Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

Rethinking the implementation and application of the Benczur-Karger minimum cuts algorithm

Hanqin Gu *

Western Reserve Academy, Hudson, USA

ARTICLE INFO

Article history:
Received 23 February 2024
Received in revised form
23 June 2024
Accepted 27 June 2024

Keywords:
Minimum cut
Graph theory
Benczur-Karger algorithm
Compression factor
Social network analysis

ABSTRACT

In graph theory and network analysis, finding the minimum cut in a graph is a fundamental algorithmic challenge. This paper explores the development and application of Benczur-Karger's minimum cut algorithms, focusing on the relationship between theoretical advancements and practical implementation. Despite the algorithm's advantages, there are challenges related to its implementation complexities and the effects of compression factor settings. To address these issues, this paper first implements Benczur-Karger's minimum cuts algorithm in Python and discusses the implementation details. Additionally, we propose a new compression factor setting for Benczur-Karger's minimum cuts algorithm and conduct an experiment with this new setting. The experimental results show that our proposed compression factor performs better than the original one. Finally, we discuss the application of Benczur-Karger's minimum cuts algorithm in social network analysis, a field where its use has been limited. The code is available at https://github.com/HarleyHanqin/Modified_BK.

1. Introduction

In the realm of graph theory and network analysis, the concept of finding the minimum cut in a graph has emerged as a fundamental and indispensable algorithmic challenge (Gayathri et al., 2024; Henzinger et al., 2018). The significance of minimum cut algorithms lies at the core of numerous real-world applications, ranging from network reliability and transportation optimization (Niu et al., 2020) to image segmentation (Niazi and Rahbar, 2024) and community detection (Becchetti et al., 2020). Moreover, minimum cut algorithms can be incorporated into pre-trained language models for natural language processing tasks (Huang et al., 2024a), such as multi-hop question and answering (Huang et al., 2024b; Jin et al., 2023). As we delve into the intricate world of graph algorithms, it becomes evident that the efficient identification of a minimum cut not only bears theoretical importance but also holds practical implications in diverse domains.

The evolution of minimum cut algorithms reflects the continuous interplay between theoretical insights and practical applicability (Manoharan and Sathesh, 2020). From classic methods such as the Ford-Fulkerson algorithm (Bulut and Özcan, 2021) and the Stoer-Wagner algorithm (Zhao et al., 2020) to more recent contributions such as randomized contraction algorithms (Cygan et al., 2020) and multi-phase algorithms (Zhou et al., 2019), researchers have tirelessly strived to enhance the efficiency, scalability, and versatility of minimum cut computations. This evolution is particularly crucial in the context of ever-expanding datasets and complex network structures encountered in modern applications.

In the dynamic and ever-evolving landscape of graph theory, the quest for efficient algorithms to address fundamental problems, such as identifying cuts in graphs, remains a focal point of research. One notable and increasingly recognized paradigm in this pursuit is the application of random sampling techniques. The use of randomness as a tool for algorithmic design has gained prominence, as it offers a fresh perspective and innovative solutions to long-standing challenges (Karger, 1994a; 1994b).

Benczur and Karger (1996) enhanced existing random sampling methodologies to approximate solutions for graph problems involving cuts. They introduce a linear time construction, which converts a graph with n vertices into an $O(n \log n)$-edge graph with the same vertices. In addition, the cuts in this

* Corresponding Author.
Email Address: guh25@wra.net
https://doi.org/10.21833/ijaas.2024.07.007
Corresponding author's ORCID profile:
https://orcid.org/0009-0003-1609-2800

transformed graph possess approximately the same value as those in the original graph. This modified graph facilitates the application of the $O(mn)$-time maximum flow algorithm by Goldberg and Tarjan, enabling the identification of an s–t minimum cut in $O(n^2)$ time. This corresponds to a $(1 + \varepsilon)$-approximation of the minimum s–t cut in the original graph. Similarly, they demonstrate an $O(n^2)$-time approximation for the sparsest cut using this approach.

However, the implementation of Benczur-Karger's minimum cuts algorithm is quite complicated, and the setting of the compression factor impacts the algorithm's performance. Moreover, Benczur-Karger's minimum cuts algorithm is mostly used for image segments, and less attention has been given to other applications, such as social network analysis. In this paper, we rethink the implementation and application of Benczur-Karger's Minimum Cuts Algorithm. The contributions of this paper are summarized below:

1. First, we implement Benczur-Karger's minimum cuts algorithm in Python and discuss the details of the implementation.
2. We introduce a new compression factor for Benczur-Karger's minimum cuts algorithm and conduct an experiment for the proposed compression factor.
3. Finally, we will conduct Benczur-Karger's minimum cuts algorithm on a social network dataset and discuss its application for social network analysis.

2. Relative work

2.1. Graph cut algorithms

Graph cut algorithms have been a cornerstone in the field of graph theory, offering essential solutions for a wide array of applications, including image segmentation, network optimization, and community detection. Pioneering works laid the groundwork for classical algorithms such as the Ford–Fulkerson algorithm (Ford and Fulkerson, 1956) and the Stoer-Wagner algorithm (Stoer and Wagner, 1997), which addressed the minimum cut problem by iteratively finding maximum flows. Nagamochi and Ibaraki (1992a, 1992b) explored the scope of the minimum cut problem and introduced a linear-time algorithm for finding a sparse k-connected spanning subgraph of a k-connected graph. Inspired by the studies of Nagamochi and Ibaraki's (1992a, 1992b) studies, we combine a linear-time algorithm with the Benczur-Karger minimum cut algorithm (Benczúr and Karger, 1996) to improve the performance. The 2-respecting min-cut problem, a subroutine within Karger's (2000) well-known randomized near-linear-time min-cut algorithm, has undergone examination. The objective is to identify, in any weighted graph G and its spanning tree T, the minimum cut among those containing at most two edges in T. Mukhopadhyay and Nanongkai (2020) introduced a novel approach to address this problem, offering an easily implementable solution across various contexts. This has subsequently led to the development of randomized min-cut algorithms for weighted graphs.

2.2. Random Sampling for Graphs

Random sampling techniques have emerged as powerful tools for addressing computationally challenging problems in graph theory. Early contributions by Karger and Stein (1996) showcased the potential of randomized contraction algorithms for approximating minimum cuts. Building upon this foundation, Batson et al. (2014) introduced innovative linear-time constructions for transforming graphs while preserving cut properties.

In the domain of sparsest cut approximation, Arora et al. (2009) proposed a polynomial-time algorithm with provable guarantees, illustrating the effectiveness of random sampling in tackling diverse graph cut problems. These approaches demonstrate the adaptability and efficiency of random sampling methodologies across various graph-related challenges.

3. Implementation of Benczur-Karger's minimum cuts algorithm

3.1. Certificate

The implementation of the Certificate is shown in Algorithm 1. The output of this algorithm is a set of edges with a maximum size $k(n-1)$, including all the edges with strong connectivity $\leq k$. Notably, not every edge in the output has strong connectivity less than or equal to k.

Algorithm 1 Certificate (G, k)

Input: A multi-graph $G = (V, E)$, standard connectivity k.
Output: A set of edges containing all edges with strong connectivity $\leq k$.

$E_1 \leftarrow \emptyset, E_2 \leftarrow \emptyset, \cdots, E_{|E|} \leftarrow \emptyset$.
For all $v \in V$, $r(v) \leftarrow 0$
while V is not empty **do**
    choose a vertex x with the largest $r(x)$ value
    for each edge$(x, y) \in G.E$ **do**
    $Er(y)+1 \leftarrow Er(y)+1 \cup \{(x, y)\}$
    if $r(x) = r(y)$ **then**
    $r(x) \leftarrow r(x) + 1$
    end if
    $r(y) \leftarrow r(y) + 1$
    $G.E \leftarrow G.E - (x, y)$
    end for
    $G.V \leftarrow G.V - x$
end while
return $E_1 \cup E_2 \cup \cdots \cup E_k$

The subroutine Certificate (G, k) runs in $O(m)$ time since the algorithm should iterate over each edge exactly once (Benczúr and Karger, 1996). The detailed proof can be found in Nagamochi and Ibaraki (1992a). In Certificate, each node is assigned a $r(v)$ value. We leverage a double linked list with

each element in the list containing an r value a set of vertices with that r value when searching for the vertex with the largest $r(v)$ value. Two elements with adjacent r values are linked in both directions. The implementation of this data structure significantly reduces the time complexity for finding the largest $r(v)$ value. In particular, adding $r(v)$ for some vertex v can be completed in $O(1)$ time with the implementation of a doubly linked list.

3.2. Partition

The output of Partition contains fewer edges with connectivity greater than k, while Certificate is capable of providing edges with low-standard connectivity. The implementation of Partition is shown in Algorithm 2. The Partition algorithm runs in $O(m)$ time because it strictly iterates over each edge exactly once. Partition requires contracting the edges of the graph and returning the original edges before they are contracted. We use a different data structure to store which vertices are contracted to the current vertex. The implementation of Partition just needs to search for each edge in the original graph once rather than calculate and confirm whether it contracts to one of the selected edges.

3.3. WeakEdges and estimation

Using Partition, WeakEdges serves to determine the k-weak edges. Unlike the two previous algorithms, WeakEdges determines edges with less strong connectivity instead of standard connectivity. The implementation of WeakEdges is shown in Algorithm 3. Based on WeakEdges, we estimate the strong connectivity of each edge in the graph through Estimation (shown in Algorithm 4).

Algorithm 2 Partition $(G, k, C1)$

Input: A graph $G = (V, E)$, connectivity k
Output: A distilled set of edges containing all edges with strong connectivity ≤ k.

1: **while** $m > C_1 * k * (n - 1)$ **do**
2: $E' \leftarrow$ Certificate(G, k)
3: **for** all edges (x, y) in $E - E'$ **do**
4: $G.E \leftarrow contract\ (x, y)$ ▷ contract vertex y onto x
5: $G.V \leftarrow V - y$
6: **end for**
7: **end while**
8: **return** All uncontracted edges of G

Algorithm 3 WeakEdges (G, k, C2)

Input: A graph G = (V, E), strong connectivity k
Output: A set of edges containing all the k-weak edges
1: $\boldsymbol{\varepsilon} \leftarrow \emptyset$
2: **for** log(n) times **do**
3: E′ ← Partition$(G, C_2 * k)$
4: $\boldsymbol{\varepsilon} \leftarrow \boldsymbol{\varepsilon} \cup E'$
5: $G.E \leftarrow G.E - E'$
6: **end for**
7: **return** ε

Algorithm 4 Estimation (G, k)

Input: A graph G = (V, E, C3), current strong connectivity k which starts at 1, C3 ∈ {1, 2}.
Output: A map that shows the estimation of the strong connectivity for every edge.
1: $E' \leftarrow$ WeakEdges$(G, C_3 * k)$
2: **for** all e ∈E' **do**
3: k¯$e \leftarrow$ k.
4: **end for**
5: $G.E \leftarrow G.E - E'$
6: **for** all nontrivial connected components $H \in G$ **do**
7: Estimation$(H, 2 * k)$
8: **end for**

4. Generic compression factor

Benczur and Karger (1996) defined the compression factor ρ as $O\left(\frac{d \ln n}{\epsilon^2}\right)$, which leads to setting the compression factor to $O(\ln n)$. We introduce a new compression factor $\rho = \frac{2k_{\text{average}}}{(2+\sigma^2 m)}$ for Benczur-Karger's Minimum Cuts Algorithm, which can improve the performance of the algorithm (see section 5. Experiment and Results).

4.1. Proof

Lemma 1: Each cut in a graph should have an expected cut size $\frac{m}{2}$.

Proof: By randomly choosing a cut, we split the set of vertices. If each vertex $v \in G.V$ is set into one of two sets, the two sides are split by the cut. For each edge $e \in G$. E, the probability of its two vertices coming up on the two different sides of the cut is exactly 1/2. Therefore, each edge is present in exactly half of the cuts, so that the average cut size is $\frac{m}{2}$.

We assume that the strong connectivity k_es remain relatively close for edges in the same graph. Consequently, the compression probability p is the same. Assuming that the probability is the same for all edges, for each edge, it is kept with probability p (with weight $\frac{1}{p}$) and discarded with probability $1 - p$. Thus, the resulting cut size for each cut can be expressed with a binomial distribution. We redefine ϵ as the expected value of $\left|\frac{\text{new cut value}}{\text{old cut value}} - 1\right|$.

Lemma 2: Let c be the value of the selected cut; given ϵ, the expected compression factor p is $\frac{1}{1+\epsilon^2 c}$

Proof: As a result of the binomial distribution, the standard deviation σ satisfies:

$$p\sigma = \sqrt{cp(1-p)} \quad (1)$$

by definition,

$$\epsilon = \frac{\sigma}{c} \quad (2)$$

simplifying, we obtain:

$$\epsilon^2 = \frac{1-p}{cp} = \frac{1}{cp} - \frac{1}{c} \quad (3)$$

by substituting $p = \frac{\rho}{k_e}$ and $c = \frac{m}{2}$, we obtain:

$$(2 + \sigma^2 m)\rho = 2k_e \quad (4)$$

Since k_e is the strong connectivity in the graph, we can obtain the average of the strong connectivities of all the edges in the graph, which accounts for the new expression for the compression factor:

$$\rho = \frac{2k_{\text{average}}}{(2+\sigma^2 m)} \quad (5)$$

5. Experiment and results

5.1. Datasets

We conducted the experiment using the social circle (Facebook) dataset (Leskovec and Mcauley, 2012). This dataset comprises 'circles' or 'friends lists' sourced from Facebook and collected through a Facebook app administered to survey participants. It encompasses node features (profiles), circles, and ego networks. To ensure privacy, Facebook data have undergone anonymization by substituting the original Facebook-internal IDs for each user with new values. Additionally, feature vectors in this dataset were presented in an anonymized manner, obscuring the interpretation of these features. For example, where the original dataset may have denoted a feature as "political=Democratic Party," the anonymized data would instead display "political=anonymized feature 1." Consequently, while the anonymized data allows for the identification of shared political affiliations between two users, they do not disclose the specific nature of individual political affiliations. The dataset statistics are shown in Table 1. Note that these statistics in Table 1 were compiled by combining the ego networks, including the ego nodes themselves (along with an edge to each of their friends).

5.2. Results

We conducted an experiment to compare the performance of our proposed compression factor with the original factor. We measured the performance of Benczur-Karger's minimum cuts algorithm by analyzing the cut values. As shown in Table 2, the results indicate that our proposed compression factor outperforms the original one for cut values ranging from 1,000 to 7,000. Performance was measured by the time taken to reach the cut values. Compared to the original algorithm, the algorithm with the new compression factor reduced the running time by approximately 51.6% on average. As the cut values increase, our proposed compression factor significantly decreases the algorithm's search time; however, it only reduces the time by 0.9 seconds in the range from 6,000 to 7,000. This suggests that while the new compression factor effectively improves the algorithm's performance, it has certain limitations.

Table 1: Dataset statistics

Nodes	4039
Edges	88234
Nodes in largest WCC	4039 (1.000)
Edges in largest WCC	88234 (1.000)
Nodes in largest SCC	4039 (1.000)
Edges in largest SCC	88234 (1.000)
Average clustering coefficient	0.6055
Number of triangles	1612010
Fraction of closed triangles	0.2647
Diameter (longest shortest path)	8
90-percentile effective diameter	4.7

Table 2: Experimental results comparing proposed compression factor with the original

Cut values	Original (sec)	Our proposed (sec)
1,000	27.2	15.2
2,000	39.7	20.5
3,000	54.2	25.7
4,000	72.8	32.3
5,000	83.2	36.3
6,000	98.5	43.7
7,000	115.2	59.5

6. Application for social network analysis

We conduct an experiment with Benczur-Karger's minimum cuts algorithm on social circle (Facebook) datasets and further discuss the application of this algorithm to social network analysis.

6.1. Community detection

Minimum cuts can be employed to identify cohesive communities within a social network. By identifying the edges with the lowest weights or, conversely, removing edges with the highest weights, the network can be divided into distinct communities. This approach aids in understanding the natural divisions and subgroups within a larger social network, revealing patterns of interactions and relationships.

6.2. Identifying key connectors

Minimum cuts help pinpoint critical edges whose removal would result in a significant disconnection

between different parts of the network. These edges often represent key connectors or bridges between various communities. Identifying and understanding these critical connectors is essential for assessing the robustness and vulnerability of a social network.

6.3. Graph partitioning for scalability

In large-scale social networks, graph partitioning based on minimum cuts can be employed to divide the network into smaller, more manageable components. This approach facilitates scalability in terms of analysis and computational efficiency, enabling researchers to focus on specific subnetworks or communities within larger social graphs.

6.4. Bi-partite graph analysis

Social networks often involve interactions between two types of entities (e.g., users and events). Minimum cuts in bipartite graphs can reveal critical interactions or connections that bridge the gap between these two types of information, providing insights into the dynamics of user–event relationships.

7. Conclusion

In this paper, we revisited the implementation and application of the Benczur-Karger minimum cuts algorithm, offering several notable contributions to the field of graph theory and network analysis. We began by implementing the Benczur-Karger algorithm in Python, providing a comprehensive discussion of the implementation details. This effort highlights the practical complexities and challenges involved in the algorithm's application.

A significant portion of our work focused on introducing a new compression factor for the Benczur-Karger algorithm. Through rigorous experimentation, we demonstrated that this proposed compression factor significantly outperforms the original, especially for cut values ranging from 1,000 to 7,000. Our results indicate an average reduction in running time of approximately 51.6%, showcasing the efficiency and potential of our proposed method. However, we also observed that the performance gains diminish for higher cut values, suggesting that while the new compression factor is effective, it has certain limitations.

Furthermore, we explored the application of the Benczur-Karger minimum cuts algorithm within the realm of social network analysis, an area where its use has been relatively limited. We demonstrated how the algorithm can be utilized for various tasks such as community detection, identifying key connectors, graph partitioning for scalability, and bipartite graph analysis. These applications underline the versatility and practical relevance of the Benczur-Karger algorithm in analyzing complex social networks. Our work not only bridges theoretical advancements with practical implementation but also extends the utility of the Benczur-Karger minimum cuts algorithm to new domains. Future research could focus on further refining the compression factor and exploring additional applications in other types of networks.

In conclusion, this paper contributes to both the theoretical and practical aspects of minimum cut algorithms, offering insights and tools that can enhance their performance and applicability in diverse fields.

Compliance with ethical standards

Ethical considerations

This study adheres to ethical standards and guidelines. The Facebook dataset used was anonymized to protect privacy, and no personal information was accessed. All algorithms were developed and tested ethically. The application in social network analysis was conducted using publicly available anonymized data, ensuring privacy and data security.

Conflict of interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

Arora S, Rao S, and Vazirani U (2009). Expander flows, geometric embeddings and graph partitioning. Journal of the ACM, 56(2): 5. https://doi.org/10.1145/1502793.1502794

Batson J, Spielman DA, and Srivastava N (2014). Twice-Ramanujan sparsifiers. SIAM Journal on Computing, 41(6): 1704-1721. https://doi.org/10.1137/090772873

Becchetti L, Clementi AE, Natale E, Pasquale F, and Trevisan L (2020). Find your place: Simple distributed algorithms for community detection. SIAM Journal on Computing, 49(4): 821-864. https://doi.org/10.1137/19M1243026

Benczúr AA and Karger DR (1996). Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In the Proceedings of the 28th Annual ACM Symposium on Theory of Computing, Philadelphia, USA: 47-55.

Bulut M and Özcan E (2021). Optimization of electricity transmission by Ford–Fulkerson algorithm. Sustainable Energy, Grids and Networks, 28: 100544. https://doi.org/10.1016/j.segan.2021.100544

Cygan M, Komosa P, Lokshtanov D, Pilipczuk M, Pilipczuk M, Saurabh S, and Wahlström M (2020). Randomized contractions meet lean decompositions. ACM Transactions on Algorithms (TALG), 17(1): 1-30. https://doi.org/10.1145/3426738

Ford LR and Fulkerson DR (1956). Maximal flow through a network. Canadian Journal of Mathematics, 8: 399-404. https://doi.org/10.4153/CJM-1956-045-5

Gayathri G, Mathew S, and Mordeson JN (2024). Max-flow min-cut theorem for directed fuzzy incidence networks. Journal of Applied Mathematics and Computing, 70(1): 149-173. https://doi.org/10.1007/s12190-023-01952-x

Henzinger M, Noe A, Schulz C, and Strash D (2018). Practical minimum cut algorithms. Journal of Experimental Algorithmics, 23: 1-22. https://doi.org/10.1145/3274662

Huang G, Li Y, Jameel S, Long Y, and Papanastasiou G (2024a). From explainable to interpretable deep learning for natural language processing in healthcare: How far from reality? Computational and Structural Biotechnology Journal, 24: 362-373. https://doi.org/10.1016/j.csbj.2024.05.004 **PMid:38800693 PMCid:PMC11126530**

Huang G, Long Y, Luo C, Shen J, and Sun X (2024b). Prompting explicit and implicit knowledge for multi-hop question answering based on human reading process. Arxiv Preprint Arxiv:2402.19350. https://doi.org/10.48550/arXiv.2402.19350

Jin W, Zhao B, Yu H, Tao X, Yin R, and Liu G (2023). Improving embedded knowledge graph multi-hop question answering by introducing relational chain reasoning. Data Mining and Knowledge Discovery, 37(1): 255-288. https://doi.org/10.1007/s10618-022-00891-8

Karger DR (1994a). Random sampling in cut, flow, and network design problems. In the Proceedings of the 26th Annual ACM Symposium on Theory of Computing, Montreal, Canada: 648-657. https://doi.org/10.1145/195058.195422

Karger DR (1994b). Using randomized sparsification to approximate minimum cuts. In the Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms, Arlington, USA.

Karger DR (2000). Minimum cuts in near-linear time. Journal of the ACM, 47(1): 46-76. https://doi.org/10.1145/331605.331608

Karger DR and Stein C (1996). A new approach to the minimum cut problem. Journal of the ACM, 43(4): 601-640. https://doi.org/10.1145/234533.234534

Leskovec J and Mcauley J (2012). Learning to discover social circles in ego networks. In the Proceedings of Advances in Neural Information Processing Systems 25 (NIPS 2012), Red Hook, USA: 539–547.

Manoharan DS and Sathesh A (2020). Improved version of graph-cut algorithm for CT images of lung cancer with clinical property condition. Journal of Artificial Intelligence and Capsule Networks, 2(4): 201-206. https://doi.org/10.36548/jaicn.2020.4.002

Mukhopadhyay S and Nanongkai D (2020). Weighted min-cut: Sequential, cut-query, and streaming algorithms. In the Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, Chicago, USA: 496-509. https://doi.org/10.1145/3357713.3384334 **PMid:32629488**

Nagamochi H and Ibaraki T (1992a). Computing edge-connectivity in multigraphs and capacitated graphs. SIAM Journal on Discrete Mathematics, 5(1): 54-66. https://doi.org/10.1137/0405004

Nagamochi H and Ibaraki T (1992b). A linear-time algorithm for finding a sparse *k*-connected spanning subgraph of a *k*-connected graph. Algorithmica, 7(1): 583-596. https://doi.org/10.1007/BF01758778

Niazi M and Rahbar K (2024). Setting the regularization coefficient based on image energy in image segmentation using kernel graph cut algorithm. Journal of Electronic Imaging, 33(2): 023031. https://doi.org/10.1117/1.JEI.33.2.023031

Niu YF, Wan XY, Xu XZ, and Ding D (2020). Finding all multi-state minimal paths of a multi-state flow network via feasible circulations. Reliability Engineering & System Safety, 204: 107188. https://doi.org/10.1016/j.ress.2020.107188

Stoer M and Wagner F (1997). A simple min-cut algorithm. Journal of the ACM, 44(4): 585–591. https://doi.org/10.1145/263867.263872

Zhao P, Yu J, Zhang H, Qin Z, and Wang C (2020). How to securely outsource finding the min-cut of undirected edge-weighted graphs. IEEE Transactions on Information Forensics and Security, 15: 315-328. https://doi.org/10.1109/TIFS.2019.2922277

Zhou Y, Li Y, Zhang Z, Wang Y, Wang A, Fishman EK, Yuille AL, and Park S (2019). Hyper-pairing network for multi-phase pancreatic ductal adenocarcinoma segmentation. In the Medical Image Computing and Computer Assisted Intervention–22nd International Conference, Springer International Publishing, Shenzhen, China: 155-163. https://doi.org/10.1007/978-3-030-32245-8_18