Clique covering of large real-world networks

Alessio Conte, Roberto Grossi, Andrea Marino

▶ **To cite this version:**

Alessio Conte, Roberto Grossi, Andrea Marino. Clique covering of large real-world networks. 31st Annual ACM Symposium on Applied Computing (SAC 2016), Apr 2016, Pisa, Italy. pp.1134-1139, 10.1145/2851613.2851816 . hal-01388477

HAL Id: hal-01388477

https://inria.hal.science/hal-01388477

Submitted on 16 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Clique Covering of Large Real-World Networks

Alessio Conte Roberto Grossi Andrea Marino*
Dipartimento di Informatica
Università di Pisa
Largo Bruno Pontecorvo 3, Pisa
{conte,grossi,marino}@di.unipi.it

ABSTRACT

The edge clique covering (ECC) problem deals with discovering a set of (possibly overlapping) cliques in a given network, such that each edge is part of at least one of these cliques. We address the ECC problem from an alternative perspective reconsidering the quality of the cliques found, and proposing more structured criteria with respect to the traditional measures such as minimum number of cliques. In the case of real-world networks, having millions of nodes, such as social networks, the possibility of getting a result is constrained to the running time, which should be linear or almost linear in the size of the network. Our algorithm for finding ECCs of large networks has linear-time performance in practice, as our experiments show on real-world networks whose number of nodes ranges from thousands to several millions.

CCS Concepts

•**Theory of computation → Design and analysis of algorithms; Graph algorithms analysis;**

Keywords

Real-World Networks, Large Networks, Edge Clique Cover

1. INTRODUCTION

The usage of cliques and similar dense subgraphs such as N-cliques, N-clans, K-plexes, K-cores, F-groups, in social network analysis is now consolidated and many methods exist to discover them under several contexts [21]. A clique is a subset of the nodes in a network such that every two nodes are connected. Among the variations of clique discovering, the *edge clique covering* (ECC) problem deals with finding a set of (possibly overlapping) cliques in a given network, such that each edge of the network is part of at least one of these cliques.

In this paper, we propose new algorithmic tools for ECC that are potentially useful for social network analysis and, at the same time, are very easy to implement. Our tools can quickly process graphs of massive size to find their ECC in practically linear time and use heuristics for attacking the ECC problem, which was born in a different context [14] and is hard to solve exactly.

Previous work. In the literature, problems related to ECC have been independently addressed in the mid 70's in social network analysis [3, 19]. From a computational point of view, its vertex clique covering version, where the cliques are disjoint and the vertices are to be covered, is one of Karp's original problems shown to be NP-complete in his famous 1972 paper. The ECC appeared with different terminology such as keyword conflict [14], covering by cliques, intersection graph basis, or clique partition [11], as discussed in [6]. The latter reports that the problem of finding the ECC having the *minimum*[1] number of cliques is NP-hard [18], even for planar graphs [4]. The parameterized exact algorithms at the state of the art [13] are probably optimal [6]. Although they can be applied to social network analysis, they do not scale to large networks.

Some heuristic algorithms for cliques work surprisingly well for large real-world networks (e.g. [1, 2, 5, 8, 20]), as even approximation algorithms with some guarantee are too expensive. Indeed, due to the massive size of available real-world large networks, it is convenient to consider algorithms or heuristics taking (almost) linear time in the size of the network. Some papers try to achieve this goal for ECC.

The state of the art in this direction is Gramm et al. [12], which gives an efficient implementation of the original Kellerman heuristic [14], with the post-processing step of Kou et al. [15]. However, their bounds are not linear. For a network with n vertices and m edges, the original results by Kellerman finds a clique covering in $O(nm^2)$ time. The post-processing of Kou et al. requires at least $\Omega(nm)$ time in the worst case. Gramm et al. show how to obtain the same heuristic solution in total $O(nm)$ time, using $O(n^2)$ cells of memory as space.

*Work partially supported by Italian MIUR under PRIN 2012C4E3KT research project AMANDA.

[1]Finding an arbitrary ECC is easy, for example the trivial one made up of just the individual edges.

Our results. In this paper, we present some new and simple algorithmic tools for heuristics on ECC that are tailored for large real-word networks. The classical well-studied measure is minimizing the number of cliques. Recently also the minimum assignment objective has been proposed, which is minimizing the sum of the sizes of the cliques [7]. In addition to them, we consider an alternative view to the ECC problem, where more options are available.

Our view is well represented by the *clique cover graph* shown in Fig. 2 for the example network in Fig. 1. This bipartite graph has nodes representing the cliques of the ECC on the left, and nodes representing the vertices of the input network on the right. There is an edge to indicate that a given vertex is part of the given clique. On the right, we may represent the edges of the network rather than the nodes. A formal definition is given in Section 2. This graph is reminiscent of the clique representation in intersection graph theory [9].

Looking at the classical measure of minimizing the number of cliques in the ECC, corresponds to having the minimum number of nodes on the left side of the clique cover graphs of the candidate ECCs. On the other hand having a minimum assignment, as in [7], corresponds to minimizing the number of edges in this graph. However, more information on the quality of the solution can be obtained from the cover graph, which suggests using also other measures when evaluating heuristics for the ECC problem. For example, the weight of an ECC, the clique size distribution, or the cover index distribution described in Section 2.

Our contributions in this paper are summarized below.

First, we introduce an experimental analysis of heuristics for the ECC problem on real-word networks that is based on a more informative evaluation tool, based on the clique cover graph, rather than the mere number of cliques in ECC. In other words, we do not look just at the minimization problem but we consider also several statistics on all the cliques emerging from an ECC, based on these networks.

Second, we propose a new efficient and simple framework for fast algorithmic tools that scale well for real-world large graphs and perform well according to a variety of measures based on the clique cover graph, when compared to the state of the art. They require $O(m+n)$ space and their time cost can be upper bounded as $O(m\Delta)$, where Δ is the maximum degree of the network. The actual running time is linear in m in our experiments.

Finally, we perform an experimental study for our framework on a set of real-world and synthetic networks. The comparison among 20 variants of our algorithms and the state of the art involved a data set of 44 networks to select the best variants. We adopted some measures based on the clique cover graph, and compared with the state of the art on an extensive data set. In this paper, we report the results concerning ECC-`rc`, one of our most effective algorithms. We show that this algorithm is faster than existing methods and it improves in practice the quality of the solution, also according to the traditional measures such as the minimum number of cliques in ECC. We perform extensive experiments involving 160 networks, showing that we can efficiently deal with large real-world networks.

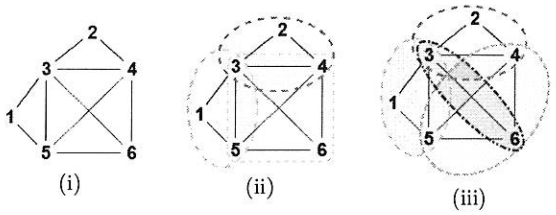

Figure 1: Graph (i) taken from [15], and two examples of ECCs in (ii) and (iii).

We hope that our findings can inspire further work on network analysis that uses clique cover graphs, as the latter ones could be customized to deal with new quality measures that depend on the domain applications.

2. CLIQUE COVERING

For our study, we consider a network as an undirected connected graph $G = (V, E)$, where V is the set of n vertices and E is the set of m edges. A clique $C \subseteq V$ is a set of vertices that are pairwise connected by edges in E, e.g., $u, v \in E$ for each pair of distinct vertices $u, v \in C$. (Equivalently, the induced subgraph $G[C]$ is complete.) In the following we will write that an edge $x, y \in C$ when both its endpoints x and y are in the clique C. A clique C is *maximal* if there is no other clique C' such that $C \subset C'$.

An *edge clique cover* for G (in short, ECC) is a set of cliques $C_1, C_2, \ldots, C_k$ such that *(1)* no clique C_i is contained in another clique C_j, where $i \neq j$, and *(2)* for each edge $u, v \in E$ there is at least one clique C_i such that $u, v \in C_i$ [15, 13].[2] Note that an ECC always exist (e.g. choose $k = m$ and the individual edges as cliques). Maximality on each C_i is not required in the definition of ECC, but it is not difficult to see how to transform each C_i into a maximal clique, if this is needed. An ECC is *minimal* if there is no clique contained in the union of the others, namely, it is $C_i \not\subseteq \cup_{j \neq i} C_j$ for $1 \leq i \leq k$. The clique number of G is the smallest k such that an ECC of k cliques exists for G. Fig. 1 illustrates our definitions.

Beside the possible definitions of cliques, the quality of a solution, i.e. a clique set, has been often measured as its cardinality, that is, the quality of the ECC $C_1, \ldots, C_k$ is simply k. Hence, the so called MINIMUM ECC aims to minimize the size k. Although finding a MINIMUM ECC has been already proved to be hard and not arbitrarily approximable, heuristics are often satisfying since in practice the quality of a solution could also depend on the application.

2.1 Clique Cover Graph

It is convenient to define a *clique cover graph* for the given

[2]Other definitions exist in the literature. In vertex clique cover, condition *(2)* is replaced by asking that for each vertex v there is a clique C_i such that $v \in C_i$. In clique partition, the cliques $C_1, C_2, \ldots, C_k$ form a partition of V. Note that these definition loose information regarding the edges of G. We focus on ECC as we can reconstruct G from it.

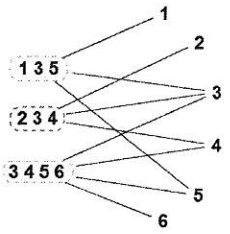

Figure 2: Clique cover graph for the ECC in Fig. 1(ii).

ECC $C_1, \ldots, C_k$ to better highlight its properties and assess its qualities (see Fig. 2). It is a bipartite undirected graph $G' = (V_1' \cup V_2', E')$, such that V_1' contains a node for each clique, hence $|V_1'| = k$, and V_2' contains a node for each vertex in V, hence $|V_2'| = n$. There is an edge $u_1, u_2 \in E'$, where $u_1 \in V_1'$ represents a clique C_i in the ECC and $u_2 \in V_2'$ represents a vertex $v \in V$, if and only if $v \in C_i$.

Depending on the application, we may define a clique cover graph with V_2' representing the edges in E, where $|V_2'| = m$. In general, the choice of V_2' representing either the vertices in V or the edges in E will be clear from the context.

While traditional work aims at minimizing the cardinality of V_1 (i.e., finding the ECC of minimum cardinality k or giving an approximation) and E' (i.e., minimum assignment), other possibilities can be offered. Not only we can obtain the measures previously known in the literature by looking at the clique cover graphs for the possible ECCs of G, but we can also define some additional ones in a smooth way.

1. Finding the ECC of *minimum weight* $\sum_{i=1}^{k} |C_i|$ is equivalent to finding the ECC whose clique cover graph has the minimum number of edges (see also [7]).
2. Finding the *clique size distribution* in the given ECC is equivalent to the degree distribution of the nodes in V_1' for the corresponding clique cover graph.
3. Finding the *covering index distribution* in the given ECC, namely, computing the number y of elements from V_2' that are contained in x cliques of the ECC, for all feasible values of x and y, is equivalent to the degree distribution of the nodes in V_2' for the corresponding clique cover graph.

3. FRAMEWORK

Our heuristics for finding ECCs rely on a simple algorithmic framework, summarized in Algorithm 1, taking a graph $G = (V, E)$ as input. It begins with an empty ECC C, and during the steps, cliques are added to C. At any step the edges in E are partitioned as *covered* and *uncovered*: edge u, v is covered if there exists a clique $C_i \in C$ such that $u, v \in C_i$; it is uncovered otherwise. Initially, all edges in E are uncovered. As long as there are uncovered edges, one of them is chosen by `SELECT_UNCOVERED_EDGE`, described in Section 3.1. Let us call u, v this edge. We then find a new clique R that contains u, v using `FIND_CLIQUE_OF`, described in Section 3.2. Note that the clique R must be new by definition as u, v is uncovered. We then add R to C, and mark all of the edges $x, y \in R$ as covered.

Algorithm 1: General framework

Input: A graph $G(V, E)$
Output: A covering C of G
All edges in E are marked as *uncovered*
$C \leftarrow \emptyset$
while *there are uncovered edges* **do**
  $u, v \leftarrow$ `SELECT_UNCOVERED_EDGE()`
  $R \leftarrow$ `FIND_CLIQUE_OF`(u, v)
  $C \leftarrow C \cup R$
  Mark all edges of R as covered

We experimentally studied a total of 20 variants of our framework, where just a subset of them was effective for ECC. These variants differ from the way they implement the two operations above. In this paper, we single out one of the best variants, called ECC-`rc`, that selects an uncovered edge u, v in uniformly random fashion, and then finds the clique R containing u, v by extracting each time the node z (if it exists) with the maximum number of uncovered edges incident to R. Specifically, these two operations are implemented as follows.

3.1 Operation `SELECT_UNCOVERED_EDGE`

Given a node $u \in V$, we denote its set of neighbors by $N(u) = \{v \mid u, v \in E\}$ and its degree by $d(u) = |N(u)|$. We also denote the set of uncovered edges by $U \subseteq E$, and define the uncovered neighbors of $u \in V$ as $N_U(u) = \{v \mid u, v \in U\}$. The operation performs the following choice.

`r` | *random*: return an uncovered edge u, v from the edge set U uniformly at random, assuming that U is nonempty

3.2 Operation `FIND_CLIQUE_OF`

Given an uncovered edge u, v in the input graph G, we can find a clique R containing u, v by following the idea of the Bron-Kerbosh technique as shown in Algorithm 2.

Algorithm 2: `FIND_CLIQUE_OF`

Input: A graph $G(V, E)$, an uncovered edge $(u, v) \in E$
Output: A clique R containing (u, v)
$R \leftarrow \{u, v\}$
$P \leftarrow N(u) \cap N(v)$
$z \leftarrow$ `EXTRACT_NODE`(P)
while $z \neq$ *null* **do**
  $R \leftarrow R \cup \{z\}$
  $P \leftarrow P \cap N(z)$
  $z \leftarrow$ `EXTRACT_NODE`(P)

After initializing R with u, v, the candidate set P is built by intersecting the neighbors of u and v. Using `EXTRACT_NODE`, a node z is suitably extracted from P, if it exists (i.e. $z \neq$ *null*), and added to R as z is surely adjacent to all the vertices in R. Since z is now part of R, the vertices in

the candidate set P must be also neighbors of z, so P is updated with the intersection with $N(z)$. In general, each of the vertices in P is a neighbor of all the vertices in R. The expansion of R terminates when either P is empty (and thus R is maximal) or z cannot be found in P: in both cases it is $z = null$. We adopt the following variant for `EXTRACT_NODE`.

c	*clean*: return any vertex z (if any) yielding maximum size $\|N_U(z) \cap R\| > 0$; otherwise, return $z = null$.

This variant aims at maximizing the number of uncovered edges that will become covered after adding z to the current clique R. We remark that some variants are difficult to compute: for example, finding the largest clique containing (uncovered) edge u, v is NP-hard.

4. PERFORMANCE

It is easy to show that Algorithm 1 is correct, i.e. returns a clique cover for the given undirected graph. Moreover it uses $O(m + n)$ space and takes $O(m\Delta)$ time for variant ECC-`rc`, where n is the number of vertices, m is the number of edges, and Δ is the maximum vertex degree.

In practice, its running time is linear in m as shown next with a dataset of 160 real-world graphs taken from `lasagne-unifi.sourceforge.net`.

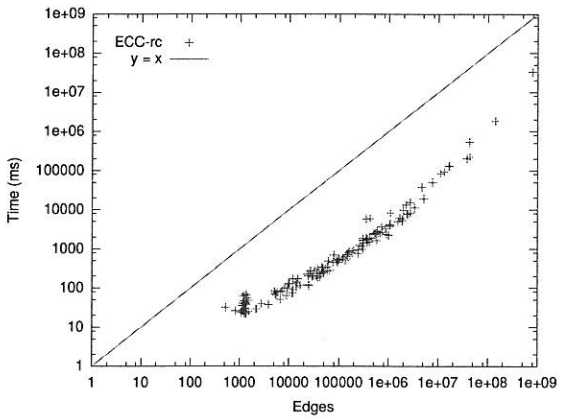

Figure 3: Time used by ecc-rc to compute clique covers for 160 real-world graphs as a function of the number of edges.

Fig. 3 reports the time used by ECC-`rc` to compute clique covers as a function of the number of edges in the graph. In other words, for each graph in our dataset, we have run ECC-`rc` placing a red cross in position (x, y) whether the graph has x edges and ECC-`rc` spent y milliseconds to finish the computation. Looking at the plot, the red crosses seem to be disposed over a line which is parallel to the line $y = x$ suggesting a linear running time of our algorithm. This hypothesis is confirmed by the results of a linear regression over the original data (not in log scale): the time y can be related to the number of edges x by using $y = a \cdot x + b$, where $a = 0.041$ and $b = -71005.486$; these estimates have respectively p-value $7.563 \cdot 10^{-136}$ and 0.0174 (the correlation is 98.961%). Due to the statistical significance of our tests, we argue that ECC-`rc` is linear when dealing with large real-world networks.

This feature clearly emerges with the largest networks we considered (see Table 1): for the largest one, a snapshot of the web, our algorithm terminates after 32 733 seconds, which is quite fast considering the size of the network, which is processed on a single core. Our computing platform is a 24 core machine with Intel(R) Xeon(R) CPU E5-2620 v3 at 2.40GHz, with 128GB of shared memory. The operating system is a Ubuntu 14.04.2 LTS, with a Linux kernel version 3.16.0-30.

CATEGORY	GRAPH	NODES	EDGES	TIME (MS)
col.	imdb	913 201	37 588 613	202 065
web	enwiki	13 834 640	42 336 692	526 448
social	LiveJournal1	4 847 571	42 851 236	225 213
P2P	p2p	5 792 297	142 038 401	1 845 340
web	web	39 454 463	783 027 125	32 733 749

Table 1: Sample of the largest networks we considered with the running time of ecc-rc.

5. EXPERIMENTAL COMPARISON

We use as a *baseline* what we call g-ALG, which is the heuristic in Gramm et al. [12]. This is an improved implementation of the Kellerman algorithm [14], with the postprocessing step of Kou et al. [15] to produce a minimal ECC. As far as we know, g-ALG is the fastest heuristic in the state of the art, achieving also good results in terms of the traditional cost that minimizes the number of cliques found. Moreover, recent works pointed out the speed and the quality of the solution provided by g-ALG also considering the minimum assignment objective [7], suggesting its usage for larger graphs. Hence, we think that g-ALG is a natural choice to evaluate the pros and the cons of our solutions.

We implemented our variant ECC-`rc` and g-ALG in Java 1.8.0_25, and the code will be released as open source.[3]

5.1 Data sets

All the networks in our experiments have been collected from the datasets SNAP (available at `snap.stanford.edu/`) and LASAGNE (at `lasagne-unifi.sourceforge.net/`). Real-world networks include autonomous system, biological, citation, collaboration, communication, word-adjacency, peer-to-peer, social networks, and web networks. Synthetic networks include networks generated using Erdos-Renyii [10], Forest Fire [17], and Kronecker [16]. For more details about the dataset, see Table 2. Note that we could not include the largest networks of LASAGNE (e.g. the ones in Table 1), since g-ALG timed out on these instances.

[3]We also considered the OCaml implementation of g-ALG provided by the authors: while for small networks we obtained results consistent with our implementation, the performance of the original implementation did not allow us to process large networks in our dataset.

CATEGORY	GRAPH	NODES	EDGES
AS	itdk0304_rlinks	192 244	609 066
biological	Drosophila	10 625	40 781
biological	Homo	1 027	1 166
biological	hprd_pp	9 465	37 039
biological	ppi_gcc	37 333	135 618
citation	cit-HepPh	34 546	420 876
citation	cit-HepTh	27 770	352 284
citation	citeseer	259 217	532 040
citation	hep-th-citations	27 400	352 021
collaboration	Cond_mat_95-99	22 015	58 578
collaboration	advogato	7 418	42 892
collaboration	ca-AstroPh	18 771	198 050
collaboration	ca-CondMat	23 133	93 439
collaboration	ca-GrQc	5 241	14 484
collaboration	ca-HepPh	12 006	118 489
collaboration	ca-HepTh	9 875	25 973
communication	email-Enron	36 691	183 830
communication	email-EuAll	265 214	364 480
P2P	p2p-Gnutella31	62 586	147 891
social	soc-Slashdot0811	77 360	469 180
social	soc-Slashdot0902	82 168	504 230
social	trust	49 288	381 036
social	wiki-Vote	7 115	100 761
synthetic	Gnp_1e4	10 000	59 849
synthetic	Gnp_2e3	2 000	8 994
synthetic	Gnp_5e3	5 000	24 809
synthetic	forest1e4	10 000	49 354
synthetic	forest1e4_2	10 000	153 925
synthetic	forest5e4	50 000	243 441
synthetic	ud_1e3	1 000	16 727
synthetic	ud_1e4	10 000	313 726
synthetic	ud_2e3	1 999	35 697
synthetic	ud_5e3	4 998	97 027
synthetic	us_1e3	1 000	14 334
synthetic	us_1e4	10 000	242 533
synthetic	us_2e3	2 000	37 928
synthetic	us_5e3	5 000	135 833
web	GoogleNw	15 763	148 585
word-adj	frenchBookInter	8 325	23 841

Table 2: Summary of the dataset used for the experimental comparison (Section 5).

5.2 Clique Quality

Our guidelines for the experiments are based on the clique cover graphs $G' = (V_1' \cup V_2', E')$ for the graphs $G = (V, E)$ in the datasets, as described in Section 2.1. In particular, we considered the following ones to establish the results.

(a) Clique size distribution: for each feasible value t, we compute the number of cliques in ECC that have size t. In the clique cover graph, this is equivalent to the degree distribution of V_1'. With this information, we also get the number of cliques, the (local) maximum clique size, and the average clique size.

(b) Node covering index distribution: given a node u in G, let $c(u)$ be the number of cliques containing u in the ECC. For each feasible value t, we compute the number of nodes u having $c(u) = t$. In the clique cover graph where $V_2' = V$, this is equivalent to the degree distribution of V_2'. With this information, we also get the maximum $c(u)$ and the average value.

(c) Edge covering index distribution: as in (b), except that the cover index $c(u, v)$ is the number of cliques containing edge u, v in the ECC. In the clique cover graph where $V_2' = E$, this is equivalent to the degree distribution of V_2'.

measure	goal	vs g-ALG
number of cliques	MIN	5%
maximum clique size	MAX	2%
average clique size	MAX	-5%
coefficient of variation — range: 0.013–0.016		

(a) clique size distribution

measure	goal	vs g-ALG
maximum node covering	MIN	23%
average node covering	MIN	11%
coefficient of variation — range: 0.026–0.035		

(b) node covering index distribution

measure	goal	vs g-ALG
maximum edge covering	MIN	29%
average edge covering	MIN	12%
coefficient of variation — range: 0.021–0.026		

(c) edge covering index distribution

measure	goal	vs g-ALG
average time	MIN	62%
std error	MIN	52%

(d) time needed to compute the covering

Table 3: Summary of the direct comparison between ecc-rc and g-alg for each measure and for the speed

For the sake of completeness, we report the *coefficient of variation*, in short *CV*, which is a measure of dispersion of a distribution and corresponds to the standard deviation divided by the mean. This measure is used as a normalized way to evaluate the variability of a covering (the closer to 0, the better is).

Table 3 summarizes the comparison between ECC-rc and g-ALG. It can be read as follows. The first column indicates the measures described above. The second column indicates whether it is good to minimize or maximize the measure in the corresponding row. The third column reports how ECC-rc compares to g-ALG: a value of p% for the minimization indicates that the measure found by ECC-rc is p% smaller than that provided by g-ALG; for the maximization, it indicates that the measure is p% larger than that by g-ALG. In other words, let a be the measure found by ECC-rc and b be the measure found by g-ALG; in the case of minimization we report $p = (1 - a/b) \cdot 100$, while in the case of maximization we report $p = (a/b - 1) \cdot 100$, where the value of p is truncated.

It is worth observing that with respect to the clique size distribution, ECC-rc and g-ALG are similar, while for node and edge covering indexes, ECC-rc clearly improves g-ALG. In particular, since minimizing the average node covering corresponds to minimizing the assignment, we improve by 11% on the average the assignments by g-ALG, whose solutions have been already observed to be often close to the optimum [7]. The improvement of ECC-rc over g-ALG is larger when considering the maximum node covering, that is 23%. A better improvement can be seen looking at the edge covering: we improve the average by 12% and the maximum by 29%.

The average time used by ECC-rc to obtain the covering is

62% smaller than the one used by g-ALG. This means that our algorithm employs on average almost one third of the time employed by g-ALG to complete the task (the standard error for the time values by ECC-rc is 52% smaller with respect to the one by g-ALG). Fig. 4 clearly shows that in the case of larger graphs, the situation is much worse for g-ALG: for each graph in our dataset we draw a red cross which is the ratio between the time needed by g-ALG and the one needed by ECC-rc, as a function of the number of the edges. Indeed the plot shows that very often the performance of g-ALG is several orders of magnitude worse than that by ECC-rc. The variability seems to dramatically affect the performance of g-ALG as the number of edges increases. Furthermore, we remark that the comparison does not take into account the (large) networks that only ECC-rc was able to process.

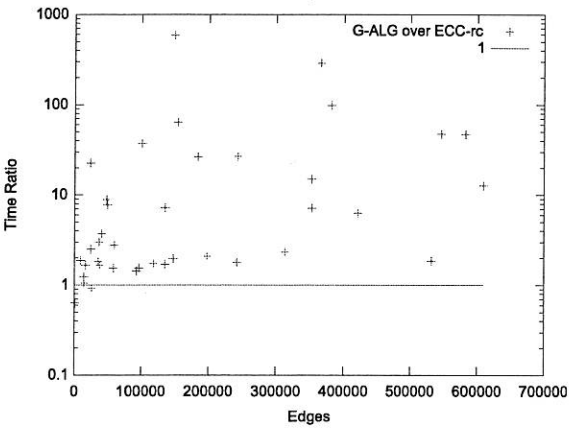

Figure 4: Ratio between the time used by g-alg and ecc-rc for each graph, as a function of the number of the edges (y-axis is in log-scale).

6. CONCLUSIONS

In this paper we proposed new and simple heuristics to deal with ECC. Inspired by the *clique cover graph*, we have introduced several measures to assess the quality of a solution rather just the classical ones, e.g. minimizing the number of cliques. Our algorithm is part of a simple framework and scales well to deal with large real-world networks. Moreover, we showed that our algorithm improves the state of the art according to the several measures we considered based on the clique cover graph and the running time, which can be considered linear in practice.

Acknowledgments. We thank Paolo Boldi and Sebastiano Vigna for providing us the machines to run some preliminary experiments.

7. REFERENCES

[1] J. Abello, P. M. Pardalos, and M. G. C. Resende. On maximum clique problems in very large graphs, 1999.

[2] E. A. Akkoyunlu. The enumeration of maximal cliques of large graphs. *SIAM J. Comput.*, 2(1):1–6, 1973.

[3] J. Berger. *Status characteristics and social interaction: an expectation-states approach.* Elsevier, 1977.

[4] M. Chang and H. Muller. On the tree-degree of graphs. In *WG: Graph-Theoretic Concepts in Computer Science*, 2001.

[5] J. Cheng, Y. Ke, A. W.-C. Fu, J. X. Yu, and L. Zhu. Finding maximal cliques in massive networks, 2011.

[6] M. Cygan, M. Pilipczuk, and M. Pilipczuk. Known algorithms for edge clique cover are probably optimal. In *Proc. ACM-SIAM Symposium on Discrete Algorithms*, pages 1044–1053. SIAM, 2013.

[7] J. M. Ennis, C. M. Fayle, and D. M. Ennis. Assignment-minimum clique coverings. *J. Exp. Algorithmics*, 17:1.5:1.1–1.5:1.17, 2012.

[8] D. Eppstein and D. Strash. Listing all maximal cliques in large sparse real-world graphs. In *Proceedings of the 10th International Conference on Experimental Algorithms*, SEA'11, pages 364–375. Springer, 2011.

[9] P. Erdos, A. W. Goodman, and L. Pósa. The representation of a graph by set intersections. *Canad. J. Math*, 18(106-112):86, 1966.

[10] P. Erdos and A. Rényi. On the evolution of random graphs. *Bull. Inst. Internat. Statist*, 38(4):343–347, 1961.

[11] M. R. Garey and D. S. Johnson. Computers and Intractability: A Guide to the Theory of NP-Completeness. *Computers and Intractability*, page 340, 1979.

[12] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Data reduction, exact, and heuristic algorithms for clique cover. In *ALENEX*, pages 86–94. SIAM, 2006.

[13] J. Gramm, J. Guo, F. Hüffner, and R. Niedermeier. Data reduction and exact algorithms for clique cover. *J. of Experimental Algorithmics (JEA)*, 13:2, 2009.

[14] E. Kellerman. Determination of keyword conflict. *IBM Technical Disclosure Bulletin*, 16(2):544–546, 1973.

[15] L. T. Kou, L. J. Stockmeyer, and C.-K. Wong. Covering edges by cliques with regard to keyword conflicts and intersection graphs. *Communications of the ACM*, 21(2):135–139, 1978.

[16] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *The J. of Machine Learning Research*, 11:985–1042, 2010.

[17] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. KDD '05, pages 177–187. ACM, 2005.

[18] J. Orlin. Contentment in graph theory: Covering graphs with cliques. *Indagationes Mathematicae (Proceedings)*, 80(5):406–424, 1977.

[19] F. Roberts. *Discrete Mathematical Models, with Applications to Social, Biological, and Environmental Problems.* Prentice-Hall, 1976.

[20] R. A. Rossi, D. F. Gleich, A. H. Gebremedhin, and M. M. A. Patwary. Fast maximum clique algorithms for large graphs. In *WWW*, pages 365–366, 2014.

[21] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*, vol. 8. Cambridge University Press, 1994.