شسمین همایش بین‌المللی افق های نویـن در
مهندسی برق، کامپیوتـر و مکانیک
6th International Conference on the New Horizons in Electrical Engineering, Computer and Mechanical
www.mhconf.ir

d-central graph model for local cohesive sub graph discovery

Arezoo Jahani
Faculty of Electrical Engineering,
Sahand University of Technology,
Tabriz, Iran.
Email: a.jahani@sut.ac.ir

Abstract— **Structural cohesion is the sociological concept and it is defined as the minimal number of actors in a social network that needs to be removed to disconnect the group. The extraction of a cohesive subgraph is an important issue in social network analysis and other networks including computer and biological networks. Detecting cohesive subgraph is a graph partitioning problem and is so prominent in network analysis to discover network structure and predicate the behavior of the network components. Existing studies of this problem are classified in implicit and explicit methods. Explicit methods try to find subgraphs with special properties. Most dense subgraphs have overlap and can construct other subgraphs which are common between two dense subgraphs that existing studies cannot find these subgraphs. This paper proposes a new method to address this problem, the proposed method (d-central graph) is able to detect local cohesive subgraphs in weighted or unweighted graphs. We demonstrate a local cohesive subgraph is the largest central subgraph including a central node with a maximum diameter d. The proposed explicit method for detecting local cohesive subgraphs tries to detect the largest central subgraph including the central node and its best neighbor which have minimum common neighbors and the min-cut between the central node and its best neighbor can construct two subgraphs with approximately the same number of nodes. The experiments demonstrate the effectiveness of the proposed method in earning high centrality and low diameter compared with other methods on some network graphs. We explain another case study that uses the d-central graph to assign priority to data center network resources.**

Keywords- Virtual Networks; cohesive graph; local subgraph; diameter; network analysis.

1. Abstract

Network analysis to detect dense sub graphs is an important issue in social networks [1-5] and also in computer [6] and biological networks which has been widely used in a variety of domains includes community detection [7-9] in the social network, motif detection in DNA, finding correlated genes, epilepsy prediction, graph visualization, finding spam link farms and web mining. Most networks, originally modeled as graphs that graph nodes are the member/actor/component and graph links are interactions between nodes [10]. Every sub graph has been extensively used to diagnose community and cluster or groups to understand network structure and prediction actors behavior or deploying network by adding new landmark nodes on the network [11, 12].

Based on previous methods a cohesive sub graph is sub graph with minimal number of actors in a social network that need to be removed to disconnect the group and mostly cohesive sub graphs have the high ratio of edges to vertex [1, 13, 14]. This paper tries to find local cohesive [15] sub graph in an unweighted or weighted graph which is local sub graph with high centrality and low diameter. In fact, there are three indices to computing coherence of each sub graph [16]: (1) distance: the shortest path size of pair nodes in sub graph. (2) degree: the number of links that attached to a node. and (3) density: ratio of real number of links to possible maximum number of links. So, it is clear that clique [17] is the perfect model of cohesive sub graph [16]. A clique sub graph is a perfect cohesive sub graph with density equal to one, degree equal to node size minus one and distance of pair nodes equal to one [18]. Also, we can understand the maximum coherence is equal to one or 100%, the maximum amount of distance is one and the maximum amount of degree is node sizes minus one [19].

Existing methods of extracting cohesive sub graph were categories in two-part; implicit and explicit method [16]. Implicit methods try to extract sub graphs with partitioning graph blindly. Existing method of this category includes, min-cut [20-22], ratio cut [23, 24], normalize-cut [25], modularity [26], extracting [27] and graph scan [28] methods. Versus explicit methods try to find sub graphs with special features includes n-clique [29], k-plex [30], k-core [31, 32], $\rho -$dense [33], n-clan [34], n-club [34] and k-truss [35] methods. There are some hierarchical methods like quasi-clique merge [36] and density friendly graphs decomposition [37]. As previously mentioned most methods are able to find clique sub graphs using distance, degree or diameter indices [16]. But in real, networks have a million nodes and links that are connected to each other without forming any clique. So, finding dense sub graphs are still prominent issue [38], Because, each proposed method try to extract sub graph with special features [39, 40]. Most of existed methods detect dense sub graphs in a hierarchical mode which obtains large network diameter, like core-decomposition [10] method. The extracted sub graphs are mostly same and do not have any priority with each other. They are not started to partitioning through center of graph and cannot find dense sub graphs that are common between two complete sub graphs. In fact, detecting common node between two dense sub graphs was being ignored in previously methods and it is clear that common nodes between each dense sub graph can introduce important features of the actual network. In order to address this problem, we proposed a new method (d-central graph) for detecting local cohesive sub graph that uses centrality and diameter indices in combination. Also, limits the detected sub graph size of the utmost diameter to finding local cohesive sub graph.

The rest of paper is organized as follows. In the second section we discussed related works. In section 3, the preliminaries of problem introduces with detail. The proposed method (d-central graph) for computing for detecting local cohesive sub graphs are described in section 4. Section 5 discussed experiment results and a case study. Section 6 includes conclusion and future works.

2. Related Work

Based on previous methods a cohesive subgraph is a subgraph with high density (ratio of links to nodes). Existing methods for detecting cohesive subgraphs have two categories; implicit and explicit [16]. In the implicit methods, the graph has been divided into many subgraphs with high inner density and low outer density. In the explicit methods, subgraphs with special features and high density should be extracted from the real graph. All of the proposed methods are using one of the three indices, including; distance, degree, and density [16]. Min-cut [6][22], ratio-cut [7], normalized-cut [8], modularity [9], extraction [10] and graph-scan [11] are implicit method. Min-cut [6] is a partition of the nodes into two groups *A* and *B* (that is, $V=A\cup B$ and, $A\cap B=\emptyset$) so that the number of edges between *A* and *B* is minimized. Ratio-cut [7] tries to divide the graph into two subgraphs *A* and *B*, with minimum $e(A,B)/|A|.|B|$. Normalize-cut [8] is a partitioning of nodes into two subgraphs with the normal size of nodes in each subgraph. Modularity [9] is expressed in terms of the eigenvectors of a characteristic matrix for the network and leads to a spectral algorithm for community detection. An extraction [10] can extract one community at a time, allowing for the arbitrary structure of the rest of the network. Graph-scan [11] uses Poisson discrepancy, for clusters in graphs, derived from spatial scan statistics on point sets.

The n-clique [12], k-plex [13], k-core [14], ρ-dense [15], n-clan [16], n-club [16], k-truss [17], local dense [15], quasi clique merge (QCM) [18], density friendly [19] and cluster coefficient are explicit methods. Each explicit method introduces some special features of subgraphs and then tries to extract subgraphs with a high ratio of features. For example, n-clique [12] extracts a maximal subgraph in which every pair of vertices is connected by a path of length *n* or less. k-plex [13] extracts maximal subgraph with the following property: each vertex of the induced subgraph is connected to at least *n-k* other vertices, where *n* is the number of vertices in the induced subgraph. The basic algorithm is a depth-first search. k-core [14] extracts maximal subgraph of *G* in which all vertices have degree at least k and do this work repeatedly to extract hierarchical cores. ρ-dense [15] computes the density of all possible subgraphs and extracts subgraphs with high density hierarchically. Clique is a 1-dense subgraph. An n-clan is an n-clique that has a diameter less than or equal to *n* as an induced subgraph. These are found by using the n-clique routine and checking the diameter condition. n-club [16] tries to extract the largest subgraph with a diameter equal to one. k-truss [17] is the largest subgraph in which each pair of nodes has a *k-2* common neighbor node. For example, 4-node-clique is a 4-truss subgraph. Density friendly computes density of all possible subgraphs and extracts high dense subgraphs hierarchically. ρ -local [19] dense uses ρ -dense with adding distance indexes to be suitable for periphery graphs, Because ρ-dense suffers from resolution limit in large graphs. QCM introduces a method for computing density in the weighted graph as the sum of weights of links to the maximum possible number of links.

All existing approaches use three indices include; distance, degree, and density. Table 1. Demonstrates all previous works and the proposed method and their ability in detecting some structures of graphs. As mentioned in Table 1, there is not any approach to detecting the ability of cohesive subgraphs for all kinds of structures and for weighted or unweighted graphs. But the proposed method is used to run on all kinds of structures and in both unweighted and weighted graphs. For example, k-core is able to detect subgraphs with ring structure or $\rho-$ dense core is able to detect subgraphs with complete and core of core structures. Previously approaches suffer from resolution limit means that large graphs without any clique cannot find suitable subgraphs and sometimes the extracted subgraphs are large as the whole graph. Also in large graphs, there are many common clique subgraphs, but existing approaches could not find these subgraphs. Hierarchical extracting can find suitable and dense subgraphs, but in the hierarchical model, the last found subgraphs are so big and have a high diameter that in this paper we want to limit the diameter of extracted subgraphs and extract only local cohesive subgraphs. Also, we want to find common nodes of pair clique subgraphs as a new subgraph.

Table 1. The categories of related works and proposed work in three indices for extracting the cohesive sub graphs and their ability in detecting the structures of graph

Structure	**Indices and related work**						
	Distance		**Degree**		**Density**		**Centrality & diameter**
	n-clique (1950)	**n-clan (1979) n-club (1979)**	**k-plex (1981) k-core (1983)**	**k-truss (2009)**	**$\rho-$ dense (1984)**	**$\rho-$ dense core (2016)**	**d-central graph (proposed)**
unweighted (uw) or weighted (w)	**uw**	**uw**	**uw**	**uw**	**uw**	**uw**	**uw/w**
Star	-	✓	-	-	-	-	✓
Ring	-	-	✓	✓	✓	-	✓
Complete	✓	✓	-	-	✓	✓	✓
Tree	-	-	-	-	-	-	✓
Core of core	-	-	-	-	-	✓	✓

3. Preliminaries

Let *G=(V,E)* be an undirected and unweighted graph with set *V* as vertices and set *E* as edges. If graph *G* has *n* vertices and *m* edges, so each subgraph of *G* like *X* that $X \subseteq V$, has $E(X) = \{(u, v) \in E \mid u, v \in X\}$. i.e., the edges of *G* that have both endpoints in *X*. For two non-overlapping subgraphs *X* and *Y*, the *cross edge* between *X* and *Y* is defines as Eq.1.

$$\mathrm{E_{cross}}(\mathrm{x}, \mathrm{y}) = \{(\mathrm{X}, \mathrm{Y}) \in \mathrm{E} \mid \mathrm{x} \in \mathrm{X}, \quad \mathrm{y} \in \mathrm{Y}\} \qquad \text{Eq. 1}$$

In Eq. 1, $\mathrm{E_{cross}}(\mathrm{x}, \mathrm{y})$ is the *cross edge* between subgraph *X* and *Y*. Each graph has a central node that is the most important node based on some indices and is important in some applications including identifying the most influential person(s) in a social network [1], key infrastructure nodes on the Internet or urban networks, and super-spreaders of disease. Centrality concepts are an important issue in social networks and are computed with indices including betweenness centrality, closeness centrality, eigenvector centrality, degree centrality, harmonic Centrality, and Katz centrality of the same graph [41].

In this paper, the centrality computed with closeness index is suitable. In a connected graph, the normalized closeness centrality (or closeness) of a node is the average length of the shortest path between the node and all other nodes in the graph. Thus the more central a node is, the closer it is to all other nodes [42, 43]. The closeness of a vertex *v* in a graph *G=(V,E)* with *V* vertices is computed as Eq. 2.

$$C(v) = \frac{1}{\sum_y d(y, \mathrm{v})} \qquad \text{Eq. 2}$$

In Eq. 2, *C(v)* is closeness centrality of vertex *v*, $\mathrm{d}(y, \mathrm{v})$ is the shortest path between vertices *v* and *y*. However, when speaking of closeness centrality, people usually refer to its normalized form, generally given by the previous formula multiplied by *N-1*, where *N* is the number of nodes in the graph. This adjustment allows comparisons between nodes of graphs of different sizes. Calculating the closeness of all vertices in a graph involves calculating the shortest paths between all pairs of vertices on a graph, which requires $\mathrm{O}(\mathrm{V}^3)$ time with the *Floyd–Warshall algorithm* [44]. However, on sparse graphs, Johnson's algorithm [45] may be more efficient, taking $\mathrm{O}(\mathrm{V}^2 \log \mathrm{V} + \mathrm{V\,E})$ time.

4. d-central graph for detecting local cohesive sub graph

Finding cohesive subgraphs is widely used in many applications like graph task applications. In this section, we want to propose a way to detect local dense subgraphs that is an explicit way and tries to detect subgraphs with special features. In fact, the proposed method detects the largest central subgraph including the central node with a given diameter *d*.

The proposed way tries to start almost the center of the graph and limits the scattering of the graph with a variable as a maximum number of sub graph diameters. In fact, the diameter of the sub graphs must not exceed a certain value (such as *d* as Diameter of sub graph). Also detected sub graphs are not the same in closeness and centrality and it is clear that first found sub graphs have high centrality and the lasts have the least centrality. Whereas, in the existing methods, the extraction sub graphs did not prioritize each other.

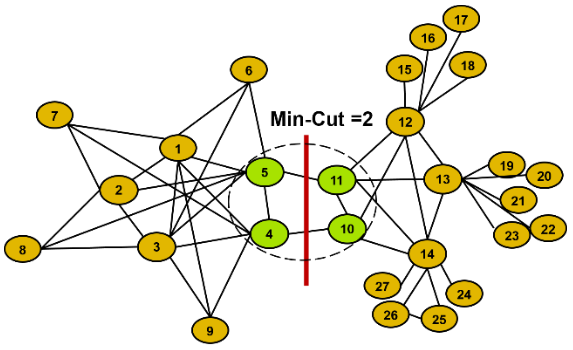
The reason for detecting such subgraphs is that always central nodes of each graph (like social or network) have important information and missing each central node can destroy to network structure. If we have a social network, central nodes are effective actors that play important role in network behavior, and detecting such actors will be so implanted in future applications, like finding important actors or finding connector people or discovering the criminals who are involved everywhere and with all people trying to sabotage. On the other hand, if we have a computer network that includes data centers or servers with a wire/wireless connection between them. In such a network, finding central nodes will work to find essential and critical resources. Therefore, using such resources cannot be used to the extent possible, because they are the whole network connector. If the network was a biological network, finding the central nodes in finding the connecting cells is useful. It can also be useful in finding the original structure of a texture.

So, it is understood that the first step in finding local cohesive subgraphs is finding the central node of the graph which can be applied using Eq. 2 [20-22] by running the Floyd algorithm to compute all shortest paths between all pairs of nodes. For ease of reference, the symbols used in this section are listed in Table 2. As shown in Table 2, used symbols have descriptions that shows their applications.

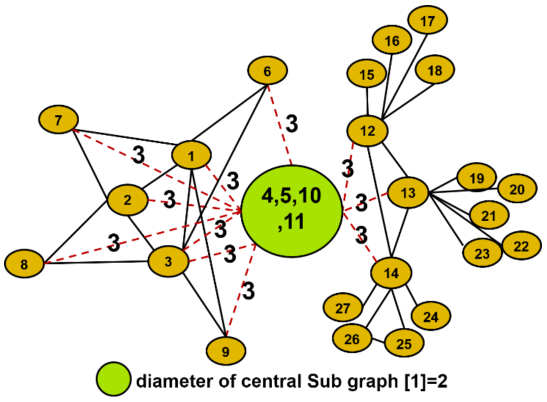
Table 2. Symbols used in this article

Symbol	Description
$\mathrm{d}(\mathrm{x},\mathrm{y})$	shortest path between nodes *x* and *y*
cn	central node
neig(cn)	neighbours of *cn*
$\mathrm{CN}(\mathrm{x},\mathrm{y})$	common neighbours between *x* and *y*
bn_{cn}	best neighbour of *cn*
ce	cross edge
tcs	temporal central subgraph
diameter(x)	diameter of subgraph *x*
sv	source vertices
dv	destination vertices

After finding the central node of the graph, we should find one neighbour of the central node with a minimum number of common neighbours and mark it as the best neighbour. So try to find Min-cut between the central node and its best neighbour provided that two partitions have almost the same node number. If the extracted sub graph has a diameter smaller than or equal to defined diameter *d*, mark it as a first central sub graph and update the graph *G* by adding a new node instead of all central sub graph's nodes and adding the diameter of central sub graph to all edges related to central sub graph nodes and other nodes. An example of finding the first central sub graph is demonstrated in Fig. 1(a).



(a) the primary graph and found center subgraph



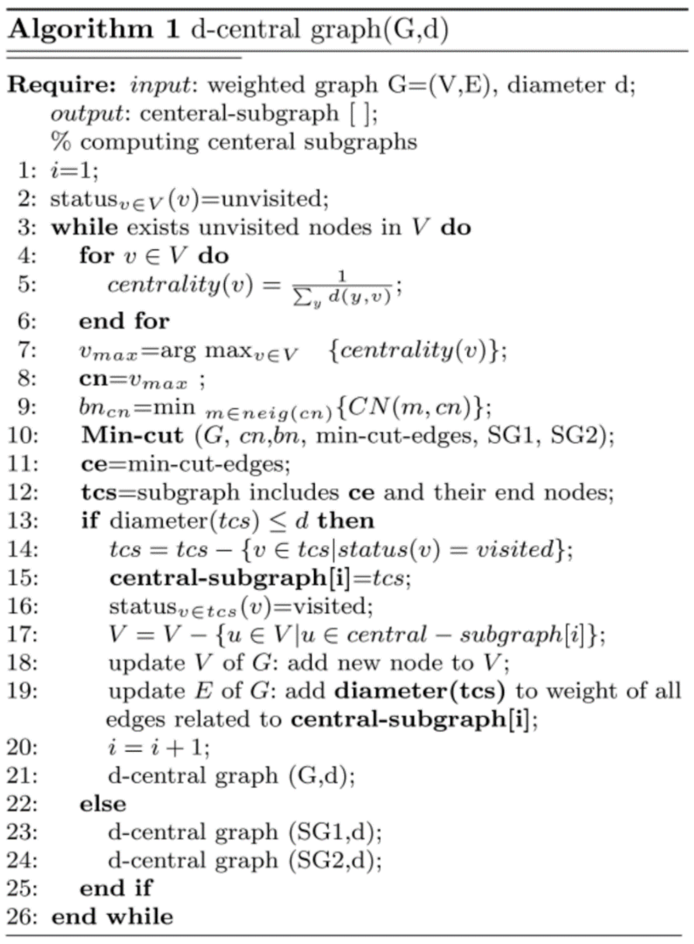
(b) changed graph after finding central subgraph

Fig. 1: An example of Algorithm \ref{alg:alg1} for finding centeral sub graph and Algorithm 3 for finding d-central graph

As shown in Fig. 1(a), the central node with maximum Closeness centrality is node 11 and a neighbor of the central node with minimum common neighbors is node 5. because the central node has five neighbors (5, 10, 12, 13 and 14) and the common neighbors between the central node and each neighbor are respectively as (0, 2, 3, 2, 3) that node 5 is the proper node with the minimum number of common neighbors. Finding Min-cut between (11,5) takes cross edges includes (v_5, v_{11}) and (v_4, v_{10}). So the subgraph contains $(v_4, v_5, v_{10}\ and\ v_{11})$ is central subgraph and the diameter of central subgraph is two that is acceptable diameter. So as shown in Fig. 1(b) we should add a new node instead of a central subgraph and connect all edges related to the central subgraph to their connected nodes as demonstrated in Fig 1(b) with adding a new node $(v_{4,5,10,11})$.This process is recursively repeated until all nodes are visited. The working process is displayed in Algorithm 1.

Algorithm 1, finds a central subgraph with computing closeness centrality (row 4) and finding a central node (row 7). Then algorithm tries to find a neighbour node with the minimum number of common neighbours (row 8) and so recall the Min-cut function to compute minimum cutting edges including the edge between the central node and its best neighbour (row 9). Note that the Min-cut algorithm will be explained in detail in the next subsection. After finding cross edges, tries to find a central subgraph that is all vertices and edges of cross edges and if the diameter of the central subgraph is an acceptable diameter (row 12), mark the extracted subgraph as a central sub graph[i] (row 14) and convert its nodes status to visited (row 15) and update graph nodes and edges with adding a new node instead of central subgraph nodes rows (16-18). If the diameter of the central subgraph is greater than the given value (row 21) recall recursively the algorithm for two subgraphs generated by cross edges (row 22-23).

Algorithm 1, uses an advanced Min-cut function for finding cross edges between two lists as source vertices and destination vertices that are explained in Algorithm 2.



Algorithm 1 d-central graph(G,d)

```
Require: input: weighted graph G=(V,E), diameter d;
    output: centeral-subgraph [ ];
    % computing centeral subgraphs
 1: $i$=1;
 2: status$_{v\in V}(v)$=unvisited;
 3: while exists unvisited nodes in $V$ do
 4:     for $v \in V$ do
 5:         $centrality(v) = \frac{1}{\sum_y d(y,v)}$;
 6:     end for
 7:     $v_{max}$=arg max$_{v\in V}$  $\{centrality(v)\}$;
 8:     cn=$v_{max}$ ;
 9:     $bn_{cn}$=min $_{m\in neig(cn)}\{CN(m, cn)\}$;
10:     Min-cut ($G$, $cn$,$bn$, min-cut-edges, SG1, SG2);
11:     ce=min-cut-edges;
12:     tcs=subgraph includes ce and their end nodes;
13:     if diameter($tcs$) $\leq d$ then
14:         $tcs = tcs - \{v \in tcs | status(v) = visited\}$;
15:         central-subgraph[i]=$tcs$;
16:         status$_{v\in tcs}(v)$=visited;
17:         $V = V - \{u \in V | u \in central - subgraph[i]\}$;
18:         update $V$ of $G$: add new node to $V$;
19:         update $E$ of $G$: add diameter(tcs) to weight of all
            edges related to central-subgraph[i];
20:         $i = i + 1$;
21:         d-central graph (G,d);
22:     else
23:         d-central graph (SG1,d);
24:         d-central graph (SG2,d);
25:     end if
26: end while
```

Algorithm 2 Min-cut(G,sv, dv,min-cut-edges, $SG1$, $SG2$)

```
Require: input: weighted graph G=(V,E), $sv$, $dv$;
    output: min-cut-edges, SG1, SG2;
    % computing min_cut of graph $G$
 1: while $| sv | \geq 2$ do
 2:     pick $(u, v)$ randomly from sv;
 3:     add the list of edges of $v$ to the list of edges of $u$;
 4:     update $E$;
 5:     delete all edges of $v$ from $E$;
 6:     delete node $v$ from $V$;
 7: end while
 8: while $| dv | \geq 2$ do
 9:     pick $(u, v)$ randomly from dv;
10:     add the list of edges of $v$ to the list of edges of $u$;
11:     update $E$;
12:     delete all edges of $v$ from $E$;
13:     delete node $v$ from $V$;
14: end while
15: while $| V | \geq 2$ do
16:     pick $x$ randomly from $V$;
17:     d-sv=$\sum_{y \in sv} short-path(x, y)$;
18:     d-dv=$\sum_{y \in dv} short-path(x, y)$;
19:     if d-sv $\leq$ d-dv then
20:         add the list of edges of $x$ to the list of edges of $sv$;
21:         update $E$;
22:         delete all edges of $x$ from $E$;
23:         delete node $x$ from $V$;
24:     else
25:         add the list of edges of $x$ to the list of edges of $dv$;
26:         update $E$;
27:         delete all edges of $x$ from $E$;
28:         delete node $x$ from $V$;
29:     end if
30: end while
31: min-cut-edges=$ce$ between $sv$ and $dv$;
32: SG1=sub graph of $sv$;
33: SG2=sub graph of $dv$;
```

As shown in Algorithm 2, we approved the Min-cut function to extract minimum cut edges in the presence of two lists that should not be on the same side of the Min-cut algorithm and the proposed Min-cut algorithm should be able to extract cross edges between two given lists. In order to find Min-cut edges in a graph G with V vertex and E edge, the cross edges with the minimum number of edges among all cuts in a graph, determine the edge connectivity of the graph. Karger's algorithm provides an efficient randomized method for finding this cut [22]. We developed Karger's algorithm by adding a new limitation (two created subgraphs must have the almost the same number of nodes) and describing it in Algorithm 2. An example of finding the almost center of graph demonstrated in Fig. 1(a) and Fig. 1(b).

As illustrated in Algorithm 2 , Algorithm tries to find Min-cut in which two detected subgraphs have almost the same vertices. For this purpose, at first, tries to merge all vertices in the source (rows 1-7) and destination list (rows 8-14). Then for other unvisited vertices, tries find the best list. The best list for each node is the list that has the nearest nodes to that node. For doing this work computes the sum of the shortest path of the mentioned node to two lists vertices (rows 17-18). So each one has a smaller amount, which adds to that list (rows 19-29). At last, min-cut is the cross edges between two detected lists (row 31).

If we run the proposed d-central graph method on the graph in Fig 1, we reach the sub graphs that are shown in Fig. 2 and Fig. 3 which has seven sub graphs with maximum diameter *d=3* and the central sub graph is$(v_4, v_5, v_{10}, v_{11})$.

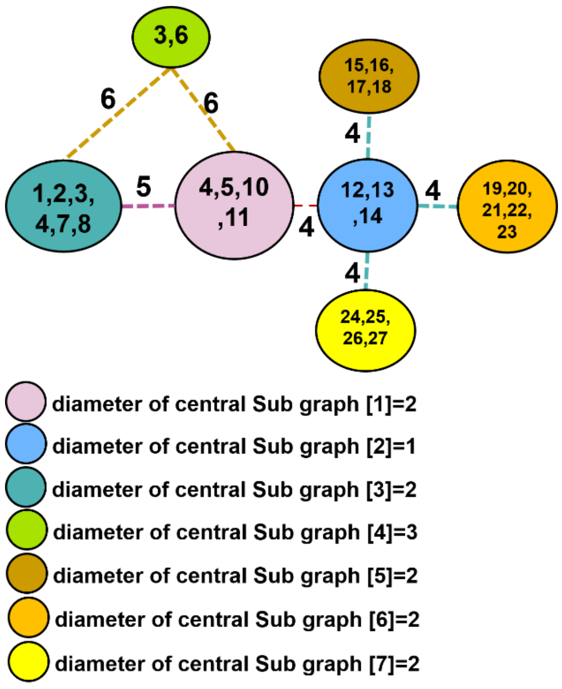

3,6
6
6
15,16, 17,18
4
1,2,3, 4,7,8
5
4,5,10 ,11
4
12,13 ,14
4
19,20, 21,22, 23
4
24,25, 26,27
diameter of central Sub graph [1]=2
diameter of central Sub graph [2]=1
diameter of central Sub graph [3]=2
diameter of central Sub graph [4]=3
diameter of central Sub graph [5]=2
diameter of central Sub graph [6]=2
diameter of central Sub graph [7]=2

Fig. 3 An example of proposed d-central graph model for detecting local cohesive subgraphs with *d=3* $(v_4, v_5, v_{10}, v_{11})$ are central sub graph and the algorithm find six other central sub graphs)

5. Experiment results

5.1. Effectiveness of d-central graph

In order to investigate the effectiveness of the suggested method, we first examined three important features of the proposed method compared with previous works shown in Table 3. We analyzed previous explicit methods include; n-clique, n-clan, n-club, k-plex, k-core, k-truss, ρ-dense and ρ-dense core and compare their operation with proposed d-central graph on a simple graph *G* with *|V|=20* and *|E|=100* and maximum *degree=8* that generated with Inet tools [45, 46].

As illustrated in Table 3, the first row shows "whether the subgraphs formed by the division of the graph have priority over each other or not?" and as shown, core decomposition and k-plex and proposed approach are able to detect subgraphs with priority. the second row of the table shows their ability in extracting local sub graphs and the third row shows their ability to start the center of the graph because the center of a graph is so important in partitioning and can have more information.

Table 3 Extracted sub graphs features

All methods	**n-clique n-clan n-club**	**k-plex k-core**	**k-truss**	**ρ-dense ρ-dense core**	**d-central graph (proposed)**
Ability to extracting sub graphs with priority	-	✓	-	✓	✓
Ability to extracting local sub graphs	✓	-	-	✓	✓
Start to partitioning from graph center	-	✓	-	-	✓

5.2. Running time

This section reports the running time of the proposed algorithm compared with k-core and n-clique algorithms on three datasets. We perform our evaluation on three graphs of different sizes and structures. We select three datasets with characteristics described in Table 4 and their description is as below which These datasets are obtained from UCIrvine Network Data Repository.

- Dolphins: an undirected social network of frequent associations between dolphins in a community living off Doubtful Sound in New Zealand.
- karate: the social network of friendships between members of a karate club at a US university in the 1970.
- lesmis: co-appearance of characters in Les Miserables novel by Victor Hugo.

We apply d-central graph, k-core decomposition, and n-clique to every dataset. In fact, we used a computer with 3 GHz Intel Core i5 and 8 GB of RAM. The results are also shown in Table 4. As expected, the d-central graph has a running time equal to the running time of Floyd–Warshall algorithm for all of the vertices of the graph, because the other calculations do not need any time. So, the running time of Floyd–Warshall algorithm is $O(V^3)$ and is fast enough as expected. The last row in Table 4 shows the running time of each method and as shown the running time of the proposed d-central graph is $O(V^3)$ because of the need to compute the shortest path for all vertices in Algorithm 1 and this is same as n-clique running time. Because the n-clique needs to know the distance of each pair of nodes to detect cohesive subgraphs. The running time of k-core is smaller than our proposed method, But k-core cannot give priority to extracted subgraphs and cannot detect local subgraphs, Because uses only distance index in cohesive subgraph detecting.

Table 4 Basic characteristics of the datasets and the running times of the algorithms

Name			Running time		
	|V|	|E|	d-central	k-core	n-clique
dolphins	62	159	2 ms	2 ms	2 ms
karate	34	78	1 ms	1 ms	2 ms
lesmis	77	254	2 ms	1 ms	3 ms
Running time			$O(V^3)$	$O(V^2)$	$O(V^3)$

5.3. Cohesive measurement

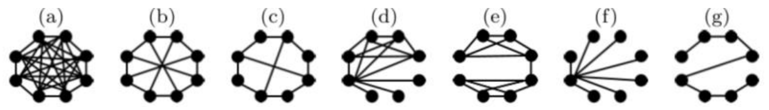
As mentioned in the introduction, a clique subgraph is a complete subgraph with high cohesion, because has a large density, large degree, and short diameter. In order to measure cohesion of different structures of graphs, we designed seven graphs in such a way that their cohesion is reduced respectively and compute their cohesion by three existed indices and proposed centrality index (d-central graph) to show the proposed method which uses centrality index is a suitable way to detecting cohesive subgraphs. The designed graphs are demonstrated in Table 5 and the cohesion of each subgraph demonstrated too.

As illustrated in Table. 5, the cohesion of subgraphs is computed by three indices and our propose index as centrality of graph nodes. A sub graph with high density, high degree and low distance is a cohesive sub graph and clique is a complete sub graph. In Table 5 the first row shows a clique and seven different graph structures. We compute the diameter of graphs and as shown the diameter index could not compute true numbers to graphs compared to each other's. Because graph (d) is more cohesion than graph (e), but their cohesion numbers are 2 and 3 respectively which is not true. In degree index, graphs with high degree are cohesion, but graph (d) and graph (e) have 1 and 3 cohesion number respectively that in fact graph (d) must has a high cohesion and high degree. The density and $\rho - dense$ core index also cannot find proper cohesion for graphs. The proposed d-central graph can find proper cohesion rate for graphs. As shown graph (a) has high cohesion rate and other graphs have a lower cohesion, respectively. Note that the cohesion rate in d-central graph is computed by Eq. 3.

$$\mathrm{cohesive(G)} = \sum_{\mathrm{v} \in \mathrm{V}} \sum_{\mathrm{u} \neq \mathrm{v} \in \mathrm{V}} \mathrm{d(u, v)} \qquad \text{Eq. 3}$$

In Eq. 3, *G* is a graph with *V* vertex and *E* edge. *d(u,v)* is shortest path between node *u* and *v*.

Table 5 Three cohesive measures in one clique and six graphs. A graph is more cohesive with a larger density, a larger degree, a shorter diameter, or a higher index of $\rho - dense$ core [16].



	(a)	(b)	(c)	(d)	(e)	(f)	(g)
Cohesive measure							
Diameter	1	2	3	2	3	2	7
Minimum degree	7	3	2	1	3	1	1
Density	1	12/28	10/28	13/28	13/28	7/28	7/28
Index of $\rho - dense$ core	1	1/3	1/4	1/7	1/16	1/7	1/16
d-central graph (proposed)	∼=1	0.72	0.66	0.63	0.628	0.53	0.396

6. Conclusion

In this paper, we have proposed a novel equation for computing density in weight graph and a new approach to detect local cohesive subgraphs gradually in a graph. A cohesive subgraph is defined as the minimal number of actors in a social network that needs to be removed to disconnect the group. we described the local cohesive subgraph which is cohesive and has a maximum *d* diameter. The proposed weight density is useful in all previous works for computing density in weighted graphs. Existed methods for computing density were not suitable for weighted graphs and this claim is stated in the article. Also proposed way to detect cohesive subgraphs (d-central graph) is able to detect desired subgraphs and start to detect from the center of the graph and is able to detect subgraphs with priority. Existed methods for extracting cohesive subgraphs could not detect subgraphs between other cliques/complete subgraphs, but the proposed d-central graph starts from the graph center and is able to detect all local cohesive subgraphs. The proposed method is local because tries to detect subgraphs with a maximum diameter d. We also explain a case study that uses the proposed d-central graph to assign priority to network resources in the data center network. Future works include improvement of scalability of the proposed d-central graph and investigation its usefulness in real applications like community detection and understanding network structure such as data center network or wireless sensor network.

References

[1] Z. Ertem, A. Veremyev, S. Butenko. Detecting large co-hesive subgroups with high clustering coe_cients in social networks. *Social Networks*, vol. 46, pp. 1-10, 2016.

[2] Y. Zhao, E. Levina, J. Zhu. Community extraction for social networks. *Proc Natl Acad Sci USA*. vol. 108, no. 18, pp. 7321-7326, 2011.

[3] H.R. Konstantin Andreev. Balanced graph partitioning. *sixteenth annual ACM symposium on Parallelism in algorithms and architectures*. Barcelona, Spain, pp. 120-124,2004.

[4] S. Fortunato. Community detection in graphs. *Physics Re-ports*. vol. 486, no. 3-5, pp. 74174, 2010.
[5] V. Batagelj, M. Zavernik. Fast algorithms for determining (generalized) core groups in social networks. *Advances in Data Analysis and Classi_cation*, vol. 5, no. 2, pp. 129-145, 2010.
[6] Z.G. Ding, D. J. Du, M. Fei. An isolation principle based distributed anomaly detection method in wireless sensor networks. *International Journal of Automation and Computing*. vol. 12, no.4, pp. 402412, 2015.
[7] J.C. Urschel. Nodal decompositions of graphs. *Linear Algebra and its Applications*. vol. 539, pp. 60-71, 2018.
[8] X. Qi, W.Tang, Y.Wu, G. Guo, E. Fuller, C.Q. Zhang. Optimal local community detection in social networks based on density drop of subgraphs. Pattern Recognition Letters. vol. 36, pp. 46-53, 2014.
[9] Aigner-Horev, Elad, Oran Danon, Dan Hefetz, and Shoham Letzter. Small rainbow cliques in randomly perturbed dense graphs. European Journal of Combinatorics 101, 103452, 2022.
[10] J. Li, C.D. Wang, P. Li, J.H. Lai. Discriminative metric learning for multi-view graph partitioning. Pattern Recognition. vol . 75, pp. 199-213, 2018.
[11] X. Liu, Y. Zhoua, X. Guana, C.Shen. A feasible graphpartition framework for parallel computing of big graph Knowledge-Based Systems. vol. 134, pp. 228-239, 2017.
[12] Hayashi, Kazuki, and Makoto Ohsaki. "Graph-based reinforcement learning for discrete cross-section optimization of planar steel frames." Advanced Engineering Informatics 51, 101512, 2022.
[13] M. Daniel Freund, D. Reichman. Contagious sets in dense graphs. European Journal of Combinatorics. vol. 68, pp. 66-78, 2018.
[14] R.S. Sinkovits, R.S.Sinkovits, J. Moody, B.T. Oztan, D.R. White. Fast determination of structurally cohesive subgroups in large networks. *Journal of Computer Science*. vol. 17, no. 1, pp. 62-72, 2016.
[15] L. Qin, R.H. Li, L. Chang, C. Zhang . Locally Densest Subgraph Discovery. *in Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '15*. pp. 965-974, 2015.
[16] S. Koujaku, S. Koujaku, L.Takigawa, M. Kudo, H. Imai. Dense core model for cohesive subgraph discovery. Social Networks. vol. 44, pp. 143-152, 2016.
[17] R. Luo. The maximum number of cliques in graphs without long cycles. Journal of Combinatorial Theory, Series B. vol. 128, pp. 219-226, 2018.
[18] D. Ben Barber, A. Lo, R. Montgomery, D. Osthus. Fractional clique decompositions of dense graphs and hyper graphs. Journal of Combinatorial Theory, Series B. vol. 127, pp. 148-186, 2017.
[19] H.P. Balister, R. Schelp. Decompositions of graphs into cycles with chords. Journal of Combinatorial Theory, Series B vol. 128, pp. 47-65, 2018.
[20] F.W. M. Stoer. A simple min cut algorithm. Journal of the ACM (JACM). vol. 44, no. 4, pp. 585-591, 1997.
[21] Yu, Wenwu, Hongzhe Liu, Wei Xing Zheng, and Yanan Zhu. "Distributed discrete-time convex optimization with nonidentical local constraints over time-varying unbalanced directed graphs." Automatica 134, 109899, 2021.
[22] R. David, C.S. Karger. A new approach to the minimum cut problem. *Journal of the ACM (JACM)*.vol. 43, no. 4, pp. 601-640, 1996.
[23] L. Hagen, A. Kahng, New spectral methods for ratio cut partitioning and clustering. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*. vol. 11, no. 9, pp. 10741085, 1992.
[24] W.F. Pan,B. Jiang, B. Li, Refactoring Software Packages via Community Detection in Complex Software Networks. *International Journal of Automation and Computing*. vol. 10, no. 2, pp. 157166, 2013.
[25] J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell*. vol. 22, no. 8, pp. 888905m 2000.
[26] M. Newman, M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E 69, 026113*. 2004.
[27] H. Hakeem. Layered software patterns for data analysis in big data environment. *International Journal of Automation and Computing*. vol. 14, no. 6, pp. 650660, 2017.
[28] R. Luce. Connectivity and generalized cliques in socio metric group structure. *Psychometrika*. vol. 15, no. 2, pp.169190, 1950.
[29] S.B. Seidman, B.L. Foster. A graph-theoretic generalization of the clique concept. *J. Math. Sociol.*. vol. 6, no. 1, pp. 139154, 1978.
[30] S. Nagaraju, M. Kashyap, M. Bhattachraya. An e_ectivedensity based approach to detect complex data clusters using notion of neighborhood di_erence. *International Journal of Automation and Computing*. vol. 14, no. 1, pp. 5767, 2017.
[31] S. Nagaraju, M. Kashyap, M. Bhattachraya. An e_ectivedensity based approach to detect complex data clusters using notion of neighborhood di_erence. *International Journal of Automation and Computing*. vol. 14, no. 1, pp. 5767, 2017.
[32] V. Levorato. Core decomposition in directed networks kernelization and strong connectivity. *Complex Networks*. vol. 549, pp. 129-140, 2014.
[33] J. Abello, M. Resende, S. Sudarsky. Massive quasi-clique detection. *Theoretical Informatics*. pp. 598612, 2002.
[34] R.J. Mokken. Cliques, clubs and clans. *Qual.Quant*. vol. 13, no. 2, pp. 161173, 1979.
[35] J. Cohen. Graph twiddling in a MapReduce world. *Comput. Sci. Eng.*. vol. 11, no. 4, pp. 2941.
[36] J.A. Sudarsky. Massive Quasi-Clique Detection. *Latin American Symposium on Theoretical Informatics, Theoretical Informatics*. pp. 598-612, 2002.
[37] N. Tatti, A. Gionis. Density-friendly Graph Decomposition. *in Proceedings of the 24th International Conference on World Wide Web - WWW '15*. pp. 1089-1099, 2015.
[38] D. H. Santo Fortunatoab. Community detection in networks: A user guide. *Physics Reports*. vol. 659, pp. 1-44, 2016.
[39] S. Vergara. Complete graph immersions in dense graphs. *Discrete Mathematics*. vol. 340, pp. 1019-1027, 2017.
[40] K. Richard, R. Darst David, Z.N. Reichman Peter Ronhovde. An edge density de_nition of overlapping and weighted graph communities. *Physics and Society (physics.soc-ph); Social and Information Networks (cs.SI)*. 2013.
[41] R. SunilKumar, K. Balakrishnan. Betweenness centrality in cartesian product of graphs. *Electronic notes in discrete mathematics*. vol. 63, pp. 287-294, 2017.
[42] F. Linton. A set of measures of centrality based upon betweenness. *Sociometry*. vol.40, pp. 3541, 1997.
[43] G. Sabidussi. The centrality index of a graph. *Psychometrika*, vol. 31, pp. 581603, 1966.
[44] T.H. Cormen, C.E. Leiserson, R. Rivest, C. Stein. Introduction to Algorithms (1st ed.). MIT Press and McGraw Hill, The FloydWarshall algorithm, pp. 558565, 1990.

[45] W. Ellen, K. Calvert, M. Je_ Donahoo. A Quantitative Comparison of Graph-based Models for Internet Topology. *IEEE/ACM Transactions on Networking, December*. 1997.
[46] Q. Chen, C. Jin, S. Jamin. Inet: Internet Topology Generator, 2000.