Algorithms for the Maximum Weight Connected k-Induced Subgraph Problem

Ernst Althaus, Markus Blumenstock(✉), Alexej Disterhoft, Andreas Hildebrandt, and Markus Krupp

Institut für Informatik, Johannes Gutenberg-Universität, Mainz, Germany
markusblumenstock@hotmail.com,
{ernst.althaus,andreas.hildebrandt,kruppm}@uni-mainz.de,
alexej@disterhoft.de

Abstract. Finding differentially regulated subgraphs in a biochemical network is an important problem in bioinformatics. We present a new model for finding such subgraphs which takes the polarity of the edges (activating or inhibiting) into account, leading to the problem of finding a connected subgraph induced by k vertices with maximum weight. We present several algorithms for this problem, including dynamic programming on tree decompositions and integer linear programming. We compare the strength of our integer linear program to previous formulations of the k-cardinality tree problem. Finally, we compare the performance of the algorithms and the quality of the results to a previous approach for finding differentially regulated subgraphs.

Keywords: Linear programming · k-cardinality tree · Tree decomposition · Heuristics · Bioinformatics · Gene regulation

1 Introduction

1.1 Problem Definition

We are considering the following problem: given a simple graph $G = (V, E)$, edge weights $w : E \mapsto \mathbb{R}$, and an integer $k \in \{1, \ldots, |V|\}$, find a subset $V' \subseteq V$ of k vertices (i.e. $|V'| = k$) such that the subgraph induced by V' is connected and has maximum total edge weight (i.e. $\sum_{e \in E \cap (V' \times V')} w(e)$ is maximized). We call this the maximum weight connected k-induced subgraph (MWCIS) problem. If connectivity is not required, we refer to it as the MWIS problem. Both problems are easily seen to be NP-complete by a reduction from the CLIQUE problem.

There are several variants of this problem. We can have vertex scores only or additionally (i.e. the weight-function is $\sum_{e \in E \cap (V' \times V')} w(e) + \sum_{v \in V'} s(v)$), or we can sum the weights of all edges with at least one endpoint in V' (i.e. the weight-function is $\sum_{e \in E | e \cap V' \neq \emptyset} w(e)$).

Notice that the latter can be solved with the induced edge-weight objective and additional vertex scores by setting $s(v) = \sum_{uv \in E} w(u, v)$ and flipping the sign of all edge weights. As all our algorithms are capable of handling vertex scores, we restrict to the induced edge-weight objective unless stated otherwise.

Z. Zhang et al. (Eds.): COCOA 2014, LNCS 8881, pp. 268–282, 2014.
DOI: 10.1007/978-3-319-12691-3_21

1.2 Application to Bioinformatics

The motivation for our work comes from bioinformatics, where interactions between biochemical entities (proteins, metabolites, DNA, ...) are often represented as graphs named biochemical networks. An important application of such networks is the detection of differentially regulated (or *deregulated*) pathways or subnetworks, where we are asked to determine which parts of the network react most drastically upon environmental changes, or changes of the phenotype. To this end, we are given a biochemical network and a set of quantitative measurements for each vertex as a function of the environmental change or the phenotype. Typically, the vertices represent proteins, and the quantitative data comes in the form of expression values. An exemplary study might, for instance, want to determine which subnetworks are significantly activated or deactivated as a result of a certain type of cancer. Hence, the input would consist of a biochemical network representing the current knowledge on protein interactions in humans, as well as of measured expression values for a number of patients and a healthy control group.

Detecting deregulated subgraphs requires a measure of deregulation for a subnetwork. While simple measures, such as the addition of vertex-based expression values, can be easily established, statistically significant results require much more elaborate procedures, such as the popular Gene Set Enrichment Analysis (GSEA) [STM+05] and its variants [DPM+09], which uses careful sampling and a Kolmogorov-Smirnov test to establish whether the set of expression changes on the vertices of a given subgraph is of statistical significance. To detect subgraphs of interest, one then tries to find connected sets of vertices with maximal total vertex score, often under additional assumptions, such as the existence of regulatory cascades, where a single so-called *key player* controls the regulation of several downstream genes. The resulting subgraphs are then scored using the full GSEA procedure to establish their statistical significance.

It has recently been argued that this approach often overestimates the significance, as it ignores inconsistencies in the data: interactions can have both a *direction* (which can be integrated easily into the procedures described above) and a *polarity*, i.e., one vertex can activate or inhibit expression of the other. As a result of noise both in the network models as well as in the expression data, expression differences are often inconsistent with the polarity of the interaction. We often find, e.g., cases where two proteins A and B both show increased expression levels in, say, the diseased sample, even though expression of A is supposed to inhibit expression of B. GSEA analysis would reward the inclusion of A and B in a subset of differentially regulated genes, as both are connected and show a differential regulation. On the other hand, the inconsistency should instead make us suspicious of their relevance.

To include consistency into the scoring of subnetworks, Geistlinger et al. have recently proposed the so-called Gene Graph Enrichment Analysis (GGEA) approach [GCK+11], which replaces the vertex-based scores of GSEA with edge weights which are computed from the expression changes of both vertices incident to the edge, as well as from its polarity. While this change has been shown

to lead to improved scoring of subnetworks [GCK+11], previous optimization approaches to detect the most strongly differentially regulated parts do not directly apply, as they are based on vertex scores instead of edge weights.

2 Related Work

2.1 Other Approaches for Finding Deregulated Networks

Backes et al. [BRK+11] modeled the problem of finding a deregulated subgraph differently in two ways. On the one hand, they use a vertex-score function, in which a vertex has a high score if the corresponding gene is deregulated. They do not take into account whether the sign of the deregulation is consistent with its predecessor gene. On the other hand, they consider a directed network, where the direction indicates the causing and the affected gene. Hence they require a designated root vertex that corresponds to a key player gene responsible for deregulation, from which all other vertices are reachable. We will review the integer linear programming approach by Backes et al. in Sect. 3.1.

ILP approaches for undirected, edge-weighted deregulated networks include finding paths [ZWCA08] and finding maximum connected subgraphs with vertex scores (often also called weights). The latter is called the MWCS problem, which was solved by transformation into the prize-collecting Steiner tree problem by Dittrich et al. [DKR+08].

2.2 Similar Problems

Álvarez-Miranda et al. [AMLM13] compared an ILP formulation for the prize-collecting Steiner tree problem to an ILP formulation of MWCS and the Backes approach by polyhedral comparison.

In the related k-cardinality tree problem, one searches for a tree with k edges that minimizes the sum of all edge weights (sometimes nonnegative) or vertex scores. ILP Formulations for the k-cardinality tree (and closely related problems) were given by Fischetti et al. [FHJM94], Garg [Gar96], and Ljubić [Lju04]. Recent works are by Quintão et al. [QdCM08, QadCML10], which use the Miller-Tucker-Zemlin constraints [MTZ60], and Chimani et al. [CKLM10], which compares the approaches by Fischetti et al., Garg, and Ljubić polyhedra-wise and gives the best separation routine in practice. Approximation algorithms were given by Blum et al. [BRV99] and Arora and Karakostas [AK00], and for metaheuristics, we refer to Blum and Blesa [BB05] for a comparison.

Another similar problem is the densest k-subgraph problem (DKS), which is defined by the average vertex degree of graphs. The average vertex degree of a simple graph $G = (V, E)$ is defined as $\mathrm{ad}(G) := \frac{2|E|}{|V|}$. The maximum average degree of G is defined as the maximum of the average degrees of all subgraphs, $\mathrm{mad}(G) := \max_{H \subseteq G} \mathrm{ad}(H)$. This will appear again later in Sect. 3.3.

Finding the subgraph with maximum average vertex degree, i.e. the densest subgraph, is computable in polynomial time using flow techniques [Law76]. If the

subgraph size is restricted to k vertices, the problem is NP-hard by a reduction from Clique and can be seen as a special case of Mwis with unit weights.

Feige et al. [FKP01] give a polynomial-time algorithm that solves DkS with an approximation ratio of $\mathcal{O}(|V|^{1/3-\epsilon})$. They show that such an algorithm can be used to approximate the Mwis problem for nonnegative edge weights with a loss in the approximation ratio of $\mathcal{O}(\log |V|)$. Bhaskara et al. [BCC+10] give an algorithm using linear and semidefinite programming relaxations that achieves an $\mathcal{O}(|V|^{1/4+\epsilon})$-approximation in polynomial time and an $\mathcal{O}(|V|^{1/4})$-approximation in $\mathcal{O}(|V|^{\log |V|})$ time. Moreover, it was shown by Khot [Kho06] that under the assumption that there are no subexponential algorithms for NP-complete problems, there is no polynomial-time approximation scheme for the DkS problem.

3 Integer Linear Programming Formulations

In all following ILP approaches, there are binary variables y_v indicating whether a vertex v is selected for the subgraph, and binary variables z_{uv} indicating selected edges for the objective function $\max \sum_{e \in E} z_e w(e)$. The induced edges can be modeled by the constraints $z_{uv} \geq y_u + y_v - 1$ and $z_{uv} \leq y_u$, y_v. The objective where an edge contributes its weight when at least one end vertex is selected is modeled by $z_{uv} \leq y_u + y_v$ and $z_{uv} \geq y_u$, y_v.

3.1 Adapting the Approach by Backes et al.

In the following, we adapt the approach by Backes et al. for the Mwcis problem. The constraints for the variables y_v are adopted from Backes et al. with removal of the root vertex constraints. The most interesting set of constraints consists of those enforcing connectivity which is done as follows. We require that for every set $C \subseteq V$ with $|C| < k$, at least one adjacent vertex is also selected, i.e. $\sum_{w \in In(C)} y_w \geq y_v \; \forall v \in C \; \forall C \subseteq V$ with $|C| < k$, where $In(C)$ is the set of all vertices in $V \setminus C$ with at least one edge incident to a vertex in C.

Instead of generating this exponentially large set of constraints beforehand, a branch-and-cut procedure is used. The relaxation of the ILP is solved with the basic constraints, and we search for connected components in the subgraph induced by the (rounded) fractional solution. If there is more than one connected component, the corresponding connectivity constraints for every set of vertices that forms a connected component are added and the solving continues. If there is one connected component, it constitutes the solution.

3.2 Formulations for the k-Cardinality Tree Problem

The ILP formulation by Fischetti et al. uses binary variables y_v for vertices and b_{uv} for spanning edges in the undirected sense. Apart from the straightforward constraints that ensure k vertices and $k-1$ spanning edges (see (2) and (3) in the next section), there are an exponential number of constraints, the generalized

subtour elimination constraints (GSEC). For every set $S \subseteq V$ with $|S| \geq 2$ and for $t \in S$, the constraint

$$\sum_{uv \in E(S)} b_{uv} \leq \sum_{v \in S} y_v - y_t \tag{1}$$

is added to the model, where $E(S)$ denotes the edges induced by S.

The directed cut formulation (DCUT) by Chimani et al. transforms the problem into the k-arborescence problem with binary variables $x_{u,v}$ for the directed edges and an additional root vertex with directed edges to all vertices.

Both approaches use maximum-flow problems for separation on fractional solutions. One or more minimum cuts are extracted from the solution and their corresponding constraints are added to the model. We will introduce a novel formulation in the next subsection and compare it to the existing approaches.

3.3 A Novel k-Cardinality Tree Formulation Based on the Maximum Average Degree Problem

We propose a novel formulation with binary variables for vertices and undirected spanning edges and $\mathcal{O}(|V| + |E|)$ constraints. The idea to enforce acyclicity is that the maximum average degree of a tree of k vertices is $2(k-1)/k = 2 - 2/k$, while a cyclic graph has a maximum average degree of at least two.

The following has been proven by Cohen [Coh10]: For a graph of maximum average degree z, we can distribute a value of 2 for each edge (the degree generated by it) to its endpoints, i.e. define continuous edge flow values $f_{uv,u}$ and $f_{uv,v}$ with $f_{uv,u} + f_{uv,v} = 2$, such that the total amount assigned to a vertex is at most z, i.e. $\sum_{uv \in E} f_{uv,v} \leq z$ for all $v \in V$. Furthermore, this is not possible for any value z smaller than the maximum average degree. This leads to the following mixed integer linear programming formulation, where we use an edge flow of one instead of two since the model is linear:

Variables

$$\begin{aligned} y_v &\in \{0, 1\} && \forall\, v \in V \\ b_e &\in \{0, 1\} && \forall\, e \in E \\ f_{uv,u}, f_{uv,v} &\in \mathbb{R}_0^+ && \forall\, uv \in E \end{aligned}$$

Constraints

$$\sum_{v \in V} y_v = k \tag{2}$$

$$\sum_{e \in E} b_e = k - 1 \tag{3}$$

$$b_{uv} \leq y_u, y_v \quad \forall\, uv \in E \tag{4}$$

$$f_{uv,u} + f_{uv,v} = b_{uv} \quad \forall\, uv \in E \tag{5}$$

$$\sum_{uv \in E} f_{uv,v} \leq 1 - \frac{1}{k} \quad \forall\, v \in V \tag{6}$$

The integrality constraints ensure that a non-selected vertex v does not receive any flow because the incident spanning edge variables $\{b_{uv}\}_{uv \in E}$ must be zero as well and therefore do not generate any flow to v.

However, we can use the stronger constraint $\sum_{uv \in E} f_{uv,v} \leq \left(1 - \frac{1}{k}\right) y_v$ instead of (6) to forbid some solutions of the relaxed model that would otherwise be feasible. If a vertex then received less than $\left(1 - \frac{1}{k}\right) y_v$, the remaining flow could not be absorbed because of (2) and (3), so we can even write

$$\sum_{uv \in E} f_{uv,v} = \left(1 - \frac{1}{k}\right) y_v \quad \forall\, v \in V. \tag{6a}$$

In reverse, either (2) or (3) can be dropped if (6a) is added. Furthermore, it implies the inequality $f_{uv,v} \leq \left(1 - \frac{1}{k}\right) y_v$ for all edges and since we solve b_{uv} to integrality, we can add

$$f_{uv,u}, f_{uv,v} \geq \frac{1}{k} b_{uv} \quad \forall\, uv \in E, \tag{7}$$

which reduces the amount of possible relaxed solutions even more. We call the above formulation with the improved constraints the strong Cohen formulation (Cs).

3.4 Polyhedral Comparison to Existing k-Cardinality Approaches

Chimani et al. have shown that Gsec, DCut and the multi-commodity flow formulation by Ljubić are equivalent and strictly stronger than Garg's formulation. To compare two formulations, one compares the polyhedra defined by the constraints without integrality requirements. A formulation is said to be stronger than another if its relaxation gives a tighter upper bound than the other. The polyhedron of the strong Cohen formulation is

$$\mathcal{P}_{\text{Cs}} := \{(y, b, f_u, f_v) \in [0,1]^{|V|+3|E|} \mid (y, b, f_u, f_v) \,\text{satisfies}\, (3) - (5), (6a), (7)\}$$

and we denote the identical projection on the (y, b) subspace as $\text{proj}(\mathcal{P}_{\text{Cs}})$.

Likewise, we have the Gsec polyhedron

$$\mathcal{P}_{\text{Gsec}} = \{(y, b) \in [0,1]^{|V|+|E|} \mid (y, b) \,\text{satisfies}\, (1) - (3)\}.$$

Lemma 3.1. Gsec *and* Cs *are not comparable.*

Proof. Consider the graph with two connected components consisting of three interconnected vertices each and $k = 6$. Cs has an LP-solution by setting $f_{uv,u} = f_{uv,v} = 5/12$ for every edge uv (Fig. 1), but Gsec is clearly infeasible since for the set of three vertices in each component, the total value of the induced edges can be two at most, but all edge variables must sum to five.

On the other hand, consider the graph consisting of a path of three vertices and an isolated vertex, and $k = 2$. Gsec allows an LP solution with $y = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ by setting the two edge variables to $\frac{1}{2}$ (Fig. 2).

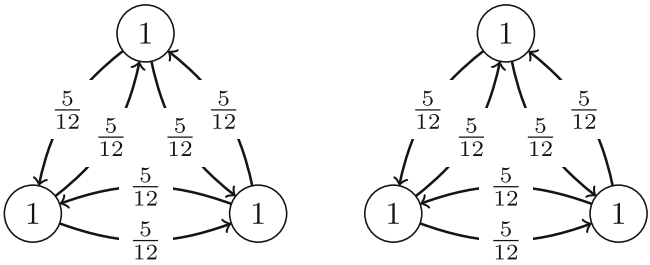

Fig. 1. A Cs LP solution for $k = 6$ on a graph with six vertices and two connected components. There are no Gsec and DCut LP solutions for this instance.

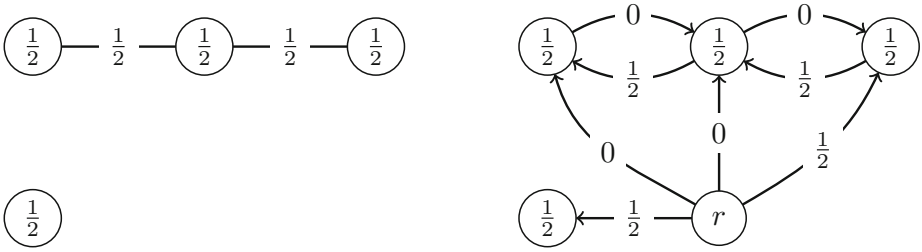

Fig. 2. A graph with four vertices with Gsec (left) and DCut (right) LP solutions for $k = 2$. There is no Cs LP solution for this choice of $y = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$.

Cs has no LP-solution with $y = (\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ because then, the isolated vertex would have to absorb an edge flow of exactly $\frac{1}{4}$ by (6a), but it has no incident edges to produce it. Note that constraint (7) is not required for this part of the proof. □

Theorem 3.2. Cs *becomes strictly stronger than* Gsec *and its equivalent formulations by adding the generalized subtour elimination constraints.*

Proof. Add the generalized subtour elimination constraints (1) to the Cs formulation (note that (4) is implied by (1)). We obtain $\text{proj}(\mathcal{P}_{\text{Cs}}) \cap \mathcal{P}_{\text{Gsec}}$ on the variable space (y, b). By Lemma 3.1, it is a proper subset of $\mathcal{P}_{\text{Gsec}}$. □

4 Dynamic Programming on a Tree Decomposition

Trees have many desirable properties: they are sparse, connected acyclic graphs that decompose into smaller trees when an edge or a vertex is removed. This often allows us to consider separated subproblems in a dynamic programming algorithm.

The tree decomposition and treewidth of a graph generalize this concept. A good introduction can be found in Kleinberg and Tardos [KT05]. Graphs similar to a tree have small treewidth and many NP-complete become tractable for graphs with bounded treewidth.

Definition 4.1. *Let $G = (V, E)$ be an undirected graph. A tree decomposition $(T, \{V_t\}_{t \in T})$ consists of a tree T on a set of vertices different from V, which we will call nodes to avoid confusion, and a subset $V_t \subseteq V$ for every node $t \in T$, which is called bag (or piece). The following properties must be satisfied:*

1. (Node coverage) Every vertex $v \in V$ belongs to at least one bag V_t.
2. (Edge coverage) For every edge $\{u, v\} \in E$, at least one bag V_t contains u and v.
3. (Coherence) Let $t_1, t_2, t_3 \in T$ where t_2 lies on a path from $t_1$to t_3. Then, if a vertex $v \in V$ belongs to both V_{t_1} and V_{t_3}, it also belongs to V_{t_2}.

Definition 4.2. *The width of a tree decomposition $(T, \{V_t\}_{t \in T})$ for a graph G is defined as $tw(T, \{V_t\}) = \max_t |V_t| - 1$.*

The treewidth of G, denoted by $tw(G)$, is the minimum width of all tree decompositions for G.

Deciding whether a graph has treewidth k is NP-complete, but solvable in linear time if k is constant [Bod96].

It is useful to introduce a special type of tree decomposition which is the analogue of a rooted binary tree:

Definition 4.3. *A tree decomposition $(T, \{V_t\}_{t \in T})$ for a graph G is nice if*

1. There is a root $r \in T$.
2. Every node t has at most 2 children.
3. If a node t is a leaf, then $|V_t| = 1$.
4. If a node t has exactly one child t', then V_t and $V_{t'}$ differ by exactly one vertex. If $|V_t| = |V_{t'}| + 1$, then t is called an *introduce* node, and if $|V_t| + 1 = |V_{t'}|$, then t is called a *forget* node.
5. If a node t has exactly two children s and x, then $V_x = V_t = V_s$, and t is called a *join* node.

Any tree decomposition can be refined into a nice decomposition with the same treewidth with a linear addition of nodes.

We now describe a dynamic programming algorithm on nice tree decompositions for the MWCIS problem. The algorithm is similar to those for the (prize-collecting) Steiner tree problem and the k-cardinality tree problem as proposed by Chimani et al. [CMZ12]. The key idea is to build the solution bottom-up (leaves to root) on tables tab_i for every bag i of the (nice) tree decomposition. Each table holds all possible sub-solutions for the vertex set of its bag. Since the treewidth is bounded, so is the size of every bag and exponentials become constants. An efficient encoding scheme for the solutions is used to save runtime and memory. Given a tree decomposition of width tw, the algorithm for the k-cardinality tree problem runs in $\mathcal{O}(B^2_{tw+2}(tw + k^2)|V|)$ time where B_i denotes the i-th Bell number.

For our problem, consider an undirected graph $G = (V, E)$ and a nice tree decomposition $(T, \{V_t\}_{t \in T})$ rooted at $r \in T$. The dynamic programming algorithm computes bottom-up a table $W(t, \mathcal{P}, a)$ of values for each node $t \in T$ and

a configuration $(\mathcal{P}, a)$: Let $\mathcal{P}$ be an arbitrary partition of a subset of V_t, $\mathcal{P} = \{P_1, ..., P_l\}$, then $W(t, \mathcal{P}, a)$ is the maximum weight of a subgraph induced by a vertices in connected components $V_1, ..., V_l \subset V_t$ such that $P_i = V_i \cap (\bigcup_{j=1}^{l} P_j)$.

The maximum of $W(t, \mathcal{P}, k)$ for $t \in T$ and $\mathcal{P}$ containing a single set is the value of the maximum connected induced subgraph. An entry $W(t, \mathcal{P}, a)$ is computed bottom-up from the children of t in the following way.

Leaf node. Let t be a leaf node with $V_t = \{v\}$. We create entries $W(t, \{\}, 0) = 0$ and $W(t, \{\{v\}\}, 1) = 0$.

Introduce node. Let t be an introduce node with child t' and $\{v\} = V_t \setminus V_{t'}$ the vertex which is added. For every $W(t', \mathcal{P}, a)$, create an entry $W(t, \mathcal{P}, a) = W(t', \mathcal{P}, a)$ and $W(t, \mathcal{P}^v, a + 1) = W(t, \mathcal{P}, a) + \sum_{uv \in E : u \in \bigcup_{P \in \mathcal{P}} P} w(u, v)$, where $\mathcal{P}^v$ is defined to be the partition $\mathcal{P}$ where $\{v\}$ is added and all sets adjacent to v are united with $\{v\}$.

Forget node. Let t be a forget node with child t' and $\{v\} = V_{t'} \setminus V_t$ the vertex which is deleted. For every configuration $(\mathcal{P}', a)$ of t', we set $W(t, \mathcal{P}, a)$ to $W(t', \mathcal{P}', a)$ where $\mathcal{P}$ is the partition $\mathcal{P}'$ with v removed from the containing set, if present at all.

Join node. Let t be a join node with children t_1, t_2. For every $(t_1, \mathcal{P}_1, a_1)$ and $(t_2, \mathcal{P}_2, a_2)$ where $\mathcal{P}_1$ and $\mathcal{P}_2$ are partitions of the same subset, set $W(t, \mathcal{P}, a_1 + a_2 - \#\mathcal{P}_1)$ to $W(t_1, \mathcal{P}_1, a_1) + W(t_2, \mathcal{P}_2, a_2)$ where $\mathcal{P}$ is the partition obtained by uniting all subsets U_1, U_2 of $\mathcal{P}_1$ such that there exist $u_1 \in U_1$ and $u_2 \in U_2$ that are in a common set in $\mathcal{P}_2$, and $\#\mathcal{P}_1$ denotes the total number of vertices contained in the partition $\mathcal{P}_1$.

5 Heuristics

5.1 Vertex Score Heuristic

Edge-weights can be approximated by vertex weights. This needs less variables in the ILP formulations and often speeds up the computation significantly. Setting $s(v) = \frac{1}{2} \sum_{uv \in E} w(uv)$ for a vertex v is a compromise between the induced weight objective and the objective which counts every edge where at least one incident vertex is selected.

5.2 A Greedy Heuristic

Our simple greedy heuristic grows the subgraph as one connected component, starting with a source vertex $s \in V$. The next vertex to be added is chosen in a greedy manner until k vertices have been found. This procedure is started once in every vertex to ensure the whole graph is explored. This does not yield the optimal result in general. A similar algorithm was given by Fischetti et al. [FHJM94].

5.3 Simulated Annealing

Simulated annealing is a local search procedure capable of escaping from local optima. In every iteration, the current solution is changed a little. This change is accepted with some probability depending on the difference in the objective values and a gradually decreasing virtual temperature.

Simulated annealing has been used in similar problems for finding subnetworks in regulatory networks [IOSS02]. For our problem, we start the simulated annealing algorithm with an arbitrary connected subgraph with k vertices. In every iteration, one randomly chosen vertex is removed. Then, a vertex adjacent to any of the remaining vertices is chosen randomly and added to the subgraph. We evaluate this new subgraph and accept the change depending on the evaluation difference to the previous subgraph and the current temperature. If the change is not accepted, it is reverted. It is easy to see that full search space is maintained (assuming the graph is connected) because there is always a sequence of changes that leads us to any given connected k-induced subgraph.

5.4 Heuristic Based on the Tree Decomposition Approach

Even though the treewidth of our input graphs is too large to use our algorithm based on the tree decomposition, we can use it in a heuristic: we create a large subgraph of the input with small treewidth and solve the problem on this graph. We compute a subgraph of small treewidth in a Kruskal-fashion: we add the edges in the order of decreasing weight as long as the treewidth stays small. As we have to compute the treewidth $|E|$ times, we use a heuristic to compute it.

We note that with this heuristic, we find much better subgraphs of small treewidth as in the approach by Fix et al. [FCBZ12].

6 Experiments

6.1 Benchmark Setting and Input

We performed tests on a real-world graph arising in the study of regulatory networks (6270 vertices, 8650 edges). This graph was formulated by the integration of the Reactome pathway knowledgebase [CMH+14]. This open-data resource for human biochemical interactions is a widespread analysis tool in which each vertex represents an encoded gene, protein or chemical compound, and each edge represents a directed biochemical interaction. Furthermore, we conducted tests on randomly generated Erdős-Rényi graphs with $|V| = 3,000$ and $|E| = 10,000$ and edge weights from a normal distribution.

The tests were carried out on an Intel Core 2 P8600 CPU with 8 GB DDR3-RAM. The tree decomposition algorithm was implemented in C++ using the TreeD library [Sub07]. All other algorithms were programmed in Java 7 [Ora12]. The integer linear programs were solved using Gurobi 5.6 [Gur14], which is free

for academic purposes. The solution of the greedy heuristic was given as an initial feasible solution for the ILP solvers, which provides them with a lower bound to the optimum value.

The strong Cohen formulation (Cs) for which the results are shown did not include Gsec or equivalent constraints. A naïve implementation of the DCut separation did not improve performance for our instances. However, according to Chimani et al. [CKLM10], advanced techniques to extract more minimum cuts from the maximum flow problem can significantly speed up the computation.

The simulated annealing algorithm was run for 100,000 steps per instance.

6.2 Quality of Solutions

We compared the objective values obtained by the different algorithms for our real-world graph. Here, we also included edges where one endpoint is chosen for the subgraph. The results are shown in Fig. 3. The tests run on randomly generated graphs showed similar results.

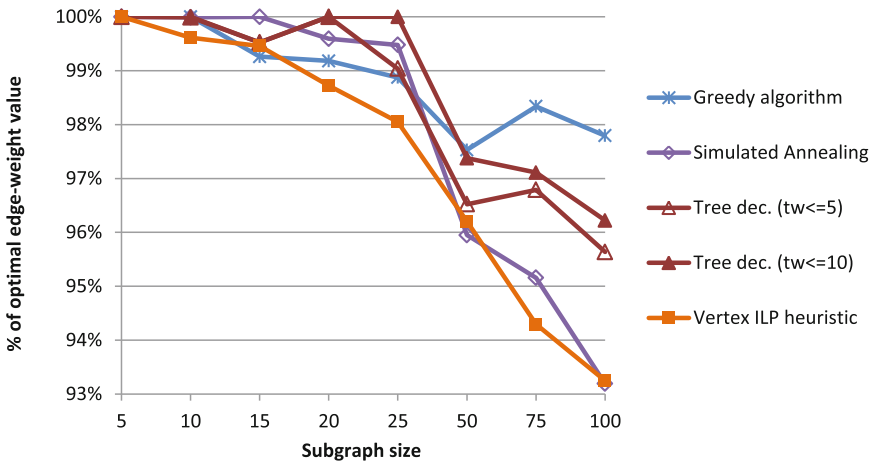

Fig. 3. Quality results of heuristics for an edge-weighted real-world graph.

To assess the biological reliability of the algorithms, we modeled an optimal biological subgraph with 15 vertices into our real-world graph. In this subgraph, the polarity (activation and inhibition) of each edge as well as its corresponding vertices were designed such that they represent a consistent biological process.

Additional instances of this modified graph were generated which exhibit several levels of additive Gaussian noise. We analyzed how well the optimal biological subgraph could be recovered from these instances. As a result, our model outreaches Backes et al. and identified more vertices in the graph for all noise levels with the induced edge weight objective (Fig. 4). Objectives that

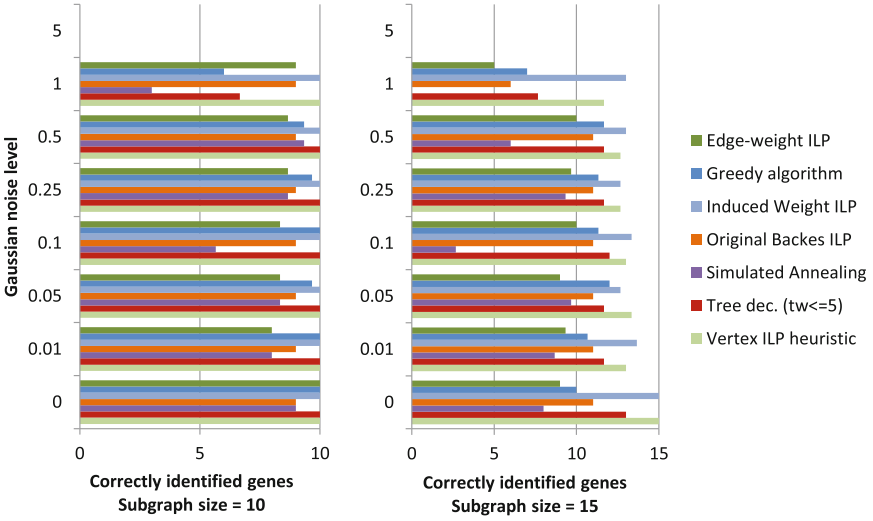

Fig. 4. Number of correctly identified genes (of 15) for $k = 10$ (left) and $k = 15$ (right) for different levels of additive Gaussian noise, averaged over three runs.

include not only induced, but all incident edges exhibit the problem that the algorithm can collect weight from crucial vertices without selecting them.

6.3 Performance Results

The benchmark results show that the integer linear programming based algorithms can compete with the greedy heuristic and simulated annealing for small values of k, but their runtimes increase quickly with the subgraph size (Fig. 5) and for worse data. The tree decomposition algorithm was generally fast for a treewidth bound of five, and its runtime is more predictable. Note that the time to build the tree decomposition with a heuristic is negligible for this bound.

Our new integer linear program performed better than the one adopted from Backes et al. for $k \geq 25$ with the edge-weight objectives, which was confirmed on the randomly generated instances. Both ILP approaches needed up to several gigabytes of RAM to solve with Gurobi, where the Backes approach usually needs more space. The space and time needed for the dynamic programming algorithm heavily depends on the treewidth bound. For a treewidth bound of five, the algorithm produced good results in minutes while needing less than 200MB of memory. For a treewidth bound of ten, it needed 5 GB, several hours, and the results improved just slightly.

The vertex-score integer linear programs were often faster than their edge-weight counterparts because they need less variables and constraints.

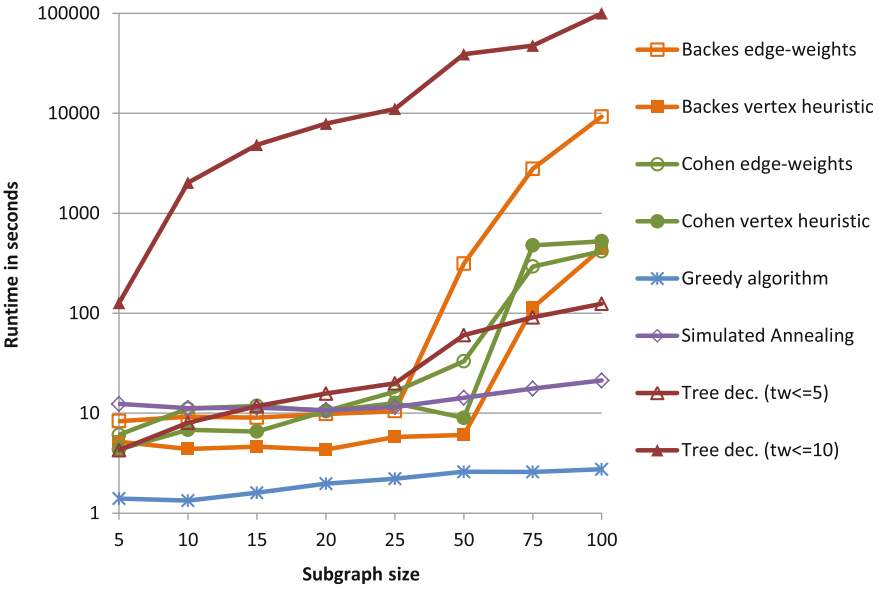

Fig. 5. Performance results of different approaches for an edge-weighted real-world graph.

7 Conclusion and Outlook

We defined a new model to compute deregulated subgraphs in a regulatory network, based on the GGEA concept introduced by Geistlinger et al. [GCK+11], that results in finding a connected induced subgraph of maximal weight.

Several algorithms were tested on the model, including a novel mixed integer linear programming method and an algorithm based on tree decompositions. The linear program can be combined with previous ILP approaches to get a stronger formulation, which will be addressed in future implementations. Due to immense treewidths of regulatory networks, the tree decomposition algorithm uses a heuristic, which initially identifies a subgraph with large weight and small treewidth. The new model yields better results than the previous approach by Backes et al. [BRK+11], as the polarity of the edges is taken into account. Our algorithms showed better performance than the adaption of the Backes approach for undirected graphs for large subgraph sizes.

References

[AK00] Arora, S., Karakostas, G.: A 2 + ϵ approximation algorithm for the k-MST problem. In: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '00, pp. 754–759. Society for Industrial and Applied Mathematics, Philadelphia (2000)

[AMLM13] Álvarez Miranda, E., Ljubić, I., Mutzel, P.: The maximum weight connected subgraph problem. In: Jünger, M., Reinelt, G. (eds.) Facets of Combinatorial Optimization, pp. 245–270. Springer, Heidelberg (2013)

[BB05] Blum, C., Blesa, M.J.: New metaheuristic approaches for the edge-weighted K-cardinality tree problem. Comput. Oper. Res. **32**(6), 1355–1377 (2005)

[BCC+10] Bhaskara, A., Charikar, M., Chlamtac, E., Feige, U., Vijayaraghavan, A.: Detecting high log-densities - an $O(n^{1/4})$ approximation for densest k-subgraph. In: Proceedings of the Forty-second ACM Symposium on Theory of Computing, STOC '10, pp. 201–210. ACM, New York (2010)

[Bod96] Bodlaender, H.L.: A linear-time algorithm for finding tree-decompositions of small treewidth. SIAM J. Comput. **25**(6), 1305–1317 (1996)

[BRK+11] Backes, C., Rurainski, A., Klau, G.W., Müller, O., Stöckel, D., Gerasch, A., Küntzer, J., Maisel, D., Ludwig, N., Hein, M., Keller, A., Burtscher, H., Kaufmann, M., Meese, E., Lenhof, H.-P.: An integer linear programming approach for finding deregulated subgraphs in regulatory networks. Nucleic Acids Res. (2011)

[BRV99] Blum, A., Ravi, R., Vempala, S.: A constant-factor approximation algorithm for the k-MST problem. J. Comput. Syst. Sci. **58**(1), 101–108 (1999)

[CKLM10] Chimani, M., Kandyba, M., Ljubić, I., Mutzel, P.: Obtaining optimal K-cardinality trees fast. J. Exp. Algorithmics **14**, 5:2.5–5:2.23 (2010)

[CMH+14] Croft, D., Mundo, A.F., Haw, R., Milacic, M., Weiser, J., Wu, G., Caudy, M., Garapati, P., Gillespie, M., Kamdar, M.R., Jassal, B., Jupe, S., Matthews, L., May, B., Palatnik, S., Rothfels, K., Shamovsky, V., Song, H., Williams, M., Birney, E., Hermjakob, H., Stein, L., D'Eustachio, P.: The reactome pathway knowledgebase. Nucleic Acids Res. **42**(D1), D472–D477 (2014)

[CMZ12] Chimani, M., Mutzel, P., Zey, B.: Improved Steiner tree algorithms for bounded treewidth. J. Discrete Algorithms **16**, 67–78 (2012)

[Coh10] Cohen, N.: Several graph problems and their LP formulation (explanatory supplement for the Sage graph library), July 2010. http://hal.inria.fr/inria-00504914

[DKR+08] Dittrich, M.T., Klau, G.W., Rosenwald, A., Dandekar, T., Müller, T.: Identifying functional modules in protein–protein interaction networks: an integrated exact approach. Bioinformatics **24**(13), 223–231 (2008)

[DPM+09] Dinu, I., Potter, J.D., Mueller, T., Liu, Q., Adewale, A.J., Jhangri, G.S., Einecke, G., Famulski, K.S., Halloran, P., Yasui, Y.: Gene-set analysis and reduction. Briefings Bioinf. **10**, 24–34 (2009)

[FCBZ12] Fix, A., Chen, J., Boros, E., Zabih, R.: Approximate MRF inference using bounded treewidth subgraphs. In: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (eds.) ECCV 2012, Part I. LNCS, vol. 7572, pp. 385–398. Springer, Heidelberg (2012)

[FHJM94] Fischetti, M., Hamacher, H.W., Jørnsten, K., Maffioli, F.: Weighted k-cardinality trees: complexity and polyhedral structure. Networks **24**(1), 11–21 (1994)

[FKP01] Feige, U., Kortsarz, G., Peleg, D.: The dense k-subgraph problem. Algorithmica **29**(3), 410–421 (2001)

[Gar96] Garg, N.: A 3-approximation for the minimum tree spanning k vertices. In: Proceedings of the 37th Annual Symposium on Foundations of Computer Science, pp. 302–309, October 1996

[GCK+11] Geistlinger, L., Csaba, G., Küffner, R., Mulder, N., Zimmer, R.: From sets to graphs: towards a realistic enrichment analysis of transcriptomic systems. Bioinformatics **27**(13), i366–i373 (2011)

[Gur14] Gurobi Optimization, Inc., Gurobi Optimizer Reference Manual (2014). http://www.gurobi.com/documentation/5.0/reference-manual/

[IOSS02] Ideker, T., Ozier, O., Schwikowski, B., Siegel, A.F.: Discovering regulatory and signalling circuits in molecular interaction networks. Bioinformatics **18**(Suppl. 1), S233–S240 (2002)

[Kho06] Khot, S.: Ruling out PTAS for graph min-bisection, dense k-subgraph, and bipartite clique. SIAM J. Comput. **36**, 1025–1071 (2006)

[KT05] Kleinberg, J., Tardos, E.: Algorithm Design. Addison-Wesley Longman Publishing Co. Inc., Boston (2005)

[Law76] Lawler, E.: Combinatorial Optimization: Networks and Matroids. Holt, Rinehart and Winston, New York (1976)

[Lju04] Ljubić, I.: Exact and memetic algorithms for two network design problems. Ph.D. thesis, Technische Universität Wien (2004). https://www.ads.tuwien.ac.at/publications/bib/pdf/ljubicPhD.pdf

[MTZ60] Miller, C.E., Tucker, A.W., Zemlin, R.A.: Integer programming formulation of traveling salesman problems. J. ACM **7**(4), 326–329 (1960)

[Ora12] Oracle corporation. Java Platform, Standard Edition 7 (2012). http://docs.oracle.com/javase/7/docs/api/

[QadCML10] Quintão, F.P., da Cunha, A.S., Mateus, G.R., Lucena, A.: The k-cardinality tree problem: reformulations and lagrangian relaxation. Discrete Appl. Math. **158**(12), 1305–1314 (2010)

[QdCM08] Quintão, F.P., da Cunha, A.S., Mateus, G.R.: Integer programming formulations for the k-cardinality tree problem. Electron. Notes Discrete Math. **30**, 225–230 (2008)

[STM+05] Subramanian, A., Tamayo, P., Mootha, V.K., Mukherjee, S., Ebert, B.L., Gillette, M.A., Paulovich, A., Pomeroy, S.L., Golub, T.R., Lander, E.S., Mesirov, J.P.: Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. Proc. Natl Acad. Sci. U.S.A. **102**(43), 15545–15550 (2005)

[Sub07] Sathiamoorthy Subbarayan. TreeD: A Library for Tree Decomposition, July 2007. http://itu.dk/people/sathi/treed/

[ZWCA08] Zhao, X.-M.M., Wang, R.-S.S., Chen, L., Aihara, K.: Uncovering signal transduction networks from high-throughput data by integer linear programming. Nucleic Acids Res. **36**(9), e48 (2008)