Graph Stream Classification using Labeled and Unlabeled Graphs

Shirui Pan†, Xingquan Zhu†, Chengqi Zhang†, Philip S. Yu⋆

†*Centre for Quantum Computation & Intelligent Systems, FEIT, University of Technology, Sydney*
{shirui.pan@student., xingquan.zhu@, chengqi.zhang@}uts.edu.au

⋆*Dept. of Computer Science, University of Illinois at Chicago*
psyu@cs.uic.edu

***Abstract*— Graph classification is becoming increasingly popular due to the rapidly rising applications involving data with structural dependency. The wide spread of the graph applications and the inherent complex relationships between graph objects have made the labels of the graph data expensive and/or difficult to obtain, especially for applications involving dynamic changing graph records. While labeled graphs are limited, the copious amounts of unlabeled graphs are often easy to obtain with trivial efforts. In this paper, we propose a framework to build a stream based graph classification model by combining both labeled and unlabeled graphs. Our method, called gSLU, employs an ensemble based framework to partition graph streams into a number of graph chunks each containing some labeled and unlabeled graphs. For each individual chunk, we propose a minimum-redundancy subgraph feature selection module to select a set of informative subgraph features to build a classifier. To tackle the concept drifting in graph streams, an instance level weighting mechanism is used to dynamically adjust the instance weight, through which the subgraph feature selection can emphasize on difficult graph samples. The classifiers built from different graph chunks form an ensemble for graph stream classification. Experiments on real-world graph streams demonstrate clear benefits of using minimum-redundancy subgraph features to build accurate classifiers. By employing instance level weighting, our graph ensemble model can effectively adapt to the concept drifting in the graph stream for classification.**

I. Introduction

Recent years have witnessed an increasing number of applications where instances (or samples) are no longer represented by a flat table with instance-feature format, but share some complex structural dependency relationships. Typical examples include XML webpages (*i.e.* an instance) which point to (or are pointed to) several other XML webpages [1], a scientific publication with a number of references [2], posts and responses generated from social networking sites (*e.g.* Tweets generated from Twitter [3]), and chemical compounds with molecules (*i.e.* nodes) linked together through some bounds (*i.e.* edges) [4], [5]. Given a collection of graph data $\{G_i, y_i\}, y_i \in \{+1, -1\}$, each of which is suitably labeled (+1 for positive graphs, -1 for negative graphs), graph classification aims to build prediction model with maximum accuracy in classifying previously unseen graphs. The main challenge of graph classification lies on the lack of the instance-feature representation for graph data, so most existing learning algorithms cannot be directly used for graph classification.

To enable graph classification, existing methods mainly fall into two categories (1) subgraph feature based methods, and (2) distance based methods (including graph kernel, graph embedding, and transformation). The former uses a set of subgraph features to transfer graph into vector space so existing machine learning methods can be applied, and the latter uses distance measures to assess similarities between each pair of graphs for classification. Some empirical studies [6] have shown that subgraph feature based methods are generally superior to distance based algorithms, and a large number of methods exist to extract subgraph features for classification including some graph boosting approaches [7], [8], [5].

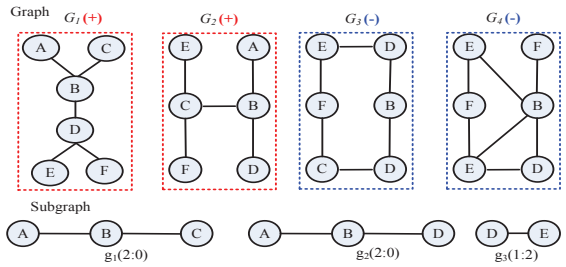

Fig. 1. An example demonstrating subgraph correlations: G_1 and G_2 are positive graphs (+), G_3 and G_4 are negative graphs (-). Each subgraph, g_1, g_2, or g_3, is marked with its frequency occurring in positive *v.s.* negative graphs.

When selecting subgraph features, common methods use an evaluation metrics, such as the frequency, to select a number of important subgraph features. The graph data can then be represented by using the selected features in a vector space (depending on whether a graph contains specific features or not). While a large number of subgraph feature selection methods exist, they consider subgraphs as independent observations without realizing that subgraph features are normally generated from the same set of graphs. As a result, subgraph features may share high correlations, which is one of the major factors attributed to the performance loss of learning methods. As shown in Fig. 1, subgraphs g_1 and g_2 only appear in positive graphs, whereas g_3 appears in both positive and negative graphs. If evaluated separately, g_1, g_2 are more informative than g_3. However, g_1 and g_2 are highly correlated and redundant. A good approach is to include either g_1 and g_3, or g_2 and g_3 to form a two feature set for graph classification.

In stream scenario, subgraph feature selection can be even

more complicated. This is because the decision concepts of the graph data may gradually (or rapidly) change, *i.e.* the concept drifting in the stream. In order to rapidly capture the concept changes in the graph stream, the graph feature selection module should take the dynamic graph stream as a whole to select informative and less redundant features. Unfortunately, existing graph feature selection methods all work on static graph set. No effective strategy exists to select informative subgraph features from graph streams. While a trivial solution is to partition graph stream into a number of stationary subsets (or chunks) and carry out feature selection in each individual subset, such a simple solution does not allow graph subsets to collaborate with each other to select highly effective subgraph features for graph streams.

In summary, when selecting subgraph features from graph stream for classification, we should take the following three factors into consideration to ensure that the whole framework is effective and efficient.

- **Identifying informative subgraphs with minimum redundancy:** Finding a set of informative subgraph features with minimum redundancy can ensure that the succeeding learning methods can achieve high accuracy for graph stream classification.
- **Capturing concept drifting in streams:** Concept drifting represents emerging changes in the streams. Our model must be able to capture such changes and emphasize on the instances represented by drifting concepts.
- **Combining labeled and unlabeled graphs:** The continuous increasing volumes of the graph stream makes the graph labels very difficult to obtain. Our model should take advantage of the large quantify of unlabeled graphs to boost graph stream classification.

Motivated by the above observations, we report in this paper, gSLU, a graph stream classification framework using both labeled and unlabeled graphs. Instead of limiting subgraph features to labeled samples, gSLU combines both labeled and unlabeled graphs to generate a rich set of subgraph features for assessment. To identify informative subgraphs with minimum redundancy, we propose a subgraph assessment criterion to assess the informativeness of individual features and the redundancy of the whole feature set at the same time. To capture instances represented by emerging concept drifting in graph streams, we employ a dynamic instance weighting mechanism, where graphs misclassified by the existing model receive a higher weight so the subgraph feature selection can emphasize on difficult graphs to find effective features to represent them. Experiments on real-world applications demonstrate that gSLU is effective for selecting informative and minimum-redundancy subgraph features to build accurate classifiers. The proposed graph ensemble model is able to tackle concept drifting in graph streams for classification.

The remainder of the paper is structured as follow. The problem definition and the overall framework is discussed in Sec. II. Sec. III reports the proposed subgraph feature assessment criterion. The gSLU framework is reported in Sec. IV, followed by the experiments in Sec. V. Sec. VI reviews related work, and we conclude the paper in Sec. VII.

II. Problem Definition & Overall Framework

In this section, we define several important notations and discuss overfall framework for graph stream classification.

DEFINITION 1 **Connected Graph**: A graph is represented as $G = (\mathcal{V}, E, \mathcal{L})$ where $\mathcal{V}$ is a set of vertices $\mathcal{V} = \{v_1, \cdots, v_{n_v}\}$, $E \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, and $\mathcal{L}$ is the set of symbols for the vertices and the edges. A connected graph is a graph such that there is a path between any pair of vertices.

A graph G_i is a labeled graph, if a class label $y_i \in \mathcal{Y}$ is assigned to G_i. For binary classification problem, we have $y_i \in \mathcal{Y} = \{-1, +1\}$. A graph G_i is either labeled (denoted by G_i^l) or unlabeled (denoted by G_i^u). In this paper, we also use G^l and G^u to denote labeled and unlabeled graphs, respectively.

DEFINITION 2 **Graph Stream**: A graph stream $\mathcal{S} = \{\cdots, G_i, G_{i+1}, G_{i+2}, \cdots\}$ contains an increasing number of graphs in a streaming fashion. In this paper, we consider that a graph chunk $D_t = D_t^l \bigcup D_t^u$ contains a fixed number of graphs collected from a consecutive stream region, where D_t^l and D_t^u denote labeled and unlabeled graphs, respectively.

By using the graph chunk notation, a graph stream $\mathcal{S}$ can be denoted as a collection of chunks $\mathcal{S} = \{D_1, \cdots, D_t\}$.

DEFINITION 3 **Subgraph**: Let $G = (\mathcal{V}, E, \mathcal{L})$ and $g_i = (\mathcal{V}', E', \mathcal{L}')$ each denotes a connected graph. g_i is a subgraph of G, *i.e.*, $g_i \subseteq G$, iff (1) $\mathcal{V}' \subseteq \mathcal{V}$, (2) $E' \subseteq E$, and (3) $\mathcal{L}' \subseteq \mathcal{L}$. If g_i is a subgraph of G, then G is a supergraph of g_i.

DEFINITION 4 **Subgraph Features**: Let $\mathbf{g} = \{g_1, \cdots, g_m\}$ denote a set of subgraph patterns discovered from a given graph set (In this paper, subgraph patterns and subgraph features are equivalent terms). For each graph G_i, we use a subgraph feature vector $x_i = [x_i^{g_1}, \cdots, x_i^{g_m}]$ to represent G_i in the feature space, where $x_i^{g_k} = 1$ iff g_k is a subgraph of G_i (*i.e.* $g_k \subseteq G_i$) and $x_i^{g_k} = 0$ otherwise.

In Fig. 1, if we use g_1, g_2, and g_3 as subgraph features, the feature vector for G_2 is denoted by $x_1 = [1, 1, 0]$.

DEFINITION 5 **Pearson's Correlation Coefficient**: Given two subgraph patterns g_p and g_q, and a graph set D, the Pearson's Correlation Coefficient [9] between g_p and g_q over the graph set D is defined as:

$$\phi(g_p, g_q) = \frac{N^D N^D_{g_p,g_q} - N^D_{g_p} N^D_{g_q}}{\sqrt{N^D_{g_p}(N^D - N^D_{g_p}) N^D_{g_q}(N^D - N^D_{g_q})}} \quad (1)$$

In Eq.(1), N^D denotes the total number of graphs in graph set D. $N^D_{g_p}$, $N^D_{g_q}$, and $N^D_{g_p,g_q}$ denote the number of graphs in D containing g_p, g_q, and g_p and g_q, respectively.

Subgraphs with high Pearson's correlation coefficient indicate that subgraphs co-occur in the same graphs. So Pearson's correlation provides a measure to assess the redundancy of the subgraph features, regardless of the labels of the underlying graph set.

Given a graph stream $\mathcal{S} = \{D_1, D_2, \cdots, D_t\}$ with a number of consecutive graph chunks, each containing some

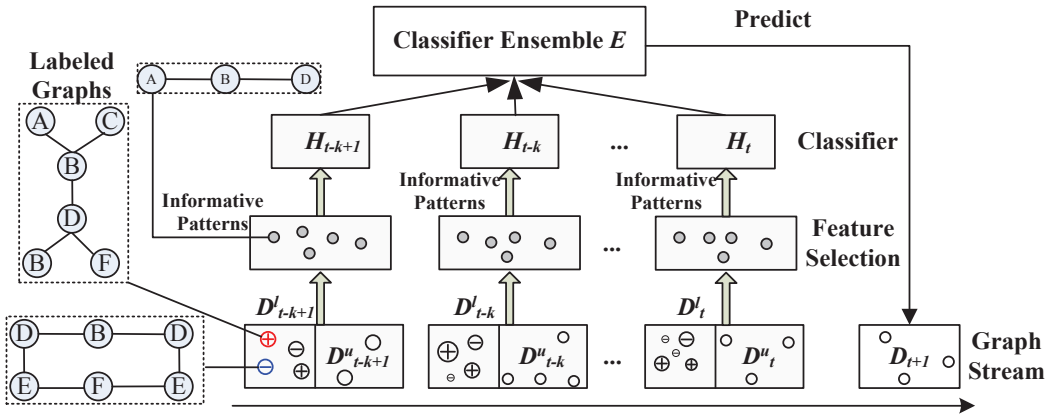

Fig. 2. A framework for semi-supervised graph data stream classification. The graph stream is divided into chunks. In each chunk $D_t = D_t^l \cup D_t^u$, the circles with '+' indicate positive graphs, and the circles with '-' are negative graphs. The size of a circle represents its weight in the chunk. In our graph stream scenario, the weight of a graph is dynamically tuned by an ensemble of classifiers built in previous chunks. In the current chunk, by taking the weight of each graph into consideration, we select a set of optimal informative features with minimum redundancy. An ensemble of classifier is built from the most recent k chunks to predict graphs in the yet-to-come chunk.

labeled and unlabeled graphs, the **aim** of the graph stream classification is to build a prediction model from the most recently observed k chunks $(D_{t-k+1}, \cdots, D_{t-1}, D_t)$ to predict graphs in the next chunk D_{t+1} with the maximum accuracy.

A. Overall Framework

In this paper, we propose an ensemble framework, with a set of minimum-redundancy subgraph features being extracted from each graph chunk to train ensemble classifiers. Our framework, as shown in Fig. 2, contains three key components: (1) partitioning graph stream into chunks, (2) selecting informative and minimum-redundancy subgraph features from each chunk, and (3) forming an ensemble model by combining classifiers trained from individual chunks. To put all three components into a unified framework, we employ an instance weighting mechanism to allow multiple graph chunks to work in a collaborative way to tackle concept drifting in graph streams. As soon as a graph chunk D_t is collected, the overall framework proceeds as follows:

- **Instance Weighting:** We use models trained from the past graph chunks to carefully weight graphs in the most recent chunk D_t with misclassified graphs (*i.e.* samples representing concept drifting in stream) receiving higher weight values.
- **Subgraph Feature Selection:** A set of informative subgraph features with minimum-redundancy are selected to represent the weighted graphs in the current chunk D_t.
- **Updating Ensemble:** By using selected features, a classifier $\mathcal{H}_t$, is trained from chunk D_t and is included into the ensemble to predict graphs in a future chunk D_{t+1}

In following sections, we first propose our subgraph feature selection module, and then discuss detailed procedures of gSLU in Sec. IV.

III. Minimum Redundancy Subgraph Feature Selection

Given a chunk of graph data D_t, let $\mathcal{F}_t$ denote the complete set of subgraphs in D_t, and $\mathbf{g} = \{g_1, \cdots, g_m\}$ be a small set of subgraphs selected from $\mathcal{F}_t$. Our subgraph feature selection aims to simultaneously achieve two goals: (1) maximize the informativeness (discriminative power) of the selected feature set $\mathbf{g}$ for classification, and (2) minimize the redundancy between subgraph features in $\mathbf{g}$. Let $\mathcal{I}(\mathbf{g})$ be a function to measure the informativeness of $\mathbf{g}$, and $\mathcal{R}(\mathbf{g})$ be a function to assess the redundancy in $\mathbf{g}$. The above objective can be formalized in Eq.(2), where $|\cdot|$ represents the cardinality of a set, and m is the number of features to selected from D_t.

$$\begin{aligned} \mathbf{g}^\star &= \arg\max_{\mathbf{g}\subset\mathcal{F}_t}(\mathcal{I}(\mathbf{g})) \\ s.\ t.\quad & (1)\ |\mathbf{g}| \le m \ and \\ & (2)\ \mathcal{R}(\mathbf{g}) \le \mathcal{R}(\mathbf{g}'), \forall \mathbf{g}' \subset \mathcal{F}_t,\ |\mathbf{g}'| == |\mathbf{g}|,\ \&\ \mathbf{g}' \neq \mathbf{g} \end{aligned} \tag{2}$$

The objective function in Eq.(2) indicates that the optimal subgraph features $\mathbf{g}^\star$ should have (1) maximum discriminative power, *i.e.*, $\max(\mathcal{I}(\mathbf{g}))$, and (2) minimum redundancy between subgraph features, *i.e.*, $\min(\mathcal{R}(\mathbf{g}))$.

A. Informativeness of the Feature Set

To discover the set of informative features, we need to measure the informativeness of a feature set $\mathbf{g}$, *i.e.*, $\mathcal{I}(\mathbf{g})$. To calculate $\mathcal{I}(\mathbf{g})$, we impose constraints on the labeled graphs in D_t. For two graphs G_i and G_j, if they have the same class labels, there is a pairwise must-link constraint between G_i and G_j in a must-link set $\mathcal{M}$. If G_i and G_j have different class labels, there is a cannot-link constraint between them in a cannot-link set $\mathcal{C}$. Such an idea is previously employed in constraint based clustering [10]. If we take labeled D_t^l and unlabeled D_t^u graphs in D_t into consideration, a good feature set should satisfy constraints as follows.

- **Weighted Must Link:** if there is a must link between G_i and G_j, their subgraph feature vectors x_i and x_j should be similar to each other. In a data stream scenario, each graph G_i is associated with a weight value w_i. For each graph pair, the higher the total weight of two graphs in the must link set, the more impact the constraint will have for selecting features to represent their resemblance.
- **Weighted Cannot Link:** if there is a cannot link between G_i and G_j, their subgraph feature vectors x_i and x_j should be distinct from each other. The higher the total weight of G_i and G_j, the more impact the constraint will have for selecting features to represent the distinctness between the two graphs.
- **Weighted Separability:** if two graphs G_i and G_j are unlabeled, their subgraph feature vectors x_i and x_j should be different. It is similar to PCA's assumption, which aims to find the component with largest possible variance.

By integrating instance weight, we can adjust the weight of the graphs to emphasize on some important samples. As a result, our framework can effectively capture the concept drifting underlying the graph stream. In this section, we take instance weight as given values. In Section IV, we will provide solutions for automatically calculating the instance weight.

Taking the above constraints into consideration, we derive a criterion for measuring the informativeness as follows:

$$\begin{aligned}\mathcal{I}(\mathbf{g}) = &\frac{1}{2A}\sum_{y_iy_j=-1}(x_i-x_j)^2(w_i+w_j)\\ &-\frac{1}{2B}\sum_{y_iy_j=1}(x_i-x_j)^2(w_i+w_j)\\ &+\frac{1}{2C}\sum_{G_i,G_j\in D_t^U}(x_i-x_j)^2(w_i+w_j)\end{aligned} \tag{3}$$

where w_i and w_j are the weights for G_i and G_j, respectively. $A=\sum_{y_iy_j=-1}(w_i+w_j)$, $B=\sum_{y_iy_j=1}(w_i+w_j)$, and $C=\sum_{G_i,G_j\in D_t^U}(w_i+w_j)$. A, B, and C assess the total weights of constraints in the must-link, cannot-link, and unlabeled set.

Significance of Unlabeled Graphs: A key property of Eq. (3) is that it considers both labeled and unlabeled graphs. The benefit of unlabeled graphs is two-fold: (1) because incorporating unlabeled graphs significantly increases the total number of graphs, it will alleviate the graph data sparsity issue and help enrich the frequent subgraphs. As a result, the subgraph feature space becomes more dense, through which a good set of subgraph features can be discovered; (2)because we emphasize on features which can better separate unlabeled graphs, we can collect a set of more informative subgraph features out of the frequent features.

By integrating weight values in Eq.(3), we define a weighted similarity matrix $W=[W_{ij}]^{n\times n}$ as follows,

$$W_{ij}=\begin{cases}\frac{w_i+w_j}{A} & : \ y_iy_j=-1\\ -\frac{w_i+w_j}{B} & : \ y_iy_j=1\\ \frac{w_i+w_j}{C} & : \ G_i,G_j\in D_t^U\\ 0 & : \ otherwise\end{cases} \tag{4}$$

Accordingly, Eq. (3) can be rewritten as follows,

$$\begin{aligned}\mathcal{I}(\mathbf{g}) &= \frac{1}{2}\sum_{y_iy_j}(x_i-x_j)^2W_{ij}\\ &= tr(X(T-W)X^T)\\ &= tr(XLX^T)\\ &= \sum_{g_p\in\mathbf{g}} f_{g_p}Lf_{g_p}^T\end{aligned} \tag{5}$$

In Eq.(5), tr denotes the trace of a matrix. $X=[x_1,x_2,\cdots,x_n]$ is the matrix consisting binary feature vectors represented D_t. T is diagonal weighted degree matrix of W, *i.e.*, $T_{ii}=\sum_j W_{ij}$. $L=T-W$ is known as a Laplacian matrix. f_{g_p} is an indicator vector of subgraph g_p with respect to all graphs G_i in chunk D_t, *i.e.*, $f_{g_p}=[f_{g_p}^{G_1},f_{g_p}^{G_2},\cdots,f_{g_p}^{G_n}]$, where $f_{g_p}^{G_i}=1$ iff $g_p\subseteq G_i$; otherwise $f_{g_p}^{G_i}=0$.

DEFINITION 6 **gScore:** Given a weighted matrix W as defined in Eq.(4) and a graph chunk D_t, the informativeness score of a subgraph g_p is defined in Eq.(6) where f_{g_p} is an indicator vector of g_p in D_t, $T=diag(d_i)$ is a diagonal matrix with $d_i=\sum_j W_{ij}$, and $L=T-W$ is a Laplacian matrix.

$$\mathbb{i}(g_p)=f_{g_p}Lf_{g_p}^T \tag{6}$$

In order to find the subgraph feature set $\mathbf{g}$ which maximizes the informativeness $\mathcal{I}(\mathbf{g})$ as defined in Eq. (5), we can calculate gScore values of all subgraphs in $\mathcal{F}_s$ and sort them, according to their gScore, in a descending order, *i.e.*, $\mathbb{i}(g_1)\geq\mathbb{i}(g_2)\cdots\geq\mathbb{i}(g_{|\mathcal{F}_t|})$. By using the top-$m$ features $\mathbf{g}=\{g_1,g_2,\cdots,g_m\}$, we can maximize $\mathcal{I}(\mathbf{g})$.

B. Informative Subgraph Feature Selection

To obtain frequent subgraph set $\mathcal{F}_s$ from D_t, we employ a Depth-First-Search (DFS) based algorithm gSpan [11] to enumerate subgraphs. The key idea of gSpan is that each subgraph has a unique DFS Code, which is defined by a lexicographic order of the discovery time during the search process. Two subgraphs are isomorphism iff they have the same minimum DFS Code. By employing a depth first search strategy on the DFS Code tree (where each node is a subgraph), gSpan can effectively enumerate all frequent subgraphs efficiently.

Intuitively, to maximize $\mathcal{I}(\mathbf{g})$ for subgraph feature selection, a simple solution is to use gSpan to discover frequent subgraph set $\mathcal{F}_s$ from each graph chunk D_t, and constantly maintain the feature set $\mathbf{g}$ with the maximum gScore during the frequent subgraph search process. In other words, assume the feature set $\mathbf{g}$ already contains m subgraph features with g_{min} being the subgraph having the minimum gScore in $\mathbf{g}$. For each newly explored subgraph g_p, if g_p's gScore is larger than $\mathbb{i}(g_{min})$, the algorithm will replace g_{min} with g_p to ensure that $\mathcal{I}(\mathbf{g})$ is maximized. The similar approach has, in fact, been employed in a previous work [12].

Take Fig. 3 as an example, because g_{12} has the lowest gScore (the value showing in the circle), it will be replaced by subgraphs with a higher gScore value to maximize $\mathcal{I}(\mathbf{g})$ for feature selection. In fact, there are a number of highly

Algorithm 1 Minimum Redundancy Subgraph Feature Selection

Require:
D_t: A graph data chunk;
min_sup: The threshold of the frequent subgraph;
m: the number of features to be selected;
Ensure:
$\mathbf{g} = \{g_1, g_2, \cdots, g_m\}$: A set of features;
1: $\mathbf{g} = \emptyset$, $\tau = 0$;
2: **while** Recursively visit the DFS Code Tree in gSpan **do**
3: $g_p \leftarrow$ current visited subgraph in DFS Code Tree;
4: **if** $freq(g_p) < min\_sup$ **then**
5: **continue**;
6: **end if**
7: Compute the gScore $\mathbb{i}(g_p)$ for subgraph g_p;
8: **if** $|\mathbf{g}| < m$ or $\mathbb{i}(g_p) > \tau$ **then**
9: $\mathbf{g} \leftarrow \mathbf{g} \bigcup g_p$;
10: **end if**
11: **if** $|\mathbf{g}| > m$ **then**
12: $g_q \leftarrow Subgraph\_Redundancy\_check(\mathbf{g})$; //Algorithm 2
13: $\mathbf{g} \leftarrow \mathbf{g}/g_q$;
14: **end if**
15: $\tau = \min_{g_i \in \mathbf{g}} \mathbb{i}(g_i)$;
16: Depth-first search the subtree rooted from node g_p;
17: **end while**
18: **return** $\mathbf{g}$;

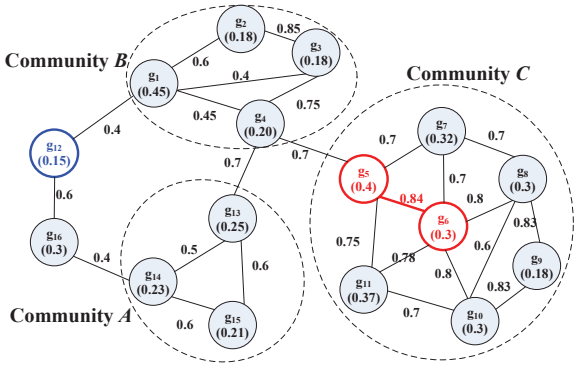

Fig. 3. An example of subgraph feature set $\mathbf{g} = \{g_1, g_2, \cdots, g_m\}$. Each node represents a subgraph g_i and its gScore value $\mathbb{i}(g_i)$, and each edge denotes the Pearson's correlation coefficient between two subgraphs. Edges with low value are omitted for clear presentation. Without considering the correlation among features, one will delete the feature g_{12}(blue color) as it has the minimum gScore value $\mathbb{i}(g_{12})$. By considering feature correlations, feature g_6 (red color) will be deleted, because (1) g_6 is located in a dense and highly correlated group, and (2) g_6 has the highest correlation with its neighbor g_5 and its informativeness value is smaller than g_5.

correlated subgraph features, such as g_5, g_6, g_7 and g_{11}, and a better approach is to replace one of the highly correlated subgraph features to ensure high informativeness and low redundancy features to be included in $\mathbf{g}$.

C. Minimum Redundancy Subgraph Feature Selection

In order to discover the most informative subgraph feature subset $\mathbf{g}$ with minimum redundancy, as defined in Eq.(2), we take feature correlations into consideration, and aim to minimize the correlation between features during the subgraph feature exploration process. This objective is achieved through a two-step optimization process as follows,

- **Maximize** $\mathcal{I}(\mathbf{g})$: Explore new informative subgraph g_p whose gScore value is greater than the minimum gScore of the current feature set $\mathbf{g}$.
- **Minimize** $\mathcal{R}(\mathbf{g})$: If some highly correlated subgraphs exist, replace g_p with a subgraph having the highest redundancy; otherwise, replace g_p with the subgraph having the minimum gScore.

Algorithm 1 lists the proposed minimum redundancy subgraph feature selection method. The algorithm starts with an empty feature set $\mathbf{g}$ and the minimum gScore $\tau = 0$, and continuously enumerates subgraphs by recursively visiting the DFS Code Tree in gSpan algorithm. If a subgraph g_p is not a frequent subgraph, both g_p and its subtree will be pruned (line 4-6). Otherwise, we calculate g_p's gScore value $\mathbb{i}(g_p)$. If $\mathbb{i}(g_p)$ is larger than τ or the feature set $\mathbf{g}$ has less than m subgraphs (*i.e.* $\mathbf{g}$ is not full), we add g_p to the feature set $\mathbf{g}$ (lines 8-10). Meanwhile, if the size of $\mathbf{g}$ exceeds the predefined size m, we need to remove one feature g_q which is highly correlated to other features in $\mathbf{g}$ and has less discriminative power (lines 11-14) (the detailed procedures to discover g_q is discussed in Section III-C.1 and Algorithm 2). After that, the algorithm continues the depth-first search by following the children of g_p (line 16), and continues until the frequent subgraph mining process by gSpan is completed.

The above process not only maintains the set of subgraphs with high discriminative power, but also minimizes the feature redundancy. As a result, more useful subgraph features can be used to cover graphs for classification. In the example showing in Fig. 3, instead of removing g_{12} which has the lowest gScore value, our method will remove g_6 to minimize the redundancy in the feature set $\mathbf{g}$.

1) Subgraph Redundancy Check: When a new informative subgraph is added into feature set $\mathbf{g} = \{g_1, g_2, \cdots, g_m\}$, we need to remove a subgraph g_q from $\mathbf{g}$ if $|\mathbf{g}| > m$. To minimize the redundancy, the subgraph g_q should satisfy the following two conditions: (1) *high correlation:* g_q should be highly correlated to other subgraphs, and (2) *low informativeness:* g_q should has low discriminative power for classification.

In order to discover a subgraph g_q highly correlated to other graphs, we employ the community detection principle in the social network analysis [13] to discover groups (or communities) from the subgraph features in $\mathbf{g} = \{g_1, g_2, \cdots, g_m\}$. Because each community represents a group of subgraphs sharing high correlations, we can select g_q from these communities to minimize the redundancy of the feature set. Accordingly, our strategy for redundant feature deletion follows two steps:

a) **Subgraph Community Discovery:** Find dense and highly correlated subgraph communities.
b) **Redundant Subgraph Deletion:** Locate the most correlated subgraph pair $< g_i, g_j >$ in the most dense community A, and remove the least informative subgraph from the pair $< g_i, g_j >$.

Subgraph Community Discovery: To discover subgraph

community from $\mathbf{g}$, we construct a correlation graph, with each node denoting a subgraph g_i, and each edge between g_i and g_j indicating their Pearson's correlation $\phi(g_i, g_j)$ (defined in Eq. (1)). A correlation graph example is in shown in Fig. 3.

To form communities in the correlation graph, we further construct a k_{nn} *correlation graph matrix* $\mathcal{Q} = [\mathcal{Q}_{ij}]^{m\times m}$. More specifically, $\mathcal{Q}_{ij} = 1$ iff subgraph g_j is among one of the k_{nn}-nearest-neighbors (k_{nn} largest correlated subgraphs) of subgraph g_i or g_i is among one of the k_{nn}-nearest-neighbors of g_j; otherwise $\mathcal{Q}_{ij} = 0$. After that, we employ Clique based community discovery as follows.

DEFINITION 7 **Clique and Maximum Clique.** A graph (subgraph) $G = (\mathcal{V}, E, \mathcal{L})$ is a clique if for $\forall v_i, v_j \in \mathcal{V}$, $< v_i, v_j > \in E$. A clique is a maximum clique if it is not a subgraph of any other clique.

In the example showing in Fig. 3, group A denotes a maximum clique with size 3 (*i.e.* a $3-$clique).

Our community discovery algorithm is based on the maximum clique finding in the correlation graph matrix. More specifically, we first discover the maximum γ-cliques in the correlation graph (we use Bron-Kerbosch algorithm [14] in our experiments), and then overlap these maximum cliques to form communities. For instance, in Fig. 3, the community B is jointed by two 3-cliques $g_1 - g_2 - g_3$ and $g_3 - g_1 - g_4$. Using 3-cliques, we can find three communities in Fig. 3.

After discovering the subgraph communities, we need to select the most dense and correlated community.

DEFINITION 8 **Community Redundancy:** Given a community $A = (\mathcal{V}_A, E_A, \mathcal{L}_A)$, where each vertices in $\mathcal{V}_A$ is a graph feature, the Community redundancy (CR) for A is defined as:

$$CR(A) = \frac{1}{|E_A|} \sum_{\mathcal{Q}_{ij}=1} \phi(g_i, g_j) \tag{7}$$

The community redundancy (CR) assesses the average correlations between linked community members. The higher the CR value, the more redundancy exists in the community. Given a feature set $\mathbf{g} = \{g_1, g_2, \cdots, g_m\}$, its redundancy ($\mathcal{R}(\mathbf{g})$) is the maximum CR value among all communities $\mathcal{C}_{om}$ discovered from $\mathbf{g}$,

$$\mathcal{R}(\mathbf{g}) = \max_{A\in\mathcal{C}_{om}} CR(A) \tag{8}$$

To minimize the redundancy of $\mathbf{g}$ as defined in Eq.(8), we need to delete the subgraph feature with the largest correlation and small informativeness in the most redundant community.

Redundant Subgraph Deletion: After locate the most redundant community, the feature pair (g_i and g_j) with the highest correlation value is identified as the most correlated feature pair in the the community. The redundant subgraph feature is the one (g_i or g_j) which has a smaller gScore value. Then the least informative redundant feature is removed from $\mathbf{g}$. In example showing in Fig. 3, community C is the most redundant community (with highest CR value). g_5 and g_6 are the most correlated feature pair in community C. Because $\mathbb{i}(g_6) < \mathbb{i}(g_5)$, g_6 is identified as the least informative redundant feature to be removed from $\mathbf{g}$.

Algorithm 2 $Subgraph\_Redundancy\_check(\mathbf{g})$

Require:
$\mathbf{g} = \{g_1, g_2, \cdots, g_m\}$: A set of subgraph features;
k_{nn}: Number of neighbors;
γ: The minimum clique size;
Ensure:
g_q: A highly correlated and less informative feature in $\mathbf{g}$;
1: $G \leftarrow$ Construct a correlation graph with nodes as features (i.e., $g_i \in \mathbf{g}$), and edges as Pearson's correlation coefficients between nodes;
2: $\mathcal{Q} \leftarrow$ Construct an k_{nn} graph matrix;
3: $\mathcal{C}_{liques} \leftarrow$ find all maximum cliques from $\mathcal{Q}$;
4: $\mathcal{C}_{om} \leftarrow$ build communities by jointing $\mathcal{C}_{liques}$ with minimum size γ;
5: **if** $\mathcal{C}_{om} \neq \emptyset$ **then**
6: $\quad Z = \arg\max_{A\in\mathcal{C}_{com}} CR(A)$;
7: $\quad < g_i, g_j > = \arg\max_{<g_p,g_q>\in Z} \phi(g_p, g_q)$;
8: $\quad g_q = \arg\min(\mathbb{i}(g_i), \mathbb{i}(g_j))$;
9: **else**
10: $\quad g_q = \arg\min_{g_x\in\mathbf{g}} \mathbb{i}(g_x)$;
11: **end if**
12: **return** g_q;

Algorithm 2 lists detailed procedures of Subgraph_Redundancy_Check() for finding the most correlated and less informative feature in a set of features $\mathbf{g} = \{g_1, g_2, \cdots, g_m\}$. It is worth noting that if the community discovery fails to find any community from the feature graph $\mathcal{Q}$ (*i.e.* $\mathcal{C}_{om} = \emptyset$), it means that subgraph features have low correlations with each other. The algorithm will return the subgraph with the smallest gScore value as the least informative most redundant subgraph feature (line 10).

IV. GSLU ALGORITHM

In graph stream settings, graph data may constantly evolve (*i.e.*, concept drifting), which makes existing models incapable of classifying instances representing emerging/changing concepts. Accordingly, we define concept drifting samples by using models trained from historical data as follows,

DEFINITION 8 **Concept Drifting Samples.** Given a classifier trained from historical graph data, concept drifting samples (or instances) are the ones which are incorrectly classified by the given classifier.

Because concept drifting samples are "difficult" instances and an existing model is incapable of classifying them, they need to be emphasized during the stream learning process. In our system, we use an instance based weighting method to capture difficult graph samples, and combine the instance weight with the feature selection module.

Instance Weighting: The idea of our weighting scheme is as follows: as soon as a new graph chunk $D_t = D_t^l \bigcup D_t^u$ is collected for processing, we use an ensemble of classifiers $E = \{\mathcal{H}_{t-k}, \mathcal{H}_{t-k+1}, \cdots, \mathcal{H}_{t-1}\}$ which is trained from historical chunks to predict labeled graphs in D_t^l. If a graph is miss-classified by E, we increase the graph's weight because it is a difficult sample, and if a graph is correctly classified we decrease its weight. This weighting mechanism is also similar to the Adaboost algorithm [15]. By doing

so, we can integrate the instance weight to the succeeding subgraph feature selection procedure, so gSLU can emphasize on difficult samples and select a set of informative features to represent concept drifting samples.

gSLU Algorithm: Algorithm 3 lists detailed procedures of the proposed gSLU framework which combines instance weighting and minimum redundancy subgraph feature selection for graph stream classification.

The "while" loop in Algorithm 3 represents a stream processing cycle which repeats as long as new graph data continuously arrive. Once a new chunk D_t (including labeled D_t^l and unlabeled graphs D_t^u) is collected, gSLU first checks whether D_t is the first chunk (line 5). For the first chunk D_1, gSLU simply retrieve a set of features $\mathbf{g}$ from D_1 by using Algorithm 1, and add the classifier $\mathcal{H}_t$ built from $\mathbf{g}$ to initialize the ensemble E (lines 5-9).

For any succeeding chunks (except the first chunk) $D_t, t = 2, 3, \cdots$, gSLU uses the ensemble E to tune the weight of each graph in D_t by using ensemble error rate ξ (line 10), where $h(G_i|E)$ returns the class label of G_i predicted by the ensemble E. If the ensemble E misclassifies a graph $G_i \in D_t^l$, gSLU increases the weight of G_i by $\ln \frac{1-\xi}{\xi}$, otherwise gSLU decreases the weight of G_i by $\ln \frac{\xi}{1-\xi}$ (line 11). On line 12, gSLU normalizes the weight values for all instances in D_t, followed by line 13 which retrieves a set of minimum redundancy subgraph features $\mathbf{g}$ from D_t. By using features in $\mathbf{g}$, a classifier $\mathcal{H}_t$ is trained and is used to update the ensemble E (lines 14-15).

At the testing phase, gSLU uses the majority vote of all classifiers in E to predict graphs in the new graph chunk D_{t+1}.

Algorithm 3 gSLU

Require:
$\mathcal{S} = \{D_1, D_2, \cdots\}$: Graph Stream
k: The maximum capacity of the ensemble

1: Initialize $E = \emptyset, t = 0$;
2: **while** $\mathcal{S}! = \emptyset$ **do**
// Training Phase:
3: $D_t \leftarrow$ A new graph chunk;
4: $D_t = D_t^l \bigcup D_t^u$; $\mathcal{S} \leftarrow \mathcal{S}/D_t$; $t = t + 1$;
5: **if** $(t == 1)$ **then**
6: $\mathbf{g} \leftarrow$ minimum redundancy features in D_t; //Algorithm 1
7: $H_t \leftarrow$ classifier built from D_t and $\mathbf{g}$;
8: $E \leftarrow E \bigcup \mathcal{H}_t$
9: **else**
10: $\xi \leftarrow \frac{\sum_{G_i \in D_t^l} (h(G_i|E)! = y_i)}{|D_t^l|}$;
11: $w_i = \begin{cases} w_i \ln \frac{1-\xi}{\xi} & : \quad h(G_i|E)! = y_i, G_i \in D_t^l \\ w_i \ln \frac{\xi}{1-\xi} & : \quad h(G_i|E) = y_i, G_i \in D_t^l \end{cases}$;
12: $w_i = \frac{w_i}{\sum_{G_i \in D_t^l \bigcup D_t^u} w_i}$;
13: $\mathbf{g} \leftarrow$ minimum redundancy features in D_t; //Algorithm 1
14: $\mathcal{H}_t \leftarrow$ classifier built from D_t and $\mathbf{g}$;
15: $E \leftarrow E \bigcup \mathcal{H}_t$
16: **if** $|E| > k$ **then**
17: $E \leftarrow E/\mathcal{H}_{t-k}$
18: **end if**
19: **end if**
// Testing Phase:
20: $D_{t+1} \leftarrow$ A new graph chunk;
21: $h(G_i|E) = \arg\max \sum_{i=t-k-1}^{t} h(G_i|\mathcal{H}_i)$
22: **end while**;

V. Experiments

We report our experiments on real-world graph data, with emphasis on (1) effectiveness of the proposed minimum redundancy subgraph feature selection module, and (2) the efficiency and effectiveness of gSLU for graph stream classification.

A. Experimental Settings

Two graph streams collected from real-world applications are used in our experiments.

DBLP Graph Stream. The DBLP dataset (http://arnetminer.org/citation) consists of bibliography data in computer science. Each record in DBLP is associated with a number of attributes such as abstract, authors, year, venue, title, and reference ID [16]. To build a graph stream, we select a list of conferences (as shown in Table I) and use the papers published in these conferences (in chronological order) to form a binary-class graph stream. The classification task is to predict whether a paper belongs to DBDM (database and data mining) or CVPR (computer vision and pattern recognition) field, by using the references and the title of each paper. Notice that DBDM and CVPR are overlapping in many aspects, such as machine learning and visual information retrieval. The shifting of the research focus makes DBLP stream an ideal test ground for concept drifting graph stream classification. For example, there are an increasing number of papers to address social network research problems for both DBDM and CVPR fields (*i.e.*, community discovery for DBDM and social tagging in CVPR), which naturally introduces concept drifting in the stream.

In our experiments, each paper in DBLP is represented as a graph, where each node denotes a Paper ID or a keyword and each edge denotes the citation relationship between papers or keyword relations in the title. More specifically, we denote that (1) each paper ID is a node; (2) if a paper P.A cites another paper P.B, there is an edge between P.A and P.B; (3) each keyword in the title is also a node; (4) each paper ID node is connected to the keyword nodes of the paper; and (5) for each paper, its keyword nodes are fully connected with each other. An example of DBLP graph data is shown in Fig. 4.

The original DBLP dataset contains a significant number of papers without references. In our experiments, we remove those papers, and choose 1000 most frequent words appearing in the title (after removing the stop words) as keywords to construct graphs. The last column in Table I shows the number of graphs in each category in our experiments.

NCI Chemical Compound Graphs. The NCI cancer screening datasets are commonly used as the benchmark for graph classification. We download nine NCI data sets from PubChem (http://pubchem.ncbi.nlm.nih.gov). Each NCI data set belongs to a bioassay task for anticancer activity prediction, where each chemical compound is represented as a graph, with atoms representing nodes and bonds as edges. A chemical compound is positive if it is active against the corresponding cancer, or

TABLE I

DBLP DATASET USED IN EXPERIMENTS

Categories	Descriptions	#Paper	#Graphs
DBDM	SIGMOD,VLDB,ICDE, EDBT,PODS, DASFAA,SSDBM,CIKM,DEXA KDD, ICDM, SDM, PKDD, PAKDD	20601	9530
CVPR	ICCV, CVPR, ECCV, ICPR, ICIP ACM Multimedia, ICME	18366	9926

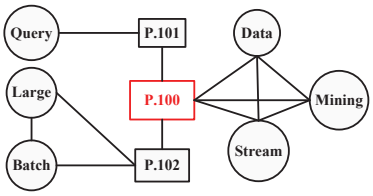

Fig. 4. Graph representation for a paper (P.100) in the DBLP. The rectangles are paper ID nodes and circles are keyword nodes. The paper P.100 cites (connects) paper P.101 and P.102, and P.100 has keywords *Data*, *Stream*, and *Mining* in its title. Paper P.101 has keyword *Query* in its title, and P.102's title include keywords *Large* and *Batch*. For each paper, the keywords in the title are linked with each other.

negative otherwise.

Table II summarizes the NCI graph data used in our experiments, where columns 2-3 show the number of positive graphs and the total number of graphs in the original datasets. After removing disconnected graphs and graphs with unexpected atoms (some graphs have atoms represented as '*'), we obtain new datasets with slightly different sizes, as shown in columns 4-5. Meanwhile, each NCI dataset is highly imbalanced, with about 5% positive graphs, so we randomly select the same number of negative graphs as the positive graphs for each dataset (*e.g.*, 1793 positive and 1793 negative graphs in NCI1). After that, we concatenate nine datasets into a stream with 33,028 graphs in total. In the NCI graph stream, the bioassay task changes from time to time (from one dataset to another), which simulates the concept drifting in the graph stream.

Baseline Methods To evaluate the effectiveness of our minimum redundancy feature selection and instance weighting based graph stream classification framework, we compare the proposed gSLU with both supervised and semi-supervised feature selection methods (all of them are based on the same ensemble framework as gSLU).

- **Information gain based method (IG+Stream).** In each chunk, we first retrieve a set of frequent subgraph features from labeled graphs, and rank the features according to their Information Gain (IG) value. The top-m subgraphs with the highest IG are used to build the classifier. This framework only uses labeled graph in the stream and does not consider feature redundancy at all.
- **gSemi based method (gSemi+Stream).** In each chunk, we employ gSemi algorithm [12] to select a set of informative features to train a classifier. This framework combines labeled and unlabeled graphs to maximize the total informativeness score for a set of features, without considering feature redundancy.

For fair comparisons, all three algorithms (gSLU, IG+Stream, and gSemi+Stream) use Nearest Neighbor (NN) classifier (trained from features extracted from each chunk by using different methods) to form an ensemble and predict graphs in a future chunk. For IG+Stream and gSemi+Stream, there is no instance weighting, and only gSLU updates the graph sample weight over stream. The ensemble method is based on a majority voting approach.

Unless specified otherwise, the default parameter settings are as follows: Ensemble size k=15, chunk size $|D_t| = 800$, number of neighbors to build correlated graphs k_{nn}=5, maximum cliques $\gamma = 4$ (for DBLP) and 6 (for NCI), minimum support threshold $min\_supp = 1\%$ (for DBLP) and 30% (for NCI) of the chunk size $|D_t|$.

All experiments are collected from a linux cluster computing node with an Interl(R) Xeon(R) @3.33GHZ CPU and 3GB fixed memory size.

TABLE II

NCI CANCER SCREEN DATASETS USED IN THE EXPERIMENTS

Bioassay-ID	Original Compounds #Pos	Original Compounds #Total	New Compounds #Pos	New Compounds #Total
NCI1	2295	42324	1793	37349
NCI33	1857	41971	1467	37022
NCI41	1733	28596	1350	25336
NCI47	2298	42333	1735	37298
NCI81	2686	42565	2081	37549
NCI83	2487	28868	1959	25550
NCI109	2309	42540	1773	37518
NCI123	3461	41732	2715	36903
NCI145	2178	42016	1641	37041

B. Experimental Results

In this section, we first compare the feature selection component of each algorithm, without considering the ensemble framework and graph weighting. Then we integrate ensemble into each algorithm to compare the overall performance of different methods for graph stream classification.

1) Minimum Redundancy Subgraph Feature Selection Results: To report our minimum redundancy subgraph feature selection results, for each stream, we built a classifier $\mathcal{H}_t$ from the current chunk D_t, by using different feature selection methods, to predict graphs in the next chunk D_{t+1}. There is no instance weighting and ensemble framework involved in the experiment. Then we report the average classification accuracy of different methods over the whole stream in Fig. 5.

The results in Fig. 5 demonstrate that our subgraph feature selection module in gSLU outperforms its peers on both DBLP and NCI streams. Among all three methods, IG only uses labeled graphs to generate feature candidates, and its results are inferior to other methods which confirm that using unlabeled graphs can tackle the data sparsity issues and help generate useful feature patterns for graph classification. Meanwhile, although both gSLU and gSemi combine labeled and unlabeled graphs for feature selection, our results show that combining each feature's informativeness score and the

redundancy of the feature set, like gSLU does, is superior to gSemi which only assesses each feature's informativeness score without considering their correlations.

In the following subsections, we report results that incorporate subgraph feature selection, graph weighting, and ensemble for graph stream classification.

2) Graph Streams Classification Accuracies: **Results on different sizes of labeled graphs $|D_t^l|$:** In Figs. 6 and 7, we vary the number of labeled graphs in each graph chunk, and report the results on DBLP and NCI streams.

The results in Figs. 6 and 7 show that gSLU consistently outperforms IG+stream and gSemi+Stream across the whole stream. Without utilizing unlabeled data, IG+Stream's performance is significantly worse than gSLU and gSemi+Stream, especially in Fig. 6.(C). This is because that labeled graphs in each chunk are very limited. By using rich unlabeled graphs, it is possible to discover a set of dense and informative subgraph patterns that better differentiate labeled graphs. Meanwhile, gSLU outperforms gSemi+Stream in data stream classification. This is mainly attributed to gSLU's two key components, including minimum redundancy subgraph feature selection and instance weighting. The former selects a set of highly informative and low redundancy features to build classifiers, and the latter allows multiple chunks (classifiers) to work in a collaborative way to form an accurate ensemble model. As a result, gSLU achieves good performance in classifying graph streams with dynamic changes. For example, in Fig. 6.(B), there are noticeable concept drifting from chunks 10-12 and from 18-20 (marked by the rectangle boxes). As a result, all three methods experience performance loss. By employing instance weighting to tackle the concept drifting, gSLU receives much less loss than gSemi+Stream and IG+Stream.

The average accuracies over the whole graph stream, as shown in Fig. 8, demonstrate that increasing the size of labeled graphs in each chunk can benefit all three methods. This result indicates that a large number of labeled graphs in each chunk can enhance the feature selection in a semi-supervised graph stream setting. Overall, gSLU is the best among the three methods.

In Table III, we report the pairwise t-test (with confident level $\alpha = 0.05$) to validate the statistical significance between three methods. Each entry (value) denotes the p-value for a t-test between two algorithms, and a p-value less than $\alpha = 0.05$ indicating that the difference is statistically significant. From Table III, gSLU statistically outperforms IG+Stream in all cases, and is superior to gSemi+Stream for eight out of nine trials.

Results on different number of features m: In Figs. 9 and 10, we report the algorithm performance with respect to different number of subgraph features in each chunk. The average results over the whole stream are also reported in Fig. 11. As expected, gSLU has the best performance among the three algorithms for both DBLP (Fig. 9) and NCI (Fig. 10) streams. In Fig. 9.(A), there is a significant concept drifting from chunks 2-3, where gSemi+Stream and IG+Stream both experience sharp accuracy drop. In contrast, gSLU's performance loss is much smaller than other two methods. This also demonstrates the effectiveness of the instance weighting scheme for capturing emerging concept drifting samples. By adjusting graph weight values, gSLU can immediately select subgraph features to represent emerging samples and force the whole classification framework to adapt to the changing concepts in the graph stream.

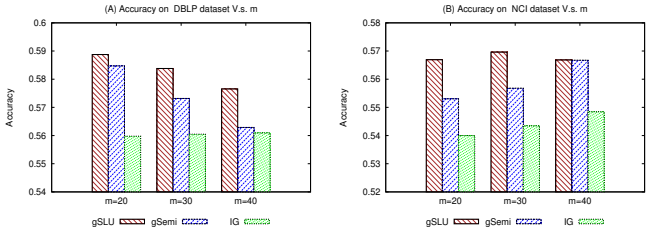

Fig. 5. Comparison of our minimum redundancy subgraph feature selection with other algorithms. For each graph stream and a chunk D_t, we build a classifier $\mathcal{H}_t$ from chunk D_t, by using features selected from different methods, to predict graphs in D_{t+1}. The results represent the average classification accuracy over all chunks in the graph stream. (A) Results on DBLP stream, with chunk size $|D_t| = 1000$, labeled graphs in each chunk $|D_t^l|$=30; (B) Results on NCI stream, with $|D_t| = 800$, $|D_t^l|$=30.

TABLE III

PAIRWISE T-TEST RESULTS WITH LABELED GRAPH SIZES $|D_t^l|$. A, B, AND C DENOTE GSLU, GSEMI+STREAM, AND IG+STREAM, RESPECTIVELY.

DBLP				NCI			
D_t^l	A-B	A-C	B-C	D_t^l	A-B	A-C	B-C
20	1.7E-03	3.0E-07	9.0E-04	30	3.1E-09	1.4E-11	1.3E-02
30	2.4E-02	7.0E-06	3.7E-09	50	5.7E-07	4.2E-11	1.2E-06
70	**8.6E-02**	1.7E-12	3.2E-10	70	4.2E-09	5.3E-19	1.1E-10

Interestingly, results in Fig. 11 show that increasing the number of features in DBLP stream actually reduces the accuracy of all three algorithms. This may suggest that for DBLP classification task, a small number of features (such as keywords or simple citation relationships) may have enough discriminate power for classification. Increasing subgraph features may introduce redundant features into the learning process, which deteriorates the accuracy of the classifier. This is, however, not the case for NCI stream where increasing the number of features consistently help improve the classifier.

In Table IV, we report the pairwise t-test with confident level $\alpha = 0.05$. The p-values (less than 0.05) in each entry assert that gSLU statistically significantly outperforms gSemi+Stream and IG+Stream.

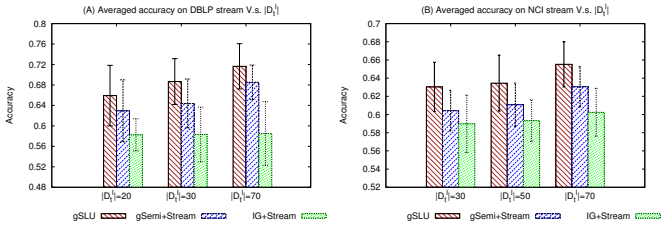

Fig. 8. Average accuracy (and standard deviation) *v.s.* labeled graph sizes $|D_t^l|$ with chunk size $|D_t|$=800, feature size $m = 20$.

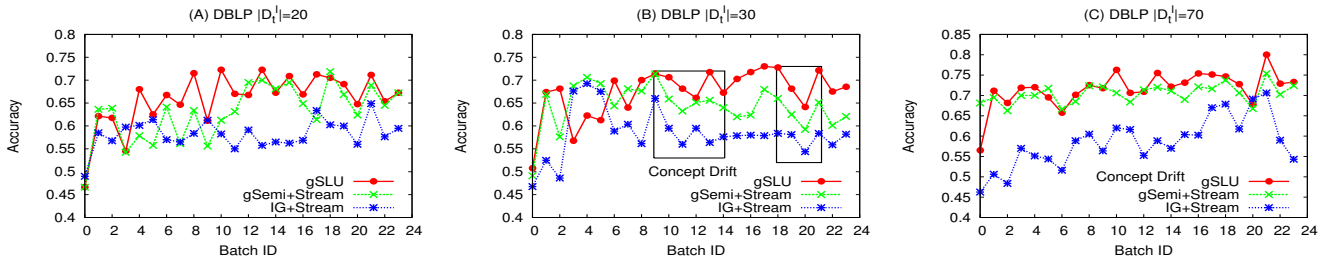

Fig. 6. Accuracy *w.r.t.* different sizes of labeled graphs on **DBLP stream** with each chunk containing 800 graphs, and the number of features in each chunk is 20. The number of labeled graphs in each chunk: (A) 20; (B) 30; (C) 70.

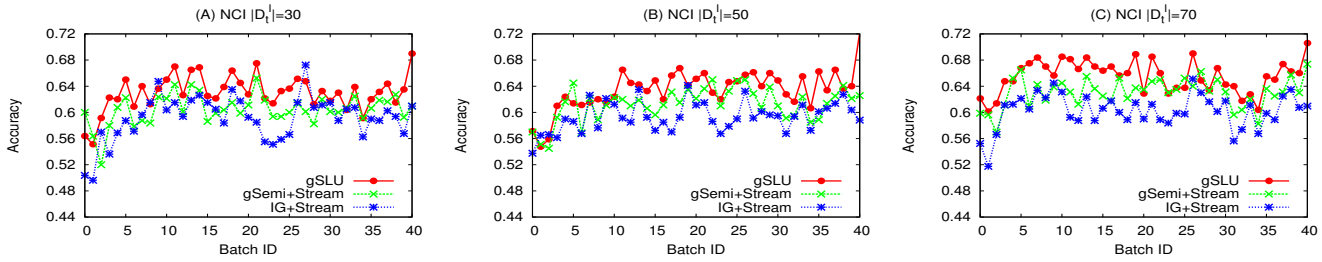

Fig. 7. Accuracy *w.r.t.* different seizes of labeled graphs on **NCI stream** with each chunk containing 800 graphs, and the number of features in each chunk is 20. The number of labeled graphs in each chunk: (A) 30; (B) 50; (C) 70.

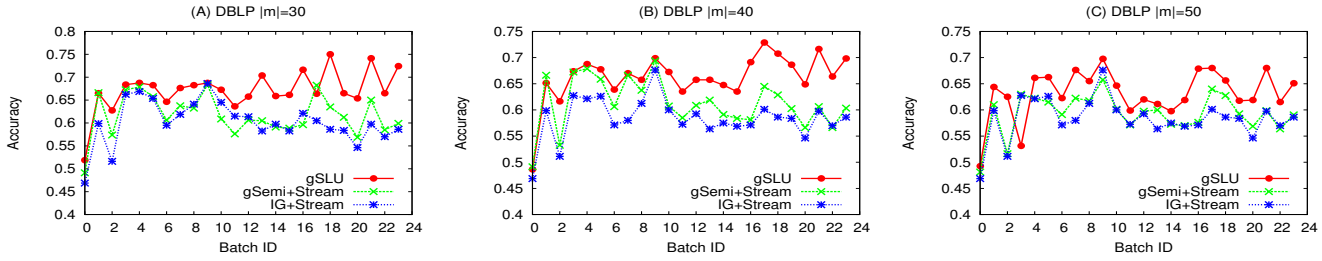

Fig. 9. Accuracy *w.r.t.* different number of features on **DBLP stream** with each chunk containing 800 graphs, and the size of labeled data in each chunk is 30. The number of feature selected in each chunk: (A) 30; (B) 40; (C) 50.

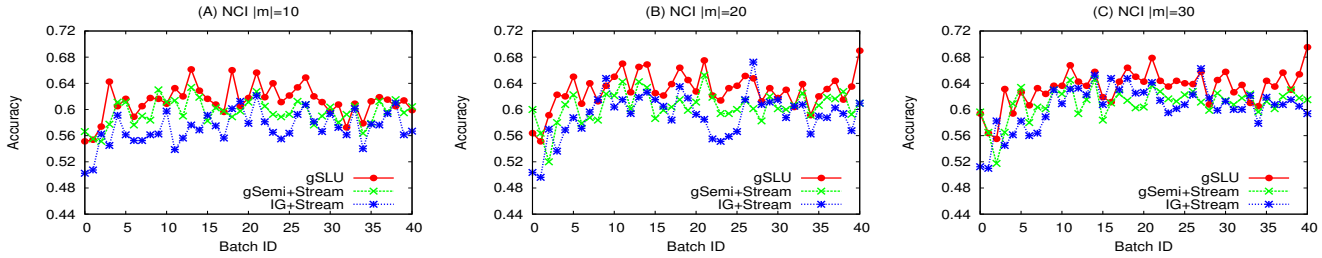

Fig. 10. Accuracy *w.r.t.* different number of features on **NCI stream** with each chunk containing 800 graphs, and the size of labeled data in each chunk is 30. The number of feature selected in each chunk: (A) 10; (B) 20; (C) 30.

Results on different chunk sizes $|D_t|$: In Fig. 12, we report the algorithm performance by using different number of graphs in each chunk $|D_t|$ (varying from 1000, 800, to 600).

Overall, the results in Fig. 12 show that the accuracies of all methods fluctuate to a large extent across the whole stream. Because we fixed the number of labeled graphs in each chunk to 30, the overall trend shows that a smaller chunk size may slightly outperform the settings with a larger chunk size. We omit the average results and t-test results due to lack of space.

3) Graph Streams Classification Efficiency: In Figs. 13 and 14, we report the system runtime performance (efficiency) for graph stream processing. The results show that IG+Stream algorithm takes much less time than semi-supervised algorithms (gSLU and gSemi+Stream). This is mainly because IG+Stream carries out frequent subgraph mining on a small set of labeled graphs whereas both gSLU and gSemi+Stream need to handle labeled and unlabeled data. In our experimental settings, the labeled graphs is less than 10% of the unlabeled graphs, so IG+Stream shows much better runtime performance. When

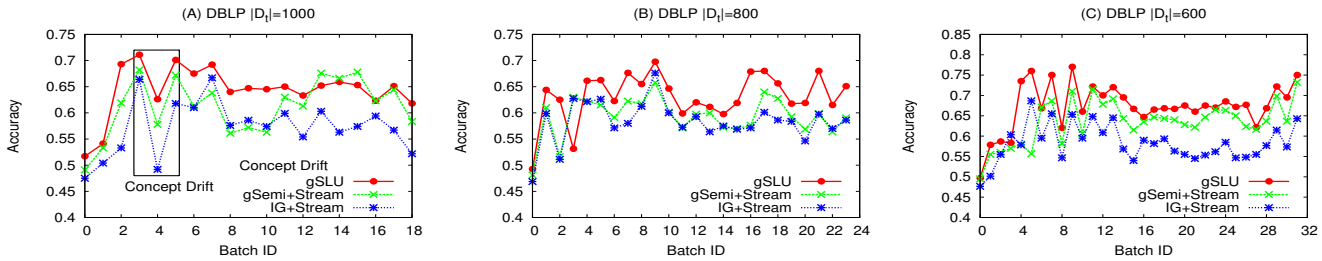

Fig. 12. Accuracy *w.r.t.* different chunk sizes on **DBLP stream** with each chunk containing 30 labeled graphs, and the number of features in each chunk is 50. The batch sizes vary as: (A) 1000; (B) 800; (C) 600.

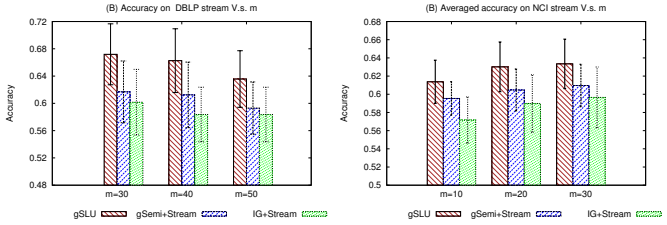

Fig. 11. Averaged accuracy (and standard deviation) *v.s.* number of features m, with chunk size $|D_t|$=800, feature size $m = 20$.

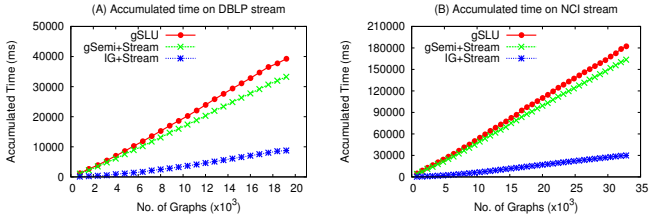

Fig. 13. System accumulated runtime *v.s.* number of graphs processed over stream. ($|D_t| = 800$, $|D_t^l| = 10\%|D_t|$, and m=20); (A) Results on DBLP stream; (B) Results on NCI stream.

TABLE IV

PAIRWISE T-TEST RESULTS WITH VARIOUS FEATURE SIZE m. A, B, AND C DENOTE GSLU, GSEMI+STREAM, AND IG+STREAM, RESPECTIVELY.

	DBLP				NCI		
m	A-B	A-C	B-C	m	A-B	A-C	B-C
30	5.4E-07	6.1E-08	2.1E-02	10	7.4E-06	1.7E-13	1.7E-08
40	1.9E-06	1.2E-11	1.1E-06	20	3.1E-09	1.4E-11	1.3E-02
50	5.3E-05	6.9E-06	9.7E-03	30	4.9E-09	5.2E-08	**2.7E-01**

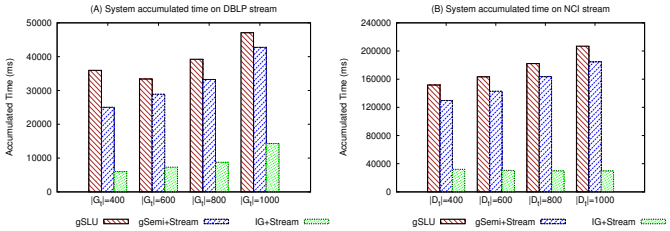

Fig. 14. System accumulated runtime *v.s.* different chunk sizes $|D_t|$, $|D_t^l| = 10\%|D_t|$. m=20; (A) Results on DBLP stream; (B) Results on NCI stream.

comparing gSLU and gSemi+Stream, gSLU requires slightly more time than gSemi+Stream because of the minimum-redundance checking during the feature selection process. Meanwhile, the accumulated system runtime *w.r.t.* different chunk sizes, as shown in Fig. 14, also indicate that system runtime remains relatively stable for various chunk sizes. Overall, the results show that gSLU linearly scales to the number of graphs and chunks, which means that gSLU is capable of handling real-world high speed graph streams.

In the experiments, we also tried to collect system runtime by treating each stream as a one single graph chunk. Unfortunately, both streams ended up with insufficient memory for processing (using a 3GB Memory cluster computing node). This also asserts that a stream based processing framework is potentially useful for handling large scale graph datasets.

VI. RELATED WORK

The proposed work is closely related to graph classification, subgraph feature selection, and graph stream mining.

A typical graph classification problem concerns about assigning a proper class label y to a previously unseen test graph, given a set of labeled training graphs $\{G_i, y_i\}; i = 1, \cdots, n; y_i \in \mathcal{Y}$. In this paper, we mainly focus on multiple graphs each associated with one single label. Because graph data involve node-edge structures whereas most existing classification methods use instance-feature representation model, the major challenge of graph classification is to transfer graph data into proper format for learning methods to train classifiers. Existing methods in the area mainly fall into two categories (1) distance based methods (including graph kernel [17], [18], graph embedding [19], and transformation [20]), and (2) subgraph feature based methods. For distance based methods, graph kernels, graph embedding, or transformation are used to calculate the distance between a pair of graphs. The distance matrix can be fed into a learning algorithm, such as k-NN, for graph classification. The major disadvantage of the distance based method is that characterizing distance between graphs is computationally expensive and practically unreliable. This is because graph isomorphism is known to be NP-complete and majority methods will rely on extensive, if not exhaustive, path traversal (or random walk) to capture graph distance. For graphs with different number of nodes and edges, the distance calculated using path traversal might be not accurate. In addition, graphs belonging to the same class may not be globally similar but share some unique subgraph structures. This is especially true for chemical compound [4]

and molecular biology data [18]. Some empirical studies [6] have shown that with limited experiments, graph kernels based classification methods are inferior to other approaches.

For subgraph feature based methods, the major goal is to identify significant subgraphs which can be used as signature for one particular class. In [21], the authors propose a structure feature selection method to consider subgraph structural information for classification. Jin [22] proposes to extract subgraph patterns and use their co-occurrence to build classifier for graph classification. Other method [23] regards subgraph selection as combinatorial optimization problem and uses heuristic rules, in combination with frequent subgraph mining algorithm such as gSpan [11], to find subgraph features. In [12], the authors propose a semi-supervised subgraph feature selection method which uses unlabeled graphs to boost subgraph feature selection for classification. Some graph Boosting methods [7], [8], [5] also exist to use each single subgraph feature as a weak classifier to build Boosting algorithm for graph classification. For all existing subgraph feature based methods, regardless of whether the labeled and unlabeled graphs are considered or not, they do not consider the redundancy between selected subgraph features, and they assume that the underlying graph set is static and there is no treatment to handle dynamic graph streams.

For graph stream mining, the underlying challenge is twofold (1) the data volumes constantly increase, and (2) the data distributions or the decision concepts to differentiate different types of graph samples are constantly changing. Several methods have been proposed for outlier detection [24] and query estimation [25] for graph streams, event detection from Twitter streams [3], and graph stream classifications [2]. For graph stream classification, the only method currently available is a random hashing based framework [2], which uses hashing function to transfer graph edges into an integer and uses discriminative edges as patterns for classification. In comparison, gSLU is the first graph stream mining framework utilizing both labeled and unlabeled graphs. It provides opportunity to tackle the scarcity of labeled data in data stream. The ensemble framework of gSLU is also reliable and efficient for handling concept drifting data streams [26], [27].

VII. Conclusion

In this paper, we investigated graph stream classification using limited labeled graphs and abundant of unlabeled graphs. We argued that existing methods for subgraph feature selection fail to consider redundancy between selected subgraphs, and dynamic graph streams require solutions to capture emerging concept drifting examples so the overall framework can adapt to the changes in the streams. In the paper, we proposed unique measures to discover informative subgraph features with minimum redundancy, through which we can build classifiers from graph streams. To capture emerging difficult graphs in the stream, we proposed an instance weighting mechanism to force the subgraph feature selection module to emphasize on emerging concept drifting graphs, so our model can quickly adapt to the changes in the stream. Experiments validated the proposed design for effective graph stream classification.

Acknowledgment

This research is sponsored by Australian Research Council (ARC) Future Fellowship under Grant No. FT100100971.

References

[1] S. Raghavan and H. Garcia-Molina, "Representing web graphs," in *Proc. of ICDE*, Atlanta, USA, 2003.
[2] C. Aggarwal, "On classification of graph streams," in *Proc. of SDM*, Arizona, USA, 2011.
[3] S. Petrovic, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to twitter," in *Proc. of Human Language Technologies*, Stroudsburg, USA, 2010.
[4] M. Deshpande, M. Kuramochi, N. Wale, and G. Karypis, "Frequent substructure-based approaches for classifying chemical compounds," *IEEE Trans. on Knowl. and Data Eng.*, vol. 17, pp. 1036–1050, 2005.
[5] H. Fei and J. Huan, "Boosting with structure information in the functional space: an application to graph classification," in *Proc. of ACM SIGKDD*, Washington DC, USA, 2010.
[6] N. Ketkar, L. Holder, and D. Cook, "Empirical comparison of graph classification algorithms," in *Proc. of IEEE CIDM*, Nashville, 2009.
[7] T. Kudo, E. Maeda, and Y. Matsumoto, "An application of boosting to graph classification," in *Proc. of NIPS*, Vancouver, Canada, 2004.
[8] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda, "gboost: a mathematical programming approach to graph classification and regression," *Machine Learning*, vol. 75, pp. 69–89, 2009.
[9] W. Zhou and H. Xiong, "Volatile correlation computation: a checkpoint view," in *Proc. of ACM SIGKDD*, 2008, pp. 848–856.
[10] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge," in *Proc. of ICML*, 2001, pp. 577–584.
[11] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *Proc. of ICDM*, Maebashi City, Japan, 2002.
[12] X. Kong and P. Yu, "Semi-supervised feature selection for graph classification," in *Proc. of ACM SIGKDD*, Washington, DC, USA, 2010.
[13] M. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical review E*, vol. 69, no. 2, p. 026113, 2004.
[14] C. Bron and J. Kerbosch, "Algorithm 457: finding all cliques of an undirected graph," *C. of the ACM*, vol. 16, no. 9, pp. 575–577, 1973.
[15] Y. Freund and R. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory*. Springer, 1995, pp. 23–37.
[16] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su, "Arnetminer: Extraction and mining of academic social networks," in *Proceeding of ACM SIGKDD*, 2008, pp. 990–998.
[17] P. Mahe, N. Ueda, T. Akutsu, J. Pettet, and J. Vert, "Externsions of marginalized graph kernels," in *Proc. of ICML*, Alberta, Canada, 2004.
[18] H. Kashima, K. Tsuda, and A. Inokuchi, *Kernels for Graphs*. Cambridge (Massachusetts): MIT Press, 2004, ch. In: Schlkopf B, Tsuda K, Vert JP, editors. Kernel methods in computational biology.
[19] K. Riesen and H. Bunke, "Graph classification by means of lipschitz embedding," *IEEE Trans. on SMC - B*, vol. 39, pp. 1472–1483, 2009.
[20] ——, *Graph Classification and Clustering Based on Vector Space Embedding*. World Scientific Publishing Company, 2010.
[21] H. Fei and J. Huan, "Structure feature selection for graph classification," in *Proc. of ACM CIKM*, California, USA, 2008.
[22] N. Jin, C. Young, and W. Wang, "Graph classification based on pattern co-occurrence," in *Proc. of ACM CIKM*, Hong Kong, China, 2009.
[23] M. Thoma, H. Cheng, A. Gretton, J. Han, H. Kriegel, A. Smola, L. Song, P. Yu, X. Yan, and K. Borgwardt, "Near-optimal supervised feature selection among frequent subgraphs," in *Proc. of SDM*, USA, 2009.
[24] C. Aggarwal, "Outlier detection in graph streams," in *Proc. of ICDE*, Hannover, Germany, 2011.
[25] P. Zhao, C. Aggarwal, and M. Wang, "gsketch: On query estimation in graph streams," in *Proc. of VLDB*, Istanbul, Turkey, 2012.
[26] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from stream data using optimal weight classifier ensemble," *IEEE Trans. on SMC - B*, vol. 40, pp. 1607–1621, 2010.
[27] H. Abdulsalam, D. Skillicorn, and P. Martin, "Classification using streaming random forests," *IEEE TKDE*, vol. 23, pp. 22–36, 2011.