2016 IEEE 55th Conference on Decision and Control (CDC)
ARIA Resort & Casino
December 12-14, 2016, Las Vegas, USA

# Randomized Algorithms with Adaptive Tuning of Parameters for Detecting Communities in Graphs*

Natalia Amelina, Oleg Granichin, Olga Granichina, Ilia Kirianovskii, and Timofey Prodanov

*Abstract*— **In the last years, the study of complex networks grows rapidly and search of tightly connected groups of nodes, or *community detection*, has proved to be a powerful tool for analyzing the real systems. Randomized algorithms are effective for detecting communities but there is no set of optimal parameters that makes these algorithms create a good partitions into communities for every input complex network. In this paper we consider two randomized algorithms and, based on the stochastic approximation, propose two new adaptive modifications that adjust parameters to the input data and create a good partitions for wider range of input networks.**

## I. INTRODUCTION

This paper is inspired by V. Blondel plenary talk at ECC-2014 [1]. Historically, the network research was a part of graph theory since the resolution of the Seven Bridges of Konigsberg problem by Leonhard Euler [2]. In 1920 the social networks research began [3]. The attention shifted from a small networks to a large networks with billions of nodes, the techniques of networks static analysis have being developed. Basically, the queueing theory, that also considers networks of requests for telephone exchange, used the Poisson distribution to describe streams of requests, for example [4].

The research of *complex networks* was developing over the last 20 years. The complex networks were successfully used in many areas including power systems, education, sociology [5], crime investigation [6], epidemiology [7], bioinformatics [8] and the research of the Internet topology [9], [10].

The *degree* is one of the typical characteristics of network nodes, and it is defined as the number of outgoing edges. During the research process of complex networks, based on the real systems [5]–[10], it revealed that $P(s)$ distribution, defined as the ratio of the number of nodes with degree $s$ to the total number of nodes, significantly differs from the Poisson distribution, that is expected for random graphs of Erdos–Renyi [11]. Besides that, networks based on real systems are characterized by a short paths between any of two nodes, and a large number of small cycles [12]. It shows that models proposed by graph theory and methods of static analysis will not work well for those graphs.

Essentially, the community presence is the common property of the networks considered in [5]–[10]. A *community* is a group of nodes with a large amount of connections inside the group but with small amount of connections with nodes outside the group. The ability to find and analyze such groups of nodes gives us a huge possibilities for research of real systems presented as complex networks [13]. Closely related groups of nodes in social networks present people belonging to communities. Closely related groups of nodes in the Internet correspond to pages devoted to common themes [12]. The communities in networks that describe interaction within genes are related to functional modules [14]. The search of such groups of nodes is called *community detection* in graphs or *clustering*.

In 2010 Ovelgoenne and Geyer-Schulz proposed the Randomized Greedy algorithm for community detection in graphs [15]. Later, in 2012 they described Core Groups Graph Cluster algorithm [16]. Both of these algorithms are very sensitive to the input parameters, which affect the quality of the final partition. Moreover, there is a question, how to choose the optimal parameters to get the best results on various sets of graphs. *In this paper we consider the new adaptive algorithms that solve this problem. Also, our algorithms are suited to the case of drifting optimal parameters.*

The paper is organized as follows. The required information about graphs, complex networks, community detection techniques and algorithms of stochastic approximation are in Section II. In Section III the Adaptive Randomized Greedy algorithm and its properties are presented. Section IV describes the Adaptive Core Groups Graph Cluster algorithm, its possible applications and its iterative modification. The conclusions are in Section V.

## II. PRELIMINARY

Consider a graph $G = (V, E)$, where $V \neq \emptyset$ is a set of *nodes*, $E$ is a set of *edges*, and the cardinalities of $V$ and $E$ are $N$ and $L$ respectively.

Communities (groups or clusters) are the sets of nodes $P = \{C_1, \dots, C_K\}$ such that $\bigcup_{i=1}^{K} C_i = V$ and $\forall i, j \in 1, \dots, K \quad \underset{i \neq j}{C_i \cap C_j = \emptyset}$.

A partition into communities $P = \{C_1, \dots, C_{K_1}\}$ is *based on* the partition $\widetilde{P} = \{\widetilde{C}_1, \dots, \widetilde{C}_{K_2}\}$ iff $\forall i \in 1, \dots, K_1 \quad \exists j \in 1, \dots, K_2$ such that $C_i \subseteq \widetilde{C}_j$.

A graph is said to have community structure if the nodes of the graph can be easily grouped into sets of nodes such

*This work was supported by the Russian Ministry of Education and Science (72/14-GZK, reg.number 1835) and by the RFBR (projects 14-08-01015 and 16-07-00890).

N. Amelina, O. Granichin, O. Granichina, I. Kirianovskii, and T. Prodanov are with the Saint Petersburg State University (Faculty of Mathematics and Mechanics, and Research Laboratory for Analysis and Modeling of Social Processes), 7-9, Universitetskaya Nab., St. Petersburg, 199034, Russia. N. Amelina and O. Granichin are also with the Institute of Problems in Mechanical Engineering, Russian Academy of Sciences. O. Granichin is also with ITMO University. O. Granichina is also with Herzen State Pedagogical University n.amelina@spbu.ru, o.granichin@spbu.ru, olga_granichina@mail.ru, ki.stfu@gmail.com, timofey.prodanov@gmail.com

that each set of nodes is densely connected internally. Two communities are adjacent to each other if there is an edge between them.

*Modularity*

In 2004 Newman and Girvan proposed the *modularity function* $Q$ to measure the strength of division of a network into communities [17].

Assume we have $K$ communities, then the *normalized adjacency matrix* $\mathbf{e}$ is a symmetric matrix of size $K \times K$, where $e_{ij}$ is the ratio of the number of edges from community $i$ to community $j$, to the total number of edges in the graph. The trace of this matrix $\operatorname{tr} \mathbf{e} = \sum_{i \in 1,\dots,K} e_{ii}$ shows the ratio of the number of nodes inside communities, to the total number of nodes in the graph. Thus, a good partition into communities has a high value of the trace of the matrix $\mathbf{e}$.

However the partition which consists of one community has the highest trace value and it says nothing about the community structure of the graph. To solve this problem, consider a vector $\mathbf{a} = (a_1, \dots, a_K)^T$ where $a_i = \sum_{j \in 1,\dots,K} e_{ij}$ is the ratio between the number of edges connected to community $i$ and the total number of edges in the graph, and consider the modularity function $Q$ [17]:

$$Q(G,P) = \sum_{i \in 1,\dots,K} (e_{ii} - a_i^2) = \operatorname{tr} \mathbf{e} - \sum_{i \in 1,\dots,K} a_i^2 \quad . \quad (1)$$

The quality of the partition is described by the value of $Q$: the higher the better (maximum is 1).

So the community detection problem can be formulated as follows: find a partition into communities with the highest modularity $Q$.

This problem was proved to be NP-complete by Brandes et al. [18], nonetheless to determine the change of the modularity after the union of the two communities $i$ and $j$ only one operation is required: $\Delta Q = 2(e_{ij} - a_i a_j)$. The union operation has a time complexity of $O(\min(n_i, n_j))$, where $n_i$ and $n_j$ are the number of adjacent communities to community $i$ and $j$ respectively.

*Randomized Greedy (RG)*

In 2010 Ovelgoenne and Geyer-Schulz proposed the Randomized Greedy algorithm, which effectively maximizes modularity [15]: first, it splits the graph into $K = N$ parts, and then on each iteration it takes $k$ arbitrary communities with their neighbors and unites the pair which gives the highest increase of the modularity (if any). The final result of the algorithm is the partition into communities which has the greatest global modularity.

In this paper, *RG* with parameter $k$ is denoted as $RG_k$.

*Core Groups Graph Cluster (CGGC)*

In 2012 Ovelgoenne and Geyer-Schulz won the 10th DIMACS Implementation Challenge in Category Graph Partitioning and Graph Clustering by presenting the Core Groups Graph Cluster scheme [16]. The main idea is to create a good initial partition using $s$ initial algorithms and then split the rest of nodes by using the final algorithm: see Algorithm 1.

**Algorithm 1** Core Groups Graph Cluster

```
Input: $G = (V, E)$, set of initial algorithms, final algorithm;
Output: clustering of $G$;
1: $S = \emptyset$;
2: for each initial algorithm do
3:     create a partition into communities $S_i$ of the graph $G$;
4: end for
5: create an intermediate partition $\widetilde{P}$ based on partitions $S$;
6: create a partition by using the final algorithm for $\widetilde{P}$;
```

Core Groups Graph Cluster scheme has an iterative version which repeats lines 1–5 several times until it gives an advantage and uses an intermediate partition $\widetilde{P}$ on the line 3.

The above described algorithms depend on various set of parameters: $RG_k$ depends on $k$, $CGGC$ depends on $s$ and the selected initial and final algorithms.

Unfortunately, there is no set of optimal parameters that make these algorithms create good partitions for every input graph, so we have to pick up the optimal arguments for each graph separately.

*Simultaneous Perturbation Stochastic Approximation (SPSA)*

Stochastic approximation was introduced in 1951 by Robbins and Monro [19] and was further developed for optimization problems by Kiefer and Wolfowitz [20]. In 1954 Blum extended the stochastic approximation algorithm to the multidimensional case [21]. In case of $m$-dimensional space, the conventional KW-procedure which is based on finite-difference approximations of the function gradient vector uses $2m$ observations on each iteration (two observations for approximations of each component of the gradient $m$-vector). In 80s-90s, the simultaneously perturbation stochastic approximation (randomized version of stochastic approximation with one (or two) measurements per iteration) was proposed by Granichin in 1989 [22], Polyak and Tsybakov in 1990 [23], and Spall in 1992 [24].

Stochastic approximation algorithms have shown to be effective in minimization of stationary functional problem solving. In [25]–[27] similar algorithms with constant step-size were applied to time varying functionals.

The simultaneous perturbation stochastic approximation algorithm with constant step size is described in Algorithm 2.

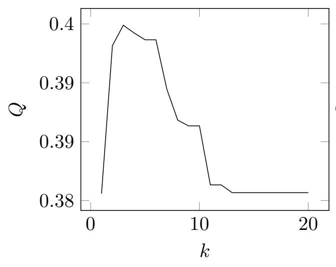
The *SPSA* is well suited for creation of adaptive modifications of algorithms which will adapt to the input parameters. In this paper we apply a discrete version of *SPSA* algorithm with constant step size to *RG* and *CGGC* for choosing a set of optimal parameters automatically.

For testing of proposed algorithms, we use the test graphs from the 10th DIMACS Implementation Challenge, which are located here: `http://www.cc.gatech.edu/dimacs10/archive/clustering.shtml`. The quality of the final partition is scored by the value of the modularity function $Q$ (see (1)).

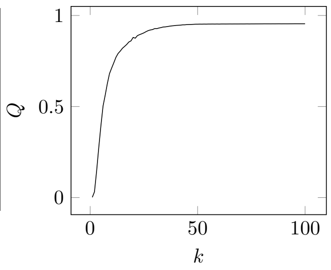
Also, for evaluation of quality of the algorithm we generated a graph *auto40* with $N = 40000$ which consists of

**Algorithm 2** SPSA with constant step size

```
Input: function $f$, initial approximation $\hat{\theta}_0 \in \mathbb{R}^m$, perturbation $d \in \mathbb{R} \setminus \{0\}$, step size $\alpha \in \mathbb{R}^m$, and $\varepsilon > 0$;
Output: $\hat{\theta}_n$;
1: $n = 0$;
2: repeat
3:     $n = n + 1$;
4:     choose $\Delta_n \in \mathbb{R}^m$ such that $\Delta_{n_i} = \pm 1$ and $\Delta_{n_i} \sim \mathrm{B}\left(1, \frac{1}{2}\right)$;
5:     $\theta_n^- = \hat{\theta}_{n-1} - d\Delta_n$ and $\theta_n^+ = \hat{\theta}_{n-1} + d\Delta_n$;
6:     $y_n^- = f(\theta_n^-)$ and $y_n^+ = f(\theta_n^+)$;
7:     $\hat{\theta}_n = \hat{\theta}_{n-1} - \alpha\Delta_n \frac{y_n^+ - y_n^-}{2d}$;
8: until $|\hat{\theta}_n - \hat{\theta}_{n-1}| < \varepsilon$
```



(a) *karate*



(b) *netscience*

Fig. 1: The dependence of modularity $Q$ and $k$ for different graphs. Note that $k \in \mathbb{N}$, but for clarity the graphs are continuous.

$K = 40$ communities and with probability $p_{same} = 0.1$ and $p_{diff} = 10^{-4}$ that there is an edge between any two nodes inside one community, and any two nodes from different communities.

## III. ADAPTIVE RANDOMIZED GREEDY (ARG)

Depending on the input graph, average modularity of the final partition of $RG_k$ is the highest on small $k$ (Fig. 1a), or grows with increasing $k$ (Fig. 1b).

The $RG_k$'s iteratrion has a time complexity of $O(k)$, so we take the following quality function:

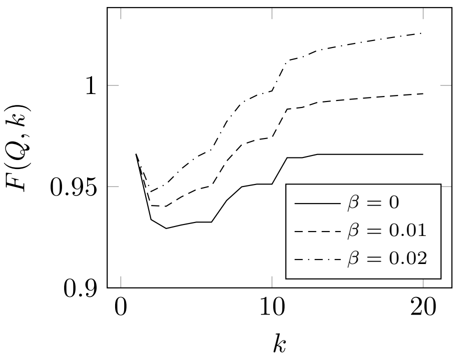
$$f(Q, k) = -\ln Q + \beta \ln k \quad , \qquad (2)$$

in which $\beta \geq 0$ can be considered as $\beta = \frac{\ln \gamma}{\ln 2}$, where $\gamma$ it the necessary increase of $Q$, if $k$ will increase by 2 times.

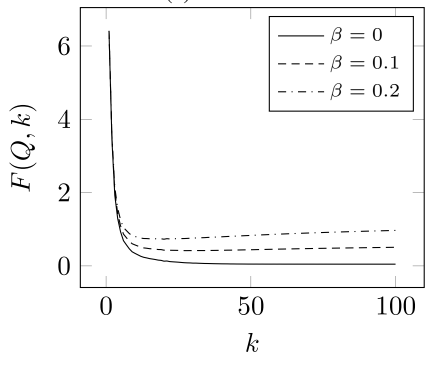
The usage of *SPSA* is substantiated for a convex quality function. Based on the function values for $k \in \mathbb{N}$, the quality function can be extended to a continuous function, using linear functions between natural numbers. On the basis of the experiments (see Fig.2), it can be said that the quality function $f$ is similar to convex.

As you can see in Fig. 2b, sometimes to have the highest value of the quality function we should use a large $\beta$. However, there is a small $\beta$ which gives high results.

For using *SPSA*, let us split the *RG* algorithm into steps of length $\sigma$ iterations. During each step we use the fixed $k$



(a) *karate*



(b) *netscience*

Fig. 2: The quality function with $\beta = 0$, $\beta = 0.1$, and $\beta = 0.2$ for different graphs

and every 2 steps we choose two new values of $k$, based on the best value of $k$ from the previous iteration.

Also, the quality function uses the average of the growth of modularity over the last $\sigma$ steps. In this way, the algorithm is Algorithm 3.

In despite of *RG*, *ARG* has five independent parameters. However, according to the test results on graphs *cond-mat-2003*, *caidaRouterLevel* and *cnr-2000*, the parameters $\alpha$ and $\sigma$ almost have no effect on the final partition, unlike $d$ and $\hat{k}_0$ which strongly affect the results. Nonetheless, there is the set of parameters that provides high results: $\alpha = 10$, $\sigma = 1000$, $d = 5$, $\hat{k}_0 = 8$.

Often, the use of a small $\beta$ gives a better division than $\beta = 0$ (see Fig. 3).

### *Comparison of RG and ARG*

In Tables I and II, $RG_k$ and $ARG$ are compared by the average modularity and by the running time for different $k$:

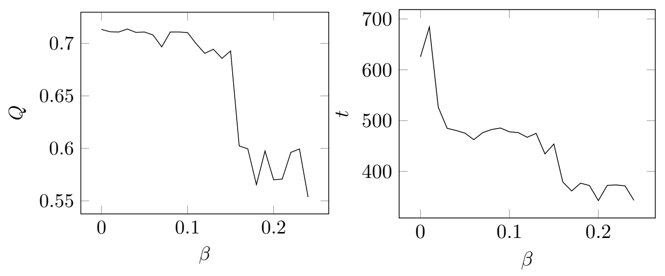
- $k = 1$ is a minimal value for $k$
- $k = 3$ is a value, which often gives high results
- $k = 10$ is a value, which gives stable results in the sense of better comparative values on average
- $k = 50$ is an example for large $k$

The $ARG$ was run with the following parameters: $\alpha = 10$, $\sigma = 1000$, $d = 5$, $\hat{k}_0 = 8$, $\beta = 0.05$.

In most cases, some $RG_k$ algorithms give a better result than $ARG$, but on the other hand, $ARG$ has stable results like $RG_{10}$ and has a higher average modularity than $RG_{10}$.

**Algorithm 3** Adaptive Randomized Greedy

```
Input: $G = (V, E)$, initial approximation $\hat{k}_0 \in \mathbb{N}$, perturbation $d \in \mathbb{N}$, step size $\alpha \geq 0$, significance of the running time $\beta \geq 0$, number of iterations per step $\sigma \in \mathbb{N}$;
Output: clustering of $G$;
1: $n = 0$, split the graph $G$ into $K = N$ communities;
2: repeat
3:     $n = n + 1$;
4:     $k_n^- = \max\{\hat{k}_{n-1} - d, 1\}$ and $k_n^+ = \hat{k}_{n-1} + d$;
5:     calculate the average of the growth of modularity $Q_n^-$ over the next $\sigma$ iterations: take $k_n^-$ arbitrary communities with their neighbors and unite the pair which gives the highest increase of the modularity, repeat again;
6:     calculate the average of the growth of modularity $Q_n^+$ over the next $\sigma$ iterations for $k_n^+$;
7:     $y_n^- = -\ln Q_n^- + \beta \ln k_n^-$ and $y_n^+ = -\ln Q_n^+ + \beta \ln k_n^+$;
8:     $\hat{k}_n = \max\left\{1, \left\lfloor \hat{k}_{n-1} - \alpha \frac{y_n^+ - y_n^-}{k_n^+ - k_n^-} \right\rfloor\right\}$;
9: until there are no communities that can be united
10: return the partition into communities which has the greatest value of modularity;
```



(a) The relationship between $Q$ and $\beta$ (b) The relationship between $t$ and $\beta$

Fig. 3: The dependence of modularity $Q$ and the running time $t$ of $ARG$ and coefficient $\beta$ for graph *cond-mat-2003*

### *Usage of ARG as the initial algorithm for CGGC*

The running time of $CGGC$ equals the sum of the running time for each initial algorithm plus the running time of the final algorithm. Therefore, on the one hand the initial algorithms should work fast but on the other hand, there is no sense in algorithms which give bad partitions.

Thus, $ARG$ is well suited as the initial algorithm for $CGGC$ for graphs with $N >> 2\sigma$.

According to Tables III and IV, $RG_k$ with small $k$ is not suitable as the initial algorithm (for example, see $RG_3$ for *cond-mat-2003*), and is also not suitable as the final algorithm for $CGGC$.

In addition, we do not know in advance the best $k$ for the specific graph and we can not estimate the quality of partition made by $RG_k$. $CGGC$ takes a long running time, so it is unprofitable to run $CGGC$ few times to find the best $k$. So it makes sense to use $ARG$ as the initial algorithm, and as the final algorithm if it is possible.

TABLE I: Average modularity for $RG_k$ and $ARG$ on different graphs

| | $RG_1$ | $RG_3$ | $RG_{10}$ | $RG_{50}$ | $ARG$ |
|---|---|---|---|---|---|
| as-22july06 | 0.65281 | 0.64658 | 0.64024 | 0.63479 | 0.64264 |
| cond-mat-2003 | 0.00012 | 0.19727 | 0.70738 | 0.69403 | 0.71193 |
| auto40 | 0.78944 | 0.79988 | 0.80417 | 0.80273 | 0.80174 |
| caidaRouterLevel | 0.01938 | 0.81101 | 0.79883 | 0.79300 | 0.80216 |
| cnr-2000 | 0.90237 | 0.91192 | 0.91144 | 0.90997 | 0.91039 |
| eu-2005 | 0.92765 | 0.92559 | 0.91780 | 0.90416 | 0.91048 |
| in-2004 | 0.00026 | 0.97836 | 0.97185 | 0.97596 | 0.97616 |

TABLE II: The running time of $RG_k$ and $ARG$ on different graphs, milliseconds

| | $RG_1$ | $RG_3$ | $RG_{10}$ | $RG_{50}$ | $ARG$ |
|---|---|---|---|---|---|
| as-22july06 | 177 | 189 | 231 | 464 | 238 |
| cond-mat-2003 | 58 | 184 | 463 | 931 | 474 |
| auto40 | 4,652 | 4,591 | 6,017 | 12,558 | 6,479 |
| caidaRouterLevel | 852 | 9,114 | 10,244 | 15,217 | 11,514 |
| cnr-2000 | 26,083 | 26,056 | 27,137 | 33,592 | 29,054 |
| eu-2005 | 202,188 | 200,686 | 207,689 | 246,170 | 225,748 |
| in-2004 | 9,208 | 487,953 | 553,196 | 607,408 | 617,345 |

## IV. ADAPTIVE CORE GROUPS GRAPH CLUSTERING (ACGGC)

To create an adaptive algorithm that will be able to work on graphs of any size, we can use $SPSA$ for choosing the initial algorithms. Also, we can build the intermediate partition based on the few best partitions. The algorithm is Algorithm 4.

The described *ACGGC* algorithm has many parameters, however the following set of them gives high results for all considered graphs (see Tables V and VI): $d = 2$, $\alpha = 1000$, $l = 6$, $\hat{k}_0 = 5$, $k_{max} = 50$, $r = 0.05$.

Decreasing $k_{max}$ reduces the computing time, and in addition, it often increases the modularity.

### *Comparison of CGGC and ACGGC*

Table VII contains test results for 5 modifications of *CGGC* and *ACGGC*, that are compared by the average modularity:

- $ACGGC^I$ is $ACGGC$ with $d = 2$, $\alpha = 1000$, $l = 6$, $\hat{k}_0 = 5$, $k_{max} = 50$, $r = 0.05$, and the final algorithm is $RG_{10}$

TABLE III: Modularity of the partitions, which were constructed by *CGGC* with the initial algorithm $A_{init}$ and the final algorithm $A_{final}$

| $A_{init}$ | $RG_3$ | | $RG_{10}$ | | $ARG$ | |
|---|---|---|---|---|---|---|
| $A_{final}$ | $RG_3$ | $RG_{10}$ | $RG_3$ | $RG_{10}$ | $RG_3$ | $RG_{10}$ |
| cond-mat-2003 | 0.16840 | 0.71155 | 0.44934 | 0.74794 | 0.42708 | 0.74872 |
| auto40 | 0.80628 | 0.80645 | 0.80633 | 0.80645 | 0.80628 | 0.80647 |
| caidaRouterLevel | 0.84078 | 0.85372 | 0.84031 | 0.84448 | 0.83671 | 0.85279 |

TABLE IV: The running time of $CGGC$ with the initial algorithm $A_{init}$ and the final algorithm $A_{final}$ on different graphs, milliseconds

| $A_{init}$ | $RG_3$ | | $RG_{10}$ | | $ARG$ | |
|---|---|---|---|---|---|---|
| $A_{final}$ | $RG_3$ | $RG_{10}$ | $RG_3$ | $RG_{10}$ | $RG_3$ | $RG_{10}$ |
| cond-mat-2003 | 2.0 | 2.3 | 4.7 | 4.8 | 4.8 | 4.9 |
| auto40 | 47.0 | 46.6 | 57.7 | 57.2 | 65.7 | 65.8 |
| caidaRouterLevel | 94.0 | 93.7 | 104.2 | 104.3 | 115.1 | 118.5 |

**Algorithm 4** Adaptive Core Groups Graph Clustering

```
Input: $G = (V, E)$, initial approximation $\hat{k}_0 \in \mathbb{N}$, perturbation $d \in \mathbb{N}$, step size $\alpha \geq 0$, number of steps $l \in \mathbb{N}$, upper bound of $k_{max} \in \mathbb{N}_{\geq 2}$, and $r \in (0, 1]$;
Output: clustering of $G$;
 1: $n = 0$, $S = \emptyset$;
 2: repeat
 3:     $n = n + 1$;
 4:     $k_n^- = \max\{1, \hat{k}_{n-1} - d\}$, $k_n^+ = \min\{k_{max}, \hat{k}_{n-1} + d\}$;
 5:     create a partition $P_n^-$ by using $RG_{k_n^-}$ and add it into $S$, the modularity is $Q_n^-$;
 6:     create a partition $P_n^+$ by using $RG_{k_n^+}$ and add it into $S$, the modularity is $Q_n^+$;
 7:     $y_n^- = -\ln Q_n^-$ and $y_n^+ = -\ln Q_n^+$ and update $Q_{best}$;
 8:     $\hat{k}_n = \max\left\{1,\ \min\left\{k_{max},\ \left\lfloor \hat{k}_{n-1} - \alpha \frac{y_n^+ - y_n^-}{k_n^+ - k_n^-} \right\rfloor\right\}\right\}$
 9: until $n = l$
10: $\widetilde{S} = \{S_i \in S$ such that $Q(G, S_i) \geq (1 - r)Q_{best}\}$;
11: create an intermediate partition $\widetilde{P}$ based on partitions $\widetilde{S}$;
12: apply the final algorithm to the intermediate partition $\widetilde{P}$;
```

TABLE V: Modularity of the partitions which were constructed by *ACGGC* with different $k_{max}$ on different graphs

| $k_{max}$ | $+\infty$ | 50 | 20 | 10 | 6 |
|---|---|---|---|---|---|
| polbooks | 0.527237 | 0.527237 | 0.527237 | 0.527082 | 0.526985 |
| adjnoun | 0.299720 | 0.299690 | 0.299859 | 0.300141 | 0.299676 |
| football | 0.603324 | 0.603324 | 0.604184 | 0.604266 | 0.604266 |
| jazz | 0.444739 | 0.444739 | 0.444739 | 0.444739 | 0.444739 |
| celegans | 0.439770 | 0.439368 | 0.439750 | 0.439460 | 0.439431 |
| email | 0.573470 | 0.573416 | 0.573652 | 0.573756 | 0.573513 |
| netscience | 0.953033 | 0.908130 | 0.842085 | 0.793289 | 0.768572 |
| cond-mat-2003 | 0.737611 | 0.743572 | 0.749595 | 0.749894 | 0.739200 |

- $ACGGC^{II}$ is $ACGGC$ with $d = 2$, $\alpha = 1000$, $l = 8$, $\hat{k}_0 = 5$, $k_{max} = 20$, $r = 0.05$, and the final algorithm is $RG_{10}$
- $CGGC_{10}^{10}$ is $CGGC$ with $RG_{10}$ as the initial algorithm, $RG_{10}$ as the final algorithm, and with $s = 16$
- $CGGC_3^{10}$ is $CGGC$ with $RG_3$ as the initial algorithm, $RG_{10}$ as the final algorithm, and with $s = 16$
- $CGGC_{10}^3$ is $CGGC$ with $RG_{10}$ as the initial algorithm, $RG_3$ as the final algorithm, and with $s = 16$

where the best result, and the second best result.

Table VII shows that *ACGGC* usually works better than *CGGC*. In [16] was shown that *CGGC* with a large $s$ provides a high value of modularity, therefore in our tests we use $s = 16$.

TABLE VI: The running time of *ACGGC* with different $k_{max}$ on different graphs

| $k_{max}$ | $+\infty$ | 50 | 20 | 10 | 6 |
|---|---|---|---|---|---|
| polbooks | 5.029 | 4.976 | 4.615 | 4.207 | 4.087 |
| adjnoun | 6.115 | 6.099 | 5.481 | 4.952 | 4.744 |
| football | 7.179 | 7.155 | 6.377 | 5.820 | 5.584 |
| jazz | 23.66 | 23.25 | 20.92 | 19.12 | 18.59 |
| celegans | 23.85 | 23.49 | 22.48 | 20.92 | 20.01 |
| email | 70.06 | 72.89 | 68.34 | 63.85 | 62.97 |
| netscience | 477.97 | 85.40 | 46.08 | 38.26 | 30.89 |
| cond-mat-2003 | 41,950 | 9,596 | 6,075 | 5,092 | 4,166 |

TABLE VII: Modularity of the partitions which were constructed by *ACGGC* and *CGGC* on different graphs, where [1]pgpGiantCompo, [2]as-22july06, [3]cond-mat-2003, [4]caidaRouterLevel

| | $ACGGC^I$ | $ACGGC^{II}$ | $CGGC_{10}^{10}$ | $CGGC_3^{10}$ | $CGGC_{10}^3$ |
|---|---|---|---|---|---|
| karate | 0.417242 | 0.417406 | 0.415598 | 0.396532 | 0.405243 |
| dolphins | 0.524109 | 0.523338 | 0.521399 | 0.523338 | 0.522428 |
| chesapeake | 0.262439 | 0.262439 | 0.262439 | 0.262439 | 0.262370 |
| adjnoun | 0.299704 | 0.299197 | 0.295015 | 0.292703 | 0.290638 |
| polbooks | 0.527237 | 0.527237 | 0.527237 | 0.526938 | 0.526784 |
| football | 0.603324 | 0.604266 | 0.604266 | 0.599537 | 0.599026 |
| celegans | 0.439604 | 0.438584 | 0.435819 | 0.436066 | 0.432261 |
| jazz | 0.444739 | 0.444848 | 0.444871 | 0.444206 | 0.444206 |
| netscience | 0.907229 | 0.835267 | 0.724015 | 0.708812 | 0.331957 |
| email | 0.573333 | 0.573409 | 0.571018 | 0.572667 | 0.567423 |
| polblogs | 0.424107 | 0.423208 | 0.422901 | 0.421361 | 0.390395 |
| pgpGiant[1] | 0.883115 | 0.883085 | 0.882237 | 0.882532 | 0.880340 |
| as-22jul[2] | 0.671249 | 0.670677 | 0.666766 | 0.669847 | 0.665260 |
| cond-mat[3] | 0.744533 | 0.750367 | 0.751109 | 0.708775 | 0.413719 |
| caidaRou[4] | 0.846312 | 0.855651 | 0.851622 | 0.858955 | 0.843835 |
| cnr-2000 | 0.912762 | 0.912783 | 0.912500 | 0.912777 | 0.912496 |
| eu-2005 | 0.938292 | 0.936984 | 0.935510 | 0.936515 | 0.936420 |
| in-2004 | 0.979844 | 0.979771 | 0.979883 | | |

TABLE VIII: Modularity of the partitions and the running time for *ACGGC* and *ACGGCi* on different graphs

| | $ACGGC$ | | $ACGGCi$ | |
|---|---|---|---|---|
| | $Q$ | $t$ | $Q$ | $t$ |
| jazz | 0.444739 | 23.68 | 0.444871 | 31.51 |
| celegans | 0.439724 | 23.92 | 0.446973 | 77.25 |
| netscience | 0.907922 | 86.38 | 0.909400 | 96.55 |
| as-22july06 | 0.671205 | 2,329 | 0.674992 | 5,801 |
| cond-mat-2003 | 0.743594 | 9,371 | 0.746731 | 11,654 |

*Iterative ACGGC (ACGGCi)*

The *ACGGCi* works the same way as *CGGCi*: we use initial algorithms few times until it gives us a growth of modularity. This approach does not greatly increase the running time because on every iteration we process fewer nodes than the last iteration.

Table VIII shows that modularity of $ACGGCi$ a bit greater than the modularity of $CGGCi$, however it takes a longer time.

The $CGGCi$ can be used as the final algorithm for $ACGGCi$, and vice versa. This approach is reasonable because one of these algorithms can be more effective in clustering of an intermediate partition $\widetilde{P}$ than another algorithm. We call this approach *combined clustering scheme*.

The test results of combined clustering scheme are shown in Table IX and Fig. 4. The $\rightarrow$ character means that the right hand side is used as the final algorithm for the left hand side.

The combined clustering scheme with $CGGCi$ and $ACGGCi$ on the first phase gives better results than $CGGCi$ and $ACGGCi$.

TABLE IX: Modularity of the partitions and the running time for $ACGGCi$ and $CGGCi$ with different final algorithms

| | | | $Q$ | $t$ |
|---|---|---|---|---|
| $ACGGCi$ | $\rightarrow RG_{10}$ | | 0.446973 | 77.25 |
| $CGGCi$ | $\rightarrow RG_{10}$ | | 0.445008 | 55.29 |
| $ACGGCi$ | $\rightarrow CGGCi$ | $\rightarrow RG_{10}$ | 0.447324 | 89.96 |
| $CGGCi$ | $\rightarrow ACGGCi$ | $\rightarrow RG_{10}$ | 0.445660 | 112.49 |

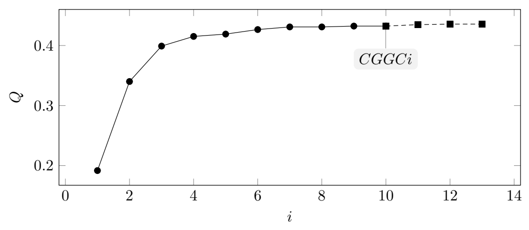


Fig. 4: Modularity of the intermediate partitions of $ACGGCi \to CGGCi \to RG_{10}$ on *celegans*, where $i$ is iteration number, circles are the intermediate partitions, squares are $CGGCi$

TABLE X: Modularity of the partitions for different iterative algorithms

| | $ACGGCi^{I}$ | $ACGGCi^{II}$ | $CGGCi$ | combined |
|---|---|---|---|---|
| karate | 0.417242 | 0.417406 | 0.417242 | 0.417242 |
| dolphins | 0.525869 | 0.525869 | 0.525869 | 0.525869 |
| chesapeake | 0.262439 | 0.262439 | 0.262439 | 0.262439 |
| adjnoun | 0.303731 | 0.303504 | 0.303571 | 0.303970 |
| polbooks | 0.527237 | 0.527237 | 0.527237 | 0.527237 |
| football | 0.604266 | 0.604407 | 0.604429 | 0.604407 |
| celegans | 0.446964 | 0.446836 | 0.445442 | 0.447234 |
| jazz | 0.444871 | 0.444871 | 0.444871 | 0.444871 |
| netscience | 0.908845 | 0.888422 | 0.725781 | 0.907443 |
| email | 0.576778 | 0.577000 | 0.576749 | 0.577110 |
| polblogs | 0.424025 | 0.422920 | 0.423281 | 0.423996 |

### *Comparison of CGGCi and ACGGCi*

Table X contains the results of testing 4 algorithms that are compared by the average modularity, where $ACGGCi^{I}$, $ACGGCi^{II}$ and *CGGCi* are the iterative versions of $ACGGC^{I}$, $ACGGC^{II}$ and $CGGC_{10}^{10}$, and the *combined* is $ACGGCi \to CGGCi \to RG_{10}$.

## V. CONCLUSION

In this paper, we described a new adaptive modifications of randomized algorithms for community detection in graphs.

In Section III we proposed Adaptive Randomized Greedy algorithm which has more stable results and works better than non-adaptive Randomized Greedy (see Tables I and III).

In Section IV we considered Adaptive Core Groups Graph Clustering scheme, which has more qualitative and stable results than its non-adaptive version (see Table VII). Also we described an iterative version of $ACGGC$ that works better than $ACGGC$ and $CGGCi$ (see Tables VIII and X).

The results of testing of algorithms with parameter adaptation revealed a greater efficiency in comparison with their non-adaptive versions in the sense they work equally well for different classes of problems, while the previous versions require manual setting of parameters for each specific task.

We think the results can be used for the development of a transport network. Further, we plan to apply our algorithms to analyze the traffic in a city with an extensive network of roads. The described techniques are also suited for assessing the quality of educational programs.

## REFERENCES

[1] *Blondel, V.* Communities and Dynamics in Social Networks and Mobile Phone Networks. Plenary lecture at 13th European Control Conference (ECC), 26 June, 2014, Strasbourg, France.

[2] *Euler, L.* Solutio problematis ad geometriam situs pertinentis // Commentarii academiae scientiarum Petropolitanae. 1741. Vol. **8**. P. 128–140.

[3] *Weber, M.* Economy and Society: An Outline of Interpretive Sociology. University of California Press. 1978. Vol. **1**.

[4] *Erlang, A.K.* Solution of some problems in the theory of probabilities of significance in automatic telephone exchanges // Elektrotkeknikeren. 1917. Vol. **13**. P. 5–13.

[5] *Scott, J.* Social Network Analysis. Sage. 2012.

[6] *Hong, W., Zhao-wen, W., Jian-bo, L., Wei, Q.* Criminal behavior analysis based on Complex Networks theory // ITIME'09. IEEE International Symposium on IT in Medicine & Education. 2009. P. 951–955.

[7] *Moore C., Newman M.E.J.* Epidemics and percolation in small-world networks // Physical Review E. 2000. Vol. **61**. No. 5. P. 5678–5682.

[8] *Zhao, J., Yu, H., Luo, J., Cao, Z.W., Li, Y.* Complex networks theory for analyzing metabolic networks // Chinese Science Bulletin. 2006. Vol. **51**. No. 13. P. 1529–1537.

[9] *Faloutsos, M., Faloutsos, P., Faloutsos, C.* On power-law relationships of the internet topology // ACM SIGCOMM Computer Communication Review. 1999. Vol. **29**. No. 4. P. 251–262.

[10] *Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.* Graph structure in the web // Computer networks. 2000. Vol. **33**. No. 1. P. 309–320.

[11] *Erdős, P., Rényi, A.* On random graphs. I // Publicationes Mathematicae. 1959. Vol. **6**. P. 290–297.

[12] *Boccaletti, S., Latora, V., Moreno, Y., Chavez, M, Hwang, D.-U.* Complex networks: Structure and dynamics // Physics reports. 2006. Vol. **424**. No. 4. P. 175–308.

[13] *Kirianovskii, I., Granichin, O., Proskurnikov, A.* A new randomized algorithm for community detection in large networks // IFAC-PapersOnLine. 2016. Vol. **49**. No. 13. P. 31–35

[14] *Dejori, M., Schwaighofer, A., Tresp, V., Stetter, M.* Mining functional modules in genetic networks with decomposable graphical models // Omics: a journal of integrative biology. 2004. Vol. **8**. No. 2. P. 176–188.

[15] *Ovelgönne, M., Geyer-Schulz, A.* Cluster cores and modularity maximization // 2010 IEEE International Conference on Data Mining Workshops (ICDMW). 2010. P. 1204–1213.

[16] *Ovelgönne, M., Geyer-Schulz, A.* An ensemble learning strategy for graph clustering // Graph Partitioning and Graph Clustering. 2012. Vol. **588**. P. 187–206.

[17] *Newman M.E.J., Girvan G.* Finding and evaluating community structure in networks // Physical Review E. 2004. Vol. **69**. No. 2. P. 026113.

[18] *Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., Wagner, D.* On modularity clustering // IEEE Transactions on Knowledge and Data Engineering. 2008. Vol. **20**. No. 2. P. 172–188.

[19] *Robbins, H., Monro, S.* A stochastic approximation method // The annals of mathematical statistics. 1951. P. 400–407.

[20] *Kiefer, J., Wolfowitz, J.* Stochastic estimation of the maximum of a regression function // The Annals of Mathematical Statistics. 1952. Vol. **23**. No. 3. P. 462–466.

[21] *Blum, J.R.* Multidimensional stochastic approximation methods // The annals of mathematical statistics. 1954. P. 737–744.

[22] *Granichin, O.* A stochastic recursive procedure with correlated noise in the observation, that employs trial perturbations at the input // Vestnik Leningrad University: Math. 1989. Vol. **22**. No. 1. P. 27–31.

[23] *Polyak, B. and Tsybakov, A.* Optimal order of accuracy of search algorithms in stochastic optimization // Problemy Peredachi Informatsii. 1990. Vol. **26**. No. 2. P. 45–53.

[24] *Spall, J.C.* Multivariate stochastic approximation using a simultaneous perturbation gradient approximation // IEEE Transactions on Automatic Control. 1992. Vol. **37**. No. 3. P. 332–341.

[25] *Kushner, H.J., Yin, G.* Stochastic Approximation and Recursive Algorithms and Applications. Springer Science & Business Media. 2003. Vol. **35**.

[26] *Borkar, V.S.* Stochastic Approximation: a Dynamical Systems Viewpoint. Cambridge University Press Cambridge. 2008.

[27] *Granichin, O., Amelina, N.* Simultaneous perturbation stochastic approximation for tracking under unknown but bounded disturbances // IEEE Transactions on Automatic Control. 2015. Vol. **60**. No. 6. P. 1653–1658