Variable neighborhood search for partitioning sparse biological networks into the maximum edge-weighted k-plexes

Milana Grbić, Aleksandar Kartelj, Savka Janković, Dragan Matić, Vladimir Filipović

Abstract

In a network, a k-plex represents a subset of n vertices where the degree of each vertex in the subnetwork induced by this subset is at least n-k. The maximum edge-weight k-plex partitioning problem (Max-EkPP) is to find the k-plex partitioning in edge-weighted network, such that the sum of edge weights is maximal. The Max-EkPP has an important role in discovering new information in large sparse biological networks. We propose a variable neighborhood search (VNS) algorithm for solving Max-EkPP. The VNS implements a local search based on the 1-swap first improvement strategy and the objective function that takes into account the degree of every vertex in each partition. The objective function favors feasible solutions, also enabling a gradual increase in terms of objective function value when moving from slightly infeasible to barely feasible solutions. A comprehensive experimental computation is performed on real metabolic networks and other benchmark instances from literature. Comparing to the integer linear programming method from literature, our approach succeeds to find all known optimal solutions. For all other instances, the VNS either reaches previous best known solution or improves it. The proposed VNS is also tested on a large-scaled dataset which was not previously considered in literature.

Keywords: k-plex partitioning; variable neighborhood search; network clustering, biochemical networks

1. Introduction

In recent years there is an increasing effort to provide algorithms for better understanding of biological structures and processes. Among many other approaches, partitioning large biological

Email addresses: milana.grbic@pmf.unibl.org (Milana Grbić), aleksandar.kartelj@gmail.com (Aleksandar Kartelj), savka.jankovic@pmf.unibl.org (Savka Janković), dragan.matic@pmf.unibl.org (Dragan Matić), vladofilipovic@hotmail.com (Vladimir Filipović)

networks into smaller clusters or functional modules is a commonly used technique for discovering new properties and functionalities of a particular structure. In this work, we deal with the partitioning of the edge-weighted networks into k-plex components, where a subset of some n vertices in a network is a k-plex if the degree of each vertex in the subnetwork induced by this subset is at least n - k. The aim of the maximum edge-weight k-plex partitioning problem (Max-EkPP) is to find the k-plex partitioning with the maximal total weight of edges.

Partitioning networks into high density subnetworks, especially cliques, has already been proven as a useful technique for obtaining new information in understanding complicated relations between biological elements. For example, partitioning in protein threading analysis - constrained threading problem can be reduced on maximum edge weight clique problem [1]. The protein side chain packing problem is transformed into a problem of finding a maximum weight clique. The edge weighting function is defined in a way that reflects the frequency of contact pairs in a database of proteins [2]. Finding cliques is also one of the methods for identification of the clusters that are later divided into protein complexes and dynamic functional modules. By analyzing the multibody structure of the network of protein–protein interactions (PPI), molecular modules that are densely connected within themselves, but sparsely connected with the rest of the network, are discovered [3]. Cliques have a similar use in modular decomposition of PPI networks. This decomposition allows to combine proteins into the actual functional complexes by identifying groups of proteins acting as a single unit [4].

On the other hand, a number of biological networks classes contain only sparse networks. Dealing with such networks, partitioning into cliques can be too restrictive method, so many potentially useful information about the interference of biological objects can be neglected. Therefore, clique relaxation approaches could be even more useful. In the approach presented in this work, partitioning is followed by the principle that the objects in each partition are still highly connected in a particular way, but not so restrictively to form a clique. By relaxing cliques to sparse graphs, biological objects become connected in semantically or functionally logical groups which we call k-plexes, having in mind that the total sum of weights in all partitions should be as large as possible.

1.1. Problem definition

Let a network be denoted as G = (V, E), where $V = \{1, 2, ..., n\}$ is the set of nodes and $E \subset V \times V$ is the set of edges. With uv we simply denote the edge $\{u, v\} \in E$. With real numbers $w_{uv} > 0$ we

denote the weight of the edge connecting nodes u and v. We call u and v the end-vertices of the edge uv.

Let $k \geq 1$ be an integer. A set of nodes S is called k-plex if the degree of each node in the sub-network induced by S is at least n-k. The weight of a k-plex is the sum of all its edge weights. The weight of the whole partition is the sum of the weights of all its k-plex components. The maximum k-plex partitioning (Max-EkPP) problem is then defined as finding such a partition of G which is of the maximum total weight and each component is a k-plex. If k = 1, the k-plex is a clique and the Max-EkPP is brought down to the maximum edge-weight clique partitioning problem (Max-ECP).

1.2. Our contributions

The contributions of this paper can be summarized as follows:

We constructed the first heuristic method for partitioning graph into densely connected components - k-plexes. The proposed method is based on the variable neighborhood search (VNS) metaheuristic.

The proposed VNS method implements a newly created objective function which takes into consideration the degree of every vertex in each k-plex. Objective function favors feasible solutions over infeasible ones as expected, but also enables a gradual increase in terms of objective function value when moving from slightly infeasible to barely feasible solutions.

We successfully applied the proposed VNS on some biological instances, as well as on some graph instances used in similar NP-hard problems.

The quality of the proposed method is proven by the fact that it achieves all previously known optimal solutions. It also proposes new lower bounds for the instances for which the optimal solutions are unknown.

Based on the obtained computational results and graphical interpretation of the solutions, we established a biological interpretation. This indicates the potential of using the proposed method in discovering new biological information of a particular structure.

The remainder of the paper is organized as follows. In the next section, we review some related researches. In Section 3 we describe the proposed VNS method. We present the computational results, comparison with an existing method and biological evaluation of the obtained data in Section 4. The last section concludes the paper and suggests the future work.

2. Literature review

A k-plex structure was introduced in an early work [5] in late 1970s, as a clique-like structure of variable strength. The k-plex structure is defined as a graph with n vertices, where each vertex is connected by at least n-k other vertices. The optimization problem which has arisen involves the identification of the k-plex of the maximum cardinality in an unweighted sparse graph. The problem is called the maximum k-plex problem (Max-kP problem).

Although one could expect that this formulation was immediately tackled by researchers, the problem has not been thoroughly analyzed for more than 30 years. Meanwhile, a progress in the Internet and other computer-based technologies, including bioinformatics, triggered the generation of tremendous amounts of various interaction data. Balasundaram et al. [6] brought the mentioned problem again into the attention of the scientific community, by recognizing its close connection with behavior of some real-world networks, particularly with social networks. In the mentioned paper, the problem of identification of a maximum cardinality k-plex in an unweighted sparse graph (Max-kP) has been proven to be NP hard and an integer programming formulation (ILP) has been presented. ILP formulations developed for Maximum clique problem in [7] can be adopted for solving other related problems, including maximum size k-plex problem. Beside these exact methods based on the integer linear programming approach, there are some heuristic methods for solving Max-kP in the unweighted sparse graphs. For instance, McClosky and Hicks [8] adapted combinatorial clique algorithms to find maximum k-plexes and proposed a new upper bound on the cardinality of k-plexes. Moser et al. [9] proposed some practical algorithms for finding maximum k-plexes which outperforms other approaches. k-plex clustering is also a way of non-hierarchical decomposition of the graph into clusters, which enables an application of parallelization algorithms.

Several other variants of clique relaxations, as well as adequate mathematical programming formulations have been studied in [10]. In large biological networks, for example PPI network, proteins with similar GO annotation can be clustered together by partitioning biological networks into highly connected components [11]. Clustering large data sets plays an important role in gene expression analysis. In [12], cluster analysis of cDNA fingerprints is used to identify clones corresponding to the same gene. In [13], many near-optimal clusterings are used to explore the dynamics of network clusterings. This is later applied on several biological and other networks. In order to show the types of insights that can be extracted from large collections of near-optimal solutions, the authors analyzed the ERK1/ERK2 mitogen-activated protein kinase (MAPK18) signal-transduction

pathway and a network of cortical-cortical connections in the human brain.

Identifying cohesive subgroups (not necessarily cliques and k-plexes) has also been performed in a number of non-biological networks: in studying terrorist and other criminal networks [14], web graphs [15], wireless networks [16], in finding structural patterns embedded within social network data [17], text mining [18], stock markets [19], etc.

A complementary problem to identifying k-plexes in the graph G is a problem of identifying co-k-plexes in \overline{G} . A subset S of a graph G is a co-k-plex if the subgraph induced by S has a maximum degree of k-1 or less. From this definition, one can conclude that S is a co-k-plex in G if and only if S is a k-plex in the complement graph \overline{G} . For k=1, we get that the co-k-plex is exactly the independent set. Such a relaxation of the maximum clique problem is also in a close connection with defective coloring problem [20, 21], which is a relaxation of the well known vertex coloring problem in a graph. A (κ, d) -coloring of a graph is a coloring of the vertices with κ colors such that no vertex is adjacent to more than d vertices of its same color. For d=0, (κ, d) -coloring is the proper vertex coloring problem. For a given number d, identifying appropriate co-d-1-plexes corresponds to the (κ, d) - coloring, where vertices in a co-d-1-plex are allowed to be colored with the same color.

In the context of partitioning weighted graphs into distinct components, in the literature one can find strong results addressing the maximum edge-weight cliques partitioning (Max-ECP) problem. In Max-ECP, the objective is to cluster all the vertices into disjoint cliques, such that the total sum of the edge weights of all partitions is as large as possible. Although Max-ECP is a special case of Max-EkPP for k=1, it has been considered on complete graphs, both in several earlier works [22, 23, 24, 25], as well as in recent state-of-the-art proposed heuristic methods [26, 27]. From the other hand, partitioning of sparse graph into cliques could be too restrictive, since many useful information regarding relations between the elements can be lost. Following that consideration, Martins [28] proposed a polynomial size integer linear programming formulation for Max-EkPP problem, also considering the inclusion of additional topological constraints in the model. The performance of the proposed ILP model has been tested on biological and artificial networks, which we also used in our paper.

3. Variable neighborhood search for Max-EkPP

Variable Neighborhood Search (VNS) algorithm is a robust metaheuristic introduced by Mladenović and Hansen [29]. The main searching principle of a VNS is based on the empirical evidences: (a) multiple local optima are correlated in some sense (usually close to each other) and (b) a local optimum found in one neighborhood structure is not necessarily a local optimum for some other neighborhood structure.

The overall structure of the VNS algorithm for Max-kP is shown on the Figure 1.

```
input: n_{min}, n_{max}, it_{max}, itrep_{max}, t_{max}, prob, k
     output: x
  \mathbf{1} \ \mathbf{x} \leftarrow \mathrm{initializeSolution}();
  n \leftarrow n_{min};
 sit \leftarrow 1;
 4 while it < it_{max} \land (it - it_{lastimpr}) < itrep_{max} \land t_{run} < t_{max} do
         \mathbf{x}' \leftarrow \operatorname{shaking}(\mathbf{x}, n);
          \mathbf{x}'' \leftarrow \text{localSearch}(\mathbf{x}', k);
          move \leftarrow \text{shouldMove}(\mathbf{x}, \mathbf{x}'', prob);
         if move then
         else if n < n_{max} then
10
           n \leftarrow n + 1;
11
          else
12
13
           n \leftarrow n_{min};
         i\dot{t} \leftarrow it + 1;
14
15 end
16 return x:
```

Figure 1: The overall structure of the VNS

The input of the VNS algorithm, beside the given input graph G, consists of:

- n_{min} and n_{max} minimal and maximal VNS neighborhood structure size;
- it_{max} , $itrep_{max}$, t_{max} maximal number of total iterations , maximal number of iterations without improvement, and maximal execution time in seconds, respectively;
- prob probability to move to the other solution of the same quality;
- k integer value that corresponds to the value of k inside term Max-kP.

VNS algorithm usually imposes two main procedures: shaking and local search (LS). Shaking procedure manages the system of the neighborhoods and in each iteration it randomly selects a new point from the current neighborhood. The main purpose of the shaking procedure is to resolve

situations when local search procedure is stuck into a local suboptimal solution. More details about shaking are given in Section 3.2.

Inside the LS procedure, the algorithm is trying to improve the solution selected by the shaking. LS systematically checks other solutions in its nearest neighborhood. Details about LS are given in Section 3.3.

In the main loop of the algorithm, shaking procedure is iteratively called, until no further improvements of the best solution can be made inside the current neighborhood. When that situation appears, the algorithm steps into the next neighborhood. When the last neighborhood n_{max} is explored, the search restarts at the first neighborhood n_{min} .

The execution of the VNS is stopped when either of the following conditions becomes satisfied: a maximum number of iterations is reached, a maximum number of iterations without any improvement of the current best solution is reached or a maximum allowed execution time is reached.

3.1. Solution representation and the objective function

Let G = (V, E) be an edge-weighted network and let k be the given integer number. Recall that the aim of the Max-EkPP is to find the k-plex partition with the maximum total weight.

The solution of the proposed VNS algorithm is represented by an integer array \mathbf{x} of the length |V|. Each element of the array corresponds to one vertex of the graph, denoting to which component (partition) the corresponding vertex is assigned. More precisely, the vertex i is assigned to the component V_j if $x_i = j$. The initial solution is created by assigning each element of the array \mathbf{x} random integer number from interval $[1, 2, ..., \sqrt{|V|}]$. The upper bound for the initial number of partitions $\sqrt{|V|}$ was empirically determined.

Unfeasible solutions are not implicitly disallowed by the representation. This is fortunate, since an objective function can direct the search to the more promising and feasible areas in a subtle way, without posing heavy penalty on a slightly infeasible solution. In other words, we constructed the objective function with two aims: to subtly avoid infeasible solutions and to maximize the total weight of the partition.

Let $(V_1, V_2, ..., V_l)$ be a (not necessarily feasible) solution of the Max-EkPP. Let w_{total} be the total sum of the weights of all edges in the network G, i.e. $w_{total} = \sum_{uv \in E} w_{uv}$.

We introduce the term "correct vertex" in a solution. A vertex $v \in V_j$, $j \in \{1, 2, ..., l\}$ is "correct" if the degree of v in the network induced by V_j is at least $|V_j| - k$. This means if each

vertex in a partition is "correct", then that partition is a k-plex. Let $correct_{total}$ be the total number of correct vertices in the solution.

In the partition $(V_1, V_2, ..., V_l)$, with w_{sol} we denote the total sum of the weights of all edges with correct end vertices.

Now we can define the objective function of the solution.

$$obj(V_1, V_2, ..., V_l) = correct_{total} + \frac{w_{sol}}{w_{total}}$$

$$\tag{1}$$

Since the value w_{sol}/w_{total} is always less or equal to 1, the objective function mainly depends on the first term, i.e. the total number of correct vertices in the solution. Consequently, the objective function of any feasible solution will be greater than the objective function of any infeasible solution. If two solutions have the same number of correct vertices, then the solution with greater total sum of the weights has also greater objective value. As a consequence, the maximization process discards solutions with many incorrect vertices and directs the search into the feasible regions.

At the same time, the proposed objective function properly orders infeasible solutions. This gives better infeasible solutions higher chance to appear, which can consequently, after local search, transform them into higher quality feasible solutions.

3.2. Shaking

The main purpose of the shaking procedure is to extend the search space of the current solution in order to reduce the possibility that the algorithm falls into suboptimal solutions.

Inside the shaking procedure, the algorithm creates a system of neighborhoods used for deriving new solutions based on the current best solution \mathbf{x} .

For defining the κ -th neighborhood we use the following procedure: some κ vertices from V are randomly chosen. For each chosen vertex v, the algorithm changes its component as follows.

If l is the total number of partitions, then an integer q is randomly chosen from the set $\{1, 2, ..., l+1\}$. If q < l+1, then the vertex v is moved to the existing partition V_q . If q = l+1, then a new singleton partition is established $(V_{l+1} = \{v\})$ and the total number of partitions is increased by one. If the old partition, from which the vertex v was chosen, becomes empty, then the total number of partitions is decreased by one. This strategy allows that the total number of partitions can be changed during the searching process.

Therefore, the aim of the proposed shaking procedure is twofold: it perturb vertices from one component to another and possibly increases or decreases the total number of partitions. The solution \mathbf{x}' , obtained by the shaking is the subject of the further improvements in the local search.

3.3. Local search

The purpose of the local search procedure is to explore the neighborhood of a new solution space obtained through shaking in order to achieve a locally optimal solution. The proposed LS is based

```
input: x', k
    output: x''
 1 \mathbf{x}'' \leftarrow \mathbf{x}';
 n \leftarrow |\mathbf{x''}|;
 3 impr \leftarrow true;
 4 while impr do
          impr \leftarrow false;
          i \leftarrow ir \leftarrow \text{random}(1,n);
               l \leftarrow \text{countDistinctValues}(\mathbf{x}'') + 1;
9
               p \leftarrow pr \leftarrow \text{random}(1,l);
               do
10
                     newObj \leftarrow \text{repositionObjectiveValue}(\mathbf{x}'', i, p, k);
11
12
                     if newObj > \mathbf{x}''.obj then
                          \mathbf{x}'' \leftarrow \text{reposition}(\mathbf{x}'', i, p, k);
13
14
                           impr \leftarrow true;
                          break;
15
                     p \leftarrow (p \!\!\mod l) + 1;
16
17
                while p \neq pr;
               \mathbf{if}\ impr\ \mathbf{then}
18
                break;
19
20
                  \leftarrow (i \mod n) + 1;
          while i \neq ir;
21
22 end
23 return x";
```

Figure 2: Local search

on the so called "1-swap first improvement" strategy (see Figure 2). Let \mathbf{x}' be a solution suggested by the shaking procedure and let \mathbf{x}'' be a solution which LS is afterwards applied on. The LS iteratively examines new solutions formed by moving a single vertex from its belonging component to some other component, in the following way. Let v be a vertex which is the subject of movement. If l is the total number of components, then a random integer p is chosen from $\{1, 2, ..., l+1\}$. If p < l+1, the vertex v is moved to the existing partition V_p , else a new partition $V_p = \{v\}$ is established. By this movement step, \mathbf{x}'' is adjusted in accordance to the partial calculation of the objective function (Figure 3). When the first improvement of the solution is found, the change is immediately applied and \mathbf{x}'' is updated. After the improvement, LS restarts the outer loop, i.e., it

tries to find new improvement by resetting the initial values of the selected vertex and candidate partition. If an improvement is still not found, LS continues with the next vertex, repeating the procedure. When the set of all vertices is exhausted without improvement, LS stops.

Since LS is usually the most time-consuming part in the entire VNS, it is of crucial importance to construct the LS procedure to be as much efficient as possible, considering also the quality of the obtained local optimum. Local search systematically checks neighborhood of the given solution by moving one vertex from its starting partition to some other partition. This infers only a slight difference in the underlying structure of the solution so it might be useful to partially calculate objective function of the newly formed solution. Figure 3 shows the pseudocode of this partial calculation. Current value of the objective function is composed of two terms: $correct_{total}$ represents the number of vertices that are correct, while w_{sol} is the sum of edges connecting correct vertices. Partial objective function calculates the new objective function value after moving the i^{th} vertex from its current partition to p^{th} partition. This movement induces the change in local solution structure since only vertices from the initial and target partition, and their related edges should be considered in the partial calculation. The set of these relevant vertices is denoted by $V_{relevant}$. For each vertex u in the set $V_{relevant}$, value of $correct_{total}$ is adjusted. For example, if u was previously correct and now it is not correct, $correct_{total}$ is decreased by one. Similarly, all edges incident with relevant vertices are checked, and consequently, value of w_{sol} is adjusted. The adjustment takes place in two situations: firstly, if edge became correct after movement, while it was not correct before movement, and vice versa, if it is not correct after movement while it was correct previously. The total objective function is afterward assembled from these two partially calculated terms. If the solution cannot be further improved inside the local search, the (hopefully improved) solution is returned back to the main VNS algorithm where it is named \mathbf{x}'' .

The next step of the algorithm is to compare the quality of the current best solution \mathbf{x} to the solution \mathbf{x}'' , obtained after shaking and LS is finished. If the value of the objective of the solution \mathbf{x}'' is greater than of the solution \mathbf{x} , then \mathbf{x}'' becomes the new current best solution $(\mathbf{x} = \mathbf{x}'')$. If the objective function value of the solution \mathbf{x}'' is less than the objective function value of the solution \mathbf{x} , then the solution \mathbf{x} remains the current best one. If the values of the objective function of both solutions are the same, then \mathbf{x} is set to \mathbf{x}'' with a probability of prob.

```
input: \mathbf{x}'', i, p, k
    output: objValue
 1 correct_{total} \leftarrow correctVertices(\mathbf{x''}, k);
 \mathbf{z} \ w_{sol} \leftarrow \text{sumOfEdges}(\mathbf{x}'');
 3 p_{old} \leftarrow x_i'';
 4 V_{relevant} \leftarrow \{u|u \in \mathbf{x}'', u = p_{old}\} \cup \{u|u \in \mathbf{x}'', u = p\};
 5 for
each u \in V_{relevant} do
         if u_{correct} then
              correct_{total} \rightarrow correct_{total} - 1:
             c_{correct} = \text{correctAfterReposition}(u, p_{old}, p, k);
          if u'_{correct} then
               correct_{total} \rightarrow correct_{total} + 1;
          \dot{E}_{incident} \leftarrow \{(u, v) | (u, v) \in E, u < v\};
11
          foreach (u, v) \in E_{incident} do
12
               v'_{correct} = \!\! \text{correctAfterReposition}(v, p_{old}, p, \, k);
13
14
               if u_{correct} \wedge v_{correct} \wedge (\neg u'_{correct} \vee \neg v'_{correct}) then
15
                   w_{sol} \leftarrow w_{sol} - w_{uv};
               else if u'_{correct} \wedge v'_{correct} \wedge (\neg u_{correct} \vee \neg v_{correct}) then
16
17
                w_{sol} \leftarrow w_{sol} + w_{uv};
         end
18
19 end
20 objValue \leftarrow correct_{total} + w_{sol}/w_{total};
21 return objValue;
```

Figure 3: Partial calculation of the objective function

4. Experimental results

In this section we evaluate the performance of the proposed VNS method. All experiments are performed on the Intel Xeon E5410 CPU @2.33 GHz with 16 GB RAM and Windows Server 2012 2R 64Bit operating system. For each execution only one thread/processor is used. The VNS is implemented in C programming language and compiled with Visual Studio 2015 compiler.

In order to make the comparison to the other method from the literature as fair as possible, we used the same benchmark data sets as in [28] and tested them for three values of k, namely $k \in \{1,2,3\}$. The first two sets contain biological instances created on metabolic reactions from [30], as it is described in [28]. In Table 1 we show these biological data in a systematized view, with shorten instance names. The third set of instances was taken from the well known DIMACS database, available at http://www.dcs.gla.ac.uk/~pat/maxClique. Testing the VNS on DIMACS instances consists of two phases. In the first phase, we followed the approach from [28], and took DIMACS instances with less than 100 vertices and larger sparse instances with less than 200 vertices and density at most 0.25. In the second phase, we tested our VNS on the rest of 73 DIMACS instances.

Since the original DIMACS instances are not weighted, like in other papers [28, 31], we also followed the weighting strategy proposed in [32], setting $w_{i,j} = ((i+j) \mod 200) + 1$. For each

instance, we performed 10 independent executions of the VNS algorithm. Termination criterion is based on the combination of three criteria: the maximum total number of iterations reached, where $it_{max} = 20000$, the maximum number of iterations without improvement, which is set to 10000, or maximum total execution time reached, which is set to 1 hour. Other control parameters are set as follows: $n_{min} = 1$ and $n_{max} = 80$ and prob = 0.1.

4.1. Experimental results obtained on SC-NIP-m-tr instances

This section reports the experimental results for the set of SC-NIP-m-tr, r = 1, ..., 5 instances. As it is already mentioned, we considered three values of k, i.e. $k \in \{1, 2, 3\}$. Table 2 columns are organized as follows. In the first two columns, values of k and a shorten name of the instance are shown. The next two columns contain the optimal value and the best known value, if the optimal value is not known; the following five columns contain data related to the VNS: the best and average VNS results obtained in 10 runs, average gap, total execution time (in seconds). If the best achieved VNS result is equal to the known optimal value, then the column VNS_{best} contains the mark opt. If the optimal value is not known, but VNS succeeds to find the best known value, then that column contains the mark best. The mark new is used if there are no previous results for the considered instance. The last two columns are related to the exact ILP method proposed in [28]: column ILP contains the result obtained by the mentioned ILP, with the marks opt or best if the result obtained by the ILP is optimal or the best known. In cases when ILP could not find any solution due to the memory limits, the mark "—" is used. Table 2 shows that the proposed VNS

Table 1: A view of considered biological metabolite networks

inst. $ V $	E	density	inst.	V	E	density
m-t1 991	4161	0.0085	r-t1	1393	56276	0.0580
m-t2 602	1520	0.0084	r-t2	1183	17776	0.0254
m-t3 177	269	0.0173	r-t3	663	1782	0.0081
m-t4 129	166	0.0201	r-t4	377	321	0.0045
m-t5 75	84	0.0303	r-t5	45	27	0.0272

succeeds to find all 9 known optimal solutions. In addition, for each of these instances, the VNS reaches the optimal value in each of 10 runs. For the rest of 6 instances, where ILP model from [28]

could not find any solution, the VNS succeeds to find solution in a reasonable time up to 1 hour. The average gap is rather small and it is less than 1 for all instances. Thus, one can conclude that the VNS is rather stable while solving this class of instances.

Table 2: Experimental results obtained on SC-NIP-m-tr instances

k	inst.	opt	best	V_{best}	V_{avg}	V_{gap}	V_t^{tot}	ILP	ILP_t
1	m-t1	1866	1866	opt	1864	0.11	3600.22	opt	2296.94
1	m-t2	1538	1538	opt	1538	0	1072.51	opt	1.25
1	m-t3	910	910	opt	910	0	92.96	opt	0.02
1	m-t4	831	831	opt	831	0	45.5	opt	0
1	m-t5	723	723	opt	723	0	15.73	opt	0
2	m-t1	-	2151	new	2147.3	0.17	3600.14	-	-
2	m-t2	-	1773	new	1771.8	0.07	1495.49	-	-
2	m-t3	1021	1021	opt	1021	0	100.74	opt	50.43
2	m-t4	907	907	opt	907	0	54.75	opt	3.03
2	m-t5	801	801	opt	801	0	16.42	opt	0.2
3	m-t1	-	2353	new	2337.1	0.68	3600.18	-	-
3	m-t2	-	1943	new	1939.4	0.19	1988.38	-	-
3	m-t3	-	1141	new	1141	0	121.08	-	-
3	m-t4	-	1022	new	1022	0	69.79	-	-
3	m-t5	887	887	opt	887	0	17.62	opt	34.2

The ILP method from [28] was more successful for k = 1 comparing to the two other values of k. It succeeded to find all optimal solutions for k = 1, three optima for k = 2 and one optimum for k = 1. The proposed VNS achieves all these optimal solutions, but also succeeds to find solutions regardless of the value k in a very reasonable time, up to 3020 seconds.

4.2. Experimental results obtained on SC-NIP-r-tr instances

This section provides experimental results obtained on the second class of biological data, i.e. SC-NIP-r-tr instances. The results are shown in Table 3, which is organized in similar way as Table 2. As it can be seen from Table 3, VNS achieves all 7 known optimal solutions. For the rest of 8 instances VNS achieves the new best results. With respect to the solution quality, the picture is similar as in the case of SC-NIP-m-tr instances. The ILP method from [28] succeeds to find 7 optimal solutions: four optima for k = 1, two optima for k = 2 and one optimum for k = 3. The proposed VNS succeeds to find all these optima, also providing high quality solutions for other cases.

Table 3: Experimental results obtained on SC-NIP-r-tr instances

k inst.	opt	best	V_{best}	V_{avg}	V_{gap}	V_t^{tot}	ILP	ILP_t
1 r-t1	-	57681	new	57544.6	0.24	3607.77	-	
1 r-t2	34576	34576	opt	34561.6	0.04	3601.2	opt	4.26
1 r-t3	5411	5411	opt	5411	0	1550.95	opt	0.08
1 r-t4	1232	1232	opt	1232	0	327.82	opt	0
1 r-t5	140	140	opt	140	0	3.71	opt	0.02
2 r-t1	-	57729	new	57496	0.4	3602.58	-	
2 r-t2	-	34592	new	34563.6	0.08	3601.65	-	
2 r-t3	-	5423	new	5423	0	1569.11	3183	> 10800
2 r-t4	1245	1245	opt	1245	0	331.75	opt	6.4
2 r-t5	140	140	opt	140	0	3.82	opt	0.01
3 r-t1	-	57775	new	57587.4	0.33	3602.19	-	
3 r-t2	-	34641	new	34572.5	0.2	3601.26	-	
3 r-t3	-	5465	new	5465	0	1496.84	-	
3 r-t4	-	1245	new	1245	0	327.45	-	
3 r-t5	140	140	opt	140	0	3.84	opt	0.14

From computational point of view, the SC-NIP-r-tr instances are more challenging than SC-NIP-m-tr because of their dimensions (see Table 1). Therefore, necessary runtime is proportionally greater comparing to the execution times for SC-NIP-m-tr instances. For five SC-NIP-r-tr instances, the algorithm stopped after the time limit is reached (1 hour), while for other instances, the termination happened after maximum number of iteration was reached. The average gap for this class is again rather small (less than 1) for all instances.

Tables 2 and 3 show that in both classes of biological instances, execution time depends on the graph density, i.e. smaller density induces smaller execution time. A natural explanation is that smaller number of edges causes the lowering of the total number of executions of the local search procedure, which further leads to the shorter overall execution time. By comparing the values in columns opt and best for the same instance and different values of k, we conclude that the value of objective functions increases with increasing the value of k. This is because the total number of edges included in clusters increases with the relaxation of the adjacency conditions in each cluster.

5. Visualization and biological explanation of the obtained results

In the following consideration, we will show how the relaxation of the clustering requirements can lead to more useful information from biological point of view. Among many metabolic processes

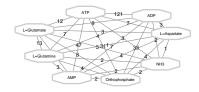


Figure 4: The largest cluster obtained for k = 1

that appeared in various k-plexes obtained by the proposed VNS algorithm, we chose to discuss following processes: amino acid degradation process, fatty acids synthesis, vitamin B6 synthesis, oxidation of the succinate to the fumarate and formaldehyde oxidation.

In order to confirm the reliability of the obtained results, particular information of the biochemical pathways of considered organism Saccharomyces cerevisiae are checked and confirmed with the data presented in Yeast Pathways Database [33].

5.1. Amino acid degradation process

In this subsection, we consider the amino acid degradation process, which is one of the most important processes in metabolism. In Figure 4 we show the largest cluster obtained for k = 1, which contains the rough representation of the amino acid degradation.

Ammonia presented in the organism is used as a resource of nitrogen for amino-acid synthesis and if it released in larger quantity, it must be removed because of its toxicity.

In the considered organism Saccharomyces cerevisiae, ammonia can be incorporated into the amino group of glutamate, by two pathways: the reductive amination of 2-ketoglutarate, catalyzed by glutamate dehydrogenase where NADPH serves as the source of electrons, or by the ATP-dependent synthesis of glutamine from glutamate and ammonia catalyzed by glutamine synthetase [34].

This cluster is shown in Figure 4 is a clique with 8 vertices and contains the main intermediates which figure in ammonia synthesis from glutamic and aspartic acids. Glutamate binds the orthophosphoric group from ATP, resulting in glutamine (ADP is formed, and orthophosphate is released). Further, in Figure 5 we show the largest cluster obtained for k = 2. A wider set of intermediates is now shown, also including additional reactions. We again see that glutamate appears in the reaction of glutamine and L-Aspartate by ATP consumption. With this reaction asparagine is formed, which is converted to aspartate by deamination, while the ammonia is released. Also,

glutamate and ammonia is released by deamination of glutamine. If the system is extended by two additional intermediates, we see one more way of glutamate synthesis, that is reaction of CO_2 and glutamine by ATP consumption.

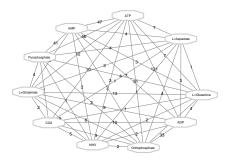


Figure 5: The largest cluster obtained for k=2

More detailed graphical interpretation is shown in Figure 6, obtained by the proposed algorithm for k=3. Since the condition for forming clusters is now more relaxed, more intermediates figures in the cluster. In addition to the previous ones, in the cluster shown in Figure 6 we see the process of the oxidative deamination. The oxidative deamination which occurs in cells involves the amino acid glutamate. Glutamate is oxidatively deaminated by the enzyme glutamate dehydrogenase, using NAD or NADP as a coenzyme. By this process, two toxic products are synthesized: hydrogen peroxide and ammonia. In Figure 6 we see that the algorithm grouped all these intermediates in one cluster, which was not the case with the more strict conditions (cases k=1 and k=2).



Figure 6: The largest cluster obtained for k=3

5.2. Fatty acids synthesis

In Figure 7 we show the second largest cluster obtained for k=1. We can see that the algorithm grouped intermediates which figure in fatty acids synthesis. Fatty acids are long molecules and the process of their synthesis can be divided in three phases. The first phase includes synthesis of the coenzyme malonil-CoA from acetil-CoA, since the malonil-CoA is more reactive molecule and more suitable for the extension of the fatty acid chain. Acetil-CoA is synthesized from CoA by consumption of the ATP which is converted to ADP with the release of orthophosphate. As it is shown in Figure 7, the edge connecting acetil-CoA and malonil-CoA is of the weight 2 since their reaction is reversible, so their link is counted twice. The second phase consists of five successive cycle reactions, starting with the binding of the acetil-CoA and malonil-CoA directly to the carrier protein, after malonil-acyl carrier protein (ACP) is formed and CoA is released. In Figure 8 are shown these reactions of binding amino acid with acyl carrier protein (ACP). From this figure one can see that the largest number of reactions are related to synthesis of acetil-CoA from CoA and to binding of the malonil-CoA to carrier protein, which proves that the algorithm recognized the overall system of the biosynthesis. The reaction is continued by extending the chain by 2C atoms (new malonil-CoA is binded), until a long fatty acid is synthesized. Each time when malonile-CoA (new malonil-CoA is binded), until a long fatty acid is synthesized.

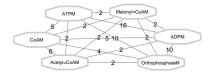


Figure 7: Fatty acids synthesis cluster for k = 1

is binded with ACP, CoA is released. During the condensation with ACP, CO₂ is removed and oxo compounds appear, which are reduced by the presence of NADPH (which is transformed to NADP+) and are hydrolyzed to enoyl compounds, which again are reduced with NADPH. In the third phase, saturated extended fatty acid accepts new malonil-CoA and further extends the chain by the same scheme (Figure 9). To conclude, the algorithm recognized the complex process by identifying the highly weighted clusters related to the fatty acids synthesis. Like in the case of amino acid degradation process explained in Section 5.1, the results obtained by the relaxation of the condition, are used to obtain useful information of the metabolic reactions.

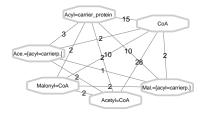


Figure 8: Extended display of fatty acids synthesis for k=2

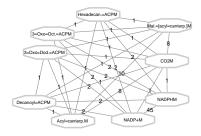


Figure 9: More detailed display of fatty acids synthesis, obtained for k=3

5.3. Other useful findings

Among other clusters obtained by the proposed algorithm on the considered network, we notice two interesting clusters obtained for k=2: vitamin B6 synthesis and glutathione dependent formaldehyde oxidation. In Figure 10, obtained for k=2 the main intermediates for vitamin B6 synthesis are shown. Pyridoxal phosphate (PLP) is the active form of vitamin B6 and is a cofactor in many reactions of amino acid metabolism [33]. From the figure, we see that the algorithm grouped different forms of vitamin B6: Pyridoxine, pyridoxal (PL) and pyridoxine 5'-phosphate (PNP). Saccharomyces cerevisiae synthesizes PLP via the fungal type de novo PLP synthesis pathway and the PLP salvage pathway. Through these biochemical pathways PLP can be obtained from PL or by synthesis from pyridoxine. This pathway consists of two steps, in the first pyridoxine 5'-phosphate is synthesized from pyridoxine by the action of pyridoxine kinase enzyme. The second step is based on the oxidation of pyridoxine 5'-phosphate to pyridoxal phosphate. This reaction requires the participation of oxygen which is reduced to peroxide in this reaction. All these intermediates are present in Figure 10.

Another interesting example, shown in Figure 11, is the process of removing very reactive and toxic formaldehyde. Although formaldehyde cannot be metabolized from methanol in S. cerevisiae,

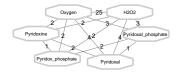


Figure 10: The main intermediates for vitamin B6 synthesis, the cluster obtained for k=2

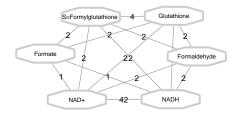


Figure 11: Main intermediates involved in removing formal dehyde, the cluster obtained for k=2

it can be adopted from plant material or in polluted air and water [33]. Therefore, a system for removing this toxic compound should be provided. The metabolite which plays an important role in this overall defense is glutathione, which has the ability to bind formaldehyde by the spontaneous reaction. The resulting S-hydroxymethylglutathione is oxidized to S-formyl-glutation with the participation of NAD+ as an oxidizing agent, which is reduced to NADH. By hydrolysis of this compound the products are glutathione and non-toxic formate.

In the last interesting example, shown in Figure 12, we mention the process of oxidation the succinate to the fumarate. This reaction is possible with the participation of the succinate dehydrogenase enzyme, which is covalently bound to flavin adenine dinucleotide (FAD), which acts as a hydrogen ion acceptor, reducing it to FADH₂. For the given reaction, all intermediates are present even in the cluster obtained for k = 1, so any further relaxation cannot include new members. The algorithm recognized this situation by throwing out the same graph across all three values of k.

5.4. Experimental results obtained on DIMACS instances

In this subsection we first continue with the comparison with the existing ILP model on a subset of DIMACS instances. In Table 4 we show the results organized in the same way as for biological instances. All these instances belong to c-fat, MANN, hamming and johnson families (in Table 4 the names of instances are shortened). From Table 4 one can conclude that the proposed

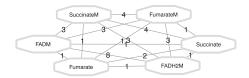


Figure 12: The same cluster of oxidation the succinate to the fumarate, obtained for k = 1, 2 and 3

VNS achieves 9/9 known optimal solutions. In the rest of 12 cases, VNS achieves 4 best known solutions, while in 8 cases finds new best results. In regards to the efficiency of the proposed VNS on the larger problem dimensions, the algorithm is also tested on the challenging set of the rest of 73 DIMACS instances. Although the Max-EkPP is mostly applied on sparse graphs, to achieve completeness of our approach, we decided to test the proposed VNS even on denser DIMACS instances. For these instances, up to now, no solution is presented in the literature. Although optimallity cannot be proved, small gap values on these instances suggest that VNS obtained high quality solutions. Since the large amount of obtained data will burden the rest of the paper, we decided not to incorporate these results in this paper, but they are available at the first author website http://matinf.pmf.unibl.org/dimacs/.

High quality of the solutions obtained by the proposed VNS clearly indicates that it is capable for real applications in partitioning biological and other networks in k-plexes. The VNS is stable and accurate, while execution times still remain relatively small, even for large instances.

6. Conclusions

In this paper, we presented a variable neighborhood search based heuristics for solving the problem of partitioning sparse biological networks into edge-weighted structures called k-plexes. This problem, called Max-EkPP problem has been solved for the first time by a metaheuristic approach. The proposed VNS implements a fast swap-based local search, as well as a specific objective function which favors feasible solutions over infeasible ones, taking into consideration the degree of every vertex in each partition. An extensive computational analysis is performed on existing sparse biological metabolic networks, as well as on the other artificial instances from literature. From computational point of view, it was shown that the proposed VNS succeeded to

Table 4: Experimental results obtained on smaller and sparser DIMACS instances									
k	inst.	opt	best	V_{best}	V_{avg}	V_{gap}	V_t^{tot}	ILP	ILP_t
1	c200-1	98711	98711	opt	98711	0	234.43	opt	47.08
2	c200-1	98711	98711	opt	98543.2	0.17	202.87	opt	567.44
3	c200-1	-	98711	new	98571.8	0.14	193.7	-	-
1	c200-2	213248	213248	opt	213246.8	0	540.89	opt	0.22
2	c200-2	213248	213248	opt	212194.6	0.49	360.5	opt	47.28
3	c200-2	-	213248	new	211143.8	0.99	292.97	-	-
1	h6-2	65472	65472	opt	65472	0	114.53	opt	0.2
2	h6-2	-	65472	best	65472	0	61.91	best	>10800
3	h6-2	-	65472	best	65472	0	46.15	best	>10800
1	h6-4	6336	6336	opt	6336	0	53.29	opt	0.34
2	h6-4	-	8184	new	8184	0	74.81	6966	>10800
3	h6-4	-	10560	new	10560	0	77.57	4567	>10800
1	j8-2-4	1260	1260	opt	1260	0	7.63	opt	0.06
2	j8-2-4	-	1365	new	1363.5	0.11	10.41	1355	>10800
3	j8-2-4	-	1996	best	1996	0	7.34	best	>10800
1	j8-4-4	-	27874	new	27874	0	169.18	27864	>10800
2	j8-4-4	-	31320	new	31147.2	0.55	124.87	12770	>10800
3	j8-4-4	-	37096	new	35910.3	3.2	155.73	12948	>10800
1	M_a9	14868	14868	opt	14865	0.02	27.55	opt	1215.34
2	M_a9	-	23055	new	23053.8	0.01	25.96	23047	>10800
3	M_a9	33660	33660	opt	33660	0	14.23	opt	319.24

achieve all already known optimal or best solutions. It was also able to find new high quality solutions for the other, previously unsolved instances, in a reasonable time.

In the deep analysis of the clusters identified by various values of k on a biological metabolic instance, we confirmed that the algorithm finds many clusters in which the intermediates, that figures in many important metabolic reactions, are highly connected. The relaxation of the adjacency condition leads to obtainment of more useful clusters, which helps in discovering new biological relations or confirming the existing ones.

This research can be extended in several ways. For example, it would be interesting to apply the VNS on solving similar problems, including both biological and non-biological applications. Another direction for the further investigation of this problem can include parallelization of the proposed VNS algorithms and running on some powerful multiprocessor system.

References

References

- [1] T. Akutsu, M. Hayashida, D. B. KC, E. Tomita, J. Suzuki, K. Horimoto, Dynamic programming and clique based approaches for protein threading with profiles and constraints, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 89 (5) (2006) 1215–1222.
- [2] B. JB, D. B. KC, E. Tomita, T. Akutsu, Multiple methods for protein side chain packing using maximum weight cliques, Genome Informatics 17 (1) (2006) 3–12.
- [3] V. Spirin, L. A. Mirny, Protein complexes and functional modules in molecular networks, Proceedings of the National Academy of Sciences 100 (21) (2003) 12123–12128.
- [4] J. Gagneur, R. Krause, T. Bouwmeester, G. Casari, Modular decomposition of protein-protein interaction networks, Genome biology 5 (8) (2004) R57.
- [5] S. B. Seidman, B. L. Foster, A graph-theoretic generalization of the clique concept, Journal of Mathematical sociology 6 (1) (1978) 139–154.
- [6] B. Balasundaram, S. Butenko, I. V. Hicks, Clique relaxations in social network analysis: The maximum k-plex problem, Operations Research 59 (1) (2011) 133–142.
- [7] P. Martins, Extended and discretized formulations for the maximum clique problem, Computers & Operations Research 37 (7) (2010) 1348–1358.
- [8] B. McClosky, I. V. Hicks, Combinatorial algorithms for the maximum k-plex problem, Journal of combinatorial optimization 23 (1) (2012) 29–49.
- [9] H. Moser, R. Niedermeier, M. Sorge, Algorithms and experiments for clique relaxationsfinding maximum s-plexes, in: International Symposium on Experimental Algorithms, Springer, 2009, pp. 233–244.
- [10] J. Pattillo, N. Youssef, S. Butenko, Clique relaxation models in social network analysis, Handbook of Optimization in Complex Networks (2012) 143–162.

- [11] F. Hüffner, C. Komusiewicz, A. Liebtrau, R. Niedermeier, Partitioning biological networks into highly connected clusters with maximum edge coverage, IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 11 (3) (2014) 455–467.
- [12] E. Hartuv, A. O. Schmitt, J. Lange, S. Meier-Ewert, H. Lehrach, R. Shamir, An algorithm for clustering cdna fingerprints, Genomics 66 (3) (2000) 249–256.
- [13] S. Navlakha, C. Kingsford, Exploring biological network dynamics with ensembles of graph partitions., in: Pacific Symposium on Biocomputing, Vol. 15, 2010, pp. 166–177.
- [14] H. Chen, W. Chung, J. J. Xu, G. Wang, Y. Qin, M. Chau, Crime data mining: a general framework and some examples, computer 37 (4) (2004) 50–56.
- [15] L. Terveen, W. Hill, B. Amento, Constructing, organizing, and visualizing collections of topically related web resources, ACM Transactions on Computer-Human Interaction (TOCHI) 6 (1) (1999) 67–94.
- [16] P. Krishna, N. H. Vaidya, M. Chatterjee, D. K. Pradhan, A cluster-based approach for routing in dynamic networks, ACM SIGCOMM Computer Communication Review 27 (2) (1997) 49–64.
- [17] M. Mukherjee, L. B. Holder, Graph-based data mining on social networks, Ph.D. thesis, University of Texas at Arlington (2004).
- [18] B. Balasundaram, Cohesive subgroup model for graph-based text mining, in: Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on, IEEE, 2008, pp. 989–994.
- [19] V. Boginski, S. Butenko, O. Shirokikh, S. Trukhanov, J. G. Lafuente, A network-based data mining approach to portfolio selection via weighted clique relaxations, Annals of Operations Research 216 (1) (2014) 23–34.
- [20] L. Cowen, W. Goddard, C. E. Jesurum, Defective coloring revisited, Journal of Graph Theory 24 (3) (1997) 205–219.
- [21] S. Trukhanov, C. Balasubramaniam, B. Balasundaram, S. Butenko, Algorithms for detecting optimal hereditary structures in graphs, with application to clique relaxations, Computational Optimization and Applications 56 (1) (2013) 113–130.

- [22] M. Grötschel, Y. Wakabayashi, A cutting plane algorithm for a clustering problem, Mathematical Programming 45 (1) (1989) 59–96.
- [23] U. Dorndorf, E. Pesch, Fast clustering algorithms, ORSA Journal on Computing 6 (2) (1994) 141–153.
- [24] M. Oosten, J. H. Rutten, F. C. Spieksma, The clique partitioning problem: facets and patching facets, Networks 38 (4) (2001) 209–226.
- [25] H. Wang, B. Alidaee, F. Glover, G. Kochenberger, Solving group technology problems via clique partitioning, International Journal of Flexible Manufacturing Systems 18 (2) (2006) 77–97.
- [26] Y. Zhou, J.-K. Hao, A. Goëffon, A three-phased local search approach for the clique partitioning problem, Journal of Combinatorial Optimization 32 (2) (2016) 469–491.
- [27] J. Brimberg, S. Janićijević, N. Mladenović, D. Urošević, Solving the clique partitioning problem as a maximally diverse grouping problem, Optimization Letters 11 (6) (2017) 1123–1135.
- [28] P. Martins, Modeling the maximum edge-weight k-plex partitioning problem, arXiv preprint arXiv:1612.06243.
- [29] N. Mladenović, P. Hansen, Variable neighbourhood search, Computers & Operations Research 24 (1997) 1097–1100.
- [30] J. Förster, I. Famili, P. Fu, B. Ø. Palsson, J. Nielsen, Genome-scale reconstruction of the saccharomyces cerevisiae metabolic network, Genome research 13 (2) (2003) 244–253.
- [31] L. Gouveia, P. Martins, Solving the maximum edge-weight clique problem in sparse graphs with compact formulations, EURO Journal on Computational Optimization 3 (1) (2015) 1–30.
- [32] W. Pullan, Approximating the maximum vertex/edge weighted clique using local search, Journal of Heuristics 14 (2) (2008) 117–134.
- [33] Yeast pathways database, http://https://pathway.yeastgenome.org/, accessed: 2017-11-11.
- [34] B. Magasanik, Ammonia assimilation by saccharomyces cerevisiae, Eukaryotic Cell 2 (5) (2003) 827–829.