

# Heuristic Algorithms for the Maximum Colorful Subtree Problem

**Kai Dührkop**

Chair for Bioinformatics, Friedrich-Schiller-University, Jena, Germany  
kai.duehrkop@uni-jena.de

**Marie A. Lataretu**

Chair for Bioinformatics, Friedrich-Schiller-University, Jena, Germany  
marie.lataretu@uni-jena.de

**W. Timothy J. White**

Chair for Bioinformatics, Friedrich-Schiller-University, Jena, Germany, and  
Berlin Institute of Health, Berlin, Germany  
tim.white@bihealth.de

 <https://orcid.org/0000-0002-1997-0176>

**Sebastian Böcker**

Chair for Bioinformatics, Friedrich-Schiller-University, Jena, Germany  
sebastian.boecker@uni-jena.de

 <https://orcid.org/0000-0002-9304-8091>

---

## Abstract

In metabolomics, small molecules are structurally elucidated using tandem mass spectrometry (MS/MS); this computational task can be formulated as the Maximum Colorful Subtree problem, which is NP-hard. Unfortunately, data from a single metabolite requires us to solve hundreds or thousands of instances of this problem – and in a single Liquid Chromatography MS/MS run, hundreds or thousands of metabolites are measured.

Here, we comprehensively evaluate the performance of several heuristic algorithms for the problem. Unfortunately, as is often the case in bioinformatics, the structure of the (chemically) true solution is not known to us; therefore we can only evaluate against the optimal solution of an instance. Evaluating the quality of a heuristic based on scores can be misleading: Even a slightly suboptimal solution can be structurally very different from the optimal solution, but it is the structure of a solution and not its score that is relevant for the downstream analysis. To this end, we propose a different evaluation setup: Given a set of candidate instances of which exactly one is known to be correct, the heuristic in question solves each instance to the best of its ability, producing a score for each instance, which is then used to rank the instances. We then evaluate whether the correct instance is ranked highly by the heuristic.

We find that one particular heuristic consistently ranks the correct instance in a top position. We also find that the scores of the best heuristic solutions are very close to the optimal score; in contrast, the structure of the solutions can deviate significantly from the optimal structures. Integrating the heuristic allowed us to speed up computations in practice by a factor of 100-fold.

**2012 ACM Subject Classification** Applied computing → Computational biology, Applied computing → Metabolomics / metabonomics

**Keywords and phrases** Fragmentation trees, Computational mass spectrometry

**Digital Object Identifier** 10.4230/LIPIcs.WABI.2018.23

**Funding** WTJW funded by Deutsche Forschungsgemeinschaft (grant BO 1910/9).



© Kai Dührkop, Marie A. Lataretu, W. Timothy J. White, and Sebastian Böcker;  
licensed under Creative Commons License CC-BY

18th International Workshop on Algorithms in Bioinformatics (WABI 2018).

Editors: Laxmi Parida and Esko Ukkonen; Article No. 23; pp. 23:1–23:14

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

## 1 Introduction

Metabolomics characterizes the collection of all metabolites in a biological cell, tissue, organ or organism using high-throughput techniques [10]. Liquid Chromatography Mass Spectrometry (LC-MS) is one of the predominant experimental platforms for this task [1] and can detect compounds at the attogram level [8]. Today, a major challenge is to determine the identities of the thousands of metabolites detected in one LC-MS run. This is also true for related fields such as natural products research [16,21], biomarker discovery [21], environmental science [13], or food science. Tandem mass spectrometry (MS/MS) is used to derive information about the structure of a metabolite by fragmenting the molecule and recording the masses of its fragments. Interpretation of the hundreds to thousands of MS/MS spectra generated in a single LC-MS run remains a bottleneck in the analytical pipeline [19]. MS/MS data is usually searched against spectral libraries [17], but only a small number of metabolites (around 2%) can be identified in this manner [4].

Fragmentation trees were introduced in 2008 [3] and were initially targeted towards identifying the molecular formula of the unknown small molecule. Later, it was shown that the structure of fragmentation trees contains valuable information for structural elucidation of the underlying molecule [11]. In particular, CSI:FingerID combines fragmentation tree computation with multiple kernel learning on these trees [6,15], and is the currently best-performing method for searching MS/MS data in *structure* databases [14]. Computing an optimum fragmentation tree naturally leads to the MAXIMUM COLORFUL SUBTREE problem [2,3]. Unfortunately, this problem is NP-hard and also hard to approximate [7,12]. Algorithms exist to solve the problem either heuristically [12] or exactly [3,12,20]. The problem is a variant of the well-studied GRAPH MOTIF problem [5,9].

Optimization problems in bioinformatics research are designed so that the optimal solution (with regards to the objective function) is “similar” to the true solution, such as the biologically correct phylogenetic tree, the “true” sequence alignment, etc. Unfortunately, many of these optimization problems are NP-hard; furthermore, the true solution is often not known to us. Approximation algorithms are algorithms for (usually) NP-hard problems with provable guarantees on the distance of the returned solution to the optimal one. But this provable guarantee is only for the objective function; in bioinformatics research, we are rarely interested in the objective function value beyond using it to find the optimal solution. Heuristics in bioinformatics are usually designed to find solutions structurally similar to the optimum solution or, even better, the true solution. This makes it intrinsically difficult to evaluate the performance of these heuristics, as we have to define a measure on the structural similarity between the heuristic solution and the true solution; furthermore, the true solution has to be known, which is often not the case.

We will use an alternative approach to evaluate the performance of a heuristic: For many applications, one biological instance results in many computational instances, corresponding to candidates or hypotheses; the score of the solution to each instance is used to rank its corresponding hypothesis. Although the true solution may not be known, we may have information regarding the correct candidate or hypothesis. To this end, we can evaluate a heuristic based on its ability to top-rank the correct candidate.

We propose several heuristics for the MAXIMUM COLORFUL SUBTREE problem, and evaluate these heuristics with regards to their ranking quality. We find that one particular heuristic allows us to quickly shrink the set of plausible candidates (molecular formulas of the precursor molecule). This can be used as a filter, such that optimum solutions have to be sought only for a (preferably small) subset of candidates. We also evaluate whether the structure of the constructed solutions is similar to the optimum solution.

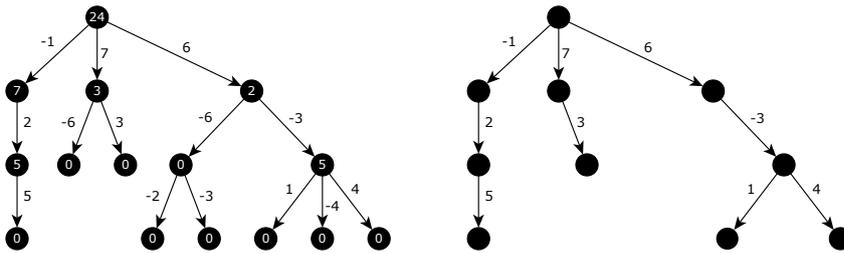
## 2 Fragmentation trees and the Maximum Colorful Subtree problem

Tandem mass spectrometry selects ions with a particular mass, fragments these ions by collision with neutral molecules such as argon or nitrogen, then records the masses of the resulting fragments. Assuming that only identical copies with identical structure are selected, we say that one *precursor ion* with some *precursor mass* was selected. Mass spectrometry measures mass-to-charge instead of mass, but we may assume that all small molecules carry a single charge. Masses of fragments of the molecule are recorded as *peaks* in the MS/MS spectrum.

Details of how to transform the MS/MS spectrum of an (unknown) compound into one or more instances of the MAXIMUM COLORFUL SUBTREE problem have been published elsewhere [2,3]; we briefly recapitulate the process. We consider all molecular formulas from some ground set, such as, all molecular formulas built from the elements CHNOPS. We *decompose* the precursor mass into all possible candidate molecular formulas from this ground set; each candidate molecular formula generates one instance (graph). For each instance, we decompose the fragment peaks in the MS/MS spectrum, ensuring that each fragment molecular formula is a subformula of the candidate molecular formula for the precursor mass. These molecular formulas constitute the nodes of a graph; each node is colored by the peak it stems from. An edge, representing a possible fragmentation event, is present between molecular formulas  $u, v$  if and only if  $v$  is a proper subformula of  $u$ . Now, both nodes and edges receive a certain weight [2], based both on prior knowledge (e.g., distribution of loss masses) and the data (e.g., mass difference between a peak and its hypothetical molecular formula); but as pointed out in [3], we may encode both kinds of weights using only edge weights. SIRIUS 4 default weights are used, see [2]. The maximum colorful subtree within each instance is computed, and its weight is used to rank the corresponding candidate molecular formula of the precursor peak.

We can formulate fragmentation tree computation as a stochastic optimization problem, where we try to “best explain” the observed data (the fragmentation spectrum of a small molecule obtained through tandem mass spectrometry) under a hypothesis (the molecular formula of the small molecule, represented by the root vertex) using a maximum a posteriori estimator. The likelihood function rewards hypotheses that explain high-intensity peaks. Each peak in the fragmentation spectrum must be explained at most once, in accordance with the parsimony principle. Böcker and Dührkop [2] then show that one can find the maximum a posteriori hypothesis by solving a particular MAXIMUM COLORFUL SUBTREE instance, where edge weights are logs of probabilities of fragmentation events under the model.

We now describe the formal MAXIMUM COLORFUL SUBTREE optimization problem. Let  $G = (V, E)$  be a node-colored, rooted, directed acyclic graph (DAG) with root  $r \in V$  and edge weights  $w : E \rightarrow \mathbb{R}$ . Let  $C(G)$  be the set of colors used in  $G$ , let  $c(v) \in C(G)$  be the color assigned to node  $v \in V$ , and let  $c(U) := \{c(v) \mid v \in U\}$  for  $U \subseteq V$ . We will consider subtrees  $T = (V_T, E_T)$  of  $G$  rooted at  $r$ . We say that  $T$  is *colorful* if all of its nodes have different colors. The MAXIMUM COLORFUL SUBTREE problem asks to find a colorful  $r$ -rooted subtree  $T$  of  $G$  of maximum weight. We may assume that  $G$  is (weakly) connected and that  $r$  is the unique source of  $G$ , as we can remove all nodes from the graph which cannot be reached by a path from the root  $r$ , without changing the optimal solution. We say that a subgraph  $G' \subseteq G$  is *full* if  $v \in G'$  implies that every edge and vertex reachable from  $v$  in  $G$  is also in  $G'$ .



■ **Figure 1** Illustration of the Remove Dangling Subtrees (RDS) postprocessing. Left: Input tree, where each node  $v$  is labeled by its score  $D[v]$ . Right: Output tree of weight 24.

We note that previous work on the problem also makes the assumption of a single source, albeit usually implicitly [3, 11, 12, 20]. From an algorithmic standpoint, problem variants with or without a given root are “basically equivalent”: Given an algorithm that does not assume a fixed root  $r$ , we can solve an instance of the problem variant with root  $r$  by introducing a superroot  $r^*$  connected solely to  $r$ , sufficiently large edge weight  $w(r^*, r)$  and a new color for  $r^*$ . For the reverse direction, we solve the problem for every  $r \in V$ , then choose the best solution.

There are two peculiarities when computing fragmentation trees that are different from the general problem, and that we will make use of here. First, any DAG used for fragmentation tree computation is *transitive*: That is,  $uv \in E$  and  $vw \in E$  implies  $uw \in E$ . Second, a coloring  $c : V \rightarrow C(G)$  of DAG  $G = (V, E)$  is *order-preserving* if there is an ordering ‘ $\prec$ ’ on the colors  $C(G)$  such that  $c(u) \prec c(v)$  holds for every edge  $uv$  of  $G$  [7]. Computing fragmentation trees naturally results in order-preserving colors, as nodes can be colored by the fragment mass that is responsible for this node, and edges exist only between nodes from larger to smaller masses. The MAXIMUM COLORFUL ARBORESCENCE problem [7] asks to find a rooted colorful subtree  $T$  of  $G$  of maximum weight, where  $G$  is a DAG with order-preserving colors and edge weights. See [7] for numerous complexity results. Here, we will stick with the name “MAXIMUM COLORFUL SUBTREE problem”, but nevertheless assume that the coloring is order-preserving, unless indicated otherwise.

### 3 Heuristics for the Maximum Colorful Subtree problem

The following postprocessing methods can be applied to a tree  $T = (V_T, E_T)$  *after* any heuristic: The **Remove Dangling Edges** (RDE) postprocessing iteratively removes edges  $uv$  from  $T$ , where  $v$  is a leaf and  $w(uv) < 0$ ; this is repeated until no more such edges are found. In contrast, the **Remove Dangling Subtrees** (RDS) postprocessing does not consider a single edge at a time, but rather full subtrees: Each node  $u \in V_T$  is scored by the maximum weight of any full subtree rooted in  $u$ . Score  $D[u]$  can be computed using dynamic programming:

$$D[u] := \sum_{uv \in E_T} \max\{0, w(u, v) + D[v]\}$$

Clearly,  $D[u] \geq 0$ . For each edge  $uv$  with  $w(u, v) + D[v] < 0$  we remove  $uv$  and the subtree below it. Both postprocessings can be computed in  $O(|V_T|)$  time using a tree traversal, as every edge is considered once and  $|E_T| = |V_T| - 1$ . Figure 1 shows an example of the RDS postprocessing.

We now present heuristics for finding a colorful subtree with root  $r$  in a transitive DAG with order-preserving coloring and unique source  $r$ .

- *Kruskal-style*. This heuristic sorts all edges of the graph by decreasing edge weight, then iteratively adds edges from the sorted list, ensuring that the growing subgraph is colorful and that each node has at most one incoming edge. Since  $r$  is the unique source of  $G$ , and since  $G$  is transitive, this will ultimately result in a colorful subtree of  $G$ . This heuristic is similar to Kruskal’s algorithm for computing a minimum spanning tree; it was called “greedy heuristic” in [3].
- *Prim-style*. This heuristic progresses similarly to Prim’s algorithm for calculating a minimum spanning tree: The tree  $T = (V_T, E_T)$  initially contains only the root  $r$  of  $G$ . In every step, we consider all edges  $uv$  with  $u \in V_T$  and  $v \notin V_T$  such that  $c(v) \notin c(V_T)$ ; among these, we choose the edge with maximum weight and add it to the tree. We repeat until all colors in the graph are used in the tree; recall that  $G$  is transitive, so  $rv \in E$  for each  $v \neq r$ . We explicitly do not quit when adding the first negative-weight edge  $uv$ , as the newly reached node  $v$  may allow us to later add other edges with positive weight. The Prim-style heuristic will usually result in a different tree than the Kruskal-style heuristic, due to the colorfulness constraint.
- *Insertion*. This heuristic is a modification of the “insertion heuristic” from [12]. We again start with a tree  $T = (V_T, E_T)$  containing only the root  $r$  of  $G$ . The heuristic greedily attaches nodes labeled with unused colors. For every node  $v$  with  $c(v) = c'$  unused, and every node  $u$  already part of the solution, we calculate how much we gain by attaching  $v$  to  $u$ . To calculate this gain  $I(u, v)$ , we take into account the score of the edge  $uv$  as well as the possibility of rerouting other outgoing edges of  $u$  through  $v$ :

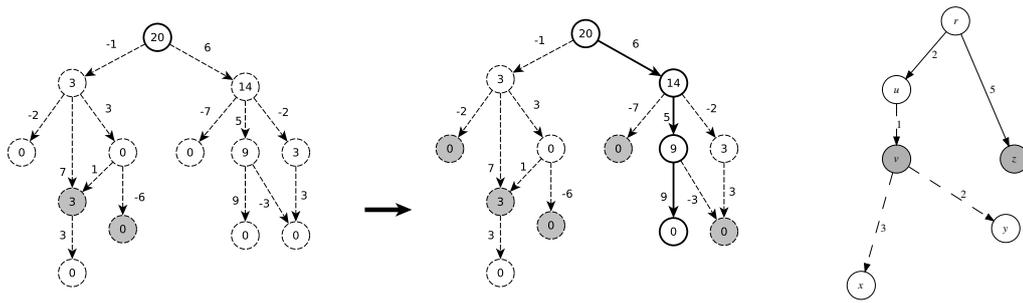
$$I(u, v) := w(uv) + \sum_{x \in V_T, w(vx) > w(ux)} (w(vx) - w(ux))$$

where we assume  $w(uv) = -\infty$  if  $uv \notin E$ . The node with maximum gain is then attached to the partial solution, and edges are rerouted as required. See [12] for details; different from there, we do not iterate over colors in some fixed order but instead, consider all unused colors in every step.

- *Top-down*. The top-down heuristic [3] is also greedy, but always adds paths beginning at the root to the partial solution. The partial solution initially contains only the root  $r$  of  $G$ . The heuristic greedily constructs a path starting at the root which is added to the partial solution; the next node of the path is chosen so that it maximizes the weight of the added edge, simultaneously ensuring that the partial solution remains colorful and does not violate the tree property. If no such edge exists, the algorithm restarts at the root, and searches for another path. It terminates if no edge at the root can be selected. In the resulting tree, all internal nodes but the root have exactly one child. This heuristic extends even simpler heuristics that attach all nodes to the root, which have been in frequent use for molecular formula determination from MS/MS data.
- *Critical Path*<sup>1</sup>. Again, we iteratively build a tree  $T = (V_T, E_T)$ ; initially, the partial solution  $T$  contains only the root  $r$  of  $G$ . The score  $S[u]$  of a node  $u \in V$  is the maximum weight of a path  $p$  from  $u$  to any node  $v$ , such that  $c(p) \cap c(V_T) \subseteq \{c(u)\}$ ; that is, the path does not use nodes with colors already present in the tree, except for the color of the starting node. We can compute  $S[u]$  using the recurrence

$$S[u] := \max_{uv \in E, c(v) \notin c(V_T)} \{0, S[v] + w(uv)\} \quad (1)$$

where we use that the coloring of  $G$  is order-preserving, since in that case no two nodes of the path have the same color. We assume  $\max \emptyset = 0$  when computing (1). We iterate over the ordered colors  $c$  in reverse order, computing  $S[u]$  for all nodes  $u$  of color  $c$ . The



■ **Figure 2** Left: Example for the *Critical Path*<sup>1</sup> heuristic. Nodes are labeled by score, solid lines show the tree, dashed lines the rest of the graph. Grayed-out nodes have colors already used in the subtree. Right: An example input graph for which *Critical Path*<sup>1</sup> produces a better tree than *Critical Path*<sup>2</sup>. Nodes  $v$  and  $z$  are the same color; all other nodes have distinct colors. The two solid edges are the suboptimal tree output by *Critical Path*<sup>2</sup>. *Critical Path*<sup>1</sup> initially chooses the path  $ruvx$  for a score of 6, then adds  $vy$  for a total of 8. *Critical Path*<sup>2</sup> begins in the same way by choosing the first edge  $ru$  of the heaviest path for a score of 2, but in its second step it chooses the weight-5 edge  $rz$ , as the heaviest path starting with  $uv$  has weight 4. No further edges can be added, so the total weight is 7.

*critical path*  $p$  of maximum score can be found by backtracing from the maximum entry  $S[u]$  with  $u \in V_T$ . We add  $p$  to  $T$ , then iterate, recomputing  $S$  for the new set of used colors  $c(V_t)$ . See Figure 2 (left) for an example.

- *Critical Path*<sup>2</sup>. This heuristic also relies on critical paths, but adds, in each iteration, only the first edge of the critical path to the partial solution. We note that this heuristic does not dominate the *Critical Path* heuristic, meaning that in certain cases, the subtree computed by this heuristic has smaller weight than that computed by the *Critical Path* heuristic; see Figure 2 (right) for an example.
- *Critical Path*<sup>3</sup>. This heuristic combines the Insertion heuristic with the *Critical Path* heuristic: In each step the heuristic chooses the edge  $uv$  with  $u \in V_T$  that maximizes the sum of critical path score and insertion score  $S[u] + I(u, v)$ .
- *Maximum*. All heuristics compute lower bounds of the maximum score; therefore the maximum score over all heuristic solutions is also a lower bound.

### 3.1 Time complexity of the heuristics

Let  $n := |V|$ ,  $m := |E|$ , and  $k := |C(G)|$ . Clearly,  $k \leq n$  and in applications, we usually have  $k \ll n$ . Furthermore,  $|V_T| \leq k$  holds for the returned subtree  $T = (V_T, E_T)$ .

- It is easy to check that the Kruskal-style heuristic has time complexity  $O(m \log n)$  for sorting all edges according to weight. Connectivity testing can then be performed in sub-logarithmic time per considered edge using a union-find data structure [18]; checking for colorfulness is easily accommodated by initially placing all nodes of the same color in the same component. The overall time complexity thus remains  $O(m \log n)$ . Similarly, the Prim-style heuristic requires  $O(m \log n)$  time.
- For the Insertion heuristic, computing the gains  $I(u, v)$  for a particular  $v$  and for all  $u$  requires  $O(k^2)$  time, since  $u, x \in V_T$ . Hence, choosing one  $v$  to attach to the growing tree requires  $O(k^2 n)$  time, resulting in  $O(k^3 n)$  total running time.

However there exists a more complicated yet faster implementation for this heuristic: For each  $v \in V$ , we maintain two scores,  $in(v)$  and  $out(v)$ , which correspond to the two terms on the RHS of the definition of  $I(u, v)$ . Specifically,  $in(v) = \max_{u \in V_T} w(uv)$ , and

$out(v) = \sum_{x \in V_T} \max\{0, w(vx) - w(p_T(x), x)\}$ , where  $p_T(x)$  is the parent of  $x$  in  $T$  for all  $x \in T$ . To choose the next node to insert, we look for the node  $v \in V$  maximizing  $in(v) + out(v)$ , ignoring nodes of already-used colors, which takes  $O(n)$  time (and could in practice often finish early if we search in decreasing order of one of these terms, and know an upper bound on the other). We then perform a single  $O(k)$ -time scan to find its optimal parent in the tree, and then make two further updates: First, for all  $u \in V$ , set  $in(u) \leftarrow \max\{in(u), w(vu)\}$  and  $out(u) \leftarrow out(u) + \max\{0, w(uv) - w(p_T(v), v)\}$ . Second, for all  $x \in V_T$ , check whether the incoming edge  $yx$  (i.e.,  $y = p_T(x)$ ) can be improved by rerouting via  $v$ ; if so, delete  $yx$ , insert  $vx$  and for all  $u \in V$  such that  $w(yx) \leq w(ux)$ , set  $out(u) \leftarrow \max\{0, out(u) - (w(vx) - w(yx))\}$ . The second update, which dominates, needs  $O(kn)$  time per inserted node, for  $O(k^2n)$  overall.

- The Top-down heuristic searches at most  $k$  times for the maximum weight edge leaving a node; since there are  $O(n)$  such edges, the running time is  $O(kn)$ .
- For the Critical Path<sup>1</sup> heuristic, we need  $O(m)$  time to compute the  $S[u]$  values and to identify the path of maximum weight. This is repeated at most  $k - 1$  times, resulting in a total running time of  $O(km)$ . The same holds true for Critical Path<sup>2</sup>. For Critical Path<sup>3</sup> we can again maintain an  $out(v)$  table that contains the score bonus we get for attaching a node in the intermediate tree as a child of  $v$  and deleting its previous incoming edge. After each insertion of an edge into the intermediate tree we have to perform the two update operations on  $out$  which takes  $O(kn)$  time per insertion. In total we need  $O(k^2n + km)$  time to compute Critical Path<sup>3</sup>. In applications,  $k$  is very small and  $n \ll m$ , so the  $O(km)$  part for calculating the critical paths requires most of the computation time.

### 3.2 Computing the $k$ -best fragmentation trees exactly

Even if we do not trust the structural quality of the heuristic solution, the above heuristics allow us to speed up fragmentation tree computation: We first select a single candidate (molecular formula of the precursor) using one of the heuristics, then compute the optimal solution for this instance using an exact method [3, 12, 20]. In practice, this approach has two shortcomings: Even though certain heuristics show a very good performance in selecting the correct molecular formula (see below), this correct answer is not known to us in application; but we will observe that the computed fragmentation tree will often not be the optimum fragmentation tree, if we also consider other molecular formula candidates and corresponding instances.

Even worse, it is usually not sufficient in application to select a single best candidate using the heuristic, then re-compute the fragmentation tree for the corresponding instance. Instead, we usually want to know optimal fragmentation trees for the  $k$  best-scoring candidates. This is independent of whether results are reported to the user, who wants to use fragmentation tree structure to survey if computations and, hence, the assigned precursor molecular formula are trustworthy; or, if we perform some downstream computational analysis based on fragmentation tree structure, such as CSI:FingerID [6]. In particular for “large” metabolites with mass beyond 600 Dalton, this is necessary because neither the heuristics nor the exact method will always allow us to find the correct candidate; only by considering several candidates can we be sufficiently sure that the correct answer is present.

We propose the following heuristic to compute optimum fragmentation trees for the  $k$ -best molecular formula candidates: First, we compute heuristic solutions for all candidates, and order molecular formula candidates according to the heuristic score. Next, we compute

optimum fragmentation trees for the  $k$  best candidates; for small  $k$ , we can instead choose some fixed parameter, such as 10 candidates. We estimate the maximum  $\Delta \geq 0$  of differences between the score of the optimum solution and the corresponding heuristic solution, using those candidates for which we know the exact solution. We now assume that the score difference is upper-bounded by  $\Delta$  for all candidates. We continue to process candidates and compute optimum fragmentation trees from the sorted list, updating the  $k$ -best candidates and the corresponding score threshold; we stop computations when the heuristic score of a candidate plus  $\Delta$  is smaller than the current score threshold. Clearly, our assumption made above may be violated for certain inputs, making this method a heuristic.

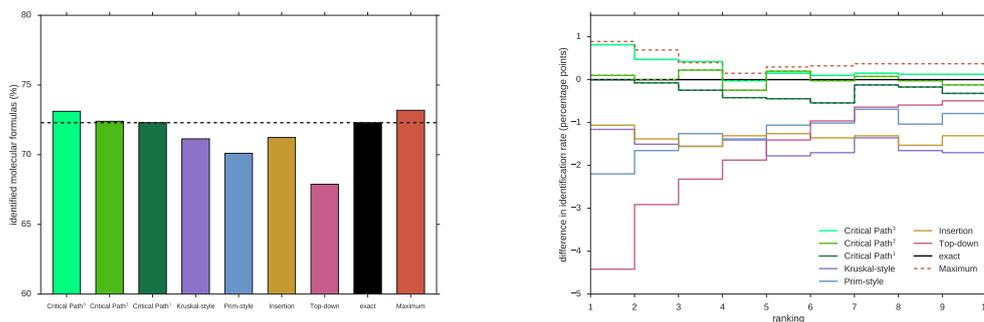
## 4 Data and Instances

To evaluate whether a heuristic is capable of ranking the correct molecular formula in the top position, we have to use reference data where the true compound structure is known for each MS/MS measurement. We use reference compounds from GNPS [19]; each reference compound is one instance, corresponding to several graphs (for the different molecular formula candidates of the precursor mass) we have to solve. We then filter instances: For example, we require a mass accuracy of 10 ppm (parts per million), and discard compounds where the precursor mass is missing or outside this mass range. All details can be found in [2]. This leaves us with 4 050 compounds, each of which is then transformed into typically many instances of the MAXIMUM COLORFUL SUBTREE problem. Each reference compound resulted in between 1 and 21 748 candidate molecular formulas, with median 53 and average 263.8 candidates. To avoid proliferating running times, we consider only the 60 most intense peaks in a MS/MS spectrum that can be decomposed, which is again SIRIUS 4 default behavior. We fix the SIRIUS tree size parameter, which is usually adapted at runtime, at  $-0.5$ . In addition, we switch off SIRIUS's spectral recalibration.

## 5 Results

We applied all but the Critical Path heuristics using the RDS postprocessing. We do not evaluate the RDE postprocessing, as it is dominated by RDS (that is, the score is at least as good, in all cases) which, in turn, dominates solutions without postprocessing. Furthermore, both kinds of postprocessing are very fast in practice. For the Critical Path heuristics, RDS cannot improve a solution for variants 1 and 2; while it is in principle possible that it could improve a solution for variant 3, this is very unlikely, and to keep results of the Critical Path heuristics consistent, we disabled the RDS postprocessing for all 3 variants. All heuristics were implemented in Java 8. For the exact method, we use the Integer Linear Program (ILP) from [12] with the CPLEX solver 12.7.1 (IBM, <https://www.ibm.com/products/ilog-cplex-optimization-studio>).

First, we evaluated the power of the different heuristics to rank the correct answer (molecular formula) at the top position; see Fig. 3 (left). We also compared against the exact solution. We observe similar identification rates for the Critical Path heuristics, the Maximum heuristic and the exact method. To test whether this trend is true not only for the top rank, but also for the top  $k$  ranks, we also evaluated how often any method is capable to rank the correct answer in its top  $k$ , for varying  $k$ ; see Fig. 3 (right). Identification rates differ much more strongly when varying  $k$  for one method than for different methods and one  $k$ ; to ease interpretation, we normalize identification rates by subtracting the identification rate of the exact method. We see that all heuristics but for the three Critical Path variants result

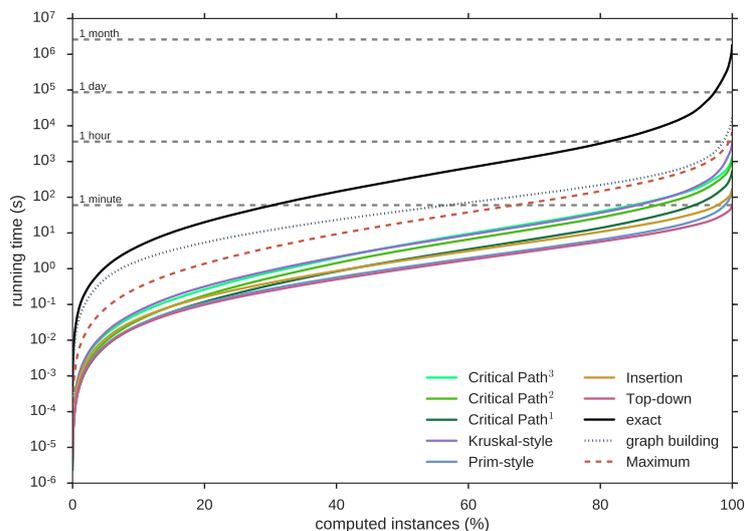


■ **Figure 3** Performance evaluation, finding the correct molecular formula. Left: *Percentage* of compounds where the correct point molecular formula received the highest score. Note the zoom of the y axis. Right: *Percentage point difference* against exact computations; how often is the correct answer part of the top  $k$  output of each method? Note that “Maximum” is one of the heuristics.

in inferior rankings, losing one or more percentage points for most ranks. In contrast, the Critical Path heuristics rank solutions with power comparable to the exact method, and the latter two variants often outperform the exact method. Somewhat surprisingly, the maximum over all heuristics performs even better than the best heuristic.

Second, we compared running times of the different methods; see Fig. 4. Running times were measured using a single thread on an Intel E5-2630v3 at 2.40 GHz with 64 GB RAM. The total running time for the exact methods over all instances is almost one month, underlining the importance of speeding up computations. But also note that solving all instances exactly requires only about 100-fold the time required for constructing the instance graphs. For each method, we sorted all instances by running time; we then determined how much time is required to solve the “easiest”  $x\%$  of instances for that method, for each  $0 \leq x \leq 100$ . Generally, the ordering of instances is different for each method. For all methods, we observe that the “hardest” 5% of the instances are responsible for most of the total running time; this has been observed before [2, 12]. In comparison to the exact method, all heuristics are very fast – at least two orders of magnitude faster. In particular, *each* heuristic is faster than the method for constructing the instance graphs; running all heuristics, as required for the Maximum heuristic, requires about the same time as the graph construction. Comparing heuristics’ running times, we see that the Kruskal-style heuristic is slowest, and that the first variant of the Critical Path heuristic is faster in practice than variants 2 and 3, but not significantly.

Third, we compared the scores reached by the different heuristics to the scores of the exact solutions; see Fig. 5 (left). For each compound, we only considered the instance of the MAXIMUM COLORFUL SUBTREE that corresponds to the true candidate molecular formula. We report scores relative to the exact solution (at 100%), and sorted instances with respect to this relative score. In the resulting plot, it is not obvious which of the heuristics “Insertion”, “Kruskal-style” and “Critical Path<sup>1</sup>” should be preferred. We see that Critical Path<sup>3</sup> heuristic and, hence, the maximum of all methods are able to compute (almost) optimal solutions for about 80% of the instances. In turn, this means that even for these methods which perform extremely well in ranking the correct answer, we miss the optimal solution in about 20% of the instances. In addition, we compared scores of the Critical Path<sup>3</sup> heuristic against the exact method in more detail, see Fig. 5 (right): We see that for instances where the heuristic does not find the optimal solution, the computed solution is only “slightly suboptimal” with



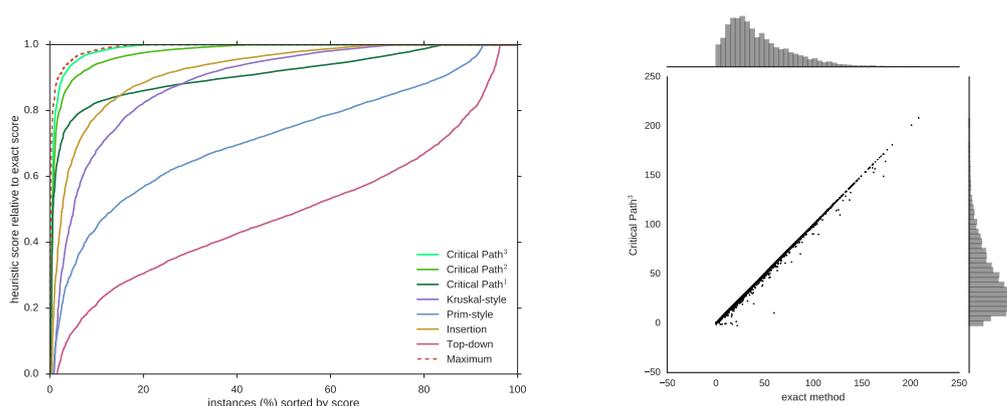
■ **Figure 4** Running times of the different methods. One instances consists of all graphs generated for one compound in the dataset, considering all decompositions of the precursor mass. For each method, instances are sorted with respect to running time, and we report amortized running times. We also report running times for constructing the instance DAGs and for the exact method.

respect to its score. In fact, Pearson correlation between the two measures is  $+1.00$ .

Fourth, we evaluate the solution structure quality of the Critical Path<sup>3</sup> heuristic. Unfortunately, the “true fragmentation tree” cannot be determined experimentally [11]. To this end, we compare heuristic tree structures against tree structures computed using the exact method. For each compound, we restrict the comparison to the true candidate molecular formula; for other candidate molecular formulas, the optimal tree cannot possibly be the “true fragmentation tree”. See Fig. 6. For tree sizes, we observe rather large deviations between heuristic and optimal trees; in contrast, the overall distribution of tree sizes is highly similar. But if we compare tree structures, we observe much larger differences between the Critical Path<sup>3</sup> heuristic and the exact method: We measure structural similarity comparing either the set of node labels (fragments) or the set of edge labels (losses) of the two trees. We estimate the similarity of two (finite) sets  $A, B$  using the *Jaccard similarity coefficient*  $J(A, B) = |A \cap B| / |A \cup B| \in [0, 1]$ . We observe that more than 20% of the heuristic trees differ from the corresponding optimal tree; for at least 10%, this difference is significant.

## 6 Conclusion

We have presented heuristics for the MAXIMUM COLORFUL SUBTREE problem. Our evaluation shows that the Critical Path<sup>3</sup> heuristic is well-suited for choosing the correct candidate molecular formula, when applied to tandem mass spectrometry data of small molecules. Our evaluation sidesteps the catch-22 that we want to evaluate solutions based on structure and not score when, at the same time, the correct solution structure is not known. Even when the heuristic returns a suboptimal solution, the score is usually very close to the optimal score. Furthermore, the heuristics allow us to rank candidates, and to restrict computation of exact solutions to the top-ranked candidates (Sec. 3.2). In application, this combination resulted in significant speedups without sacrificing fragmentation tree quality.



■ **Figure 5** Left: Relative scores of the heuristics. For each MAXIMUM COLORFUL SUBTREE instance, we consider the relative score in comparison to the exact method at 100%. For each method, instances are sorted with respect to relative score. Right: Comparison of the score of the Critical Path<sup>3</sup> heuristic in comparison to the optimal score of the instance. For each compound, we consider only the true molecular formula candidate.

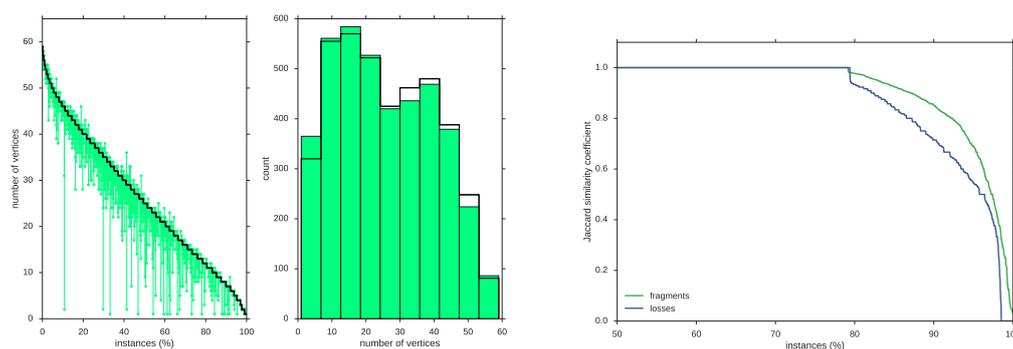
The structure of the heuristic tree deviates significantly from the optimal tree for more than 20% of the instances. We therefore argue against using this tree for downstream analysis, such as machine learning [6, 15]. A back-of-the-envelope calculation indicates the problem: If we assume that 20% of the heuristic trees are “structurally faulty”, then a pairwise comparison of trees will result in 36% tree pairs where at least one of the trees is “structurally faulty”.

Building an instance DAG requires more time than running any of the presented heuristics. We conjecture that there is only limited potential for speeding up the graph building phase. To this end, whereas searching for better (and not significantly slower) heuristics is still a valid undertaking, faster heuristics are of little practical use. It is worth mentioning that computing exact solutions for the NP-hard MAXIMUM COLORFUL SUBTREE problem takes only about 100-fold the time needed for constructing the graph instances; further speed-up is possible using data reductions and a stronger ILP formulation of the problem from [20].

Even elaborate heuristics for a bioinformatics problem, which are capable of finding solutions with objective function value very close to the optimum, can result in solutions which are structurally very dissimilar from the optimum structure. We showed that this is not only a theoretical possibility, but happens regularly for real-world instances. This underlines the importance of finding exact solutions for bioinformatics problems; the structure of solutions found by heuristic, including local search heuristics such as Markov chain Monte Carlo, may deviate significantly from the optimal solution.

### Availability

The Critical Path<sup>3</sup> heuristic and the exact method are available through the SIRIUS 4 software (<https://bio.informatik.uni-jena.de/software/sirius/>) and also from GitHub (<https://github.com/boecker-lab/sirius>). Source code for all other heuristics will be made available upon request. Instances are available from <https://bio.informatik.uni-jena.de/data/>.



■ **Figure 6** Left: Size of the fragmentation tree. Instances are sorted with respect to the size of the optimal fragmentation tree (black); green bars indicate the corresponding tree sizes for the Critical Path<sup>3</sup> heuristic. Middle: Distribution of tree sizes for the exact method (black) and the Critical Path<sup>3</sup> heuristic (green). Right: Comparison of the fragmentation tree structure, optimal tree vs. the tree computed by the Critical Path<sup>3</sup> heuristic. Note the zoom of the x axis. In all three cases, we consider only the true molecular formula candidate for each compound.

---

## References

- 1 Alexander A Aksenov, Ricardo da Silva, Rob Knight, Norberto P Lopes, and Pieter C Dorrestein. Global chemical analysis of biology by mass spectrometry. *Nature Reviews Chemistry*, 1(7):0054, 2017.
- 2 Sebastian Böcker and Kai Dührkop. Fragmentation trees reloaded. *J Cheminform*, 8:5, 2016. doi:10.1186/s13321-016-0116-8.
- 3 Sebastian Böcker and Florian Rasche. Towards de novo identification of metabolites by analyzing tandem mass spectra. *Bioinformatics*, 24:I49–I55, 2008. Proc. of *European Conference on Computational Biology (ECCB 2008)*. doi:10.1093/bioinformatics/btn270.
- 4 Ricardo R. da Silva, Pieter C. Dorrestein, and Robert A. Quinn. Illuminating the dark matter in metabolomics. *Proc Natl Acad Sci U S A*, 112(41):12549–12550, 2015. doi:10.1073/pnas.1516878112.
- 5 Riccardo Dondi, Guillaume Fertin, and Stéphane Vialette. Complexity issues in vertex-colored graph pattern matching. *J Discrete Algorithms*, 9(1):82–99, 2011. doi:10.1016/j.jda.2010.09.002.
- 6 Kai Dührkop, Huibin Shen, Marvin Meusel, Juho Rousu, and Sebastian Böcker. Searching molecular structure databases with tandem mass spectra using CSI:FingerID. *Proc Natl Acad Sci U S A*, 112(41):12580–12585, 2015. doi:10.1073/pnas.1509788112.
- 7 Guillaume Fertin, Julien Fradin, and Géraldine Jean. Algorithmic aspects of the maximum colorful arborescence problem. In *Proc. of Theory and Applications of Models of Computation (TAMC 2017)*, volume 10185 of *Lect Notes Comput Sci*, pages 216–230, 2017. doi:10.1007/978-3-319-55911-7\_16.
- 8 Alaa Khedr, Soad S Abd El-Hay, and Ahmed K Kammoun. Liquid chromatography-tandem mass spectrometric determination of propofol in rat serum and hair at attogram level after derivatization with 3-bromomethyl-propyphenazone. *Journal of pharmaceutical and biomedical analysis*, 134:195–202, 2017. doi:10.1016/j.jpba.2016.11.051.
- 9 Vincent Lacroix, Cristina G. Fernandes, and Marie-France Sagot. Motif search in graphs: Application to metabolic networks. *IEEE/ACM Trans Comput Biology Bioinform*, 3(4):360–368, 2006.

- 10 Gary J. Patti, Oscar Yanes, and Gary Siuzdak. Metabolomics: The apogee of the omics trilogy. *Nat Rev Mol Cell Biol*, 13(4):263–269, 2012. doi:10.1038/nrm3314.
- 11 Florian Rasche, Aleš Svatoš, Ravi Kumar Maddula, Christoph Böttcher, and Sebastian Böcker. Computing fragmentation trees from tandem mass spectrometry data. *Anal Chem*, 83(4):1243–1251, 2011. doi:10.1021/ac101825k.
- 12 Imran Rauf, Florian Rasche, François Nicolas, and Sebastian Böcker. Finding maximum colorful subtrees in practice. *J Comput Biol*, 20(4):1–11, 2013. doi:10.1089/cmb.2012.0083.
- 13 Emma L Schymanski, Heinz P Singer, Jaroslav Slobodnik, Ildiko M Ipolyi, Peter Oswald, Martin Krauss, Tobias Schulze, Peter Haglund, Thomas Letzel, Sylvia Grosse, Nikolaos S Thomaidis, Anna Bletsou, Christian Zwiener, María Ibáñez, Tania Portolés, Ronald de Boer, Malcolm J Reid, Matthias Onghena, Uwe Kunkel, Wolfgang Schulz, Amélie Guillon, Naïke Noyon, Gaëla Leroy, Philippe Bados, Sara Bogialli, Draženka Stipančev, Pawel Rostkowski, and Juliane Hollender. Non-target screening with high-resolution mass spectrometry: critical review using a collaborative trial on water analysis. *Analytical and bioanalytical chemistry*, 407:6237–6255, 2015. doi:10.1007/s00216-015-8681-7.
- 14 Emma Louise Schymanski, Christoph Ruttkies, Martin Krauss, Céline Brouard, Tobias Kind, Kai Dührkop, Felicity Ruth Allen, Arpana Vaniya, Dries Verdegem, Sebastian Böcker, Juho Rousu, Huibin Shen, Hiroshi Tsugawa, Tanvir Sajed, Oliver Fiehn, Bart Ghesquière, and Steffen Neumann. Critical Assessment of Small Molecule Identification 2016: Automated methods. *J Cheminf*, 9:22, 2017. doi:10.1186/s13321-017-0207-1.
- 15 Huibin Shen, Kai Dührkop, Sebastian Böcker, and Juho Rousu. Metabolite identification through multiple kernel learning on fragmentation trees. *Bioinformatics*, 30(12):i157–i164, 2014. Proc. of *Intelligent Systems for Molecular Biology (ISMB 2014)*. doi:10.1093/bioinformatics/btu275.
- 16 Aleksandra Skirycz, Sylwia Kierszniowska, Michaël Méret, Lothar Willmitzer, and George Tzotzos. Medicinal bioprospecting of the amazon rainforest: A modern eldorado? *Trends in biotechnology*, 34:781–790, 2016. doi:10.1016/j.tibtech.2016.03.006.
- 17 Stephen E. Stein. Mass spectral reference libraries: An ever-expanding resource for chemical identification. *Anal Chem*, 84(17):7274–7282, 2012. doi:10.1021/ac301205z.
- 18 Robert Endre Tarjan. A class of algorithms which require nonlinear time to maintain disjoint sets. *J Comput System Sci*, 18(2):110–127, 1979.
- 19 Mingxun Wang, Jeremy J. Carver, Vanessa V. Phelan, Laura M. Sanchez, Neha Garg, Yao Peng, Don Duy Nguyen, Jeramie Watrous, Clifford A. Kapon, Tal Luzzatto-Knaan, Carla Porto, Amina Bouslimani, Alexey V. Melnik, Michael J. Meehan, Wei-Ting Liu, Max Crüsemann, Paul D. Boudreau, Eduardo Esquenazi, Mario Sandoval-Calderón, Roland D. Kersten, Laura A. Pace, Robert A. Quinn, Katherine R. Duncan, Cheng-Chih Hsu, Dimitrios J. Floros, Ronnie G. Gavilan, Karin Kleigrewe, Trent Northen, Rachel J. Dutton, Delphine Parrot, Erin E. Carlson, Bertrand Aigle, Charlotte F. Michelsen, Lars Jelsbak, Christian Sohlenkamp, Pavel Pevzner, Anna Edlund, Jeffrey McLean, Jörn Piel, Brian T. Murphy, Lena Gerwick, Chih-Chuang Liaw, Yu-Liang Yang, Hans-Ulrich Humpf, Maria Maansson, Robert A. Keyzers, Amy C. Sims, Andrew R. Johnson, Ashley M. Sidebottom, Brian E. Sedio, Andreas Klitgaard, Charles B. Larson, Christopher A. Boya P, Daniel Torres-Mendoza, David J. Gonzalez, Denise B. Silva, Lucas M. Marques, Daniel P. Demarque, Egle Pociute, Ellis C. O’Neill, Enora Briand, Eric J N. Helfrich, Eve A. Granatosky, Evgenia Glukhov, Florian Ryffel, Hailey Houson, Hosein Mohimani, Jenan J. Kharbush, Yi Zeng, Julia A. Vorholt, Kenji L. Kurita, Pep Charusanti, Kerry L. McPhail, Kristian Fog Nielsen, Lisa Vuong, Maryam Elfeki, Matthew F. Traxler, Niclas Engene, Nobuhiro Koyama, Oliver B. Vining, Ralph Baric, Ricardo R. Silva, Samantha J. Mascuch, Sophie Tomasi, Stefan Jenkins, Venkat Macherla, Thomas Hoffman, Vinayak Agarwal, Philip G. Williams,

- Jingqui Dai, Ram Neupane, Joshua Gurr, Andrés M C. Rodríguez, Anne Lamsa, Chen Zhang, Kathleen Dorrestein, Brendan M. Duggan, Jehad Almaliti, Pierre-Marie Allard, Prasad Phapale, Louis-Felix Nothias, Theodore Alexandrov, Marc Litaudon, Jean-Luc Wolfender, Jennifer E. Kyle, Thomas O. Metz, Tyler Peryea, Dac-Trung Nguyen, Danielle VanLeer, Paul Shinn, Ajit Jadhav, Rolf Müller, Katrina M. Waters, Wenyuan Shi, Xueting Liu, Lixin Zhang, Rob Knight, Paul R. Jensen, Bernhard Ø. Palsson, Kit Pogliano, Roger G. Linington, Marcelino Gutiérrez, Norberto P. Lopes, William H. Gerwick, Bradley S. Moore, Pieter C. Dorrestein, and Nuno Bandeira. Sharing and community curation of mass spectrometry data with Global Natural Products Social molecular networking. *Nat Biotechnol*, 34(8):828–837, 2016. doi:10.1038/nbt.3597.
- 20 W. Timothy J. White, Stephan Beyer, Kai Dührkop, Markus Chimani, and Sebastian Böcker. Speedy colorful subtrees. In *Proc. of Computing and Combinatorics Conference (COCOON 2015)*, volume 9198 of *Lect Notes Comput Sci*, pages 310–322. Springer, Berlin, 2015. doi:10.1007/978-3-319-21398-9\_25.
- 21 David S Wishart. Emerging applications of metabolomics in drug discovery and precision medicine. *Nature reviews. Drug discovery*, 15:473–484, 2016. doi:10.1038/nrd.2016.32.