A Faster Algorithm for Betweenness Centrality Based on Adjacency Matrices

Feng Yelai, Wang Huaixi, Lu Hongyi

Abstract—Betweenness centrality is essential in complex network analysis; it characterizes the importance of nodes and edges in networks. It is a crucial problem that exactly computes the betweenness centrality in large networks faster, which urgently needs to be solved. We propose a novel algorithm for betweenness centrality based on the parallel computing of adjacency matrices, which is faster than the existing algorithms for large networks. The time complexity of the algorithm is only related to the number of nodes in the network, not the number of edges. Experimental evidence shows that the algorithm is effective and efficient. This novel algorithm is faster than Brandes' algorithm on small and dense networks and offers excellent solutions for betweenness centrality index computing on large-scale complex networks.

Keywords—Complex Network, Betweenness Centrality, Adjacency Matrix, Parallel Computing, GPU Acceleration.

1 Introduction

With the rapid developments of digital society, an increasing number of real networks appear in our lives, which are large scale and complex. Real networks can be regarded as complex networks in different fields, such as industrial networks[1], communication networks[2], transportation networks[3], social networks[4], and the Internet of Things[5]. Research on complex networks has attracted considerable attention from academic fields[6],[7], to deeply study the characteristics of a complex network, network science has emerged[8]. Network science focuses on network modeling, centrality measures and global characteristics of networks, link prediction and recommendation algorithms based on networks, the control and optimization of networks, and network dynamics.

In order to characterize the importance of nodes and edges in the networks, a large volume of research on networks has been devoted to the concept of centrality, such as degree centrality[9], closeness centrality[10], betweenness centrality[11], Katz centrality[12] and PageRank centrality[13]. Betweenness centrality is a core measure used to estimate the influence of vertices and edges in complex networks, and the calculation of betweenness centrality was incorporated into DARPA HPCS SSCA#2 in 2008, which is a benchmark used extensively to evaluate the performance of emerging high-performance computing architectures for graph analysis[14],[15].

The scales of complex networks grow rapidly, which makes comparative centrality analysis with more than one million nodes prohibitive. Brandes' algorithm is the fastest algorithm for computing betweenness centrality[16],[17]; however, it cannot fully meet the demand for analysis of complex networks. There is a bottleneck in the existing algorithm, since Brandes' algorithm is based on single-source shortest-path traversal algorithms and the accumulation technique[4]. To a certain extent， this bottleneck restricts the acceleration of Brandes' algorithm by parallel computing and thus makes it perform poorly on large-scale networks.

In previous studies on the calculation of betweenness centrality through parallel and distributed computation, there are four main technical schemes: solving multiple single-source shortest-path problems at the same time [14],[18],[19]; accelerating Brandes' algorithm with GPUs [20],[21],[22],[23]; calculating the shortest paths and then searching for neighbors via parallelized processing after the source node is determined[24]; and dividing and conquering a large network, calculating the subgraphs individually, and synthesizing the calculation results to obtain the betweenness centrality[25],[26]. These studies have focused on sparse networks and have provided a variety of parallel optimizations of Brandes' algorithm. However, these methods also severely degenerate in performance on dense networks. Exactly computing the betweenness centrality in large-scale networks faster has become a critical issue in network science, and there is an urgent need for its application.

In this paper, we propose a more efficient algorithm based on aDjacency mAtrix operations for computing betWeenness ceNtrality, named DAWN. The feasible network scale for the computation of betweenness centrality can be extended much further with the rapid development of hardware. DAWN has the following characteristics:

(1) DAWN is suitable for sparse networks, dense networks, unweighted networks, weighted networks, undirected networks and directed networks.

(2) The time complexity of DAWN depends only on the number of nodes and is insensitive to the density of the networks. Hence, its performance advantage is more prominent in large-scale networks.

(3) The operations of DAWN are mainly matrix operations which can be accelerated with GPUs, and improvements to hardware will bring greater computation efficiency.

In Section 2, we introduce the theoretical foundation

of betweenness centrality and its typical algorithms on co mplex networks. In Section 3, we describe the novel algori thm based on the adjacency matrix. In Section 4, we demo nstrate the experimental results and a comparative perfor mance analysis with existing algorithms. Finally, we concl ude the work of this paper and propose future problems i n Section 5.

2 Theoretical Foundations

In this section, we describe the theoretical foundation of t his paper, in three aspects: the concept and properties of b etweenness centrality, typical algorithms for computing b etweenness centrality and an introduction to GPUs.

2.1 Betweenness Centrality

Bavelas et el. proposed betweenness centrality for social n etwork analysis and research[10]. Freeman created a rigor ous mathematical definition of betweenness centrality an d proposed a method of evaluating the influence of nodes based on betweenness centrality[9]. This method is based on the fact that a critical node with a high frequency on t he shortest paths of a network has a greater impact in the network than a subordinate node with a low frequency o n the shortest paths of the network, and and it has comple xities of $\Theta(n^3)$ in time and $\Theta(n^2)$ in space, where n is the n umber of nodes in the network.

Brandes proposed a fast betweenness centrality algor ithm that requires $\Theta(m+n)$ space and runs in $\Theta(mn)$ and $\Theta(mn+n^2\log n)$ time on unweighted and weighted netw orks, respectively, where m is the number of links[4]. This algorithm has been integrated into a notable open-source project in network analysis "NetworkX"[27].

Construct a connected network $G=(V,E)$ according to the physical structure of the network, where V represe nts the set of nodes in the network and E represents the se t of edges. We use n and m to denote the numbers of nod es and edges in the networks, respectively. Betweenness c entrality is usually divided into node betweenness central ity and edge betweenness centrality. The definition of nod e betweenness is the ratio of the number of shortest paths passing through v between v_i and v_j to the total number o f shortest paths between two nodes in the network. The b etweenness of node v in the network is defined as:

$$Bet\,(v)=\sum_{v_i,v_j\in V}\frac{\sigma(v_i,v_j|v)}{\sigma(v_i,v_j)}, \tag{1}$$

where $\sigma(v_i,v_j)$ represents the number of shortest paths pa ssing between v_i and v_j and $\sigma(v_i,v_j|v)$ represents the num ber of shortest paths passing through v between v_i and v_j.

The definition of edge betweenness is similar to that of node betweenness, except that the object is changed fr om nodes to edges. The betweenness of edge e is defined as:

$$Be\,t\,(e)=\sum_{v_i,v_j\in V}\frac{\sigma(v_i,v_j|e)}{\sigma(v_i,v_j)}. \tag{2}$$

To facilitate comparative experiments, we adopt the n ormalized form of betweenness centrality in this paper, w hich is defined as follows:

$$\begin{cases} bet\,(v)=\dfrac{2}{(n-1)\cdot(n-2)}\cdot Bet(v), \\ be\,t\,(e)=\dfrac{2}{n(n-1)}\cdot Bet(e). \end{cases} \tag{3}$$

Wang et al. proved that the sums of the normalized form of betweenness centrality satisfy the following identi ties in a connected network[28],[29]:

$$\begin{cases} \sum_{v\in V} bet\,(v) = l-1, \\ \sum_{e\in E} bet\,(e) = l, \end{cases} \tag{4}$$

where l is the average length of the shortest paths in the n etwork.

2.2 Typical Algorithms of Betweenness Centrality

At present, the most known algorithm that exactly compu tes the betweenness centrality is Brandes' algorithm and Matthias et al. proposed its optimization[30]. It is comput ationally too expensive for us to calculate betweenness ce ntrality based on Brandes' algorithm in large-scale networ ks with tens of thousands of nodes. To solve the computat ion problem, Matteo Riondato et al. proposed an approxi mate Brandes algorithm (the randomized approximate Br andes algorithm, named the RA-Brandes algorithm)[31].

The motivation to complete operations faster makes a pproximation algorithms a viable choice in scenarios with low precision requirements. Therefore, approximate calc ulation methods have become an important research field. In this subsection, we introduce Brandes' algorithm, the K-shell approximation algorithm and the RA-Brandes alg orithm.

2.2.1 Brandes' Algorithm

Given network $G=(V,E)$, the betweenness centrality of n ode v is defined as:

$$Bet(v)=\sum_{s\neq t\neq v\in V}\frac{\sigma_{st}(v)}{\sigma_{st}}=\sum_{s\neq t\neq v\in V}Bet_{st}(v). \tag{5}$$

Brandes proposed the following formula to calculat e betweenness centrality:

$$Bet_{s\cdot}(v)=\sum_{w:v\in P_s(w)}\frac{\sigma_{sv}}{\sigma_{sw}}[1+Bet_{s\cdot}(w)], \tag{6}$$

$Bet_{s\cdot}(v)$ represents the betweenness centrality of node v, which has the shortest path to other nodes, with node s as the source node in the network. As the network becomes increasingly dense, the performance of Brandes' algorith m tends to degenerate, and it runs in $\Theta(n^{2k})$ time on both unweighted and weighted networks, where k is the lengt h of the diameter.

2.2.3 K-Shell Approximation Algorithm

In 1983, Seidman introduced the concept of K-core decom position[32]. In 2010, Maksim Kitsak et al. proposed the K -shell decomposition algorithm, which can be used for fas t approximate calculation of betweenness centrality[33].

The K-shell decomposition algorithm layers the netw ork according to degree, forms subnets with degree k, an d extracts some nodes to form a new subnet within each s ubnet. The new subnets have some of the characteristics o

f the original networks, and betweenness centrality indices can be calculated quickly, however they are approximate. The open-source project NetworkX integrates this algorithm to perform the approximate computation of betweenness centrality.

2.2.4 RA-Brandes Algorithm

Matteo Riondato et al. proposed the RA-Brandes algorithm[31], which appended a sampling strategy to Brandes' algorithm by selecting a certain subset of the entire network as a sampling set to calculate the betweenness centrality of node v.

In the open-source project Neo4j, the RA-Brandes algorithm uses two sampling strategies[34]. The first sampling strategy randomly selects nodes from the network to form a sample set. The sampling probability can be adjusted, and the default value is $\frac{log_{10}N}{e^2}$. If the probability is 1, the algorithm runs in the same way as the exact betweenness centrality algorithm. The second sampling strategy is to select nodes by degree. The mean degree of the nodes is calculated, and only the nodes whose degrees are higher than the mean are selected to form the sample set.

The RA-Brandes algorithm pays more attention to nodes with higher degrees, and the K-shell algorithm can retain nodes with diverse degrees.

2.3 GPU Acceleration

A graphics processing unit (GPU) is a visual processor focused on graphics processing. Researchers have found that GPUs are more suitable for large-scale computing than central processing units (CPUs). This led to the birth of the general-purpose graphics processing unit (GPGPU).

NVIDIA has developed a dedicated GPU programming language, CUDA, and released high-performance computing libraries such as cuBLAS[35]. CPU, GPU, FPGA, etc. processors each have fields in which they perform best [36], and GPUs are more suitable for linear algebra.

In experiments, the matrix multiplication function in cuBLAS takes only a ten thousandth of the time of a function written based on Openmp. As the scale of the matrix grows, this gap continues to increase. In view of the powerful performance of GPUs in dealing with matrix operations, we utilize a GPU to accelerate the computation in this paper.

3 DESIGNS OF THE BETWEENNESS CENTRALITY ALGORITHM

DAWN adopts a GPU for computational acceleration and uses Openmp and CUDA/C++ to compile. This requires $\Theta(n^2)$ space and $\Theta(n^{2.387})$ time (approximate values in the limit state), where n is the number of nodes in the network. DAWN addresses the issue of calculating the betweenness centrality index in large-scale networks.

Classic complex network models mainly include the Erdős–Rényi random network model[38],[39], Barabási–Albert scale-free network model[40],[41],[42], and the Watts-Strogatz small world network model[43],[44]. We choose the ER and BA models as the reference network models for algorithm testing. In this section, we introduce an algorithm for unweighted networks, an algorithm for weighted networks, the theoretical design of auxiliary functions and the parallel computing model of the algorithms.

3.1 Unweighted Networks

The unweighted network $G = (V, E)$ is conveniently described as an adjacency matrix:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix}. \quad (7)$$

The entry $a_{ij} = 1$ in adjacency matrix A means that there is an edge with a distance of 1 between node i and node j. We strip node v and the edges connecting it to other nodes from the network, which means setting all the values in row i_v and column i_v to 0 in adjacency matrix A to obtain a matrix B with node v stripped out. We define the iterative formulas of matrices $A^{(k)}$, $B^{(k)}$ and $C^{(k)}$ as:

$$\begin{cases} A^{(k+1)} = A^{(k)} \times A^{(1)}, \\ B^{(k+1)} = B^{(k)} \times B^{(1)}, \\ C^{(k+1)} = A^{(k+1)} - B^{(k+1)}. \end{cases} \quad 1 \le k < k+1 \le n-1 \quad (8)$$

Although the definitions of the three matrices are similar, the slight gaps between them are the focus of the DAWN algorithm. We define them as follows:

(1) The element $a_{ij}^{(k)}$ in matrix $A^{(k)} = \left(a_{ij}^{(k)}\right)$ represents the number of paths with length k between nodes i and j in the network.

(2) The element $b_{ij}^{(k)}$ in matrix $B^{(k)} = \left(b_{ij}^{(k)}\right)$ represents the number of paths with length k that do not pass through node v in going between nodes i and j in the network.

(3) The element $c_{ij}^{(k)}$ in matrix $C^{(k)} = \left(c_{ij}^{(k)}\right)$ represents the number of paths with length k that pass through node v in going between nodes i and j in the network.

Given the above definitions of the matrices, we propose a new betweenness centrality calculation formula:

$$Bet_{AM}(v) = \sum_{1 \le k \le n-1}^{1 \le i,j \le n \ and \ i \ne j} F_{ij}(k). \quad (9)$$

We define the function $F_{ij}(k)$ as:

$$F_{ij}(k) = \begin{cases} \frac{c_{ij}^{(k)}}{a_{ij}^{(k)}}, & a_{ij}^{(k)} \ne 0, i \ne j \ and \sum_{1 \le p \le k-1} a_{ij}^{(p)} = 0 \\ 0. & \text{otherwise} \end{cases} \quad (10)$$

To better analyze betweenness centrality, we suggest using the normalized formulas (11) and (12) for calculation. We define the formula used for undirected networks as:

$$bet_{AM}(v) = \frac{2}{(n-1) \times (n-2)} \sum_{1 \le k \le n-1}^{1 \le i,j \le n \ and \ i \ne j} F_{ij}(k). \quad (11)$$

Similiary, the formula used for directed networks is:

$$bet_{AM}(v) = \frac{1}{(n-1) \times (n-2)} \sum_{1 \le k \le n-1}^{1 \le i,j \le n \ and \ i \ne j} F_{ij}(k). \quad (12)$$

We use pseudocode to describe the algorithm above.

Algorithm 1 Unweighted Networks

Input: $A^{(n)}, B^{(n)}$
Output: $bet_{AM}(v)$

function MAIN($A^{(n)}, B^{(n)}, bet_{AM}(v)$)
  $bet_{AM}(v) \leftarrow 0$
  for $i = 1 \rightarrow n-2$ **do**
    $A^{(i+1)} \leftarrow A^{(i)} \times A$
    $B^{(i+1)} \leftarrow B^{(i)} \times B$
    $C^{(i+1)} \leftarrow A^{(i+1)} - B^{(i+1)}$
  end for
  for $k = 1 \rightarrow n$ **do**
    $sum \leftarrow sum + F(k)$
  end for
  $bet_{AM}(v) \leftarrow \frac{2}{(n-1)\times(n-2)} \times sum$
  return $bet_{AM}(v)$
end function

3.2 Weighted Networks

The weighted network $G = (V, E, W)$ is conveniently described as an adjacency matrix:

$$A_w = \begin{bmatrix} w_{11} & \cdots & w_{1n} \\ \vdots & \ddots & \vdots \\ w_{n1} & \cdots & w_{nn} \end{bmatrix}. \quad (13)$$

Let w be the weight indices of the edges. We assume that $w \in N$, where N is the set of natural numbers. In matrix A_w, w_{ij} represents the weight of the edge between nodes i and j. First, we define matrices $A_w^{(k)}$, $B_w^{(k)}$, $C_w^{(k)}$ and $H_{Aw}^{(k)}$:

(1) Matrix $A_w^{(k)}$ represents the number of paths with weight k between nodes i and j in the network. Matrix $A_w^{(1)}$ represents the adjacency matrix corresponding to the subnetwork formed by the edges with weight 1 in the adjacency matrix A_w.

(2) Matrix $B_w^{(k)}$ represents the number of paths with weight k that do not pass through node v between nodes i and j in the network. Matrix $B_w^{(1)}$ represents the adjacency matrix corresponding to the subnetwork formed by the edges with weight 1 that do not pass through node v in the adjacency matrix A_w.

(3) Matrix $C_w^{(k)}$ represents the number of paths with weight k that pass through node v between nodes i and j in the network. Matrix $C_w^{(1)}$ represents the adjacency matrix corresponding to the subnetwork formed by the edges with weight 1 that pass through node v in the adjacency matrix A_w.

(4) Matrix $H_{Aw}^{(k)}$ represents the adjacency matrix corresponding to the subnetwork formed by the edges with weight k in the adjacency matrix A_w. If there is no edge with weight k in the network, then $H_{Aw}^{(k)}$ is a null matrix, and we also define $H_{Aw}^{(1)}$ as a null matrix.

Second, we define iterative formulas for matrix $A_w^{(k)}$ as follows:

$$\begin{cases} A_w^{(k+1)} = A_w^{(k)} \times A_w^{(1)} + H_{Aw}^{(k+1)} & 1 \le k < k+1 \le w_0 \\ A_w^{(k)} = A_w^{(r)} \times A_w^{(\lambda w_0)}. & w_0 < k < n \end{cases} \quad (14)$$

Third, similar to matrix $A_w^{(k)}$, we define iterative formulas for matrix $B_w^{(k)}$ as follows:

$$\begin{cases} B_w^{(k+1)} = B_w^{(k)} \times B_w^{(1)} + H_{Bw}^{(k+1)} & 1 \le k < k+1 \le w_v \\ B_w^{(k)} = B_w^{(r)} \times B_w^{(\lambda w_v)}. & w_v < k < n \end{cases} \quad (15)$$

$H_{Bw}^{(k)}$ represents the adjacency matrix corresponding to the subnetwork formed by the edges with weight k in the adjacency matrix $B_w^{(k)}$, and $H_{Bw}^{(1)}$ is a null matrix. w_0 represents the upper limit of the weight of the weighted network represented by matrix $A_w^{(k)}$, and w_v represents the upper limit of the weight of the weighted network represented by matrix $B_w^{(k)}$.

Finally, according to (8), (14) and (15), we define the matrix $C_w^{(k)}$ as:

$$C_w^{(k)} = A_w^{(k)} - B_w^{(k)}. \quad (16)$$

We define the variable parameters r and λ in the above formulas as:

$$\begin{cases} r = k \bmod w_0, \quad \lambda = \left\lfloor \frac{k-r}{w_0} \right\rfloor & \text{In matrix } A_w^{(k)} \\ r = k \bmod w_v. \quad \lambda = \left\lfloor \frac{k-r}{w_v} \right\rfloor & \text{In matrix } B_w^{(k)} \end{cases} \quad (17)$$

The calculation processes of the unweighted and weighted networks are the same, except for the different matrix definitions and iterative formulas. Finally, we calculate the betweenness centrality of unweighted and weighted networks with (9)~(12).

3.3 Auxiliary Functions

For a network with n nodes, DAWN needs at most k_0 loops to complete the calculation, where k_0 is the maximum length of the shortest paths. We build auxiliary functions to reduce the number of loops, which are the search function based on the matrix and the threshold function.

3.3.1 Search Function for the Matrix

We define the matrix $\Gamma^{(n)}$ to register the value of the function $F(k)$. According to the definition of $F(k)$, the new operations do not affect the betweenness centrality indices after the matrix is filled, so we suggest ending the loop at this time and calculating $bet_{AM}(v)$.

For example, we define the matrix $\Gamma^{(8)}$ as:

$$\Gamma^{(8)} = \begin{bmatrix} \# & \# & \# & \# & \# & \# & \# & \# \\ \# & \# & * & * & * & * & * & * \\ \# & * & \# & * & * & * & * & * \\ \# & * & * & \# & * & * & * & * \\ \# & * & * & * & \# & * & * & * \\ \# & * & * & * & * & \# & * & * \\ \# & * & * & * & * & * & \# & * \\ \# & * & * & * & * & * & * & \# \end{bmatrix}. \quad (17)$$

"#" means that the value of this position is not included in the calculation of the betweenness centrality indices. "$*$" means that the value is to be calculated, and if the value does not appear, it will finally be recorded as 0. $\Gamma^{(8)}$ above corresponds to the operations of betweenness centrality for node 0 in the networks. The values in row i_0 and column i_0 are not included in the calculations. We do not consider the situation in which the path starts and ends overlap in calculating the shortest path, so we exclude the case of $i = j$.

3.3.2 Threshold Function

The threshold function is used to estimate the number of loops based on the average path length of the network. We define the threshold of the ER network as:

$$K_0 = \left\lceil \mu \times \frac{\ln n}{\ln \langle deg \rangle} \right\rceil. \quad (18)$$

$\lceil \cdot \rceil$ is the ceiling function, μ is defined as the tolerance coefficient ($\mu \geq 1$), $\langle deg \rangle$ represents the average degree of the network, and $\frac{\ln n}{\ln \langle deg \rangle}$ represents the average shortest path length in the network. $\langle deg \rangle$ can be expressed as:

$$\langle deg \rangle = p \times (n-1). \quad (19)$$

p represents the connection probability of any pair of nodes in the network.

The degree distribution of the BA network obeys $p(k) \sim k^{-\gamma}$, where γ is the power exponent. We define the threshold of the BA network as:

$$\begin{cases} K_0 = \lceil \mu \times \ln(\ln n) \rceil, & 2 < \gamma < 3 \\ K_0 = \left\lceil \mu \times \frac{\ln n}{\ln(\ln n)} \right\rceil. & \gamma = 3 \end{cases} \quad (20)$$

After adding the auxiliary functions, the maximum number of loops is reduced from $n-1$ to K_0, and the time complexity of the algorithm is also reduced. When the number of loops far exceeds K_0, special circumstances such as isolated nodes or disconnected subnets in the networks need to be considered. In these circumstances, there is no connected path between some nodes. Even if the operation is performed $n-1$ times, the matrix $\Gamma^{(n)}$ will not be filled, and the calculation can be ended earlier.

3.4 Parallel Computing Model of the Algorithm

We propose a parallel computing model named EVENT parallelism for betweenness centrality calculation, which aggregates the advantages of data parallelism and task parallelism. Event parallelism means dividing and conquering the target event and generating subevents, assigning subevents to each thread, selecting the running processor according to the characteristics of each program in the subevents, supporting data sharing and for load balancing across threads.

The betweenness centrality calculations of the nodes in the networks are independent of each other. We define the betweenness centrality calculation of the entire network as the target event and the betweenness centrality calculation of each node as a subevent. The calculation time is inversely proportional to the number of threads enabled, and thread conflicts are effectively avoided.

3.5 Optimized Matrix Algorithm

We achieve acceleration of the matrix algorithms through data precomputation and matrix block multiplication, thereby significantly reducing the time complexity of the algorithm.

3.5.1 Precomputation

The matrix $A^{(k)}$ is constant for nodes in the network, and the betweenness centrality calculation of each node requires the data of matrices $A^{(1)} \sim A^{(k)}$. We calculate this part of the data before dividing and conquering the event and then set it as shared data. These data do not need to be kept in memory; instead, they can be read from the disk when needed.

This reduces the number of matrix multiplication operations from $2k_0n$ to $k_0(n+1)$, when calculating the betweenness centrality of all nodes; here, k_0 represents the number of times the algorithm runs until the matrix $\Gamma^{(k)}$ is filled. This optimization does not fundamentally change the spatial complexity of the algorithm, but it significantly reduces its temporal complexity. For large-scale networks, we recommend storing this part of the data on disk to save the relatively limited available memory.

3.5.2 Matrix Block Multiplication

GPUs have significant advantages over CPUs in linear algebra calculations; however, there is an upper limit on the order of matrices that can be calculated with GPUs. We use the CPU to control the operation of subevents and complete the main calculations based on the GPU.

We adopt the Coppersmith-Winograd matrix multiplication algorithm to propose a novel implementation by GPUs[45],[46],[47]. This novel implementation uses 7 multiplications and 3 nested loops ($O(n^2)$) which are calculated by GPUs and CPUs, respectively, and it can be executed iteratively. The following is the pseudocode for the proposed novel implementation.

Algorithm 2 Matrix Block Multiplication

Input: $A^{(n)}, B^{(n)}$
Output: $C^{(n)}$

function MAIN($A^{(n)}, B^{(n)}, C^{(n)}$)
  parallel section 1:
  $S1 \leftarrow A21 + A22$, $T1 \leftarrow B12 - B11$
  $S3 \leftarrow A11 - A21$, $T3 \leftarrow B22 - B12$
  $S2 \leftarrow S1 - A11$, $T2 \leftarrow B22 - T1$
  parallel section 2:
  $M5 \leftarrow S1 \times T1$
  $M7 \leftarrow S3 \times T3$
  $M6 \leftarrow S2 \times T2$
  parallel section 3:
  $M1 \leftarrow A11 \times B11$
  $M2 \leftarrow A12 \times B21$
  parallel section 4:
  $C11 \leftarrow M1 + M2$, $U2 \leftarrow M1 + M6$
  $U3 \leftarrow U2 + M7$, $C22 \leftarrow U3 + M5$
  $U4 \leftarrow U2 + M5$, $S4 \leftarrow A12 - S2$
  $T4 \leftarrow T2 - B21$
  parallel section 5:
  $M4 \leftarrow S4 \times B22$
  $M3 \leftarrow A22 \times T4$
  parallel section 6:
  $C21 \leftarrow U3 - M4$, $C12 \leftarrow U4 + M3$
  return $C^{(n)}$
end function

We believe that this order of calculations achieves the best balance between temporal and spatial complexity. It requires $O(3n^2 + 7\left(\frac{n}{2}\right)^2)$ space and $O(n^{\omega})$ time, where $2.387 < \omega < 2.81$. Changing the order of the calculations may lead to calculation errors. The details of the impleme

ntation code are given in the appendix.

When the order of the matrix is odd, we add a new row and column to the matrix in which all values are set to 0. This operation, which requires at most $\log_2 n$ time, does not affect the result of matrix block multiplication and extends the applicability of the algorithm. The temporal complexity of the algorithm slightly increases to $O((n+\log_2 n)^{\omega})$.

4 Experimental Results

We analyze the experimental results in this section, which is divided into two subsections: the introduction of the experimental environment and parameters, the experimental results and analysis. The calculation of betweenness centrality for a network with millions of nodes and connection probabilities at the level of tens of percent can be achieved on a single-GPU system. The experimental results show that a multi-GPU system achieves nearly linear speedup.

4.1 Experimental Environment Introduction

We compiled the algorithmic solution and compared it with the code based on NetworkX. NetworkX is a Python package for complex networks analysis; it has excellent performance and a wide range of application scenarios[27]. We noticed that the maximum of matrices order in multiplications that TASLE V100 can support is 45000 when DAWN is running, and this upper limit is slightly floating. When the scale of the network is above this upper limit, matrix block multiplication becomes an inevitable choice. Table 1 shows some of the parameters of the test machine.

TABLE 1
Partial Parameter Table of the Test Machine

Hardware	Parameters
CPU	Intel Xeon Gold 6151(2)
RAM	1TB
GPU	NVIDIA TESLA V100(4)
IDE	Visual Studio 2019
OS	Windows Server 2019
Toolkit	CUDA 11.4

4.2 Results and Analysis

Unweighted undirected networks (UWD) and weighted directed networks (WD) are the models with the highest and lowest operating efficiency for Brandes' algorithm, respectively. We show the running time of NetworkX and DAWN on Erdős–Rényi networks and Barabasi-Albert networks with 10000 nodes. The unit for the running time is seconds, with 6 significant digits reserved. The time shown in the table is the average time of 5 tests.

TABLE 2
Calculation Time on Multiple Network Models

MODEL	Connection Probability	Brandes'	DAWN
UWD-ER	0.05	7196.63	15467.1
	0.1	16934.5	15694.3
	0.2	38541.3	15413.7
	0.5	97671.5	15573.4
WD-ER	0.05	41164.8	15854.2
	0.1	84756.4	15891.8
	0.2	181734	15803.0
	0.5	458414	15872.5
UWD-BA	0.05	17614.5	20874.3
	0.1	41169.7	17502.8
	0.2	83451.6	17279.6
	0.5	217589	15327.4

The scale, density and complexity of the networks will have a significant impact on the efficiency of Brandes' algorithm. As the connection probability increases, the calculation time of DAWN on the networks decreases. Katzav et al. proposed the phenomenon of network diameter shrinkage[48]. As the density increases, the diameter of the network shrinks and the calculation time decreases.

A. McLaughlin et al. studied how to accelerate Brandes' algorithm with GPUs and used the edge traversal speed to evaluate the algorithm performance[20]. Most of the data used in the research of A. McLaughlin et al. were represented by relatively sparse UWD networks. The public dataset released by Stanford University includes dense networks, such as: Social Circles（Google+）[49]; Email-Eu-core Temporal Network[50]; Dynamic Face-to-Face Interaction Networks[51]; Social Network(MOOC User Action Dataset)[52]; Disease-gene Association Network[53]; Enhanced Tissue-specific Gene Interaction Networks[54].

These dense networks have appeared in datasets in recent years, which, in some sense, indicates that dense networks have gradually become a research focus in the field of complex network analysis. Therefore, we believe that it is also necessary to test the performance of our algorithm on dense networks.

The results include experiments running on the UWD-ER model with a connection probability of 0.1. We show the betweenness centrality calculation time DAWN operations can be completed on a single-GPU system, and multi-GPU acceleration is useless when the number of nodes in the network is fewer than 2^{15}. We present separate calculation times for a network with a number of nodes in the range of 2^{10} to 2^{15}.

Figure 1 shows that DAWN achieves approximately linear speedup on the multi-GPU systems. A. McLaughlin et al. gave the calculation time of sampling and edge-parallel methods running on delaunay_n20 via 192 GPUs [20], [55], which is a random triangulation network with over a million nodes and an average degree of about 3. DAWN has performance advantages within the scope of the experiment and running on a Erdős–Rényi network with connection probability of 10%.

The distribution of the nodes and edges in a real network is uneven. We performed validation experiments using the real datasets mentioned in this section, and the results show that the computation times for some nodes exceed the values shown in Figure 1, while the times for other nodes are lower than these values. If the node to be calculated is an edge node or an isolated node in the network, the calculation time will drop significantly. The maximum value of the calculation time for a node is related to the network diameter, but this is only an individual special case.

In the experiments, DAWN shows three performance advantages:

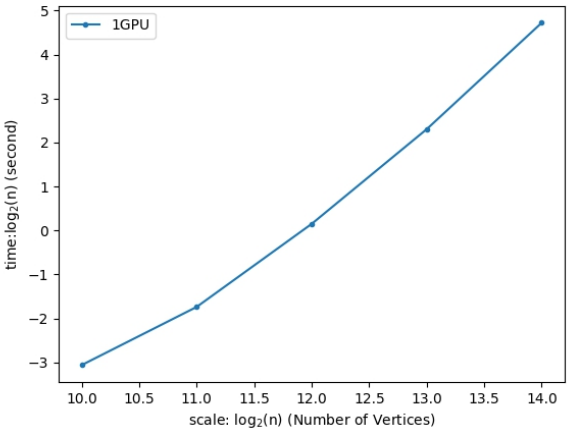

Fig. 1(a). Calculation Time of betweenness centrality on the single-GPU system

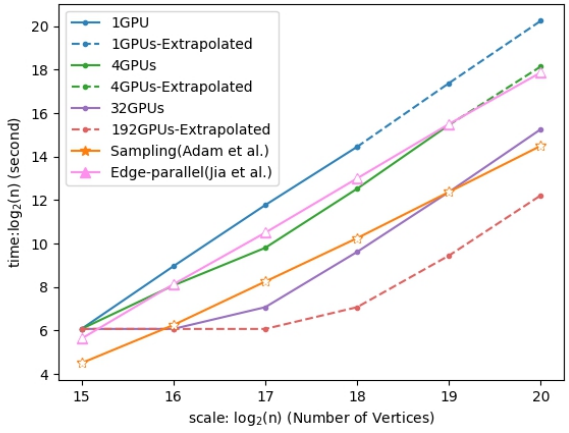

Fig. 1(b). Calculation Time of betweenness centrality on multi-GPU systems

(1) On networks with the same number of nodes, as the density increases, DAWN gradually shows greater advantages in computational efficiency;

(2) On networks with the same number of nodes, as the complexity increases, such as by becoming directed or weighted, DAWN has a more obvious advantage over Brandes' algorithm in terms of computational efficiency.

(3) For networks with an increasing number of nodes, DAWN can continue to process their adjacency matrices even on a single-GPU system and achieves approximately linear speedup on multi-GPU systems.

5 Conclusion

We propose a novel algorithm for betweenness centrality that uses a parallel computing method based on the adjacency matrix to improve the calculation efficiency. It addresses the insufficient calculation efficiency of Brandes' algorithm on dense networks with thousands of nodes and solves an urgent problem in the field of complex network analysis: how to compute the betweenness centrality index on large networks faster and more exactly. DAWN can be widely used in large-scale network application fields such as social network analysis, information analysis and recommendation on the internet, biological network structure analysis, and communication network vulnerability analysis.

References

[1] S. Jin, Z. Huang, Y. Chen, D. Chavarría-Miranda, J. Feo and P. C. Wong, "A novel application of parallel betweenness centrality to power grid contingency analysis," 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS), 2010, pp. 1-7.

[2] A. Tizghadam and A. Leon-Garcia, "Betweenness centrality and resistance distance in communication networks," IEEE Network, vol. 24, no. 6, pp. 10-16.

[3] Puzis R, Altshuler Y, Elovici Y, et al. Augmented betweenness centrality for environmentally aware traffic monitoring in transportation networks, Journal of Intelligent Transportation Systems, vol.17, no.1, 2013, pp. 91-105.

[4] Brandes U. A faster algorithm for betweenness centrality. Journal of mathematical sociology, vol.25, no.2, 2001, pp. 163-177.

[5] Mehmood A, Choi G S, von Feigenblatt O F, et al. Proving ground for social network analysis in the emerging research area "Internet of Things" (IoT). Scientometrics, vol. 109, no. 1, 2016, pp. 185-201.

[6] Tao Zhou, Wenjie Bai, Binghong Wang et al. Overview of complex network research, Physics, vol.34, no.01, 2003.

[7] Guo Lei, Xu Xiaoming, etc. Complex Networks, (Shanghai Science and Technology Education Press, Shanghai, ed. 1,2006), pp.1-15.

[8] Albert-László Barabási. Network Science, (The United Kingdom: Cambridge University press, Cambridge, ed. 1, 2014), pp.7-9.

[9] Freeman L C. Centrality in social networks conceptual clarification, Social Networks, vol.1, no.3, 1978, pp. 215-239.

[10] Bavelas A. Communication patterns in task-oriented groups, The journal of the acoustical society of America, vol.22, no.6, 1950, pp. 725-730.

[11] Freeman Linton C. A Set of Measuring Centrality Based on Betweenness, Sociometry, vol.40, 1977, pp. 35-41.

[12] Katz, L. A new status index derived from sociometric analysis, Psychometrika, vol.18, 1953, pp. 39-43.

[13] Broder, A. Z., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J. L. Graph structure in the Web. (Comput., Netw., ed.1 2000), pp. 309-320.

[14] K. Madduri, D. Ediger, K. Jiang, D. A. Bader and D. Chavarria-Miranda, "A faster parallel algorithm and efficient multithreaded implementations for evaluating betweenness centrality on massive datasets," 2009 IEEE International Symposium on Parallel & Distributed Processing, 2009, pp. 1-8.

[15] DARPA Information Processing Technology Office, High productivity computing systems project, 2004, http://www. highproductivity.org.

[16] Bader D.A., Kintali S., Madduri K., Mihail M. Approximating Betweenness Centrality, WAW 2007, Lecture Notes in Computer Science: vol. 4863, Springer (2007).

[17] Furno A., El Faouzi NE., Sharma R., Zimeo E. Fast Approximated Betweenness Centrality of Directed and Weighted Graphs, COMPLEX NETWORKS 2018: vol. 812, Springer (2019).

[18] D. A. Bader and K. Madduri, "Parallel Algorithms for Evaluating Centrality Indices in Real-world Networks," 2006 International Conference on Parallel Processing (ICPP'06), 2006, pp. 539-550.

[19] G. Tan, D. Tu and N. Sun, A Parallel Algorithm for Computing Betweenness Centrality, 2009 International Conference on Parallel Processing, 2009, pp. 340-347.

[20] A. McLaughlin and D. A. Bader, "Scalable and High Performance Betweenness Centrality on the GPU," SC '14: Proceedings of the International

Conference for High Performance Computing, Networking, Storage and Analysis, 2014, pp. 572-583.
[21] A. Davidson, S. Baxter, M. Garland and J. D. Owens, "Work-Efficient Parallel GPU Methods for Single-Source Shortest Paths," 2014 IEEE 28th International Parallel and Distributed Processing Symposium, 2014, pp. 349-359.
[22] Z. Shi and B. Zhang, "Fast Network Centrality Analysis using GPUs," BMC bioinformatics, vol. 12, no. 1, 2011, pp. 149.
[23] Y. Jia, V. Lu, J. Hoberock, M. Garland, and J. C. Hart, "Edge v. Node Parallelism for Graph Centrality Metrics," GPU Computing Gems, vol. 2, 2011, pp. 15–30.
[24] N. Edmonds, T. Hoefler and A. Lumsdaine, "A space-efficient parallel algorithm for computing betweenness centrality in distributed memory," 2010 International Conference on High Performance Computing, 2010, pp. 1-10.
[25] Erdős D, Ishakian V, Bestavros A, et al. A divide-and-conquer algorithm for betweenness centrality, Proceedings of the 2015 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2015, pp. 433-441.
[26] W. Wang and C. Y. Tang, "Distributed computation of node and edge betweenness on tree graphs," 52nd IEEE Conference on Decision and Control, 2013, pp. 43-48.
[27] Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, "Exploring network struture, dynamics, and function using NetworkX", in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008.
[28] H. Wang, C. Wang, N. Shu, C. Ma and F. Yang, "The Betweenness Identities and Their Applications," 2020 IEEE 20th International Conference on Communication Technology (ICCT), 2020, pp. 1065-1069.
[29] N. Shu, H. Wang, C. Wang, X. Zhu, C. Ma and C. Chang, "Global Betweenness Identities in the Networks," 2021 IEEE 21st International Conference on Communication Technology (ICCT), 2021, pp. 783-786.
[30] Bentert M, Dittmann A, Kellerhals L et al. An Adaptive Version of Brandes' Algorithm for Betweenness Centrality. Journal of Graph Algorithms and Applications, vol. 24, no.3, 2018, pp. 483-522.
[31] Matteo Riondato, Evgenios M. Kornaropoulos. Fast approximation of betweenness centrality through sampling, Data Min. Knowl. Discov, vol. 30. no.2, 2016, pp. 38-47.
[32] Seidman S B. Network structure and minimum degree[J]. Social networks, vol.5, no.3, 1983, pp. 269-287.
[33] Kitsap M. Gallos L K. Havlin S. et al. Identification of Influential Spreaders in Complex Networks [J] Nature Physics, vol.6, no.11, 2010, pp. 888-893.
[34] Neo4j inc. and Developers. Neo4j, https://www.github.com/neo4j/neo4j, 2021.
[35] Developer Nvidia. CUDA toolkit, https://developer.nvidia.com/zh-cn/cuda-toolkit, 2022.
[36] Wu Jiangxing, Qi Xiaofeng, Gao Yanzhao. Overview of Heterogeneous Computing Parallel Programming Models, Aerospace Shanghai, vol.38, no.3, 2021, pp. 1-11
[37] Wang Lei, Cui Huimin, Chen Li, et al. Research and Progress of Task Parallel Programming Model, Journal of Software, vol. 24, no.1, 2013, pp. 77-90.
[38] Erdős P, Rényi A. On random graphs, Publ. Math., vol. 6, 1959, pp. 290-296.
[39] Erdős P, Rényi A. On the evolution of random graphs, Publ. Math. Inst. Hung. Acad. Sci., vol. 5, 1960, pp. 17-60.
[40] Barabási A L, Albert R. Emergence of scaling in random networks, Science, vol. 286, 1999, pp. 509-512.
[41] Albert R, Jeong H, Barabási A L. Diameter of the world-wide web, Nature, vol. 401, 1999, pp. 130-131.
[42] Faloutsos M, Faloutsos P, Faloutsos C. On power-law relationships of the Internet toplogy, Comp. Cornm. Rev., vol. 29, 1998, pp. 251-262.
[43] Watts D J, Strogatz S H. Collective dynamics of 'small-world' networks [J]. Nature, vol.393, 1998, pp. 440-442.
[44] Amaral L, Scala A, Bathélémy M et al. Classes of small-world networks, (Proc. Natl. Acad. Sci. USA, 2000), vol. 97: pp.11149-11152.
[45] Strassen V. Gaussian elimination is not optimal, Numerische Mathematik, vol.13, no.4, 1969, pp. 354-356.
[46] Don Coppersmith, Shmuel Winograd. Matrix Multiplication via Arithmetic Progressions, Symb. Comput., vol. 9, 1990, pp. 251-280.
[47] Virginia Vassilevska Williams. Multiplying Matrices Faster Than Coppersmith-Winograd, Proceedings of the 44th Symposium on Theory of Computing, 2012, pp. 887-898.
[48] Katzav E, Nitzan M, Ben-Avraham D, et al. Analytical results for the distribution of shortest path lengths in random networks, Epl, vol.111, no.2: 26006, 2015.
[49] J. McAuley and J. Leskovec. Learning to Discover Social Circles in Ego Networks. NIPS, 2012.
[50] Ashwin Paranjape, Austin R. Benson, and Jure Leskovec. "Motifs in Temporal Networks." In Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017.
[51] S. Kumar, C. Bai, V.S. Subrahmanian, J. Leskovec. Deception Detection in Group Video Conversations using Dynamic Interaction Networks. 15th International AAAI Conference on Web and Social Media (ICWSM), 2021.
[52] S. Kumar, X. Zhang, and J. Leskovec. Predicting Dynamic Embedding Trajectory in Temporal Interaction Networks. ACM SIGKDD, 2019.
[53] The Comparative Toxicogenomics Database: update 2017. Davis, Allan Peter, et al. Nucleic Acids Research. 2016.
[54] Network Enhancement as a general method to denoise weighted biological networks. Bo Wang, Armin Pourshafeie, Marinka Zitnik, Junjie Zhu, Carlos D. Bustamante, Serafim Batzoglou, and Jure Leskovec. Nature Communications. 2018.
[55] D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner, Eds., Graph Partitioning and Graph Clustering - 10th DIMACS Implementation Challenge, ser. Contemporary Mathematics, vol. 588, 2013.

First A. Author All biographies should be limited to one paragraph consisting of the following: sequentially ordered list of degrees, including years achieved; sequentially ordered places of employ concluding with current employment; association with any official journals or conferences; major professional and/or academic achievements, i.e., best paper awards, research grants, etc.; any publication information (number of papers and titles of books published); current research interests; association with any professional associations. Author membership information, e.g., is a member of the IEEE and the IEEE Computer Society, if applicable, is noted at the end of the biography.

Second B. Author Jr. biography appears here.

Third C. Author biography appears here.