Computing and Informatics, Vol. 37, 2018, 385–404, doi: 10.4149/cai_2018_2_385

ANALYSIS OF ITERATED GREEDY HEURISTIC FOR VERTEX CLIQUE COVERING

David Chalupa

Operations Research Group, Department of Materials and Production
Aalborg University, Fibigerstræ de 16
Aalborg 9220, Denmark
&
Computer Science, School of Engineering and Computer Science
University of Hull, Cottingham Road, Hull HU6 7RX, United Kingdom
e-mail: dc@m-tech.aau.dk

Jiří Pospíchal

Faculty of Natural Sciences, University of Ss. Cyril and Methodius
Námestie J. Herdu 2, 917 01 Trnava, Slovakia
e-mail: jiri.pospichal@ucm.sk

Abstract. The aim of the vertex clique covering problem (CCP) is to cover the vertices of a graph with as few cliques as possible. We analyse the iterated greedy (IG) algorithm for CCP, which was previously shown to provide strong empirical results for real-world networks. It is demonstrated how the techniques of analysis for randomised search heuristics can be applied to IG, and several practically relevant results are obtained. We show that for triangle-free graphs, IG solves CCP optimally in expected polynomial time. Secondly, we show that IG finds the optimum for CCP in a specific case of sparse random graphs in expected polynomial time with high probability. For Barabási-Albert model of scale-free networks, which is a canonical model explaining the growth of social, biological or computer networks, we obtain that IG obtains an asymptotically optimal approximation in polynomial time in expectation. Last but not least, we propose a slightly modified variant of IG, which guarantees expected polynomial-time convergence to the optimum for graphs with non-overlapping triangles.

Keywords: Vertex clique covering problem, iterated greedy, randomised search heuristics, complex networks, random graphs

Mathematics Subject Classification 2010: 68R10, 05C69, 05C70

1 INTRODUCTION

This paper is dedicated to the analytical study of an *iterated greedy* (IG) heuristic for the vertex *clique covering problem* (CCP) in several practically relevant classes of graphs, including triangle-free graphs, sparse random graphs and models of *complex networks*. These networks include social networks [21, 35], biological networks [12], research citation and collaboration networks [21, 34], language networks [29] or the Internet [4]. Both methods and software tools for exploration of complex networks are developed [14].

The problem we study in this work is closely related to the popular areas of community detection [28], graph clustering [34] and graph mining [9]. The aim of CCP is to partition the vertices into as few pairwise disjoint subsets as possible such that each subset induces a clique. In the context of social networks, CCP is a problem of "strict" community detection, in which the vertices are partitioned into the minimum number of groups so that everybody knows each other within each group.

Definition 1 (Definition of CCP)**.** Let $G = [V, E]$ be an undirected graph on n vertices and m edges. Let $d(G) = \frac{2m}{n(n-1)}$ be its density, with $d(G) = 1$ if $0 \leq n \leq 1$. The objective of CCP is to minimise $k \leq n$ such that there are classes $V_1, V_2, \dots, V_k$, satisfying the following constraints:

1. each vertex is in exactly one class, i.e.
$$\forall\, i, j = 1..k, i \neq j : V_i \cap V_j = \emptyset, \quad \bigcup_{i=1}^{k} V_i = V,$$
2. each class induces a clique, i.e.
$$\forall i = 1..k : d(G(V_i)) = 1,$$

where $G(V_i) = [V_i, E(V_i)]$ is a *subgraph induced by* V_i, containing only edges between vertices of class V_i. The minimum value of k for which there is a clique covering will be referred to as the *clique covering number* and denoted by $\vartheta(G)$ [10].

CCP is one of the classical NP-hard problems [24]. It corresponds to graph colouring of the complementary graph, which perhaps explains why the current literature mostly overlooks this problem and focuses more on graph colouring. The relationship between CCP and graph colouring influences the approximation results on CCP. To the best of our knowledge, the best general approximation algorithm for CCP is the one for graph colouring, which achieves approximation ratio $\mathcal{O}(n(\log \log n)^2/(\log n)^3)$ [23]. However, better approximation ratio may be obtained or the problem may be solved in polynomial time for restricted graph classes [8].

We note that the similar edge clique covering problem (ECCP) is also studied and is NP-hard, too. For ECCP, more studies seem to be currently published, especially for specific classes of graphs [5, 22, 25].

Iterated greedy (IG) algorithm was previously demonstrated to provide encouraging empirical results for CCP in real-world networks [11]. IG is a heuristic algorithm, which utilises the block-based properties of CCP to find high-quality solutions efficiently. In this context, it is closely related to evolutionary algorithms, as well as randomised search heuristics [3].

Even though IG does not guarantee that the best solution is always found, it usually performs well in practice. It is able to find optimal or near-optimal solutions for social and research collaboration networks [11], as well as protein-protein interaction networks [12]. In addition, IG does not use any prior knowledge of a specific graph class to make the optimisation more efficient. Therefore, even though more suitable algorithms can be found for specific families of graphs, the aim of this paper is to explore the capabilities of a more general approach. Similarly to the research on other randomised search heuristics [32], we obtain that IG mimics the behaviour of classical algorithms to some extent, provably finding optimal or asymptotically optimal solutions in polynomial time for several practically relevant graph classes.

1.1 Contributions

It was previously shown that IG finds the optimal solution for paths in polynomial time [10]. We extend this result by first showing that the behaviour of IG for triangle-free graphs can be modelled using random walks and we prove that the optimal solution is found in expected $\mathcal{O}(n^5m^2)$ time. This bound is based on rather pessimistic assumptions. IG seems to be much faster in practice.

Next, we show that these arguments can be generalised to sparse random graphs generated according to the Erdős-Rényi model [16] $\mathbb{G}(n, c/n)$, i.e. graphs on n vertices with randomly generated edge with probability c/n for each pair of vertices. We show that for graphs generated with $c < 1$, IG will find the optimal clique covering in expected $\mathcal{O}(n^3(\log n)^5)$ time with probability $1 - o(1)$.

As a next step, we study the behaviour of IG for the Barabási-Albert (BA) model of scale-free networks, which is a canonical model explaining the growth of social and other complex networks [4]. We obtain that IG achieves approximation ratio $1+\mathcal{O}\left(\frac{(\log n)^3}{n}\right)$ for graphs generated by BA model in expected polynomial time. This approximation ratio is asymptotically optimal.

Last but not least, we show that even though IG can fail to provide the optimum for graphs with non-overlapping triangles with probability $1 - o(1)$ [10], this drawback can be overcome by putting the triangles as blocks in the initial solution. Such modification leads to an algorithm, which finds the optimum in expected polynomial time.

Even though most of these results are not particularly surprising, our analysis introduces several insights into the behaviour of heuristics, which combine a classical greedy approach with randomised search. It confirms that IG can be viewed as a randomised local search algorithm, with its behaviour modelled using methods of analysis of evolutionary algorithms. This includes the fitness levels method [27, 32, 36], as well as methods modelling the optimisation process as a random walk [2].

The rest of the paper is structured as follows. In Section 2, we briefly review the background of CCP, IG algorithm and related work. In Section 3, we show that IG finds the optimal solution for triangle-free graphs in expected polynomial time. In Section 4, we show that IG finds the optimal solution for the specific case of sparse random graphs in expected polynomial time with high probability. In Section 5, we consider the impact of triangles upon our problem. In Section 6, we show that IG achieves asymptotically optimal approximation ratio for graphs generated by BA model in expected polynomial time. In Section 7, we show how to extend IG so that it guarantees that the optimum is found for graphs with non-overlapping triangles. In Section 8, we give conclusions and summarise the current open problems.

2 ITERATED GREEDY CLIQUE COVERING

IG is a randomised search heuristic, i.e., it does not guarantee that optimal solution is found but it might provide very good results for certain types of problem instances. IG was previously successfully used to solve graph colouring [13], train scheduling [42] or flowshop scheduling problem [33].

Over the last years, analysis of randomised search heuristics in combinatorial optimisation problems has become a very active research area [3, 32]. Problems, for which results have been published, include polynomial-time solvable problems such as the maximum matching problem [20], Eulerian cycle problem [30] or minimum spanning tree problem [31]. However, NP-hard problems are also often considered, including the vertex cover problem [18, 26, 41], Euclidean travelling salesperson problem [38] or the graph colouring problem [37].

At this point, we move on to the description of our IG algorithm for CCP. The roots of this algorithm date back to the work by Culberson and Luo [13], who used a similar approach to solve the graph colouring problem. Inspired by this work, we have relatively recently developed an IG algorithm for CCP. Interestingly, our previous results indicated that IG has all the features of typical *local search.* For paths, IG converges to the optimum in polynomial time, for complements of bipartite graphs, it can get stuck in local optima and there are also specific graph classes, where IG will get stuck in local optima almost certainly [10].

However, the current theoretical results for IG are still relatively distant from its main application in social and other complex networks. In our previous empirical study, IG was able to find optimal solutions for real-world graphs in many cases, while in the rest of the cases, the obtained solutions were very close to the optimum [11]. Therefore, further analytical results for IG are of a high interest.

2.1 Description of IG

Our IG algorithm uses *greedy clique covering* (GCC) [10]. GCC begins with an empty clique covering. Technically, the cliques are marked with labels, similarly to the graph colouring problem. GCC takes the vertices in an order determined by input permutation P. In each iteration, it puts a vertex into the first clique (i.e. with the lowest index of its label) such that the clique property is not violated. If this is not possible, a new label is used, leading to a new clique being created. This way, a solution is iteratively constructed. We will refer to the choice of the first clique as the *First Fit rule* [39]. Efficient implementation techniques are available for GCC to run in $\mathcal{O}(m)$ time, where m is the number of edges in the graph. This makes the algorithm particularly suitable for large but sparse networks. For more detailed information on GCC, the reader may refer to the previous work [10].

```
1  begin with an uniformly random permutation $P$
2  repeat until convergence
3      construct solution $[V_1, V_2, \ldots, V_k]$ with greedy clique covering for $P$
4      let $P = [V_1, V_2, \ldots, V_k]$ so that $V_1$, $V_2$, $\ldots$, $V_k$ form blocks in $P$
5      perform *block_jump* for a uniformly randomly chosen block
       from $V_1, V_2, \ldots, V_k$ to create new $P$
```

Algorithm 1. Iterated greedy (IG) clique covering

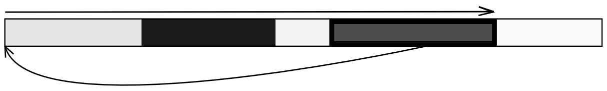

Figure 1. Illustration of the *block_jump* operator, which was introduced as a canonical block-based operator for IG [10]. Operator *block_jump* takes a chosen block representing a clique and puts it to the first position in the permutation. The other blocks are then shifted to the right.

The pseudocode of IG is given in Algorithm 1. First, GCC is used with a uniformly random initial permutation of vertices to construct the initial clique covering. Then, IG groups vertices of the identified cliques into blocks, as shown in Figure 1. One of these blocks is then taken uniformly at random and is put to the first position in the permutation. The other blocks are then shifted to the right. This operation will be further referred to as *block_jump*. GCC is used once again with the resulting permutation to construct clique covering for the next iteration. This new clique covering will never consist of more cliques than the previous one, because of the greedy nature of GCC and the fact that cliques of the previous solution form blocks. The process is repeated until a stopping criterion is met. In this paper, we will investigate the time until IG finds the optimal solution for specific graph classes.

3 RESULT FOR TRIANGLE-FREE GRAPHS

In this section, we move on to our analysis. Although IG is not a typical evolutionary algorithm, it is a closely related method. Therefore, we will use the methods of runtime analysis for evolutionary algorithms, which have been demonstrated as suitable for runtime analysis of IG.

We build our results on a relatively widely used method of *fitness levels* [27, 32, 36]. We divide the search space into levels such that each level contains all solutions with the same number of cliques. Then, Lemma 1 can be used to find an upper bound for the expected running time of our algorithm.

Lemma 1 ([32]). The expected optimization time I of a stochastic search algorithm that works at each time step with a population of size 1 and produces at each time step a new solution from the current solution is upper bounded by:

$$I \leq \sum_{i=1}^{m-1} \frac{1}{p_i}. \tag{1}$$

In Lemma 1, m represents the number of fitness levels and p_i is the minimum probability that in time step i, the stochastic change will cause an improvement. In Lemma 2, we recall the previous result on the quality of initial solution for IG for paths. This result will be used in our next discussions, since paths are a special case of triangle-free graphs.

Lemma 2 ([10]). For paths, the initial solution for IG can contain at most $\lceil 2/3n \rceil$ cliques and there are at most $\lceil n/3 \rceil$ 1-cliques in the result.

We now show that in expectation, IG finds the optimal vertex clique covering in polynomial time for *triangle-free graphs*. Even though CCP can be solved in polynomial time for triangle-free graphs using maximum matching [8], and the simple (1+1) evolutionary algorithm has previously been shown to be a polynomial-time randomised approximation scheme for maximum matching [20], it is interesting to investigate the behaviour of a more general randomised search heuristic for CCP. We will see that IG is able to guarantee polynomial-time convergence to the optimum in expectation. Additionally, analysis for triangle-free graphs represents a step towards analysis for random graphs, as well as complex networks.

It is worth noting that our bound is very pessimistic, due to assumptions used to make the proof simpler. IG seems to be much faster in most practical scenarios.

As a consequence of this result, we also have that IG solves CCP optimally in polynomial time for *trees* and *bipartite graphs* in general. This will be an extension of our previous result of IG for paths, for which IG behaves similarly [10]. However, in contrast to paths, general triangle-free graphs do not have a bounded maximum degree. This makes the random walks, which arise in the analysis of IG, to be slightly more complex than simply "left versus right". Hence, several new ideas will be introduced in the following analysis.

Theorem 1. For triangle-free graphs on n vertices and m edges, the expected time for IG to find the optimal vertex clique covering is upper bounded by $\mathcal{O}(n^5m^2)$.

Proof. Based on Lemma 1, the initial clique covering contains $\mathcal{O}(n)$ more cliques than the optimum. These will determine our fitness levels.

The size of the maximum clique $\omega \leq 2$, since we have a triangle-free graph. Cliques of size one will be called 1-*cliques* and two-vertex cliques will be called 2-*cliques.* An improvement occurs if random changes cause 1-cliques move so that some pair of 1-cliques are next to each other and form a 2-clique.

Suppose that we have a fitness level with $\vartheta + d$ cliques, where $d \geq 1$. Then, the number of 1-cliques is at least $2d \geq 2$. We will now show that 1-cliques perform a fair random walk [32] on the triangle-free graph.

We first look at what happens if *block_jump* occurs. If *block_jump* is applied to a 2-clique, only the ordering of the 2-cliques can be changed. No vertex can be taken by a 2-clique, since that would create a triangle. If *block_jump* is applied to a 1-clique, the 1-clique will form a 2-clique with its nearest following neighbour in the permutation. This is due the First Fit rule, which was mentioned in Section 2.

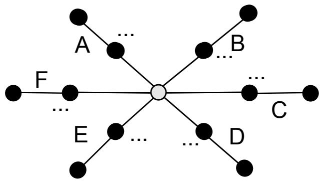

Figure 2. An illustration of the situation with a 1-clique between several 2-cliques. This picture should be perceived as a subgraph, other subgraphs can be attached to the black vertices. The direction, where the 1-clique moves when *block_jump* is applied on it depends on which of the blocks A–F comes first in the permutation.

In Figure 2, we illustrate the situation, when a 1-clique was left between several 2-cliques. Each 1-clique must necessarily have only 2-cliques around it. If it did not have, it would be joined with another 1-clique in a 2-clique.

Let X now be the waiting time until a *block_jump* of this 1-clique occurs. Before the final move of the 1-clique, there are $X - 1$ *block_jump* operations.

The direction of the movement of this 1-clique is determined by which neighbour (in Figure 2, determined by blocks A–F) comes first in the permutation. The probabilities for the directions will depend on what happens during the waiting time. More particularly, which of the blocks around the 1-clique was taken for *block_jump* as the last one. To make the proof simpler, we will pessimistically assume that the *block_jump* operations performed on the other 1-cliques during the waiting time did not lead to an improvement. Now, we will have two cases, what can happen during this waiting time.

Case 1. None of the blocks around the 1-clique jumped. Let X be the waiting time. We have that $X = 1$ (i.e., it takes only one move to choose the 1-clique) with probability $1/(\vartheta + d) \leq 2/n$. In this case, no other moves could surely be chosen. For $X > 1$, we observe that for our 1-clique vertex v with $\deg(v)$ neighbours, the probability of this event will be:

$$\left(1 - \frac{\deg(v)}{X-1}\right)^{X-1} = \left(1 - \frac{\deg(v)}{X-1}\right)^{\frac{X-1}{\deg(v)}\deg(v)} \leq e^{-\deg(v)}. \tag{2}$$

This is because in all $X - 1$ steps in the waiting time, only non-neighbour blocks were taken. Thus, the direction of movement for our 1-clique stays the same. Therefore, this case occurs with probability, which is upper bounded by $e^{-\deg(v)} + 2/n$.

Case 2. Some block around the 1-clique jumped. We are interested in which of the $\deg(v)$ blocks was the last to jump. The probability of this case is at least $1 - e^{-\deg(v)} - 2/n$, because of the bound shown in Case 1. We will now argue that this portion of probability is distributed fairly among all $\deg(v)$ neighbour blocks. This is because the probability of *block_jump* is uniformly distributed among the blocks. Thus, for each situation, where A was the last to jump, there are equally probable situations, where the last *block_jump* was performed on B, C, etc.

Hence, the probability of changing the direction of movement of 1-clique is at least $(1 - e^{-\deg(v)} - 2/n)/\deg(v)$ for each neighbour block.

During the waiting time, the solution can be changed a lot. However, if considering the neighbours of our 1-clique only, then only 2-cliques must be around it during the whole waiting time. Otherwise, an improvement would be achieved, which is a possibility that we pessimistically exclude.

Let us now consider the event outlined in Case 1. None of the blocks around the 1-clique jumped, i.e., the direction will be determined by the block, which is currently the first in the permutation. Based on the previous arguments, the probability that one fixed neighbour block (in Figure 2, one of the blocks A–F) was the first one in the beginning of the waiting time, is uniformly distributed, too. This is implied by the fact that the initial permutation is uniformly random, and the probability of *block_jump* is also uniformly distributed among the blocks. Therefore, 1-cliques actually perform fair random walks on the triangle-free graph.

From the cover time of random walks, it takes $\mathcal{O}(nm)$ *block_jump* moves of a 1-clique to visit each vertex at least once [2]. For two such random walks, we have that it takes $\mathcal{O}(n^2m^2)$ *block_jump* moves in expectation for two 1-cliques to arrive at two adjacent vertices. $\mathcal{O}(n)$ is the time needed to obtain a *block_jump* of the 1-clique and $\mathcal{O}(n)$ is the complexity of GCC.

We have $\mathcal{O}(n)$ fitness levels, on which all this happens. Therefore, the expected time to obtain the optimum is bounded by $\mathcal{O}(n^5m^2)$. □

4 RESULT FOR SPARSE RANDOM GRAPHS

We have shown that IG finds the optimum in polynomial time for triangle-free graphs. At this point, we extend this result by studying sparse random graphs, generated by the well-known Erdős-Rényi model [16]. Consider the model in the form $\mathbb{G}(n, c/n)$, generating graphs on n vertices such that an edge is put between each pair of vertices independently with probability c/n. This model has an interesting property that for $c < 1$, the graph will consist of small components with specific properties with high probability. These properties have previously been used to prove results for iterated local search algorithms for vertex cover [41] and graph colouring [37].

Theorem 2. Let $0 < c < 1$. Then, for an Erdős-Rényi random graph G from $\mathbb{G}(n, c/n)$, the expected time for IG to find an optimal clique covering for G is upper bounded by $\mathcal{O}(n^3(\log n)^5)$ with probability $1 - o(1)$.

Proof. Bollobás [6], Sudholt and Zarges [37], and Witt [41] state that, with probability $1 - o(1)$, a random graph G from $\mathbb{G}(n, c/n)$, $0 < c < 1$, will consist of components on $\mathcal{O}(\log n)$ vertices and edges, which are trees or graphs with at most one cycle.

For a tree, or a graph with cycle with at least 4 vertices, we have that the component is triangle-free, i.e., the arguments from Theorem 1 can be applied directly. The remaining case is a component with a single triangle. We first prove that for such a component, each suboptimal solution contains at least two 1-cliques. We use enumeration based on whether the optimal/suboptimal solutions contain the triangle.

Case 1. The optimum does not contain the triangle. Hence, the optimum contains only 2-cliques and 1-cliques, i.e., overestimation can occur only by using two 1-cliques instead of a 2-clique.

Case 2. The optimum contains the triangle. If the suboptimum also contains the triangle, we have the same situation as in Case 1, since overestimation can occur only by using two 1-cliques instead of a 2-clique. Suppose that the suboptimum does not contain the triangle and it does not contain a 1-clique, too. Thus, it can only contain 2-cliques. However, such a solution cannot be improved, since a substitution of two of its 2-cliques by a triangle would leave the fourth vertex for a 1-clique. Therefore, such a solution must be the optimum.

This proves that each suboptimum contains at least two 1-cliques. We now analyse the expected time to obtain a situation when the two 1-cliques visit a configuration, in which they form a 2-clique.

If the 1-cliques are in the same subtree of the component, they need to visit $\mathcal{O}((\log n)^4)$ vertices to visit adjacent vertices simultaneously, and form a 2-clique. This is implied by the fact that a component contains $\mathcal{O}(\log n)$ vertices.

When the two 1-cliques are in different subtrees, we must explore the expected time needed for them to visit vertices of the triangle simultaneously. If we assume that events in both subtrees do not lead to an improvement, we can treat them as independent. Therefore, we have that 1-cliques need to visit $\mathcal{O}((\log n)^4)$ vertices to arrive at the triangle at the same time.

Expected waiting time for a *block_jump* of a 1-clique is $\mathcal{O}(n)$. GCC has complexity $\mathcal{O}(n \log n)$ in the worst case, since we have at most n components with $\mathcal{O}(\log n)$ edges. An improvement is obtained when $\mathcal{O}((\log n)^4)$ vertex pairs are visited by two 1-cliques in a component in expectation. Expected waiting time until an improvement to a better fitness level is therefore upper bounded by $\mathcal{O}(n^2(\log n)^5)$. We have $\mathcal{O}(n)$ fitness levels, which proves our theorem. □

5 ON THE IMPACT OF TRIANGLES

Up to this point, the analysis was only taking graphs into consideration with at most one triangle per connected component. Lemma 3 summarises the negative result for a graph with linear number of non-overlapping triangles. For graph $H_{\vartheta/2}$ depicted in Figure 3, IG will get stuck in a suboptimal clique covering with probability $1 - o(1)$.

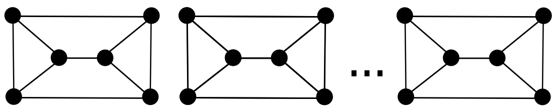

Figure 3. An illustration of the graph $H_{\vartheta/2}$, consisting of $\vartheta/2 = n/6$ connected components, for which IG does not produce the optimal vertex clique covering with probability $1 - o(1)$. This is due to the fact that if two horizontal edges are selected instead of the two triangles in at least one of the components, *block_jump* will not be able to suitably regroup the vertices [10].

Lemma 3 ([10])**.** For graph $H_{\vartheta/2}$, IG will not be able to produce the optimal vertex clique covering with probability $1 - o(1)$.

However, for graphs with a limited number of triangles, IG may achieve a good approximation of the optimum in polynomial time. In Lemma 4, we recall a lower bound for ϑ_n based on maximum independent set size α_n and maximum clique size ω_n. Consequently, Theorem 3 formulates the main approximation result.

Lemma 4 ([11])**.** Let α_n and ω_n be the sizes of maximum independent set and maximum clique for a class of graphs on n vertices, respectively. Then, it holds that $\max\{\alpha_n, n/\omega_n\} \leq \vartheta_n$.

Theorem 3. Let G be a graph on n vertices with τ_n triangles such that $\tau_n < n/3$. Then, IG will achieve approximation ratio:

$$1 + 6\frac{\tau_n}{n - 3\tau_n} \tag{3}$$

for G in expected polynomial time.

Proof. Let $V_T \subseteq V$ be the set of vertices in G, which are in at least one triangle. Based on the premises, we have that $|V_T| \leq 3\tau_n$. Let G_{TF} be the subgraph induced by $V \backslash V_T$, i.e. the triangle-free subgraph, which excludes the vertices in V_T and their incident edges.

For the triangle-free subgraph G_{TF}, we have that the situation around each 1-clique can be modelled using the analysis illustrated in Figure 2. Therefore, the fair random walk argument remains valid for the triangle-free "segments" between triangles.

Let $\vartheta'_n(G)$ be the number of cliques used by IG when triangle-free subgraphs are already covered optimally after $\mathcal{O}(n^5m^2)$ time in expectation, based on the arguments of Theorem 1, and let $\vartheta_n(G_{TF})$ be the clique covering number of the triangle-free subgraph G_{TF}. For the number of cliques used by IG, we have that $\vartheta'_n(G) \leq \vartheta_n(G_{TF}) + 3\tau_n$, since G_{TF} is covered optimally. The clique covering number $\vartheta_n(G)$ satisfies $\vartheta_n(G) \geq \vartheta_n(G_{TF})$. Therefore, the achieved approximation ratio is upper bounded by:

$$\frac{\vartheta_n(G_{TF}) + 3\tau_n}{\vartheta_n(G_{TF})} = 1 + 3\frac{\tau_n}{\vartheta_n(G_{TF})} \leq 1 + 3\frac{\tau_n}{(n - 3\tau_n)/2} = 1 + 6\frac{\tau_n}{n - 3\tau_n} \tag{4}$$

where the fact that $(n - 3\tau_n)/2 \leq \vartheta_n(G_{TF})$ is implied by Lemma 4 and $\omega(G_{TF}) = 2$, since G_{TF} is triangle-free. □

6 RESULT FOR BARABÁSI-ALBERT MODEL OF SCALE-FREE NETWORKS

At this point, we relate the previous result to models of real-world complex networks. Complex networks are networks with non-trivial structure. This structure is closely related to the process of their evolution. Complex networks are often statistically characterised by their degree distribution $P(k)$, which denotes the fraction of vertices, which have degree k. Many real-world networks are believed to be *scale-free*, which means that their degree distribution follows the power law, i.e. $P(k) \sim ck^{-\gamma}$, where γ is a coefficient of steepness of the distribution and c is a suitable constant.

Therefore, scale-free networks contain many vertices with low degree but also several vertices with very high degree. In real-world networks, it usually holds that $\gamma \in [2,3]$ [1].

One of the most famous models used to explain the process of evolution of scale-free networks is the *Barabási-Albert (BA) model* [4]. Its pseudocode is given in Algorithm 2.

```
1   begin with a connected seed graph $G_0 = [V_0, E_0]$
2   for $t = (n_0 + 1) \dots n$
3       $V_t = V_{t-1} \cup \{v_t\}$
4       attach $v_t$ to vertices from $V_{t-1}$ based on preferential attachment rule
```

Algorithm 2. Barabási-Albert (BA) model of scale-free networks [4]

In BA model, we begin with a connected seed graph on n_0 vertices and m_0 edges. Then, at each time step t, one new vertex comes and brings w new edges to the network, where w is a parameter of the model, which remains constant over time. These edges are attached to the existing vertices preferentially, i.e., the probability of attachment to vertex v is $\frac{(\deg(v))_t}{2m_t}$, where $(\deg(v))_t$ is the degree of v in time step t and m_t is the number of all edges at this time step. In the context of social networks, this can be interpreted in the way that a person with a larger number of contacts is more likely to get a new contact. It is known that BA model generates networks with degree distributions, which follow the power law in form $P(k) \sim ck^{-3}$, i.e. $\gamma = 3$ [4].

Lemma 5. In BA model with w incoming edges per vertex and with a seed graph with maximum clique size at most $w+1$, the maximum clique number ω_n satisfies $\omega_n \leq w+1$ for any n.

Proof. We prove this by contradiction. Suppose that $\omega_n > w+1$. Then, the last vertex of the maximum clique must have been attached to at least $w+1$ other vertices. This contradicts the fact that we have w incoming edges per vertex. □

Lemma 6. Suppose that the seed graph for BA model is a tree. If $w = 1$, then the resulting graph will also be a tree.

Proof. From Lemma 5, we have that the maximum clique number $\omega \leq 2$, i.e., it will be triangle-free. For generation of a cycle, one would have to have at least two incoming edges for the last vertex, which "closes" the cycle. Hence, the resulting graph will be connected and acyclic, i.e., it will be a tree. □

Corollary 1. Let G be a graph on n vertices generated by BA model with 1 incoming edge per vertex and with a tree as a seed graph. Then, IG finds the optimal clique covering for G in polynomial time.

The previous results are relatively straightforward. It is more interesting to see how good solution IG produces for BA model with $w \geq 2$. We first recall a classical result on the number of triangles in BA model in Lemma 7 and Theorem 4 applies it to show that the approximation achieved by IG is asymptotically optimal.

Lemma 7 ([7])**.** Let $w \geq 1$ be fixed. The expected number of triangles in a graph on n vertices generated by BA model with w incoming edges per vertex is given by:

$$(1 + o(1))\frac{w(w-1)(w+1)}{48}(\log n)^3 \tag{5}$$

as $n \to \infty$.

Theorem 4. Let G be a graph on n vertices generated by BA model with a triangle-free seed graph and an arbitrary number w of incoming edges per vertex. Then, IG achieves approximation ratio $1 + \mathcal{O}\left(\frac{(\log n)^3}{n}\right)$ for G in expected polynomial time.

Proof. Based on Lemma 7, we have that the number of triangles $\tau_n = \mathcal{O}((\log n)^3)$. The triangle-free seed graph assures that this upper bound also holds for small n. Theorem 3 implies that IG achieves approximation ratio:

$$1 + 6\frac{\mathcal{O}((\log n)^3)}{n - \mathcal{O}((\log n)^3)} = 1 + \mathcal{O}\left(\frac{(\log n)^3}{n}\right) \tag{6}$$

in expected polynomial time. □

It is worth mentioning that this result is similar to the result of evolutionary algorithms in the NP-hard makespan scheduling problem, where asymptotically vanishing discrepancies in the obtained solutions were proven [40]. However, Theorem 3 cannot be applied to graphs with linear or superlinear numbers of triangles, which may be encountered in other network models [15]. In the next section, we investigate the impact of non-overlapping triangles on the design of a suitable algorithm for CCP.

7 RESULT FOR GRAPHS WITH NON-OVERLAPPING TRIANGLES

The previous results were mostly positive. However, Lemma 3 has also outlined the limitations of IG. At this point, we investigate the behaviour of IG for graphs with non-overlapping triangles. Consider the initial permutation being generated such that non-overlapping triangles are placed into it as blocks. The rest of the permutation is generated uniformly at random. In the following, we show that such a modification of IG guarantees that the optimal clique covering is found in expected polynomial time.

Lemma 8. Let G be a graph with maximum clique size $\omega = 3$. Let S be an optimum and let S' be a suboptimum for CCP in G. There are three cases of how IG can overestimate $\vartheta(G)$:

Case 1. Instead of a 2-clique in S, there are two 1-cliques in S',

Case 2. Instead of a triangle in S, there is a 2-clique and a 1-clique in S',

Case 3. Instead of two triangles in S, there are three 2-cliques in S'.

All possible ways of overestimation represented by S' represent compositions of these three cases.

Proof. Graphs and clique coverings generated by GCC, where the first two cases can occur, are common and can be found very easily. The existence of the third case is proven by Lemma 3. To exclude the existence of other ways, we use simple enumeration.

- Substitution of any number of 2-cliques by 1-cliques is a composition of events included in Case 1.
- Substitution of one triangle by three 1-cliques is a composition of Case 1 and Case 2.
- If we consider three triangles, the first two can be substituted based on Case 2 or Case 3 and the last triangle will remain for Case 2.
- If we consider four or more triangles, we can apply Case 2 and Case 3 iteratively. The resulting 2-cliques are further divided according to Case 1.

□

Lemma 9. Let G be a graph with maximum clique size $\omega = 3$. If there is a sub-optimal clique covering S' of G, which contains more triangles than an optimum S, then S' must also contain at least two 1-cliques.

Proof. Let c_1, c_2 and c_3 be the numbers of cliques in S with 1, 2 or 3 vertices, respectively. Let c'_1, c'_2 and $c'_3 = c_3 + d$ be the respective values for S, and for $d \geq 1$. All vertices must be covered and S must contain less cliques than S'. Hence:

$$c_1 + 2c_2 + 3c_3 = c'_1 + 2c'_2 + 3(c_3 + d) = n, \tag{7}$$

$$c_1 + c_2 + c_3 < c'_1 + c'_2 + c_3 + d.$$

From these formulas, $3d = (c_1 - c'_1) + 2(c_2 - c'_2)$ and $d > (c_1 - c'_1) + (c_2 - c'_2)$. This implies that $(c_2 - c'_2) > 2d$ and, thus, $(c_1 - c'_1) < -d$. The value $-d$ can be at most -1, i.e. $c'_1 > 1$. □

We now split the number of 1-cliques into two values. Let an optimum S contain c_1 1-cliques. We will call this the number of *free* 1-*cliques*. If a suboptimum S' contains c'_1 1-cliques, then $(c'_1 - c_1)$ is the number of *extra* 1-*cliques*. The idea now is to model the process as the minimisation of the number of extra 1-cliques, rather than the number of all cliques in the covering.

Lemma 10. Let G be a graph with maximum clique size $\omega = 3$. Let IG begin with a suboptimal clique covering S' with at least as many triangles as in an optimal clique covering S for G. Then, IG cannot get stuck in a local optimum and will be in the global optimum if the number of extra 1-cliques in the solution is minimal.

Proof. Let S contain c_1 1-cliques, c_2 2-cliques and c_3 triangles. The analogous values for S' are c'_1, c'_2 and $c'_3 \geq c_3$. The premises imply that an improvement cannot be obtained by making the number of triangles higher. Therefore, it is necessary that $c'_2 < c_2$ and $c'_1 > c_1$. Since c_1 1-cliques are present is S, the only way to obtain an improvement is to reduce the number of extra 1-cliques by 2 and increase the number of 2-cliques by 1. This holds for all suboptima, which proves the second statement.

For the first statement, suppose that IG got stuck. Then, by Lemma 8, two of the triangles must have been substituted by three 2-cliques. However, this is in contradiction with the fact that these three 2-cliques must lie between the triangles and *block_jump* cannot cause a transformation, in which vertices between 2 different blocks are regrouped to 3 blocks in between. □

Theorem 5. Let G be a graph on n vertices with maximum clique size $\omega = 3$, containing only non-overlapping triangles. Let P be the initial permutation for IG, constructed by placing the triangles into P as blocks first and the rest of vertices are placed into P uniformly at random. Then, IG will find the optimal solution in $\mathcal{O}(n^5m^2)$ time in expectation.

Proof. Based on Lemma 9, the initial solution must be a global optimum or it contains a 1-clique. Suppose that it is a suboptimum. Lemma 10 implies that the following process is a minimisation of the number of extra 1-cliques and getting stuck in local optima is avoided.

In each time step, we have a situation, in which a 1-clique is stuck between 2-cliques and triangles, similarly to Figure 2. The probability of moving towards each direction is naturally determined by which block comes first. This is not influenced by the fact that we can have triangles. We have to examine two cases, depicted by Figure 4.

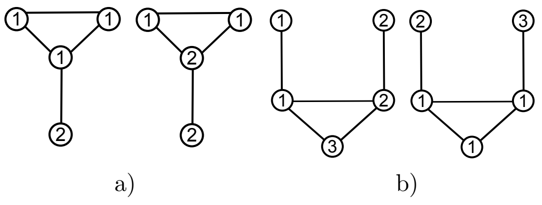

Figure 4. Illustration of the cases for the non-overlapping triangles for the proof of Theorem 5. Case a) represents a 1-clique, which can freely emerge both in suboptima and optima. Case b) illustrates a situation, when 1-clique performs a random walk by "jumping" over the triangle.

Case 1. If the 1-clique is not in a triangle, it can be surrounded by 2-cliques or triangles, to which it can be connected only by a single edge (since it is in no triangle). The 2-clique case is handled by the arguments from Theorem 1. In

Figure 4 a), we depict the situation, when it is adjacent to a vertex in a triangle block. Such a 1-clique can be freely enhanced to a 2-clique and reduced back to 1-clique afterwards. Such a transformation can occur between two optima, which shows that such a clique does not contribute to the number of extra 1-cliques.

Case 2. In this case, the 1-clique is in a single triangle. In this case, the other vertices of the triangle must be separated into different cliques. Since they cannot be in a triangle, they must each be in its own 2-clique, as shown by Figure 4 b). When *block_jump* is applied to the 1-clique, this 1-clique is transformed into the triangle, and a new 1-clique can emerge on the opposite side of the triangle. This position depends on the ordering of cliques on the other side, for which the probability is uniformly distributed, leading to validity of the fair random walk argument.

In each suboptimal solution, we have that the number of extra 1-cliques is at least 2. Therefore, by applying the same cover time arguments as in Theorem 1, we have that the expected time to obtain the optimum is upper bounded by $\mathcal{O}(n^5m^2)$. □

Even though the assumption of non-overlapping triangles is still strong, it gives us some insight into the impact of triangles on the problem structure and design of suitable algorithms. We hope that these results may pave the way to more sophisticated analyses of heuristics for CCP, as well as other combinatorial optimisation problems for different models of complex networks and practically relevant scenarios.

8 CONCLUSIONS

We presented an analysis of an *iterated greedy* (IG) heuristic for the vertex *clique covering problem* (CCP) in several practically relevant graph classes. As our analytical results indicate, IG can be viewed as a variant of local search, with non-trivial methods needed to quantify the convergence and runtime properties of this randomised search heuristic.

The classes of graphs concerned include *triangle-free graphs*, *sparse random graphs*, scale-free networks generated by *Barabási-Albert (BA) model*, and *graphs with non-overlapping triangles.*

We have shown that for triangle-free graphs, IG finds the optimum in expected polynomial time. For sparse random graphs generated by the *Erdős-Rényi model* in its form $\mathbb{G}(n, c/n)$, where c/n is the probability of edge generation, we have shown that IG finds the optimum in expected polynomial time with high probability if $c < 1$.

For BA model, we have shown that IG achieves approximation ratio $1 + \mathcal{O}\left(\frac{(\log n)^3}{n}\right)$ in expected polynomial time.

Last but not least, we have shown that for graphs with non-overlapping triangles, putting the triangles in the initial permutation for IG as blocks helps to improve the worst-case performance of IG from getting stuck with probability $1 - o(1)$ to finding the optimum in expected polynomial time.

We believe that these results provide a valuable insight into the behaviour of heuristics, which combine ideas of classical greedy algorithms with randomised iterative improvement processes. This insight may represent a foundation of analysis for other graph classes, as well as for other problems such as graph colouring [13, 37], independent sets [11], or other similar algorithms such as the greedy randomised adaptive search procedures (GRASP) [17].

Acknowledgement

This contribution was supported by Grant Agency VEGA SR under the grant 1/0145/18.

REFERENCES

[1] Albert, R.—Barabási, A.-L.: Statistical Mechanics of Complex Networks. Reviews of Modern Physics, Vol. 74, 2002, No. 1, pp. 47–97, doi: 10.1103/RevModPhys.74.47.

[2] Aleliunas, R.—Karp, R. M.—Lipton, R. J.—Lovasz, L.—Rackoff, C.: Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems. Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS '79), 1979, pp. 218–223, doi: 10.1109/SFCS.1979.34.

[3] Auger, A.—Doerr, B. (Eds.): Theory of Randomized Search Heuristics. World Scientific, Series on Theoretical Computer Science, Vol. 1, 2011.

[4] Barabási, A.-L.—Albert, R.: Emergence of Scaling in Random Networks. Science, Vol. 286, 1999, No. 5439, pp. 509–512, doi: 10.1126/science.286.5439.509.

[5] Behrisch, M.—Taraz, A.: Efficiently Covering Complex Networks with Cliques of Similar Vertices. Theoretical Computer Science, Vol. 355, 2006, No. 1, pp. 37–47, doi: 10.1016/j.tcs.2005.12.005.

[6] Bollobás, B.: Random Graphs. Cambridge University Press, Cambridge, 2001, doi: 10.1017/CBO9780511814068.

[7] Bollobás, B.—Riordan, O. M.: Mathematical Results on Scale-Free Random Graphs. In: Bornholdt, S., Schuster, H. G. (Eds.): Handbook of Graphs and Networks. Wiley, 2005, pp. 1–34.

[8] Cacetta, L.—Purwanto, P.: Deficiencies and Vertex Clique Covering Numbers of a Family of Trees. Australasian Journal of Combinatorics, Vol. 1, 1990, pp. 15–27.

[9] Chakrabarti, D.—Faloutsos, C.: Graph Mining: Laws, Generators, and Algorithms. ACM Computing Surveys, Vol. 38, 2006, No. 1, Article No. 2.

[10] Chalupa, D.: An Analytical Investigation of Block-Based Mutation Operators for Order-Based Stochastic Clique Covering Algorithms. In: Blum, C., Alba, E. (Eds.): Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13). ACM, 2013, pp. 495–502, doi: 10.1145/2463372.2463436.

[11] Chalupa, D.: Construction of Near-Optimal Vertex Clique Covering for Real-World Networks. Computing and Informatics, Vol. 34, 2015, No. 6, pp. 1397–1417.

[12] Chalupa, D.: On Combinatorial Optimisation in Analysis of Protein-Protein Interaction and Protein Folding Networks. In: Squillero, G., Burelli, P. (Eds.): Proceedings of the 19th European Conference on Applications of Evolutionary Computation (EvoApplications 2016). Springer, Lecture Notes in Computer Science, Vol. 9597, 2016, pp. 91–105, doi: 10.1007/978-3-319-31204-0_7.

[13] Culberson, J. C.—Luo, F.: Exploring the k-Colorable Landscape with Iterated Greedy. In: Johnson, D. S., Trick, M. (Eds.): Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge. American Mathematical Society, 1995, pp. 245–284.

[14] Czech, W.—Dzwinel, W.—Goryczka, S.—Arodz, T.—Dudek, A. Z.: Exploring Complex Networks with Graph Investigator Research Application. Computing and Informatics, Vol. 30, 2011, No. 2, pp. 381–410.

[15] Dorogovtsev, S.—Mendes, J. F. F.: Evolution of Networks. Advances in Physics, Vol. 51, 2002, No. 4, pp. 1079–1187, doi: 10.1080/00018730110112519.

[16] Erdős, P.—A. Rényi: On Random Graphs. Publicationes Mathematicae Debrecen, Vol. 6, 1959, pp. 290–297.

[17] Feo, T. A.—Resende, M. G. C.: Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization, Vol. 6, 1995, No. 2, pp. 109-133.

[18] Friedrich, T.—He, J.—Hebbinghaus, N.—Neumann, F.— Witt, C.: Analyses of Simple Hybrid Algorithms for the Vertex Cover Problem. Evolutionary Computation, Vol. 17, 2009, No. 1, pp. 3–19, doi: 10.1162/evco.2009.17.1.3.

[19] Garey, M. R.—Johnson, D. S.: Computers and Intractability. Series of Books in the Mathematical Sciences, Vol. 29, W. H. Freeman, New York, 2002.

[20] Giel, O.—Wegener, I.: Evolutionary Algorithms and the Maximum Matching Problem. In: Alt, H., Habib, M. (Eds.): Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS '03). Springer, Lecture Notes in Computer Science, Vol. 2607, 2003, pp. 415–426.

[21] Girvan, M.—Newman, M. E. J.: Community Structure in Social and Biological Networks. Proceedings of the National Academy of Sciences of the United States of America, Vol. 99, 2002, No. 12, pp. 7821–7826, doi: 10.1073/pnas.122653799.

[22] Gramm, J.—Guo, J.—Hüffner, F.—Niedermeier, R.: Data Reduction and Exact Algorithms for Clique Cover. Journal of Experimental Algorithmics, Vol. 13, 2009, Article No. 2, doi: 10.1145/1412228.1412236.

[23] Halldórsson, M. M.: A Still Better Performance Guarantee for Approximate Graph Coloring. Information Processing Letters, Vol. 45, 1993, No. 1, pp. 19–23, doi: 10.1016/0020-0190(93)90246-6.

[24] Karp, R. M.: Reducibility Among Combinatorial Problems. In: Miller, R., Thatcher, J., Bohlinger, J. D. (Eds.): Proceedings of a Symposium on the Complexity of Computer Computations. Plenum Press, 1972, pp. 85–103, doi: 10.1007/978-1-4684-2001-2_9.

[25] Keil, J. M.—Stewart, L.: Approximating the Minimum Clique Cover and Other Hard Problems in Subtree Filament Graphs. Discrete Applied Mathematics, Vol. 154, 2006, No. 14, pp. 1983–1995.

[26] Kratsch, S.—Neumann, F.: Fixed-Parameter Evolutionary Algorithms and the Vertex Cover Problem. Algorithmica, Vol. 65, 2013, No. 4, pp. 754–771, doi: 10.1007/s00453-012-9660-4.

[27] Lehre, P. K.: Fitness-Levels for Non-Elitist Populations. In: Krasnogor, N., Lanzi, P. L. (Eds.): Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11). ACM, 2011, pp. 2075–2082.

[28] Leskovec, J.—Lang, K. J.—Dasgupta, A.—Mahoney, M. W.: Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. Internet Mathematics, Vol. 6, 2009, No. 1, pp. 29–123, doi: 10.1080/15427951.2009.10129177.

[29] Náther, P.—Markošová, M.: Positional Word Web and Its Numerical and Analytical Studies. Computing and Informatics, Vol. 30, 2011, No. 6, pp. 1287–1302.

[30] Neumann, F.: Expected Runtimes of Evolutionary Algorithms for the Eulerian Cycle Problem. Computers and Operations Research, Vol. 35, 2008, No. 9, pp. 2750–2759, doi: 10.1016/j.cor.2006.12.009.

[31] Neumann, F.—Wegener, I.: Randomized Local Search, Evolutionary Algorithms, and the Minimum Spanning Tree Problem. Theoretical Computer Science, Vol. 378, 2007, No. 1, pp. 32–40, doi: 10.1016/j.tcs.2006.11.002.

[32] Neumann, F.—Witt, C.: Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity. Springer, 2010.

[33] Ruiz, R.—T. Stützle: A Simple and Effective Iterated Greedy Algorithm for the Permutation Flowshop Scheduling Problem. European Journal of Operational Research, Vol. 177, 2007, No. 3, pp. 2033–2049.

[34] Schaeffer, S. E.: Graph Clustering. Computer Science Review, Vol. 1, 2007, No. 1, pp. 27–64, doi: 10.1016/j.cosrev.2007.05.001.

[35] Stanimirović, Z.—Mišković, S.: A Hybrid Evolutionary Algorithm for Efficient Exploration of Online Social Networks. Computing and Informatics, Vol. 33, 2014, No. 2, pp. 410–430.

[36] Sudholt, D.: A New Method for Lower Bounds on the Running Time of Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation, Vol. 17, 2013, No. 3, pp. 418–435, doi: 10.1109/TEVC.2012.2202241.

[37] Sudholt, D.—Zarges, C.: Analysis of an Iterated Local Search Algorithm for Vertex Coloring. In: Cheong, O., Chwa, K. Y., Park, K. (Eds.): Algorithms and Computation (ISAAC 2010). Springer, Lecture Notes in Computer Science, Vol. 6506, 2011, pp. 340–352.

[38] Sutton, A. M.—Neumann, F.: A Parameterized Runtime Analysis of Evolutionary Algorithms for the Euclidean Traveling Salesperson Problem. Proceedings of the 26th Conference on Artificial Intelligence (AAAI-12), AAAI Press, 2012, pp. 1105–1111.

[39] Welsh, D. J. A.—Powell, M. B.: An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems. The Computer Journal, Vol. 10, 1967, No. 1, pp. 85–86.

[40] Witt, C.: Worst-Case and Average-Case Approximations by Simple Randomized Search Heuristics. In: Diekert, V., Durand, B. (Eds.): Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS 2005). Springer, Lecture Notes in Computer Science, Vol. 3404, 2005, pp. 44–56, doi: 10.1007/978-3-540-31856-9_4.

[41] Witt, C.: Analysis of an Iterated Local Search Algorithm for Vertex Cover in Sparse Random Graphs. Theoretical Computer Science, Vol. 425, 2012, pp. 117–125, doi: 10.1016/j.tcs.2011.01.010.

[42] Yuan, Z.—Fügenschuh, A.—Homfeld, H.—Balaprakash, P.—Stützle, T.— Schoch, M.: Iterated Greedy Algorithms for a Real-World Cyclic Train Scheduling Problem. In: Blesa, M. J., Blum, C., Cotta, C., Fernández, A. J., Gallardo, J. E., Roli, A., Sampels, M. (Eds.): Proceedings of the 5th International Workshop on Hybrid Metaheuristics (HM 2008). Springer, Lecture Notes in Computer Science, Vol. 5296, 2008, pp. 102–116.

David Chalupa received his Master's degree in software engineering and his Ph.D. degree in applied informatics from the Slovak University of Technology in Bratislava, Slovakia, in 2011 and 2014, respectively. He is currently Postdoctoral Fellow in the Operations Research Group at the Aalborg University in Denmark. Prior to that he was with the Computational Science Research Group (CSRG) at the University of Hull in the United Kingdom. His research interests include heuristics and metaheuristics, combinatorial optimisation and complex networks.

Jiří Pospíchal received his diploma degree in physical chemistry from the University of Jan Evangelista Purkyně in Brno, Czech Republic in 1984, and his Ph.D. degree in chemistry from Faculty of Chemical and Food Technologies at the Slovak University of Technology, Bratislava, in 1990. From quantum chemistry and computer assisted organic synthesis he soon transferred his interests to computer science and is now Professor of applied informatics at the Faculty of Natural Sciences at the University of Ss. Cyril and Methodius in Trnava, Slovakia. His research interests are evolutionary algorithms, artificial intelligence, neural networks, and graph theory.