Journal of Statistical Mechanics: Theory and Experiment
An IOP and SISSA journal

PAPER: Interdisciplinary statistical mechanics

Core influence mechanism on vertex-cover problem through leaf-removal-core breaking

Xiangnan Feng[1,2,3], Wei Wei[1,2,3,4], Xing Li[1,2,3] and Zhiming Zheng[1,2,3,4]

[1] School of Mathematics and Systems Science, Beihang University, Beijing, People's Republic of China
[2] Key Laboratory of Mathematics Informatics Behavioral Semantics, Ministry of Education, People's Republic of China
[3] Peng Cheng Laboratory, Shenzhen, Guangdong, People's Republic of China
[4] Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University, Beijing, People's Republic of China
E-mail: weiw@buaa.edu.cn

Received 24 October 2019
Accepted for publication 8 May 2019
Published 1 July 2019

Online at stacks.iop.org/JSTAT/2019/073401
https://doi.org/10.1088/1742-5468/ab25e1

Abstract. Leaf-removal process has been widely researched and applied in many mathematical and physical fields to help understand the complex systems. A lot of problems including the minimum vertex-cover are deeply related to this process and the leaf-removal cores. In this paper, based on the structural features of the leaf-removal cores, a method named core influence is proposed to break the graphs into no-leaf-removal-core ones, which takes advantages of identifying some significant nodes by localized and greedy strategy. By decomposing the minimum vertex-cover problem into the leaf-removal core breaking process and maximal matching of the remaining graphs, it is proved that any minimum vertex-cover of the whole graph can be located into these two processes and the best boundary is achieved at the transition point. Compared with other node importance indices, the core influence method could break down the leaf-removal cores much faster and get the no-core graphs by removing fewer nodes from the graphs. Also, the vertex-cover numbers resulted from this method are lower than those of existing node importance measurements, and compared with the exact minimum vertex-cover numbers, this method performs appropriate accuracy and stability at different scales.

1742-5468/19/073401+20$33.00

This research provides a new localized greedy strategy to break the hard leaf-removal cores efficiently and heuristic methods could be constructed to promote the comprehension of the intrinsic hardness of NP problems.

Keywords: classical phase transitions, optimization over networks, random graphs, networks, typical-case computational complexity

Contents

1. Introduction

The minimum vertex-cover problem (vertex-cover) belongs to one of Karp's 21 NP-complete problems [1] and the six basic NP-complete problems [2, 3], which has a wide range of applications in the related real networks, such as immunization strategies in networks [4] and monitoring of internet traffic [5]. The minimum vertex-cover problem on the Erdös–Rényi random graph is hard-solving, and in the worst-case complexity, this problem is NP-hard for all connectivities. But when only regarding the typical-case complexity, there is an easy-hard transition at average degree equaling to Euler number e, which is caused by the typical emergence of the leaf-removal core especially when the average degree gets larger than e [6]. This hard core brings obstacles for solving the minimum vertex-cover, as correlations among the nodes/variables produce frustrations (even long-range frustrations [7]) in finding the optimal solutions, and how to decouple the leaf-removal core determines the effect of the solving strategy.

Generally speaking, leaf-removal mechanism plays important roles in understanding complex systems or graphs, such as detecting the hierarchical architectures for networks and performing the simplification for solving NP problems. All the leaves will be removed recursively until no such ones exist, and the *leaf-removal core* is got

J. Stat. Mech. (2019) 073401

for the remaining graph if it is not a null graph. Different leaf-removal strategies focus on different definitions of the *leaves*: for giant connected components, the leaves are degree-one nodes with corresponding edges, and random graphs with average degree larger than 1 typically have leaf-removal cores in the giant connected components in the viewpoint of the leaf-removal mechanism [8]; for k-XORSAT problem, the leaves are equations with variables only emerging once in the whole problem, and the leaf-removal core can result in the clustering of the all the solutions and one-step replica symmetric breaking with positive structural entropy [9]; for minimum vertex-cover problem, the leaves are degree-one nodes with their neighbors and related edges, and the existence of the leaf-removal core produces higher-order replica symmetric breaking and sets obstruction of understanding the solving complexity of NP-complete problems [10]. Also, the leaf-removal mechanism works as an important tool in so many such applications, such as solving the k-SAT problem [11] and MAS problem [12], the k-core [13] and k-shell [14] organizations of complex network.

In this paper, we propose a method to break the leaf-removal cores for minimum vertex-cover problem, namely strategy to delete some nodes such that the resulted graphs will contain no leaf-removal cores or the leaf-removal cores become empty. The complexity of the leaf-removal core can be measured by the cost to break it, but to break the leaf-removal core with the lowest cost (e.g. least number of nodes) always involves with NP problems, which makes the understanding of its structure complicated. The collective influence [15] was proposed to break the giant connected component, which can also be viewed as breaking the leaf-removal core in the above viewpoint, and it was studied to approximate the optimal percolation and reach appropriate accuracy.

The existing algorithms for solving minimum vertex-cover mainly focus on greedy searching strategy and heuristic strategy. In statistical mechanics, some algorithms based on the solution space structures were studied and good approximations were achieved in a heuristic way, such as the survey propagation algorithm [16] and the MBEA algorithm [17]. These algorithms took advantage of the backbones and clustering of the solutions to improve the solving efficiency. In this paper, it is aimed to break the leaf-removal core of the minimum vertex-cover problem in a greedy way, and an order for all the nodes will be investigated to break the leaf-removal core as fast as possible. Then, when some nodes are deleted to break the leaf-removal core, for the remaining graphs which are the subgraphs induced by the remained nodes, it enters into the easily-solving phase and some simple coverage strategies can be used to obtain the approximated optimal solutions.

2. Leaf-removal for minimum vertex-cover

2.1. The leaf-removal process

As mentioned before, there are different definitions on the leaves in the graphs. In this paper, the leaves in the graph stand for the nodes of which the degree is one and their neighbors, namely, removing the leaves means to remove the degree-one nodes and their neighbors. At the same time, all the links connected to these nodes are removed from the original graph.

J. Stat. Mech. (2019) 073401

Usually, when leaves are removed from the graph, new leaves will appear and removing the new leaves may bring more new leaves. The leaf-removal is a recursive process, and keeps removing leaves in the new graph until there is no leaf in the remaining graph. Clearly the remaining graph is composed of some isolated nodes and several relatively densely connected cores (the *leaf-removal cores*), and there is no node in the cores whose degree is one. An example is shown in figure 1 to perform a complete leaf-removal process and identify the leaf-removal core.

2.2. Greedy strategy for the coverage by breaking the leaf-removal core

For an undirected graph $G = (V, E)$ containing N nodes and M edges where V is the vertex set and E is the edge set, a set of nodes $C(G)$ in G is called to be a cover of G if for every edge $(i, j) \in E$, there exists node $i \in C(G)$ or $j \in C(G)$. The minimum vertex-cover, denoted as $C_m(G)$, is defined as:

$$C_m(G) = \arg\min_{C(G)} |C(G)|, \tag{1}$$

where $|C(G)|$ is the number of nodes in the set $C(G)$.

Let $\mathbf{n} = (n_1, n_2, \ldots, n_N)$ be the indices array of graph G where $n_i = 0$ indicates that node i and all the edges connected to it are removed from the graph and otherwise $n_i = 1$. In this way the graph induced by the remaining nodes i with $n_i = 1$ is denoted as $G(\mathbf{n})$, and the corresponding vertex-cover and minimum vertex-cover are denoted as $C(G(\mathbf{n}))$ and $C_m(G(\mathbf{n}))$. The number of remaining nodes is $N(\mathbf{n}) = \sum_{i=1}^{N} n_i$ and the vertex set is $V(\mathbf{n}) = \{i \in V | n_i = 1\}$.

When removing nodes one by one from graph G, the No-leaf-removal core might grow in the process. Yet at a certain step the graph will be turned into a no-leaf-removal-core graph, especially when the graph is close to an empty one. Assuming by a specific method, the nodes are deleted in a certain order, and the indices arrays are $\mathbf{n}_0 = (1, 1, \ldots, 1)$, $\mathbf{n}_1$, $\mathbf{n}_2$, ..., $\mathbf{n}_N = (0, 0, \ldots, 0)$. At a certain step t, the graph turns into a no-core graph, namely $G(\mathbf{n}_t)$, and contains no leaf-removal cores. For all $t \leqslant j \leqslant N$, $G(\mathbf{n}_j)$ is no-core graph, and it is not for all $j < t$. t is called the transition point.

By the König's theorem, the remaining graph $G(\mathbf{n}_j)$ for $t \leqslant j \leqslant N$ has no leaf-removal core and is a König–Egerváry graph, its minimum coverage number is equal to its maximal matching number $M(G(\mathbf{n}_j))$ [18], and finding the maximal matching for a graph is an easily-solving problem. If all the deleted nodes i (namely with $n_i = 0$ in $\mathbf{n}_j$ where $t \leqslant j \leqslant N$) are covered, a greedy strategy for approximation of the minimum coverage can be achieved: for any vertex-cover $C(G)$ of graph G, there exists a removing order which produces arrays $\mathbf{n}_0$, $\mathbf{n}_1$, ..., $\mathbf{n}_N$ with transition point t, and at the step t the cover of the graph G could be decomposed as:

$$C(G) = (V \setminus V(\mathbf{n}_t)) \cup C_m(G(\mathbf{n}_t)), \tag{2}$$

and the cover number is:

$$|C(G)| = t + M(G(\mathbf{n}_t)). \tag{3}$$

An example of getting a cover of a graph through this strategy, which changes the graph into a no-core one, is presented in figure 2. The maximal matching can be solved by an algorithm within polynomial time for general graphs [19], and the main effort for

J. Stat. Mech. (2019) 073401

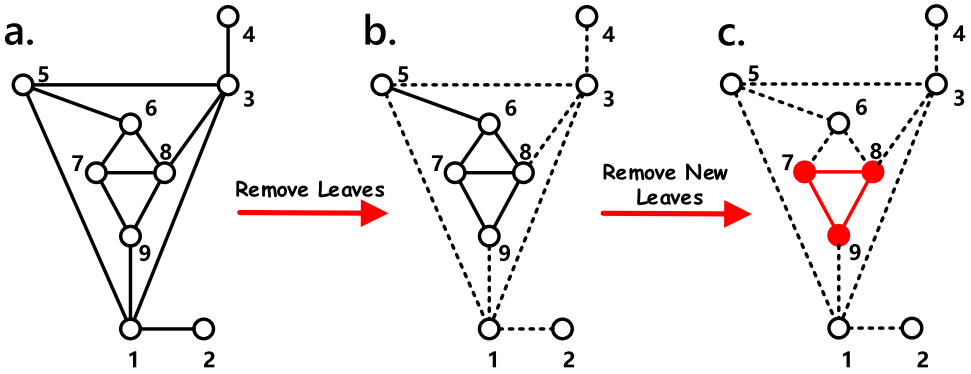

Figure 1. The leaf-removal process of the graph in (a). Nodes 1 and 2 form one leaf, and nodes 3 and 4 form the other. The graph in (b) is got with the two leaves removed, and in the remaining graph, a new leaf with nodes 5 and 6 is produced. Removing this new leaf, we could get the graph in (c). There is no more one-degree node and the remaining triangle composed by nodes 7, 8 and 9 is the leaf-removal core, which is marked as red in (c). The dotted lines denote the deleted edges when nodes are removed.

this greedy strategy relies on determining an optimal deleting order to make the transition point t as small as possible. The deleting strategy aims to break the leaf-removal core with the lowest cost, and how to distinguish the important vertices and recognize the polarized vertices as the covered backbones is the core difficulty.

2.3. Relations between the core-breaking and optimal solutions

Proposition. *For nodes arrays* $\mathbf{n}_i$ *and* $\mathbf{n}_j$ *where* $j \geqslant i \geqslant t$*, the coverage numbers for the whole graph* G *by the two core-breaking ways satisfy* $|C_m(G(\mathbf{n}_j))| + j \geqslant |C_m(G(\mathbf{n}_i))| + i$.

Proof. When one more node is deleted from $G(\mathbf{n}_i)$ and the graph becomes $G(\mathbf{n}_{i+1})$ with $N(\mathbf{n}_{i+1}) + 1 = N(\mathbf{n}_i)$. Yet since one node owns to at most one match, the maximal matching number of $G(\mathbf{n}_{i+1})$ will decrease by one or remain the same as $G(\mathbf{n}_i)$, namely $M(G(\mathbf{n}_{i+1})) + 1 \geqslant M(G(\mathbf{n}_i)) \geqslant M(G(\mathbf{n}_{i+1}))$. In this way $|C_m(G(\mathbf{n}_{i+1}))| + i + 1 \geqslant |C_m(G(\mathbf{n}_i))| + i$, and thus $|C_m(G(\mathbf{n}_j))| + j \geqslant |C_m(G(\mathbf{n}_i))| + i$ for all $j \geqslant i \geqslant t$. □

This proposition shows that the best step for the approximation of minimum coverage under the above core-breaking strategy occurs at the transition point t, i.e. the first time that leaf-removal cores disappear, and so our algorithm greedily deletes the least number of nodes to produce a no-core graph with simple minimum coverage. It is evident that the obtained coverage is not necessary to be a minimum one, but the following theorem illustrates that all the minimum vertex-cover can be found in such a core-breaking way.

Theorem. *For any graph* G *and its minimum vertex-cover* $C_m(G)$*, there exists a deleting method to produce nodes arrays* $\mathbf{n}_0$, $\mathbf{n}_1$, …, $\mathbf{n}_N$ *and transition point* t*, such that this minimum vertex-cover could be represented as:*

$$C_m(G) = (V \setminus V(\mathbf{n}_t)) \cup C_m(G(\mathbf{n}_t)), \tag{4}$$

and the minimal cover number is:

$$|C_m(G)| = t + M(G(\mathbf{n}_t)). \tag{5}$$

Proof. Without loss of generality, assume that the minimum vertex-cover $C_m(G)$ has $1, \cdots, s$ as covered nodes with $s+1, \cdots, N$ as uncovered nodes of graph G. For any core-breaking strategy $\mathbf{n}_0$, $\mathbf{n}_1$, ..., $\mathbf{n}_t$, we have $t < s$. And, if we remove the nodes according to the natural order $1, \cdots, s$, i.e. ensuring all the removed nodes are covered ones, after the core-breaking strategy $\mathbf{n}_0$, $\mathbf{n}_1$, ..., $\mathbf{n}_s$, the remaining graph $G(\mathbf{n}_s)$ should be a null graph composed of isolated nodes. The original graph has leaf-removal core and remaining graph $G(\mathbf{n}_s)$ has no core, then there must exist a certain number t at which the core is firstly broken, and the nodes in the minimum vertex-cover evidently have a decomposition representation $C_m(G) = (V \setminus V(\mathbf{n}_t)) \cup C_m(G(\mathbf{n}_t))$ with $|C_m(G)| = t + M(G(\mathbf{n}_t))$. □

As for a no-core graph, each leaf contains a covered node with the other uncovered, and the number of covered nodes must be fewer or equal to the number of uncovered ones. Thus generally after deleting at least $\max\{0, N - (N-s) * 2\} = \max\{0, 2s - N\}$ covered nodes, it will produce a no-core graph which satisfies $t \geqslant \max\{0, 2s - N\}$. Besides, by deleting one of neighbored covered nodes for all such pairs, the remaining graph has each covered node with no covered neighbors, and it is almost a no-core graph with high probability; when the original graph has no triangles, this deleting strategy only deletes at most $s/2$ nodes and $t \leqslant s/2$ with high probability to break the leaf-removal core. Usually for one minimum vertex-cover of a graph, there are more than one corresponding deleting methods. The deleting order of nodes for core-breaking can also be arranged in other greedy strategy, such as deleting the nodes with the biggest degrees, in the orders of different centralities or node importance measurements.

When a graph has leaf-removal core, with high probability it is not a König–Egerváry graph and the relation between the maximal matching and the minimum vertex-cover is quite complicated. Though a minimum vertex-cover can be found in such a core-breaking way, the fast way to break the leaf-removal core does not always correspond to achieving a minimum vertex-cover. As the problem of breaking the giant connected component fastest is NP hard, the optimal solution of breaking the leaf-removal core is also a hard problem, which should be related to but different from the minimum vertex-cover problem. Here, it is safe to say that the minimum vertex-cover does not always correspond to the fastest method to change the graph into a no-core one. Yet, it is meaningful to study the process of deleting nodes since it provides us a new perspective to understand the complicated organization of the leaf-removal core and study the minimum vertex-cover problem.

J. Stat. Mech. (2019) 073401

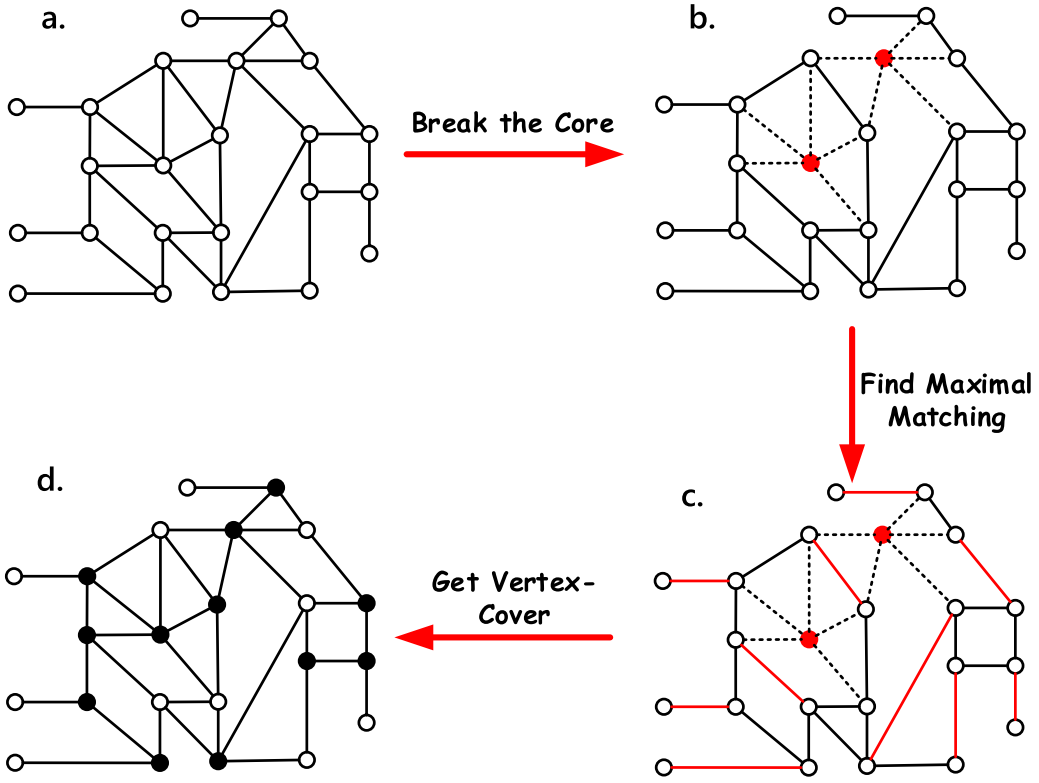

Figure 2. The procedures to get a vertex-cover of graph in (a) by the strategy of removing the graph into a no-core one. (a) owns one leaf-removal core. By the algorithm proposed in the following section, the red nodes in (b) are removed and the remaining graph is a no-core graph. The remaining graph satisfies the property that the maximal matching number equals to the minimum cover number. In (c), one maximal match is marked by red edges. Combing the removing nodes in (b) and results of the maximal matching in (c), the vertex-cover could be got, denoted by black nodes in (d). The dotted lines denote the removed links when nodes are removed. The maximal matching number in (c) is 10 and thus the cover number of graph in (a) got by this strategy is 12, which exactly equals to the minimum cover number of this graph.

3. Delete a graph into the no-core graph

In this section we will discuss how to turn graphs into no-core ones by deleting nodes with the lowest cost. As we said before, this problem will concern a lot of applications in real-world networks.

The leaf-removal cores will appear when the leaf-removing process is completed. For node i, we define c_i in $\mathbf{c} = (c_1, c_2, \ldots, c_N)$ to indicate whether node i belongs to the leaf-removal core:

$$c_i = \begin{cases} 1 & \text{if node } i \text{ is in a leaf-removal core,} \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

Obviously, if the degree of node i is one or node i is connected to a degree-one node in the leaf-removal process, we have $c_i = 0$.

The problem of breaking the leaf-removal cores could be described in the language of optimization, and it could be regarded as finding the $\mathbf{n} = (n_1, n_2, \ldots, n_N)$ with its corresponding $\mathbf{c}$ satisfying:

$$\mathbf{n} = \arg\max_{\mathbf{n}} \ N(\mathbf{n}), \quad \text{subject to } \sum_{i=1}^{N} c_i n_i = 0. \tag{7}$$

Consider two connected nodes i and j in the original graphs. It is clear that removing node j may affect the value of c_i, yet this relation is complex. There are four cases for c_i and c_j: (1) $c_i = 0$ and $c_j = 0$, (2) $c_i = 0$ and $c_j = 1$, (3) $c_i = 1$ and $c_j = 0$, (4) $c_i = 1$ and $c_j = 1$. For case (1), node i and j do not belong to the cores, and in our following proposed model this situation will not affect the results. For case (2) and (3), the analysis is the same, and only case (3) is discussed here. For case (3), the node j has been removed in one of the leaf-removal steps. Clearly, the degree of node j should not be one and it is connected to a degree-one node, otherwise node i will be deleted. Thus the removal of node j will not affect the value of c_i. Case (4) is complicated: removing node j may produce new leaves and more nodes including node i may be affected in the original cores.

Let $c_{i\to j}$ be the indicator of node i belonging to the core with node j removed. Consider all the nodes that are connected to i. The value of c_i with j removed from the graph is determined by the status of all the neighbors of i. For a network with tree-like structure locally, this relation could be formulated by:

$$c_{i\to j} = c_i n_i [1 - \prod_{h\in\partial i, h\neq j} (1 - c_{h\to i})], \tag{8}$$

where the ∂i is the set of neighbors of node i. This system takes into account the first two cases we have mentioned above: if node i does not belong to any of the final leaf-removal cores ($c_i = 0$), $c_{i\to j}$ will keep to be 0 no matter whether node j is removed or not. Obviously $c_{i\to j} = 0$ if $n_i = 0$, namely node i is removed from the graph. By the cavity method, it is assumed that the neighbors of node i are independent with the absence of i in the locally tree-like graphs. The influence of i to j, the $c_{i\to j}$, corresponds to the case that at least one of node i's other neighbors performs influence to i, and the independence of i's neighbors except j makes the product $\prod_{h\in\partial i, h\neq j}(1 - c_{h\to i})$ on the right side of equation (8). In statistical mechanics, this independence only works in a replica symmetry phase of the system. The leaf-removal core of a graph is a subgraph which involves subtle and complex structures. To break it down, finding out the crucial nodes which have large spreading influence on the core structure is an intuitive method, and removing these nodes will bring collapse of the core. It could be understood as the ability of nodes in spreading information confined on the leaf-removal cores, which is performed heuristically.

In this system, the set $\{c_{i\to j} = 0 | \forall i, j \in V\}$ will always be one solution. It is worth noting that this solution will not always correspond to a no-core graph. For example, a weakly connected network, like a cycle, could be very fragile and removing any node will turn the graph into a no-core one, although in the original graph all the nodes belong to the leaf-removal core. In the viewpoint of statistical mechanics, the stability of the solution $\{c_{i\to j} = 0 | \forall i, j \in V\}$ corresponds to the replica symmetry phase of

equation (8), and the breaking of the stability will correspond to the emergence of the hard core. So in this paper we will regard that the stable solution $\{c_{i\to j}=0|\forall i,j\in V\}$ corresponds to an no-core graph.

With knowledge in dynamic system, the solution will be stable if the largest eigenvalue of the linear operation R is smaller than one [20]. R is a matrix with $2M$ rows and $2M$ columns, and each row or column stands for an edge with one direction $i\to j$. Since edge between node i and node j has two directions $i\to j$ and $j\to i$, this matrix will take into account all the directions and defined as:

$$R_{w\to x,y\to z}=\frac{\partial c_{w\to x}}{\partial c_{y\to z}}|_{c_{y\to z}=0}. \tag{9}$$

Let $\lambda(\mathbf{c},\mathbf{n})$ be the largest eigenvalue of matrix R, and clearly this value is determined by $\mathbf{n}$ and $\mathbf{c}$. The stability of the solution corresponding to no-core graph is determined by $\lambda(\mathbf{c},\mathbf{n})<1$.

It could be found that, in a graph with tree-like structure locally, this matrix R is calculated by the non-backtracking matrix B [21, 22]:

$$R_{w\to x,y\to z}=c_y n_y B_{w\to x,y\to z}, \tag{10}$$

where B is defined as:

$$B_{w\to x,y\to z}=\begin{cases}1 & \text{if } x=y, w\neq z,\\ 0 & \text{otherwise.}\end{cases} \tag{11}$$

The non-backtracking has received a lot of attention recently and been used to identify the non-backtracking walks on graphs. According to the Perron–Frobenius theorem [23], since all the entries in B are non-negative, its largest eigenvalue and all the entries of the corresponding eigenvector are all positive. Krzakala *et al* showed that the spectral method based on the non-backtracking method, namely the eigenvector corresponding to the second largest eigenvalue, could be applied to divide communities in graphs and reach the detectability threshold [24]. Neumann *et al* used this matrix to overcome the pathology of nodes near the high-degree nodes or hubs when eigenvector centrality is applied in some graphs to evaluate the influence of nodes [25]. Hernan *et al* applied the non-backtracking matrix to the optimal percolation problem and find an index to reduce the size of giant component [26, 27], which is crucial in influence optimization and immunization [15].

As nodes get deleted in graphs, the largest eigenvalue $\lambda(\mathbf{c},\mathbf{n})$ of the modified non-backtracking matrix R changes. When a node is deleted or the node is no longer in the cores, the corresponding entries in R turn into zero. According to the Perron–Frobenius theorem, the largest eigenvalue will decrease. The problem of deleting the graph into a no-core one is then equivalent to finding the fastest way to reduce the largest eigenvalue to be less than one, namely this problem could be represented as:

$$\min_{\mathbf{n}}|N-N(\mathbf{n})|,\quad \text{subject to } \lambda(\mathbf{n},\mathbf{c})<1. \tag{12}$$

It is impossible to solve it directly, since $\mathbf{n}$ and $\mathbf{c}$ is sophisticatedly tangled and there is no formula of $\lambda(\mathbf{n},\mathbf{c})$ in the form of $\mathbf{n}$ and $\mathbf{c}$. To overcome this, we will apply the power method [28], a classical numerical technique, to approximate the largest eigenvalue.

J. Stat. Mech. (2019) 073401

Let $\mathbf{v}_0$ be a vector with nonzero projection on the direction of the eigenvector corresponding to $\lambda(\mathbf{n},\mathbf{c})$ and $\mathbf{v}_l(\mathbf{n},\mathbf{c})$ be the vector after l times iterations by R:

$$\mathbf{v}_l(\mathbf{n},\mathbf{c}) = R^l\,\mathbf{v}_0. \tag{13}$$

By the power method, we have:

$$\lambda(\mathbf{n},\mathbf{c}) = \lim_{l\to\infty}\lambda_l(\mathbf{n},\mathbf{c}) = \lim_{l\to\infty}\left(\frac{|\mathbf{v}_l(\mathbf{n},\mathbf{c})|}{|\mathbf{v}_0|}\right)^{\frac{1}{l}}, \tag{14}$$

where for $\mathbf{v}_l(\mathbf{n},\mathbf{c})$, and:

$$|\mathbf{v}_l(\mathbf{n},\mathbf{c})|^2 = \langle\mathbf{v}_l(\mathbf{n},\mathbf{c})|\mathbf{v}_l(\mathbf{n},\mathbf{c})\rangle = \langle\mathbf{v}_0|(R^l)^\dagger\,R^l\,|\mathbf{v}_0\rangle. \tag{15}$$

Firstly, for $l=1$, the approximated eigenvector corresponding to $\lambda(\mathbf{n},\mathbf{c})$ is:

$$|\mathbf{v}_1(\mathbf{n},\mathbf{c})\rangle = R\,|\mathbf{v}_0\rangle. \tag{16}$$

To further calculate this, the matrix R could be calculated by:

$$R_{w\to x,y\to z} = c_y n_y A_{wx}A_{yz}\delta_{xy}(1-\delta_{wz}), \tag{17}$$

where the A_{wx} and A_{yz} are entries of the adjacency matrix and δ_{xy} and δ_{wz} are the Kronecker symbols with:

$$\delta_{ij} = \begin{cases} 1 & \text{if } i=j,\\ 0 & \text{otherwise.}\end{cases} \tag{18}$$

Here, $n_y=1$ and $c_y=1$ guarantee that node y is not deleted from the graph and belongs to the core. $A_{wx}=1$, $A_{yz}=1$, and $\delta_{xy}=1$ ensure that there is a path from node w to node z through x (or y) and $\delta_{wz}=1$ ensures the non-backtracking property.

For simplicity, let s_i stand for the state of node i and q_i the excess degree, namely:

$$s_i \equiv c_i n_i, \quad q_i \equiv d_i - 1. \tag{19}$$

Choosing the $2M$-dimension vector $|\mathbf{v}_0\rangle = |1\rangle$, we could get the left vector:

$$_{w\to x}\langle\mathbf{v}_1(\mathbf{n},\mathbf{c})| =_{w\to x} \langle\mathbf{v}_0|\,R^\dagger = \sum_{y\to z}{}_{y\to z}\langle\mathbf{v}_0|\,R_{y\to z,w\to x} = s_w A_{wx} q_w, \tag{20}$$

and the right vector:

$$|\mathbf{v}_1(\mathbf{n},\mathbf{c})\rangle_{w\to x} = R\,|\mathbf{v}_0\rangle_{w\to x} = \sum_{y\to z} R_{w\to x,y\to z}\,|\mathbf{v}_0\rangle_{y\to z} = s_x A_{wx} q_x. \tag{21}$$

The norm could be presented as:

$$\begin{aligned}|\mathbf{v}_1(\mathbf{n},\mathbf{c})|^2 &= \sum_{w\to x}{}_{w\to x}\langle\mathbf{v}_1(\mathbf{n},\mathbf{c})|\mathbf{v}_1(\mathbf{n},\mathbf{c})\rangle_{w\to x}\\ &= \sum_{w\to x} A_{wx}q_w q_x s_w s_x.\end{aligned} \tag{22}$$

https://doi.org/10.1088/1742-5468/ab25e1

J. Stat. Mech. (2019) 073401

Since $|\mathbf{v}_0|^2 = \sum_{i=1}^{N} d_i = 2M$, we have:

$$\lambda_1(\mathbf{n}, \mathbf{c}) = \left(\frac{1}{2M} \sum_{w \to x} A_{wx} q_w q_x s_w s_x \right)^{\frac{1}{2}}. \tag{23}$$

From the equation above, it could be observed that the approximation $\lambda_1(\mathbf{n}, \mathbf{c})$ to the largest eigenvalue at order 1 involves all the edges in the graphs. The non-backtracking requires that if there exist self-edges, they should not be taken into account in the approximation.

For $l = 2$, the left vector ${}_{w\to x}\langle \mathbf{v}_2(\mathbf{n}, \mathbf{c})|$ could be derived directly from the same process:

$$\begin{aligned} {}_{w\to x}\langle \mathbf{v}_2(\mathbf{n}, \mathbf{c})| &= \sum_{y\to z} {}_{y\to z}\langle \mathbf{v}_1| \, R_{y\to z, w\to x} \\ &= s_w A_{wx} \sum_{i}^{N} A_{iw} s_i q_i (1 - \delta_{xi}), \end{aligned} \tag{24}$$

and the same to the right vector:

$$\begin{aligned} |\mathbf{v}_2(\mathbf{n}, \mathbf{c})\rangle_{w\to x} &= \sum_{y\to z} R_{w\to x, y\to z} \, |\mathbf{v}_1\rangle_{y\to z} \\ &= s_x A_{wx} \sum_{i}^{N} A_{xi} s_i q_i (1 - \delta_{wi}). \end{aligned} \tag{25}$$

So the norm is:

$$|\mathbf{v}_2(\mathbf{n}, \mathbf{c})|^2 = \sum_{w\to x, y\to z} (A_{wx} A_{xy} A_{yz} (1 - \delta_{wy})(1 - \delta_{xz}) q_w q_z s_w s_x s_y s_z), \tag{26}$$

and the approximation to the largest eigenvalue $\lambda_2(\mathbf{n}, \mathbf{c})$ is:

$$\begin{aligned} \lambda_2(\mathbf{n}, \mathbf{c}) = \Bigg(\frac{1}{2M} \sum_{w\to x, y\to z} & (A_{wx} A_{xy} A_{yz} \\ & \times (1 - \delta_{wy})(1 - \delta_{xz}) q_w q_z s_w s_x s_y s_z) \Bigg)^{\frac{1}{4}}. \end{aligned} \tag{27}$$

By the same derivation, the approximation at order 3 to the largest eigenvalue $\lambda_3(\mathbf{n}, \mathbf{c})$ is:

$$\begin{aligned} \lambda_3(\mathbf{n}, \mathbf{c}) = \Bigg(\frac{1}{2M} \sum_{u\to v, w\to x, y\to z} & (A_{uv} A_{vw} A_{wx} A_{xy} A_{yz} \\ & \times (1 - \delta_{uw})(1 - \delta_{vx})(1 - \delta_{wy})(1 - \delta_{xz}) \\ & \times q_u q_z s_u s_v s_w s_x s_y s_z) \Bigg)^{\frac{1}{6}}. \end{aligned} \tag{28}$$

J. Stat. Mech. (2019) 073401

And for any $l > 0$, we have:

$$\lambda_l(\mathbf{n},\mathbf{c}) = \Bigg(\frac{1}{2M}\sum_{x_1,x_2,\ldots,x_{2l}} (A_{x_1x_2}A_{x_2x_3}\ldots A_{x_{2l-1}x_{2l}} \times (1-\delta_{x_1x_3})(1-\delta_{x_2x_4})\ldots(1-\delta_{x_{2l-2}x_{2l}}) \times q_{x_1}q_{x_{2l}}s_{x_1}s_{x_2}\ldots s_{x_{2l}})\Bigg)^{\frac{1}{2l}}. \tag{29}$$

As we could see, the $l = 2$ approximation involves all the paths of length 3 and all the pathes satisfy the non-backtracking property; the $l = 3$ approximation requires all the paths of length 5 to be searched. It could be inferred that in the order l approximation, the paths with length $2l-1$ are needed to be considered. It is worth noting that in the approximation with order larger than 2, cycles are allowed in the path when it meets the non-backtracking requirement, and the path could walk by the nodes that have been already visited.

The formula of the approximation to the largest eigenvalue at order l could be written in this way:

$$\lambda_l(\mathbf{n},\mathbf{c}) = \left(\frac{1}{2M}\sum_{i=1}^{N} q_i \sum_{\mathbf{path}\in P_{2l-1}(i,j)} \left[\prod_{k\in\mathbf{path}} s_k\right] q_j\right)^{\frac{1}{2l}}, \tag{30}$$

where the $P_{2l-1}(i,j)$ contains all the non-backtracking paths whose length is $2l-1$.

The formula above involves all the paths in the graph with length $2l-1$. Yet this calculation could be generalized to the situation of length $2l$. By power method, the largest eigenvalue could also approximated by:

$$\lambda(\mathbf{n},\mathbf{c}) = \lim_{l\to\infty}\left(\frac{\langle \mathbf{v}_l(\mathbf{n},\mathbf{c})|\, R\,|\mathbf{v}_l(\mathbf{n},\mathbf{c})\rangle}{\langle \mathbf{v}_0|\mathbf{v}_0\rangle}\right)^{\frac{1}{2l+1}}. \tag{31}$$

For example, when $l = 1$, we have:

$$\langle \mathbf{v}_1(\mathbf{n},\mathbf{c})|\, R\,|\mathbf{v}_1(\mathbf{n},\mathbf{c})\rangle = \sum_{x\to y, y\to z} A_{xy}A_{yz}(1-\delta_{xz})q_xq_zs_xs_ys_z. \tag{32}$$

Thus, for any l, according to the same derivation introduced above, the largest eigenvalue could be approximated by:

$$\lambda'_l(\mathbf{n},\mathbf{c}) = \left(\frac{1}{2M}\sum_{i=1}^{N} q_i \sum_{\mathbf{path}\in P_{2l}(i,j)} \left[\prod_{k\in\mathbf{path}} s_k\right] q_j\right)^{\frac{1}{2l+1}}. \tag{33}$$

From all the calculation above, when paths of length l are considered in the approximation, ignoring the constant $1/2M$, the leading part of the approximation formula is:

$$H_l = \sum_{i=1}^{N} q_i \sum_{\mathbf{path}\in P_l(i,j)} \left[\prod_{k\in\mathbf{path}} s_k\right] q_j. \tag{34}$$

To reduce the largest eigenvalue quickly, a direct idea is to find the nodes which could reduce the H_l fastest. To illustrate this, we define the *core influence of node i*, $H_l(i)$ to be:

J. Stat. Mech. (2019) 073401

$$H_l(i) = q_i \sum_{\mathbf{path} \in P_l(i,j)} \left[\prod_{k \in \mathbf{path}} s_k \right] q_j. \tag{35}$$

This value will be the indicator in deleting nodes to turn the graph into a no-core one. First calculate $H_l(i)$ for all the nodes, and delete the node who owns the highest $H_l(i)$ value and all the edges connected to this node. Repeat this process until the remaining graph is a no-core one. The detailed algorithm is presented in algorithm 1. After the graph is turned into a no-core one, find the maximal matching for the remaining graph. By unifying the deleted nodes and matched nodes, a vertex-cover could be got.

Algorithm 1. The process of deleting the graph into a no-core one.

Input graph G;
While G is not a no-core graph, namely $\sum c_i \neq 0$ **do**
  For each node i, calculate c_i, q_i and the state value $s_i = c_i n_i$;
  For each node i, calculate all the core influence $H_l(i)$;
  Select node j with largest core influence value;
  Delete node j and all the edges linked to it from the graph;
  Update the graph G, set $n_j = 0$;
end while
Output Nodes that are deleted in each step, namely **n**.

The computational complexity of leaf-removal process is $O(N)$, and calculating the core influence takes $O(N \log N)$ time cost. In this way, to spot and remove the node with the highest core influence costs $O(N)$ time complexity. Thus, the total complexity of algorithm 1 is $O(N^2 \log N)$.

One problem in applying the algorithm 1 is how to update the state indicator values c_i. The leaf-removing is a fast process with time complexity $O(N)$, however, it will cost a lot to update them for every step. To overcome this, we propose to use a relatively coarse yet much faster strategy to update the indicator values. Before node i is removed from the graph, we search all the neighbors of i. Then neighbors with degree two, namely the nodes which will become leaves if i is removed, are specified. The state values s of these neighbors and nodes connected to them are set to be zero. A detailed algorithm is shown in algorithm 2. This will reduce the complexity to $O(N(\log N)^2)$. Since the structure of the core is sophisticated, removing one node in core may produce a long-lasting effect on the core structure, more nodes rather than neighbors will turn into leaves and be deleted from the core, and this method in algorithm 2 is only a local approximation to the complete update process. Better methods require a much deeper understanding to the core structure, which deserves further research.

Algorithm 2. The updating method of indicator values.

Input graph G, state values **c** and **n**, node i that is going to be deleted;
for Each neighbor j of node i **do**
  if $d_j = =2$, the two nodes connected to j is i and k **then**
    Set $c_j = 0$, $c_i = 0$, $c_k = 0$;
  end if
end for

J. Stat. Mech. (2019) 073401

Table 1. The percentages of differences between results of core influence method and exact minimum vertex-cover numbers against nodes numbers. The experiments are conducted on the ER graphs with 80, 100 and 120 nodes with average degrees from 3 to 7 and every result is the average of 30 graphs.

	Average degrees				
Nodes numbers	3 (%)	4 (%)	5 (%)	6 (%)	7 (%)
$N = 80$	0.21	0.71	1.33	1.63	1.38
$N = 100$	0.27	0.80	1.53	1.43	1.47
$N = 120$	0.05	0.67	1.28	1.33	1.53

4. Experiments

In this section some experiments are conducted to present the efficiency of several centrality methods of deleting nodes. We examine the steps required to delete the graphs into no-core ones by each centrality and compare the numbers of steps. Also, combining with the theorems mentioned above, the minimal cover numbers of all the methods are compared.

4.1. Results for leaf-removal cores breaking

We compare our Core influence method H_l with centralities including high degree centrality (DC) [29], k-core (KC) [13], betweenness centrality (BC) [30], closeness centrality (CC) [31, 32], collective influence (CI) [15], high degree adaptive (HDA), pagerank (PR) [33] and eigenvector centrality (EC) [34] on Erdös–Rényi (ER) graphs and scale-free (SF) graphs [35]. All these methods are applied to calculate how many nodes are required to be removed to turn the random graphs into no-core ones. Results are shown in figure 3.

The core influence method is applied at order $l = 1$ in equation (35). As we could see, the core influence method performs great efficiency in this problem. With the average degree increasing, the graphs get more denser, yet compared to other node importance indices the core influence method still keeps requiring fewer nodes to be deleted to turn the graphs into no-core ones. The core influence method with an approximated states updating method in algorithm 2 performs similar results. Results on SF graphs are better than that on the ER graphs, namely on SF graphs fewer nodes are required to be deleted to transform the graphs into no-core ones. That's because there exist nodes with very high degrees or hubs and deleting them will bring large changes on the topological structure. As a localized greedy method aiming at breaking the leaf-removal cores, the proposed core influence measurement works more efficiently on different topologies.

4.2. Results for minimal vertex-cover number

At the same time, the vertex-cover is determined by the deleted nodes and the remaining no-core graphs, and different node importance measurements are applied to delete the graphs into no-core ones and then by equations (4) and (5) the minimal vertex-cover

J. Stat. Mech. (2019) 073401

Table 2. The average numbers of nodes required to be deleted to turn the graphs into no-core ones and the corresponding vertex-cover numbers. The experiments are conducted on regular random graphs with average degrees ranging from 3 to 10. For each average degree value, 30 graphs are generated and each graph contains 1000 nodes. Results of CI, HDA, core influence H_l and H_l (approximated), the four best performed methods, are presented on the table.

	Average degrees							
Deleted nodes numbers	3	4	5	6	7	8	9	10
CI	257.8	337.5	423.0	470.8	505.3	535.5	565.0	593.8
HDA	243.9	337.3	384.6	443.4	481.2	513.4	557.0	581.9
H_l (approximated)	247.0	336.4	396.6	448.4	490.1	523.0	547.0	580.4
H_l	207.3	288.9	364.9	418.2	461.4	496.5	526.5	552.9
	Average degrees							
Vertex-cover numbers	3	4	5	6	7	8	9	10
CI	612.6	656.0	693.8	719.2	738.7	754.3	769.9	783.2
HDA	592.1	636.4	667.4	693.8	715.8	733.4	749.8	764.8
H_l (approximated)	609.4	653.0	685.8	712.5	733.1	750.8	764.7	778.9
H_l	602.1	643.3	681.1	707.9	729.5	747.2	761.9	775.4

numbers can be calculated. The specific process for getting the minimal vertex-cover by core influence is presented in algorithm 3. Results of the experiments are shown in figure 4.

Algorithm 3. The vertex-cover number by core influence.

Input graph G;
Get the deleted node array $\mathbf{n}$ by algorithm 1;
Get the maximal matching number $M(G(\mathbf{n}))$ for graph $G(\mathbf{n})$;
Output The vertex-cover number $N - N(\mathbf{n}) + M(G(\mathbf{n}))$.

As we could see, by the leaf-removal process, the coverage numbers are much smaller compared to other methods, especially when the average degrees are small. The core influence method proposed in the paper works better and could lower the vertex-cover numbers. When the average degree gets higher (than 6), the approximated method in algorithm 2 works similar to the HDA method and they both work better than other centralities, yet the complete strategy H_l still works better than HDA. These results perform that the solution by deleting the graphs into no-core ones is a new perspective in studying graph-related problems and has shown great potential in minimum vertex-covering problem.

Compared to the theoretical results by first-step replica symmetric breaking (1RSB) [16] or energy density of the belief propagation-guided decimation (BPD) algorithm [37] which is very close to the results predicted by 1RSB, when the average degrees are low, outcomes of the core influence method are close to the theoretical results; when the average degrees are higher, there are small gaps between results of H_l and the lower bound. More researches on the updating strategy are expected to improve the performance.

To further explore the efficiency of the core influence method, the cover numbers resulted from this method are compared with the exact results of minimum vertex-cover.

J. Stat. Mech. (2019) 073401

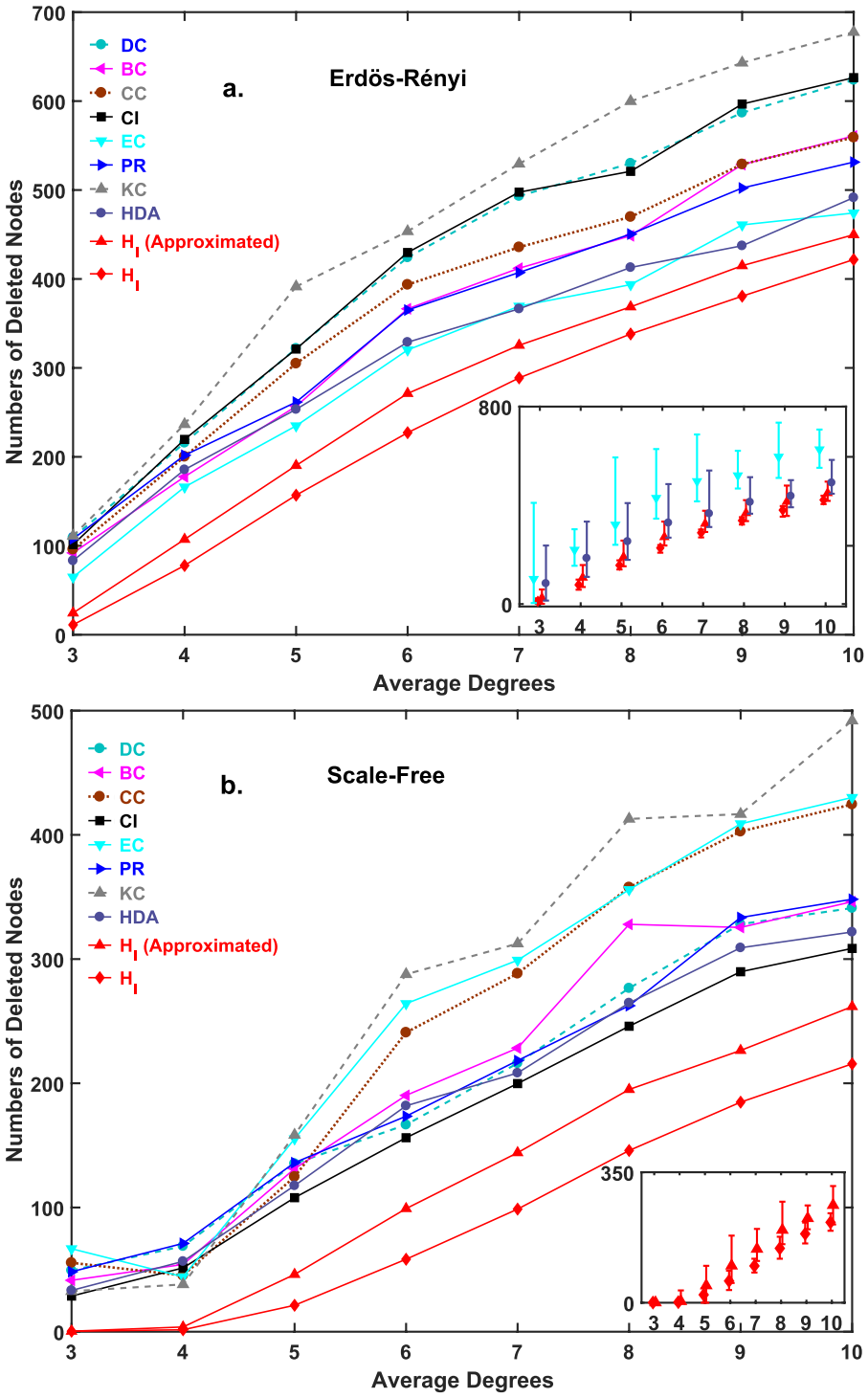

Figure 3. The average numbers of nodes required to be deleted to turn the graphs into no-core ones. The experiments are conducted on ER and SF graphs with average degrees ranging from 3 to 10. For each average degree value, 30 graphs are generated and each graph contains 1000 nodes. Results of DC, KC, BC, CC, CI, HDA, PR, EC and core influence H_l are drawn on the graph. (a) Results on ER graphs. The error bars of EC, HDA, H_l and approximated H_l (The four best methods) are plotted in the inside figure. (b) Results on scale-free graphs with $\gamma = 3$.

J. Stat. Mech. (2019) 073401

Table 3. The numbers of nodes required to be deleted to turn the GR-QC collaboration network into no-core one and the corresponding vertex-cover number. There are 5242 nodes and 14 496 edges in the network and results of DC, KC, BC, CC, CI, HDA, PR, EC and core influence H_l are shown.

	Deleted nodes numbers	Vertex-cover numbers
DC	4042	4220
BC	5240	5241
CC	4778	4956
CI	1563	2820
EC	5211	5220
PR	4577	4582
KC	3918	4213
HDA	1831	2795
H_l	**779**	**2785**
H_l (approximated)	**855**	**2787**

The differences between the results of core influence method and real C_m are calculated and the percentages of these differences against the total nodes number are shown in table 1. As we could see, the core influence performs well and as the nodes number increases, this method stays efficient. With constant nodes number, as the edges number increases, the gaps are getting a little larger. Yet the core influence method still keeps high efficiency and the cover numbers keep a small difference from the exact minimum vertex-cover numbers.

The algorithm on the ER and SF graphs performs well due to the existence of local structures. To examine the influence of local structure on the results, experiments on regular random graphs (RRG) are implemented. As shown in table 2 where results of the best four methods are presented, this algorithm still performs well in breaking the leaf-removal cores, while HDA performs best in finding the vertex-cover. When the graph is homogeneous and uniform, there are fewer local structural features and differences among nodes are little, and this core-breaking strategy by identifying crucial nodes seems to be an inefficient way in approximating the minimum vertex-cover.

4.3. Experiments on GR-QC collaboration network

Experiments are conducted on the general relativity and quantum cosmology (GR-QC) collaboration network [36]. This network is from the *arXiv* site and records the papers submitted to the GR-QC category from January 1993 to April 2003. There are 5242 nodes and 14 496 edges in the network. Each node represents an author, and if two authors cooperated and submitted a paper to the GR-QC category in *arXiv* together, there will be an undirected link between the corresponding nodes.

The leaf-removal process and core influence method are implemented on the network and results are shown in table 3. As the results show, the core influence method keeps the great efficiency and requires deleting less than 1000 nodes to get the no-core graph. The BC requires 5240 nodes to be deleted to turn the graph into no-core one, which has only 2 nodes less than the total nodes number. This is due to the appearance of isolated triangles in the leaf-removal process, which is common in scientific research cooperations. At the same time, the vertex-cover number by core influence method is

J. Stat. Mech. (2019) 073401

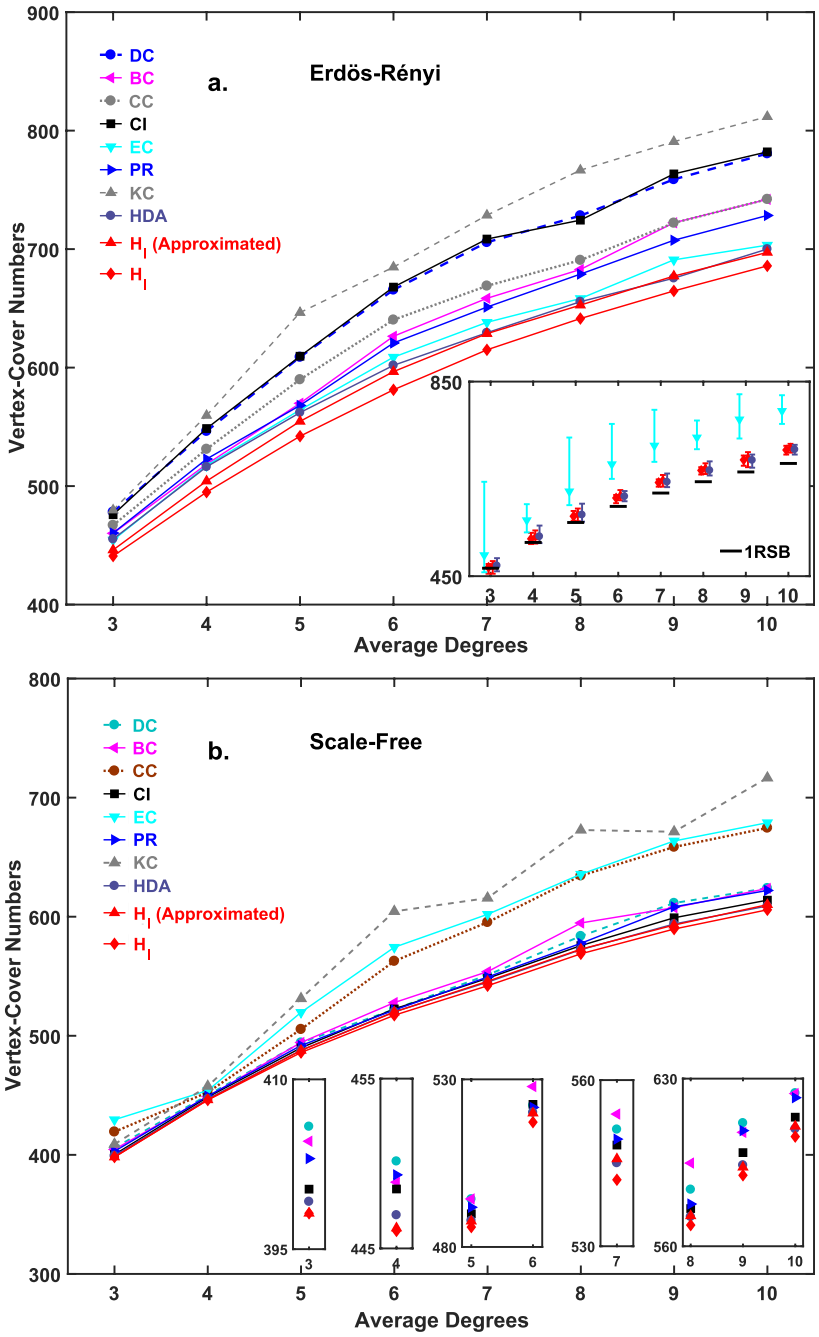

Figure 4. The vertex-cover number by deleting the graphs into no-core ones. The experiments are conducted on ER and SF graphs with average degrees ranging from 3 to 10. For each average degree value, 30 graphs are generated and each graph contains 1000 nodes. Results of DC, KC, BC, CC, CI, HDA, PR, EC and core influence H_l are drawn on the graph. (a) Minimal vertex-cover numbers on ER graphs. The error bars of EC, HDA, H_l and approximated H_l (best methods) are plotted in the inside figure, and the theoretical results from 1RSB ansatz [16] are drawn as comparison. (b) Minimal vertex-cover numbers on SF graphs. Inside figures are the local amplifications.

J. Stat. Mech. (2019) 073401

also the lowest. It is hard to get the exact minimum vertex-cover for networks at this scale, yet we could see the H_l gets reasonable results to approximate the minimum vertex-cover problem.

For the collaboration network, observing the representative/typical authors involved in all the papers of some fields, namely finding the minimal authors set, can be viewed as the minimum vertex-cover problem. The covered nodes correspond to the representative authors, and the edges covered by one node are regarded as the research work this author participated in; the minimum vertex-cover is related to the representative authors set in this field. The fewer the representative authors in one field are, the more concentrated the researchers are; the larger the typical authors set is, the more diversified this research field is. The minimum vertex-cover in the collaboration network could work as the indicator of the activeness and participation of this research field, which could help discover more about the development of various fields in science.

5. Conclusion and discussion

In this paper, focusing on the leaf-removal core of minimum vertex-cover problem, a method based on dynamical system and power method is proposed. This core influence method could transform the graphs into no-leaf-removal-core ones fast by deleting selected nodes and breaking the leaf-removal cores. It is proved that any minimum vertex-covers of the whole graph can be located into leaf-removal cores breaking and maximal matching of the remaining graphs and the minimum coverage is achieved at the transition point. Compared to other node importance indices, the proposed core influence method could break down the leaf-removal core more efficiently, and the coverage number resulted from this method is better. Since the leaf-removal process can reveal some intrinsic characteristics of structures and percolations, this study provides a useful tool for researches on dynamical and hierarchical features on networks and a new perspective of the minimum vertex-cover problem.

Since the structures and organizations of leaf-removal cores are sophisticated and it involves some fundamental difficulties to understand the complex systems clearly, researches related to it are expected to further improve this method. More understandings and rigorous analysis on the transition to no-core graphs will help find accurate expressions on the leaf-removal cores, and efficient methods for judging whether a node belongs to the leaf-removal cores will improve the performance of core breaking and the recognition of the transition point. The investigation on the core percolation of some topological characteristics of NP problems will promote the comprehension of the intrinsic hardness, and it is believed the combination of functional structure of NP problems with some new strategies of complex networks will be an interesting direction.

Acknowledgments

This work is supported by the Fundamental Research Funds for the Central Universities, the National Natural Science Foundation of China (No.11201019), the International Cooperation Project No.2010DFR00700, Fundamental Research of Civil Aircraft No.

J. Stat. Mech. (2019) 073401

MJ-F-2012-04 and Beijing Natural Science Foundation (1192012, Z180005). The authors would like to thank Ph.D. candidate Jabeen Shamoona for useful discussion and careful reading for revising this manuscript.

References

[1] Karp R M 1972 *Proc. Symp. IBM Thomas J. Watson Research Center* (New York: Plenum) pp 85–103
[2] Cook S A 1971 *Proc. 3rd Ann. ACM Symp. on Theory of Computing* (New York: Association for Computing Machinery) pp 151–8
[3] Papadimitriou C H 2003 *Computational Complexity* (Chichester: Wiley)
[4] Gomez-Gardenes J, Echenique P and Moreno Y 2006 *Eur. Phys. J.* B **49** 259
[5] Breitbart Y, Chan C Y, Garofalakis M, Rastogi R and Silverschatz A 2001 *Proc. IEEE INFOCOM (IEEE Communication Society, Anchorage, Alaska)* pp 933–42
[6] Weigt M and Hartmann A K 2000 *Phys. Rev. Lett.* **84** 6118
[7] Zhou H 2005 *Phys. Rev. Lett.* **94** 217203
[8] Bollob'as B 1985 *Random Graphs* (New York: Academic)
[9] Mzard M, Ricci-Tersenghi F and Zecchina R 2003 *J. Stat. Phys.* **111** 505
[10] Weigt M and Hartmann A K 2001 *Phys. Rev.* E **63** 056127
[11] Mertens S, Mzard M and Zecchina R 2005 *Algorithms* **28** 340–73
[12] Wei W, Guo B and Zheng Z 2009 *J. Stat. Mech.* P02010
[13] Dorogovtsev S N, Goltsev A V and Mendes J F F 2006 *Phys. Rev. Lett.* **96** 040601
[14] Kitsak M, Gallos L K, Havlin S, Liljeros F, Muchnik L, Stanley H E and Makse H A 2010 *Nat. Phys.* **6** 888–93
[15] Morone F and Makse H A 2015 *Nature* **524** 65
[16] Weigt M and Zhou H 2006 *Phys. Rev.* E **74** 046110
[17] Wei W, Zhang R, Guo B and Zheng Z 2012 *Phys. Rev.* E **86** 016112
[18] Bondy J A and Murty U S R 1976 *Graph Theory with Applications* (Oxford: Elsevier)
[19] Gabow H N and Tarjan R E 1991 *J. ACM* **38** 815–53
[20] Wiggins S 2003 *Introduction to Applied Nonlinear Dynamical Systems and Chaos* vol 2 (Berlin: Springer)
[21] Bordenave C, Lelarge M and Massoulié L 2015 *IEEE 56th Annual Symp. on Foundations of Computer Science* (IEEE) pp 1347–57
[22] Hashimoto K I 1989 *Automorphic Forms and Geometry of Arithmetic Varieties* (Cambridge: Academic Press) pp 211–80
[23] Horn R A and Johnson C R 2012 *Matrix Analysis* (New York: Cambridge University Press)
[24] Krzakala F, Moore C, Mossel E, Neeman J, Sly A, Zdeborov L and Zhang P 2013 *Proc. Natl Acad. Sci.* **110** 20935–40
[25] Karrer B, Newman M E J and Zdebrova L 2014 *Phys. Rev. Lett.* **113** 208702
[26] Albert R, Jeong H and Barabási A 2000 Error and attack tolerance of complex networks *Nature* **406** 378–82
[27] Cohen R, Erez K, ben-Avraham D and Havlin S 2001 Breakdown of the internet under intentional attack *Phys. Rev. Lett.* **86** 3682–5
[28] Kincaid D, Kincaid D R and Cheney E W 2009 *Numerical Analysis: Mathematics of Scientific Computing* (Providence, RI: American Mathematical Society)
[29] Newman M 2018 *Networks* (Oxford: Oxford University Press)
[30] Freeman L C 1977 A set of measures of centrality based on betweenness *Sociometry* 35–41
[31] Bavelas A 1950 Communication patterns in task-oriented groups *The J. Acoust. Soc. Am.* **22** 725–30
[32] Sabidussi G 1966 The centrality index of a graph *Psychometrika* **31** 588–603
[33] Page L, Brin S, Motwani R and Winograd T 1999 The PageRank Citation Ranking: Bringing Order to the Web *Stanford InfoLab Report* 1999-66
[34] Freeman L C 1978 *Soc. Netw.* **1** 215–39
[35] Newman M E J 2000 *Networks: an Introduction* (Oxford: Oxford University Press)
[36] Leskovec J, Kleinberg J and Faloutsos C 2007 *ACM Transactions on Knowledge Discovery from Data (TKDD)* vol **1**
[37] Zhao J and Zhou H 2014 *Chin. Phys.* B **23** 078901