Partitioning Networks into Cliques: A Randomized Heuristic Approach

David Chalupa[*]
Institute of Applied Informatics
Faculty of Informatics and Information Technologies
Slovak University of Technology in Bratislava
Ilkovičova 2, 842 16 Bratislava, Slovakia
david.chalupa@stuba.sk

Abstract

In the context of community detection in social networks, the term community can be grounded in the strict way that simply everybody should know each other within the community. We consider the corresponding community detection problem. We search for a partitioning of a network into the minimum number of non-overlapping cliques, such that the cliques cover all vertices. This problem is called the clique covering problem (CCP) and is one of the classical NP-hard problems. For CCP, we propose a randomized heuristic approach. To construct a high-quality solution to CCP, we present an iterated greedy (IG) algorithm. IG can also be combined with a heuristic used to determine how far the algorithm is from the optimum in the worst case. Randomized local search (RLS) for maximum independent set was proposed to find such a bound. The experimental results of IG and the bounds obtained by RLS indicate that IG is a very suitable technique for solving CCP in real-world graphs. In addition, we summarize our basic rigorous results, which were developed for analysis of IG and understanding of its behavior on several relevant graph classes.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Heuristic methods; F.2.2 [**Nonnumerical Algorithms and Problems**]: Computations on discrete structures; G.2.3 [**Discrete Mathematics**]: Applications; I.5.3 [**Pattern Recognition**]: Clustering

[*]Recommended by thesis supervisor: Prof. Jiří Pospíchal Defended at Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava on June 25, 2014.

Chalupa, D. Partitioning Networks into Cliques: A Randomized Heuristic Approach. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 6, No. 3 (2014) 1-8

Keywords

clique covering problem, heuristic algorithms, complex networks, iterated greedy, community detection

1. Introduction

Many significant real-world problems can be modeled using networks. For the purpose of this work, a network will be formalized as a structure consisting of vertices and undirected edges between pairs of vertices. Thus, a network will simply be an undirected, unweighted graph $G = [V, E]$.

The problem we study in this work is the (vertex) *clique covering problem* (CCP). In CCP, the aim is to divide the vertices of a graph into the minimum number of subsets, in which every pair of vertices has an edge between them. A subgraph with this property is called *clique*. For example, in the context of social networks, the aim of CCP is to partition the network into minimum number of groups, such that everybody knows each other within each group.

One can intuitively see the relation between CCP and *community detection*. An interesting fact is that community detection is a relatively loosely defined problem [24] and the properties of communities found by an algorithm strongly depend on the optimized objective [19]. On the other hand, CCP has a strict definition. The optimal number of cliques needed to solve CCP is a specific value for a network, while there might be more reasonable values for the number of communities. In community detection, this is a rather rare but desirable property of CCP. However, it is *NP-hard* to solve CCP for an arbitrary graph [16].

Our work is closely related to two major fields within computer science. The first field is *artificial intelligence*, since we use heuristic algorithms (sometimes also referred to as heuristics) to solve our problem. Such algorithms provide a solid solution to a problem but are not designed directly with respect to a guarantee of solution quality. Design, analysis and applications of heuristic algorithms are integral parts of the artificial intelligence field [23].

The second related field is *graph mining*, which aims at analysis of large real-world graphs using algorithmic methods. Emphasis in graph mining is not only on using the properties of real-world graphs, but also on scalability of the algorithms, which are used to discover information from structure of the graphs [3].

In the following, we present a randomized heuristic approach to solve CCP. This approach uses our own *greedy clique covering* (GCC) algorithm, which receives a permutation of vertices and partitions the vertices into cliques in the order determined by the permutation. Next, we use *iterated greedy* (IG) as a technique to optimize the permutation, thus, optimizing the resulting clique covering. This approach showed much promise for real-world networks. As the last step, we extended IG with a heuristic to compute a lower bound, since for real-world networks, we do not know the optimum in advance and have to bound it. A lower bound based on a heuristic for maximum independent set was used. This, in combination with GCC and IG, allowed us to show that CCP can be solved near-optimally and often even optimally for many real-world networks.

The paper is structured as follows. In Section 2, we briefly provide the background and motivation for our investigations, as well as a short review of related work. In Section 3, we present our approach to solve CCP. In Section 4, we provide an overview of experimental results achieved by our approach. In Section 5, we shortly summarize the analytical results, which shed light on how our approach really is able to find solutions to CCP and what are its disadvantages. Finally, in Section 6, we summarize the contribution.

2. Background, Motivation and Related Work

We now formulate the studied problem formally. Let *density* of a graph $G = [V, E]$ be determined by $d(G) = \frac{|E|}{|V|(|V|-1)/2}$. Furthermore, let $G(V_i)$ denote the *subgraph of G induced by class $V_i \subseteq V$*, i.e. the graph $G' = [V_i, E_i]$, where E_i contains only the edges in E between the vertices in V_i. We will simplify the notation of graph's metrics in the way that the number of vertices $|V| = n$ and the number of edges $|E| = m$.

Let $S = \{V_1, V_2, ..., V_k\}$ be a partitioning of the vertex set V of an undirected, unweighted graph G into classes $V_1, V_2, ..., V_k$ such that

- the classes cover all vertices, i.e. $\cup_{i=1}^{k} V_i = V$ and
- the classes are non-overlapping, i.e. $\forall\ i, j$ such that $i \neq j$ it holds that $V_i \cap V_j = \emptyset$.

Then, we will call S a *(vertex) clique covering* of G if and only if all subgraphs induced by this partitioning are cliques, i.e. $\forall\ i = 1, 2, ..., k \quad d(G(V_i)) = 1$. We will refer to the minimum k, for which CCP can be solved, as the *clique covering number* of G and denote it by $\vartheta(G)$.

We note that CCP is closely related to the *graph coloring problem.* Suppose that $\overline{G}$ is the complementary graph to G, i.e. a graph containing edges between pairs of vertices, which were not present in G and vice versa. In a social network, an edge in $\overline{G}$ would mean that two persons do not know each other. Then, a solution to graph coloring problem for $\overline{G}$ represents a solution to CCP for G.

The best known approximation algorithm for graph coloring (and, thus, also for clique covering) is due to Haldórsson [13]. The approximation ratio, i.e. the ratio between the result of this algorithm and the optimum is at most $n(\log\log n)^2/(\log^3 n)$. Such an approximation ratio is not favorable for real-world applications. Therefore, *heuristic algorithms* are a suitable choice to solve both graph coloring and clique covering problems. For graph coloring, greedy heuristics are more scalable for large graphs. For smaller but structurally difficult problem instances, a large spectrum of local search and evolutionary algorithms is available [11].

It is however known that heuristics tend to perform better for some instances of a problem, while being less efficient for other types of instances. Therefore, structure of the studied graphs plays an important role in design of an efficient heuristic for CCP.

We have already indicated that CCP is practically interesting for real-world networks. Such networks often have a non-trivial structure and include social and biological networks [12], research citation networks, computer networks [25] or language networks [20]. Real-world networks are often *sparse*, i.e. they may contain a high number of vertices but the vertices are adjacent only to a limited number of other vertices. Let $m(n)$ be the number of edges as a function of the number of vertices of a network. For real-world networks, it often holds that $m(n) \prec n^2$ (in other notation, $m(n) = o(n^2)$), i.e. the number of edges grows much more slowly than the number of pairs of vertices. However, this means that for the complementary graph, the number of edges will grow quadratically. To put it into the context of social networks, a person knows only a limited number of other people, but the number of people not known will be much higher.

Even simple graph coloring heuristics have $\Omega(m)$ complexity (i.e. a complexity lower bounded by the number of edges), including greedy graph coloring [26], Brélaz's heuristic [2] and Leighton's recursive largest first heuristic [18]. For a dense graph, these are not well scalable. Therefore, we proposed our own specialized technique to solve CCP directly, which is well scalable for large sparse graphs and gives high-quality results for real-world networks.

3. Our Approach to Solve the Clique Covering Problem (CCP)

In this section, we introduce our original results. We first describe our greedy clique covering (GCC) algorithm and an iterated greedy (IG) algorithm used to improve the results obtained by GCC [4, 5]. Next, we present a technique to compute a lower bound for the optimal solution to CCP based on maximum independent set. To compute this lower bound, we proposed a randomized local search (RLS) algorithm, conceptually very similar to IG [6].

3.1 Greedy Clique Covering (GCC) and an Order-based Representation of CCP

GCC is an algorithm, which takes the vertices in a particular order and labels them. Then, the vertices, which have the same label, form a clique. The way how GCC chooses a label for a vertex is illustrated by Figure 1. Let us have a global information on how many vertices have which label. This information will be stored in array *sizes*. Let $\Gamma(v, c)$ be the number of vertices, which are neighbors of v and have label c. Then, vertex v can join the vertices with label c without violating the clique property, when $\Gamma(v, c) = sizes(c)$. When there are more values of c satisfying this condition, the one with the lowest index is chosen.

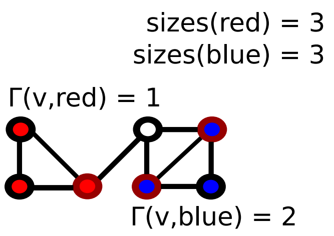

Figure 1: Illustration of the way how GCC chooses labels. The currently unlabeled vertex will receive label c if $\Gamma(v, c) = sizes(c)$. In this case, there are only 2 blue neighbors and 1 red neighbor of v. Therefore, the currently unlabeled vertex must receive its own new label.

Algorithm 1: Greedy Clique Covering (GCC)

```
Greedy Clique Covering (GCC)
Input: graph $G = [V, E]$
permutation $P = [P_1, P_2, ..., P_n]$ of vertices in $V$
Output: clique covering $S$ of $G$
1  for $c = 1..n$
2      $sizes(c) = 0$
3  for $i = 1..n$
4      $j = P_i$
5      $c = find\_equal(\Gamma(v_j, c), sizes(c))$
6      $V_c = V_c \cup \{v_j\}$
7  return $S = \{V_1, V_2, ..., V_k\}$
```

Algorithm 1 describes how GCC can be implemented. The input is a permutation P of vertices of our graph G. In steps 1-2, *sizes* array is initialized. Then, the vertices are iteratively labeled. In step 4, vertex v_j is chosen for labeling, based on permutation P. In step 5, GCC finds the label for v_j in operation $find\_equal$. This can be done in such a way that neighbors of v are iterated and values in *sizes* are decremented, based on which label each neighbor has. If some value in $sizes(c)$ reaches 0, then apparently, c is a candidate label for v_j. The original values in *sizes* can then be restored by iterating the neighbors once again. In step 6, the labeling itself is performed. The advantage of this strategy is that for each vertex, we iterate only over its neighbors to choose a label. Therefore, the time complexity of GCC is $\mathcal{O}(m)$, since $\sum_{v \in V} \deg(v) = 2m$.

It is important to note that GCC performs differently for different input permutations. Therefore, GCC can be viewed as a mapping from the space of permutations of vertices to the search space of clique coverings. More formally, GCC is a mapping μ from the space of all permutations of n objects S_n to the space of clique coverings Φ, i.e. $\mu : S_n \to \Phi$, as shown by Figure 2. In evolutionary computation, such a mapping is often called *genotype-phenotype mapping*. The aim is to find a permutation P_{opt} such that $\mu(P_{opt}) = S_{opt}$, where S_{opt} is an optimal clique covering. It can be shown that for an arbitrary graph, there always is a permutation P_{opt} with this property.

Theorem 1. For an arbitrary graph G, there is a permutation, for which greedy clique covering will produce the optimal solution with $\vartheta(G)$ cliques.

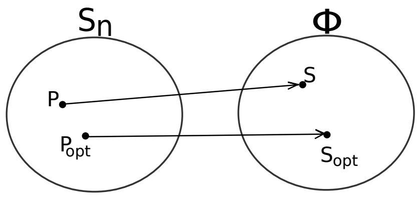

Figure 2: Illustration of GCC as a mapping. GCC can be viewed as function which, for a permutation $P \in S_n$, returns a clique covering $S \in \Phi$. Therefore, the problem is solved indirectly. Instead of searching for an optimal clique covering S_{opt}, one can search for an optimal permutation P_{opt}, for which GCC constructs S_{opt}.

Algorithm 2: Iterated Greedy (IG) Algorithm for CCP

```
Iterated Greedy (IG) Algorithm
for CCP
Input: graph $G = [V, E]$
Output: clique covering $S$ of $G$
1   $P = random\_permutation(1, 2, ..., n)$
2   while stopping criterion is not met
3       $\{V_1, V_2, ..., V_k\} = GCC(G, P)$
4       with $p_{rev}$ probability
5           $P = [V_k, V_{k-1}, ..., V_1]$
6       else
7           $P = random\_permutation(V_1, V_2, ..., V_k)$
8       if $\vartheta(G)$ is known and $k = \vartheta(G)$
9           return $S = \{V_1, V_2, ..., V_k\}$
10  return $S = \{V_1, V_2, ..., V_k\}$
```

With GCC, we established an *order-based representation of CCP*, i.e. the problem of searching for the optimal clique covering is now transformed into a problem of searching for an optimal ordering. As we have already indicated, the key property of our approach is its scalability. GCC performs in $\mathcal{O}(m)$ time, which is favorable for sparse graphs. This will allow one to use GCC repeatedly many thousands of times in a stochastic search algorithm and achieve reasonable results in a reasonable time even for large graphs.

3.2 Iterated Greedy (IG) Clique Covering

Iterated greedy (IG) is a stochastic local search technique [14], which combines a greedy algorithm with stochastic optimization of its component. It was used for the first time by Chvátal to solve the set covering problem [7], which suggests that it is suitable for covering problems.

Our main inspiration comes from the iterated greedy algorithm for graph coloring proposed by Culberson and Luo [9, 10]. This algorithm used block-based properties of greedy graph coloring algorithm. We now describe a way how to use such an idea analogically to improve the results obtained by GCC.

The way how our IG works is given by the pseudocode of Algorithm 2. In step 1, we generate a permutation of vertices uniformly at random. Next, an iterative procedure is performed. In step 3, we use GCC with permutation P to construct a clique covering $\{V_1, V_2, ..., V_k\}$. Then, one of two possible block-based mutations is performed. The first is the *reverse mutation*, where the blocks $V_1, V_2, ..., V_k$ are put into the next permutation in the reverse order. The second one is the *random mutation*, by which the blocks are randomly shuffled. We note that the vertices within blocks are internally preserved. This is done in steps 4-7. If we know the optimal value $\vartheta(G)$, the improvement process is stopped in steps 8-9. Steps 2-9 are repeated until a stopping criterion is met.

One should note that by preserving the blocks internally, the number of cliques used to cover the graph can only become lower. This is because a new block simply cannot be created this way, because of the greedy nature of GCC. Therefore, although the random mutation operator performs at the level of blocks entirely at random, the search is not blind. The objective function is non-increasing during the run of IG, which is typical for local search algorithms.

3.3 Randomized Local Search (RLS) for a Lower Bound

In our experimental work, we quickly noticed that IG seems to perform well for real-world networks [4]. However, one usually does not know the optimal number of cliques $\vartheta(G)$ for a network with an unknown structure. Therefore, it is practically interesting to bound the optimum from below. This way, we can determine how well our algorithm performed. It can be easily shown that the clique covering number is bounded from below and from above according to Lemma 1.

Lemma 1. Let G be an undirected graph with minimum degree $\delta_{min}(G)$, clique covering number $\vartheta(G)$, maximum independent set size $\alpha(G)$ and maximum clique size $\omega(G)$. Then, $\vartheta(G)$ is bounded in the following way:

$$\max\left\{\alpha(G), \frac{n}{\omega(G)}\right\} \leq \vartheta(G) \leq n - \delta_{min}(G). \quad (1)$$

For real-world networks, the max operation in the lower bound will almost always evaluate to $\alpha(G)$, i.e. $\alpha(G)$ will be a tighter bound. The size of maximum independent set can be easily interpreted in the context of social networks. It is the maximum size of a group, in which nobody knows each other. Obviously, if $\alpha(G)$ is the size of such a set, it takes $\alpha(G)$ cliques to cover its vertices.

The only issue here is that maximum independent set problem is also NP-hard [16]. Therefore, it is a problem for heuristic algorithms, too. We designed a randomized local search algorithm (RLS) to estimate the maximum independent set. It is specified in pseudocode of Algorithm 3. We note that RLS_p^1 means that 1 mutation is performed per iteration and the algorithm searches in the space of permutations of vertices.

RLS_p^1 starts with a random permutation of vertices, generated in step 1. In step 3, greedy independent set procedure is used. This procedure takes the vertices in order determined by the current permutation P. For each vertex, we simply determine whether it can be added to the independent set. This occurs when the vertex is not adjacent to any vertex within the independent set. This way, some independent set is generated. In step 6, we choose a random vertex from the current permutation P and move it to the first position in the permutation. The other vertices are then shifted to the right. Greedy independent set procedure is used once again. The new permutation is accepted if and only if the new independent set is at least as large as the previous one. This is repeated until a stopping criterion is met.

Algorithm 3: RLS_p^1 Algorithm for the Maximum Independent Set Size

```
   RLS_p^1 Algorithm for the Maximum
   Independent Set Size

   Input: graph $G = [V, E]$
   Output: the size $\alpha(G)$ of the estimated maximum
   independent set
1  $P = random\_permutation(1, 2, ..., n)$,
   $P^* = P$, $k^* = 1$
2  while $stopping\_criterion$ is not met
3      $k = |greedy\_independent\_set(G, P)|$
4      if $k \geq k^*$
5          $k^* = k$, $P^* = P$
6      $j = uniformly\_random(2, n)$
7      $P = jump(j, 1, P^*)$
8  return $\alpha(G) = k^*$
```

4. Overview of Experimental Results

The verification of our approach was conducted using both experimental and analytical methodologies. Let us first briefly present the experimental results we obtained for our approach. These results can be divided into three main parts:

- a case study of IG on graphs with planted cliques,
- experimental comparison of IG with Brélaz's heuristic and its modification,
- verification of quality of results obtained by IG and RLS on real-world networks.

In the first step, we conducted a case study of IG on *graphs with planted cliques*. A graph with planted cliques consists of embedded cliques with some predefined size, it is a special case of planted partition model of clustered graphs [8]. Planted cliques are connected uniformly at random. The main aim of our study was to determine how well IG is able to restore these planted cliques. On graphs, which were sparse enough between the planted cliques, IG performed very efficiently and was able to find the optimum. Additionally, we discovered that larger planted cliques took less time to discover than smaller cliques [4].

In the next step, we compared IG to Brélaz's heuristic (denoted by BRE) and its modification we called saturation-based greedy clique covering (SAT-GCC). Brélaz's heuristic was chosen because of its good tradeoff between scalability and quality of results, since it works in $\mathcal{O}(n^2)$ time [2]. SAT-GCC uses a strategy similar to Brélaz's heuristic but can be implemented to run faster. In SAT-GCC, after assignment of a vertex into a clique, for each of its neighbors, which are adjacent to all of the vertices of the current clique, its saturation is incremented. Vertices are ordered primarily based on highest saturation, secondarily based on lowest degree and possible ties are resolved at random.

Table 1: The comparison of the quality of results obtained by the Brélaz's heuristic (BRE), saturation-based greedy clique covering (SAT-GCC) and the iterated greedy (IG) heuristic using GCC on CCP. The numbers denote the approximations of the clique covering number $\vartheta(G)$ for each graph and each heuristic. These were computed for uniform random graphs, Leighton graphs, which contain embedded cliques of different sizes and graphs obtained by crawling a local social network. The best results are highlighted in bold.

G	BRE	SAT-GCC	IG
Erdős-Rényi uniform random graphs			
$unif1000\_0.1$	302	310	**242**
$unif5000\_0.1$	1241	1290	**1064**
$unif10000\_0.1$	2326	2403	**2030**
$unif20000\_0.01$	7640	7863	**6403**
Leighton graphs from DIMACS instances			
$le450\_15a$	85	89	**80**
$le450\_15b$	92	90	**82**
$le450\_15c$	72	76	**58**
$le450\_15d$	73	73	**59**
$le450\_25a$	**91**	94	**91**
$le450\_25b$	81	83	**80**
$le450\_25c$	61	62	**55**
$le450\_25d$	60	59	**51**
Social graphs			
$soc1000$	**759**	**759**	**759**
$soc2000$	**1471**	1475	**1471**
$soc10000$	6620	6671	**6618**
$soc20000$	12770	12899	**12764**

Table 1 shows the comparison of the numbers of cliques found by the three studied algorithms. The columns denote the three algorithms, the rows denote the instances and the values are the numbers of cliques found by a particular algorithm for the respective test graph. As test instances, we used Erdős-Rényi uniform random graphs, in which edges are put between each pair of vertices with constant probability. Leighton graphs are also pseudorandom and contain embedded cliques with several different sizes. These graphs were designed to model large scheduling problems [18]. However, searching for the original embedded cliques is also a relevant task. Last but not least, we used several extracts of a Slovak web-based social network. Table 1 illustrates the clear dominance of our IG algorithm over these well-scalable techniques. One can see that the difference in performance of Brélaz's heuristic and IG is more pronounced in random graphs and Leighton graphs. However, the most encouraging result is probably the fact that IG surpassed Brélaz's heuristic on large social graphs with 10000 and 20000 vertices. This gives a hint that IG might be a very suitable choice to solve CCP in large real-world networks.

Finally, we tested IG and RLS on set of real-world networks. For this purpose, we used the extracts of a web-based Slovak social network, denoted by Social network I, which were already used in experiments summarized in Table 1. Social network II is a neighborhood of a single user from another social network. Network science instances are taken from other sources. These instances include a language network describing adjective-noun adjacencies ($adjnoun$), a collaboration network for the field of network science ($netscience$), a network of friendships in a karate club ($zachary$), a network describing games in a season of an American college football league ($football$) and a snapshot of Internet on the level of autonomous systems ($as - 22july06$)[1]. The last set of instances includes coappearance networks for several classical literary works, which describe whether two characters in a book encounter each other[2].

As we have already mentioned, the aim of using RLS is to find a lower bound for the clique covering number $\vartheta(G)$. This way, we are able to determine how good solutions our IG algorithm is in fact able to find. Table 2 summarizes the lower bounds $\vartheta_L(G)$ obtained by RLS and the numbers of cliques $\vartheta_U(G)$ used by IG to solve CCP for a particular graph. The table contains names and sources for the graphs, the values obtained by the algorithms and the average size of a clique $\frac{n}{\vartheta_U(G)}$ in the obtained solutions. The cases, for which we obtained $\vartheta_L(G) = \vartheta_U(G)$, are highlighted in bold. Naturally, for these cases, we can declare that *IG was able to find the optimal solutions.*

Using the bounds obtained by RLS, one can see that IG was able to solve CCP optimally for 13 out of 17 of these real-world networks. For the other 4 instances, the optimal value $\vartheta(G)$ was limited to a small interval. Interestingly, the instances where IG performed optimally, include instances from all categories. It was able to solve the problem for social networks, coappearance networks and a collaboration network. We also note that the results were performed with high number of successful runs over 30 independent runs of both IG and RLS.

5. Overview of Analytical Results

In the previous section, we have shown that IG performs well experimentally. However, this did not give us much insight into how the approach really works. Especially, one can wonder how the random decisions by block-based mutation operators are able to improve solutions. Insight into this is important to understand what are the advantages and disadvantages of IG. Therefore, we aimed also to conduct an analytical investigation on the behavior of IG [5].

At this point, we give an overview of the methods we used and the analytical results we obtained. As we have already indicated, IG is a stochastic local search algorithm. Therefore, methods of analysis for evolutionary algorithms are suitable to analyze its convergence properties and obtain bounds on its expected runtime [21].

There are two methods we used to analyze the runtime of IG on several chosen classes of graphs.

1. *Fitness levels (fitness-based partitions).* This method is based on partitioning of the search space into several levels, based on values of the objective function, such that an algorithm can only move to a better

[1]To the best of our knowledge, this Internet snapshot was not previously published. It is available on this site: `http://www-personal.umich.edu/~mejn/netdata/`.

[2]All these instances are available or a link to their source is provided at: `http://www.fiit.stuba.sk/~chalupa/benchmarks/ccp`

Table 2: Summary of the lower bounds $\vartheta_L(G)$ obtained by RLS and the numbers of cliques $\vartheta_U(G)$ used by IG on complex network instances. The results, for which we obtained that $\vartheta_L(G) = \vartheta_U(G)$, are highlighted in bold. There are 13 such networks, for which we showed this way that the result obtained by IG is actually the optimal clique covering [6].

source of G	file name	$\vartheta_L(G)$	$\vartheta_U(G)$	$\frac{n}{\vartheta_U(G)}$
Web-based social network extracts [4]				
Social network I.	$soc500$	**377**	**377**	1.33
Social network I.	$soc1000$	**759**	**759**	1.32
Social network I.	$soc2000$	1470	1471	1.36
Social network I.	$soc10000$	**6618**	**6618**	1.51
Social network I.	$soc20000$	**12764**	**12764**	1.57
Social network II.	$soc52$	**15**	**15**	3.47
Network science instances				
Adjective-noun adjacencies [22]	$adjnoun$	53	55	2.04
Network science collaborations [22]	$netscience$	**690**	**690**	2.30
Les Miserables network [17]	$lesmis$	**35**	**35**	2.20
Zachary Karate Club [27]	$zachary$	**20**	**20**	1.70
American College Football [12]	$football$	21	22	5.23
Snapshot of the Internet	$as-22july06$	19660	19661	1.17
Characters' coappearance networks from DIMACS coloring instances [15]				
Anna Karenina	$anna$	**80**	**80**	1.73
David Copperfield	$david$	**36**	**36**	2.42
Huckleberry Finn	$huck$	**27**	**27**	2.74
Iliad and Odyssey	$homer$	**341**	**341**	1.65
Jean Valjean	$jean$	**38**	**38**	2.11

fitness level or stagnate. If p_i is the minimum probability of improvement from i-th level to a better fitness level, then the expected waiting time for an improvement to a better fitness level is $1/p_i$. Consequently, runtime is determined by the sum of waiting times over all suboptimal fitness levels [21].

2. *Cover time of random walks.* This method is used to analyze the runtime of evolutionary algorithms on plateaus, i.e. areas consisting of solutions with equal values of objective function. On a plateau, a search algorithm behaves like a random walk. For a plateau with $|V|$ solutions and $|E|$ transformations between them, it holds that the expected time to visit each solution at least once is $2|E|(|V|-1)$ [1].

In runtime analysis of IG, we usually used fitness levels to upper bound the expected runtime of IG. A particular fitness level usually consisted of clique coverings with equal number of cliques. To estimate the expected time to obtain an improvement to a better fitness level, we used either the cover time of random walks or some specific knowledge about the structure of the graph.

Table 3 summarizes the analytical results we proved for GCC and IG. For the cases, where the algorithms behave suboptimally, we used metrics of approximation quality. Let f_{opt} be the optimal value of objective function for a minimization problem and let f be the worst possible value obtained by the algorithm. Then, we define approximation ratio as $R = f/f_{opt}$ and maximum discrepancy as $D = f - f_{opt}$.

Star graphs consist of one central vertex adjacent to several vertices with degree 1. Both GCC and IG behave optimally in linear time on this class of graphs. This is an example of a class of graphs, where block-based mutation is not needed, since GCC gives the optimum for any permutation. On *paths* (graphs consisting of vertices chained to a "line"), GCC achieves only 4/3-approximation, while IG behaves optimally in $\mathcal{O}(n^5)$ time. On *complements of bipartite graphs*, it depends on the number of edges between the partitions. If the graph is sparse enough, IG behaves optimally in $\mathcal{O}(n^3)$ time, while GCC may overestimate with high discrepancy. However, there is a complement of a bipartite graph, where IG can get stuck with probability lower bounded by 1/15. Finally, both GCC and IG have a similar worst-case result. There is a class of graphs, where GCC gives a suboptimal solution and IG is not able to improve it using block-based mutations with an overwhelming probability.

6. Conclusions

We proposed a new technique to solve the (vertex) *clique covering problem* (CCP). The aim of CCP is to partition the vertices of a network into minimum number of groups, which induce cliques (subgraphs with all pairs of vertices being adjacent). In the context of social networks, CCP can be viewed as a strict variant of *community detection*, where community is viewed as a group of persons, in which everybody knows each other.

The main contributions of our work were presented in three parts. Each part was published in a separate paper [4, 5, 6].

In the first part, we proposed our *greedy clique covering* (GCC) algorithm as a mapping of a permutation of vertices to a clique covering. Next, we extended it to an iterated greedy (IG) algorithm. The key advantage of GCC over existing approaches is that it works in $\mathcal{O}(m)$ time, where m is the number of edges of the graph. This makes GCC well-scalable and useable thousands of times

Table 3: Overview of the analytical results on GCC and IG on several studied graph classes. Quality of results and runtime of the algorithms are given, R denotes approximation ratio and D denotes maximum discrepancy between the found solution and the optimum.

graph class	GCC	IG
star graphs	optimal, $\mathcal{O}(n)$ time	optimal, $\mathcal{O}(n)$ time
paths	suboptimal, $R = 4/3$	optimal, $\mathcal{O}(n^5)$ time
complements of bipartite graphs: partitions of size $n/2^*$, $m_{out} < n/2$	suboptimal, $D = n/2 - 1$	optimal, $\mathcal{O}(n^3)$ time
complements of bipartite graphs: partitions of size $n/2^*$, $m_{out} \geq n/2$	suboptimal, $D = n/2 - 1$	can be suboptimal with probability $p \geq 1/15$
worst-case result	can be suboptimal with probability $1 - o(1)$	can be suboptimal with probability $1 - o(1)$

* Here, we assume that n is even, i.e. $n/2$ is an integer.

in an iterative optimization algorithm. We designed IG as such an algorithm. First experimental results showed that IG is able to perform well on graphs with planted cliques, where the aim was to restore the planted cliques. Additionally, IG experimentally surpassed the results of the well-known Brélaz's graph coloring heuristic applied to solve CCP.

Secondly, we studied the properties of GCC and IG analytically. We managed to show analytically, when the block-based mutation operators used by IG are able to improve the solutions and what are their disadvantages. We formulated a sufficient condition for an improvement by block-based mutation operator to occur. We have shown that IG is able to find the optimum in *polynomial time* on *path graphs* (chain graphs) and *sparse complements of bipartite graphs*. On the other hand, we showed that there is a class of graphs, where IG will tend to get stuck in a suboptimal solution with an overwhelming probability. Fortunately, such a class of graphs is highly artificial and unlikely to appear in real-world applications.

Last but on least, we experimentally demonstrated that IG is a suitable technique to solve CCP in *real-world networks*. For real-world networks, one usually does not know the optimal value in advance. Therefore, we proposed another heuristic to determine how far the solutions obtained by IG are from the optimum. We designed a randomized local search (RLS) algorithm for maximum independent set to accomplish this goal. IG and RLS were experimentally tested on social networks, on several graphs studied in network science, as well as on a set of coappearance networks for classical literary works. For 13 out of 17 graphs, IG found the optimum, while for the other 4 graphs, the optimum was bounded inside a very small interval.

Acknowledgement. This contribution was supported by Grant Agency VEGA SR under the grants 1/0553/12 and 1/0458/13.

References

[1] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovasz, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *Proc. of the 20th Annual Symposium on Foundations of Computer Sc., FOCS '79*, pp 218–223, 1979.

[2] D. Brélaz. New methods to color vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[3] D. Chakrabarti and C. Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38(1):Article No. 2, 2006.

[4] D. Chalupa. On the efficiency of an order-based representation in the clique covering problem. In T. Soule and J. Moore, editors, *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, GECCO '12, pages 353–360, New York, NY, 2012. ACM.

[5] D. Chalupa. An analytical investigation of block-based mutation operators for order-based stochastic clique covering algorithms. In C. Blum and E. Alba, editors, *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, GECCO '13, pages 495–502, New York, NY, 2013. ACM.

[6] D. Chalupa. Construction of near-optimal vertex clique covering for real-world networks. *Computing and Informatics*, to appear.

[7] V. Chvátal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[8] A. Condon and R. M. Karp. Algorithms for graph partitioning on the planted partition model. In D. S. Hochbaum, K. Jansen, J. D. P. Rolim, and A. Sinclair, editors, *Proceedings of the 3rd International Workshop on Approximation Algorithms for Combinatorial Optimization Problems: Randomization, Approximation, and Combinatorial Algorithms and Techniques*, RANDOM-APPROX '99, pages 221–232, London, 1999. Springer.

[9] J. C. Culberson. Iterated greedy graph coloring and the difficulty landscape. Technical Report TR92-07, University of Alberta, 1992.

[10] J. C. Culberson and F. Luo. Exploring the k-colorable landscape with iterated greedy. In D. S. Johnson and M. Trick, editors, *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*, pages 245–284, Providence, RI, 1995. American Mathematical Society.

[11] P. Galinier and A. Hertz. A survey of local search methods for graph coloring. *Computers & Operations Research*, 33(9):2547–2562, 2006.

[12] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[13] M. M. Halldórsson. A still better performance guarantee for approximate graph coloring. *Information Processing Letters*, 45(1):19–23, 1993.

[14] H. H. Hoos and T. Stützle. *Stochastic local search: Foundations & applications*. Elsevier, Amsterdam, 2004.

[15] D. S. Johnson and M. Trick. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*. American Mathematical Society, Providence, RI, 1996.

[16] R. M. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Proceedings of a Symposium on the Complexity of Computer Computations*, pages 85–103. Plenum Press, New York, NY, 1972.

[17] D. E. Knuth. *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, Reading, MA, 1993.

[18] F. T. Leighton. A graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards*, 84(6):489–503, 1979.

[19] J. Leskovec, K. J. Lang, and M. W. Mahoney. Empirical comparison of algorithms for network community detection. In J. F. M. Rappa, P. Jones and S. Chakrabarti, editors, *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 631–640, New York, NY, 2010. ACM.

[20] P. Náther and M. Markošová. Positional word web and its numerical and analytical studies. *Computing and Informatics*, 30(6):1287–1302, 2011.

[21] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization - Algorithms and Their Computational Complexity*. Springer, Berlin / Heidelberg, 2010.

[22] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(036104):036104–1–036104–19, 2006.

[23] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach (2nd ed.)*. Prentice Hall, Upper Saddle River, NJ, 2003.

[24] S. E. Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007.

[25] J. Sun, Y. Xie, H. Zhang, and C. Faloutsos. Less is more: Sparse graph mining with compact matrix decomposition. *Statistical Analysis and Data Mining*, 1(1):6–22, 2008.

[26] D. J. A. Welsh and M. B. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967.

[27] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

Selected Papers by the Author

D. Chalupa. Construction of Near-Optimal Vertex Clique Covering for Real-world Networks. *Computing and Informatics*, to appear.

D. Chalupa. An Analytical Investigation of Block-based Mutation Operators for Order-based Stochastic Clique Covering Algorithms. In C. Blum and E. Alba, editors, *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, GECCO 2013, Amsterdam, the Netherlands, 2013. ACM, New York, 495–502.

D. Chalupa. On the Efficiency of an Order-based Representation in the Clique Covering Problem. In T. Soule and J. Moore, editors, *Proceedings of the 14th annual conference on Genetic and evolutionary computation*, GECCO 2012, Philadelphia, PA, USA, 2012. ACM, New York, 353–360.

D. Chalupa. Solving Clique Covering in Very Large Sparse Random Graphs by a Technique Based on k-Fixed Coloring Tabu Search. In M. Middendorf and C. Blum, editors, *Proceedings of the 13th European conference on Evolutionary Computation in Combinatorial Optimization*, EvoCOP 2013, Vienna, Austria, 2013. Springer, Berlin / Heidelberg, 238–249.

D. Chalupa, J. Pospíchal. Metaheuristically Optimized Multicriteria Clustering for Medium-Scale Networks. In V. Snášel, A. Abraham and E. S. Corchado, editors, *Soft Computing Models in Industrial and Environmental Applications, 7th International Conference*, SOCO '12, Ostrava, Czech Republic, Advances in Intelligent Systems and Computing, Vol. 188, 2013. Springer, Berlin / Heidelberg, 337–346.

D. Chalupa. Population-based and learning-based metaheuristic algorithms for the graph coloring problem. In N. Krasnogor and P. L. Lanzi, editors, *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, GECCO 2011, Dublin, Ireland, 2011. ACM, New York, 465–472.

D. Chalupa. Adaptation of a Multiagent Evolutionary Algorithm to NK Landscapes. In C. Blum and E. Alba, editors, *Proceedings of the 15th annual conference companion on Genetic and evolutionary computation*, GECCO 2013, Amsterdam, the Netherlands, 2013. ACM, New York, 1391–1398.

D. Chalupa and J. Pospíchal. On the Growth of Independent Sets in Scale-free Networks. In *Nostradamus 2014: Prediction, Modeling and Analysis of Complex Systems*, Ostrava, Czech Republic, Advances in Intelligent Systems and Computing, 2014. Springer, Berlin / Heidelberg, to appear.

D. Chalupa. On the Ability of Graph Coloring Heuristics to Find Substructures in Social Networks. *Information Sciences and Technologies Bulletin of ACM Slovakia*, 3(2):51–54, 2011.

D. Chalupa. An optimization strategy based on social insect behavior. In M. Bieliková, I. Černá, T. Gyimóthy, J. Hromkovič, K. Jeffery, R. Královič, M. Vukolić and S. Wolf, editors, *SOFSEM 2011: Student Research Forum*, Nový Smokovec, Slovakia, 2011. Okat, Bratislava, 35–50.