Brief Announcement: Distributed k–Core Decomposition[*]

Alberto Montresor
DISI - University of Trento
via Sommarive 14
IT – 38123
Povo, Trento, Italy
alberto.montresor@unitn.it

Francesco De Pellegrini
CREATE-NET
via Alla Cascata 56/D
IT – 38123
Povo, Trento, Italy
fdepellegrini@create-net.org

Daniele Miorandi
CREATE-NET
via Alla Cascata 56/D
IT – 38123
Povo, Trento, Italy
dmiorandi@create-net.org

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Design

Introduction

In the last few years, a number of metrics and methods have been introduced for studying the relative "importance" of nodes within complex network structures [6]. Among these metrics, k-core decomposition is a well-established method for identifying particular subsets of the graph called *k-cores*, or *k-shells* [7]. Informally, a k-core is obtained by recursively removing all nodes of degree smaller than k, until the degree of all remaining vertices is larger than or equal to k. Nodes are said to have coreness k (or, equivalently, to belong to the k-shell) if they belong to the k-core but not to the $(k+1)$-core. We consider an undirected graph $G = (V, E)$ with $N = |V|$ nodes and $M = |E|$ edges. We denote by $d_G(u)$ the degree of node u within G, and by $k(u)$ its coreness index.

Definition 1. *A subgraph $G(C)$ induced by the set $C \subseteq V$ is a* k-core *if and only if $\forall u \in C : d_{G(C)}(u) \geq k$, and $G(C)$ is the maximum subgraph with this property.*

k-core decomposition has found a number of applications; for example, it has been used to characterize social networks [7], to help in the visualization of complex graphs [1], to determine the role of proteins in complex proteinomic networks [2], and finally to identify nodes with good "spreading" properties in epidemiological studies [4].

Efficient centralized algorithms for the k-core decomposition already exist [3]. Here, we consider the distributed version of this problem, motivated by the following scenarios. *One Host, one Node Scenario*: The graph to be analyzed could be a "live" distributed system, such as a P2P overlay, that needs to inspect itself; *one* host is also *one* node in the graph, and connections among hosts are the edges. This information could be used at run-time to optimize the diffusion of messages in epidemic protocols [4].
One Host, Multiple Nodes Scenario: The graph could be so large to not fit into a single host, due to memory restrictions; or its description could be inherently distributed over a collection of hosts, making it inconvenient to move each portion to a central site. So, *one* host stores *many* nodes and their edges.

The two scenarios turn out to be related: the former can be seen as a special case of the "inherent distribution" of the latter taken to its extreme consequences, with each host storing only one node and its edges.

[*]This work is supported by the Autonomous Security project, financed by MIUR Programme PRIN 2008.

Main Results

The main result of our work is the definition of a distributed algorithm able to efficiently compute the coreness index for the *one host, one node* and *one host, multiple nodes* cases.

Our distributed algorithm is based on the property of locality of the k-core decomposition: due to the maximality of cores, the coreness of node u is the largest value k such that u has at least k neighbors that belong to a k-core or a larger core. More formally,

Theorem 1 (Locality). *For each $u \in V$, $k(u) = k$ iff*

(i) there exist a subset V_k of the neighbours of u such that $|V_k| = k$ and $\forall v \in V_k : k(v) \geq k$;

(ii) there is no subset V_{k+1} of the neighbours of u such that $|V_{k+1}| = k+1$ and $\forall v \in V_{k+1} : k(v) \geq k+1$.

The locality property tells us that the information about the coreness of the neighbors of a node is sufficient to compute its own coreness. Our algorithm works as follows. Each node produces an *estimate* of its own coreness and communicates it to its neighbors. The initial estimate is set equal to the node degree. Each nodes receives estimates from its neighbors and uses them to recompute its own estimate. In the case of a change, the new value is sent to the neighbors and the process goes on until convergence. The complete algorithm, together with optimizations, is reported in [5].

The procedure can be easily generalized to the case where a host x is responsible for a collection of nodes $V(x)$. In this case, x runs the algorithm on behalf of its nodes, storing the estimates for all of them and sending messages to the hosts that are responsible for their neighbors. The algorithm can be optimized by having each node x, upon reception of a message for a node $u \in V(x)$, to "internally emulate" the estimation update protocol. The estimates received from outside can indeed generate new estimates for some of the

Name	$\|V\|$	$\|E\|$	$\oslash$	d_{max}	k_{max}	k_{avg}	t_{avg}	t_{min}	t_{max}	m_{avg}	m_{max}
1) CA-AstroPh	18 772	198 110	14	504	56	12.62	19.55	18	21	47.21	807
2) CA-CondMat	23 133	93 497	15	280	25	4.90	15.65	14	17	13.97	410
3) p2p-Gnutella31	62 590	147 895	11	95	6	2.52	27.45	25	30	9.30	131
4) soc-sign-Slashdot090221	82 145	500 485	11	2 553	54	6.22	25.10	24	26	29.32	3 192
5) soc-Slashdot0902	82 173	582 537	12	2 548	56	7.22	21.15	20	22	31.35	3 319
6) Amazon0601	403 399	2 443 412	21	2 752	10	7.22	55.65	53	59	24.91	2 900
7) web-BerkStan	685 235	6 649 474	669	84 230	201	11.11	306.15	294	322	29.04	86 293
8) roadNet-TX	1 379 922	1 921 664	1049	12	3	1.79	98.60	94	103	4.45	19
9) wiki-Talk	2 394 390	4 659 569	9	100 029	131	1.96	31.60	30	33	5.89	103 895

Table 1: Results with the one-to-one algorithm. Name of the data set, number of nodes, number of edges, diameter, maximum degree, maximum coreness, average coreness, average-minimum-maximum number of cycles to complete, average/maximum number of messages sent per node.

nodes in $V(x)$; in turn, these can generate other estimates, again in $V(x)$; and so on, until no new internal estimate is generated and the nodes in $V(x)$ become quiescent. At that point, all the new estimates that have been produced by this process are sent to the neighboring hosts, where they can ignite these cascading changes all over again. Such an optimization proved to reduce consistently the number of messages sent. The algorithm can be proved to be correct and to eventually terminate [5].

Theorem 2 (Safety). *During the execution, the local estimate of coreness index at each node is always larger or equal than the real coreness index.*

By induction, we can prove that the algorithm eventually converges to the exact value.

Theorem 3 (Liveness). *There is a time after which the local estimate of the coreness index is always equal to the real coreness index.*

Both centralized termination mechanisms as well as distributed ones can be introduced. As shown in [5], most of real-world graphs can be completed in a very small number of rounds (few tens); if an approximate k-core decomposition could be sufficient, running the protocol for a fixed number of rounds is also an option. The next results provide bounds on the execution time, i.e., the time it takes for the distributed algorithm to converge to the exact value and become quiescent. The proofs are again in [5].

Theorem 4. *Given a graph $G = (V, E)$, the execution time is bounded by $1 + \sum_{u\in V} [d(u) - k(u)]$.*

A bound on the execution time that depends only on the graph size (and not on the knowledge of the actual coreness index of nodes) can be introduced.

Theorem 5. *The execution time is not larger than N.*

The result can be slightly improved as:

Corollary 1. *Let K be the number of nodes with minimal degree in G. Then the execution time on G is not larger than $N - K + 1$ rounds.*

While one may intuitively associate the execution time of the algorithm to the diameter of the network, this turns out not to be always the case. In [5] we identified a class of graphs with constant diameter 3 having execution time equal to $N - 1$.

We have also analysed the message complexity of the proposed algorithm, leading to the following result:

Corollary 2. *Given a graph $G = (V, E)$, the message complexity is bounded by $\left[\sum_{v\in V(G)} d^2(v)\right] - 2M$, where Δ is the largest degree of nodes in the graph.*

Based on such result, the message complexity of the distributed k-core computation results $O(\Delta \cdot M)$.

Outlook

The *one host, one node* scenario is relevant for optimizing diffusion of messages in unstructured P2P systems. The *one host, multiple nodes* scenario may lend itself to a number of applications related to the analysis of massive-scale networks. The next step is the implementation and optimization of the proposed techniques in frameworks like Hadoop [8].

References

[1] Alvarez-hamelin, J. I., Barrat, A., and Vespignani, A. Large scale networks fingerprinting and visualization using the k-core decomposition. In *Proc. of NIPS* (2005), vol. 18, MIT Press, pp. 41–50.

[2] Bader, G., and Hogue, C. Analyzing yeast protein–protein interaction data obtained from different sources. *Nature biotechnology 20*, 10 (2002), 991–997.

[3] Batagelj, V., and Zaversnik, M. An $O(m)$ algorithm for cores decomposition of networks. *CoRR cs.DS/0310049* (2003). http://arxiv.org/abs/cs.DS/0310049.

[4] Kitsak, M., Gallos, L. K., Havlin, S., Liljeros, F., Muchnik, L., Stanley, H. E., and Makse, H. A. Identification of influential spreaders in complex networks. *Nature Physics 6* (Nov. 2010), 888–893.

[5] Montresor, A., De Pellegrini, F., and Miorandi, D. Distributed k-core decomposition. *CoRR cs.OH/1103.5320* (2011). http://arxiv.org/pdf/1103.5320v2.

[6] Newman, M. The structure and function of complex networks. *SIAM Review 45* (2003), 167–256.

[7] Seidman, S. Network structure and minimum degree. *Social Networks 5*, 3 (1983), 269–287.

[8] White, T. *Hadoop: The definitive guide (2nd ed.)*. Yahoo Press, 2010.