Saturated Betweenness Centrality Sets of Vertices in Graphs

Cristina Maier and Dan Simovici
University of Massachusetts Boston, Boston, USA
Email: {cmaier, dsim}@cs.umb.edu

***Abstract*—Betweenness centrality of a set of vertices is a measure of centrality in a graph based on the notion of geodesics between vertices. Starting from betweenness centrality of vertices a similar measure can be defined on sets of vertices which allows us to determine the control of the flow of information that these sets have within a network. We introduce the notion of saturated betweenness centrality set, as a set with a positive betweenness centrality, such that its betweenness centrality is greater than the betweenness centrality of any proper subset and is smaller or equal than the betweenness centrality of any possible superset. We examine various properties of saturated betweenness centrality sets. These findings allow us to introduce an algorithm aiming to detect sets of vertices of minimal size that have a high level of control of flow of information. We provide some experimental results run on artificial and on several real-world networks.**

***Index Terms*—saturated betweenness centrality set, social networks, biological networks, network control algorithm**

I. Introduction

Vertex centralities are measures that quantify the importance of a vertex within a network. There are different types of vertex centrality: degree centrality, eigenvector centrality, closeness centrality, and betweenness centrality to name a few. The *betweenness centrality* of a vertex was first introduced in [1], and formalized in [2]. It counts the number of shortest paths between any other two vertices that pass through that vertex.

A fast algorithm for computing the betweenness centrality is introduced in [3] enabling the usage of this centrality method on a larger scale. To speed up the betweenness centrality computations in large networks, algorithms that approximate the betweenness centrality have been introduced. A distributed-memory algorithm, named *Min Rounds BC* (MRBC) to efficiently compute the betweenness centrality is introduced in [4]. There are algorithms, such as [5], that approximate the shortest paths when calculating the betweenness centralities. Other algorithms approximate the betweenness centrality by calculating only a subset of the shortest paths. One such algorithm is presented in [6]. ABRA (Approximating Betweenness with Rademacher Averages) [7], a family of algorithms based on progressive sampling, is another example. A *Graph Neural Network* (GNN) approach to approximate the betweenness centrality is proposed in [8]. GNN was also used to identify vertices with high betweenness centrality [9]. In [10], [11] some other techniques to approximate the betweenness centrality are proposed. Several algorithms that approximate the betweenness centrality are compared in [12], [13]. Recomputing the betweenness centrality after removal of nodes in large graphs can be computationally expensive. An algorithm that approximates the interactive betweenness centrality was introduced in [14].

The betweenness centrality for the cartesian products of graphs is investigated in [15]. The authors discuss several types graphs, such as *grid*, *hypercube*, *Hamming* graphs, which can all be written as cartesian product of graphs. In [16], the betweenness centrality in different classes of graphs is examined.

Changes in the graph topology, such as updating, inserting or deleting vertices or edges, could cause changes in the betweenness centrality. Recomputing the betweenness centrality for each vertex can be computationally expensive. Several dynamic algorithms for calculating the betweenness centrality in evolving graphs were introduced [17]-[22]. A temporal graph is a graph that has a fixed set of vertices, and its edges vary over a discrete set of time steps. Different variants of betweenness centrality for temporal graphs are investigated in [23].

The betweenness centrality has applications in several domains such as networking, medicine [24] and social networks. In [25], four centrality measures (i.e. degree, eigenvector, closeness and betweenness), have been interpreted in the context of diffusion of information, concluding that the betweenness and closeness measures do not work well for quantifying a vertex diffusion's capabilities. Instead, betweenness centrality is seen as measuring the extent to which a vertex facilitates the flow between other vertices that diffuse information. Thus, the betweenness centrality can be seen as a measurement of the control a vertex has on the flow of information within that graph or network [26]. In a social network or a computer network, information between two persons or two nodes (represented as two vertices in the graph) can travel across any existing paths between those persons or nodes.

The betweenness centrality assumes the information travels via the shortest paths. Thus, in a social network, the

Manuscript received December 12, 2020; revised July 21, 2021.

doi: 10.12720/jait.12.4.287-295

higher the betweenness centrality of a person, the more flow of information that person controls. Three *flow-based centrality measures* have been introduced in [27]. These differ from the betweenness centrality in two ways: the edges in the graph can be weighted, and instead of using the shortest path, these measurements can use all independent paths within the graph. In [28] the author defines the *random-walk betweenness centrality*, which is a modified version of the betweenness centrality, where instead of assuming that the messages travel via the shortest paths, the assumption is that the messages would randomly take any path, by performing a random walk. In a real world scenario, the messages routing would probably fall between taking the shortest path and performing a random walk. Some experiments in [28] show that the two approaches give similar results. In [29], the authors propose two betweenness centralities methods based on *Randomized Shortest Paths* (RSP) [30], [31] framework, which compromises between the shortest and random paths in an optimal way.

A topology of network flow based on two dimensions is introduced in [32] as an aid in determining which centrality measures work better for different network flow use cases. The two dimensions are: node-to-node transmission, and trajectory. The node-to-node transmission has to do with whether the message diffusion occurs via replication (i.e. copy) or transfer (i.e. move), and whether this transmission occurs serially or in parallel. Therefore, the three possible values for this dimension are: serial duplication, parallel duplication, and transfer. The second dimension, the trajectory, takes into consideration whether the traffic flows non-deterministically, having four possible values: geodesics, paths, trails, and walks. In [32], the relationship between this topology and some centrality measures is examined. This can be helpful in determining which centrality approach would work best on different scenarios. For example, while betweenness centrality might be a better fit for a package delivery process network, the degree centrality might work better for the diffusion of a gossip using a topology that involved parallel duplication and walk trajectory. In [33], the degree, the betweenness and the eigenvector centralities are used to examine whether the time-related performance of a distribution network (i.e. supply network) is influenced by the network topology. While the betweenness centrality might indicate nodes that control important material flow, the degree centrality might indicate nodes that have a large operational load, and the eigenvector centrality might indicate nodes that are crucial due to their connections to other nodes. Based on the experiments, the authors conclude that the degree centrality seems to be associated with time-related performance issues. The betweenness and eigenvector centralities do not seem to significantly be associated with time-related performance issues.

Centrality measures can also be applied to sets of vertices. In [34] the authors talk about degree, closeness, betweenness, and flow betweenness centralities applied to groups of individuals. In [35], the authors introduce the Routing Based Centrality for single vertices, ordered sequences of vertices and sets of vertices. Given a loop-free routing scheme, Routing Based Centrality measures the degree to which vertices are exposed to network traffic.

Flow of information within a network is very important. In [36], the authors show that brain networks evolve by maximizing their information flow capacity. Flow Authority Model [37] uses degree discount and maximum degree heuristics to maximize information spread within a network.

We examine the notion of betweenness centrality of a set of vertices, which measures the amount of control of flow of information those vertices exercise as a group. Adding or removing a vertex to/from an existing set of vertices can increase, decrease or leave unchanged the betweenness centrality of that set. We are interested in finding associations of vertices which exhibit a high betweenness centrality. We introduce the notion of *saturated betweenness centrality se*t as a set whose association exhibit a maximal control of flow of information. Such a set represents an association of vertices in which all its members positively contribute to the control of information, and which can not be further increased by adding additional vertices. Verifying all possible combinations of vertices is prohibitively expensive. Therefore, optimization techniques must be used. Some properties of these sets allowed us to introduce an optimized algorithm that significantly reduces the search space to find saturated betweenness centrality sets as well as the largest betweenness centrality association of sets up to a given size.

Grouping or clustering vertices also has applicability in other areas besides controlling the flow of information in a graph. Detecting communities within a social network is an example where such a grouping could be used.

In social networks, a *clique* [26], [38]-[40] is a maximal complete subgraph in a graph $G = (V,E)$. As the definition of a clique is too strict for most cases, the notion of *k-clique* [39] was introduced. In a graph $G(V,E)$ with $|V| = n$ vertices, a k-clique is defined as a set of vertices $M \subset V$, with $|M| \geq 3$, such that between any two vertices in M there is at least one path having length at most k. Note that a proper subset of an k-clique is not a k-clique. However, multiple k-cliques can overlap. We can observe that the considered paths between two vertices of a k-clique M can contain vertices in V - M. If a graph G contains a k-clique M this has no impact on the connectedness and the diameter of the subgraph induced by M [38]. This suggests the notions of *k-clan* (also called *sociometric k-clique*) [38], [41], and *k-club* [41]. These are just a few examples on how sets of vertices could be characterized or grouped according to the structure of the subgraph they form. For a graph $G = (V,E)$, the approach proposed in this paper is not mainly focused on the connection properties between vertices from a subset $S \subset V$. Instead, it is focused on the role vertices from S play in the connections between vertices in $V - S$.

In Section II we define the betweenness centrality of a set of vertices. Next, we introduce the notion of *Saturated Betweenness Centrality (SBC) set.* Section IV presents an algorithm for detecting SBC sets, as well as sets with highest control of flow of information for sets of up to a given size. Next, in Section V, we provide some

experimental results from running the algorithm in some real-world networks. We conclude with Section VI which contains final remarks and discusses possible future work.

The current paper is an extended version of a paper we presented at the ICISDM 2020 conference [42].

II. Betweenness Centrality of Sets of Vertices

Definition 2.1: Let $G = (V, E)$ be an undirected graph without multiple edges. A *geodesic* between two vertices $v_1, v_2 \in V$ is a shortest path between those vertices.

The number of distinct geodesics between vertices s and t is denoted by σ_{st}. The number of distinct geodesics between vertices s and t that pass through at least a vertex in set of vertices *S*, $S \subseteq V$ is denoted by $\sigma_{st}(S)$. Note that if no path exists between two vertices *s* and *t* (seen in the case when the graph has multiple connected components), then σ_{st} is not defined.

Next we define the betweenness centrality of sets of vertices. Note that this definition reduces to the definition of Group Betweenness Centrality introduced in [34].

Definition 2.2. Let $G = (V, E)$ be an undirected graph with $|V| \geq 3$. The *betweenness centrality of a set of vertices S*, where *S* is a strict subset of *V* and $1 \leq |S| \leq |V| - 2$, is given by

$$C_B(S) = \sum_{s,t \in V-S, s<t} \frac{\sigma_{st}(S)}{\sigma_{st}}$$

When $S=\{v\}$, the betweenness centrality of the singleton $\{v\}$ reduces to the sum of ratios of the number of geodesics passing through *v* and the total number of geodesics linking two distinct nodes in $V-\{v\}$, which is the classical definition of betweenness centrality of a vertex [2].

Note that we do not consider shortest paths that start or end with a vertex in *S*. Also, if one or more vertices from *S* are present on the same geodesic path between two other vertices, we only account for their presence once. Therefore, it does not make a difference if on a geodesic path we have a single vertex from S or multiple vertices from *S*.

Given a graph $G = (V, E)$, with n vertices, the upper bound for betweenness centrality of a vertex equals

$$\binom{n-1}{2} = \frac{(n-1)\cdot(n-2)}{2}$$

Because in a graph with *n* vertices there exist a total of $\frac{n\cdot(n-1)}{2}$ shortest distances, and we are left with $\binom{n-1}{2}$ possible shortest distances that can pass through one vertex.

Definition 2.3. A *star graph* is a graph of n vertices, $|n| \geq 3$, where one vertex has degree *n*-1, and the rest of the vertices have degree 1.

Only star graphs have vertices with betweenness centrality equaling the upper bound defined above. In such a graph, $G = (V, E)$, the vertex with maximum betweenness centrality is unique, and it is the vertex with degree $|V| - 1$.

Note that if S is a subset of the set of vertices of a graph $G = (V,E)$ with n vertices and $|S| = k$, we have $C_B(S) \leq \binom{n-k}{2}$ because the number of geodesics between points outside S is $\binom{n-k}{2}$ and this is the maximum number of geodesics that can pass through S. Since $C_B(S) = \sum_{s,t \in V-S, s<t} \frac{\sigma_{st}(S)}{\sigma_{st}}$, it follows that $C_B(S) \leq \binom{n-k}{2}$.

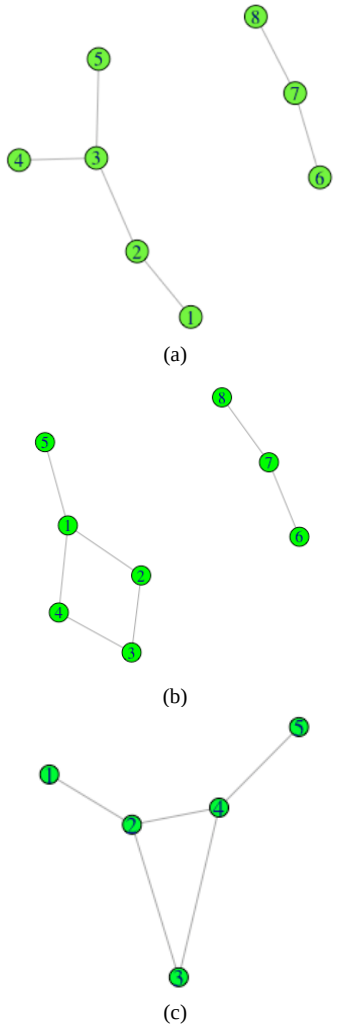

Figure 1. Examples of graphs.

Below we give some examples of computations of betweenness centrality of sets of vertices. Given a graph $G = (V, E)$, when we add a new vertex $x \in V$ to a set of vertices $S \subset V - \{x\}$, we can increase, reduce, or leave unchanged the betweenness centrality of set *S*.

Example 2.4. Given the graph from Fig. 1(a) we have the following betweenness centralities of single vertices:

$$C_B(\{1\}) = 0, C_B(\{2\}) = 3, C_B(\{3\}) = 5, C_B(\{4\}) = 0$$

$$C_B(\{5\}) = 0, C_B(\{6\}) = 0, C_B(\{7\}) = 1, C_B(\{8\}) = 0$$

Table I shows cases when the betweenness centrality of a set increases, decreases or stays the same when a new vertex is added to that set:

TABLE I. Example 2.4

Set S	{x}	$C_B(S)$	$C_B(\{x\})$	$C_B(S \cup \{x\})$
{2}	{7}	3	1	4
{2}	{3}	3	5	3
{2}	{1}	3	0	0
{2,3}	{8}	3	0	3

Example 2.5. In this example we use the graph from Fig. 1(b). Below we have the betweenness centralities of vertices:

$$C_B(\{1\}) = 3.5, C_B(\{2\}) = 1, C_B(\{3\}) = 0.5, C_B(\{4\}) = 1$$

$$C_B(\{5\}) = 0, C_B(\{6\}) = 0, C_B(\{7\}) = 1,\ C_B(\{8\}) = 0$$

The betweenness centralities when adding a new vertex to existing sets, are presented in Table II:

TABLE II. EXAMPLE 2.5

Set S	{x}	$C_B(S)$	$C_B(\{x\})$	$C_B(S \cup \{x\})$
{3}	{2}	0.5	1	0
{2,4}	{5}	2	0	1
{2,4}	{6}	2	0	2

Example 2.6. For the graph shown in Fig. 1(c), the single betweenness centralities are:

$$C_B(\{1\}) = 0, C_B(\{2\}) = 3, C_B(\{3\}) = 0$$

$$C_B(\{4\}) = 3, C_B(\{5\}) = 0$$

Table III contains some examples of what can happen when two vertices are associated:

TABLE III. EXAMPLE 2.6

Set S	{x}	$C_B(S)$	$C_B(\{x\})$	$C_B(S \cup \{x\})$
{1}	{2}	0	3	0
{1}	{4}	0	3	2
{2}	{4}	3	3	3

III. SATURATED BETWEENNESS CENTRALITY SETS

Next, we introduced the notion of *saturated betweenness centrality set.*

Definition 3.1. A *saturated betweenness centrality set* (shortened as *SBC* set) in a graph $G = (V, E)$ is a subset S of V, $|S| \leq |V| - 2$, having $C_B(S) > 0$, such that:

1) $C_B(S) > C_B(T)$ for every proper subset T of S, and
2) $C_B(S) \geq C_B(U)$ for every superset U of S.

Note that multiple SBC sets can overlap.

We observe that for any graph $G = (V, E)$ and a saturated betweenness centrality set S in G, each vertex from S contributes with a positive value to the betweenness centrality of S.

Theorem 3.2. Given a graph $G = (V, E)$ and a saturated betweenness centrality set $S = \{v_1, \dots, v_k\}$ we have:

$$\mathrm{C_B}(\{\mathrm{v_1}\}) < \mathrm{C_B}(\{\mathrm{v_1}, \mathrm{v_2}\}) < \cdots < \mathrm{C_B}(\{\mathrm{v_1}, \dots, \mathrm{v_{k-1}}\}) < \mathrm{C_B}(\mathrm{S})$$

regardless of the order of the indices of vertices.

Thus, for $S'' \subset S' \subset S$ we have $C_B(S'') < C_B(S')$. Furthermore, if $Z \subseteq V$ has a subset U with $\mathrm{C_B(U)} \geq \mathrm{C_B(Z)}$, then Z is not an SBC set or a subset of an SBC set.

Proof: Since each vertex from S contributes with a positive value to the centrality of S, each time we add an element, the betweenness centrality of the newly formed set increases.

Note that the above observations exhibit a recursive property. Any set S is an SBC set if all subsets of size $|S|$ - 1 have a betweenness centrality greater than any of their subsets, and then all their subsets of size $|S|$ - 2 satisfy the same property, and so on, until we get to sets of size one.

Theorem 3.3. For a graph $G = (V, E)$ and $k \in N$, where $k \leq |V| - 3$, the largest betweenness centrality set among sets of size up to k is either an SBC set or a subset of an SBC set of size greater than k. When $k = |V| - 2$, the largest betweenness centrality set is an SBC set.

Thus, the largest betweenness centrality set is an SBC set. A complete graph is a graph in which there is an edge between any two vertices belonging to that graph.

Theorem 3.4. For an unweighted complete graph $G = (V, E)$ we have $C_B(v) = 0$ for any vertex $v \in V$. We also have $C_B(S) = 0$ for any set of vertices $S \subset V$ where $|S| \leq |V| - 2$. Furthermore, this means we have no SBC sets.

Proof: Since any vertex is directly connected to any other vertex from the graph, and the edges are unweighted, it means that the shortest path between any two vertices is via the edge that connects them. Therefore, the shortest paths do not contain any intermediate vertices.

IV. DETECTING SATURATED BETWEENNESS CENTRALITY SETS

For a given undirected graph $G = (V, E)$, we would like to find all saturated betweenness centrality sets.

Considering all possible combinations and verifying for each of them the properties of the saturated betweenness centrality set, is prohibitively expensive. In this section we provide an optimized algorithm which reduces the search space, making it scalable for larger graphs.

Using the findings from Section III, when searching for SBC sets, we start from sets of size one and build up towards larger SBC sets. Using Theorem 3.2 it follows that if the betweenness centrality of a set S is not greater than the betweenness centrality of every subset of size $|S|$ - 1, then S and all its supersets cannot be SBC sets. Therefore, no superset of S will be considered when building saturated sets of sizes greater than $|S|$. Using this property, we significantly reduce the search space.

The pseudocode is given in Algorithm 4.1.

The function *searchSatBCSets*, shown in Algorithm 4.2. searches for SBC sets, getBC returns the BC of a set, while *extractAllSubsetsOfSizeM1*(S) returns a list of all subsets of S of size $|S|$-1. Function *generateCandidates* generates candidate sets of size k. At each step k, we generate candidate sets of size k by only expanding candidate sets of size k - 1, which are the sets present in *prev* (i.e. sets of size k - 1 that satisfy all checks, and therefore they represent either SBC sets of size k - 1, or subsets of SBC sets). We keep these sets of vertices ordered. We expand an existing set, by adding another vertex that has a larger value than all the elements from the set it gets added to. Note that during the generation, some techniques such as verifying that a couple of subsets of newly generated sets are present in subsets *prev*, could be used to reduce the number of candidates.

Algorithm 4.1: Main - FindSBCSets(A(G), maxSize, includeNextHighest)

```
input  : graph's adjacency matrix A(G) ∈ ℝ^{n×n},
         maxSize, includeNextHighest;
output : A map ⟨SBCSet,betweennessValue⟩ containing
         as keys all SBC sets of size up to maxSize, and as
         values, their corresponding betweenness
         centrality. If includeNextHighest is True, output
         includes sets of size maxSize + 1 (if any exists)
         that are SBC sets or SBC subsets of greater size.
1  begin
2      n = number of rows in A(G);
3      sp = getShortestPaths (A(G));
4      W = emptySet(); sat = emptyMap();
       // single vertices
5      for v = 1 : n do
6          bc = getBC(sp,v);
7          if bc > 0 then
8              W.append(v); sat.put(set(v), bc);
9          end
10     end
11     prev = sat;
12     resultSat = searchSatBCSets(sp,W,
13     prev,sat,maxSize,includeNextHighest,n);
14     return resultSat;
15 end
```

Algorithm 4.2: searchSatBCSets

```
1  Function searchSatBCSets(sp,W,
2  prev,sat,maxSize,includeNextHighest,n)
3  begin
4      num = size(prev.keys); k = 1; maxSize =
         min(maxSize,n-3);
5      while num > 0 AND k ⩽ maxSize do
6          num = 0; k = k + 1;
7          newW = emptySet(); newPrev = emptyMap();
8          newListSets = generateCandidates(prev,W,k);
9          if size(newListSets) == 0 then
10             return sat;
11         end
12         for i = 1 : size(newListSets) do
13             S = newListSets[i]; bc = getBC(sp,S);
               // all subsets of S of size k - 1
14             listSubsetsSizeM1 = extractAllSubsetsM1(S);
               // verify SBC subsets property of S
15             ok = True; z = 0;
16             while ok AND z < size(listSubsetsSizeM1) do
17                 z = z + 1; S' = listSubsetsSizeM1[z];
18                 if Not (prev.containsKey(S') AND
                     prev.get(S') < bc) then
19                     ok = False;
20                 end
21             end
22             if ok then
23                 sat.removeAll(listSubsetsSizeM1);
24                 if ((k ⩽ maxSize ) OR (k == maxSize+1 AND
                     includeNextHighest)) then
25                     num = num + 1; sat.put(S, bc);
                       newPrev.put(S, bc);
26                     newW = concatSets(newW, S);
27                 end
28             end
29         end
30         prev = newPrev; W = newW;
31     end
32     return sat;
33 end
```

The algorithm can be used to generate SBC sets up to a given size, *maxSize*, as well as sets of size *maxSize* +1 with high BC (these will be either SBC sets or subsets of SBC sets of higher size). Sometimes we might not be interested in finding the SBC sets, and instead we might be interested in detecting sets up to a given size that exhibit the highest control of information. For example, if we are interested in finding vertices that exhibit the highest control of information in sets of four vertices, we could run *FindSBCSets*(A(G), 3, True). This will return all SBC sets of size up to three, and all candidate sets of size four that satisfy the properties (i.e. they are either SBC sets or subsets of SBC sets of size higher than four). The returned sets can be sorted based on their betweenness centrality value. Note that SBC sets do not benefit from further associations with other vertices (when more vertices are added to such sets, their BC value will only decrease or stay unchanged), but the subsets of higher size SBC sets, when associated with other vertices in larger groups might exhibit an increase in the BC value. If instead of this, we run *FindSBCSets*(A(G),3, False), the algorithm returns all SBC sets of up to size three. In the algorithm described below, sat represents a list containing all candidate SBC sets found so far; when the process finishes, this variable will contain the output. Variable *prev* is a map of format map <*set*,$C_B(set)$> containing all candidate sets of size k-1 that passed the checks (i.e. they will be either SBC sets or subsets of SBC sets).

We present below two examples, where we show basic checks that are performed at several steps throughout the algorithm. Note that not all checks and actions are mentioned.

Example 4.1. Given the graph from Fig. 1(b) we want to find the SBC sets. Below we describe the steps of the algorithm starting from sets of size 1. Note that at the beginning of each step *prev* variable is instantiated with an empty map.

Step k = 1: sets of size 1. Table IV describes checks performed at this step.

TABLE IV. SOME CHECKS AT K = 1 (EXAMPLE 4.1)

Set S	$C_B(S)$	Checks	OK?	Actions
{1}	3.5	$C_B(S) > 0$	yes	add S to sat
{2}	1	$C_B(S) > 0$	yes	add S to sat
{3}	0.5	$C_B(S) > 0$	yes	add S to sat
{4}	1	$C_B(S) > 0$	yes	add S to sat
{5}	0	$C_B(S) > 0$	no	n/a
{6}	0	$C_B(S) > 0$	no	n/a
{7}	1	$C_B(S) > 0$	yes	add S to sat
{8}	0	$C_B(S) > 0$	no	n/a

After these checks, we have:

$sat = \{(\{1\},3.5), (\{2\},1), (\{3\},0.5), (\{4\},1), (\{7\},1),$ and $prev = sat$.

These are the only subsets of size one that will be combined to form sets of size two.

Because *size(prev.keys)* > 0 we proceed to step k = 2.

Step k = 2: sets of size 2.

Table V describes some checks performed at this step.

TABLE V. SOME CHECKS AT K = 2 (EXAMPLE 4.1)

Set S	$C_B(S)$	Checks	OK?	Actions
{1,2}	2	$C_B(S) > C_B(\{1\})$ $C_B(S) > C_B(\{2\})$	no	n/a
{1,3}	3	$C_B(S) > C_B(\{1\})$ $C_B(S) > C_B(\{3\})$	no	n/a
{1,4}	2	$C_B(S) > C_B(\{1\})$ $C_B(S) > C_B(\{4\})$	no	n/a
{1,7}	4.5	$C_B(S) > C_B(\{1\})$ $C_B(S) > C_B(\{7\})$	yes	remove {1} & {7} from sat; add S to sat; add S to prev
{2,3}	0	$C_B(S) > C_B(\{2\})$ $C_B(S) > C_B(\{3\})$	no	n/a
{2,4}	2	$C_B(S) > C_B(\{2\})$ $C_B(S) > C_B(\{4\})$	yes	remove {2} & {4} from sat; add S to sat; add S to prev
{2,7}	2	$C_B(S) > C_B(\{2\})$ $C_B(S) > C_B(\{7\})$	yes	remove {2} & {7} from sat; add S to sat; add S to prev
{3,4}	0	$C_B(S) > C_B(\{3\})$ $C_B(S) > C_B(\{4\})$	no	n/a
{3,7}	1.5	$C_B(S) > C_B(\{3\})$ $C_B(S) > C_B(\{7\})$	yes	remove {3} & {7} from sat; add S to sat; add S to prev
{4,7}	2	$C_B(S) > C_B(\{4\})$ $C_B(S) > C_B(\{7\})$	yes	remove {4} & {7} from sat; add S to sat; add S to prev

After these checks, we have:
$sat = \{(\{1,7\},4.5), (\{2,4\},2), (\{2,7\},2), (\{3,7\},1.5), (\{4,7\},2)\}$
and
$prev = \{(\{2,4\},2), (\{1,7\},4.5), (\{2,7\},2), (\{3,7\},1.5), (\{4,7\},2)\}$

Because *size(prev.keys)* > 0 we proceed to step k = 3.

Step k = 3: sets of size 3. At this step we generate candidate sets of size three by increasing the size of sets in *prev*. Table VI describes some checks performed at this step.

TABLE VI. SOME CHECKS AT K = 3 (EXAMPLE 4.1)

Set S	$C_B(S)$	Checks	OK?	Actions
{2,4,7}	3	$C_B(S) > C_B(\{2,4\})$ $C_B(S) > C_B(\{2,7\})$ $C_B(S) > C_B(\{4,7\})$	yes	remove {2,4}, {2,7} & {4,7} from sat; add S to sat; add S to prev

After these checks, we have:
$sat = \{(\{1,7\},4.5), (\{3,7\},1.5), (\{2,4,7\},3)\}$, and $prev = \{(\{2,4,7\},3)\}$.

Because *size(prev.keys)* > 0 we proceed to step k = 4.

Step k = 4: sets of size 4.

No more candidate sets of size 4 can be generated. The algorithm stops.

The saturates betweenness centrality sets are: {1, 7}, {3,7}, {2, 4, 7}.

Example 4.2. Given the graph from Fig. 1(a), we want to find all SBC sets.

Below we describe the steps of the algorithm starting from sets of size 1, and then increasing the sets size step by step, until no further candidate sets could be formed.

Step k = 1: Here we verify candidate sets of size 1. Table VII describes all these checks.

TABLE VII. SOME CHECKS AT K = 1 (EXAMPLE 4.2)

Set S	$C_B(S)$	Checks	OK?	Actions
{1}	0	$C_B(S) > 0$	no	n/a
{2}	3	$C_B(S) > 0$	yes	add S to sat
{3}	5	$C_B(S) > 0$	yes	add S to sat
{4}	0	$C_B(S) > 0$	no	n/a
{5}	0	$C_B(S) > 0$	no	n/a
{6}	0	$C_B(S) > 0$	no	n/a
{7}	1	$C_B(S) > 0$	yes	add S to sat
{8}	0	$C_B(S) > 0$	no	n/a

After these computations for sets of size 1 we have:
$sat = \{(\{2\},3), (\{3\},5), (\{7\},1)$, and $prev = sat$.

Because *size(prev.keys)* > 0 we proceed to step k =2.

Step k = 2: We form sets of size 2 using elements from prev. Table VIII describes some checks performed.

TABLE VIII. SOME CHECKS AT K = 2 (EXAMPLE 4.2)

Set S	$C_B(S)$	Checks	OK?	Actions
{2,3}	3	$C_B(S) > C_B(\{2\})$ $C_B(S) > C_B(\{3\})$	no	n/a
{2,7}	4	$C_B(S) > C_B(\{2\})$ $C_B(S) > C_B(\{7\})$	yes	remove {2} & {7} from sat; add S to sat; add S to prev
{3,7}	6	$C_B(S) > C_B(\{3\})$ $C_B(S) > C_B(\{7\})$	yes	remove {3} & {7} from sat; add S to sat; add S to prev

After these checks we have:
$sat = \{(\{2,7\},4), (\{3,7\},6)\}$, and
$prev = \{(\{2,7\},4), (\{3,7\},6)\}$.

Because *size(prev.keys)* > 0 we proceed to step k = 3.

Step k = 3: When trying to form sets of size 3, set {2, 3, 7} does not satisfy the checks.

The algorithm finds the SBC sets: {2, 7}, {3, 7}.

V. IMPLEMENTATION AND EXPERIMENTAL RESULTS

In this section we present experiments ran on three well-known real-world networks. The experimental results show that the algorithm significantly reduces the search space. Note that in addition to limiting the search space, the algorithm also speeds up the verification of the SBC properties. We implemented the algorithm in R. The algorithm was parallelized using *doParallel* and *foreach* libraries. We parallelized the verification of whether a set of size k is either an SBC set or subset of an SBC set, and the generation of candidate sets of a given size k that will be sent for verification. We sorted the returned sets based on their BC values. We used a Mac OS computer with a 3.5 GHz Intel Core i7 processor, and 16 GB of RAM.

A. Zachary's Karate Network

First, we present experimental results ran on the well-studied Zachary's Karate Network [43]. This is a network of 34 vertices. Each vertex represents a member of the karate club. We have an edge between two members that interact outside of the karate club.

We are interested in finding all SBC sets of this network. Note that the collection of possible candidate sets contains 2^{34}-32=17,179,869,152 sets, which is obviously a

prohibitive large number. Our implemented optimized algorithm only checks 3,845 sets of vertices, running in about 11.58 seconds.

Finding the SBC sets provides us with possible sets of members of karate club which reach a maximal control of the flow of information (i.e. all vertices have a contribution to this control, and adding any other member to such a set, it would only decrease or leave unchanged the control). The algorithm finds that we have 194 saturated sets, and it sorts them according to the betweenness centrality of each sets. In Table IX given below, we show the top five SBC sets with the highest betweenness centrality value. In Table X we provide the SBC sets with the lowest BC values.

TABLE IX. TOP FIVE SBC SETS WITH HIGHEST BC VALUE

SBC Set S	$C_B(S)$	Comments
{1, 3, 33, 34}	384.6	highest BC
{1, 2, 34}	344.2238	
{2, 3, 32, 33, 34}	294.2667	
{2, 3, 9, 32, 34}	286.6000	
{1, 2, 28, 33}	284.8381	

The SBC set {1, 3, 33, 34} exhibits the highest control of flow of information. This is expected because these vertices represent the instructor (1), the administrator of the club (34) and two players (3 and 33). No superset of this set can exercise a higher level of

communication control.

Table X shows the SBC sets with lowest BC values.

TABLE X. SBC SETS WITH LOWEST BC VALUES

SBC Set S	$C_B(S)$
{4, 5, 26, 29, 30, 31}	16.31587
{5, 10, 26, 29, 30, 3}	11.36429
{10, 11, 26, 29, 30, 31}	11.36429
{10, 11, 25, 29, 30, 31}	11.36429
{5, 10, 25, 29, 30, 31}	11.36429

B. Dolphins Social Network

Dolphins Network [44] is an undirected network, showing the frequent interactions between 62 bottlenose dolphins living in Doubtful Sound, New Zealand. An edge exists between any two dolphins that interact frequently. This is a topic relevant in marine biology research, for finding sets up to a given size of dolphins that exhibit the highest control of information.

To find sets of up to four dolphins which exhibit high control of information we ran *FindSBCSets*(A(dolphins), 3, True). The top five associations are given in Table XI.

TABLE XI. TOP FIVE ASSOCIATIONS

Dolphins Associations S	$C_B(S)$
{8, 29, 37, 52}	868.8041
{8, 29, 30, 37}	863.5694
{2, 8, 37, 52}	862.6365
{29, 37, 41, 52}	861.2382
{29, 30, 37, 41}	856.2728

The algorithm ran in 9.5 minutes and verified 155,939 sets, compared to 597,618 possible combinations. Note that none of these dolphins' associations are SBC sets, because no set is returned when running *FindSBCSets*(A(dolphins), 4, False). Thus, these sets are subsets of SBC sets of higher size, which means they can further increase the control of information by adding additional vertices (i.e. adding other dolphin(s) to these associations). For example, one of smallest size SBC sets we detected contains 5 dolphins: {3, 7, 30, 37, 48} and has a BC value of 608.344.

C. American Football Network

The American Football Network [45] is a graph representation of games played in Division I in year 2000. Teams are represented by vertices and payed games are represented by edges. The network incorporates 115 vertices and 613 edges. The network is divided into twelve conferences/communities. We want to detect SBC sets within each community. We ran our algorithm on the subgraphs generated by each community. The results are presented in Table XII.

We expected communities '(1) Big East' and '(7) Mountain West' to have no SBC sets, as they are complete graphs. Within community '(2) Big Ten' there are eleven SBC sets which have the highest BC value 3.142857. One of such SBC sets is {7, 14, 16, 48}. In community '(6) Mid-American', the SBC set {15, 32, 35, 39, 62} has the largest BC value 11. All SBC sets from community '(8) Pacific Ten' have same BC value, 1.5. One of such SBC set is {22, 23, 79, 112}. Each of the two SBC sets within community '(10) Sun Belt' contains only one vertex. We have SBC set {1} with BC value 9, and SBC set {7} with BC value 8. The only SBC set in community '(11) Western Athletic' is {89, 115} with BC value 6.

TABLE XII. SBC SETS WITHIN THE TWELVE COMMUNITIES

Community/ Conference	Num. of Teams	Num. of SBC Sets	Running Time
(0) Atlantic Coast	9	0	0.2764931 s
(1) Big East	8	0	0.03440094 s
(2) Big Ten	11	46	0.6319609 s
(3) Big Twelve	12	52	0.8352158 s
(4) Conference USA	10	24	0.411195 s
(5) Independents	5	0	0.009389877 s
(6) Mid-American	13	42	1.013496 s
(7) Mountain West	8	0	0.039608 s
8) Pacific Ten	10	10	0.456816 s
(9) Southeastern	12	77	0.7859218 s
10) Sun Belt	7	2	0.09870005 s
(11) Western Athletic	10	1	0.1379929 s

VI. CONCLUSION AND FUTURE WORK

We examined the notion of betweenness centrality of a set of vertices in a graph and introduced the notion of Saturated Betweenness Centrality (SBC) set. For such a set its betweenness centrality decreases when any of its members are removed and, if any new vertex is added to this set, its betweenness centrality either decreases or stays the same. This means that these sets represent associations of vertices in which each member positively contributes to the betweenness centrality of the set. It also means that the betweenness centrality of such an association can not be further increased by adding additional vertices. This notion of SBC sets has applicability in areas related to controlling the flow of information in social or computer networks.

We introduced an optimized algorithm that finds the SBC sets, by reducing the search space. This algorithm can also be used to find sets (which can be SBC sets or subsets of SBC sets of higher size), which exhibit the highest control of flow of information for associations of up to a given size. We presented some experimental results ran on Zachary's Karate Club, Dolphins, and American Football Networks.

We plan to investigate how a similar approach would apply to flow-based centrality [27], and random-walk betweenness centrality [28]. We also contemplate a variation of this algorithm, that could be ran on local subgraphs, which would make it more practical for larger graphs.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Cristina Maier and Dan Simovici conducted the research. Both authors wrote and approved the final version of the manuscript.

REFERENCES

[1] A. Bavelas, "A mathematical model for group structures," *Human Organization*, vol. 7, no. 3, pp. 16-30, 1948.

[2] L. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, no. 1, pp. 35-41, 1977.

[3] U. Brandes. (2004). A Faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*. [Online]. 25. Available: https://doi.org/10.1080/0022250X.2001.9990249

[4] L. Hoang, M. Pontecorvi, R. Dathathri, G. Gill, B. You, K. Pingali, and V. Ramachandran, "A round-efficient distributed betweenness centrality algorithm," in *Proc. the 24th Symposium on Principles and Practice of Parallel Programming*, Washington, 2019, pp. 272-286.

[5] J. Pfeffer and K. M. Carley, "K-Centralities: Local approximations of global measures based on shortest paths," in *Proc. the 21st International Conference on World Wide Web*, Lyon, France, 2012, pp. 1043-1050.

[6] M. Riondato and E. M. Kornaropoulos, "Fast approximation of betweenness centrality through sampling," *Data Mining and Knowledge Discovery*, vol. 30, pp. 438-475, 2015.

[7] M. Riondato and E. Upfal, "ABRA: Approximating betweenness centrality in static and dynamic graphs with Rademacher averages," *ACM Trans. Knowl. Discov. Data*, vol. 12, no. 5, article 61, July 2018.

[8] S. K. Maurya, X. Liu, and T. Murata, "Fast approximations of betweenness centrality with graph neural networks," in *Proc. the 28th ACM International Conference on Information and Knowledge Management*, Beijing, China, 2019, pp. 2149-2152.

[9] C. Fan, L. Zeng, Y. Ding, M. Chen, Y. Sun, and Z. Liu, "Learning to identify high betweenness centrality nodes from scratch: A novel graph neural network approach," in *Proc. the 28th ACM International Conference on Information and Knowledge Management*, Beijing, China, 2019, pp. 559-568.

[10] R. Matsuo, R. Nakamura, and H. Ohsaki, "A study on sparse-modeling based approach for betweenness centrality estimation," in *Proc. COMPSAC*, 2018, pp. 973-976.

[11] A. V. D. Grinten and H. Meyerhenke, "Scaling betweenness approximation to billions of edges by MPI-based adaptive sampling," in *Proc. IPDPS*, 2020, pp. 527-535.

[12] Z. AlGhamdi, F. Jamour, S. Skiadopoulos, and P. Kalnis, "A benchmark for betweenness centrality approximation algorithms on large graphs," in *Proc. the 29th International Conference on Scientific and Statistical Database Management*, Chicago, IL, USA, 2017, article 6.

[13] J. Matta, G. Ercal, and K. Sinha, "Comparing the speed and accuracy of approaches to betweenness centrality approximation," *Computational Social Networks*, vol. 6, 2019.

[14] S. Wandelt, X. Shi, and X. Sun, "Approximation of interactive betweenness centrality in large complex networks," *Complexity*, pp. 1-16, 2020.

[15] S. Kumar and K. Balakrishnan, "Betweenness centrality in Cartesian product of graphs," AKCE *International Journal of Graphs and Combinatorics*, 2019.

[16] R. S. Kumar, K. Balakrishnan, and M. Jathavedan, "Betweenness centrality in some classes of graphs," *International Journal of Combinatorics*, pp. 1-12, 2014.

[17] E. Bergamini, H. Meyerhenke, M. Ortmann, and A. Slobbe, "Faster betweenness centrality updates in evolving networks," arXiv preprint arXiv:1704.08592, 2017.

[18] O. Green, R. McColl, and D. A. Bader, "A fast algorithm for streaming betweenness centrality," in *Proc. International Conference on Privacy, Security, Risk and Trust and International Conference on Social Computing*, 2012, pp. 11-20.

[19] T. Hayashi, T. Akiba, and Y. Yoshida, "Fully dynamic betweenness centrality maintenance on massive networks," *Proc. VLDB Endow.*, vol. 9, no. 2, pp. 48-59, Oct. 2015.

[20] M. Kas, M. Wachs, K. M. Carley, and L. R. Carley, "Incremental algorithm for updating betweenness centrality in dynamically growing networks," in *Proc. the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, Niagara, Ontario, Canada, 2013, pp. 33-40.

[21] N. Kourtellis, G. D. Francisci Morales, and F. Bonchi, "Scalable online betweenness centrality in evolving graphs," in *Proc. IEEE 32nd International Conference on Data Engineering*, 2016, pp. 1580-1581.

[22] M. Nasre, M. Pontecorvi, and V. Ramachandran, "Betweenness centrality – Incremental and faster," in *Mathematical Foundations of Computer Science*, E. Csuhaj-Varjú, M. Dietzfelbinger, and Z. Ésik, Eds., Springer Berlin Heidelberg, 2014, pp. 577-588.

[23] S. Buß, H. Molter, R. Niedermeier, and M. Rymar, "Algorithmic aspects of temporal betweenness," in *Proc. the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Virtual Event, CA, USA, 2020, pp. 2084-2092.

[24] J. Ramasamy, "A betweenness centrality guided clustering algorithm and its applications to cancer diagnosis," in *Proc. International Conference on Mining Intelligence and Knowledge Exploration*, 2017, pp. 35-42.

[25] S. P Borgatti, "Centrality and AIDS," *Connections*, vol. 18, no. 1, pp. 112-114, 1995.

[26] M. Newman, *Networks*, USA: Oxford University Press, 2018.

[27] L. C. Freeman, S. P. Borgatti, and D. R. White, "Centrality in valued graphs: A measure of betweenness based on network flow," *Social Networks*, vol. 13, pp. 141-154, 1991.

[28] M. E. J. Newman, "A measure of betweenness centrality based on random walks," *Social Networks*, vol. 27, no. 1, pp. 39-54, 2005.

[29] I. Kivimäki, B. Lebichot, J. Saramäki, and M. Saerens, "Two betweenness centrality measures based on randomized shortest paths," *Scientific Reports*, vol. 6, 2016.

[30] I. Kivimaki, M. Shimbo, and M. Saerens, "Developments in the theory of randomized shortest paths with a comparison of graph node distances," *Physica A-statistical Mechanics and Its Applications*, vol. 393, pp. 600-616, 2014.

[31] M. Saerens, Y. Achbany, F. Fouss, and L. Yen, "Randomized shortest-path problems: Two related models," *Neural Computation*, vol. 21, pp. 2363-2404, 2009.

[32] S. P. Borgatti, "Centrality and network flow," *Social Networks*, vol. 27, pp. 55-71, 2005.

[33] C. Wallmann and M. Gerschberger, "The association between network centrality measures and supply chain performance: The case of distribution networks," *Procedia Computer Science*, vol. 180, pp. 172-179, 2021.

[34] M. Everett and S. Borgatti, "The centrality of groups and classes," *Journal of Mathematical Sociology*, vol. 23, pp. 181-201, 1999.

[35] S. Dolev, Y. Elovici, and R. Puzis, "Routing betweenness centrality," *J. ACM*, vol. 57, 2010.

[36] C. G Antonopoulos, S. Srivastava, S. E. D. S. Pinto, and M. S Baptista, "Do brain networks evolve by maximizing their information flow capacity?" PLOS Computational Biology, vol. 11, no. 8, p. e1004372, 2015.

[37] M. Faisan and S. Bhavan, "Maximizing information or influence spread using flow authority model in social networks," in *Distributed Computing and Internet Technology*, Springer International Publishing, 2014, pp. 233-238.
[38] R. D. Alba, "A graph-theoretic definition of a sociometric clique*," *Journal of Mathematical Sociology*, vol. 3, pp. 113-126, 1973.
[39] R. D. Luce, "Connectivity and generalized cliques in sociometric group structure," *Psychometrika*, vol. 15, pp. 169-190, 1950.
[40] A. D. Perry and R. D. Luce, "A method of matrix analysis of group structure," *Psychometrika*, vol. 14, no. 1, pp. 95-106, March 1949.
[41] R. Mokken, "Cliques, clubs and clans," *Quality and Quantity*, vol. 13, pp. 161-173, 1979.
[42] C. Maier and D. Simovici, "Betweenness centrality of sets of vertices in graphs: Which vertices to associate," in *Proc. the 4th International Conference on Information System and Data Mining*, Hawaii, HI, USA, 2020, pp. 20-24.
[43] W. W. Zachary, "An information flow model for conflict and fission in small groups," *Journal of Anthropological Research*, vol. 33, no. 4, pp. 452-473, 1977.
[44] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology*, vol. 54, no. 4, pp. 396-405, 2003.
[45] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 12, pp. 7821-7826, 2002.

Cristina Maier is a PhD candidate in Computer Science at the University of Massachusetts Boston. Her current research interest is in the area of data mining and machine learning, where she focuses on vertex centralities in graphs, clustering, measurements in bipartite graphs, recommendation systems and optimization problems.

Dr. Dan Simovici is a professor of Computer Science at the University of Massachusetts Boston and an associate of Dana-Farber Cancer Institute in Boston. Dr. Simovici obtained his Ph.D. from the University of Bucharest, Romania. Dr. Simovici served as a visiting professor at the Tohoku University in Japan and at the University of Science and Technology of Lille, France. His current research is in the area of data mining and machine learning and served on program committees of the major data mining conferences. Dr. Simovici's work includes eighteen books and 193 research papers. His latest book, "Clustering - Theoretical and Practical Aspects" will appear in Fall 2021 at World Scientific.