ABSTRACT

Motivation: Distinguishing direct from indirect influences is a central issue in reverse engineering of biological networks because it facilitates detection and removal of false positive edges. Transitive reduction is one approach for eliminating edges reflecting indirect effects but its use in reconstructing cyclic interaction graphs with true redundant structures is problematic.

Results: We present TRANSWESD, an elaborated variant of TRANSitive reduction for WEighted Signed Digraphs that overcomes conceptual problems of existing versions. Major changes and improvements concern: (i) new statistical approaches for generating high-quality perturbation graphs from systematic perturbation experiments; (ii) the use of edge weights (association strengths) for recognizing true redundant structures; (iii) causal interpretation of cycles; (iv) relaxed definition of transitive reduction; and (v) approximation algorithms for large networks. Using standardized benchmark tests, we demonstrate that our method outperforms existing variants of transitive reduction and is, despite its conceptual simplicity, highly competitive with other reverse engineering methods.

Contact: klamt@mpi-magdeburg.mpg.de

Supplementary information: Supplementary data are available at Bioinformatics online.

1 INTRODUCTION

Reverse engineering of cellular networks has become a key methodology in analysing and exploiting the increasing amount of data generated by omics technologies (Gardner and Faith, 2005; Hecker et al., 2009; Markowetz and Spang, 2007). Whereas the structure of metabolic reaction networks could be reconstructed—mainly from genomic information—in great detail for many organisms (Oberhardt et al., 2009) knowledge of the topology of regulatory and signal transduction networks is in many cases still incomplete and wiring diagrams even of ‘canonical signalling pathways’ may differ in different cell lines (Saiz-Rodriguez et al., 2009). The ultimate goal of reverse engineering methods is the identification of interactions between the involved players (genes, proteins, etc.) by analysing data of systematic and controlled perturbation experiments. The result is a network, in many cases represented as a graph, which can be directed or undirected and may have signs and/or weights at its edges. Some algorithms deliver refined representations such as Boolean networks (Akutsu et al., 2003; Saez-Rodriguez et al., 2009), reaction networks (Durzinsky et al., 2008) or differential equations (Nelander et al., 2008) but the main result is still the underlying network topology.

A simple yet smart method for reverse engineering is based on Transitive Reduction, a graph theoretical method (Aho et al., 1972) whose potential for reconstructing regulatory networks was first recognized by Wagner (2001). The basic idea is as follows: to reconstruct a regulatory network with n nodes, one first measures the state of the nodes in the wild-type and then performs at least n perturbation experiments: in experiment i node i is perturbed, whereas all other n-1 nodes are screened whether they changed their state compared to the wild-type. If a perturbation in i affected j, a directed edge from node i to j, denoted by i → j, is drawn. The complete set of these observed effects in all perturbation experiments yields the perturbation graph. Each edge in the perturbation graph reflects either a direct or an indirect effect of one node upon another. The next step deals with a central issue in network reconstruction, namely identification and removal of edges that represent indirect effects. Transitive reduction as used by Wagner (2001) aims at finding the minimal (most parsimonious) subgraph that can explain all effects seen in the experiments. Transitive reduction in its most general form allows removal and addition of edges to find the minimum graph (Aho et al., 1972). However, in the context of network reconstruction, one usually focuses on the special case where edges may only be removed, i.e. where one searches for a minimal subgraph explaining the perturbation graph [also known as minimum equivalent graph problem (Berman et al., 2009; Moyle and Thompson, 1969)]. Herein, we only consider transitive reduction based on edge removals. Wagner (2001) determined the minimal subgraph from the perturbation graph by removing all edges i → j for which a (simple) path starting in i and ending in j (not using i → j) can be found, assuming the effect of i on j to be indirect, thus explainable by the path. The resulting graph is the transitive reduction of the perturbation graph. A simple example is depicted in Figure 1a. Every acyclic graph has a unique transitive reduction (with a minimal number of edges; Aho et al., 1972) explaining all measured perturbation effects.

The method proposed by Wagner (2001) is easy to implement but has some drawbacks that might be the reason for its rare application. First, transitive reduction as described above does not consider the full amount of information that is available from perturbation experiments, even when considering only qualitative observations. If a node shows a significant response to a perturbation, one can at least classify the measured effect as ‘up’ or ‘down’. This information
Network inference with TRANSWESD

Fig. 1. Examples of perturbation graphs and their transitive reductions. See text for discussion and explanations.

2 METHODS

2.1 Definitions

We summarize some standard terminology and notations from graph theory as will be used herein. A graph \( G = (V, E) \) consists of a set \( V \) of nodes (or vertices) and a set \( E \) of edges connecting pairs of nodes. In our case, nodes may represent genes, mRNA, proteins etc., whereas edges correspondingly represent physical or influential node-to-node interactions. We will only be concerned with directed graphs (digraphs) where edges are directed (also called arcs), i.e., \( e = (u, v) \) is an ordered pair of distinct nodes \( u, v \in V \), also denoted by \( u \rightarrow v \), where \( u \) is the start node and \( v \) the end node. A signed digraph \( G = (V, E, \varphi) \) contains additionally a sign mapping \( \varphi : E \rightarrow \{-, +\} \) indicating for each edge whether its start node has a promoting or inhibiting effect on its end node. A directed edge \((u, v)\) with sign \( s\) and weight \( w\) is denoted \( u \rightarrow_s w v\). A walk in a digraph is an alternating sequence of nodes and edges \( v_0, e_1, v_1, e_2, \ldots, e_n, v_n \) starting and ending with node \( v_0 \) and \( v_n \), respectively, which fulfills the condition that nodes \( v_{i-1} \) and \( v_i \) are connected by the
2.2 Workflow of the whole procedure

We start with a general outline of our algorithm.

Step 1: As explained in the introduction, starting point is a wild-type experiment plus n perturbation experiments in each of which one of the n nodes is perturbed and the resulting response of the other nodes is measured, either in transient phase or in steady state (we assume the latter if not stated otherwise). We denote the wild-type states with \( u_i \) (i.e. the state of the \( i \)-th node) and assume that the measurements for each species \( i \) is denoted by \( x_i \). The vector of normalized steady states is denoted by \( \mathbf{x}\).

Step 2 (Section 2.3): for each node, we compare the unperturbed state \( x_i \) with the perturbed states \( x_i \) (e.g. by over-expression) or \( x_i \) (e.g. by knockdown). The amount/activity of node \( k \) after a perturbation is denoted by \( A_k \), i.e. \( A_k \in \{0,1\} \). The vector of normalized steady states measured in the \( k \)-th perturbation experiment (where node \( k \) is perturbed) is denoted by \( \mathbf{x}_k \), i.e. \( \mathbf{x}_k \in \mathbb{R}^n \). The normalized steady states of wild-type and of perturbed states for edges \( e \) that do not have a causal explanation in the true network—we introduce a threshold strategy for obtaining a high-quality perturbation graph defining the strength of associations. Hence, for each ordered pair of nodes \( (u,v) \) from the perturbation graph, one can either use correlation analysis of the entire data or only direct variation measures quantifying the change of \( x_i \) when perturbing \( x_i \). The correlation measure of the entire data is beneficial for determining the strength of association between nodes (see Section 2.4) but disadvantageous for detecting the direction of interaction. We, therefore, make use of a direct variation measure allowing us to detect sign and direction of edges. In the first place, we might initially ignore the presence of noise and define the variation measure for node pair \((k,l)\) as \( \Delta x_{kl} = \mathbf{x}_k - \mathbf{x}_l \) (p is the indicator of direction as described above). Initially, we may introduce edges \( k \rightarrow l \) if \( \Delta x_{kl} < 0 \) and \( k \rightarrow l \) if \( \Delta x_{kl} > 0 \) yielding a signed perturbation graph. Clearly, this graph will capture true direct as well as indirect effects. However, since experimental data are subject to stochastic fluctuations due to measurement and intrinsic noise, many non-zero \( \Delta x_{kl} \), and thus edges \( k \rightarrow l \) would not correspond to true causal influences, neither direct nor indirect ones. A naive use would thus lead to a very dense perturbation graph (in the extreme case, all nodes are connected to each other), which essentially contains only little meaningful wiring information of the real graph. To reduce the number of FP edges—in particular, those that do not have a causal explanation in the true network—we introduce two threshold parameters for the magnitude of the variation measure \( |\Delta x_{kl}|\). This is motivated by the assumption that most of the true interactions produce detectable experimental signals that can be distinguished from fluctuations due to noise and, in some cases, indirect interactions. Consequently, true interactions that produce insufficient variations cannot be reconstructed from the data.

The threshold \( \Delta \) is introduced to set a required overall minimal magnitude of the variation measure when searching for edges. It is kept constant for all pairs of nodes. The second threshold \( \beta \) accounts for the individual dynamic nature of each node and is derived from the variance of node \( i \)'s entire perturbation profile excluding the perturbation of node itself.

We thus introduce an edge from node \( k \) to \( i \) in the perturbation graph only if the two conditions, (i) \( |\Delta x_{ki}| > \Delta \) and (ii) \( |\Delta x_{ik}| > \beta \), are met. We compute \( \beta = \Gamma \), with variance scaling factor \( \Gamma \) and SD \( \sigma_i \) of gene \( x_i \). Notice that depending on the chosen values for \( \Delta \) and \( \Gamma \) and the fluctuations of node \( x_i \), we either have \( \Delta > \beta \) or \( \Delta > \beta \). Benchmark tests (Section 3) indicated that Condition (i) or (ii) alone leads to weaker prediction performance (results not shown). Employing only Condition (i) neglects individual node dynamics. For instance, edges of nodes that have small absolute variations due to suppression by other nodes are likely to be missed. Condition (ii) alone is error prone to experimental data from dense graphs, which raises the probability to measure noise. Magnitudes for both parameters \( \Delta \) and \( \Gamma \) may be estimated from perturbation data of known interaction graphs, which should be functionally close to the investigated system. Alternatively, if the noise distribution function is known, it is straightforward to calculate the thresholds for a given P-value.

A suitable threshold strategy for obtaining a high-quality perturbation graph from noisy data is an important step, since there is a critical edge density for each graph up to which transitive reduction-related algorithms work well in terms of pruning result and computational time. Whereas edges reflecting indirect effects may be filtered by TRANSWESD at a later stage (see below), edges indicating neither direct nor indirect (thus noise) effects cannot be corrected and will lead to reconstruction errors. On the other hand, the number of FNs is also to be minimized as they cannot be recovered by transitive reduction. An example illustrating our thresholding strategy is given in the Supplementary Material and we note that our approach has some parallels to noise learning models as proposed by Yip et al. (2010) for filtering non-deterministic effects. The main difference is that our approach does not assume a certain kind of noise distribution function.

2.4 Quantifying the strength of associations

For our variant of transitive reduction, we need to assign weights to the signed edges in the perturbation graph that quantify the strength of the directed relationships. Hence, for each ordered pair of nodes \( (u,v) \), we determine the pairwise conditional correlation \( \rho_{uv} \) from the \( u \)-th and \( v \)-th element of the measured state vectors \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \) (cf. Rivas et al., 2005) \( \rho_{uv} \) is computed as linear correlation coefficient where we exclude the data from the \( v \)-th experiment because we want \( \rho_{uv} \) to quantify the dependency of \( v \) on \( u \) and the external perturbation in \( v \) cannot be explained by \( u \). Accordingly, \( \rho_{uv} \) is not symmetric.

For each edge \( u \rightarrow v \) captured in the perturbation graph \( P \) derived in Section 2.3, we assign its weight to be \( 1 - |\rho_{uv}| \), i.e. the smaller the weight the higher the association. This weighting scheme, where an edge weight indicates the ‘distance’ between the behaviours of two nodes, is somewhat contrary to other works where a large weight usually indicates a high association. However, we need this representation because we will employ shortest path calculations to find paths with highest overall associations (lowest overall weights).
2.5 Transitive reduction with TRANSWESD

At this stage, we have obtained a weighted, signed and directed perturbation graph $P = (V, E, \omega, \sigma)$, where we assume that its edges display direct or indirect relationships. Keeping edges capturing indirect effects would result in FP predictions and transitive reduction seeks to remove FPs to obtain true graphs. We, therefore, remove removable edges, we obtain the unique minimal equivalent graph $PTR$. The positive factor $\alpha$ is elementary, i.e. $\alpha<1$ (but it would be with $\alpha=1$). We may employ the double label algorithm, a generalized version of the Dijkstra algorithm for computing shortest positive/negative paths in $\Sigma_1$. It delivers exact results in polynomial time if the signed graph is acyclic (Hansen, 1984; Klamt and von Kamp, 2009). We store the lengths of the shortest positive and negative paths in a matrix $S^+$ and $S^-$, respectively. For example, $S^+(u,v)$ stores the length of the shortest positive path from $u$ to $v$. An infinite length (inf) is stored if no path exists.

In a second step, we prune $P$ to the minimal graph $PTR$ (minimal with respect to number of edges) satisfying

\[ S^+(u,v) < \text{inf} \quad \text{for all removed positive edges } u \rightarrow v \text{ in } P \tag{2a} \]

\[ S^-(u,v) < \text{inf} \quad \text{for all removed negative edges } u \rightarrow v \text{ in } P \tag{2b} \]

In acyclic signed graphs, the unique solution can easily be found with the help of $S^+$ and $S^-$. We check for each edge $u \rightarrow v$ whether we can find a successor $z \in V$ of such that an edge $u \rightarrow z$ and a path $z \rightarrow v$ exist fulfilling the sign condition $q = t = \infty$ (this path exists if $S^+(z,v) < \text{inf}$). If so, we can conclude that the influence $u \rightarrow v$ can be explained by the augmented path $u \rightarrow z \rightarrow v$, which is ensured to be elementary as we have an acyclic graph. We, therefore, remove $u \rightarrow v$ and continue with the next edge. Note that it is not necessary to re-compute the shortest paths lengths $S^+$ and $S^-$ after removal of an edge $u \rightarrow v$ in all paths using this edge, we can replace the latter by $u \rightarrow z \rightarrow v$. Now, $z \rightarrow v$ is elementary, i.e. $\sigma(z,v) \neq 0$ does not contain edge $u \rightarrow z \rightarrow v$, and is thus a valid explanation for $u \rightarrow z \rightarrow v$. And, second, that we do not need to recompute $S^+$ and $S^-$ after removal of an edge. Therefore, similar as in the previous section, if Condition (4) is fulfilled for the removed edges, it will also be fulfilled for all other edges.

Figure 1c demonstrates that an edge is kept if alternative paths cannot explain its high association strength. In contrast to Figure 1b, $A \rightarrow s \rightarrow B \rightarrow s \rightarrow D$ has length 0.6 (i.e. is thus not a valid explanation when choosing $u < 1$ (but it would be with $u > 2$). We note that the triangle reduction scheme presented in Rice et al. (2005) uses an analogous version of the procedure described in this section, however, this scheme was only applied to triangles, i.e. an edge $u \rightarrow v$ was removed only if two consecutive edges $u \rightarrow z \rightarrow v$ is explained it.

2.5.2 Transitive reduction in signed and weighted acyclic graphs

As explained in Section 2.4, rigorous transitive reduction cannot detect redundant structures such as coherent feed-forward loops implying a possibly large number of FNs. An attenuated pruning strategy could be achieved by considering also edge weights quantifying the overall strength of the associations. We now allow the removal of an edge (and consider it as an indirect influence) only if its sign and also its weight can be explained by another path. Condition (2) is thus generalized now demanding that the pruned graph $PTR$ should be minimal and satisfy

\[ S^+(u,v) < \text{inf} \quad \text{for all removed positive edges } u \rightarrow v \text{ in } P \tag{4a} \]

\[ S^-(u,v) < \text{inf} \quad \text{for all removed negative edges } u \rightarrow v \text{ in } P \tag{4b} \]
cycles—an inherent property of many cellular networks. Feedbacks may not only lead to complex dynamic network behaviour, they also hamper the inference of causal relationships. It is thus not surprising that also transitive reduction becomes more complicated, not only in the structure of the algorithm but also in terms of computational complexity.

As in the acyclic case, our procedure TRANSWESD starts with the computation of shortest path lengths $S^+$ and $S^-$. Here, we face an intrinsic algorithmic problem: in graphs containing negative cycles this problem is known to be NP-complete for elementary paths (Lapbaugh and Papadimitriou, 1984). Fortunately, one can check with low computational demand whether a negative cycle exists or not. If not, we may again use the double label algorithm computing exact results in polynomial time. Even if negative cycles exist, it turned out that exactly shortest path computation is often possible in realistic cellular networks with several hundreds of nodes (such as the gene regulatory network stored in RegulonDB) by using a depth first search or special variants thereof (Klamt and von Kamp, 2009). The latter article also describes a polynomial algorithm that produced reasonable approximations in large-scale networks.

A second technical issue concerns the interpretation of causality in negative cycles. In Figure 1d, we see a small example of a perturbation graph containing the negative cycle $C \rightarrow -C \rightarrow -A \rightarrow C$. The key question is whether we consider the negative non-elementary path (walk) $A \rightarrow -C \rightarrow -A$ as a valid explanation for the negative influence $A \rightarrow -A$ we observe when perturbing $A$. With $u < 1$, sign and length of this walk would actually allow that. Treisch et al. (2007) considered walks as possible explanations and although Albert et al. (2007) did not consider weights, their approach is also based on this interpretation. This brings the advantage that one only needs to compute the shortest positive/negative walks, which is computationally easy (e.g. by an adapted Floyd-Warshall algorithm (Albert et al., 2007; Treisch et al., 2007)) in contrast to shortest elementary paths. However, we think that the negative edge between $A$ and $B$ should be kept for the following reasons: we assume that the network is in steady state when it is perturbed in $A$ (without loss of generality we assume an over-expression in $A$). The negative edge in the perturbation graph in Figure 1d indicates that we measured a decreased steady-state response in $B$ upon perturbating $A$. Since removal of edge $A \rightarrow -C \rightarrow -A$ destroys this degradation cycle, it is not surprising that also non-uniqueness may occur as illustrated in Figure 1g. In a first step, we may remove edge $A \rightarrow -A$ (with $u = 0.95$ valid by path $A \rightarrow -A \rightarrow C \rightarrow -A$ or, alternatively, by $A \rightarrow -A \rightarrow C \rightarrow -A \rightarrow C \rightarrow -A$). In a second step, we may either remove edge $A \rightarrow -C$ (explainable by $A \rightarrow -C \rightarrow -A \rightarrow C$) or edge $D \rightarrow -C$ (explainable by $D \rightarrow -C \rightarrow -A \rightarrow C \rightarrow -C \rightarrow -A \rightarrow C$). We can only remove one of both and then have to stop pruning because otherwise no explanation for the removed edge $A \rightarrow -A$ would remain in the network and thus violate Condition (4). Hence we may end up with two possible minimal solutions for the reconstructed graph. In general, such case can only occur if for a given edge at least two explaining paths exist and, again, if the network contains negative cycles. In our algorithm, we use a greedy strategy, i.e. in each iteration we try to remove the explainable edge with largest weight (lowest association strength) fulfilling Condition (4).

Accordingly, we proceed as follows: after computing $S^+$ and $S^-$, we use these matrices to detect potentially explainable edges. A potentially explainable edge $u \rightarrow -v$ is one where we can find a succession $r \neq \varnothing$ of $u$ such that an edge $u \rightarrow z \rightarrow v$ and a path $z \rightarrow v$ exist fulfilling the sign condition $q = r$ and the weight condition $\max(d) < c - a$. In contrast to acyclic networks, it may happen that the augmented path $u \rightarrow z \rightarrow v$ is not explainable because the path $\tau q z$ may run over $u$ thus introducing a cycle in $u$. Whether a candidate edge is really explainable will be seen when recalculating the path lengths after removal of this edge (see below). All potentially explainable edges are ordered with respect to their weights (highest first) and one now iterates over these edges in descending order. Hence, in Figure 1g, we would first remove $A \rightarrow -A$ and then $A \rightarrow -C$ and we have to keep $D \rightarrow -C$. This example illustrates also a fourth issue that we have to account for in cyclic perturbation graphs: in principle, $D \rightarrow -C$ could be explained by $D \rightarrow -C \rightarrow -A \rightarrow C \rightarrow -A$. However, as mentioned before, removal is not allowed because then influence $A \rightarrow -A$ originally contained in $P$ would not be explainable anymore in the pruned graph and Condition (4) would be violated. Thus, in cyclic graphs, if an edge might be explainable by a path we cannot expect that all other elementary paths remain intact when removing this edge. In our example, the original edge $A \rightarrow -A$ is not explainable anymore by the path $A \rightarrow -C \rightarrow -A \rightarrow C \rightarrow -A \rightarrow C \rightarrow -A$, if we removed edge $D \rightarrow -C$. The explanation of the latter, $D \rightarrow -C \rightarrow -A \rightarrow C \rightarrow -A$, cannot be embedded in the path $A \rightarrow -C$ since the resulting path would not be elementary. Again, negative cycles cause such complicated structures.

Before cutting an explainable edge, we therefore have to check whether after its removal the shortest path lengths $S^+$ and $S^-$ still fulfill Condition (4) in the resulting intermediate graph $P_I$. Accordingly, we have to recalculate the shortest signed paths. As this will be the most time-consuming part of the whole algorithm we may try to simplify this step, e.g. by fast computation of approximations of the shortest path lengths mentioned above. Furthermore, one may completely ignore the recalculcation step (i.e. use the original $S^+$ and $S^-$ throughout all iterations) and check after the whole procedure whether Condition (4) is violated in the pruned graph. In fact, as we will see in Section 4, in many realistic applications no or only few errors are introduced if $u < 1$ and (4) therefore holds. If not, one may accept a low number of errors or re-introduce edges of node pairs $(u, v)$ violating (4).

Using the exact algorithm, the resulting pruned graph fulfills Condition (4) but is not necessarily unique or minimal with respect to the total number of edges. However, in the Supplementary Material it is shown that the pruned graph is usually unique and minimal with respect to a partial order on a sorted list of edge weights. Furthermore, TRANSWESD also includes the special cases of acyclic weighted/unweighted perturbation graphs as discussed in previous sections: if no (experimentally derived) weights are available one would simply set all edge weights to 1 and $u = 0$. In those acyclic cases,
the solution is also minimal with respect to the total number of edges as in the original definition of transitive reduction. A pseudo-code version of the TRANSWESD algorithm is given in the Supplementary Material and an implementation has been integrated as API function in our MATLAB toolbox CellNetAnalyzer (Klaua et al., 2007).

3 RESULTS

Using the developed reverse engineering methodology presented herein, we took part in the fourth challenge of the Dialogue of Reverse Engineering Assessments and Methods (DREAM4) on in silico gene network reconstruction. The DREAM initiative offers a platform for objective assessment of rivalling methods based on in silico data providing a realistic scenario for high-throughput gene expression profiling and reconstruction of gene regulation networks (Marbach et al., 2009; Stolovitzky et al., 2007, 2009). From the DREAM4 challenge, we present the results of our method for the dataset Insilico_Size_100 subchallenge, which can be downloaded from the DREAM website (http://wiki.c2b2.columbia.edu/dream/index.php/The_DREAM_Project). Based on 5 sub-networks of 100 nodes sampled from gene networks of Escherichia coli and yeast, realistic kinetic models with randomly selected parameters were generated and simulated with GeneNetWeaver (Marbach et al., 2009) using stochastic differential equations. For reconstructing these networks, in silico measurement data were provided containing noisy steady-state mRNA expression levels of wild-type and single-gene knockout and knockdown experiments as well as time-series data. The gold standards of the five networks were provided after announcing the results of all submissions and we can thus compare our computed results to the real networks.

For each network, we first generated the perturbation graph as described above using the wild-type and knockout steady-state data. The two required parameters were trained from the DREAM3 challenges. Edge weights were computed as conditional correlation coefficients from knockout and knockdown data. The results were very similar when using only the knockout data. The provided time series data were not used at all. We then applied transitive reduction with TRANSWESD to the generated perturbation graphs yielding the final reconstructed graph. Found edges were sorted according to their weight required for performance analysis (relevant for AUROC (area under the receiver operator characteristics curve) and AUPR (area under the precision-recall curve) values; see below). For comparison with the method of Albert et al. (2007), we used NET-SYNTHESIS (Kachalo et al., 2008) to apply their algorithm of transitive reduction in the unweighted version of the perturbation graph. We also implemented the SOS (save our signs) pruning procedure proposed by Tresch et al. (2007). This transitive reduction method also operates on weighted graphs but differs from TRANSWESD in two key aspects: (i) as discussed in Section 2 and analogous to NET-SYNTHESIS, this pruning approach accepts non-elementary paths (containing negative cycles) for explaining edges and (ii) the length of a path is computed as the product of edge probabilities. Based on this metric, an edge $\epsilon \rightarrow v$ is removed if there is a (elementary or non-elementary) path $\epsilon \Rightarrow v$ of the same sign whose overall probability is larger than the probability of the edge $\epsilon \rightarrow v$. Actually, SOS pruning distinguishes two probabilities, one for having a positive edge ($p_+$) and one for having a negative edge ($p_-$). The sign of the edge is given by the sign of $p_+ - p_-$. For the DREAM setting, we used the same perturbation graph as for TRANSWESD and we assigned the conditional correlation coefficients either with full amount to $p_+$ (if $s$ is positive) or to $p_-$ (if $s$ is negative), whereas the other probability was set to zero.

Table 1 summarizes the results for all five networks and indicates for each inferred network the number of TP/FP/TN/FN edges, the computation time and standard statistical metrics assessing the quality of reconstructed networks. The latter were determined by the DREAM evaluation scripts and include AUROC, AUPR as well as pAUROC and pAUPR (probabilities that a given or larger AUROC/AUPR value is obtained by random network link permutation as estimated from 100100 runs; see Stolovitzky et al., 2009). As the AUPR value is more sensitive in sparse networks it is especially useful to assess the quality of reconstructed gene regulatory networks.

The $P$-values reveal that our method produces results that are significantly better than randomly chosen networks. This even holds for the perturbation graph alone indicating that an appropriate thresholding strategy for classifying observed changes as relevant or not delivers large amounts of meaningful information. Note that even the raw perturbation graphs obtained by our thresholding strategy outperform many solutions submitted to the DREAM4 challenge. Applying TRANSWESD to the perturbation graphs increased the AUPR value in four of five networks, whereas in one network (2) AUPR decreased marginally. As expected, the desired removal of FPs by transitive reduction is inevitably accompanied by the removal of some TPs. This often implies a decrease in the AUROC value. However, in most of the cases this reduction is about one magnitude lower than the improvement in AUPR. The positive effect of transitive reduction becomes more apparent when looking at the $P$-values and at the numbers of true and FPs. In Network 3, for example, the number of FPs could be reduced by 98 (from 291 to 193) sacrificing only 3 TPs (reducing the TPs from 85 to 82), which results in a moderate increase of AUPR from 0.309 to 0.326 and a more significant increase in the $P$-value from 6.23e-111 to 1.97e-116. The effect of TRANSWESD becomes even more pronounced if we take the same data but based on deterministic simulation without any noise (these data were provided when releasing the results of the challenge). For illustration, Table 1 displays the results when taking noise-free data for Network 5. Without noise we can choose small thresholds, e.g. $\theta = 0.005$ and $\Gamma = 0$. As expected, AUROC and AUPR are much higher already in the perturbation graph. But using TRANSWESD we can further increase the AUPR ($p$AUPR) value from 0.442 (6.82e-102) to 0.567 (9.11e-132). The number of FPs is reduced from 476 to 110, whereas the number of FNs increases moderately from 33 to 61. Hence, a perturbation graph with high quality increases the effectiveness of transitive reduction.

In Network 2, we observed a decrease of the AUPR measures when applying TRANSWESD, although the number of FPs is reduced by 71 compared to 12 additional FNs. Strikingly, when announcing the results of DREAM4 challenge, it was mentioned that Network 2 exhibited oscillations and that the knockout data provided represented, therefore, transient and not steady-state data. Under those conditions, some edges in the perturbation graph may correspond to perturbation responses resulting from the action of non-elementary influence paths (with negative cycles) leading to a higher error rate of TRANSWESD. As expected, the computational costs for generating the perturbation graph are constantly low. In contrast, running times for applying the full TRANSWESD algorithm vary for the
Table 1. Benchmark results: Networks 1–5 correspond to the five networks of the Insilico_Size_100 sub-challenge of DREAM4 for which noisy simulation data were provided for network reconstruction.

| NETWORK | network/reconstruction method | TP  | TN  | FP  | FN  | AUROC (pAUROC) | AUPR (pAUPR) | Running Time |
|---------|--------------------------------|-----|-----|-----|-----|----------------|--------------|-------------|
| NETWORK 1 (100 nodes, 176 edges) | Perturbation graph | 99  | 9495| 229 | 77  | 0.873 (7.91e-35) | 0.467 (6.21e-11) | <5 s |
|        | Unweighted perturbation graph + NET-SYNTHESIS | 67  | 9550| 74  | 109 | 0.856 (1.98e-32) | 0.394 (7.96e-03) | <5 s |
|        | Perturbation graph + SOS pruning | 97  | 9524| 200 | 79  | 0.869 (2.78e-34) | 0.465 (2.27e-10) | <5 s |
|        | Perturbation graph + TRANSWESD | 97  | 9562| 162 | 79  | 0.870 (1.88e-34) | 0.490 (1.97e-11) | full: 55 s; appr.: <5 s (0 errors) |
| NETWORK 2 (100 nodes, 249 edges) | Perturbation graph | 98  | 9731| 280 | 151 | 0.779 (2.96e-39) | 0.333 (7.20e-14) | <5 s |
|        | Unweighted perturbation graph + NET-SYNTHESIS | 51  | 9752| 79  | 198 | 0.785 (7.70e-36) | 0.257 (1.55e-10) | <5 s |
|        | Perturbation graph + SOS pruning | 94  | 9396| 255 | 155 | 0.775 (3.19e-38) | 0.329 (1.54e-14) | <5 s |
|        | Perturbation graph + TRANSWESD | 86  | 9442| 209 | 163 | 0.773 (8.77e-38) | 0.327 (6.07e-14) | full: >5 h; appr.: <5 s (0 errors) |
| NETWORK 3 (100 nodes, 195 edges) | Perturbation graph | 85  | 9414| 291 | 110 | 0.844 (3.65e-51) | 0.309 (1.21e-74) | <5 s |
|        | Unweighted perturbation graph + NET-SYNTHESIS | 52  | 9726| 79  | 143 | 0.827 (1.08e-46) | 0.282 (1.24e-67) | <5 s |
|        | Perturbation graph + SOS pruning | 84  | 9447| 258 | 111 | 0.842 (8.68e-51) | 0.311 (4.24e-75) | <5 s |
|        | Perturbation graph + TRANSWESD | 82  | 9512| 193 | 113 | 0.844 (1.88e-34) | 0.490 (1.97e-116) | full: 23 min; appr.: <5 s (0 errors) |
| NETWORK 4 (100 nodes, 211 edges) | Perturbation graph | 105 | 9377| 312 | 106 | 0.835 (1.51e-41) | 0.374 (3.58e-88) | <5 s |
|        | Unweighted perturbation graph + NET-SYNTHESIS | 54  | 9592| 97  | 157 | 0.798 (2.84e-34) | 0.292 (5.72e-68) | <5 s |
|        | Perturbation graph + SOS pruning | 101 | 9422| 267 | 110 | 0.829 (2.52e-40) | 0.374 (3.79e-40) | <5 s |
|        | Perturbation graph + TRANSWESD | 98  | 9485| 204 | 113 | 0.827 (6.71e-40) | 0.400 (1.44e-94) | <5 s |
| NETWORK 5 (100 nodes, 193 edges) | Perturbation graph | 68  | 9236| 469 | 125 | 0.774 (1.11e-29) | 0.155 (1.78e-33) | <5 s |
|        | Unweighted perturbation graph + NET-SYNTHESIS | 32  | 9607| 100 | 161 | 0.747 (6.07e-25) | 0.143 (9.2e-30)  | <5 s |
|        | Perturbation graph + SOS pruning | 66  | 9298| 409 | 127 | 0.769 (8.95e-29) | 0.156 (1.14e-33) | <5 s |
|        | Perturbation graph + TRANSWESD | 58  | 9384| 323 | 135 | 0.758 (7.63e-27) | 0.159 (2.32e-34) | full: >5 h; appr.: <5 s (0 errors) |
| NETWORK 5 without noise (100 nodes, 193 edges) | Perturbation graph | 160 | 9231| 476 | 33  | 0.936 (4.31e-67) | 0.442 (6.82e-102) | <5 s |
|        | Unweighted perturbation graph + NET-SYNTHESIS | 83  | 9660| 47  | 110 | 0.910 (4.50e-60) | 0.456 (3.96e-105) | <5 s |
|        | Perturbation graph + SOS pruning | 136 | 9576| 131 | 57  | 0.923 (1.71e-63) | 0.534 (5.55e-124) | <5 s |
|        | Perturbation graph + TRANSWESD | 132 | 9605| 102 | 61  | 0.923 (2.06e-63) | 0.567 (9.11e-132) | full: >5 h; appr.: <5 s (0 errors) |

Shown are the reconstruction results for the (raw) perturbation graphs and for the pruned graphs obtained by applying NET-SYNTHESIS/TRANSWESD/SOS pruning to the perturbation graph. Running times (Intel Core2 Quad CPU Q6700; 2.67 GHz) are given for NET-SYNTHESIS and for full and approximate algorithm (appr.) TRANSWESD. Network 5 was additionally reconstructed with non-noisy simulation data.

different networks and may become extensive. In two networks (1 and 4), we were able to apply the exact algorithm to the perturbation graph in reasonable time. For the other networks, we interrupted the exact algorithm after 5 h and used the approximate variant (approximate shortest path computation and no recalculation of paths after edge removals), which in all networks needed <5 s. Checking the approximation results for errors [removed edges violating Condition (4)], we saw that the simplified algorithm introduced no errors in any of the five networks indicating that the simplified algorithm delivers reasonable approximations in realistic gene regulatory networks. However, when choosing α > 1 the number of errors may quickly increase, especially if the network contains positive cycles (data not shown).

We also computed the transitive reduction using the method of Albert et al. (2007) implemented in NET-SYNTHESIS taking as input the same perturbation graph as used for TRANSWESD (without weights). As the computed result is non-unique and very sensitive to edge ordering, we repeated the computation for 10 different (random) orderings and took the best result [in terms of AUPR value; for fair comparison, we sorted the (remaining) edges in the final graph also with respect to the edge weights taken from the perturbation graph]. The algorithm is much faster than full TRANSWESD (but comparable with the approximate version: <5 s in all networks) and removes much more edges resulting in significantly less FNs. However, this comes at the price of eliminating a relatively high number of TPs. The resulting AUPR values are constantly significantly lower than in networks obtained by TRANSWESD. It even turns out that all networks obtained by NET-SYNTHESIS have (partially considerably) lower AUROC and AUPR values than the perturbation graphs from which they were produced. These results indicate that the attenuated pruning strategy of TRANSWESD based on edge weights is highly advantageous for reconstructing regulatory networks. Probably for this reason, the SOS pruning strategy of Tresch et al. (2007)—which also operates on weighted graphs—yielded better results than NET-SYNTHESIS. However, the improvement in the AUPR value (if it increased the AUPR value of the perturbation graph at all) is significantly lower than for TRANSWESD except in the problematic case of the non-stationary Network 2, where SOS pruning performed slightly better but also led to a decrease of the AUPR value compared to the perturbation graph. Generally, SOS pruning deleted significantly less edges than TRANSWESD, which can probably be attributed to the very conservative metric (multiplication of probabilities) used by SOS pruning for quantifying path lengths.

As a proof of principle, with the results presented in Table 1, our algorithm was ranked on place 3 (out of 19 submissions) in the DREAM4 Insilico_Size_100 sub-challenge. This result is
encouraging, especially due to the fact that our method needs only a fraction of the simulation data that were provided.

4 DISCUSSION AND CONCLUSION

In this work we presented TRANSWESD, an elaborated variant of transitive reduction, which is applicable to an extended class of perturbation graphs, i.e. cyclic, signed and weighted digraphs. Major changes and improvements concern: (i) new statistical approaches for generating weighted and signed perturbation graphs; (ii) the use of edge weights (association strengths) for recognizing true redundant structures; (iii) causal interpretation of cycles; (iv) relaxed definition of transitive reduction; and (v) approximation algorithms for large networks. The success of transitive reduction depends to a large extent on the quality of the perturbation graph, and thus on the chosen threshold method and (indirectly) on type and quality of the available data. Whereas the quality is mainly governed by the signal to noise ratio, the type of data (e.g. gene expression, protein level, protein phosphorylation level, etc.) may have a profound effect on the observable perturbation effects.

We presented a modular procedure for generating perturbation graphs providing a basis of FP reduction methods. The main task of this procedure is to filter causally explainable (direct and indirect) effects from noise. Indirect effects may also be filtered during this process although this will be the main task of FP reduction methods such as TRANSWESD. Our workflow for generating the perturbation graph consists of three sequential modules: (i) planning and conducting perturbation experiments; (ii) generation of signed perturbation graph from experimental data; and (iii) assign edge weights (reflecting association strengths) from correlation measures. Each module might be exchanged or adapted, e.g. if other types of data are available. For example, certain interactions may not be deducible from single perturbations or/and steady-state data and may require special perturbation strategies in Module (i). Only multiple knockouts, for instance, will detect a positive influence of one node upon another if this influence is combined with others via Boolean OR-logic. It is straightforward to integrate information of single and multiple perturbations when deriving the perturbation graph in Module (ii). Furthermore, data of the transient response phase combined with suitable data analysis in Modules (ii) and (iii) could also be considered when generating the perturbation graph. Notice that depending on the specific perturbation data (transient, steady-state, time-courses) other, possibly non-linear correlation measures such as mutual information might be better suited to quantify strengths of associations (Daub et al., 2004), though linear measures appear to be appropriate if monotone dependencies (unique edge signs) can be assumed. Perhaps the most crucial step in generating the perturbation graph is to classify a perturbation effect as significant (an edge is introduced) or not. In contrast to the correlation measures, this classification is based on one single value. Therefore, experimental replicates would help much in providing a higher confidence level for the edges. Generally, the amount of data needed by our approach for generating the perturbation graph is considerable since all nodes must be perturbed separately and the respective responses in all other nodes have to be measured. However, the core procedure of TRANSWESD is independent of the method employed for deducing the perturbation graph, hence, for sparse datasets, other approaches could be used.

Benchmark tests demonstrated that our two-threshold strategy for generating the perturbation graph delivers networks that already have a comparably high reconstruction quality on its own. A similar observation was made by Yip et al. (2010): simple noise models filtering noise from relevant perturbation effects had higher accuracy than more elaborate differential equation models (though a combination of both could slightly improve the results). The authors did not address the removal of edges from indirect effects but they mentioned it as a potential means to improve the results. In fact, the benchmarks showed that our TRANSWESD algorithm can significantly enhance the reconstruction quality by carefully removing edges that are likely to be FP.

We have illustrated that transitive reduction of signed acyclic graphs is, algorithmically, rather straightforward. However, even in acyclic graphs, the use of edge weights may be highly beneficial as it helps to avoid eliminating true redundant structures. Pruning cyclic graphs raises several problems many of which are tackled in TRANSWESD by using edge weights. We further illustrated the question of interpreting causality that arises for negative cyclic structure and propose to use elementary paths as a solution to the cost of more computational time. By these features, TRANSWESD outperformed available transitive reduction algorithms in realistic and objective benchmarks. Another advantage of TRANSWESD is that—if all edge weights are distinct—a unique graph in terms of a partial order (defined on the edge weights) will be delivered. It would be interesting to compare TRANSWESD also with other pruning strategies, e.g. based on partial correlations (de la Fuente et al., 2004) or data processing inequalities (Margolin et al., 2006). The latter two were originally developed for undirected graphs but an adaptation to directed graphs could be possible.

In large networks with many cycles, TRANSWESD may become inefficient as it requires the computation of shortest signed paths, an NP-complete problem. We suggested two approaches for calculating approximate solutions either by approximate computation of shortest signed paths or/and by waiving the recalculation of shortest paths when removing edges. In the benchmarks, approximate solutions were identical to the exact solutions.

Similar as the method of Albert et al. (2007), TRANSWESD may easily account for prior knowledge by assigning a weight of 0 to known interactions. One may also introduce an upper boundary for the number of edges that a path may contain if this path is used to explain an edge. So far, TRANSWESD is restricted to edge removal. When integrating multiple sets of perturbation data, it might be beneficial to extend TRANSWESD to the general case of transitive reduction, namely to also allow edge insertion in a smart, data-driven manner. This needs further investigations.

In summary, our presented reconstruction workflow requires simple data and delivers edge candidates with relative high probability to exist. Edges are identified with weights, signs and directions providing additional crucial information for designing new experiments and for testing new hypotheses. TRANSWESD was presented as an essential component of this workflow but it may independently serve as a general FP reduction method in combination with other reverse engineering methods.
REFERENCES

Ahm, A.V. et al. (1973) The transitive reduction of a directed graph. SIAM J. Comput., 1, 131.
Akutsu, T. et al. (2000) Identification of genetic networks by Bayesian network selection. Bioinformatics, 16, 1429–1438.
Albert, R. et al. (2007) A novel method for signal transduction network inference from indirect experimental evidence. J. Comput. Biol., 14, 927–949.
Bennett, P. et al. (2009) Approximating transitive reductions for directed networks. Algorithms Data Struct., 5864, 74–85.
Draht, C.O. et al. (2004) Estimating mutual information using B-spline functions—a new similarity measure for analysing gene expression data. BMC Bioinformatics, 5, 118.
de la Fuente, A. et al. (2004) Discovery of meaningful associations in genomic data using partial correlation coefficients. Bioinformatics, 20, 3565–3574.
Drozdzynski, M. et al. (2008) Automatic reconstruction of molecular and genetic networks from discrete time series data. Bioinformatics, 23, 181–190.
Gardner, T.S. and Faith, J.J. (2005) Reverse-engineering transcriptional control networks. Phys. Life Rev., 2, 65–88.
Hausser, P. (1994) Shortest paths in signed graphs. Ann. Discrete Math., 59, 201–214.
Hecker, M. et al. (2009) Gene regulatory network inference: data integration in dynamic models—a review. Biosystems, 96, 86–103.
Kaschek, D. et al. (2009) NET-SYNTHESIS: a software for synthesis, inference and simplification of signal transduction networks. Bioinformatics, 24, 293–295.
Klamt, S. et al. (2007) Structural and functional analysis of cellular networks with CellNetAnalyzer. BMC Syst. Biol., 1, 2.
Klamt, S. and von Kamp, A. (2009) Computing paths and cycles in biological interaction graphs. BMC Systems Biol., 3, 57.
Lapin, R.L. and Papadimitriou, C.H. (1999) The even-path problem for graphs and digraphs. Networks, 14, 507–513.
Markushin, D. et al. (2009) Generating realistic in silico gene networks for performance assessment of reverse engineering methods. J. Comput. Biol., 16, 229–239.
Maginlin, A.A. et al. (2006) ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. BMC Bioinformatics, 7 (Suppl. 1), S7.
Markowetz, F. and Spang, R. (2007) Inferring cellular networks—a review. BMC Bioinformatics, 8 (Suppl. 6), S5.
Maurya, M.R. et al. (2003) A systematic framework for the development and analysis of signed digraphs for chemical processes. 1. Algorithms and analysis. Ind. Eng. Chem. Res., 42, 4789–4810.
Meyers, D.M. and Thompson, G.L. (1969) Finding a minimum equivalent graph of a digraph. J. Assoc. Comput. Mach., 16, 455–460.
Nelander, S. et al. (2008) Models from experiments: combinatorial drug perturbations of cancer cells. Mol. Syst. Biol., 4, 116.
Oberhardt, M.A. et al. (2009) Applications of genome-scale metabolic reconstructions. Mol. Syst. Biol., 5, 311.
Rice, J.F. et al. (2005) Reconstructing biological networks using conditional correlation analysis. Bioinformatics, 21, 765–773.
Saez-Rodriguez, J. et al. (2009) Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. Mol. Syst. Biol., 5, 311.
Shen-Orr, S.S. et al. (2002) Network motifs in the transcriptional regulatory network of Escherichia coli. Nat. Genet., 31, 64–68.
Stodolakay, G. et al. (2007) Dialogue on reverse-engineering assessment and methods: the DREAM of high-throughput pathway inference. Ann. N. Y. Acad. Sci., 1115, 1–22.
Stodolakay, G. et al. (2009) Lessons from the DREAM2 Challenges. Ann. N. Y. Acad. Sci., 1158, 159–191.
Tesch, A. et al. (2007) Discrimination of direct and indirect interactions in a network of regulatory effects. J. Comput. Biol., 14, 1217–1228.
Wagner, A. (2001) How to reconstruct a large genetic network from gene perturbations in fewer than n2 easy steps. Bioinformatics, 17, 1183–1197.
Yip, K.Y. et al. (2010) Improved reconstruction of in silico gene regulatory networks by integrating knockdown and perturbation data. PLoS One, 5, e8121.