Parallel Multi-graph Classification Using Extreme Learning Machine and MapReduce

Jun Pang, Yu Gu, Jia Xu, Xiaowang Kong and Ge Yu

Abstract A multi-graph is represented by a bag of graphs and modelled as a generalization of a multi-instance. Multi-graph classification is a supervised learning problem for multi-graph, which has a wide range of applications, such as scientific publication categorization, bio-pharmaceu-tical activity tests and online product recommendation. However, existing algorithms are limited to process small datasets due to high computation complexity of multi-graph classification. Specially, the precision is not high enough for a large dataset. In this paper, we propose a scalable and high-precision parallel algorithm to handle the multi-graph classification problem on massive datasets using MapReduce and extreme learning machine. Extensive experiments on real-world and synthetic graph datasets show that the proposed algorithm is effective and efficient.

Keywords Multi-graph · Classification · Extreme learning machine · MapReduce

J. Pang (✉) · Y. Gu · X. Kong · G. Yu
College of Information Science and Engineering, Northeastern University, Liaoning 110819, China
e-mail: pangjun@research.neu.edu.cn

Y. Gu
e-mail: guyu@ise.neu.edu.cn

X. Kong
e-mail: kongxiaowang.neu@gmail.com

G. Yu
e-mail: yugu@ise.neu.edu.cn

J. Xu
School of Compute, Electronics and Information, Guangxi University, Guangxi 530004, China
e-mail: xujia@gxu.edu.cn

J. Cao et al. (eds.), *Proceedings of ELM-2015 Volume 1*,
Proceedings in Adaptation, Learning and Optimization 6,
DOI 10.1007/978-3-319-28397-5_7

1 Introduction

Multi-graph learning is a generalization of multi-instance learning in which instances are organized as graphs instead of feature vectors. Multi-graph representation maintains rich structure information which makes it outperforms other multi-instance representation. Nowadays, multi-graph learning has many successful application scenarios. We give two typical examples as follows. (1) Scientific publication categorization [1]: a paper is represented as a multi-graph, i.e., the abstract of the paper is modelled as a graph and the abstract of every reference is also modelled as a graph. If the paper or one of its references is related with the topic, this paper is positive. Otherwise, it is negative. Given training papers with classification labels, we can predict the unseen papers' labels. (2) Bio-pharmaceutical activity tests [2]: a molecule has a lot of forms. If one of its forms resists the disease, the molecule be used to manufacture drugs. Otherwise, it cannot be applied. A specific form of a molecule can be described as a graph and a multi-graph demotes different forms of the molecule. Multi-graph learning can predict the molecules' activities.

Although multi-graph learning has important practical applications, existing multi-instance learning algorithms can not be directly used to solve this problem. Because these multi-instance learning algorithms are designed to process tabular instances which are represented in a common vectorial feature space. To the best of our knowledge, few work focuses on exploring it. Wu et al. propose the *gMGFL* approach, which mines informative feature subgraphs and has higher accuracy than the extended multi-instance learning algorithms. Nowadays, the quantity of information is very large and fast growing. And more and more multi-graphs are modelled from these increasing information. It is a non-trivial task to mine valued knowledge from so large scale multi-graphs. Specifically, the following challenges need to be tackled. (1) *gMGFL* is not suited to deal with large-scale datasets because *gMGFL* adopts in-memory frequent subgraph mining and classification algorithms. (2) To support high-quality exploratory analysis and decision making, the precision and recall are desired to be improved. Our experimental results show the traditional parallel Bayes algorithm does not obtain high precisions and recalls on large-scale multi-graph datasets.

For the first challenge, we adopt popular MapReduce framework to execute large-scale multi-graph binary classification. MapReduce is a popular parallel programming framework to handle big data owning to its good properties of fault tolerance, high scalability and low deployment cost [3]. We propose a parallel approach based on MapReduce, named *ME-MGC*, to solve multi-graph binary classification problem on massive multi-graph dataset. For the second challenge, we adapt a parallel ELM approach to improve the classification performance. Experimental results display that our approaches obtain higher precisions and recalls on both real and synthetic datasets than extended approaches adopting NBayes or SVM.

Specifically, the major contributions we have made in this paper are summarized as follows. (1) We propose a parallel approach based on MapReduce to solve the massive multi-graph binary classification problem. (2) We adapt extreme learning machine (ELM) to process multi-graph classification for improving the performance of classification. Moreover, we study the variation of the precision with a different number of hidden nodes for ELM algorithm. (3) We have conducted extensive experiments on both real and simulated data sets and the results demonstrate that our approaches are effective and efficient.

The remainder of this paper is organized as follows. Related works are introduced in Sect. 2. Problem definition and backgrounds are discussed in Sect. 3. Our approach is provided in Sect. 4. Experimental results and discussions are presented in Sect. 5. We conclude the paper in Sect. 6.

2 Related Works

Related works with our study include multi-graph classification, extreme learning machine based on MapReduce and frequent subgraph mining based on MapReduce.

2.1 Multi-graph Learning

Although multi-graph learning is very valuable in real applications, the researches on it are still quite limited. Wu et al. [1] propose *gMGFL* approach to solve multi-graph classification problem. Inspired by multi-instance learning, *gMGFL* mines feature subgraph from overall graph dataset and converts multi-graphs into feature-value vectors which fit conventional classification models, such as naive Bayes, *k*NN classifier, decision tree and support vector machine. Feature subgraphs are top-*k* frequent subgraphs with a score function $score(g)$. *gMGFL* is not suitable to process large-scale datasets because it is an in-memory algorithm.

2.2 Extreme Learning Machine Based on MapReduce

Extreme learning machine(ELM) is a type of artificial neural network [4, 5]. Parallel ELM based on MapReduce has attracted attention of many researchers. He et al. [6] propose a MapReduce version of ELM, named *PELM*, to implement regression for large-scale datasets. *PELM* consists of two MapReduce jobs. Xin et al. [7] design and implement ELM^* which combines the previous two MapReduce jobs into one MapReduce job. Xin et al. [8] propose an incremental algorithm E^2LM to process large-scale updating training datasets. Bi et al. [9] propose a distributed extreme learning machine with kernels based on MapReduce. Wang et al. [10] design a parallel online sequential extreme learning machine based on MapReduce. To the best of our knowledge, ELM^* is the-state-of-the-art parallel ELM algorithm processing large static training datasets. In this paper, we adapt ELM^* into the scenario of massive multi-graph classification.

2.3 Frequent Subgraphs Mining Based on MapReduce

A typical problem of large-scale frequent subgraphs mining can use two settings: (1) one single big graph: its target is to mine subgraphs from one single big graph such that supports of these subgraphs are not smaller than a given support threshold [11, 12]; (2) a large collection of graphs: its target is to mine frequent subgraphs from a large collection of graphs [13, 14]. In this paper, we consider the second setting. Hill et al. [13] propose iterative *MapReduce-FSG* algorithm which is an incremental approach to mine frequent subgraphs from a large collection of graphs. Lin et al. [14] design and implement *MRFSM* approach which contains only three MapReduce jobs. To the best of our knowledge, *MRFSM* is the state-of-the-art frequent subgraphs mining approach for a large collection of graphs. In this paper, *MRFSM* is leveraged during the process of frequent subgraphs generation.

3 Preliminaries

In this section, we firstly define related basic concepts. Then, we give a simple overview of the extreme learning machine.

3.1 Problem Definition

Definition 1 (*Labeled multi-graph*) A multi-graph is a bag of graphs. A labeled multi-graph mg is a multi-graph with binary class label $l(mg) \in \{positive, negative\}$. If the class label for one graph of multi-graph is positive, the multi-graph has a positive class label $l(mg) = positive$. Otherwise, the multi-graph has a negative class label.

Definition 2 (*Feature subgraph representation of multi-graph*) Given a multi-graph set $MG = \{mg_1, mg_2, \ldots, mg_n\}$ and k feature subgraphs $F = \{f_1, \ldots, f_k\}$, $mg_i \in MG$ is represented as a feature vector $v(mg_i)$ of k dimensions. The weight w_i of the *ith* dimension is 1 if the *ith* feature subgraph $f_i \in F$ is a subgraph of one graph for mg_i. Otherwise, w_i is set to 0.

Definition 3 (*Score function of the frequent subgraphs*) The score function of the frequent subgraphs is used to mine feature subgraphs [1]. The score function, named $score(g)$, is as follow.

$$score(g) = 1/2(s(A)/A - s(B)/B + s(C)/C - s(D)/D)$$

Definition 4 (*Multi-graph classification*) Given a labeled multi-graph dataset, we aim to construct prediction model from labeled multi-graphs to predict unseen multi-graphs with maximum precision.

Definition 5 (*Massive multi-graph classification*) Massive multi-graph classification is a special multi-graph classification with large-scale training dataset and large-scale test dataset.

Based on *gMGFL*, we propose a parallel algorithm *ME-MGC* to solve massive multi-graph classification problem. Next, we simply introduce ELM.

3.2 Extreme Learning Machine

Huang proposes ELM for single hidden-layer feedforward neural networks (SLFNs) and then extends it to the "generalized" SLFNs [4, 5, 15–22]. Compared to traditional feedforward neural networks, ELM has better generalization performance, faster learning speed and less training error. The training process of ELM approach is described in Algorithm 1. ELM approach has a wide range of applications, such as protein secondary structure prediction [23], XML document classification [24], classification in P2P networks [25] and graph classification [26].

Algorithm 1: *training process of ELM*

```
Input  : a training set $V$=$(x_i, t_i)|x_i \in R^n, t_i \in R^m, i = 1, \dots, N$, the number of hidden node L
         and activation function g(v)
Output: an ELM instance
1 1)randomly generate every input weight $w_i$ and bias $b_i$, i = 1,…,$L$;
2 2)calculate hidden node output matrix $H$;
3 3)calculate output weight $\beta = H^{\dagger}$T, where $H^{\dagger}$ is the Moore-Penrose generalized inverse of
  matrix $H$, $T = [t_1, \dots, t_N]^T$.
```

4 *ME-MGC* Algorithm

In this section, we propose a *ME-MGC* approach.

4.1 Overview of *ME-MGC*

In this section, an overview of *ME-MGC* algorithm based on MapReduce is provided. Given a multi-graph set $MG = \{mg_1, mg_2, \dots, mg_n\}$ and the graph set $G = \{g|g \in mg_i, mg_i \in MG\}$ consisting of overall graphs of MG. *ME-MGC* contains three steps: (1) mining frequent subgraphs FG of G, (2) mining feature subgraphs F and (3) constructing the predict model. The first step is implemented based on *MRFSM* [14] which has three-round MapReduce jobs: *getCFS* (i.e. getting candidate

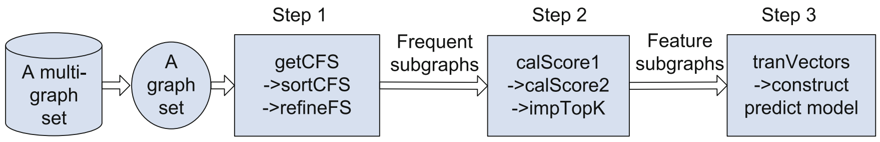

Step 1
Step 2
Step 3
A multi-graph set
A graph set
getCFS ->sortCFS ->refineFS
Frequent subgraphs
calScore1 ->calScore2 ->impTopK
Feature subgraphs
tranVectors ->construct predict model

Fig. 1 An overview of *ME-MGC*

frequent subgraphs); *sortCFS* (i.e. sorting candidate frequent subgraphs); *refineFS* (i.e. refining and obtaining frequent subgraphs). Also, the second step needs three-round MapReduce jobs: *calScore*1 (i.e. calculating partial scores *ps* of frequent subgraphs produced in the first step); *calScore*2 (i.e. aggregating partial results output by calScore1 to get final scores); *impTopK* (i.e. obtaining top-*k* subgraphs, namely feature subgraphs). The last step is completed based on *ELM**. Figure 1 shows the overview processing framework of *ME-MGC*. Next, the aforementioned three steps of *ME-MGC* are represented in detail.

4.2 Mining Frequent Subgraphs

In order to mine frequent subgraphs of a large graph dataset, a MapReduce job chain is implemented consisting of *getCFS*, *sortCFS* and *refineFS*.

4.2.1 Getting Candidate Frequent Subgraphs

GetCFS retrievals candidate frequent subgraphs described as Algorithm 2. In the map phase, each map task outputs the local frequent subgraphs which are candidate subgraphs. In the reduce phase, the upper bounds of frequency of candidate subgraphs are estimated. A candidate frequent subgraph is eliminated whose upper bound is less than minimum frequency threshold.

4.2.2 Sorting Candidate Frequent Subgraphs

SortCFS sorts the candidate subgraphs produced by *getCFS* according to edge size shown as Algorithm 3. We utilize the sort function of MapReduce to improve the performance. The size of every candidate subgraph is used as the key for a map task. Meanwhile, only one reduce task is adopted. The records received by the same reduce task are then sorted by the sort function of MapReduce. In addition, inclusion relations of candidate frequent subgraphs are calculated and output by this reduce task.

Algorithm 2: *getCFS*

```
1  //map task
2  List $graphPartition$;// store a subset of graphs set
3  estimate $f$; //local frequent threshold
4  Map ($< Offset, a\ multi-graph >$)
5      add into $graphPartition$ all graphs of this multi-graph;
6  Cleanup ()
7      calculate local frequent subgraphs $LFS = \{lfs_1, lfs_2, \dots, lfs_i\}$ for $graphPartition$ with
       frequency $fre \geq f$;
8      encode frequent subgraphs $EFS = \{v(lfs_1), v(lfs_2), \dots, v(lfs_i)\}$;
9      //$v(lfs_i)$ is the code of $lfs_i$
10     emit($< v(lfs), (partitionId, fre) >$);
11     //$v(lfs) \in EFS$, $partitionId$ is id of $graphPartition$, $fre$ is local frequency of $v$
12 //reduce task
13 Reduce ($v, list < partitionId, fre >$)
14     calculate the frequency upper bound $fub(v)$ of $v$;
15     if $fub(v) \geq f$ then
16         emit($< v, efs >$);
17     //$efs$ means the sum of exact frequent for $v$
```

4.2.3 Refining and Obtaining Frequent Subgraphs

RefineFS refines the candidate subgraphs and gets the final results. A map task reads a subset S_i of graph data set and sorted candidate subgraphs(SCS). After that, we calculate the local frequency $f_i(cg)$ of candidate subgraph $cg \in SCS$ for the subset S_i in the map phase which outputs key-value pair $< cg, f_i(cg) >$. If the local frequencies of candidate subgraphs have been calculated in the *getCFS*, they do not need to be recalculated. After that, exact global frequency of every candidate subgraph is calculated in the reduce tasks. If the global frequency of a subgraph is no less than the minimum frequency threshold, this subgraph is a desirable frequent subgraph and is output.

After mining frequent subgraphs of G, we mine feature subgraphs from the derived frequent subgraphs.

4.3 Mining Feature Subgraphs

Feature subgraphs $F = \{f_1, f_2, \dots, f_k\}$ are top-k frequent subgraphs with score function $score(fg), fg \in FG$. Different from the traditional top-k query problem, the score calculation does not depend on one record but the overall dataset. So, we mine feature subgraphs using the following two steps instead of adopting traditional top-k query techniques. At first, we calculate scores of frequent subgraphs. Then, we answer top-k query. A MapReduce job chain is designed to complete this. *calScore*1 gets partial scores, which are aggregated by *calScore*2 to get scores $\{score(fg)|fg \in FG\}$. After that, *impTopK* answers top-k query to get F. In the following, we discuss the details of these MapReduce jobs.

Algorithm 3: *sortCFS*

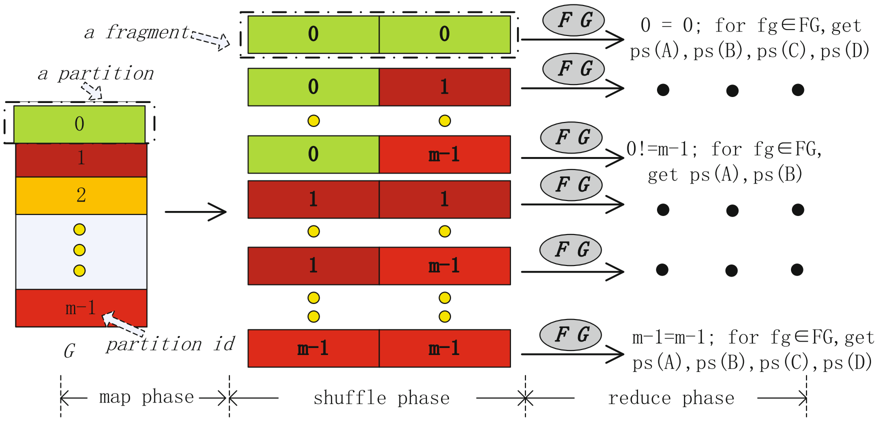
```
//map task
Map (v, efs)
    emit(< s, (v, efs) >);// s is the edge size of v
//reduce task
List canSubGraph=empty; //candidate subgraphs in current layer.
List Id=empty;//ids of candidate subgraphs in canSubGraph
layer=0;
currentId=0;
Reduce (< s, list(v, efs) >)
    if layer==0 then
        layer++;
        for each element (v, efs) ∈ list(v, efs) do
            add v into canSubGraph;
            add currentId into Id;
            currentId++;
            emit(< (v, efs), NULL >)//NULL means having no subgraphs;
    else
        List subGraph=empty;
        for each element (v, efs) ∈ list(v, efs) do
            for each element v' ∈ canSubGraph do
                if subgraphIsomorphismTest(v',v) then
                    add id(v') into subGraph;
            emit(< (v, efs), subGraph >);
        update canSubGraph and Id;
```

4.3.1 Calculating Partial Scores of Frequent Subgraphs

We define some concepts before we introduce *CalScore*1.

Definition 6 (*Partition*) Given a dataset D, a *partition* p_i is a subset of D which meets the following two conditions: (1) $\cup p_i = D$, where $i \geq 0$ and $i \leq m-1$; (2) $p_i \cap p_j = \emptyset$, where $i/j \geq 0$, $i/j \leq m-1$ and $i \neq j$.

Definition 7 (*Fragment*) Given a dataset D and its *partition* set $P = \{p_0, p_1, \dots, p_{m-1}\}$, a fragment is a partition pair $< p_i, p_j >$, $p_i, p_j \in P$. In total, there are $m * (m+1)/2$ fragments for D and P.



a fragment
a partition
0
1
2
m-1
G
partition id
0 0
0 1
0 m-1
1 1
1 m-1
m-1 m-1
F G
0 = 0; for fg∈FG, get ps(A), ps(B), ps(C), ps(D)
0!=m-1; for fg∈FG, get ps(A), ps(B)
m-1=m-1; for fg∈FG, get ps(A), ps(B), ps(C), ps(D)
map phase
shuffle phase
reduce phase

Fig. 2 overview of *calScore*1

*calScore*1 gets the partial results. In map phase, with hash and copy techniques, the multi-graph dataset *MG* is divided into a *partition* set $P = \{p_0, p_1, \dots, p_{m-1}\}$. If $hash(mg_j) = i$, $mg_j \in p_i (i \geq 0$ and $i \leq m - 1)$. In reduce phase, overall fragments for *MG* and *P* are produced to calculate the partial scores of subgraphs showed in Fig. 2.

4.3.2 Getting Final Scores

*CalScore*2 aggregates the results of *calScore*1 to get the final scores of frequent subgraphs. After reading outputs of *canScore*1, map tasks output $< v(fg), partial\ result >$ pairs. Overall partial results of a frequent subgraph are shuffled to the same reduce task and are aggregated to the final score.

4.3.3 Obtaining Feature Subgraphs

ImpTopK selects as feature subgraphs k frequent subgraphs from *FG* whose scores are larger than others. Every map task computes the local top-k frequent subgraphs in the corresponding input split. A reduce task is launched to aggregate overall local top-k frequent subgraphs and to calculate the global top-k frequent subgraphs.

4.3.4 Building Prediction Model

After getting feature subgraphs, multi-graphs are preprocessed to vectors according to *feature subgraph representation of multi − graph*. Then, we build prediction model based on ELM^*.

5 Performance Evaluation

In this section, we compare the precision and recall for ELM and other classification models, and compare the training time, speedup and scalability for *gMGFL* and *ME-MGC* over both real and synthetic datasets.

DBLP dataset. Every paper p_i is regarded as a multi-graph mg_i. The abstract of p_i is a graph $g \in mg_i$, which is obtained using E-FCM [27]. In addition, each reference is modelled as a graph. For example, p_i has m references. So p_i can be represented as a multi-graph including $m + 1$ graphs. Domain field of a paper is treated as its class label. Two domain fields are selected in this paper namely artificial intelligence AI and computer vision CV. After preprocessing, there are 7661 AI multi-graphs and 1817 CV multi-graphs. We randomly select 1817 AI multi-graphs and using all 1817 CV multi-graphs to test.[1]

Synthetic dataset (SYN). Each National Cancer Institute (NCI) data set belongs to a bio-assay task for anti-cancer activity prediction [28]. If a chemical compound is active against the corresponding cancer, it is positive. Otherwise, it is negative. We generate a synthetic multi-graph dataset with a graph data set(with ID 1) [1]. We randomly select one to four positive graphs and several negative graphs to build a positive multi-graph. A negative multi-graph is build by randomly selecting a number of negative graphs. The number of graphs in each multi-graph varies from 1 to 10. In total, we built 500,000 positive and 500,000 negative multi-graphs. The total number of graphs is 4,997,537.

A 31-node (1 master and 30 slaves) cluster is used to test. Every machine is collocated with two 3.1 HZ CPUs, 8 GB Memory, 500 GB hard disk, Redhat 4.4.4-13 operation system and Hadoop-1.2.1. 10-fold cross-validation is adopted. Mean precision and recall are reported in this paper.

5.1 *Precision and Recall*

In this section, we compare precision and recall over variable real and synthetic datasets.[2]

Figure 3 shows the precision on variable DBLP datasets. Figure 3 displays precisions of $gMGFL + ELM$ is highest among all algorithms. Because the precision of *ELM* is higher than *NBayes* and *SVM* on the same DBLP dataset.

[1]DBLP dataset can be downloaded from http://arnetminer.org/citation.

[2]$gMGFL + NBayes(orSVM, orELM)$ denotes *gMGFL* using NBayes, SVM and ELM classification model, respectively. *ME-MGC* + *PNBayes*(*ELM*) represents *ME-MGC* using parallel NBayes and parallel ELM prediction model, respectively. In the case of without causing ambiguity, *ME-MGC* represents *ME-MGC* + *ELM*.

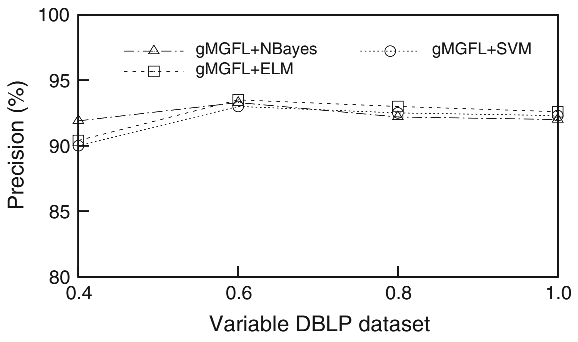


Fig. 3 Precision on variable DBLP datasets ($s = 0.04$, $k = 20$)

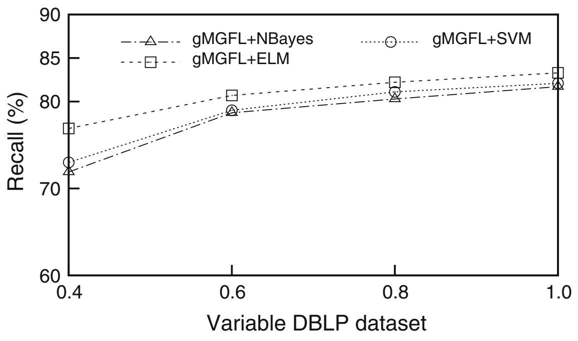


Fig. 4 Recall on variable DBLP datasets ($s = 0.04$, $k = 20$)

Figure 4 shows the recalls on variable DBLP datasets. The recalls of *gMGFL + ELM* are highest among all algorithms because *gMGFL + ELM* adopt *ELM* whose recall is higher than those algorithms embedded with *NBayes* and *SVM* on the same DBLP dataset.

Figure 5 shows the precisions of different methods on variable SYN datasets. Figure 6 shows the recalls of different methods on variable SYN datasets. The experimental results on variable synthetic datasets are similar with variable DBLP datasets. The reasons can also be referred to DBLP datasets.

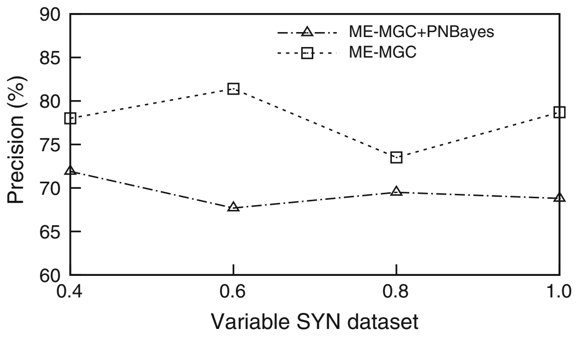


Fig. 5 Precision on variable SYN datasets ($s = 0.12$, $k = 15$)

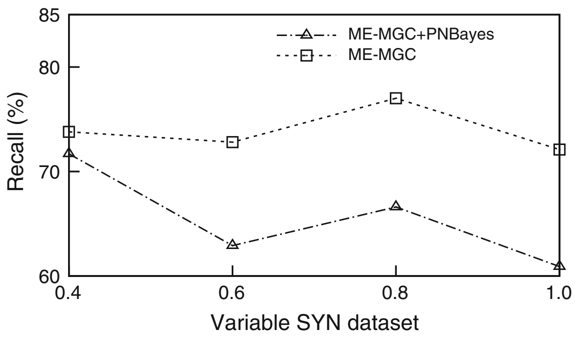


Fig. 6 Recall on variable SYN datasets ($s = 0.12$, $k = 15$)

5.2 Training Time

In this section, we compare the classification model constructing time(training time) of *gMGFL* and *ME-MGC* on variable DBLP and synthetic datasets. *gMGFL* runs in stand-alone. *ME-MGC* runs in 31-node cluster.

Figure 7 exhibits the training time on variable DBLP datasets. *gMGFL* is faster than *ME-MGC*. Because *ME-MGC* consists of several MapReduce jobs. The launch of these jobs costs much time. In addition, *gMGFL* is in-memory algorithm suitable for small datasets.

Figure 8 exhibits the training time on variable synthetic datasets. *gMGFL* can not run on these synthetic datasets successfully.

5.3 Speedup and Scaleup

We evaluate the speedup and scaleup of *ME-MGC* + *PNBayes* and *ME-MGC* + *ELM* on synthetic dataset.

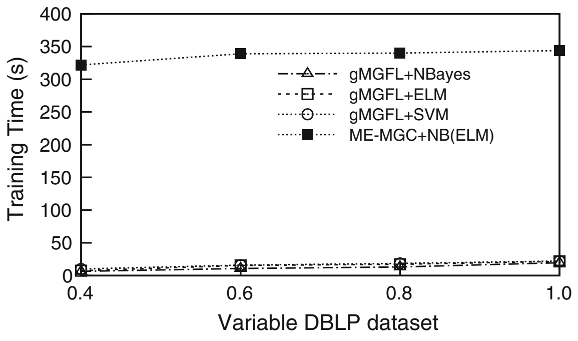


Fig. 7 Training time on variable DBLP datasets ($s = 0.04$, $k = 20$)

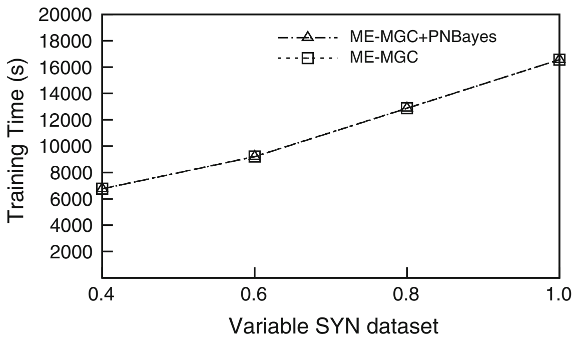


Fig. 8 Training time on variable SYN datasets (s = 0.12, k = 15)

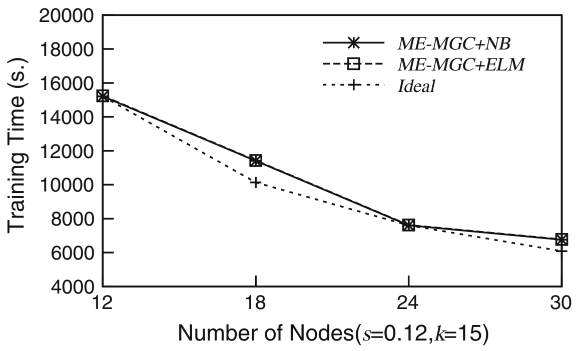


Fig. 9 Training time for processing *SYN* dataset on m-node cluster (where m = 12, 18, 24 and 30)

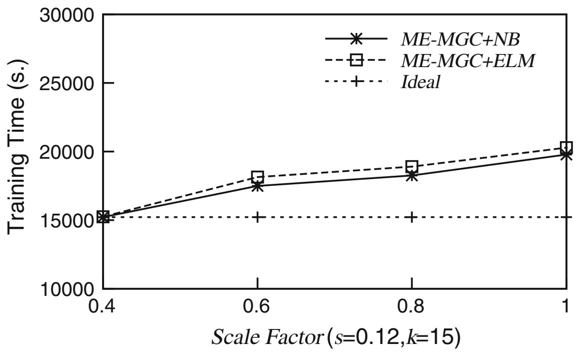


Fig. 10 Training time for processing *SYN**n datasets (where n = 0.4, 0.6, 0.8 and 1.0) on a 31-node cluster

Figure 9 exhibits that both *ME-MGC* + *PNBayes* and *ME-MGC* + *ELM* have a good speedup on synthetic dataset. Figure 10 exhibits that both *ME-MGC* + *PNBayes* and *ME-MGC* + *ELM* have a good scalability on synthetic dataset.

5.4 Performance with Different Hidden Node Number

In this section, we evaluate the precisions and recalls of *gMGFL* and *ME-MGC* for different hidden node number on DBLP and SYN dataset.

Figure 11 shows comparisons of precision and recall for *gMGFL* with variable hidden node number on DBLP dataset. Figure 12 shows comparisons of precision and recall for *ME-MGC* with a variable number of hidden nodes on SYN dataset. With the increment of the hidden node number, precision and recall for *gMGFL* and *ME-MGC* are stable.

Fig. 11 Precision and recall with variable number of hidden nodes on DBLP dataset ($s = 0.04$, $k = 20$)

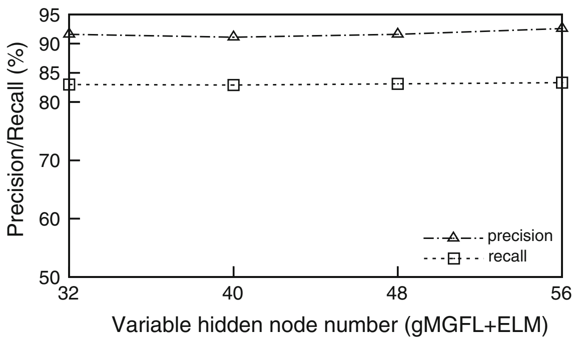
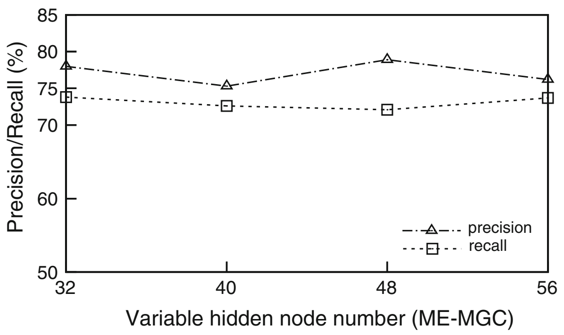


Fig. 12 Precision and recall with variable number of hidden nodes on SYN dataset ($s = 0.12$, $k = 15$)



6 Conclusions

In this paper, we propose a parallel approach *ME-MGC* based on MapReduce to resolve massive multi-graph classification problem. Meanwhile, ELM prediction model is applied to predict multi-graph data type for improving the algorithm performance. Extensive experimental results on both real and synthetic datasets display that our algorithm apparently outperforms *gMGFL* and the extended algorithm with parallel Bayes because of its high precision and good scalability.

Acknowledgments The work is partially supported by the National Basic Research Program of China (973 Program) (No. 2012CB316201), the National Natural Science Foundation of China (No. 61272179, No. 61472071).

References

1. Wu, J., Zhu, X., Zhang, C., et al.: Bag constrained structure pattern mining for multi-graph classification. TKDE **26**(10), 2382–2396 (2014)
2. Wu, J., Pan, S., Zhu, X., et al.: Boosting for multi-graph classification. T. Cybern. **45**(3), 430–443 (2015)
3. MapReduce. http://en.wikipedia.org/wiki/MapReduce
4. Huang, G., Zhu, Q., Siew, C.K.: Extreme learning machine: a new learning scheme of feedforward neural networks. In: IJCNN, pp. 985–990 (2004)
5. Huang, G., Liang, N., Rong, H., et al.: On-line sequential extreme learning machine. In: IASTED, pp. 232–237 (2005)
6. He, Q., Shang, T., Zhuang, F., et al.: Parallel extreme learning manchine for regression based on MapReduce. Neurocomputing **102**(2), 52–58 (2013)
7. Xin, J., Wang, Z., Chen, C., et al.: *ELM**: distributed extreme learning machine with MapReduce. World Wide Web **17**(5), 1189–1204 (2014)
8. Xin, J., Wang, Z., Qu, L., et al.: Elastic extreme learning machine for big data classification. Neurocomputing, **149**(Part A), 464–471 (2015)
9. Bi, X., Zhao, X., Wang, G., et al.: Distributed extreme learning machine with kernels based on MapReduce. Neurocomputing **149**, 456–463 (2015)
10. Wang, B., Huang, S., Qiu, J., et al.: Parallel online sequential extreme learning machine based on MapReduce. Neurocomputing **149**, 224–232 (2015)
11. Kuramochi, M., Karypis, G.: Finding frequent patterns in a large sparse graph. In: SDM, pp. 345–356 (2004)
12. Kuramochi, M., Karypis, G.: Grew-a-scalable frequent subgraph discovery algorithm. In: ICDM, pp. 439–442 (2004)
13. Hill, S., Srichandan, B., Sunderraman, R.: An Iterative mapreduce approach to frequent subgraph mining in biological datasets. In: BCB, pp. 661–666 (2012)
14. Lin, W., Xiao, X., Ghinita, G.: Large-scale frequent subgraph mining in MapReduce. In: ICDE, pp. 844–855 (2014)
15. Huang, G., Zhu, Q., Siew, C.K., et al.: Extreme learning machine: theory and applications. Neurocomputing **70**(1–3), 489–501 (2006)
16. Huang, G., Chen, L.: Enhanced random search based incremental extreme learning machine. Neurocomputing **71**(16–18), 3460–3468 (2008)
17. Huang, G., Ding, X., Zhou, H.: Optimization method based extreme learning machine for classification. Neurocomputing **74**(1–3), 155–163 (2010)

18. Huang, G., Wang, D., Lan, Y.: Extreme learning machines: a survey. Int. J. Mach. Learn. Cybern **2**(2), 107–122 (2011)
19. Huang, G., Zhou, H., Ding, X., et al.: Extreme learning machine for regression and multiclass classification. In: TSMC. Part B Cybern. **42**(2), 513–529 (2012)
20. Huang, G., Wang, D.: Advances in extreme learning machines (ELM2011). Neurocomputing **102**, 1–2 (2013)
21. Huang, G.: An insight into extreme learning machines: random neurons, random features and kernels. Cogn. Comput. **6**(3), 376–390 (2014)
22. Huang, G., Bai, X., Kasun, L.L.C., et al.: Local receptive fields based extreme learning machine. In: Comp. Int. Mag. **10**(2), 18–29 (2015)
23. Wang, G., Zhao, Y., Wang, D.: A protein secondary structure prediction framework based on the extreme learning machine. Neurocomputing **72**(1–3), 262–268 (2008)
24. Zhao, X., Wang, G., Bin, X., et al.: XML document classification based on ELM. Neurocomputing **74**(16), 2444–2451 (2011)
25. Sun, Y., Yuan, Y., Wang, G.: An OS-ELM based distributed ensemble classification framework in P2P networks. Neurocomputing **74**(16), 2438–2443 (2011)
26. Wang, Z., Zhao, Y., Wang, G., et al.: On extending extreme learning machine to non-redundant synergy pattern based graph classification. Neurocomputing **149**, 330–339 (2015)
27. Perusich, K., Senior, M.: Using fuzzy connitive maps for knowledge management in a conflict environment. TSMC. Part C **36**(6), 810–821 (2006)
28. Pubchem. http://pubchem.ncbi.nlm.nih.gov