4

A multiple testing method for hypotheses structured in a Directed Acyclic Graph

Abstract

We present a novel multiple testing method for testing null-hypotheses that are structured in a directed acyclic graph (DAG). The method is a top-down method that strongly controls the familywise error rate and can be seen as a generalization of Meinshausen's procedure for tree-structured hypotheses. Just as Meinshausen's procedure, our proposed method can be used to test for variable importance, only the corresponding variable clusters can be chosen more freely, because the method allows for multiple parent nodes and partially overlapping hypotheses. An important application of our method is in gene set analysis, in which one often wants to test multiple gene sets as well as individual genes for their association with a clinical outcome. By considering the genes and the gene sets as nodes in a DAG, our method enables us to test both for significant gene sets as well as for significant individual genes within the same multiple testing procedure. The method will be illustrated by testing Gene Ontology terms for evidence of differential expression in a survival setting and is implemented in the `R` package `cherry`.

This chapter has been published as: Rosa J. Meijer and Jelle J. Goeman (2015). A multiple testing method for hypotheses structured in a directed acyclic graph. *Biometrical Journal 57* (1), 123–143.

4.1 Introduction

In many situations in which the aim is to discover which variables are associated with the value of a certain outcome, it can be valuable to not only focus on proving associations on the level of individual variables, but to investigate possible associations on the level of smaller or larger groups of variables as well. Especially when individual effects are small and variables are jointly associated with the response, group association might be detected by powerful group-level tests (e.g. Goeman et al., 2004; Mansmann and Meister, 2005), whereas association on the individual level will probably not be demonstrable. By using suitable sequentially rejective multiple testing procedures, the hypotheses relating to larger groups of variables can not only be used to prove general association but can also lead the way in a directed search procedure where smaller groups of variables or individual variables are only tested as a result of earlier rejections of larger sets in which they are contained. In this way, the procedure will "zoom in" on the real regions of interest (Ehm et al., 2010).

A well-known example of such a "zooming-in procedure" is the hierarchical testing procedure presented by Meinshausen (2008), which is meant for testing hypotheses that can be structured within a specific tree-structure in which hypotheses relating to larger sets of variables are ancestors of hypotheses relating to smaller subsets. The procedure is a top-down approach, meaning that each node is only tested when its parent node was found significant, and strongly controls the familywise error rate (FWER). Unfortunately, the method is limited to situations in which hypotheses are either disjoint or fully overlapping; assumptions that are often not met in practical situations. In this paper, we generalize Meinshausen's method to a far more flexible directed acyclic graph (DAG) framework. Meinshausen (2008) already suggested that his approach could be made feasible for more general hierarchical structures, but we will show that some important changes have to be made in order to achieve this.

Our proposed multiple testing procedure for DAG-structured hypotheses can again be used to test for variable importance or for model selection purposes, just as Meinshausen's tree-based method, only the corresponding variable clusters can be chosen more freely. Another specific application of our method is to test multiple gene sets, obtained from the Gene Ontology (GO) (Ashburner et al., 2000), for association with a certain outcome variable. GO terms are often nested and testing them by means of a hierarchical testing procedure is very natural, because if a gene set is not associated with the response, the same must hold for all smaller gene sets within this bigger set. Furthermore, by expanding the GO-DAG with individual genes, our method enables us to test both for significant gene sets as well as for significant individual genes within the same multiple testing procedure. The corresponding graph structure will even be more informative, because when a gene set is associated with the outcome, this must be because at least one of the corresponding genes is associated with the outcome, and all these genes are present in the DAG. Using these logical relations, known as restricted combinations (Shaffer, 1986), improves the power of our multiple testing procedure and even allows us to formulate confidence

statements of the form given by Goeman and Solari (2011) about the minimum number of genes within a gene set that have to be associated with the outcome.

Many methods have been developed specifically for testing GO terms (see e.g. Khatri and Draghici, 2005, for an overview), but most of these methods do not use the underlying DAG structure. An exception is the innovative hidden Markov model approach by Liang and Nettleton (2010) in which the structure of the GO-graph plays an important role. In this approach however, the FWER is not completely controlled and the GO-DAG is first transformed to a tree-structure. Also the Focus Level procedure of Goeman and Mansmann (2008) uses the DAG-structure by combining a top-down and bottom-up approach. Using it as a full top-down approach will however only be feasible for small DAGs (in that case the method becomes the closed testing method of Marcus et al. (1976)). Other methods that can be used on DAGs are the graphical approaches by Bretz et al. (2009) and Burman et al. (2009). These methods could also be used as a top-down method, but the methods do not use restricted combinations to improve their power.

In the next section, we will describe our method and its properties in detail. To benefit from possible logical relationships, we have to repeatedly solve instances of an NP-hard problem. In the Algorithms section, we will illustrate how this problem can either be solved exactly or how it can be approximated. In the Applications section, we will illustrate our method by testing Gene Ontology terms for evidence of differential expression in a survival setting and we will compare the results to results obtained from already existing methods. We will furthermore explore our method's behavior in more detail by using simulations. Finally, some suggestions for future work can be found in the discussion. Software is available in the R package `cherry`.

4.2 The method

Suppose we want to test n null-hypotheses $H_1, \ldots, H_n$ that are structured in a DAG. In this DAG, each node represents an hypothesis, whereas the edges represent the underlying structure of those hypotheses. In the remainder of this article, we will assume that the edges are chosen in such a way that the falsehood of H_i implies the falsehood of all its ancestors. We will refer to DAGs for which this assumption holds as DAGs in which "*one-way* logical relationships" are present. In a later stage, we will also focus on a special subcategory of the just described DAGs, namely the DAGs in which "*two-way* logical relationships" are present, meaning that the falsehood of H_i not only implies the falsehood of all its ancestors, but also the falsehood of at least one of its child nodes. Two-way logical relationships exist when the null-hypothesis of every node is the intersection of the hypotheses of its child nodes. Note that in both DAG types the truth of H_i implies the truth of all its descendants.

Hypotheses forming a DAG in which one-way logical relationships are present would for example be hypotheses of the following form:

$$H_I \colon \beta_i = 0, \quad \text{for all } i \in I \subseteq \{1, \ldots, m\}. \tag{4.1}$$

When we have two such hypotheses, H_I and H_J, the truth of H_I will imply the truth of H_J if $J \subseteq I$. In this same situation, the falsehood of H_J will imply the falsehood of H_I. Given this type of hypotheses, using proper subset relations in order to form the graph structure, meaning that H_I is an ancestor of H_J if and only if $J \subset I$, will thus result in a graph with the desired properties. To give an example of such a DAG, in the first picture of Figure 4.1, we draw the DAG structure corresponding to 4 hypotheses H_I with $I = \{1, 2, 3, 4\}$, $I = \{2, 3\}$, $I = \{2, 4\}$ and $I = \{2\}$.

(1) DAG with one-way relations (2) DAG with two-way relations

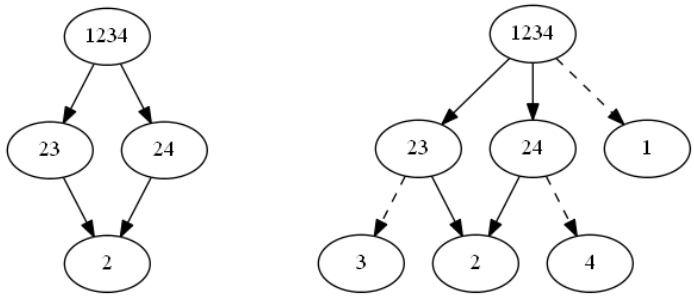


Figure 4.1: Forming DAG structures: a toy example.

Hypotheses of the form given in equation (4.1) are commonly encountered. For example when we want to test several gene sets for association with a certain response variable such as illness. For each gene set, a regression model containing only the genes in this specific set could be constructed and then equation (4.1) would correspond to the null-hypothesis that no gene in this gene set is associated with the response. As soon as many gene sets are proper subsets of other gene sets, which will for example be the case when we look at gene sets from the Gene Ontology graph, their corresponding hypotheses will form the desired DAG structure. Going back to our toy example, the first picture in Figure 4.1 could correspond to the situation in which we want to test one single gene for association with a certain response variable (the node indicated by "2") and three gene sets containing this gene as well as a few others.

A natural way to extend a DAG in which multiple gene-sets are tested to a DAG with two-way logical relationships, is by adding all elementary hypotheses relating individual genes to the outcome variable. In this way, we can test both for significant gene sets as well as for significant individual genes within the same multiple testing procedure. Furthermore, we can use the resulting logical structure to make this testing procedure more powerful than a procedure in which only one-way logical relationships are present. In the second picture in Figure 4.1 we extended the DAG with one-way logical relationships to a DAG with two-way logical relationships, by simply adding the single genes that were missing.

Although DAGs are often linked to gene set testing, they could be used in any variable selection framework in which the importance of sets of variables is tested. For now, we will assume the hypotheses and the graph structure to be given. Given the graph structure, we want to have a multiple testing procedure that tests all hypotheses while using this structure and while strongly controlling the familywise error rate.

In the remainder of the article, we will denote the graph relationships between the hypotheses with the following notation: for a node H, the sets consisting of respectively its parents, ancestors, children and offspring will be denoted by pa(H), an(H), ch(H) and of(H). Furthermore, let $\mathcal{H} = \{H_1, \ldots, H_n\}$ denote the full set of hypotheses and $\mathcal{L}$ the collection of leaf nodes (i.e. nodes without offspring). For each node $H \in \mathcal{H}$, its corresponding leaf nodes are denoted by

$$\mathcal{L}_H = \big(\{H\} \cup \text{of}(H)\big) \cap \mathcal{L}.$$

We will make no assumptions on the test that is performed on each hypothesis or on the correlation structure of the test statistics or p-values. We will simply assume that valid p-values are available for each hypothesis.

4.2.1 Sequential rejection principle

Our proposed method closely resembles the FWER controlling method of Meinshausen (2008) which is designed to work on tree structures. The biggest difference between trees and DAGs in general is the number of parents a node can have. In trees, nodes can only have one parent, whereas nodes can have multiple parents in more general DAG structures. This property of trees is a necessary condition in order for Meinshausen's procedure to work. Furthermore, Meinshausen's procedure requires that for any two hypotheses H_I and H_J of the from given in equation (4.1), either I and J are disjoint sets or one is a subset of the other. Both the single parent assumption and the assumption of disjoint sets are too restrictive when we want to consider general DAG structures.

Our proposed method is a top-down method, just as Meinshausen's procedure. This means that we start testing at the root of the graph, where for the moment we will assume a single root node, so at a node without any parents. If the null-hypothesis corresponding to this root node can be rejected, we test the children of the root node, and we continue to test only those nodes that have *at least one* of their parents or *all* their parent nodes rejected. Applying either variant to a tree structure results in the testing order of Meinshausen's procedure, namely testing only those nodes from which the parent node is already rejected. We will refer to our two variants as respectively the *any-parent* approach and the *all-parents* approach. Because the DAG is structured in such a way that the truth of a parent node implies the truth of its children, a top-down approach is a logical choice. Although failing to reject a certain node will in practice not always mean that the corresponding hypothesis is true, not having the power to reject the parent node will often indicate lack of power to reject the child nodes as well.

Apart from choosing the testing order, the significance levels (or α-levels) must be selected carefully for each test in order to control the FWER. In Meinshausen's procedure,

the significance level for an hypothesis H_I is chosen to be proportional to its cardinality $|I|$. This leads to a procedure with an increasing penalty for multiplicity while traversing the tree from the root to its leaf nodes. We would like to have a similar multiplicity correction for our DAG procedure; giving higher α-levels to the nodes higher in the graph because failing to reject these high level nodes will prevent us from testing their lower level descendants. Unfortunately, as soon as for hypotheses H_I and H_J the sets I and J can arbitrarily overlap, as is the case for DAGs, it is not longer possible to make the significance levels proportional to the cardinality of the nodes, while still controlling the FWER.

We assign significance levels to nodes within a DAG based on the sequential rejection principle (SRP), as formulated by Goeman and Solari (2010). Let us assume that we have raw (i.e. non-multiplicity corrected) p-values p_H for every hypothesis $H \in \mathcal{H}$ within our DAG. The exact statistical test that is used to obtain these raw p-values is not important for our multiple testing method and can, if desired, even vary between the hypotheses. We now want to formulate an iterative multiple testing procedure that starts from an empty rejection set $\mathcal{R}_0 = \emptyset$. In every subsequent step, critical values for all hypotheses will be calculated, and those hypotheses that have p-values smaller than their assigned α-level, will be added to the current rejection set. Subsequently, new critical values will be computed, based on the new rejection set, and this procedure will continue until no further rejections can be made. Formally,

$$\mathcal{R}_{i+1} = \mathcal{R}_i \cup \{H \in \mathcal{H} \setminus \mathcal{R}_i : p_H \leq \alpha_H(\mathcal{R}_i)\}, \tag{4.2}$$

where $\mathcal{R}_i$ is the collection of rejected hypotheses after step i and $\alpha_H(\mathcal{R})$ is a critical value function that, based on a current rejection set $\mathcal{R} \subset \mathcal{H}$, assigns certain significance levels $\alpha_H(\mathcal{R})$ to not yet rejected hypotheses $H \in \mathcal{H} \setminus \mathcal{R}$. Goeman and Solari (2010) showed that in order to strongly control the familywise error rate at level α, we can use any critical value function $\alpha_H(\mathcal{R})$ that satisfies two conditions.

Before stating these two conditions, we first have to distinguish between two types of rejection sets; namely *congruent* rejection sets and *incongruent* rejection sets. When (one-way or two-way) logical relationships are present in the graph, it can happen that not all remaining hypotheses can simultaneously be true given a rejection set $\mathcal{R} \subset \mathcal{H}$, due to restricted combinations (Shaffer, 1986). Given that we have a graph with one-way logical relationships and a rejection set $\mathcal{R}$ for which $H_i \in \mathcal{R}$, but $H_j \notin \mathcal{R}$ for some $H_j \in \text{pa}(H_i)$, we know that $\mathcal{R}$ cannot be the complete set of false hypotheses, because the falsehood of H_i implies the falsehood of all its parents. Similarly, if we have two-way logical relationships and H_i is an element of $\mathcal{R}$ but none of its children are included, the rejection set cannot be the complete set of false hypotheses either. We call a rejection set $\mathcal{R}$ congruent if it can be the complete set of false hypotheses, and incongruent otherwise.

Given this definition, we can state the first condition of the SRP, which is the *monotonicity condition*. This condition tells us that for every $\mathcal{R} \subseteq \mathcal{S} \subset \mathcal{H}$ and for every $H \in \mathcal{H} \setminus \mathcal{S}$, we must have

$$\alpha_H(\mathcal{R}) \leq \alpha_H(\mathcal{S}). \tag{4.3}$$

The α-levels of not yet rejected hypotheses H can thus never diminish in subsequent steps of the procedure. Although the monotonicity condition is formulated for general sets $\mathcal{R}$ and $\mathcal{S}$, from the proof of the theorem as given in (Goeman and Solari, 2010), it follows that this condition can be slightly sharpened. The monotonicity condition only has to be met for every $\mathcal{R}$ that can be an actual rejection set in the chosen multiple testing procedure and every *congruent* set $\mathcal{S}$.

The second condition, known as the *single-step condition*, is met when for all congruent sets $\mathcal{R} \subset \mathcal{H}$ the following holds:

$$\sum_{H \in \mathcal{H} \setminus \mathcal{R}} \alpha_H(\mathcal{R}) \leq \alpha. \tag{4.4}$$

In every step, the α_H's distributed over possibly simultaneously true hypotheses H can thus never exceed the total α-level. A sequential procedure that satisfies these two conditions will strongly control the familywise error rate, without any further assumption on the dependence structure of the individual p-values.

4.2.2 Testing procedure for DAGs with one-way logical relationships

Let us first consider graphs in which only one-way logical relationships are present. As mentioned earlier, we will have two approaches, the any-parent approach and the all-parents approach. Given a current rejection set $\mathcal{R}_i$, for the any-parent approach we have to make sure that $\alpha_H(\mathcal{R}_i) \neq 0$ for all nodes H for which at least one of their parent nodes is included in $\mathcal{R}_i$, whereas $\alpha_H(\mathcal{R}_i)$ equals 0 for all other nodes. Similarly, for the all-parents approach, we only want $\alpha_H(\mathcal{R}_i)$ to be non-zero for nodes that have all their parents rejected. We will refer to the nodes that receive a positive α-value as candidate nodes, or *candidates* for short. Besides we want to choose our function $\alpha_H(\mathcal{R})$ roughly in such a way that candidates with more offspring receive higher α-values, because their rejection is essential in order for their offspring to be tested. We accomplish this in the following way: each leaf node L is assigned a certain weight v_L, that we for now assume to be equal to 1. Given a rejection set $\mathcal{R}_i$, the weights from all unrejected leaf nodes $L \notin \mathcal{R}_i$ will be moved upwards through the graph in an iterative manner. Each node can receive weight through its unrejected children and can pass this weight further on to its own unrejected parents. When there are no unrejected parents left, the weight flow terminates, leaving us with a final weight-distribution in which the only nodes with a non-zero weight are those that are not yet rejected and that have, depending on the chosen variant, some or all of their parents rejected. Furthermore, loosely speaking the more routes there are from the leaves of the graph towards a certain node, the higher its corresponding weight and therefore α-level will be. The outline of the corresponding algorithm is given in Algorithm 4.2.1. In this outline, we assume that we have a topological ordering of our DAG, which means that we have a linear ordering of the nodes such that for every node H, all its child nodes come before H in the ordering. For every DAG, such an ordering exists. This topological ordering will usually not be unique, but the outcome of the algo-

rithm will be the same for all possible orderings.

Algorithm 4.2.1. ***update weights***

Requires:
initial weights v_L for each leaf node L
current rejection set $\mathcal{R}_i$

initialize weights: $w_L := v_L$ for all $L \in (\mathcal{L} \cap \mathcal{R}_i^c)$, $w_H := 0$ otherwise

for each *$H \in \mathcal{R}_i^c$ in topological order*
  if *$pa(H) \setminus \mathcal{R}_i \neq \emptyset$*
    for each *P in $pa(H) \setminus \mathcal{R}_i$*
      $w_P := w_P + f(w_H)$
    $w_H := w_H - g(w_H)$
return w

In this algorithm, the functions $f(w_H)$ and $g(w_H)$ depend on the chosen variant and are given as follows:

$$f(w_H) = \begin{cases} \frac{w_H}{|\text{pa}(H)|} & \text{for the any-parent procedure} \\ \frac{w_H}{|\text{pa}(H) \setminus \mathcal{R}_i|} & \text{for the all-parents procedure} \end{cases}, \tag{4.5}$$

and

$$g(w_H) = \begin{cases} \frac{|\text{pa}(H) \setminus \mathcal{R}_i|}{|\text{pa}(H)|} w_H & \text{for the any-parent procedure} \\ w_H & \text{for the all-parents procedure} \end{cases}, \tag{4.6}$$

where we use the notation $|A|$ to denote the number of elements in a set A. We see that in the all-parents variant, a node will distribute all its weight uniformly over its unrejected parents, in case there are any, whereas in the any-parent variant a node distributes its weight uniformly over all its parents, keeping the parts of already rejected parents to itself.

Once the weight-flow has terminated, for each node H the α-level is computed as follows:

$$\alpha_H(\mathcal{R}_i) = \frac{w_H}{\sum_{L \notin \mathcal{R}_i} v_L} \alpha, \tag{4.7}$$

where w_H is the weight that is present in node H at the end of the weight-flow procedure. Note that w_H should actually be denoted as $w_H(\mathcal{R}_i)$ because its value depends on the current rejection set $\mathcal{R}_i$, but we will usually omit this for ease of notation.

For both variants, the obtained α-levels can now be substituted in equation (4.2) which completes the multiple testing procedure. In case adjusted p-values are required, the procedure does not start at the fixed α-level on which we want to control the FWER, but

in each iteration the minimum overall α-level needed to reject at least one candidate is determined, and the maximum over this and the previous level is taken to ensure that the α-level can only increase. This gives us the minimum α-value needed to reject this candidate (or candidates) and all its ancestors, i.e. its adjusted p-value. As long as this α-level stays below 1, new rejections will be made. If the level exceeds 1, the procedure ends and the adjusted p-values of the not yet rejected hypotheses are set to 1.

In case of the any-parent rule, it can happen that a node H gets rejected before all its parents are rejected. In that case, all its not yet rejected parents are automatically rejected as well (with the same adjusted p-values). If we view those "free rejections" as tests on an α-level of 1, we see why the two conditions of the SRP are not violated by these rejections. The monotonicity condition is not violated because the previous α-levels of the parent nodes can never have exceeded 1. The single step condition is met because, by the one-way logical relationships, for each unrejected parent $P \in \text{pa}(H)$ of $H \in \mathcal{R}_i$ there will exist no congruent rejection set $\mathcal{S} \supseteq \mathcal{R}_i$, for which $P \notin \mathcal{S}$. Because the single step condition only imposes restrictions on α-levels of nodes not contained in $\mathcal{S}$, it does not impose any restrictions on the α-levels of the unrejected parents of H. Following a more loose argumentation, we would also come to the conclusion that free rejections will not violate the FWER control, because these rejections can only be wrong if the previously made rejection of H was a false rejection, and if we can prove that this rejection did not violate the FWER control, the same must hold for the rejections implied by this first rejection.

To prove that both variants strongly control the FWER on level α, it suffices to check whether the two conditions imposed by the sequential rejection principle are met. The single step condition is almost trivially met for both procedures. In every step, all α-values add up exactly to the overall α, because the sum of all the weights in the graph equals the sum of the initial weights from the unrejected leaf nodes, from which it follows that

$$\sum_{H \in \mathcal{H}} \alpha_H(\mathcal{R}) = \frac{\sum\limits_{H \in \mathcal{H}} w_H}{\sum\limits_{L \notin \mathcal{R}} v_L} \alpha = \alpha.$$

To prove that the monotonicity condition holds, we have to show that for an arbitrary rejection set $\mathcal{R}$ that can be encountered in our procedure and an arbitrary congruent set $\mathcal{S}$, with $\mathcal{S} \supset \mathcal{R}$, for every node $H \notin \mathcal{S}$ we must have that $\alpha_H(\mathcal{R}) \leq \alpha_H(\mathcal{S})$. We will show that this holds by going over the unrejected nodes in topological order (meaning that we will never encounter a node before we have encountered all its children) and first proving that the weight they receive via their child nodes is monotone (i.e. this incoming weight can only increase with more rejections). Let H be the current unrejected node in this ordering. If H is a leaf node, in both variants its incoming weight cannot diminish with extra rejections, because the weight the node receives equals a fixed initial weight v_H. If H is no leaf node, we already know that the incoming weights of its children are monotone. Furthermore, all its children have to be unrejected, otherwise H itself had to part of the set $\mathcal{S}$ by the presence of one-way logical relationships and the fact

that $\mathcal{S}$ is a congruent set. As a result, the weight that H will receive via its children has to be monotone as well, because in both variants of our multiple testing procedure, a child sends a certain fraction of the weight it receives to its unrejected parents, and this fraction can only increase when the number of rejected parents increases. All incoming weights are thus monotone, and furthermore, the fraction of this weight that stays in the node itself can only increase with more rejected parents, from which it follows that $w_H(\mathcal{R}) \leq w_H(\mathcal{S})$ for each node $H \notin \mathcal{S}$. In conclusion, w_H can only increase with more rejections, which means that the numerator in equation (4.7) can only increase. Furthermore, the denominator can only decrease, because a larger rejection set can never lead to fewer unrejected leaf nodes. From this, we can conclude that the α-values can only increase with more rejections, which means that both conditions of the SRP are satisfied and the FWER controlling property of our method is proven.

4.2.3 Using two-way logical relationships: a more powerful procedure

Until now, we did not assume two-way logical relationships in our DAG. However, if these exist, more powerful variants of our procedure can be developed by exploiting the fact that the SRP's single-step condition only prescribes the individual α-levels to sum up to α in case of congruent rejection sets. In case of incongruent rejection sets, the individual α-levels may therefore sum up to a value larger than the overall α. Nonetheless, the sum of the α-values in case of an incongruent rejection set is still restricted because the monotonicity condition has to be satisfied as well. Given an incongruent rejection set $\mathcal{R}$ and an arbitrary congruent rejection set $\mathcal{S}$ with $\mathcal{R} \subset \mathcal{S}$, for each node $H \notin \mathcal{S}$ we must have that $\alpha_H(\mathcal{R}) \leq \alpha_H(\mathcal{S})$. The α-level for a certain node H in an incongruent situation (i.e. a situation in which we have an incongruent rejection set) can thus never exceed the α-level this node will get in a future congruent situation. If we denote the set of all congruent rejections sets by Φ, we get

$$\alpha_H(\mathcal{R}) = \min_{\mathcal{S} \in \Phi:\ \mathcal{S} \supseteq \mathcal{R}, H \notin \mathcal{S}} \alpha_H(\mathcal{S}),$$

as given in (Goeman and Solari, 2010). For our procedure, this would result in

$$\alpha_H(\mathcal{R}) = \min_{\mathcal{S} \in \Phi:\ \mathcal{S} \supseteq \mathcal{R}, H \notin \mathcal{S}} \frac{w_H(\mathcal{S})}{\sum_{L \notin \mathcal{S}} v_L} \alpha, \tag{4.8}$$

where $w_H(\mathcal{S})$ can be obtained either by the any-parents or all-parents rule. Note that this expression will reduce to equation (4.7) in case $\mathcal{R}$ itself is a congruent rejection set.

Given that we have a DAG with two-way logical relationships, using equation (4.8) in combination with equation (4.2) leads to a more powerful multiple testing procedure than the one described previously for DAGs with one-way logical relationships. In case of one-way relationships, we could only use that a congruent rejection set $\mathcal{R}$ is restricted

to satisfy the condition that $H \in \mathcal{R}$ implies $\text{pa}(H) \subset \mathcal{R}$, leading to the previously mentioned "free rejections" of unrejected nodes having rejected children. In the presence of two-way relationships, these free rejection can still be made, but $\mathcal{R}$ additionally has to satisfy the condition that $H \in \mathcal{R}$ implies $\text{ch}(H) \cap \mathcal{R} \neq \emptyset$ in order to be congruent. This extra restriction can lead to larger individual α-values.

The proof that the procedure still strongly controls the FWER is completely analogous to the former proof. The single step condition only has to be satisfied in congruent situations, and in congruent situations the procedure is equivalent to the previous procedure. The monotonicity condition is satisfied because a minimum is taken over α-values which can only increase with more rejections, as shown previously.

The question is now whether there is a way to calculate the minimum in equation (4.8) relatively quickly. Calculating all possible α-values and taking their minimum is not an option, because there are combinatorially many extensions from an incongruent to a congruent set, and in each step of the algorithm and for each candidate, these extensions will differ. Unfortunately, minimizing the fraction in equation (4.8) for general DAG-structures turned out to be an NP-hard problem, which means that no algorithms exist to solve this problem within polynomial time. We could show this by making a reduction from Hitting Set, which is known to be an NP-hard problem itself (see e.g. Vinterbo and Ohrn, 2000), to the problem of finding this minimum. We will not explicitly give the reduction in this article, but later in this paragraph it will become apparent that calculating the minimum of the fraction in (4.8) by only considering variations in the denominator is actually equivalent to (a weighted variant of) Hitting Set.

Although finding optimal α-levels in reasonable time will thus theoretically be difficult, in practice we could still try to use (highly optimized) existing software to find these values. In small DAGs or specific DAG-types (DAGs that resemble trees for example), finding the optimal solution might still be feasible. In the any-parent approach, we were able to reformulate the problem as an integer linear program (ILP) (see e.g. Dasgupta et al., 2006, for an exact definition of ILPs) from which an "acceptance/rejection"-statement could be obtained. For a given candidate C with p-value p_C, and for a given overall α-level, we cannot calculate the exact α-value, but we can determine whether $p_C \leq \alpha_C(\mathcal{R})$, that is, whether we can reject C given the current rejection set $\mathcal{R}$ and overall α-value α. The exact formulation is shown in the appendix. An advantage of an ILP formulation is that there are many ILP solvers available, and in addition, for every ILP there exists a linear programming relaxation (as we will explain in the Algorithms section) which can be solved in polynomial time and from which the solution can be used as an approximation of the solution of the original integer linear program. In case of the all-parents approach, we were not able to formulate the problem as an ILP. The value of the numerator (i.e. the weight a node receives from the weight-flow process) depends on the exact configuration of all rejection sets and is for that reason very difficult to model.

Instead of minimizing the numerator and denominator simultaneously, we can also minimize the numerator or maximize the denominator and keep the other part of the fraction fixed at the value it has in the current incongruent rejection set $\mathcal{R}$. Because the

denominator and the numerator will respectively decrease and increase with more rejections, we will in this way find a lower bound for the actual minimum of the full fraction, leading to a procedure that can be too conservative but still controls the FWER. We chose to maximize the denominator, partly because minimizing the numerator in the all-parent approach seemed very difficult for the previous given reasons, but mostly because looking at the denominator instead of the numerator offers a second possibility to simplify the problem. Whereas the numerator cannot be optimized once for all candidates simultaneously, this can be done for the denominator. If for all candidates H,

$$\max_{\mathcal{S}\in\Phi:\ \mathcal{S}\supseteq\mathcal{R}, H\notin\mathcal{S}} \sum_{L\notin\mathcal{S}} v_L$$

is approximated by

$$\max_{\mathcal{S}\in\Phi:\ \mathcal{S}\supseteq\mathcal{R}} \sum_{L\notin\mathcal{S}} v_L$$

we can be sure that the actual maximum is approximated from above, because the second maximum is taken over the same and probably more congruent sets $\mathcal{S}$. Although again potentially adding to the method's conservativeness, this approximation will also increase the method's computational speed.

In the Algorithms section we will show that maximizing the denominator is equivalent to solving an instance of a weighted version of Hitting Set, which shows that even maximizing the denominator separately leads to an NP-hard problem. We can again formulate the problem as an ILP (as we will show in the Algorithm section) which we can approximate in polynomial time by solving its linear programming relaxation or (try to) solve exactly by leveraging the wide availability of ILP solvers.

In conclusion, for larger problems and for the all-parents approach it will generally not be possible to find exact solutions in reasonable time. However, we can use approximations of the optimal α-values to come to a final rejection set. Better approximations of this optimal value will lead to a procedure with more power to detect false hypotheses, but the gain in power comes at the cost of higher computation time. The mentioned approximation methods are given below in increasing order of accuracy and computation time:

- Use the current value of the denominator. This leads to the approach described for DAGs containing only one-way logical relationships.
- Use a linear program to approximate the maximum value of the denominator. This can be done in polynomial time.
- Use an integer linear program to find the exact maximum value of the denominator. This cannot be done in polynomial time.

For the last two approaches, there is also the choice to approximate the maximum value of the denominator for all candidates simultaneously (which is faster, but possibly more

conservative) or for each candidate separately. In the any-parent approach there is also the possibility to use the (relaxation of the) ILP given in the appendix that optimizes the numerator and denominator simultaneously.

If one of the approaches would be chosen beforehand, we can again calculate adjusted p-values by increasing the overall α-level in each iteration up to the minimum value on which new rejections occur. However, for larger graphs, we would recommend to start with the procedure that does not use two-way logical relationships and to switch to more powerful approximation methods when no new rejections can be made. If the method is used in this way, adjusted p-values cannot be reported, because rejections might may have occurred on smaller overall α-levels, if a more powerful procedure would have been used from the beginning. Furthermore, using the (relaxation of the) ILP given in the appendix will not result in adjusted p-values, because the actual α-values are not calculated.

4.2.4 Confidence sets for the number of true findings in sets chosen post-hoc

Usually, after establishing the final rejection set $\mathcal{R}$, we only make statements about the presence or absence of a certain hypothesis within this set. However, using the rejection set $\mathcal{R}$ in combination with the given DAG structure can sometimes lead to more informative statements. As a motivating example, suppose our hypotheses are of the form given in equation (4.1). In that case we can view each hypothesis as an intersection of one or more elementary hypotheses $H_i\colon \beta_i = 0$, relating one variable (e.g. a certain gene) to a specific outcome variable. In addition to the information whether a certain intersection hypothesis H_I is part of $\mathcal{R}$, an interesting question is whether we can say something about the number of elementary hypotheses within this intersection that have to be false. When the intersection hypothesis H_I would state that none of the genes in gene set I are associated with a response variable, rejecting this hypothesis means that at least one of the genes is in fact associated with the response. If only H_I is tested (or rejected), this is all the information we have, but if we have other hypotheses H_J in our rejection set $\mathcal{R}$, with $J \subset I$, we can use this information to possibly say more about the minimum number of genes in gene set I that have to be associated with the response.

The idea to derive statements regarding the number of true findings (i.e. false hypotheses) in a certain set of elementary hypotheses was introduced by Goeman and Solari (2011). They have shown that for any set, or in other words any intersection, of elementary hypotheses chosen *post hoc*, an exact simultaneous confidence interval can be constructed for the number of true findings in this specific set, where these confidence sets are based on a rejection set $\mathcal{R}'$ generated from a closed testing procedure. The reasoning behind the validity of their procedure is essentially based on the following observation: because all rejections within $\mathcal{R}'$ are *simultaneously* valid with a probability of at least $1-\alpha$, the same holds for all statements based on these rejections.

Although the method was initially proven for confidence statements based on a rejection set obtained from a closed testing procedure, the same reasoning applies to statements

based on a rejection set obtained from our DAG method. Because our DAG method is a coherent multiple testing method that strongly controls the FWER at a pre-specified α-level, all rejections within the final rejection set $\mathcal{R}$ will be simultaneously valid with probability at least $1 - \alpha$ and the same will hold for the derived confidence intervals for the number of true discoveries within arbitrary sets of elementary hypotheses.

To construct a confidence set for the number of false elementary hypotheses within an arbitrary set $\mathcal{A}$ of k elementary hypotheses, we should check for each $0 \leq l \leq k$ whether having l false hypotheses within $\mathcal{A}$ could be in accordance with the obtained rejection set $\mathcal{R}$. In other words, we should check whether there exists a congruent extension $\mathcal{S} \supseteq \mathcal{R}$ in which exactly l elementary hypotheses $H_i \in \mathcal{A}$ are included. Those values of l that are in accordance with the set $\mathcal{R}$ constitute the $100(1 - \alpha)\%$ confidence set for the number of false hypotheses within the set $\mathcal{A}$. Note that such confidence intervals will always be of the form $\{m, \ldots, k\}$, where m is the minimum number of false hypotheses and k the size of the set.

To obtain the lower bound m, we look at all possible congruent extensions $\mathcal{S} \supseteq \mathcal{R}$ and count the number of elementary hypotheses that are present both in $\mathcal{S}$ and in $\mathcal{A}$. Because one of these congruent extensions has to be the true set of false hypotheses with confidence level $1 - \alpha$, the true number of false hypotheses in $\mathcal{A}$ can never be smaller than the minimum of these numbers which thus equals the desired lower bound m. In order for this procedure to work, we should have a DAG in which the elementary hypotheses are present as leaf nodes. However, the original graph does not have to contain these leaf nodes. For the purpose of finding the confidence sets, the elementary hypotheses can be added after the multiple testing procedure is performed on the original graph. Their presence immediately ensures that the current DAG has two-way logical relationships, which enables us to construct congruent sets.

In addition to the construction of a confidence set for the number of false elementary hypotheses within an arbitrary set, we can also construct a confidence set for the number of false intersection hypotheses, i.e. nodes in our DAG, within an arbitrary collection of those. In our example of testing gene sets, given an arbitrary collection of gene sets (rejected as well as unrejected), we could again construct a confidence set for the number of false gene *sets* (instead of single genes) within this collection.

In the Algorithms section, we will discuss the exact algorithm to construct the confidence sets.

4.3 Algorithms

In this section we will give further details on the exact implementation of the various procedures to calculate or approximate the maximum value of the denominator in equation (4.8). Furthermore, we will demonstrate how confidence sets for the number of false hypotheses within arbitrarily chosen sets of (elementary) hypotheses can be constructed.

Before we can explain the different procedures, we first have to discuss how to extend an incongruent rejection set $\mathcal{R}$ to a congruent rejection set $\mathcal{S}$ with $\mathcal{S} \supseteq \mathcal{R}$, given that

we have a graph in which two-way logical relationships are present. For this purpose, the nodes that are rejected themselves but have no rejected offspring will prove very useful. We will call these nodes *implications* or *implying nodes*. A rejected leaf node is by definition also an implication. The nodes are called "implications", because they *imply* which congruent sets are an extension of $\mathcal{R}$. Remember that in a DAG with two-way logical relationships, each parent hypothesis is the intersection of its child hypotheses, and because this is true for all hypotheses in the graph, each hypothesis H is eventually the intersection of its corresponding leaf nodes $\mathcal{L}_H$:

$$H = \bigcap_{L \in \mathcal{L}_H} L.$$

From this it follows that a congruent extension $\mathcal{S}$ of $\mathcal{R}$ must contain at least one element from $\mathcal{L}_H$ for every $H \in \mathcal{R}$. Given our definition of implying nodes, this is equivalent to the requirement that $\mathcal{S}$ contains at least one element from $\mathcal{L}_I$ for every implication $I \in \mathcal{R}$, because each $H \in \mathcal{R}$ is either an implication itself or the ancestor of an implication and if $H \in \text{an}(I)$, then $\mathcal{L}_I \subset \mathcal{L}_H$. From now on, every rejection set $\mathcal{R}$ can thus be extended to a congruent set by only looking at the implications.

4.3.1 Maximizing the denominator

To maximize the denominator, a congruent extension $\mathcal{S} \supseteq \mathcal{R}$ has to be chosen in such a way that the sum of the initial weights v_L of the unrejected leaf nodes $L \notin \mathcal{S}$ is as large as possible. This is equivalent to choosing $\mathcal{S}$ in such a way that the sum of the weights v_L of the leaf nodes L that are actually *included* in $\mathcal{S}$ is minimized. Given the previous observation that a congruent extension $\mathcal{S}$ of $\mathcal{R}$ is completely determined by the implying nodes of $\mathcal{R}$, we can now formulate the maximization problem of the denominator (for all candidates simultaneously) as follows: Choose a subset $\mathcal{L}' \subseteq \mathcal{L}$ such that

$$\sum_{L \in \mathcal{L}'} v_L \tag{4.9}$$

is minimized and for each implication $I \in \mathcal{R}$, $\mathcal{L}_I \cap \mathcal{L}' \neq \emptyset$. This problem is known as a weighted variant of Hitting Set, which is an NP-hard problem.

If we introduce a binary variable X_i for each leaf node L_i, our minimization problem can be translated into the following ILP:

$$\begin{aligned} &\textbf{minimize} && \sum_i v_{L_i} X_i \\ &\textbf{with respect to} && \sum_{i:\ L_i \in \mathcal{L}_I} X_i \geq 1, \forall I \ \textbf{and}\ X_i \in \{0,1\}. \end{aligned} \tag{4.10}$$

The first constraint ensures that for each implication $I \in \mathcal{R}$, at least one of its corresponding leaf nodes is added to the congruent extension of $\mathcal{R}$. The second constraint ensures

that a leaf node is either chosen or not, but cannot be added to the congruent set partly. The ILP formulation can easily be extended to the situation in which we want to minimize the denominator for each candidate H separately. In that case we want a congruent rejection set $\mathcal{S} \supseteq \mathcal{R}$ such that in addition $\mathcal{S} \cap \mathcal{L}_H = \emptyset$. To fulfill this condition, we can just add the following constraint to the previous ILP:

$$\sum_{i:\ L_i \in \mathcal{L}_H} X_i = 0.$$

By using an ILP solver we can thus find exact answers for the maximum value of the denominator, either for all candidates simultaneously (a single ILP), or for all candidates separately (as many ILPs as there are candidates), but obtaining these exact answers can be computationally intensive, since our problem stays NP-hard. In order to speed up the procedure, we can use approximations for the maximum value of the denominator, instead of the true maximum. We have to be careful however to approximate this maximum from above, because only then we can be certain that our resulting multiple testing method will remain conservative. An easy way to do so is by relaxing the ILP in (4.10), which means that the constraint $X_i \in \{0, 1\}$ is replaced by $X_i \in [0, 1]$. The resulting problem is a regular linear program (LP), which is known to be solvable in polynomial time. Because there is more freedom to find the minimum value of $\sum_i v_{L_i} X_i$ as soon as the X_i's are no longer binary, we can be sure that the minimum of the LP is at least as small as the the minimum of the corresponding ILP. Because the maximum value of the denominator in equation (4.8) can be rewritten as

$$\max_{\mathcal{S} \in \Phi:\ \mathcal{S} \supseteq \mathcal{R}, H \notin \mathcal{S}} \sum_{L \notin \mathcal{S}} v_L = \sum_{L \in \mathcal{L}} v_L - \min_{\mathcal{S} \in \Phi:\ \mathcal{S} \supseteq \mathcal{R}, H \notin \mathcal{S}} \sum_{L \in \mathcal{S}} v_L$$

we see that a smaller minimum value (i.e. the outcome of the LP or ILP) will lead to a higher maximum value of the denominator. The ILP relaxation will thus lead to a conservative multiple testing procedure.

To get an idea of the possible degree of conservativeness, we should investigate how far the optimal solution of the relaxation is from the optimal solution of the actual ILP. Let us define the *integrality gap* of an ILP-relaxation for a minimization problem as

$$\sup_I \frac{OPT(I)}{OPT_R(I)}, \tag{4.11}$$

where $OPT(I)$ is the optimal solution of the ILP and $OPT_R(I)$ is the optimal solution of the relaxation, given an instance I. A higher integrality gap thus means that the approximation is further away from the true value, given the worst case-scenario. The integrality gap for (unweighted) Hitting Set can be shown to be of the order at most $\log m$ where m is the number of sets to be hit (or implications in our application) (Vazirani, 2001). Furthermore, it has been proven that Hitting Set cannot be approximated in polynomial time to within a factor of $c \log m$, unless NP has slightly superpolynomial time algorithms (Lund

and Yannakakis, 1994; Feige, 1998). This means that the solutions of other polynomial approximation algorithms will not be (much) better than the solutions of our relaxation.

Although LPs are solvable in polynomial time, the resulting multiple testing procedure can still be time-consuming. To speed up the procedure even further, it could be worth considering other approximation algorithms, such as greedy algorithms. One should keep in mind however that these algorithms must result in a conservative procedure, meaning that Hitting Set must be approximated from below.

In our data examples, it was feasible to compute both the relaxation and the ILP. The differences in performance will be shown in the Applications section.

4.3.2 Constructing confidence sets

To calculate the minimum number of false hypotheses within an arbitrarily chosen set $\mathcal{A}$ of k elementary or intersection hypotheses, we again need to solve Hitting Set. Remember that, given a rejection set $\mathcal{R}$, the confidence set for the number of false hypotheses in $\mathcal{A}$ is of the form $\{m, \ldots, k\}$, with

$$m = \min_{\mathcal{S} \supseteq \mathcal{R}} |\mathcal{A} \cap \mathcal{S}|\,, \tag{4.12}$$

where the minimum is taken over all possible congruent extensions $\mathcal{S}$ of $\mathcal{R}$.

We know that, in order for $\mathcal{S}$ to be a congruent extension of $\mathcal{R}$, for each implication $I \in \mathcal{R}$ at least one element of $\mathcal{L}_I$ should be included in $\mathcal{S}$. To find the minimum number of elements that have to be chosen within the set $\mathcal{A}$, we only have to consider those implications I for which $\mathcal{L}_I \subseteq \bigcup_{J \in \mathcal{A}} \mathcal{L}_J$, because for the other implications a leaf node that is no element or child of any hypothesis within $\mathcal{A}$ can be included in the congruent extension $\mathcal{S}$ without changing the value of m. Finding m now comes down to choosing a set of leaf nodes $\mathcal{L}'$ in such a way that for each remaining implication I, $L_I \cap \mathcal{L}' \neq \emptyset$ and $|(\mathcal{L}' \cup \text{an}(\mathcal{L}')) \cap \mathcal{A}|$ is minimal. Solving this problem means solving an instance of Hitting Set. Because the calculation only has to be done a few times (once for every chosen set $\mathcal{A}$), it will often be feasible to solve the corresponding ILP exactly.

4.4 Applications

In this section, we will apply the just described multiple testing method to a real data set in order to be able to answer the following two questions; "What are the differences in terms of performance and speed between the possible approaches that can be used in case of two-way logical relationships?" and "How well does our method perform compared to already existing FWER controlling methods?".

To answer those questions, we analyze a data set from Van De Vijver et al. (2002), which contains (possibly censored) survival times for 295 women diagnosed with breast cancer together with the gene expression profiles of 4919 gene expression probes. To test whether specific probes and sets of probes are associated with survival time, we first formed meaningful probe sets based on the gene ontology biological process graph. This

biological process graph consists of biological process terms, which represent groups of genes known to operate together within certain biological processes such as for example DNA repair or cell death.

The first collection of sets we investigate are all GO terms in the biological process graph. To create the corresponding probe sets, we took the top node with corresponding GO-id "GO:0008150" and all its descendants from the biological process ontology and mapped the probes from our data set to each of these GO terms. This resulted in 4952 unique probe sets for testing. For each probe set A, the corresponding null-hypothesis we want to test is the hypothesis that there is no association between the probes in this probe set and the measured survival time, as given by:

$$H_A : \beta_i = 0 \qquad \forall i \in A,$$

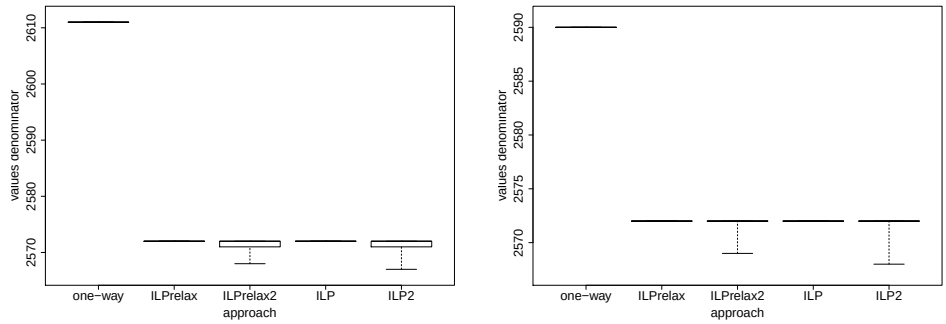
where the β_i's are the regression coefficients in a Cox proportional hazards model containing only the probes from probe set A as covariates. We used the global test of (Goeman et al., 2004) to test these null-hypotheses. This test has most power against alternatives in which many small effects are present, which is the situation we expect to be in. All the obtained null-hypotheses can subsequently be structured within a DAG, where H_A is the parent node of H_B if $A \supset B$ and there is no H_C with $A \supset C \supset B$. The resulting DAG by construction contains one-way logical relationships.

We followed the same procedure to obtain smaller DAG-structures. This time taking the GO terms "DNA repair" (GO:0006281), "apoptosis" (GO:0006915) and "cell proliferation" (GO:0008283) as top node of our graphs. This resulted in DAGs with respectively 22, 92 and 126 nodes.

Apart from DAGs containing only probe sets from the biological process ontology, we also extended each DAG to a DAG containing the probe sets as well as all individual probes present in these probe sets. For the DAG containing all probe sets, the extended version contained 6971 nodes (compared to the former number of 4952). The DAGs with respectively "DNA repair", "apoptosis" and "cell proliferation" as top nodes contained 70, 461 and 494 nodes after extension. In the resulting graphs, each hypothesis is the intersection of its child hypotheses and for that reason we could use the potentially more powerful versions of the any-parent and all-parents rule, as described in section 4.2.3.

4.4.1 Comparison of the approaches that can be used when two-way logical relationships are present

As mentioned before, there are multiple possibilities to increase the power of our methods in case of two-way logical relationships. To compare the properties of the various approaches in terms of power gain and computational cost, we first applied both the any-parent and the all-parents rule for one-way logical relations to the DAG containing all possible GO terms and the individual probes, and stopped after these variants could not find new rejections on an overall α-level of 0.05. Let us denote the corresponding rejection sets both with $\mathcal{R}$. Subsequently, for both variants, we changed the way of calculating



all-parents method any-parents method

Figure 4.2: Values of the denominator as given by different optimization methods for DAGs where two-way logical relationships are present, for one iteration in the algorithm.

the α-levels by using one of the four approaches that are only valid in case of two-way logical relationships. As described in the previous sections, all these approaches work by calculating or approximating the maximum value of the denominator in equation (4.8). Lower values will result in higher α-levels and possibly more rejections. For each approach and each candidate, the value of the denominator was calculated.

In figure 4.2, boxplots of the resulting denominator values per approach are given. Because we used weights v_L equal to 1, the denominator values can be interpreted as the maximum number of simultaneously true leaf nodes given the rejection set $\mathcal{R}$. The denominator values given in the "one-way approach" column are just the numbers of unrejected leaf nodes after performing both multiple testing rules without using the two-way logical relationships. The numbers corresponding to the "ILPrelax approach" and the "ILP approach" are the denominator values obtained from running respectively the relaxation of the ILP or the ILP itself for all candidates simultaneously. The boxplots corresponding to the "ILPrelax2 approach" and the "ILP2 approach" are the denominator values from respectively running the relaxation or the ILP itself for all candidates separately. We see that there is a clear difference between the non-optimized value for the denominator and the values obtained from one of the more optimized approaches. However, the differences between the four optimization approaches are small. Solving either the ILP or its relaxation often leads to the exact same value for the denominator, meaning that we are in a situation wherein the actual difference between the ILP and its relaxation is far away from the theoretical bound on the integrality gap as formulated in equation (4.11).

In table 4.1, we see whether the optimized values for the denominators also lead to more rejections and how much computation time is involved in computing the denominators once for all candidates. In this particular case, all approaches lead to the same

Table 4.1: Summary of the behavior of the different methods for optimizing the denominator.

	all-parents method		any-parents method	
	extra rejections	extra comp. time	extra rejections	extra comp. time
ILPrelax	3	0.06s	3	0.02s
ILPrelax2	3	55.02s	3	77.22s
ILP	3	0.14s	3	0.04s
ILP2	3	121s	3	96.54s
exact	-	-	3	176.96s

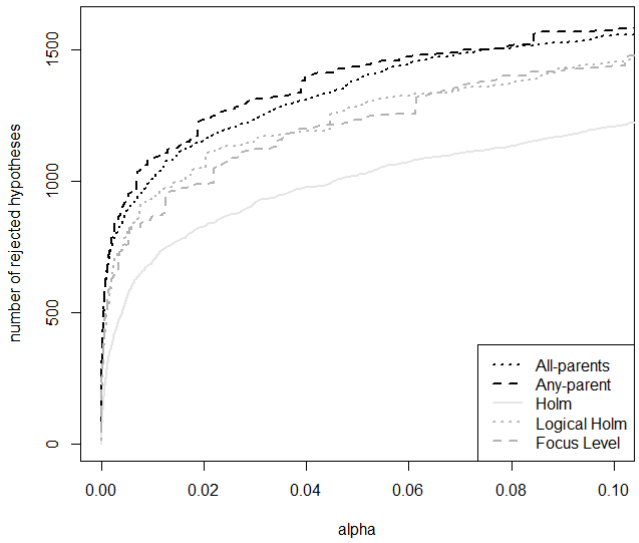
number of extra rejections, namely 3, compared to using the one-way approach. For the any-parent variant we can also compare these numbers to the maximal possible number of extra rejections, as given by the exact method outlined in the appendix that optimizes both the numerator and denominator. We see that all optimization approaches find the maximum number of rejections possible. The time to optimize the denominator for all candidates simultaneously is small, even for the real ILP optimization, but becomes considerable when the denominator is optimized for all candidates separately.

Although we only showed the results for one graph and an overall α-level of 0.05, we found the same behavior for other α-levels and the smaller DAGs. Usually, a few rejections extra can be achieved by using a method specifically designed for using the presence of two-way logical relationships, but the gain is generally small. The relaxation performs very well compared to the actual ILP, but is also not that much faster. We have to stress however, that these results not necessarily carry over to different graphs. As described in the Algorithms section, ILP problems can be very difficult to solve, while their relaxations are more easily solvable but not always as accurate as in our examples.

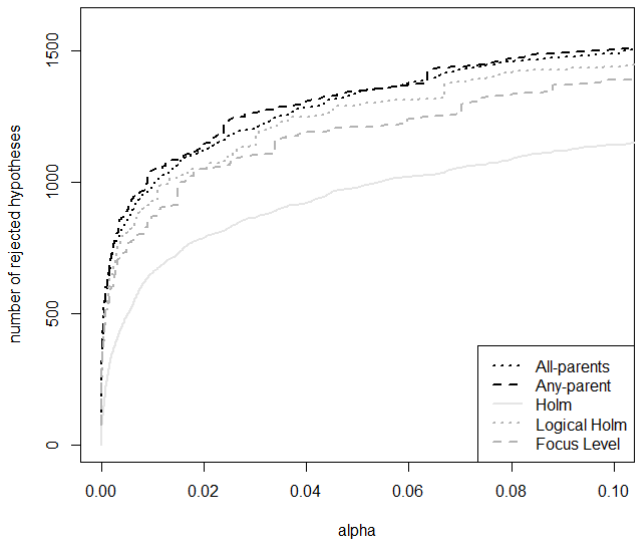
Because the actual choice for one of the optimization methods in case of two-way logical relationships does not influence the actual performance of the method that much, in the upcoming comparisons between our own method and existing methods we always use the approach that optimizes the denominator by solving the actual ILP for all candidates simultaneously. This approach is close to optimal and fast enough to enable us to calculate adjusted p-values.

4.4.2 Comparing the any-parent and all-parents rule to existing FWER controlling methods

In order to compare the any-parent and the all-parents rule to existing multiple testing methods, in all eight DAGs (4 with one-way and 4 with two-way logical relationships), we located the significant probe sets by means of these two variants and three already existing methods. Because we only want to compare our method to similar procedures, we only



one-way logical relationships



two-way logical relationships

Figure 4.3: Number of rejections as a function of the overall α-level in the DAG with all biological process GO-terms and its extended version.

compared it to other FWER controlling methods. We used both the Focus Level procedure of Goeman and Mansmann (2008) where the actual focus level was determined by using the default values of the procedure as implemented in the R-package `globaltest`, and two versions of Holm's multiple testing procedure (Holm, 1979). Naturally, the first version consists of applying Holm's procedure directly to all probe sets included in the DAG, without using the DAG structure. In the second version, Holm's method is again applied to all the nodes in the graph, but this time the ancestors of the rejected nodes are subsequently rejected as well. By the presence of logical relationships, this variant will still control the FWER and will always be at least as powerful as the normal Holm procedure. We will refer to this variant as the "Logical Holm" method. The difference in results between Holm's method and the Logical Holm method shows what can be easily gained by using the relationships that are present in the graph.

In figure 4.3, for the DAG containing all GO-terms (top) and its extension containing in addition all individual probes (bottom), the number of rejections at different overall α-levels for the different procedures is reported. We see that in both situations, the all-parents and any-parent variant outperform the existing methods for all α-values. Similar figures were made for the DAGs with different top nodes. In those figures we saw the same behavior as for the DAG with cell proliferation as its top node and all probe sets included. However, in all other DAGs there is no method that uniformly outperforms the others.

4.4.3 A simulation study to further explore the method's behavior

In addition to the analysis of a real data set, we also performed a simulation study in order to get more insight in the behavior of our new methods. We have to stress however, that this simulation study is only a first step in trying to get a better idea of when to use what multiple testing method. There are many factors that can be varied, but in order to keep the simulation study feasible, we could not look into all of these.

The first important choice is how to choose the graph structure. This is a far from trivial decision. The DAG can be chosen to resemble a tree structure, or to have many nodes with multiple parents. It can be large or small, the edges can be chosen in a randomized manner or in a more deterministic way. In order to keep it simple, we chose not to create our own (arbitrary) DAG structure, but to use the DAG structures from our real data example. We chose to use the one-way and two-way DAG structure of the full GO DAG, containing respectively 4952 and 6971 nodes. In addition, we also used the one- and two-way DAG structure from the DAG with cell proliferation as its top node with respectively 126 and 494 nodes.

Another difficult choice is how to generate and spread signal through the graph structure. We chose to spread signal in two different ways. The first approach was to choose an existing node in the DAG structure, and to make the outcome variable depend on its k corresponding leaf nodes. The second approach was to choose k leaf nodes, out of a total number of p leaves, at random, and to make the outcome variable depend on those. To

make the outcome variable depend on the k leaf nodes, we took the following approach: we generated an n-dimensional outcome vector $\boldsymbol{y}$ as follows:

$$\boldsymbol{y} = \beta \boldsymbol{z} + \boldsymbol{u},$$

where β is a scalar and the vector $\boldsymbol{z}$ as well as the error term $\boldsymbol{u}$ are n-dimensional standard-normal vectors. Subsequently, we generated a $n \times p$ design matrix X with k influential columns (corresponding to the leaf nodes), where each influential column is given by

$$\boldsymbol{x}_i = \boldsymbol{z} + \boldsymbol{\epsilon}_i,$$

where $\boldsymbol{\epsilon}_i$ is again standard normally distributed. All remaining uninfluential columns are filled by randomly generated variables v_i with $v_i \sim \mathcal{N}(0, 1)$.

Important factors that can be varied are the number of influential leaf nodes k and the effect size (as given by β). With multiple values for k as well as β, we performed 100 simulations per setting. The number of overall covariates p was determined by the DAG structures we chose (so 4 different values) and the number of subjects n was set to 200. Although the exact position of the k influential leaf nodes within the DAG could be important as well, we did not study this potential effect. The test we used was again the global test and the FWER was controlled on $\alpha = 0.05$.

In every setting we kept track of four outcome measures, namely the total number of hypotheses that could be rejected by the any-parent approach, by the all-parents approach, by Logical Holm and by the Focus Level procedure. Unfortunately, the Focus Level procedure was too time-consuming to use repeatedly on the larger DAGs, so we could only include it in the comparison when using the cell proliferation DAG. For each of the four DAG structures, we produced two tables, one for the randomly selected leaf nodes, and one for the leaf nodes selected by using an existing gene set, to summarize our results. We chose to only show two of those tables here, namely the tables for the one- and two-way structure of the cell proliferation DAG (on which we could also use the Focus Level procedure) where the leaf nodes are selected based on an existing gene set.

From Table 4.2 and 4.3, we see that our new approaches do rather well compared to Logical Holm when the effect size β is small to moderate. Although the differences are not significant, our new methods seem to find more rejections. For larger effect sizes, this effect seems to disappear however. In comparison with the Focus Level procedure, we see little difference with our methods on the one-way DAG, but for the two-way DAG in combination with a small to moderate effect size, our new approaches seem to perform a bit better than the Focus Level procedure. We were expecting that Logical Holm would perform better in cases where the signal was induced via randomly selected leaf nodes, but also in those tables (although not shown here) we see approximately the same results. For the large DAGs, based on all GO terms, in which we can only use Logical Holm as a comparison, we see hardly any differences between the three methods for the small and large effect sizes. For the moderate effect size ($\beta = 0.2$), our new procedures again seem to do a bit better than Logical Holm, although it is no significant difference. From the simulations, we can furthermore not draw conclusions on when to use the any-parent or

cellproliferation DAG, oneway, genes chosen from gene set						
			multiple testing method			
beta	# genes	# sets	all-parents	any-parent	Focus Level	Logical Holm
0.1	4	28	0.73 (0.22)	0.65 (0.19)	0.76 (0.28)	0.40 (0.19)
	10	55	2.84 (0.68)	2.52 (0.62)	2.65 (0.70)	1.81 (0.53)
	23	99	1.62 (0.63)	1.46 (0.60)	2.00 (0.75)	1.11 (0.47)
0.2	4	28	5.03 (0.74)	4.58 (0.70)	4.59 (0.80)	3.40 (0.69)
	10	55	11.90 (1.44)	11.28 (1.40)	11.67 (1.60)	8.83 (1.35)
	23	99	15.77 (1.99)	13.63 (1.81)	16.02 (2.05)	10.06 (1.72)
0.5	4	28	25.00 (0.58)	24.78 (0.61)	25.34 (0.58)	23.94 (0.70)
	10	55	51.53 (0.61)	51.16 (0.68)	52.73 (0.55)	50.55 (0.75)
	23	99	95.11 (0.96)	94.56 (1.06)	95.55 (0.87)	94.06 (1.22)

Table 4.2: Mean number of rejections for four different methods over 100 iterations per case. The standard error of the mean is denoted in brackets. Number of genes are the number of genes that are associated with the response. These genes are chosen such that the group of all genes is an actual node in the graph. Total number of genes is 43. Number of sets are the total number of nodes that are associated with the response. Total number of sets (including single genes) is 126.

cellproliferation DAG, twoway, genes chosen from gene set						
			multiple testing method			
beta	# genes	# sets	all-parents	any-parent	Focus Level	Logical Holm
0.1	11	53	1.42 (0.39)	1.16 (0.31)	0.18 (0.09)	0.45 (0.21)
	28	70	1.35 (0.42)	1.21 (0.39)	0.78 (0.32)	1.26 (0.41)
	63	137	1.78 (0.53)	1.40 (0.44)	1.14 (0.39)	0.98 (0.36)
0.2	11	53	7.02 (1.02)	6.58 (1.01)	4.14 (0.90)	4.23 (0.95)
	28	70	8.93 (1.07)	7.90 (1.01)	5.09 (0.93)	5.43 (0.94)
	63	137	15.15 (1.99)	13.01 (1.88)	10.18 (1.73)	11.00 (1.77)
0.5	11	53	46.33 (0.83)	46.20 (0.85)	46.34 (0.96)	46.03 (0.96)
	28	70	60.64 (1.02)	60.40 (1.06)	58.99 (1.25)	59.94 (1.18)
	63	137	118.8 (2.10)	118.2 (2.19)	117.4 (2.31)	117.4 (2.33)

Table 4.3: Mean number of rejections for four different methods over 100 iterations per case. The standard error of the mean is denoted in brackets. Number of genes are the number of genes that are associated with the response. These genes are chosen such that the group of all genes is an actual node in the graph. Total number of genes is 398. Number of sets are the total number of nodes that are associated with the response. Total number of sets (including single genes) is 494.

the all-parents approach. More research and simulations would be needed to answer that question.

From the simulations we see that, in specific situations, we can gain from using our new multiple testing approaches in comparison to already existing methods. However, there are many possible scenarios that we did not look into, and there will certainly be situations in which Logical Holm or Focus Level will be the preferred multiple testing method. Our new approaches should mostly be seen as a useful alternative to the already existing methods, for which the top-down testing direction is a novelty, but it will definitely not be the best option in all situations. In general, we should not expect any DAG method to uniformly outperform other DAG methods. The Logical Holm procedure will be a good choice when there are a few small effects present in the DAG. The Focus Level procedure benefits from the fact that it will always create two-way logical relationships in the lower part of the DAG, determined by the chosen focus level, by using a closed testing procedure. But when the nodes in the graph have many children and there are many leaf nodes, the focus level procedure can only choose its focus level close to the leaf nodes of the graph. Our own methods will be a good choice when a significant signal is expected to be found on higher levels in the graph, but not necessarily on lower levels.

4.5 Discussion

We have presented a multiple testing method that can be used for testing hypotheses that are structured in a directed acyclic graph. The only assumption on this DAG structure is that the truth of a parent node implies the truth of all its children. Our method is a top-down method that comes in two variants; in the first variant a node can only be tested when all its parents are already rejected, in the second variant a node can be tested as soon as one of its parents is rejected. Both variants strongly control the familywise error rate without making additional assumptions on the joint distribution of the test statistics used to calculate the individual raw p-values. Furthermore, the method makes use of logical relationships between the hypotheses to further enhance its power in case of restricted combinations.

Our method can be seen as a generalization of the multiple testing method for tree-structured hypotheses proposed by Meinshausen (2008). When appropriate initial weights are chosen for the leaf nodes, both variants of our method reduce to Meinshausen's procedure when they are applied to a tree-structure.

A competitive FWER controlling procedure that can be applied to DAGs is the Focus Level procedure of Goeman and Mansmann (2008). The Focus Level procedure is not a top-down method however and was not originally designed for testing groups of variables as well as the individual variables itself, as would for example be the situation when one wants to test gene sets as well as individual genes for association with a certain response. In such situations, our method can easily be applied, but the Focus Level procedure will not have much freedom to choose its focus level for computational reasons and will closely resemble Holm's procedure on the leaf nodes of the graph, where the

parents of rejected children will be rejected as well. Although applying Holm only on the leaf nodes and subsequently rejecting all ancestors of the rejected nodes, is an interesting strategy and will sometimes even be the optimal strategy if one is mainly interested in finding rejections in the leaf nodes (e.g. in finding influential genes), in this article we focused on methods that are non-consonant and able to find nodes higher in the graph, even if all their offspring nodes are non-significant.

Although we only exemplified our method on graphs with one specific top node, our method can, without any alterations, be applied to graphs having multiple top nodes. If we again look at the gene ontology terms for example, instead of testing all GO terms we could also limit ourselves to testing only those terms that contain no more than some pre-specified number of genes. In this situation, even though we use a top-down approach, we don't have to prove global association on a single top node.

Another feature of our method that we did not explore yet is the possibility to give different weights to different leaf nodes. Different weights could for example reflect that some genes and their corresponding gene sets are a priori more likely to be associated with the outcome.

Other possibilities for future work could be to change the way the initial weight is flowing through the graph. For now, each child node divides its weight equally over its unrejected parents, but variations on this type of weight flow could be considered. Even the direction of the weight flow, which is now from leaves to the root, could be changed to go from the top node(s) to the leaves, resulting in a procedure that gives more weight to nodes that can be accessed via multiple rejected parent nodes. Many potential variants can thus still be explored.

In addition to exploring new multiple testing variants, examining the strengths and weaknesses of existing methods in more detail will also be very valuable. More simulation studies should be performed to give a better idea of when to use what procedure.

Appendix

In this appendix, we will discuss the ILP that can be used to determine whether a certain candidate node H can be rejected by the any-parent multiple testing method that uses the presence of two-way logical relationships, given a rejection set $\mathcal{R}$. The ILP will make the decision whether the quantity given in (4.8) (i.e the minimum α-level candidate H can receive based on any congruent extension of the current rejection set) is bigger than or equal to the raw p-value p_H. If so, this candidate can be rejected. Although we cannot calculate the actual minimum, we can determine whether this value exceeds p_H and we can therefore use this ILP to find the exact number of rejections for a certain overall α-level.

If we again introduce a binary variable X_i for each leaf node L_i, and moreover create a binary variable Y_j for each of the m parent nodes P_j of H, the α-level given in (4.7)

can for each congruent extension $\mathcal{S}$ of $\mathcal{R}$ that does not contain H, be written as

$$\frac{\frac{w_H^{max}}{m} \sum_{j=1}^{m} Y_j}{\sum_i v_{L_i} - \sum_i v_{L_i} X_i} \alpha,$$

where X_i and Y_j are 1 if the nodes they correspond to are included in the rejection set $\mathcal{S}$ and 0 otherwise. The value for w_H^{max} does not depend on $\mathcal{S}$ and represents the total incoming weight H will (always!) receive from its offspring nodes. If we substitute constants $c_1 = \frac{w_H^{max}}{m}$ and $c_2 = \sum_i v_{L_i}$, the α-level as given in (4.8) can now be written as:

$$\min_{\mathcal{S}} \frac{c_1 \sum_{j=1}^{m} Y_j}{c_2 - \sum_i v_{L_i} X_i} \alpha,$$

where the minimum is taken over all previously specified rejection sets $\mathcal{S}$, and for which we want to evaluate whether or not it exceeds p_H.

To be able to answer this question we can rewrite this problem into the following ILP:

$$\begin{aligned}
\textbf{minimize} \quad & c_1 \sum_{j=1}^{m} Y_j - \frac{p_H}{\alpha} \left(c_2 - \sum_i v_{L_i} X_i \right) \\
\textbf{with respect to} \quad & \sum_{i:\, L_i \in \mathcal{L}_I} X_i \geq 1, \quad \forall I \\
\textbf{and} \quad & \sum_{i:\, L_i \in \mathcal{L}_H} X_i = 0 \\
\textbf{and} \quad & \sum_{i:\, L_i \in \mathcal{L}_{P_j}} X_i \leq |\mathcal{L}_{P_j}| Y_j, \quad \forall P_j \in \text{pa}(H) \\
\textbf{and} \quad & X_i, Y_j \in \{0, 1\}.
\end{aligned}$$

The first type of constraint ensures that for each implication $I \in \mathcal{R}$, at least one of its corresponding leaf nodes is added to the congruent extension of $\mathcal{R}$. The second type of constraint states that H is no part of $\mathcal{S}$, whereas the third type of constraints makes sure that whenever a leaf node of P_j is in $\mathcal{S}$, the same holds for P_j itself.

If and only if the minimum value found by this ILP is non-negative, we can be sure that H can be rejected on overall α-level α.