Graph Clustering Evaluation Metrics as Software Metrics

MILOŠ SAVIĆ and MIRJANA IVANOVIĆ, University of Novi Sad

Graph clustering evaluation (GCE) metrics quantify the quality of clusters obtained by graph clustering (community detection) algorithms. In this paper we argue that GCE metrics can be applied on graph representations of software systems in order to evaluate the degree of cohesiveness of software entities. In contrast to widely known cohesion measures used in software engineering, GCE metrics do not ignore external dependencies among software entities, but contrast them to internal dependencies to quantify cohesion. Using the theoretical framework of cohesion measurement in software engineering introduced by Briand et al. we investigate the properties of GCE metrics. Our analysis shows that GCE metrics are theoretically sound with respect to the monotonicity and merge property, but also reveals that they possess certain limitations whose importance is discussed in the paper. Finally, we propose a set of research questions for further empirical studies on this topic.

Categories and Subject Descriptors: D.2.8 [**Software engineering**]: Metrics—*Product metrics*

General Terms: Measurement, Theory

Additional Key Words and Phrases: cohesion, clustering, metrics

1. INTRODUCTION

Graph clustering is one of the most important method in complex network analysis [Boccaletti et al. 2006]. Identification of clusters (also known as communities or modules) in a network helps us to reduce the complexity of the network in order to be able to understand its underlying structure. Other applications of graph clustering techniques include parallel processing of graph based data, route planning, image segmentation and VLSI physical design [Buluç et al. 2013], to mention a few. Intuitively speaking, a cluster in a network is a part of the network where connections among members of the cluster are much denser than with the rest of the network [Fortunato 2010]. There is a variety of graph clustering (community detection) algorithms [Schaeffer 2007]. Naturally, they are accompanied by graph clustering evaluation (GCE) metrics that quantify the quality of partitions produced by them [Leskovec et al. 2010].

"Low coupling, high cohesion" is one of the basic design principles in software engineering [Yourdon and Constantine 1979]. This principle states that the coupling between modules of a software system has to be minimal as possible keeping at the same time strong relations between elements of each module. The main idea of this work is that highly cohesive modules that are loosely coupled to other modules can be viewed as clusters in a graph that encompasses software entities at lower levels of

This work was supported by the Serbian Ministry of Education, Science and Technological Development through project *Intelligent Techniques and Their Integration into Wide-Spectrum Decision Support*, no. OI174023.
Author's address: M. Savić, M. Ivanović, Department of Mathematics and Informatics, Faculty of Sciences, University of Novi Sad, Trg Dositeja Obradovića 4, 21000 Novi Sad, Serbia, email: {svc, mira}@dmi.uns.ac.rs.

In: Z. Budimac, T. Galinac Grbac (eds.): Proceedings of the 3rd Workshop on Software Quality, Analysis, Monitoring, Improvement, and Applications (SQAMIA), Lovran, Croatia, 19.-22.9.2014, published at http://ceur-ws.org.

abstraction. Therefore, the aim of this paper is to investigate existing GCE metrics as metrics reflecting cohesion of software modules.

The rest of the paper is structured as follows. Related work is presented in Section 2. Graph clustering evaluation metrics explored in this work as software metrics are defined in the next section of the paper. Section 4 explains how GCE metrics can be applied to graph representation of software systems. Theoretical analysis of GCE metrics as software metrics is given in Section 5. The last section concludes the paper and presents research questions for our future work.

2. RELATED WORK

Cohesion of a software module reflects how strongly related are the elements of the module. Perhaps the widest known cohesion metric in software engineering is LCOM (Lack of cohesion in methods) introduced by Chidamber and Kemerer [1994] in their object-oriented metrics suite. As the name of the metric suggests, LCOM is an inverse cohesion metric: a low value of LCOM indicates high cohesion of a class and vice versa. LCOM is based on a specific coupling between methods: two methods in a class are considered as data coupled if they use at least one common class attribute. Then LCOM is the number of non-coupled methods (P) reduced by the number of coupled methods (Q) if $P > Q$, or zero otherwise. From the definition of the LCOM it can be seen that this metric does not take external dependencies into account nor method invocations (another form of method coupling).

The approach of Hitz and Montazeri [1995] to measure cohesion of software entities followed the research of Chidamber and Kemerer. For a class we can construct graph G whose nodes are methods defined in the class, and two methods are connected by an undirected link if they are data coupled. LCOM of Hitz and Montazeri is the number of connected components in G. The same authors also proposed another variant of the same metric where G includes method calls relations. Finally, they introduced a metric called *connectivity* which quantifies how much G is far from being completely connected.

Bieman and Kang [1995] introduced two cohesion metrics called tight class cohesion (TCC) and loose class cohesion (LCC). The basic element in their metrics is again a graph representing a class that encompasses methods of the class. TCC/LCC is the density (the actual number of links divided by the maximal number of links) of a TCC/LCC graph. Two methods are connected in TCC graph if they both access the same variable or there is a direct call between them. LCC graph is an extension of TCC graph that includes indirect method calls.

Lee et al. [1995] introduced a class cohesion metric based on information flow. The basic idea is that the strength of call coupling between invoking and invoked method is determined by the number of parameters of invoked method: the more information passed through formal parameters, the stronger call coupling between methods. Then, the cohesion of a method is defined as the number of calls to other methods multiplied by the number of formal parameters. Finally, the cohesion of a class is the sum of cohesion of its methods.

From the review of widely used software engineering cohesion metrics it can be concluded that the cohesiveness of a software entity is estimated in isolation. In other words, those metrics rely only on internal dependencies, while external dependencies, dependencies reflecting coupling between software modules, are not taken into account. However, external dependencies can be also important when estimating cohesiveness of software modules. Firstly, a module that has much more external than internal dependencies hardly can be considered as strongly cohesive regardless of the density or the connectedness of its internal parts. Secondly, having two modules that have the same degree of internal density the one with the smaller number of external dependencies can be considered as more cohesive compared to the other.

3. GRAPH CLUSTERING EVALUATION METRICS

Let $G = (V, E)$ be a directed graph where V is the set of nodes and E set of links. Let C denote a cluster in V ($C \subseteq V$), and let c be a node from C. An intra-cluster link emanating from c connects c to another node from C, while an inter-cluster link emanating from c connects c to a node that does not belong to C. Intra-cluster out-degree of node c is the number of intra-cluster links emanating from c, while inter-cluster out-degree of node c is the number of inter-cluster links emanating from c.

The most common formulation of the graph partitioning problem asks for a division of the set of nodes into balanced, disjoint subsets of nodes such that the edge cut (links connecting nodes from different clusters) is minimized. Therefore, the basic graph clustering evaluation (GCE) metrics are based on the size of the edge cut. Let E_C denote the size of the cut (the number of inter-cluster links) for cluster C,

$$E_C = \mid \{(x, y)\} : x \in C, y \notin C \mid = \sum_{x \in C} \text{inter-cluster out-degree}(x),$$

I_C the number of intra-cluster links for C,

$$I_C = \mid \{(x, y)\} : x \in C, y \in C \mid = \sum_{x \in C} \text{intra-cluster out-degree}(x),$$

N_C the number of nodes in C, and N the number of nodes in the graph. Then cut based GCE metrics, conductance, expansion and cut-ratio, are defined as follows [Leskovec et al. 2010]:

(1) **Conductance** of cluster C is the size of the cut normalized by the total number of links incident to nodes contained in C,

$$\text{Conductance}(C) = \frac{E_C}{E_C + I_C}.$$

(2) **Expansion** of cluster C is the size of the cut divided by the total number of nodes in C,

$$\text{Expansion}(C) = \frac{E_C}{N_C}.$$

(3) **Cut-ratio** of cluster C is the size of the cut divided by the size of maximal cut,

$$\text{Cut-ratio}(C) = \frac{E_C}{N_C(N - N_C)}.$$

Probably the oldest definition of graph cluster originate from circuit theory which is furtherly adopted in social network analysis. Namely, Luccio and Sami [1969] introduced the notion of LS-set that is also known as Raddichi strong community in social network analysis [Radicchi et al. 2004]. For directed graphs, an LS-set is a subgraph such that the intra-cluster out-degree of each node in the set is higher than its inter-cluster out-degree. The nodes having zero out-degree are not taken into account when determining whether the cluster is Radicchi strong. If the number of intra-cluster links is higher than the number of inter-cluster links then the subgraph is considered as Radicchi weak cluster. Each Radicchi strong cluster is at the same time Radicchi weak cluster, while the converse is not generally true. If a cluster is Radicchi weak or strong then its conductance is smaller than 0.5. The difference between the number of intra- and inter-cluster links inspired ODF (out-degree fraction) family of cluster quality measures [Leskovec et al. 2010]:

(1) **Maximum-ODF** of cluster C is the maximum fraction of inter-cluster links of a node observed in the cluster,

$$\text{Maximum-ODF}(C) = \max_{c \in C} \frac{\mid \{(c, d)\} : d \notin C \mid}{D_{out}(c)},$$

where $D_{out}(c)$ stands for out-degree of node c.

(2) **Average-ODF** of cluster C is the average fraction of inter-cluster links of nodes from C,

$$\text{Average-ODF}(C) = \frac{1}{N_C} \sum_{c \in C} \frac{\mid \{(c,d)\} : d \notin C \mid}{D_{out}(c)}$$

(3) **Flake-ODF** of cluster C is the fraction of nodes in C that have higher intra-cluster out-degree than inter-cluster out-degree,

$$\text{Flake-ODF}(C) = \frac{\mid \{x : x \in C, \mid \{(x,y)\} : y \notin C \mid < D_{out}(c)/2 \mid}{N_C}.$$

In other words Flake-ODF measures how C is close to being Radicchi strong cluster: if Flake-ODF(C) is equal to 1 then C is Radicchi strong.

4. SOFTWARE NETWORKS AND CLUSTERING METRICS

Software networks are graph-based representations of a software system. The architecture of the whole system can be represented by one directed graph that we refer to as a General Dependency Network (GDN) [Savić et al. 2014]. The nodes of GDN represent software entities such as packages/units, classes/modules, methods/functions and class attributes/global variables, while links represent relations between them. We can distinguish between two types of links in GDN: "vertical" (CONTAINS) links that maintain the hierarchy of software entities and "horizontal" links that show dependencies between entities from the same level of abstraction. Two software entities A and B are connected by a CONTAINS link $A \rightarrow B$ if entity A defines or declares entity B. A group of entities that are contained in the same highly cohesive and loosely coupled entity naturally form a cluster of contained entities. Examples of such clusters for object-oriented software systems are: (1) classes and interfaces contained in the same package or workspace, and (2) methods and class attributes contained in the same class. When a software is written in a procedural programming language then procedures (functions) and global variables defined in a module form a cluster.

We can separate horizontal links of GDN into two categories:

—Intra-cluster link connects two entities from the same level of abstraction that are contained in the same software entity, i.e.

$$A \rightarrow B \text{ is an intra-cluster link} \Leftrightarrow (\exists O)\ \text{CONTAINS}(O \rightarrow A) \wedge \text{CONTAINS}(O \rightarrow B).$$

—Inter-cluster link connects two entities from the same level of abstraction that are contained in two different software entities, i.e.

$$A \rightarrow B \text{ is an inter-cluster link} \Leftrightarrow (\exists O_1, O_2)\ O_1 \neq O_2 \wedge \text{CONTAINS}(O_1 \rightarrow A) \wedge \text{CONTAINS}(O_2 \rightarrow B).$$

The separation of links into intra- and inter-cluster links enables us to apply graph clustering evaluation (GCE) metrics to:

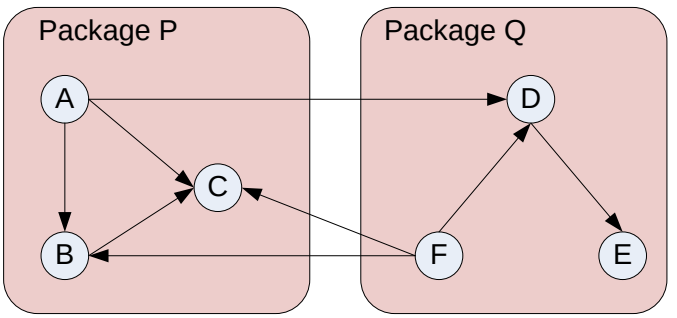
(1) Class collaboration networks in order to evaluate cohesiveness of packages. Class collaboration network is a subgraph of GDN that shows dependencies between classes and interfaces.

(2) Extended static call graphs in order to evaluate cohesiveness of classes in OO systems or modules in procedural software systems. Functions (methods) and global variables (class attributes) constitute the set of nodes in an extended static call graph, while links denote call relationships between functions and uses (access) relationships between functions and global variables.

It can be easily seen from the definition of GCE metrics that only the Flake-ODF metric measures cohesion, while other metrics introduced in the previous section are inverse cohesion measures. In

contrast to cohesion metrics widely used in software engineering (see Section 2), GCE metrics do not ignore references to external entities. On the contrary, they use the number of dependencies to external references to determine to what extent the entity is isolated from the rest of the system. In other words, GCE metric are based on the following principle: an entity can be considered as highly cohesive if its elements are better connected themselves than with the entities defined outside the entity.

Figure 1 shows a class collaboration network that represent a simple software system that consists of two packages P and Q where both packages contain three classes. It can be observed that class F has higher inter-cluster out-degree than intra-cluster out-degree: this class references one class from its package and two classes from package P. Therefore, package Q is not Radicchi strong cluster. This package is neither Radicchi weak cluster since the number of intra-cluster links is not higher than the the number of inter-cluster links. It can be also observed that the system presented in Figure 1 can be refactored in order improve the overall degree of cohesion: if we move class F from package Q to package F then both packages will be Radicchi strong.



	Package P	Package Q
Intra-cluster links	3	2
Inter-cluster links	1	2
Radicchi strong	yes	no
Radicchi weak	yes	no
Conductance	0.25	0.5
Expansion	0.33	0.66
Cut-ratio	0.11	0.22
Maximum-ODF	0.33	0.66
Average-ODF	0.11	0.22
Flake-ODF	1.0	0.66

Fig. 1. Class collaboration network of a simple software system and appropriate cluster quality measures.

5. THEORETICAL ANALYSIS

Briand et al. [1996; 1998] defined several properties that a software metric should satisfy in order to be theoretically sound (lack of) cohesion metric. Those properties are:

(1) **Nonnegativity**. A cohesion (lack of cohesion) metric cannot take a negative value.
(2) **Normalization**. The metric belongs to an interval [0, M], where M is the fixed maximal value.
(3) **Null value**. The cohesion of a software entity is null if R_c is empty, where R_c denotes the set of relationships within the software entity. This means that if there are no intra-cluster links the cohesion of the entity should be zero. On the other side, a metric measuring the lack of cohesion should be zero if R_c is maximal. R_c is maximal if all possible relationships within the entity are present.
(4) **Maximum value**. If R_c is maximal then a metric of cohesion takes the maximal value. If $R_c = \emptyset$ then a metric measuring the lack of cohesion takes the maximal value.
(5) **Monotonicity**. Let e be a software entity. Let e' be the software entity such that $R_e \subseteq R_{e'}$, i.e. we added some relationships (intra-cluster links) in e to obtain e'. Then the following inequalities must hold

$$C(e) \leq C(e'), \tag{1}$$
$$L(e) \geq L(e'), \tag{2}$$

where C and L denote a cohesion and lack of cohesion metric, respectively. In other words, the property states that addition new intra-cluster links must not decrease/increase the value of the cohesion/lack of cohesion metric.

(6) **Merge property**. Let e_1 and e_2 be two unrelated (unconnected) software entities. This means that e_1 does not reference e_2 and vice versa, i.e. there are no relationships (inter-cluster links) between e_1 and e_2. Let e be the software entity which is the union of e_1 and e_2. Then the following inequalities must hold

$$C(e) \leq max\{C(e_1), C(e_2)\}, \tag{3}$$

$$L(e) \geq min\{L(e_1), L(e_2)\}. \tag{4}$$

Namely, this property says that merging two unrelated entities must not increase/decrease the value of the cohesion/lack of cohesion metric.

As a first step in our theoretical analysis of graph clustering evaluation metrics, we state and prove the following lemma that will be frequently used in this section.

LEMMA 1. *Let P and Q be two nonnegative numerical properties of a module. If P and Q are additive under the merge operation then a (lack of) cohesion metric defined as $C = P/Q$ satisfies the merge property.*

PROOF. Let m_1 and m_2 be two modules. Without loss of generality we can assume that $C(m_1) \leq C(m_2)$. Due to the nonnegativity of P and Q the following inequality holds

$$P(m_1)Q(m_2) \leq P(m_2)Q(m_1). \tag{5}$$

Let m denote the module obtained by merging m_1 and m_2. Due to the additivity of P and Q we have that

$$C(m) = \frac{P(m_1) + P(m_2)}{Q(m_1) + Q(m_2)}.$$

C is a cohesion metric. Let us suppose that the merge property is not satisfied, i.e.

$$C(m) > max\{C(m_1), C(m_2)\} = C(m_2).$$

By elementary algebraic transformation we obtain that

$$P(m_1)Q(m_2) > P(m_2)Q(m_1) \tag{6}$$

which is in contradiction with inequality 5.

C is a lack of cohesion metric. Again we give a proof by contradiction. If

$$C(m) < min\{C(m_1), C(m_2)\} = C(m_1)$$

then by elementary algebraic transformation we again obtain inequality 6. □

From the definition of GCE metrics (see Section 3) it can be easily seen that all of them are non-negative. The maximal value of conductance is equal to 1 when $R_c = \emptyset$ and consequently this measure satisfies both the normalization property and the maximum value property. When R_c is maximal conductance is not necessarily equal to zero. Namely, conductance is equal to zero if and only if a module does not depend on other modules. Adding intra-cluster relationship increases only the denominator of conductance and consequently conductance satisfies the monotonicity property. The merge property of conductance is the consequence of Lemma 1 when P is the number of inter-cluster links and Q the sum of the number of inter- and intra-cluster links. Namely, the number of intra-cluster links is an

additive property under the merge operation. Secondly, if two modules are unrelated then they have disjoint sets of inter-cluster links. This means that the number of inter-cluster links is also an additive property for unrelated modules.

In contrast to conductance, expansion does not satisfy the normalization property. If we modify expansion to be a value in the interval [0,1] then we actually obtain the cut-ratio metric. Expansion also does not satisfy the null value property and the maximum value property: both the numerator and denominator in the definition of expansion are independent on the number of intra-cluster links. The expansion of a module remains the same under the addition of intra-cluster links. Therefore, this metric also satisfies the monotonicity property. As already mentioned, the number of inter-cluster links is an additive property of disjoint modules. The number of nodes in a module is also an additive property under the merge operation. Therefore, by Lemma 1 expansion satisfies the merge property.

Cut-ratio satisfies the normalization property: the maximal value of cut-ratio is equal to 1 which is obtained when each entity from the module references all entities defined outside the module. Both the numerator and the denominator of cut-ratio are independent of the number of intra-cluster links and similarly as expansion this measure does not satisfy the null and the maximum value property. If we add a new intra-cluster link the cut-ratio does not change and consequently this metric satisfies monotonicity property. The cut-ratio metric satisfies the merge property which shows the following lemma.

LEMMA 2. *Cut-ratio satisfies the merge property.*

PROOF. Let C_x denote the number of inter-cluster links emanating from nodes contained in module x, N_x the number of nodes in module x, and N the number of nodes in the whole network. Let p and q be two disconnected modules such that the cut-ratio of p is smaller than the cut-ratio of q, i.e.

$$\frac{C_p}{N_p(N-N_p)} \leq \frac{C_q}{N_q(N-N_q)} \quad \Leftrightarrow \quad C_pN_q(N-N_q) \leq C_qN_p(N-N_p). \tag{7}$$

Let r denote the union of p and q. Let us suppose that the merge property is not satisfied, i.e.

$$\frac{C_p + C_q}{(N_p+N_q)(N-N_p-N_q)} < \frac{C_p}{N_p(N-N_p)} \tag{8}$$

If we multiply both sides of inequality 8 by $(N_p + N_q)(N - N_p - N_q)N_p(N - N_p) > 0$, then we obtain

$$\begin{aligned} (C_p + C_q)N_p(N-N_p) &< C_p(N_p+N_q)(N-N_p-N_q) && (9)\\ C_qN_p(N-N_p) &< C_pN_q(N-N_q) - 2C_pN_pN_q && (10)\\ &\leq C_pN_q(N-N_q) && (11) \end{aligned}$$

which is in contradiction with inequality 7. □

From the definition of ODF measures it can be easily seen that they take values in the range [0, 1], which means that they satisfy bot nonnegativity and normalization property. When $R_c = \emptyset$ then Maximum-ODF and Average-ODF are equal to 1, while Flake-ODF is equal to 0, which means that Maximum- and Average-ODF satisfy the maximum value property, while Flake-ODF satisfies the null value property (recall that Flake-ODF measures cohesion, while Maximum- and Average-ODF are lack of cohesion metrics). The numerator of Maximum- and Average-ODF is independent of the number of intra-cluster links. Consequently, those metrics does not satisfy the null value property and satisfy the monotonicity property (addition of intra-cluster links does not change Maximum- and Average-ODF). The merge property is trivially satisfied for Maximum-ODF.

LEMMA 3. *Average-ODF satisfies the merge property.*

PROOF. Let D'_a denote the number of inter-cluster links emanating from node a, D_a out-degree of node a ($D'_a \leq Da$), and N_x the number of nodes in module x. Let p and q be two disconnected modules such that the Average-ODF of p is smaller than the Average-ODF of q, i.e.

$$\frac{1}{N_p}\sum_{u\in p}\frac{D'_u}{D_u} \leq \frac{1}{N_q}\sum_{u\in q}\frac{D'_u}{D_u} \quad \Leftrightarrow \quad N_q\alpha \leq N_p\beta, \quad \text{where} \quad \alpha = \sum_{u\in p}\frac{D'_u}{D_u},\ \beta = \sum_{u\in q}\frac{D'_u}{D_u} \tag{12}$$

Let r denote the union of p and q. The Average-ODF of r is equal to

$$\text{Average-ODF}(r) = \frac{1}{N_p+N_q}\sum_{u\in r}\frac{D'_u}{D_u} = \frac{1}{N_p+N_q}\left(\sum_{u\in p}\frac{D'_u}{D_u} + \sum_{u\in q}\frac{D'_u}{D_u}\right) = \frac{\alpha+\beta}{N_p+N_q} \tag{13}$$

Let us suppose that Average-ODF does not satisfy the merge property, i.e. $(\alpha+\beta)/(N_p+N_q) < \alpha/N_p$. Then we obtain that $\beta N_p < \alpha N_q$ which is in contradiction with inequality 12. □

The addition of new intra-cluster links can only increase the number of entities defined in a module whose intra-cluster out-degree is greater than inter-cluster out-degree. Therefore, Flake-ODF satisfies the monotonicity property. The number of entities in the module whose intra-cluster out-degree is greater than inter-cluster out-degree is additive property under the merge operation. Therefore, Flake-ODF also satisfies merge property by Lemma 1.

Table I. Properties of graph clustering metrics as (lack of) cohesion software metrics.

Metric	Nonnegativity	Normalization	Null value	Maximum value	Monotonicity	Merge
Conductance	yes	yes	no	yes	yes	yes
Expansion	yes	no	no	no	yes	yes
Cut-ratio	yes	yes	no	no	yes	yes
Maximum-ODF	yes	yes	no	yes	yes	yes
Average-ODF	yes	yes	no	yes	yes	yes
Flake-ODF	yes	yes	yes	no	yes	yes

The properties of graph clustering metrics as (lack of) cohesion software metrics are summarized in Table I. This table indicates the limitations of GCE metrics as software metrics. As observed by Briand et al. [1998], only a few widely known software cohesion metric fulfill all of the cohesion properties. In other words, a measure which does not satisfy all of the properties can be considered as poorly defined. Secondly, we can see that GCE metrics reflecting cohesion does not satisfy the null value property, while GCE metrics reflecting lack of cohesion does not satisfy the maximum value property. However, we believe that this is not the disadvantage of GCE metrics. Firstly, it is very unlikely to observe full connected software modules in practice (each class from a package reference each other; each method from a class calls each other and access to each class attribute). Secondly, in such cases GCE metrics favour loosely coupled software modules emphasizing another quality principle of good software design, i.e. the principle of low coupling.

6. CONCLUSION AND FUTURE WORK

In this paper we introduced the idea of applying graph clustering evaluation (GCE) metrics to graphs representing software systems in order to evaluate cohesiveness of software entities. In contrast to standard cohesion metrics, GCE metrics do not ignore external references. They are based on the idea that reducing coupling between an entity and the rest of the system increases cohesion of the elements contained in the entity. Using the theoretical framework introduced by Briand et al. we investigated the properties of graph clustering evaluation metrics. This analysis showed that GCE metrics are theoretically sound with respect to the most important properties of software cohesion

metrics (monotonicity and merge property), but also showed that they possess certain limitations we should be aware of when using GCE metrics as software metrics. Our future work will extend the present work with an empirical investigation of the following research questions:

(1) Do GCE metrics correlate to standard software cohesion metrics (LCOMs, TCC, LCC, etc.) and to what extent?
(2) Each software entity can be described by a numerical vector containing metrics of internal complexity (such as LOC, Halstead measures, cyclomatic complexity, etc.) and metric of design complexity (metrics quantifying importance of the entity such as betweenness centrality and page rank, its coupling to other entities such as degree centrality/CBO, inheritance for classes such as NOC and DIT, and invocation for methods/functions). Each of these vectors can be, according to the degree of cohesion, classified as Radicchi strong (strongly cohesive), Radicchi weak (weakly cohesive) or poorly cohesive (entity that is neither Radicchi strong nor Radicchi weak). Therefore, our second research question will be: are there any differences in internal and design complexity between strongly, weakly and poorly cohesive software entities?
(3) Is it possible to automatically remodularize software system using simple refactorings such as move class/method in order to improve the degree of cohesion of the overall system (to increase the number of Raddichi strong clusters, to minimize the average conductance, etc.).

REFERENCES

James M. Bieman and Byung-Kyoo Kang. 1995. Cohesion and Reuse in an Object-oriented System. In *Proceedings of the 1995 Symposium on Software Reusability (SSR '95)*. ACM, New York, NY, USA, 259–262.

S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D-U. Hwang. 2006. Complex Networks : Structure and Dynamics. *Physics Reports* 424, 4-5 (2006), 175–308.

Lionel C. Briand, John W. Daly, and Jürgen Wüst. 1998. A Unified Framework for Cohesion Measurement in Object-OrientedSystems. *Empirical Software Engineering* 3, 1 (1998), 65–117.

Lionel C. Briand, Sandro Morasca, and Victor R. Basili. 1996. Property-Based Software Engineering Measurement. *IEEE Transactions on Software Engineering* 22, 1 (1996), 68–86.

Aydin Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. 2013. Recent advances in graph partitioning. *CoRR* abs/1311.3144 (2013).

S. R. Chidamber and C. F. Kemerer. 1994. A metrics suite for object oriented design. *IEEE Transactions in Software Engineering* 20, 6 (1994), 476–493.

Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3-5 (2010), 75 – 174.

Martin Hitz and Behzad Montazeri. 1995. Measuring Coupling and Cohesion in Object-Oriented Systems. In *Proc. International Symposium on Applied Corporate Computing*. 25–27.

Y. S. Lee, B. S. Liang, S. F. Wu, and F. J. Wang. 1995. Measuring the coupling and cohesion of an object-oriented program based on information flow. In *Proceedings of International Conference on Software Quality*.

Jure Leskovec, Kevin J. Lang, and Michael Mahoney. 2010. Empirical Comparison of Algorithms for Network Community Detection. In *Proceedings of the 19th International Conference on World Wide Web (WWW '10)*. ACM, New York, NY, USA, 631–640.

F. Luccio and M. Sami. 1969. On the decomposition of networks in minimally interconnected subnetworks. *IEEE Transactions on Circuit Theory* 16, 2 (1969), 184–188.

F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. 2004. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences* 101, 9 (2004), 2658–2663.

Milos Savić, Gordana Rakić, Zoran Budimac, and Mirjana Ivanović. 2014. A language-independent approach to the extraction of dependencies between source code entities. *Information and Software Technology,* 56, 10 (2014), 1268–1288.

Satu Elisa Schaeffer. 2007. Graph Clustering. *Compututer Science Review* 1, 1 (2007), 27–64.

Edward Yourdon and Larry L. Constantine. 1979. *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design* (1st ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.