CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. 2017; 29: e3769
Published online 28 January 2016 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3769

SPECIAL ISSUE PAPER

Parallel algorithms for anomalous subgraph detection

Jieyu Zhao[1], Jianxin Li[1,*], Baojian Zhou[2], Feng Chen[2], Paul Tomchik[2] and Wuyang Ju[1]

[1]*State Key Laboratory of Software Development Environment, Beihang University, Beijing, China*
[2]*College of Engineering and Applied Sciences, University at Albany, SUNY, Albany, NY, USA*

SUMMARY

For the many application domains concerning entities and their connections, often their data can be formally represented as graphs and an important problem is detecting an anomalous subgraph within it. Numerous methods have been proposed to speed-up anomalous subgraph detection; however, each incurs non-trivial costs on detection accuracy. In this paper, we formulate the anomalous subgraph detection problem as the maximization of a non-parametric scan statistic and then approximate it to a submodular maximization problem. We propose two parallel algorithms: non-coordination anomalous subgraph detection (NCASD) and under-coordination anomalous subgraph detection (UCASD)for the anomalous subgraph detection. To the best of our knowledge, this paper is the first to solve this problem in parallel. NCASD emphasizes speed at the expense of approximation guarantees, while UCASD achieves a higher approximation factor through additional coordination controls and reduced parallelism. The experiments demonstrate the effectiveness and efficiency of our proposed approaches in a real-world application domain (water pollution detection), comparing them with five other state-of-the-art methods. Copyright © 2016 John Wiley & Sons, Ltd.

Received 1 November 2015; Revised 15 December 2015; Accepted 15 December 2015

KEY WORDS: anomalous subgraph detection; submodular maximization; parallel algorithm

1. INTRODUCTION

Graphs can be used to describe the entities and the connections between them. Detecting a small subset of vertices with anomaly in one graph is an important problem, which we call anomalous subgraph detection in this paper. Previous works have applied numerous methods to solve the anomalous subgraph detection problem including using sparse principal component analysis [1] or eigenvector L_1 norms of a graph's modularity matrix [2]. In fact, many current works adopt a subset scan approach to detect the anomalous subgraph by maximizing a score function $F(S)$ with or without constraints on the subsets of the data. Existing methods for subset scan statistic can be categorized into two main groups: parametric and non-parametric [3]. The latter does not assume specific forms of distributions for the nodes. Because in practice it is difficult to accurately capture the probability distribution, in this work, we focus on non-parametric scan statistic (NPSS) and consider its general optimization framework:

$$\max_{S \subseteq V} F(S), s.t.\ S \text{ is connected}, \tag{1}$$

where $F(S)$ refers to a pre-defined NPSS function, V refers to the ground set of nodes in the input graph and the constraint on S shows our aim of finding a connected subgraph.

*Correspondence to: Jianxin Li, State Key Laboratory of Software Development Environment, Beihang University, Beijing, China.
†E-mail: lijx@act.buaa.edu.cn

However, three main technical challenges remain:

- The NPSS function $F(S)$ is not a submodular function, and therefore, it does not have known approximation algorithms with theoretical guarantees. Here, submodular is a property of set functions satisfying diminishing marginal returns: $F : 2^V \rightarrow \mathbb{R}$, $\forall A \subseteq B \subseteq V$, and $e \in V \setminus B$, $F(A \cup \{e\}) - F(A) \geqslant F(B \cup \{e\}) - F(B)$.
- The connectivity constraint is difficult to handle. For example, even if we consider a simple modular function $F(S)$: $F(S) = \sum_{v \in S} w(s)$, where $w(v) : v \rightarrow \mathbb{R}$ maps each vertex to a real value [4], the resulting anomalous subgraph detection problem is NP-hard and does not admit any finite approximation algorithm.
- The explosive growth of data volume sizes necessitates the development of new, scalable detection methods. However, most algorithms are inherently serial, making them unsuitable for this problem.

To address the first challenge, we decompose the non-submodular function into the difference between two submodular functions and then approximate the function to its lower-bound submodular function. To address the second challenge, we replace the connectivity constraint with a regularization component based upon the number of connected components of S. Note that the traditional methods consider penalty functions such as graph Laplacian [5] or graph cut penalty [6], because these functions are computationally efficient. However, as our experiments show, these methods do not approximate the connectivity constraint well. To address the third problem, we propose two parallel algorithms for large-scale data sets. These algorithms offer trade-offs between speed and accuracy. The primary contributions of our study are summarized as follows:

- **Development of parallel algorithms for anomalous subgraph detection.** We propose two parallel algorithms for the anomalous subgraph detection: non-coordination anomalous subgraph detection (NCASD) and under-coordination anomalous subgraph detection (UCASD). They are designed with large data volumes in view. To the best of our knowledge, this is the first work that uses parallel algorithms to solve the anomalous subgraph detection problem.
- **Formulation of NCASD/UCASD framework.** We propose the parallel algorithms using empirical p-value of each node [3] to measure its current anomalous level. We show that the basic problem can be reformulated as a non-submodular maximization problem subject to the connectivity constrict on the subset of nodes.
- **Comprehensive experiments to validate the effectiveness and efficiency of the proposed algorithms.** Our algorithms are evaluated by extensive experiments on real-world data sets. The results demonstrate that our algorithms can detect the anomalous subgraph in the large data set quickly with some trade-offs in the accuracy.

2. RELATED WORK

Anomalous subgraph detection. Anomalous subgraph detection can be applied in a wide variety of applications such as event detection and disease outbreak detection. Non-parametric methods do not assume data follow a specific probability distribution and have several advantages over typical parametric scan statistics, including the ability to integrate information from multiple sources and to adapt to different data distributions. Additive GraphScan algorithm, proposed by Speakman [7], allows dynamic subset scan to enforce both soft temporal consistency constraints and hard connectivity constraints while scaling to large, real-world networks. Sharpnack presents Edge Lasso [6], which investigates sparsistency of fused lasso for general graph structures. The results provide necessary and sufficient conditions on the graph properties and the signal-to-noise ratio needed to ensure sparsistency. Chen [3] realizes the event detection with a non-parametric approach. They propose a non-parametric heterogeneous graph scan (NPHGS) model where 'sensor' network is first built and p-value is used to evaluate the anomalous level of each node. The experiment results show that their algorithm can enable early event detection. In this paper, we also adopt the non-parametric scan statistic for the anomalous subgraph detection. All these algorithms discussed previously serve as the baseline algorithms in our experiments.

DOI: 10.1002/cpe

Maximization Submodular function. Submodularity is a desirable property, and the literature on submodular maximization problems is very large. It can be applied in a large class of real-world applications, particularly in machine learning, such as statistical machine translation [8], document summarization [9, 10] and information gathering [11]. Here we just mention a few of the most relevant works. For monotone ($F(A) \leqslant F(B), \forall A \subseteq B$) submodular function, Nemhauser [12] provides a $\frac{1}{2}$-approximation discrete greedy algorithm. It is improved to a tight $1 - \frac{1}{e}$ approximation by [13] with a continuous greedy algorithm. For non-monotone submodular function, an approximation solution with a $\frac{1}{2}$ guarantee in [14] is provided using a double greedy algorithm. Pan [15] improves it to parallel double greedy algorithms with different coordination constraints. In our paper, we will show our problem can be formulated as a non-monotone submodular maximization problem and will adopt the corresponding algorithms for it.

3. PROBLEM FORMULATION

Consider a graph as $G(V, E, p)$, where $V = \{v_1, v_2, \ldots, v_{|V|}\}$ refers to the set of nodes, $E \subseteq V \times V$ refers to the set of edges, and $p : V \to [0, 1]$ defines a function that maps a node v to an empirical p-value, which can be calculated based on empirical calibration by comparing the current features of this vertex with its features in historical data of no event [3]. The empirical p-value can be seen as the signal strength of the node's current attribute values inkling an indicator of some ongoing or newly emerging event.

3.1. Non-parametric scan statistic

As described, our goal is to find a subset of vertices $S \subseteq V$ that has the most anomaly according to the NPSS. Given a subset S, the general form of NPSS is defined as:

$$F(S) = \varphi(\alpha, N_\alpha(S), N(S)), \tag{2}$$

where $N_\alpha(S) = |v_i,\ v_i \in S,\ p(v_i) \leqslant \alpha|$ is the number of p-values significant at level α, $N(S) = |S|$ is the total number of p-values in S and α refers to the predefined significant level (e.g. 0.05). The function $\varphi(\alpha, N_\alpha(S), N(S))$ refers to a non-parametric scan statistic that compares the observed number of p-values that are significant at level α to the expected number of significant p-values. In this work, we explore the use of one non-parametric scan statistic: the Berk–Jones (BJ) statistic [16], stated as

$$F_{BJ}(S) = N(S) \times KL\left(\frac{N_\alpha(S)}{N(S)}, \alpha\right), \tag{3}$$

where KL is the Kullback–Liebler divergence between the observed and expected proportions of p-values less than α:

$$KL(x, y) = x \log \frac{x}{y} + (1 - x) \log \frac{1 - x}{1 - y}. \tag{4}$$

Recent theoretical and empirical studies have demonstrated that the BJ statistic exhibits outstanding performance for detecting anomalous patterns, in comparison with other non-parametric statistics, such as the Higher Criticism (HC) statistic in a number of applications [3, 17–19].

Whether the empirical p-values follow a uniform or piecewise constant distribution is tested by the BJ statistic, in a log-likelihood ratio form. By combining Eqs. (3) and (4), the BJ function can be decomposed to the difference between two submodular functions $r(S)$ and $g(S)$ [28]:

$$F_{BJ}(S) = r(S) - g(S), \tag{5}$$

where

$$r(S) = -N(S) \log N(S), \tag{6}$$

DOI: 10.1002/cpe

$$g(S) = -N_\alpha(S)\log\left(\frac{N_\alpha(S)}{\alpha}\right) - \bar{N}_\alpha(S)\log\left(\frac{\bar{N}_\alpha(S)}{1-\alpha}\right), \tag{7}$$

$$\bar{N}_\alpha(S) = N(S) - N_\alpha(S). \tag{8}$$

The submodularity of function $r(S)$ and $g(S)$ can be proved by the definition and Proposition 6.1 of [20].

3.2. Anomalous subgraph detection

Our target aims at detecting the anomalous subgraph, which makes a connectivity constraint that must be considered when we choose the subset of nodes, while the aforementioned statistic only considers the maximization problem by finding a subset S of V that is interesting to us without any extra constraints. In this work, we use a connected-component-related factor $\phi(S)$ as the connectivity penalty of subset S.

PROBLEM 1 (ANOMALOUS SUBGRAPH DETECTION) *Given a graph $G = (V, E, p)$, a trade-off parameter λ and a predefined significant parameter α, try to find a subset of vertices $S \subseteq V$ maximizing the function*

$$\max_{S\subseteq V} F_{ASD}(S) = \max_{S\subseteq V} r(S) - g(S) + \lambda\phi(S), \tag{9}$$

where $\phi(S) = -[c(S)-(|V|-|S|)-1],\ S \subseteq V$. Here $c(S)$ is the number of connected components in subgraph $G(V, E_S, p)$, where E_S stands for the edge set corresponding to S. $|V|-|S|$ calculates the number of connected components of the subgraph $G(V \setminus S, E_S, p)$. The subtraction of one element in the function ensures that when the subgraph is connected, the function will not obtain any penalty. $\lambda \in \mathbb{R}$ is the trade-off parameter. If λ is a small positive value, the penalty of the optimal subset of this problem S will be weak. If λ is a large positive value, the penalty of the optimal subset of this problem S will be strong.

Corollary 1
The connected component set function $\phi(S)$ is a submodular function.

Proof
According to [21], $c(S)$ is a supermodular set-function. Here, a supermodular function means that $-c(S)$ is submodular. If f is both supermodular and submodular, it is modular. Thus, we know that $|V| - |S|$ is a modular function. So $c(S) - (|V| - |S|) - 1$ is a supermodular function and thus we obtain $\phi(S)$ is submodular. □

3.3. Submodularized anomalous subgraph detection

Submodular functions have been adopted in numerous areas such as game theory and graph theory. When we begin to discuss the anomalous subgraph detection problem by looking at the properties of the objective function $F(S)$, we find that $r(S)$, $g(S)$ and $\phi(S)$ are all submodular. However, the difference between two submodular functions is no longer submodular. Because we deal with the maximization problem, next we will approximate the difference function as its submodular lower-bound function, which we denote by SASD and seek for efficient solutions based on submodular maximization techniques.

PROBLEM 2 (SASD) *Given a graph $G = (V, E, p)$, a trade-off parameter λ and a predefined significant parameter α, try to find a subset of vertices $S \subseteq V$ maximizing the function*

$$\max_{S\subseteq V} F_{SASD}(S) = \max_{S\subseteq V} r(S) - M^g_{G_X}(S) + \lambda\phi(S), \tag{10}$$

where $M^g_{G_X}(S)$ is the modular upper bound of function of $g(S)$ (i.e. $g(S) \leqslant M^g_{G_X}(S)$) [22] shown in Eqs. (11) and (12). X means a given set, and V means the whole ground set.

DOI: 10.1002/cpe

$$M^{g}_{G_{X,1}}(S) \triangleq g(X) - \sum_{j \in X \setminus S} g(j|X \setminus j) + \sum_{j \in S \setminus X} g(j|\emptyset), \tag{11}$$

$$M^{g}_{G_{X,2}}(S) \triangleq g(X) - \sum_{j \in X \setminus S} g(j|V \setminus j) + \sum_{j \in S \setminus X} g(j|X). \tag{12}$$

Here $f(A|B) \triangleq f(A \cup B) - f(B)$ is the gain of adding A in the context of B. We observe that non-parametric scan statistic functions are mostly non-submodular. As any non-submodular function can be decomposed to the difference between two submodular functions [22], our approach is applicable to these non-parametric scan statistic functions.

4. PARALLEL ALGORITHMS FOR SUBMODULARIZED ANOMALOUS SUBGRAPH DETECTION

Many of today's applications consist of a very large ground set $|V| = n$, which cannot be stored and calculated in one single computer. Hence, this work focuses on seeking a solution that is suitable for large-scale computation. In this section, we present our parallel algorithms for the SASD problem, starting with a brief description of double greedy algorithm, which is the base of our algorithm.

4.1. Greedy algorithms

Many methods have been proposed to solve the submodular functions, including [23], which offers parallel version of a greedy algorithm for maximizing the monotone submodular functions. However, many applications cannot be reduced to a monotone function. For these non-monotone submodular functions, the naive greedy algorithm may bring an arbitrarily poor performance. A serial double greedy algorithm running in linear time was proposed by Buchibinder [14] that works well for the non-monotone submodular functions. This algorithm can guarantee a $\frac{1}{2}$ approximation expectation with any order of the elements.

The double greedy algorithm is based on the fact that if function $f(S)$ is submodular, $f'(S) = f(V \setminus S)$ is also a submodular function. When we obtain an optimal set S^* for f, $V \setminus S^*$ is the optimal solution for the function $f'(S)$ [24]. This algorithm searches for the optimal solutions for both f and $\overline{f}$, and it can be seen as two greedy (double greedy) algorithms simultaneously. This algorithm maintains two sets, S and T. Initially, $S_0 = \emptyset$ and $T_0 = V$. In iteration i, S_{i-1} contains those elements selected before i, T_{i-1} contains S_{i-1} and the elements whose inclusion in S or exclusion from T has yet to be decided. Clearly, $S_{i-1} \subseteq T_{i-1}$. The algorithm serially considers each item (e.g. item u) in the ground set V to decide whether to keep the item (add to S_i) or discard it (remove from T_i) based on the marginal gains. The marginal gain of keeping item u equals $\delta_+(u) = F(S_{i-1} \cup \{u\}) - F(S_{i-1})$, and the marginal gain of discarding u equals $\delta_-(u) = F(T_{i-1} \setminus \{u\}) - F(T_{i-1})$. After each iteration, the algorithm updates the result by growing S or shrinking T. When $S = T$, the algorithm stops.

4.2. Parallel algorithms for the SASD *problem*

Currently, data volumes are increasing fast, which brings a great challenge for algorithms to process them. However, even the most state-of-art algorithms solve the anomalous subgraph detection problem serially, and as data sizes increase, it will become unsuitable.

Our algorithms are based on the parallel perspective of Pan [15]. Program states (the set A and B) are recast as data, and the operations (add the element to A and remove element from B) are seen as transactions. A coordinated bounds approach is designed to address the parallel transaction processing: on each client, it constructs a transaction with one element under the assumption of a change to the program state (i.e add to A or remove from B). When the change is under bound, the transaction can be constructed successfully; if not, the transaction will be sent to the server and the client starts to deal with the next element. On the server, it reconstructs the transaction and applies it under the global program state. It maintains two sets A and B for solutions of f and $\overline{f}$, respectively.

DOI: 10.1002/cpe

Initially, $A = \emptyset,\ B = V$. For each element, the strategy decides whether to add it to A or remove it from B. All the clients deal with the transactions under the bound generated by the server. By adjusting the bound, it can span from non-coordination to serial executions.

Algorithm 1 UCASD: Parallel algorithm for Under Coordination Anomalous SubGraph Detection based on the approach of [15]

1: $\widehat{A} = \tilde{A} = \emptyset,\ \widehat{B} = \tilde{B} = V,\ S^t = \emptyset$
2: do in parallel
3: **while** $\exists\, u \in V\ undecided$ **do**
4: with $m_u \sim Unif(0, 1)$
5: $\widehat{A}_u = \widehat{A},\ \widehat{B}_u = \widehat{B},\ \tilde{A}_u = \tilde{A} \cup \{u\},\ \tilde{B}_u = \tilde{B} \setminus \{u\}$
6: // calculate marginal gains with Eq. (10)
7: $\alpha_{max}(u) = F_{SASD}(\widehat{A}_u \cup \{u\}; S^t) - F_{SASD}(\widehat{A}_u; S^t)$
8: $\beta_{max}(u) = F_{SASD}(\widehat{B}_u \setminus \{u\}; S^t) - F_{SASD}(\widehat{B}_u; S^t)$
9: $\alpha_{min}(u) = F_{SASD}(\tilde{A}_u; S^t) - F_{SASD}(\tilde{A}_u \setminus \{u\}; S^t)$
10: $\beta_{min}(u) = F_{SASD}(\tilde{B}_u; S^t) - F_{SASD}(\tilde{B}_u \cup \{u\}; S^t)$
11: **if** $m_u < \dfrac{[\alpha_{min}(u)]_+}{[\alpha_{min}(u)]_+ + [\beta_{max}(u)]_+}$ **then**
12: $FLAG = 0$
13: **else if** $m_u > \dfrac{[\alpha_{max}(u)]_+}{[\alpha_{max}(u)]_+ + [\beta_{min}(u)]_+}$ **then**
14: $FLAG = 1$
15: **else**
16: $FLAG = getGlobalFlag(A, B, u)$
17: **end if**
18: **if** $FLAG = 0$ **then**
19: $\widehat{A} = \widehat{A} \cup \{u\},\ \tilde{A} = \tilde{A} \cup \{u\},\ S^t = \widehat{A}$
20: **else**
21: $\tilde{B} = \tilde{B} \setminus \{u\},\ \widehat{B} = \widehat{B} \setminus \{u\},\ S^t = \widehat{A}$
22: **end if**
23: **end while**
24: **return** $\widehat{A}$

We solve our anomalous subgraph detection problem from the view in the previous discussion. Two parallel algorithms for the anomalous subgraph detection problem are proposed with different coordination constraints. NCASD runs with fewer coordination guarantees, while UCASD is guaranteed with more concurrency controls. Because NCASD can be obtained when we remove some concurrency controls from the UCASD, in this paper, we just give a detailed description of the UCASD algorithm, which is shown in Algorithms 1 and 2. In this algorithm, when the bound is unable to construct the transaction, the client sends the proposed change to the server. The server handles these elements under global result sets $\widehat{A}$ and $\widehat{B}$, which is described as *getGlobalFlag* in Line 17 of Algorithm 1. The notion $[x]_+$ means $\max\{x, 0\}$. When approximating the ASD problem to the SASD problem, we have two different forms, which may bring different results. In this situation, we choose the result that can make the F_{ASD} larger, which is shown in Line 14 in Algorithm 2. In fact, when calculating the marginal gains in Algorithm 1, we also consider the two different forms, which are not shown here for brevity.

Non-coordination anomalous subgraph detection just uses α_{max} and β_{max} to generate one bound and all the transactions will be finished under this bound, which makes a weaker guarantee than UCASD. When the thread number equals 1, NCASD is the same with the serial double greedy algorithm [14]. During the parallel processing, we set the result sets as global variables. On each client, before each element to be handled, we need to get all the latest copies of these result sets, that is $\widehat{A}$, $\widehat{B}$, $\tilde{A}$, $\tilde{B}$. When finishing the current element, the changes to the result sets will be update.

Figure 1 describes the parallel algorithms schematically. For the SASD problem, the parallel algorithms have the following interpretation: several clients proceed the data in parallel. At one

DOI: 10.1002/cpe

Algorithm 2 UCASD: getGlobalFlag(A, B, u)

wait until all the elements before u has processed
if $FLAG = -1$ **then**
  for $i = \{1, 2\}$ **do**
    // with Eq. (11) and (12)
    $F^i_{SASD}(S) = r(S) - M^g_{G_{X,i}}(S) + \lambda\phi(S)$
    $\alpha^i_{real}(u) = F^i_{SASD}(\widehat{A} \cup u; S^t) - F^i_{SASD}(\widehat{A}; S^t)$
    $\beta^i_{real}(u) = F^i_{SASD}(\widehat{B} \setminus \{u\}; S^t) - F^i_{SASD}(\widehat{B}; S^t)$
    if $m_u < \dfrac{[\alpha^i_{real}(u)]_+}{[\alpha^i_{real}(u)]_+ + [\beta^i_{real}(u)]_+}$ **then**
      $S^t_i = \widehat{A} \cup \{u\}$, $FLAG_i = 0$
    else
      $S^t_i = \widehat{A}$, $FLAG_i = 1$
    end if
  end for
  if $F_{ASD}\left(S^t_1\right) \geqslant F_{ASD}\left(S^t_2\right)$ **then**
    $FLAG = FLAG_1$
  else
    $FLAG = FLAG_2$
  end if
end if
return $FLAG$

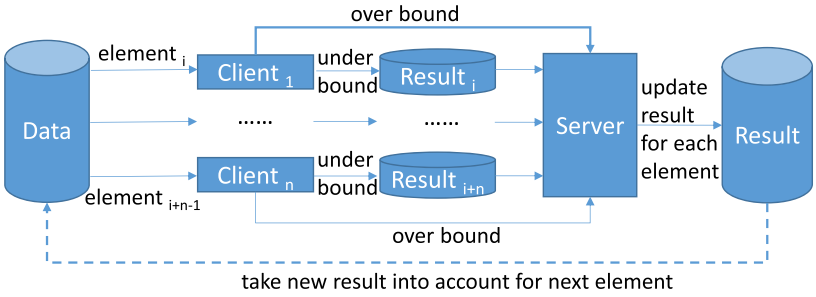


Figure 1. Illustration of the parallel algorithm. Each element is processed by one client. If this transaction can be finished under the bound, the client can directly inform the server of the changed result. Otherwise, the element will be processed on the server. After each element is finished, the server updates the result, which will be taken into account for the next element to be processed.

moment, one vertex of a graph is examined in one client to decide whether to keep this vertex in the solution or remove it. This is finished randomly and the probabilities depend on the marginal gains relating to the F_{SASD} function with respect to the current lower and upper bound solutions, $\widehat{A}$ and $\widehat{B}$. We have the following.

Theorem 1 ([15])
The UCASD algorithm provides us a $\frac{1}{2}$-approximation factor for the SASD problem.

Proof
Let A^s and B^s stand for the solution generated by serial double greedy algorithm, then we have that, at the beginning,$A_0 = A^s_0 = \emptyset$, $B_0 = B^s_0 = V$. Suppose that $A_{i-1} = A^s_{i-1}$, $B_{i-1} = B^s_{i-1}$, and u is the i-th element to process. We see in Algorithms 1 and 2 $u \in A$ iff $m_u < \frac{[\alpha_{min}(u)]_+}{[\alpha_{min}(u)]_+ + [\beta_{max}(u)]_+}$ or $m_u < \frac{[\alpha_{real}(u)]_+}{[\alpha_{real}(u)]_+ + [\beta_{real}(u)]_+}$. If in both cases it satisfies $m_u < \frac{[\delta_+(u)]_+}{[\delta_+(u)]_+ + [\delta_-(u)]_+}$, we can obtain $A_i = A^s_i$ by induction. Then the NCASD offers us a $\frac{1}{2}$-factor approximate algorithm for function $F_{SASD}(S)$, $F_{SASD}(A) \geqslant \frac{1}{2} F_{SASD}(OPT)$.

DOI: 10.1002/cpe

Next, we are going to prove that inequality of m_u works. For element u, we use $\iota(u)$ to denote its index in all the elements. Consider any element $u' \in V \setminus u$. If $u' \in \widehat{A}_u$, it implies that $\iota(u') < \iota(u)$. So we obtain $u' \in A^{\iota(u)-1}$, which means $\widehat{A}_u \subseteq A^{\iota(u)-1}$.

Now suppose $u' \in A^{\iota(u)-1}$, we can infer that we have added u' to $\tilde{A}_{u'}$ (guaranteed by the algorithm). Thus, $u' \in \tilde{A}_u$. So we can obtain that

$$\widehat{A}_u \subseteq A^{\iota(u)-1} \subseteq \tilde{A}_u \setminus u$$

Similarly, if $u' \notin \widehat{B}_u$, it means that we have removed it from B, so $\iota(u') < \iota(u)$. Therefore, $u' \notin A^{\iota(u)-1}$ and $u' \notin B^{\iota(u)-1}$. So $\widehat{B}_u \supseteq B^{\iota(u)-1}$.

Now suppose $u' \notin B^{\iota(u)-1}$, we can infer that we have removed u' from $\tilde{B}_{u'}$ (guaranteed by the algorithm). Thus $u' \notin \tilde{B}_u$. So we can obtain that

$$\widehat{B}_u \supseteq B^{\iota(u)-1} \supseteq \tilde{B}_u \cup u$$

Because the algorithm ensures the committed order of all the elements, we can know that $A^{\iota(u)-1} = A^s$ and $B^{\iota(u)-1} = B^s$. By submodularity, we can obtain that

$$\alpha_{min}(u) \leqslant \delta_+(u) \leqslant \alpha_{max}(u),$$

and

$$\beta_{min}(u) \leqslant \delta_-(u) \leqslant \beta_{max}(u).$$

Because in the UCASD, when committing element u, all the elements before it has to be processed, we have $\widehat{A} = A^{\iota(u)-1}$ and $\widehat{B} = B^{\iota(u)-1}$. Thus, we obtain $\alpha_{real}(u) = \delta_+(u)$ and $\beta_{real}(u) = \delta_-(u)$.

By using the aforementioned bounds, we can obtain that in both cases, $m_u < \frac{[\delta_+(u)]_+}{[\delta_+(u)]_+ + [\delta_-(u)]_+}$, which brings a $\frac{1}{2}$-approximation factor for the SASD problem. □

4.3. Experiment setting

4.3.1. Data source. We obtain two data sources for anomalous subgraph detection: (1) benchmark data sets. We conduct experiments on benchmark data sets, where the graphs are made up with grid data. We generate these data sets with different numbers of nodes varying from 100 to 10 000, using the random walk to produce the subgraphs. Each node of the graph has four neighbours. (2) water pollution data set. The 'Battle of the Water Sensor Networks' [25] provides a real-world network of 12 527 nodes, among which 25 nodes are with chemical contaminant plumes distributed in four different areas. The spreads of these contaminant plumes on graph are simulated using the water network simulator EPANET that is used in Battle of the Water Sensor Networks for a period of 8 h. For each hour, each node has a sensor that reports 1 if it is polluted; otherwise, reports 0. We randomly selected K percent vertices and flip their sensor binary values, where $K = 0, 2, 4, 6, 8, 10, 14, 18, 20, 24, 28, 30$, in order to test the robustness of subgraph detection methods to noises. In order to apply non-parametric graph scan methods to this data set, we map the sensors whose report values are 1s to empirical p-value 0.05 and those whose report values are 0s to 1.0.

4.3.2. Experiment setup. We implement the parallel anomalous subgraph detection algorithms in Java. In this section, our experiments are conducted on one physical machine, configured with 32 Intel(R) Xeon(R) CPU E5-26500 @ 2.00 GHz processors, 256 GB memory and Linux 2.6.32-5-amd64. During our experiments, we occupy up to 10 processors and our algorithms can run in any environment with Java Runtime Environment. We also run our algorithm in a distributed system, which is described in Section 4.5.

DOI: 10.1002/cpe

4.3.3. Parameter settings. Our methods adopt non-parametric scan statistics, which makes the specification of parameters relatively simple. In our model, we just need to define the value of parameter α and λ. During our experiments, we need α to judge if a node is anomalous and calculate the function $F(S)$. An inappropriate α may lead to failure in detecting a few highly anomalous p-values (much smaller than α) or many subtly anomalous p-values (slightly smaller than α). In practice, we make α slightly greater than typical significance levels predefined by users with the value 0.15, which is based on empirical values. $\lambda \in \mathbb{R}$ is the trade-off parameter in Eq. (10). If λ is a small positive value, the penalty of the optimal subset S will be weak. In our experiments, we set $\lambda = 1000$.

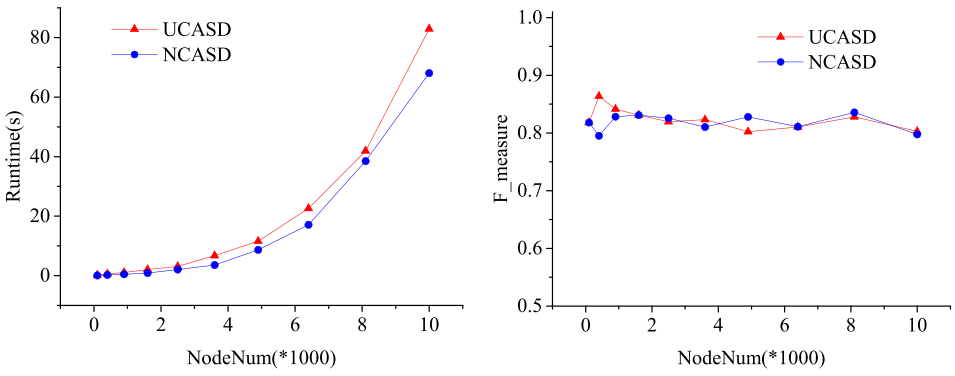
4.3.4. Comparison methods. We compare our proposed parallel algorithms with five existing representative anomalous subgraph detection methods, including Edge Lasso, NPHGS, Additive Graph Scan, graph Laplacian regularization [5] and depth first [26]. The first three algorithms have been introduced in Section 2. Here, we just give some brief introduction of the last two algorithms. Graph Laplacian regularization presents a solution for detecting changepoints in networks under Gaussian noise. The performance of this method depends on the spectrum of the graph, and this result can derive the asymptotic properties on few graph topologies. Speakman uses a depth-first search with backtracking to solve the graph scan. By ruling out subsets provably suboptimal, it gains speed improvements. The experiments of depth-first algorithm show that it can improve spatial accuracy and increase robustness to the occurrence of false negatives. The main difference between these methods is that Edge Lasso and graph Laplacian regularization do not consider the connectivity of the subgraph.

4.3.5. Performance metrics. We evaluate the experiment result by its similarity with true water pollution nodes. For each experiment, we consider related performance metrics including the *Runtime* and accuracy. The accuracy is evaluated by the *F_measure*, where $F\_measure = \frac{2 \cdot precision \cdot recall}{precision + recall}$.

4.4. Experimental results

This section compares our proposed algorithms and five baseline methods on overall runtime and accuracy with the benchmark data sets and the real-world water pollution data set. The experiment results are shown in Figures 2 and 3.

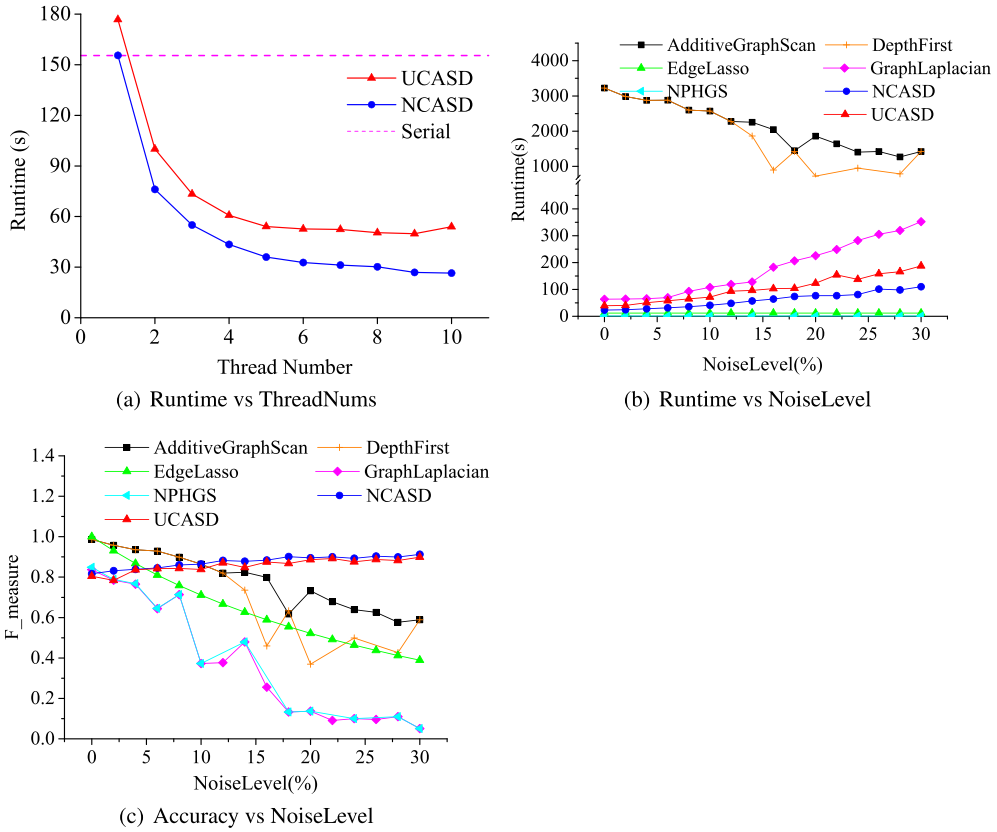
4.4.1. Benchmark data sets. Both of our algorithms can solve the SASD problem efficiently. Figure 2 shows the comparison between our algorithms on the runtime and accuracy. The difference



(a) Runtime vs NodeNum (b) Accuracy vs NodeNum

Figure 2. Experiment results of non-coordination anomalous subgraph detection (NCASD)/under-coordination anomalous subgraph detection (UCASD) on benchmark data sets. It shows the *Runtime* and accuracy of our algorithms with different numbers of nodes.

DOI: 10.1002/cpe



(a) Runtime vs ThreadNums

(b) Runtime vs NoiseLevel

(c) Accuracy vs NoiseLevel

Figure 3. Experimental results. (a) Runtime of our algorithms with different thread numbers. (b) Runtime of different algorithms at different noise levels. (c) Accuracy measures of different algorithms.

on *Runtime* of these two algorithms can be seen from Figure 2(a). Both algorithms need to run longer when the number of nodes increases. And NCASD can always run faster than UCASD no matter how many nodes they need to process. It seems that more nodes will lead to the difference between the runtime of two algorithms becoming more apparent. As for the accuracy shown in Figure 2, both algorithms keep a stable accuracy. Even though the number of nodes increases from 100 to 10 000, they can always guarantee a *F_measure* about 0.8, which means a pretty high accuracy. Note that our proposed algorithms are parallel ones, when we compare the runtime and accuracy of UCASD and NCASD in those data sets, we run them with the same thread number. In this case, we set the thread number equal to 2.

4.4.2. Water pollution data set. Besides using the benchmark data sets, we also test our algorithms on the real-world water pollution data set. We compare the *Runtime* between two algorithms when we change the parallelism. At the same time, we make a comparison between our algorithms and five other baseline methods. When comparing the *Runtime* of the two parallel algorithms, we run the experiments on the data set without any noise. When running these algorithms on data sets with different noise levels, we set the thread number fixed with a value of 10 as one case.

We can see from Figure 3(a) that both parallel algorithms we proposed are faster than the serial algorithms, and they show good speedup properties when more threads are added. In general, the NCASD can run faster than the UCASD. When the thread number equals 1, UCASD has a longer runtime because of the concurrency controls before each element is to be proceeded, which brings a longer runtime.

We test the ability of our algorithms to detect anomalous subgraph with comparison with five state-of-the-art methods. Figures 3(b) and (c) presents results for the runtime and the accuracy under different noise levels. Figure 3(b) shows that under different noise levels, our algorithms can run

DOI: 10.1002/cpe

faster than Additive Graph Scan, depth first and graph Laplacian regularization. Still, NCASD runs faster than UCASD. And they are both a little slower than Edge Lasso and NPHGS.

When it comes to the accuracy, we can see that although Edge Lasso and NPHGS can run faster than our algorithms, the accuracy of these two algorithms decrease acutely when the noise level increases, which is shown in Figure 3. It can also be seen that the accuracy of the graph Laplacian regularization also decreases dramatically when increasing the noise level. Both NPHGS and Graph Laplacian regularization decrease from about 0.85 to below 0.1. Although Additive Graph Scan, depth first and Edge Lasso can have a higher $F\_measure$ at first, they will decrease under our algorithms when the noise level increases. For our proposed algorithms, the accuracy remains stable and high, always around 0.8 even for noise levels of 30%, which is better than most of the other algorithms.

4.5. *Distributed execution experiments*

To satisfy the large data volume, like our previous implementation experience of event detection system BEE+ on Spark [27], we also test our algorithm on the distributed system. *Spark* is a fast and general engine that can be used in large-scale data processing. The fundamental programming abstraction is called resilient distributed datasets (RDDs), which is a logical collection of data partitioned across machines. Result sets $\widehat{A}$ and $\widehat{B}$ are redefined as RDD_A and RDD_B. Because of its specific framework, we modify the algorithm to make it run in iteration. In each iteration, the worker nodes in *Spark* will act as the client. We assign four worker nodes in this case, and these worker nodes go through the whole input graph and obtain the result RDDs. Then the algorithm will take these two result RDDs (RDD_A and RDD_B) into account for the next iteration. Considering the high communication cost when updating parameters on all worker nodes, we delay the operation of updating the result sets. During each iteration, the algorithm no longer updates the result sets $\widehat{A}$ and $\widehat{B}$ after each node is processed. Instead, the update operations of RDD_A and RDD_B occur only when all the worker nodes finish the execution on the data partitions. For the brevity, we just modify the NCASD algorithm and the experimental result is shown in Figure 4. In this situation, NCASD and UCASD algorithms both run with thread number equal to 2.

Because the distributed algorithm runs in iteration, it will bring a result with higher accuracy while trading-off of the runtime. During the experiment, we stop the distributed algorithm when the number of newly added nodes is less than 30 (about 2% of true abnormal nodes). This threshold can be modified by users. A larger threshold will bring a shorter runtime with lower accuracy. It takes 1743 seconds, 21 iterations to finish the detection, which means each iteration takes about

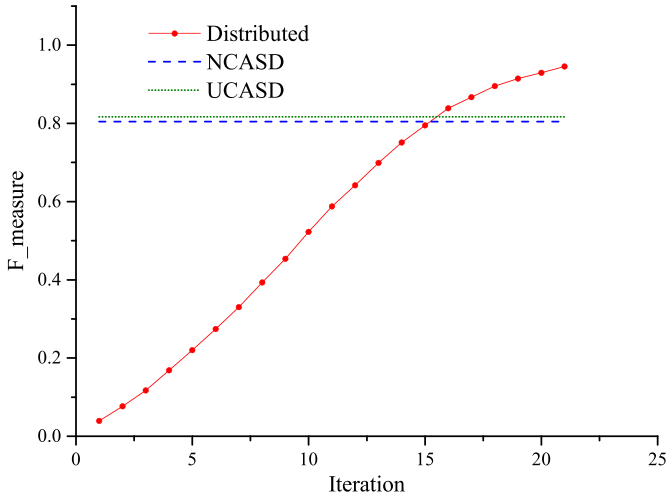


Figure 4. Experimental result of distributed anomalous subgraph detection based on NCASD algorithm. It shows the accuracy of different algorithms when the distributed algorithm runs in different iterations. NCASD, non-coordination anomalous subgraph detection; UCASD, under-coordination anomalous subgraph detection.

DOI: 10.1002/cpe

83 seconds approximately equal to the runtime of NCASD when the thread number is equal to 2. The result in Figure 4 proves that the distributed algorithm can detect more abnormal nodes after each iteration, and it eventually brings a higher accuracy than the NCASD and UCASD algorithms. While communication cost in *Spark* results in longer runtime, the experiment result shows that the distributed algorithm can bring a more accurate detection of the anomalous subgraph.

5. CONCLUSIONS

In this paper, we propose a non-parametric scan statistic model for the anomalous subgraph detection using BJ statistic to modify it. Considering that the problem of adding a simple connectivity constraint will also lead to an NP problem, we add the number of connected component as the penalty to the object function. We can prove that this penalty function has the submodularity, which can guarantee the submodularity of the whole object function. We eventually approximate the object function as a submodular function. While existing works mainly work serially, which is not suitable for the large data volumes, we propose two algorithms for the submodular model: NCASD and UCASD. There are differences between these two algorithms in complexity and approximation factor. Both of these two algorithms analyse the anomaly of each node in the graph and calculate the marginal gain of adding or removing the node to the result sets. Comparing with the threshold, we can make the final change to this node. NCASD can obtain the final result with only one threshold, while UCASD needs two thresholds, and will submit those elements out of the bounds to the server. As far as we know, it is the first work to solve the anomalous subgraph detection problem in parallel. The experiment results show that our methods achieve significant improvements in speed with some trade-offs of accuracy. These two algorithms can be formulated to distributed methods. Unlike the parallel algorithms where the communication cost is far less than the computing process, the distributed algorithms need to reconsider the high communication costs and delays in the distributed system. In the future, our ongoing works will include improving distributed algorithms in the communication efficiency, dealing with graphs which have different properties in each node and detecting several anomalous subgraphs at the same time.

ACKNOWLEDGEMENTS

This work is supported by the NSFC Program (No.61472022), China MOST project (no. 2012BAH46B04), and SKLSDE-2014ZX-04.

REFERENCES

1. Singh N, Miller BA, Bliss NT, Wolfe PJ. Anomalous subgraph detection via sparse principal component analysis. *2011 IEEE Statistical Signal Processing Workshop (SSP)*, IEEE, Nice, France, 2011; 485–488.
2. Miller B, Bliss N, Wolfe PJ. Subgraph detection using eigenvector l1 norms. *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2010; 1633–1641.
3. Chen F, Neill DB. Non-parametric scan statistics for event detection and forecasting in heterogeneous social media graphs. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York City, 2014; 1166–1175.
4. Bogdanov P, Mongiovi M, Singh AK. Mining heavy subgraphs in time-evolving networks. *2011 IEEE 11th International Conference on Data Mining (ICDM)*, IEEE, Vancouver, Canada, 2011; 81–90.
5. Sharpnack J, Rinaldo A, Singh A. Changepoint detection over graphs with the spectral scan statistic. *arXiv preprint arXiv:1206.0773* 2012.
6. Sharpnack J, Rinaldo A, Singh A. Sparsistency of the edge lasso over graphs. *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, La Palma, Canary Islands, 2012; 1028–1036.
7. Speakman S, Zhang Y, Neill D.B. Dynamic pattern detection with temporal consistency and connectivity constraints. *2013 IEEE 13th International Conference on Data Mining (ICDM)*, IEEE, Dallas, Texas, 2013; 697–706.
8. Kirchhoff K, Bilmes JA. Submodularity for data selection in machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, A Meeting of SIGDAT, A Special Interest Group of the ACL*, Doha, Qatar, 2014; 131–141.
9. Lin H, Bilmes J. Multi-document summarization via budgeted maximization of submodular functions. *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, California, USA, 2010; 912–920.

DOI: 10.1002/cpe

10. Lin H, Bilmes J. A class of submodular functions for document summarization. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Association for Computational Linguistics, Stroudsburg, USA, 2011; 510–520.
11. Krause A, Guestrin C. Near-optimal observation selection using submodular functions. *AAAI*, Vol. 7, Vancouver, Canada, 2007; 1650–1654.
12. Nemhauser GL, Wolsey LA, Fisher ML. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming* 1978; **14**(1):265–294.
13. Calinescu G, Chekuri C, Pál M, Vondrák J. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing* 2011; **40**(6):1740–1766.
14. Buchbinder N, Feldman M, Naor J, Schwartz R. A tight linear time (1/2)-approximation for unconstrained submodular maximization. *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, California, USA, 2012; 649–658.
15. Pan X, Jegelka S, Gonzalez JE, Bradley JK, Jordan MI. Parallel double greedy submodular maximization. *Advances in Neural Information Processing Systems*, Montreal, Canada, 2014; 118–126.
16. Berk RH, Jones DH. Goodness-of-fit test statistics that dominate the Kolmogorov statistics. *Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete* 1979; **47**(1):47–59.
17. Chen F, Neill DB. Human rights event detection from heterogeneous social media graphs. *Big Data* 2015; **3**(1): 34–40.
18. Chen F, Neill DB. Non-parametric scan statistics for disease outbreak detection on twitter. *Online Journal of Public Health Informatics* 2014; **6**(1):e155.
19. Li J, Siegmund D. Higher criticism: p-values and criticism. *The Annals of Statistics* 2015; **43**(3):1323–1350.
20. Bach F. Learning with submodular functions: a convex optimization perspective. *arXiv preprint arXiv:1111.6453* 2011.
21. Lovász L. Submodular functions and convexity. In *Mathematical Programming the State of the Art*. Springer: Berlin Heidelberg, 1983; 235–257.
22. Iyer R, Bilmes J. Algorithms for approximate minimization of the difference between submodular functions, with applications. *arXiv preprint arXiv:1207.0560* 2012.
23. Mirzasoleiman B, Karbasi A, Sarkar R, Krause A. Distributed submodular maximization: identifying representative elements in massive data. *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2013; 2049–2057.
24. Rozenshtein P, Anagnostopoulos A, Gionis A, Tatti N. Event detection in activity networks. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM, New York City, USA, 2014; 1176–1185.
25. Ostfeld A, Uber JG, Salomons E, Berry JW, Hart WE, Phillips CA, Watson J-P, Dorini G, Jonkergouw P, Kapelan Z, Di Pierro F, Khu S, Savic D, Eliades D, Polycarpou M, Ghimire S, Barkdoll B, Gueli R, Huang J, McBean E, James W, Krause A, Leskovec J, Isovitsch S, Xu J, Guestrin C, VanBriesen J, Small M, Fischbeck P, Preis A, Propato M, Piller O, Trachtman G, Wu Z, Walski T. The battle of the water sensor networks (BWSN): a design challenge for engineers and algorithms. *Journal of Water Resources Planning and Management* 2008; **134**(6):556–568.
26. Speakman S, McFowland Iii E, Neill DB. Scalable detection of anomalous patterns with connectivity constraints. *Journal of Computational and Graphical Statistics* 2015:1014–1033.
27. Li J, Wen J, Tai Z, Zhang R, Yu W. Bursty event detection from microblog: a distributed and incremental approach. *Concurrency and Computation: Practice and Experience* 2015. DOI: 10.1002/cpe.3657.
28. Chen F, Zhou B, Ravi SS, Ramakrishnan N. Fast non-parametric subgraph scan for anomalous pattern detection on graphs. *Technical Report* 2016.