodes. Specifically, we have

$$\pi^M\Big(\boldsymbol{X}\big(\text{Neigh}_k(i), n\big)\Big) = \frac{1}{|\text{Neigh}_k(i)|}\sum_{j=1}^{|\text{Neigh}_k(i)|}\boldsymbol{X}(j, n). \tag{3}$$

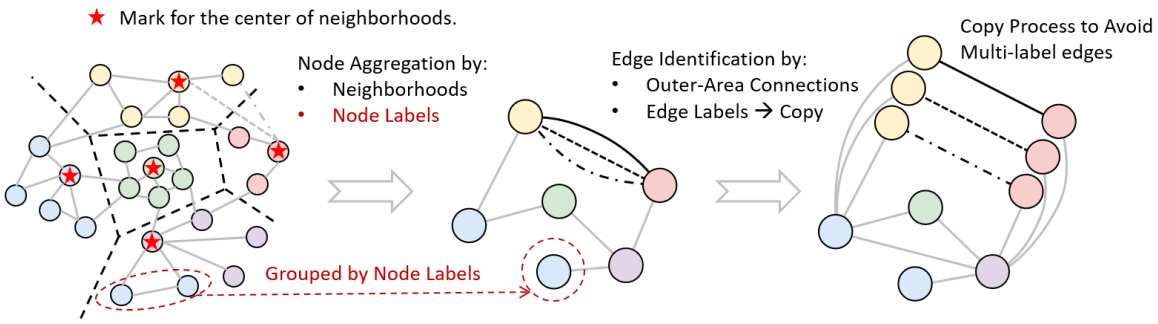


Figure 1: The procedure of graph coarsening. Different colors of nodes represent different labels. Different shape of lines represent different labels.

3.2.2 Edge Identification with Label Information. After obtaining V_C, we identify E_C with a simple and popular V_C-induced approach [4]. Namely, $(h_1, h_2) \in E_C$ if and only if there is an edge $(i_1, i_2) \in E$ and labels $1 \le k_1, k_2 \le K_2$ such that $\pi\big(\text{Neigh}_{k_1}(i_1)\big) = h_1$ and $\pi\big(\text{Neigh}_{k_2}(i_2)\big) = h_2$. In other words, the connection between two neighborhoods in the original graph implies a connection in the coarser graph.

However, the identified edge may have multiple labels as in the original graph, there may be many connections with multiple labels between two neighborhoods. This causes issues when formalizing edge measurement and label maps. For example, in the left and middle part of Fig. 1, the connections between yellow and pink nodes have multiple labels (i.e., different shapes of lines). To tackle this problem, for nodes in the coarser graph with multi-labeled edges, we propose to copy them according to the number of labels and assign each copied pair a uni-labeled edge. In the meantime, the outer connections to other nodes remain unchanged for each copied pair, shown in the right part of Fig. 1. Finally, our coarser graph only contains uni-labeled edges. For simplicity of later derivations, we denote $\text{Conne}_k(h_1, h_2) \subset E$ to be the set of edges in the original graph that has a label k ($1 \le k \le K_2$) and connects nodes in $\pi^{-1}(h_1)$ and $\pi^{-1}(h_2)$, where π^{-1} represents the inverse map of π. Then, the edge map is

$$\omega\big(\text{Conne}_k(h_1, h_2)\big) = h \in E_C. \tag{4}$$

Further, the edge label map can be defined as

$$\omega^L\Big(\boldsymbol{M}\big(\text{Conne}_k(h_1, h_2), n\big)\Big) = k. \tag{5}$$

Similarly, we can average the edge measurements in $\text{Conne}_k(h_1, h_2)$ to obtain the edge measurement in the coarser graph:

$$\omega^M\Big(\boldsymbol{Y}\big(\text{Conne}_k(h_1, h_2), n\big)\Big) = \frac{1}{|\text{Conne}_k(h_1, h_2)|} \sum_{j=1}^{|\text{Conne}_k(h_1,h_2)|} \boldsymbol{Y}(j, n). \tag{6}$$

Finally, we can follow the above procedures and generate the coarser graph for the target grid. For the two coarser graphs, the number of nodes can vary due to different numbers of neighborhoods and the node copy process in this subsection. Further, the connectivity can be different due to the varying topology of the two original graphs. Thus, our obtained spatial-temporal coarser graphs $\{G_C^n\}_{n=1}^N$ and $\{\widetilde{G_C^n}\}_{n=1}^{\widetilde{N}}$ are usually different.

3.3 GNA to Learn Common Graph Features

The two coarser graphs bring two sets $\{D_C^n\}_{n=1}^N$ and $\{\widetilde{D}_C^n\}_{n=1}^{\widetilde{N}}$ of samples (measurements and labels) converted from source and target physical systems, which are generated via Equations (2), (3), (5), and (6). With the data, we develop Graph kerNel-based distribution Adaptation (GNA).

3.3.1 Distribution Adaptation for Two Sample Sets. A prevalent measure for the distribution difference with two sample sets is the so-called Maximum Mean Discrepancy (MMD) [14, 25, 30]. MMD can compare distributions based on reproducing Hilbert Space (RKHS). Specifically, the Joint Domain Adaptation (JDA) in [25] tries to minimize:

$$\begin{aligned} &\min_{\boldsymbol{W}^\top \boldsymbol{K}\boldsymbol{H}\boldsymbol{K}^\top \boldsymbol{W} = \boldsymbol{I}} MMD(\{D_C^n\}_{n=1}^N, \{\widetilde{D}_C^n\}_{n=1}^{\widetilde{N}}) \\ &= \sum_{k=0}^{K_3} \text{tr}(\boldsymbol{W}^\top \boldsymbol{K}\boldsymbol{N}_k\boldsymbol{K}^\top \boldsymbol{W}) + \lambda||\boldsymbol{W}||_F^2, \end{aligned} \tag{7}$$

where $\boldsymbol{H} = \boldsymbol{I} - \frac{1}{N+\widetilde{N}}\boldsymbol{1}$, $\boldsymbol{I}$ is the identity matrix, $\boldsymbol{1} \in \mathbb{R}^{(N+\widetilde{N})\times(N+\widetilde{N})}$ is a matrix of ones, and $\boldsymbol{W} \in \mathbb{R}^{(N+\widetilde{N})\times n_0}$ ($n_0 < N + \widetilde{N}$) is the learnable linear transformation to project kernel features to a low-dimensional space. $\boldsymbol{K}$ is the kernel matrix obtained from samples in sets $\{D_C^n\}_{n=1}^N$ and $\{\widetilde{D}_C^n\}_{n=1}^{\widetilde{N}}$. λ is a positive penalty term and $||\cdot||_F$ is the Frobenius norm. $\boldsymbol{N}_k$ is a weight matrix. When $k = 0$, we have:

$$\boldsymbol{N}_0(i,j) = \begin{cases} \frac{1}{N^2}, & \text{if } 1 \le i, j \le N, \\ \frac{1}{\widetilde{N}^2}, & \text{if } N+1 \le i, j \le N+\widetilde{N}, \\ \frac{-1}{N\widetilde{N}}, & \text{otherwise,} \end{cases} \tag{8}$$

When $k \ge 1$, we have:

$$\boldsymbol{N}_k(i,j) = \begin{cases} \frac{1}{N_k^2}, & \text{if } 1 \le i, j \le N \text{ and } \boldsymbol{h}(i) = \boldsymbol{h}(j) = k, \\ \frac{1}{\widetilde{N}_k^2}, & \text{if } N+1 \le i, j \le N+\widetilde{N} \text{ and } \widetilde{\boldsymbol{h}}(i) = \widetilde{\boldsymbol{h}}(j) = k, \\ \frac{-1}{N_k\widetilde{N}_k}, & \text{if } 1 \le i \le N, N+1 \le j \le N+\widetilde{N}, \boldsymbol{h}(i) = \widetilde{\boldsymbol{h}}(j) = k \\ \frac{-1}{N_k\widetilde{N}_k}, & \text{if } 1 \le j \le N, N+1 \le i \le N+\widetilde{N}, \boldsymbol{h}(j) = \widetilde{\boldsymbol{h}}(i) = k \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where N_k and $\widetilde{N}_k$ represent the samples with task label k for the source and the target coarser graph, respectively. To calculate the kernel matrix, traditional kernels like the polynomial kernel (k_{poly})

or the Radial Basis Function (RBF) kernel (k_{rbf}) are largely used. However, they are not suitable for datasets from physical systems.

3.3.2 Spatial-Temporal Graph Kernel with Measurements and Labels. In this subsection, we propose a graph kernel design for $\{G_C^n\}_{n=1}^N$ and $\{\widetilde{G_C^n}\}_{n=1}^{\widetilde{N}}$ to formalize Equation (7). The designed kernel is a modified version of the random walk kernel [2, 38] to process both the measurements and labels from nodes and edges. Random walk kernels [38] perform random walks on two graphs and calculate the number of matching walks to describe the graph similarity.

Specifically, we consider two graphs $G_C^{n_1}$ and $\widetilde{G_C^{n_2}}$. For simplification, we eliminate the unnecessary time slot index n_1 and n_2 in the later derivations. Then, the direct product graph, denoted as $G_C \times \widetilde{G_C}$, contains the node, edge set, and the adjacency matrix: $V_\times = \{(i,j) : i \in V_C, j \in \widetilde{V_C}\}$, $E_\times = \{\big((i_1,j_1),(i_2,j_2)\big) : (i_1,i_2) \in E_C \cap (j_1,j_2) \in \widetilde{E_C}\}$, and $\boldsymbol{A}_\times = \boldsymbol{A}_C \otimes \widetilde{\boldsymbol{A}_C}$, where $\boldsymbol{A}_C$ and $\widetilde{\boldsymbol{A}_C}$ are the adjacency matrices of the G_C and $\widetilde{G_C}$, respectively. $\otimes$ is the kronecker product. Then, the classical random walk kernel considers the weighted sum of random walks on the direct product graph:

$$\boldsymbol{K}(n_1,n_2) = \sum_{i,j=1}^{|V_\times|} \sum_{l=0}^{\infty} \alpha(\boldsymbol{A}_\times)^l(i,j), \tag{10}$$

where α is a positive weight factor to guarantee convergence. To better process nodal/edge measurement and label data, [2] proposes to develop a modified random walk by replacing $\boldsymbol{A}_\times$ in Equation (10) with a new weighted adjacency matrix $\boldsymbol{B}_\times$ such that:

$$\begin{aligned}&\boldsymbol{B}_\times(i_1 + |\widetilde{V_C}| \times (j_1-1), i_2 + |\widetilde{V_C}| \times (j_2-1))\\ &= \begin{cases} k_{step}\big((i_1,j_1),(i_2,j_2)\big), & \text{if } (i_1,j_1),(i_2,j_2) \in E_\times, \\ 0, & \text{otherwise,} \end{cases}\end{aligned} \tag{11}$$

where $k_{step}\big((i_1,j_1),(i_2,j_2)\big)$ is the step kernel to measure the similarity between steps $(i_1,i_2) \in E_C$ and $(j_1,j_2) \in \widetilde{E_C}$. However, the defined step kernel in [2] can hardly capture the temporal correlations. Therefore, we denote $\boldsymbol{l}_1 = [n_1-\delta_n, n_1-\delta_n+1, \cdots, n_1, n_1+1, \cdots, n_1+\delta_n]^\top$ and $\boldsymbol{l}_2 = [n_2-\delta_n, n_2-\delta_n+1, \cdots, n_2, n_2+1, \cdots, n_2+\delta_n]^\top$ to be index vectors for the neighbors of time slot n_1 and n_2 in the temporal dimension, respectively. Then, we propose a new step kernel definition:

$$\begin{aligned}
&k_{step}\big((i_1,j_1),(i_2,j_2)\big) = k^L\big((i_1,j_1),(i_2,j_2)\big) \times k^M\big((i_1,j_1),(i_2,j_2)\big),\\
&k^L\big((i_1,j_1),(i_2,j_2)\big) = \mathbb{1}\Big(\pi^L\Big(\boldsymbol{L}\big(\pi^{-1}(i_1),n_1\big)\Big) = \widetilde{\pi}^L\Big(\widetilde{\boldsymbol{L}}\big(\widetilde{\pi}^{-1}(j_1),n_2\big)\Big)\Big)\\
&\times \mathbb{1}\Big(\pi^L\Big(\boldsymbol{L}\big(\pi^{-1}(i_2),n_1\big)\Big) = \widetilde{\pi}^L\Big(\widetilde{\boldsymbol{L}}\big(\widetilde{\pi}^{-1}(j_2),n_2\big)\Big)\Big)\\
&\times \mathbb{1}\Big(\omega^L\Big(\boldsymbol{M}\big(\omega^{-1}(i_1,i_2),n_1\big)\Big) = \widetilde{\omega}^L\Big(\widetilde{\boldsymbol{M}}\big(\widetilde{\omega}^{-1}(j_1,j_2),n_2\big)\Big)\Big),\\
&k^M\big((i_1,j_1),(i_2,j_2)\big) = k_{rbf}\Big(\pi^M\Big(\boldsymbol{X}\big(\pi^{-1}(i_1),\boldsymbol{l}_1\big)\Big),\widetilde{\pi}^M\Big(\widetilde{\boldsymbol{X}}\big(\widetilde{\pi}^{-1}(j_1),\boldsymbol{l}_2\big)\Big)\Big)\\
&\times k_{rbf}\Big(\pi^M\Big(\boldsymbol{X}\big(\pi^{-1}(i_2),\boldsymbol{l}_1\big)\Big),\widetilde{\pi}^M\Big(\widetilde{\boldsymbol{X}}\big(\widetilde{\pi}^{-1}(j_2),\boldsymbol{l}_2\big)\Big)\Big)\\
&\times k_{rbf}\Big(\omega^M\Big(\boldsymbol{Y}\big(\omega^{-1}(i_1,i_2),\boldsymbol{l}_1\big)\Big),\widetilde{\omega}^M\Big(\widetilde{\boldsymbol{Y}}\big(\widetilde{\omega}^{-1}(j_1,j_2),\boldsymbol{l}_2\big)\Big)\Big),
\end{aligned} \tag{12}$$

where $\mathbb{1}(\cdot)$ is the indicator function that takes 1 if the condition inside is true and 0 otherwise. One can replace the RBF kernel k_{rbf} with other kernels to evaluate the measurement similarity. Notably, to calculate the measurement kernel $k^M\big((i_1,j_1),(i_2,j_2)\big)$, we include the neighboring time slots in $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ to extract features with temporal correlations. Then, we can use above equations to calculate $\boldsymbol{K}$. Specifically, [38] provides an equivalent equation to convert Equations (10) to:

$$\boldsymbol{K}(n_1,n_2) = \boldsymbol{q}_\times^\top (\boldsymbol{I} - \alpha \boldsymbol{B}_\times)^{-1} \boldsymbol{q}_\times, \tag{13}$$

where $\forall 1 \le i \le |V_C|, \boldsymbol{q}_\times(i) = \frac{1}{|V_C|}$ and $\forall |V_C|+1 \le i \le |V_C| + |\widetilde{V_C}|, \boldsymbol{q}_\times(i) = \frac{1}{|\widetilde{V_C}|}$. To calculate the inverse matrix in Equation (13), [38] provides an efficient way by solving Sylvester equations. One can refer to [38] for the procedure and Section 4.2 provides the time complexity of the calculation. Notice that the kernel calculation can be directly implemented on the source and the target graphs. However, the large size in the original graphs causes inefficiency for the computation, which prevents the realistic prediction for the classifier trained with kernel features. Thus, graph coarsening in Section 3.2 is essential to guarantee model efficiency.

Finally, after the calculation of $\boldsymbol{K}$, we can construct the optimization in Equation (7), which can be efficiently solved as a generalized eigendecomposition problem [25, 30]. Then, we summarize the complete learning algorithm in Algorithm 1.

Algorithm 1: Cross-Graph Domain Adaptation

Function *Coarsening-GNA*

Input: Spatial-temporal graphs $\{G^n\}_{n=1}^N$ and $\{\widetilde{G}^n\}_{n=1}^{\widetilde{N}}$;
Hyper-parameters: Distance threshold d_{max} for node aggregation, temporal interval δ_n for temporal correlation integration, parameters of RBF kernel k_{rbf} (or polynomial kernel k_{poly}), and penalty term λ for the Frobenius norm in Equation (7);
Graph coarsening: Utilize Equations (1) to (6) to conduct graph coarsening with d_{max}. Then, obtain the coarser graphs $\{G_C^n\}_{n=1}^N$ and $\{\widetilde{G_C^n}\}_{n=1}^{\widetilde{N}}$;
Compute graph kernels: Utilize data of $\{G_C^n\}_{n=1}^N$ and $\{\widetilde{G_C^n}\}_{n=1}^{\widetilde{N}}$ to calculate weighted adjacency matrix $\boldsymbol{B}_\times$ under Equations (11) and (12). Then, solve Sylvester equations to compute kernel matrix for Equation (13);
Solve GNA: Construct GNA model in Equation (7) and solve the model as a generalized eigendecomposition problem;
Train classifier: Train a classifier f based on the obtained common features $\boldsymbol{W}^\top \boldsymbol{K}$ in Equation (7) and graph labels $\boldsymbol{h}$ and $\widetilde{\boldsymbol{h}}$;
Output: Kernel matrix $\boldsymbol{K}$, linear transformation matrix $\boldsymbol{W}$, and classifier f;

end

4 THEORETICAL ANALYSIS

4.1 Validity of the Proposed Graph Kernel

According to [14, 30], a universal kernel is required to guarantee MMD to be a correct statistic to measure distribution distance. Further, theorems in [30, 34] show that the positive definite kernel matrix can guarantee the kernel is universal. Thus, we have the following theorem.

Theorem 4.1. *The proposed random walk kernel from Equations* (10) *to* (12) *is positive definite and universal.*

Proof. First, we prove the matrix of step kernel k_{step} is positive definite. Equation (12) shows that the step kernel is a multiplication of Dirac kernels and RBF kernels. Since the Dirac and RBF kernels are positive definite [33], and the multiplication preserves the positive definiteness, the step kernel matrix is positive definite. Second, according to [2], the positive definiteness of step kernels leads to the positive definiteness of the proposed random walk kernel, which further implies that the proposed random walk kernel is universal. □

4.2 Computational Cost of the Kernel Matrix

In this subsection, we evaluate the computational cost of the proposed graph kernel. [38] obtain results of Equation (13) by solving Sylvester equations. Due to the space limit, we eliminate the derivations, and one can refer to [38] for more details. Based on the Sylvester equation methods, we propose the following theorem.

Theorem 4.2. *If* $|V|_C \approx |\widetilde{V}_C| \approx M$*, the computational complexity to calculate the proposed graph kernel matrix* $\boldsymbol{K}$ *is* $O\big((N+\widetilde{N})^2(M^3+\delta_n M^4)\big)$ *for Sylvester equation-based method.*

Proof. Equation (12) shows that the calculation of the step kernel has a computational time complexity $O(\delta_n)$. Then, it takes $O(\delta_n M^4)$ to construct $M^2 \times M^2$ weighted adjacency matrix $\boldsymbol{B}_\times$. Further, [38] shows that it takes $O(M^3)$ for the Sylvester equation-based method to calculate one entry of $\boldsymbol{K}$ using the obtained matrix $\boldsymbol{B}_\times$. Thus, for $(N+\widetilde{N}) \times (N+\widetilde{N})$ kernel matrix $\boldsymbol{K}$, the total time complexity is $O\big((N+\widetilde{N})^2(M^3+\delta_n M^4)\big)$. □

4.3 Computational Cost of GNA

For the complete training algorithm, we follow the idea in [25] to report the time complexity.

Theorem 4.3. *If* $|V|_C \approx |\widetilde{V}_C| \approx M$*, the computational complexity for Optimization* (7) *is* $O\big((n_0+K_3+1)(N+\widetilde{N})^2\big)$.

Proof. One can refer to [25] for the computational complexity evaluation of the training algorithm. □

We note that it is possible to further reduce this complexity, by leveraging recent advances on scalable Sylvester equation solvers, such as [8].

5 EXPERIMENT

5.1 Dataset

Power Systems. Power systems transmit electric power from generation sides to load sides. For system profiles, we employ data from Illinois 200-node system and South Carolina 500-node system [9]. The profiles provide the label information for nodes and edges. Specifically, nodes can be categorized into generators, loads, and the slack bus. Edges can be divided into transformers and lines. After simulation, we can obtain nodal measurements of voltage magnitude, angle, and frequency and edge measurements of current magnitude and angle. Finally, the system labels include line trip, generator trip, single-phase fault, phase-to-phase fault, three-phase fault, load shedding, and transformer failure. For simplification, we denote the two systems as $P200$ and $P500$.

Mass-damper Systems. Mass damper systems study the mechanical functions of a structure to reduce the dynamic responses. Using MATLAB, we simulate the dynamic process of two mass-damper systems with 5 nodes and 10 nodes for transfer learning. Then, the force and the speed measurements of nodes are utilized for DA. The labels of the system include edge trip and node trip. For simplification, we denote the two systems as $M5$ and $M10$.

Human Activity Sensing Systems. They are action measuring systems to measure the acceleration and the angular acceleration when a person is conducting an action [43]. In general, the researchers employ 14 subjects with 6 nodes for measuring. The node measurements include 3-axis acceleration and 3-axis angular acceleration data measurements for each time slot, with a total of around 10s to 30s. The labels include acceleration and angular acceleration. The system labels include 12 actions. We study the knowledge transfer between sensing systems of two subjects. For simplification, we denote the two systems as $H_1 6$ and $H_2 6$.

For power systems, graph coarsening is utilized to reduce the system size and increase the model efficiency. For other systems, we directly utilize the raw system data.

5.2 Benchmark Methods and Evaluations

Our GNA model can learn features in the kernelized feature space. Then, we utilize a deep Residual network (Resnet) [15] to conduct the classification task. Further, we use the following benchmark methods for comparison.

- Resnet + Principal Component Analysis (PCA): we utilize PCA to project data into a low-dimensional feature space that has the same dimensionality as features of GNA. Then, the features are input to Resnet *without transfer learning* as a benchmark method. The same Resnet is used for the classification.
- Transfer Component Analysis (TCA) [30] + Resnet: TCA minimizes the MMD of marginal distributions. The same Resnet is used for the classification.
- Joint Distribution Adaptation (JDA) [25] + Resnet: JDA jointly minimizes the MMD of the marginal and conditional distributions. The same Resnet is used for classification.
- Domain Adversarial Neural Network (DANN) [13]: DANN employs adversarial training with Deep Neural Networks (DNNs) to learn domain-invariant features for classification.
- Cross-network Deep Network Embedding (CDNE) [31]: CDNE utilizes DNNs to learn label-discriminative and network-invariant representations. The network structure and networked data are employed.

Table 1: Average test accuracy (%) of cross-system DA for different methods.

	$P500 \rightarrow P200$	$P200 \rightarrow P500$	$M10 \rightarrow M5$	$M5 \rightarrow M10$	$H_16 \rightarrow H_26$	$H_26 \rightarrow H_16$	AVERAGE
GNA + RESNET	**93.28**	**94.34**	**95.53**	**98.87**	**90.41**	**91.23**	**93.93**
JDA + RESNET	86.71	86.19	91.15	90.24	84.82	81.03	86.69
TCA + RESNET	81.25	73.37	89.64	90.08	82.32	83.15	83.30
DANN	86.65	86.53	91.13	89.35	84.51	83.47	86.94
CDNE	88.25	87.93	90.34	91.02	84.33	82.25	87.35
GCAN	83.19	85.57	89.08	90.36	83.46	85.25	86.15
PCA + RESNET	78.75	75.21	87.85	88.96	78.79	77.56	81.19

- Graph Convolutional Adversarial Network (GCAN) [29]: GCAN employs Data Structure Analyzer to project source and target samples into instance graph, defined in Section 2.1.2. The graph captures the similarity between different samples. Then, GNN-based adversarial training is employed to learn invariant features over the graphs.

These methods are inclusive and representative according to the major DA categorizations, covering distribution metric-based (TCA, JDA, and CDNE) and adversarial learning-based (DANN, GCAN) methods. Further, traditional optimizations (TCA and JDA) and deep learning models (DANN, CDNE, and GCAN) are considered. In addition, CDNE considers networked data with structure information and GCAN considers graph structures between samples.

Note that for TCA, JDA, and DANN models, the dimensionality of the source and the target datasets should be the same. Thus, we copy some nodal measurements of the smaller system to align the dimensionality. For model evaluations, we conduct 5-fold cross-validation for the physical datasets and report the average test accuracy as the final score for each method.

5.3 Results of Test Accuracy

In this subsection, we evaluate the average test accuracy for different methods. We utilize the arrow to show the transferring process. For example, $P500 \rightarrow P200$ shows that 500-node power system is the source grid and 200-node power system is the target grid. Then, Table 1 demonstrates the results. In general, our proposed GNA performs the best over other methods, with improvements of 7.24%, 10.63%, 6.99%, 6.58%, 7.78%, and 12.74% compared to JDA, TCA, DANN, CDNE, GCAN, and Resnet without transfer learning. This demonstrates that our design can obtain better common knowledge over graphs to help train the classifier.

Second, by comparing GNA, JDA, and TCA, we find that the graph label information can help improve DA. Specifically, TCA does not include the graph label information, thus inducing over-compressed features with different classes. On the other hand, graph labels re-weight the feature space and encourage adapting distributions within one class, leading to better performances of GNA and JDA. Third, GNA has an even better improvement since the node/edge labels enable a second layer of re-weighting using graph kernels. Namely, GNA can encourage the minimization over nodes/edges from two graphs with the same label.

Fourth, GNA also has a stable improvement compared to deep learning models DANN, CDNE, and GCAN. Though the deep models have a high capacity for extracting domain-invariant features with discriminability, they may not include the structural, temporal, and node/edge label information properly. For example, DANN uses a convolutional kernel to integrate spatial correlations, which can not sufficiently process graph structures. Further, DANN can not incorporate the label information. Thus, DANN lacks the graph information as regularization, and the learned common feature representation could suffer from overfitting. GCAN considers the graph structure over instances. In our setting, it studies the temporal correlations between samples. However, the continuous change of system states encourages us to consider an interval of samples rather than the structures among samples. Thus, GCAN does not perform well. CDNE projects networked data into a common feature space under network embedding, but the embedding vector space is restricted to be the same for minimizing the distribution discrepancy. Thus, there is an information loss for DA between systems of different sizes. In contrast, we design graph kernels to maximally preserve the original information and achieve better performances.

5.4 Ablation Study

In Section 5.3, for power systems, we utilize GNA with graph coarsening to process datasets of node/edge measurements and labels. In this subsection, we conduct an ablation study to understand the effects of different factors. Specifically, we test our models by independently removing the following factors as comparisons: (1) graph coarsening, (2) node measurements, (3) node labels, (4) edge measurements, and (5) edge labels.

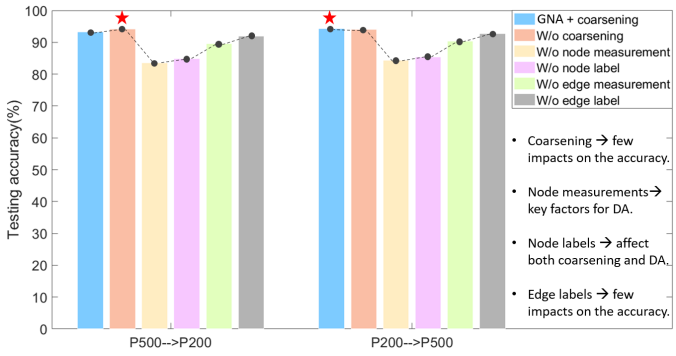
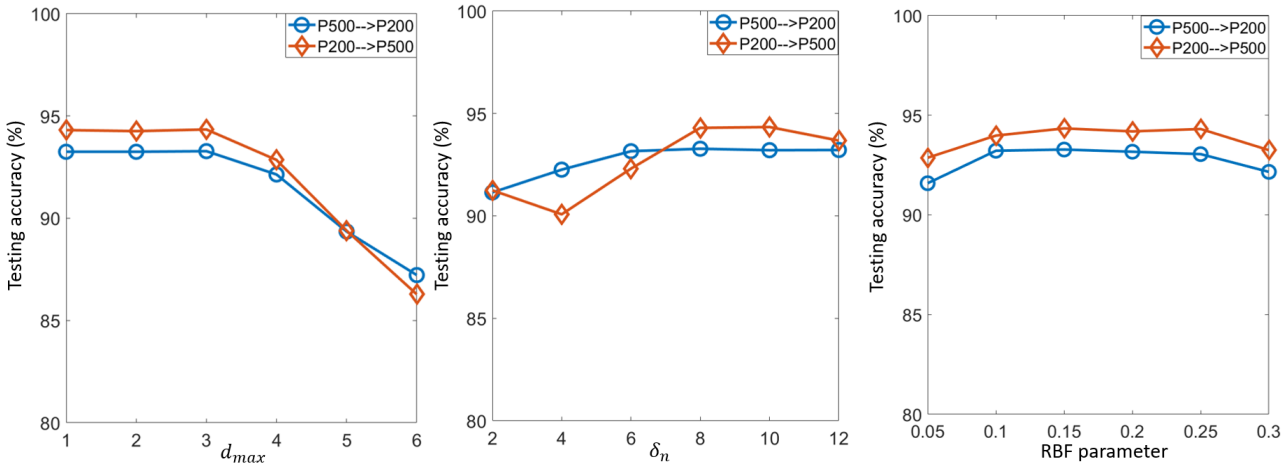


Figure 2: The ablation study for GNA and coarsening.

The result is shown in Fig. 2. We have the following observations. Firstly, if we compare the model with and without coarsening, we find the accuracy does not have a significant change. This is because our coarsening process can correctly concentrate graphs



(a) Accuracy with respect to d_{max}. **(b) Accuracy with respect δ_n.** **(c) Accuracy with respect to k_{rbf} parameters.**

Figure 3: Results of the sensitivity analysis with respect to different hyper-parameters.

based on the local structures and the label information, largely saving the graph properties. For the scenario of $P200 \rightarrow P500$, the coarsening even helps to refine the raw data and improves the performance slightly. However, the running time of computing the complete kernel matrix for the original graphs and the coarser graphs is 19.98min and **0.23**min, respectively. This implies that the coarsening process significantly reduces the computational time.

Secondly, we find that the availability of node measurements and labels is the key to superior performances. If we do not input node measurements, the accuracy reduces by around 9.8%. This implies the nodal measurements, i.e., the voltage magnitude, angle, and frequency, contain important patterns to reflect the graph labels. Further, even if we consider the node measurements but without node labels, the performance reduces by around 8.7%. This is because the node labels are essential to categorize the similarity of node measurements. Specifically, the three types of nodes: generators, loads, and the slack bus, have completely different data distributions. Thus, the labels of nodes not only help to correctly concentrate nodes in the coarsening process but also regularizes the calculation of graph kernels. Namely, based on Equation (12), the node measurements of different types will not be considered together when calculating the step kernel. This demonstrates the importance of importing node labels for power systems.

Thirdly, the non-existence of edge measurements and labels reduces the performance slightly. The edge measurements can be approximately determined by the node measurements due to the underlying physical equations, i.e., power flow equations. Thus, we can still obtain good results with node measurements. Further, the edge labels in power systems bring limited capacity to categorize edge measurements since there is a limited difference between the flow of electric transformers and lines. Thus, the edge labels do not affect the performance too much. This phenomenon brings a potential future direction of studying the roles of physical constraints in the DA to improve the efficiency, shown in Section 6.

5.5 Sensitivity Analysis

In this subsection, we study the model sensitivity with respect to different hyper-parameters. In particular, we investigate the threshold $d_{max} \in \{1, 2, 3, 4, 5, 6\}$ for spatial correlations, the interval range $\delta_n \in \{2, 4, 6, 8, 10, 12\}$ for temporal correlations, and the parameter of the RBF kernel k_{rbf} in the range of $\{0.05, 0.1, 0.15, 0.2, 0.25, 0.3\}$. It is noteworthy that the polynomial kernel generally performs worse than the RBF kernel. Thus, we only report the results of RBF kernel to save space. Fig. 3 demonstrates the final results. Specifically, Fig. 3a shows that when $d_{max} \leq 3$, the accuracy has a small oscillation. Under this distance, nodes can be well grouped. Especially, the node label information can effectively avoid over-grouping and keep the node separate by similarity. When $d_{max} \geq 4$, the accuracy decreases as d_{max} increases. This is because when d_{max} is too large, many non-similar and non-local nodes are aggregated together, which deteriorates the coarsening process.

Fig. 3b shows that when $6 \leq \delta_n \leq 10$, the temporal correlations are well-captured, and our model can achieves the best performance. However, when $\delta_n < 6$, the calculated RBF kernels can not consider the correct system changes to identify system labels. Further, they may be non-robust with respect to measurement noises or anomalies. When $\delta_n > 10$, the incorporation of state change may be too long and prevent the model from understanding system dynamics. Thus, δ_n should be set in a proper range to obtain high performances. Fig. 3c implies that GNA is robust to the change of the RBF kernel parameter: a wide range of parameter values (e.g., $0.1 \sim 0.25$) can lead to a relatively good performance.

6 CONCLUSION AND FUTURE WORK

In this paper, we study the problem of Domain Adaptation (DA) for two physical systems. We show that the spatial-temporal correlations and the label information of nodes and edges jointly determine the graph similarity between two spatial-temporal graphs. With specifically-designed graph kernels, we can integrate all the above information to quantify the similarity of two graphs, which facilitates the learning of common knowledge. We denote our model

as Graph kerNel-based distribution Adaptation (GNA). However, the computational cost of GNA is high for large systems. Thus, we develop a graph coarsening procedure to yield coarser graphs with much smaller sizes and preserved graph similarity. We also have the following future work. Firstly, the edge-level and node-level tasks need to be further investigated for DA between graphs. Secondly, we can embed the designed graph kernel into deep learning methods like GNN, which can benefit from both the high capacity of similarity measuring and the power of hierarchical feature extractions.

ACKNOWLEDGMENTS

This work is supported by Nation Science Foundation (NSF) under grants No. 1947135, No. 2134079, No. 1939725, ECCS-1810537, and ECCS-2048288, and the Department of Energy (DOE) under grants DE-AR00001858-1631 and DE-EE0009355.

REFERENCES

[1] Firoj Alam, Shafiq Joty, and Muhammad Imran. 2018. Domain adaptation with adversarial training and graph embeddings. *arXiv preprint arXiv:1805.05151* (2018).
[2] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. 2005. Protein function prediction via graph kernels. *Bioinformatics* 21, suppl_1 (2005), i47–i56.
[3] Gecia Bravo Hermsdorff and Lee Gunderson. 2019. A unifying framework for spectrum-preserving graph sparsification and coarsening. *Advances in Neural Information Processing Systems* 32 (2019).
[4] Chen Cai, Dingkang Wang, and Yusu Wang. 2021. Graph Coarsening with Neural Networks. *arXiv preprint arXiv:2102.01350* (2021).
[5] Yiming Chen, Shiji Song, Shuang Li, and Cheng Wu. 2019. A graph embedding framework for maximum mean discrepancy-based domain adaptation algorithms. *IEEE Transactions on Image Processing* 29 (2019), 199–213.
[6] Quanyu Dai, Xiao-Ming Wu, Jiaren Xiao, Xiao Shen, and Dan Wang. 2022. Graph Transfer Learning via Adversarial Domain Adaptation with Graph Convolution. *IEEE Transactions on Knowledge and Data Engineering* (2022).
[7] Zhengming Ding, Sheng Li, Ming Shao, and Yun Fu. 2018. Graph adaptive knowledge transfer for unsupervised domain adaptation. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 37–52.
[8] Boxin Du and Hanghang Tong. 2018. Fasten: Fast sylvester equation solver for graph mining. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1339–1347.
[9] Engineering Texas A&M University. 2016. SouthCarolina 500-Bus System: ACTIVSg500. (2016). https://electricgrids.engr.tamu.edu/electric-grid-test-cases/activsg500/
[10] Meng Fang, Jie Yin, and Xingquan Zhu. 2013. Transfer learning across networks for collective classification. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, 161–170.
[11] Chenbo Fu, Yongli Zheng, Yi Liu, Qi Xuan, and Guanrong Chen. 2019. Nes-tl: Network embedding similarity-based transfer learning. *IEEE Transactions on Network Science and Engineering* 7, 3 (2019), 1607–1618.
[12] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17, 1 (2016), 2096–2030.
[13] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17, 1 (2016), 2096–2030.
[14] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *Journal of Machine Learning Research* (2012).
[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*. 770–778.
[16] Youngeun Kim and Sungeun Hong. 2021. Adaptive graph adversarial networks for partial domain adaptation. *IEEE Transactions on Circuits and Systems for Video Technology* 32, 1 (2021), 172–182.
[17] Nils M Kriege, Fredrik D Johansson, and Christopher Morris. 2020. A survey on graph kernels. *Applied Network Science* 5, 1 (2020), 1–42.
[18] Jaekoo Lee, Hyunjae Kim, Jongsun Lee, and Sungroh Yoon. 2017. Transfer learning for deep learning on graph-structured data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.
[19] Haoran Li, Zhihao Ma, and Yang Weng. 2022. A Transfer Learning Framework for Power System Event Identification. *IEEE Transactions on Power Systems* (2022), 1–1. https://doi.org/10.1109/TPWRS.2022.3153445
[20] Haoran Li and Yang Weng. 2021. Physical Equation Discovery Using Physics-Consistent Neural Network (PCNN) Under Incomplete Observability. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 925–933.
[21] Haoran Li, Yang Weng, Evangelos Farantatos, and Mahendra Patel. 2019. A Hybrid Machine Learning Framework for Enhancing PMU-based Event Identification with Limited Labels. In *IEEE International Conference on Smart Grid Synchronized Measurements and Analytics*. 1–8.
[22] H. Li, Y. Weng, E. Farantatos, and M. Patel. 2019. An Unsupervised Learning Framework for Event Detection, Type Identification and Localization Using PMUs Without Any Historical Labels. In *IEEE Power Energy Society General Meeting*. 1–5. https://doi.org/10.1109/PESGM40551.2019.8973580
[23] Haoran Li, Yang Weng, and Hanghang Tong. 2020. Heterogeneous Transfer Learning on Power Systems: A Merged Multi-modal Gaussian Graphical Model. In *2020 IEEE International Conference on Data Mining (ICDM)*. 1088–1093. https://doi.org/10.1109/ICDM50108.2020.00130
[24] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael Jordan. 2015. Learning transferable features with deep adaptation networks. In *International conference on machine learning*. PMLR, 97–105.
[25] Mingsheng Long, Jianmin Wang, Guiguang Ding, Jiaguang Sun, and Philip S Yu. 2013. Transfer feature learning with joint distribution adaptation. In *Proceedings of the IEEE international conference on computer vision*. 2200–2207.
[26] Mingsheng Long, Han Zhu, Jianmin Wang, and Michael I Jordan. 2016. Unsupervised Domain Adaptation with Residual Transfer Networks. In *NIPS*.
[27] Andreas Loukas. 2019. Graph Reduction with Spectral and Cut Guarantees. *J. Mach. Learn. Res.* 20, 116 (2019), 1–42.
[28] Andreas Loukas and Pierre Vandergheynst. 2018. Spectrally approximating large graphs with smaller graphs. In *International Conference on Machine Learning*. PMLR, 3237–3246.
[29] Xinhong Ma, Tianzhu Zhang, and Changsheng Xu. 2019. Gcan: Graph convolutional adversarial network for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8266–8276.
[30] Sinno Jialin Pan, James T Kwok, Qiang Yang, et al. 2008. Transfer learning via dimensionality reduction.. In *AAAI*, Vol. 8. 677–682.
[31] Xiao Shen, Quanyu Dai, Sitong Mao, Fu-Lai Chung, and Kup-Sze Choi. 2021. Network Together: Node Classification via Cross-Network Deep Network Embedding. *IEEE Transactions on Neural Networks and Learning Systems* 32, 5 (2021), 1935–1948. https://doi.org/10.1109/TNNLS.2020.2995483
[32] Changjian Shui, Qi Chen, Jun Wen, Fan Zhou, Christian Gagné, and Boyu Wang. 2020. Beyond H-Divergence: Domain Adaptation Theory With Jensen-Shannon Divergence. *arXiv preprint arXiv:2007.15567* (2020).
[33] Alex J Smola and Bernhard Schölkopf. 1998. *Learning with kernels*. Vol. 4. Citeseer.
[34] Le Song. 2008. Learning via Hilbert space embedding of distributions. (2008).
[35] Baochen Sun, Jiashi Feng, and Kate Saenko. 2017. Correlation alignment for unsupervised domain adaptation. In *Domain Adaptation in Computer Vision Applications*. Springer, 153–171.
[36] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 7167–7176.
[37] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. 2014. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474* (2014).
[38] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *Journal of Machine Learning Research* 11 (2010), 1201–1242.
[39] Fangzhao Wu and Yongfeng Huang. 2016. Sentiment domain adaptation with multiple sources. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 301–310.
[40] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.
[41] Pengcheng Xu, Prudhvi Gurram, Gene Whipps, and Rama Chellappa. 2019. Wasserstein distance based domain adaptation for object detection. *arXiv preprint arXiv:1909.08675* (2019).
[42] Chaohui Yu, Jindong Wang, Yiqiang Chen, and Meiyu Huang. 2019. Transfer learning with dynamic adversarial adaptation network. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 778–786.
[43] Mi Zhang and Alexander A. Sawchuk. 2012. USC-HAD: A Daily Activity Dataset for Ubiquitous Activity Recognition Using Wearable Sensors. In *ACM International Conference on Ubiquitous Computing (Ubicomp) Workshop on Situation, Activity and Goal Awareness (SAGAware)*. Pittsburgh, Pennsylvania, USA.
[44] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1345–1354.