BIOINFORMATICS ORIGINAL PAPER

Vol. 26 no. 17 2010, pages 2160–2168
doi:10.1093/bioinformatics/btq342

Systems biology

Advance Access publication July 6, 2010

TRANSWESD: inferring cellular networks with transitive reduction

Steffen Klamt[1,2,*], Robert J. Flassig[1] and Kai Sundmacher[1,3]

[1]Max Planck Institute for Dynamics of Complex Technical Systems, [2]MaCS—Magdeburg Centre for Systems Biology, Sandtorstrasse 1 and [3]Process Systems Engineering, Otto-von-Guericke-University, Universitätsplatz 2, D-39106 Magdeburg, Germany

Associate Editor: Olga Troyanskaya

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.

Received on April 8, 2010; revised on June 1, 2010; accepted on June 22, 2010

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 (Saez-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 $i \rightarrow j$, 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; Moyles 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 $i \rightarrow j$ for which a (simple) path starting in i and ending in j (not using $i \rightarrow j$) 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

*To whom correspondence should be addressed.

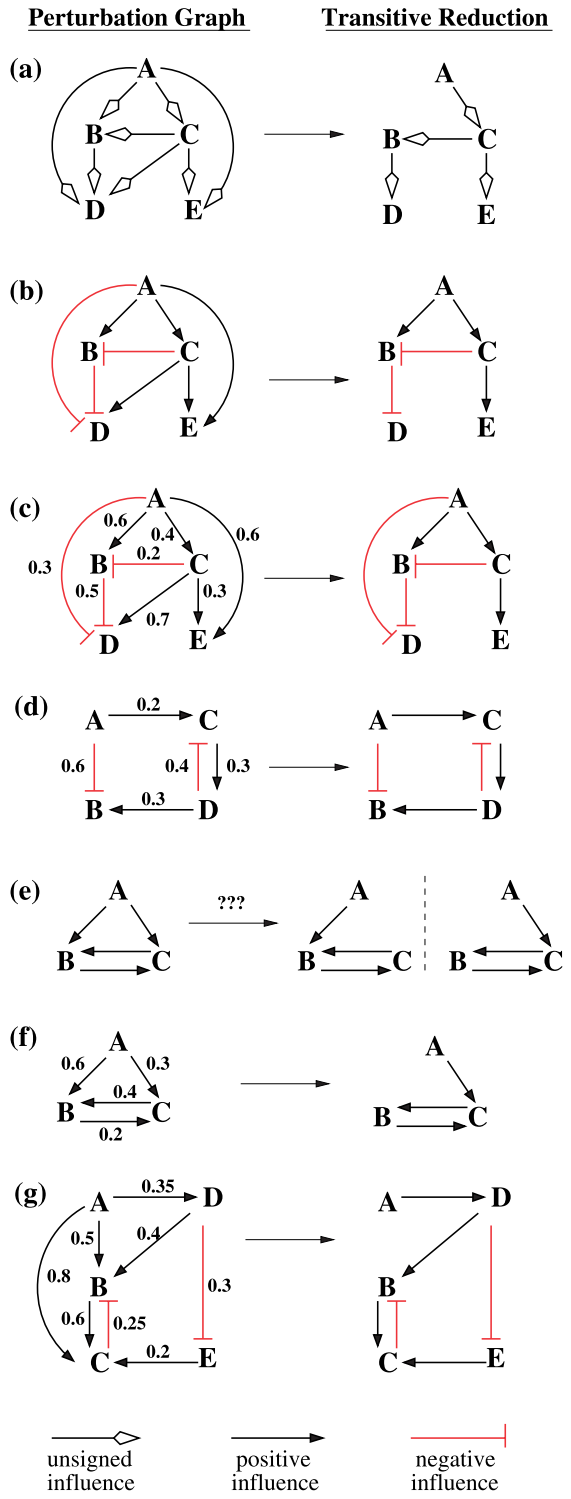

Fig. 1. Examples of perturbation graphs and their transitive reductions. See text for discussion and explanations.

can be taken into account by adding a sign label to each edge in the perturbation graph, which becomes then a signed directed graph (see Fig. 1b—a signed version of Fig. 1a). Transitive reduction can then be performed in a similar way: an edge $i \to j$ is deleted only if there is a path from i to j whose overall sign (product of the signs of the involved edges) corresponds to the sign of this edge. As can be seen in the example in Figure 1b, this may save edges that were mistakenly deleted in the unsigned version. A second drawback of the original approach of transitive reduction is the risk to remove true edges, even in signed perturbation graph. The radical pruning strategy of transitive reduction aims at minimizing false positive (FP) edges in the reconstructed network but it may lead to a high number of false negatives (FNs). This effect becomes visible in networks comprising many (coherent) feed-forward loops where a node may affect another node via direct (edge) and indirect (path) links of the same sign. Since feed-forward loops have been shown to occur frequently in gene regulatory networks (Shen-Orr *et al.*, 2002), this property can become a serious limitation of the method. A third shortcoming is the prerequisite that the perturbation graph is acyclic—a condition that is often not fulfilled in realistic biological networks. If the perturbation graph is cyclic, the solution of transitive reduction is, in general, not unique. As we will see, negative cycles in signed perturbation graphs may bring about even more complications for transitive reduction.

In this article, we will present TRANSWESD (TRANSitive Reduction in WEighted Signed Digraphs), a new variant of transitive reduction that seeks to overcome these problems. Generalizations of transitive reduction to signed and cyclic directed graphs have been proposed by other authors (Albert *et al.*, 2007; Tresch *et al.*, 2007). However, our approach combines and extends existing variants and differs in several key aspects (e.g. use of *weighted* perturbation graphs, treatment of negative cycles and handling of elementary versus non-elementary paths). We also discuss issues related to the identification of significant perturbation effects, a key step in generating the perturbation graph. 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.

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. $e \in E$ is an ordered pair $e=(u,v)$ of distinct nodes $u,v \in V$, also denoted by $u \to v$, where u is the start node and v the end node. A *signed digraph* $G=(V,E,\varphi)$ contains additionally a sign mapping $\varphi : E \to \{-,+\}$ 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 is denoted by $u \to_s v$. A *weighted signed digraph* $G=(V,E,\varphi,\gamma)$ contains additionally a weight mapping $\gamma : E \to \Re_{\geq 0}$ that assigns each edge a weight which we assume here to be non-negative. An edge (u,v) with sign s and weight w is denoted by $u \to_{s,w} v$.

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

edge e_i. In our terminology, a *path* is a walk with the additional condition that no node occurs twice, i.e. a path does not contain a cycle. The latter property is sometimes emphasized by calling a path 'simple' or 'elementary' and walks are sometimes called non-elementary paths. Finally, an (elementary) *cycle* is a closed walk with no repeated nodes except for the first and last node which coincide. Concrete paths or cycles are written as alternating sequences of nodes and arrows (edges), e.g. $u \to v \to w$, which gives a unique identifier for a path or cycle if no parallel edges exist between the involved nodes. A path with start node u and end node v is denoted by $u \Rightarrow v$ and may also consist of a single edge $u \to v$.

The *length* of a path/cycle is calculated from the weights of the involved edges, e.g. by summing them up (Σ-metric). We also need another variant, called MAX-metric, where the length of a path is the maximum weight of all its edges. The sign of a path/cycle is obtained by multiplying the edge signs (a signed digraph is, therefore, not equivalent to a weighted digraph with positive and negative edge weights). A path from u to v with overall sign s is denoted by $u \Rightarrow_s v$. If this path has length d then we write $u \Rightarrow_{s,d} v$.

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 $\mathbf{x}^0$ (x_i^0 denotes the wild-type state of the i-th node) and assume that the measurements for each species are normalized to the maximum value that has ever been observed for the respective species, i.e. $x_i^0 \in [0,1]$. 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. x_i^k is the state of the i-th node in experiment k. With $p^k \in \{-1,+1\}$ we denote whether the pertubation in k forced a decrease (-1) (e.g. by knockout or knockdown) or increase ($+1$) (e.g. by over-expression) of the amount/activity of node k.

Step 2 (Section 2.3): for each node, we compare the unperturbed state (x_i^0) to the measured states in the perturbation experiments (x_i^k). Using an appropriate threshold strategy, significant changes are identified and included as signed edges $k \to i$ in the resulting perturbation graph.

Step 3 (Section 2.4): each identified edge in the perturbation graph is endowed with a weight extracted from the measurements indicating the association/interaction strength between the two connected nodes.

Step 4 (Section 2.5): the final step is the computation of the transitive reduction using our novel TRANSWESD algorithm, which can handle weighted signed digraphs that may also contain cycles. (Note that, in principle, TRANSWESD may accept any perturbation graph, even if the way to generate the graph is different from Steps 1–3.)

2.3 Generating the perturbation graph: thresholding

To decide whether a perturbation of k induces a significant effect on node i (and is thus integrated as edge $k \to i$ in 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_k. 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 completely ignore the presence of noise and define the variation measure for node pair (k,i) as $\Delta_i^k := (x_i^k - x_i^0)p^k$ (p^k is the indicator of perturbation direction as described above). Initially, we may introduce edges

$$k \to_- i \quad \text{if} \Delta_i^k < 0 \quad \text{and} \quad k \to_+ i \quad \text{if} \Delta_i^k > 0 \qquad (1)$$

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 Δ_i^k, and thus edges $k \to i$ 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 $|\Delta_i^k|$. 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 ϑ 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 β_i accounts for the individual dynamic nature of each node and is derived from the variance of node's i entire perturbation profile excluding the perturbation of node i itself.

We thus introduce an edge from node k to i in the perturbation graph only if the two conditions, (i) $|\Delta_i^k| > \vartheta$ and (ii) $|\Delta_i^k| > \beta_i$, are met. We compute $\beta_i := \Gamma\sigma_i$ with variance scaling factor Γ and SD σ_i of gene x_i. Notice that depending on the chosen values for ϑ and Γ and the fluctuations of node x_i, we either have $\vartheta \geq \beta_i$ or $\vartheta < \beta_i$. 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 ϑ and Γ 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 ($\rho_{u,v}$) from the u-th and v-th element of the measured state vectors $\boldsymbol{x}^0, \boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{v-1}, \boldsymbol{x}^{v+1}, \ldots, \boldsymbol{x}^n$ (cf. Rice *et al*., 2005). $\rho_{u,v}$ is computed as linear correlation coefficient where we exclude the data from the v-th experiment because we want $\rho_{u,v}$ to quantify the dependency of v on u and the external perturbation in v cannot be explained by u. Accordingly, $\rho_{u,v}$ is not symmetric.

For each edge $u \to_s v$ captured in the perturbation graph P derived in Section 2.3, we assign its weight to be $1 - |\rho_{u,v}|$, 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).

It sometimes happens that $s \neq \mathrm{sgn}(\rho_{u,v})$ indicating that the response of v upon perturbing u does not reflect the sign derived from the correlation analysis. It appeared nevertheless useful to keep this edge but with high weight (close to the maximal possible weight 1) indicating a weak association. We also tested other weights, e.g. the change of v when perturbing u, $|x_v^u - x_v^0|$, but it turned out that the algorithm performs better with $\rho_{u,v}$, simply because it evaluates many (n) experiments. On the other hand, for deciding whether an edge $u \to_s v$ exists at all in P (and for fixing its sign), the effect a perturbation in u induces in node v appeared to be better suited than conditional correlation (Section 2.3).

2.5 Transitive reduction with TRANSWESD

At this stage, we have obtained a weighted, signed and directed perturbation graph $P=(V,E,\varphi,\gamma)$, 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 negatives (TNs)—but with the risk to remove true positive (TP) predictions resulting in FNs. Starting with simple and ending with cyclic perturbation graphs, we generalize the idea of transitive reduction step by step and explain our extensions in TRANSWESD that seek to minimize shortcomings of previous variants.

2.5.1 Transitive reduction in signed acyclic graphs Wagner (2001) used transitive reduction to prune unsigned acyclic perturbation graphs. It is straightforward to generalize this procedure to signed acyclic perturbation graphs P (at this point we neglect the weights). The basic idea is to check for each edge $u \to_s v$ in P whether there is an elementary path $u \Rightarrow_s v$ not involving this edge, which can then be seen as an explanation for the observed influence $u \to_s v$ allowing one to remove this edge. For this purpose, in a first step we compute for each pair of nodes (u,v) the shortest positive and shortest negative path telling us whether a positive and/or negative path from u to v exists at all. As we are only interested in the existence of paths we may use arbitrary edge weights, e.g. setting all to one, and arbitrary metric. We may employ the double label algorithm, a generalized version of the Dijkstra algorithm for computing shortest positive/negative paths in Σ-metric. 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 exits.

In a second step, we prune P to the minimal graph P_{TR} (minimal with respect to number of edges) satisfying

$$\begin{aligned} S^+_{TR}(u,v) < inf \text{ for all removed positive edges } u \to_+ v \text{ in } P \text{ and} \\ S^-_{TR}(u,v) < inf \text{ for all removed negative edges } u \to_- v \text{ in } P \end{aligned} \quad (2)$$

In acyclic signed graphs, the unique solution can easily be found with the help of S^+ and S^-: We check for each edge $u \to_s v$ whether we can find a successor $z \neq v$ of u such that an edge $u \to_q z$ and a path $z \Rightarrow_t v$ exist fulfilling the sign condition $q \cdot t = s$ (this path exists if $S^t(z,v) < inf$). If so, we can conclude that the influence $u \to_s v$ can be explained by the augmented path $u \to_q z \Rightarrow_t v$, which is ensured to be elementary as we have an acyclic graph. We, therefore, remove $u \to_s v$ 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 edge $u \to_s v$: in all paths using this edge, we can replace the latter by $u \to_q z \Rightarrow_t v$ because, again, in acyclic graphs it is ensured that the resulting path is still elementary and thus a valid explanation. Eliminating all removable edges, we obtain the unique minimal equivalent graph P_{TR} which produces the same perturbation effects as the original graph P. Transitive reduction in unsigned graph uses the same algorithm but neglects the sign condition.

Our definition of transitive reduction differs in some aspects from the version used in Albert *et al*. (2007). First, only elementary paths (not involving cycles) are considered as possible explanations for edges. Second, instead of Condition (2) Albert *et al*. follow the original (stronger) definition of transitive reduction, namely that

$$\begin{aligned} S^+_{TR}(u,v) < inf \text{ wherever } S^+(u,v) < inf \text{ and} \\ S^-_{TR}(u,v) < inf \text{ wherever } S^-(u,v) < inf \end{aligned} \quad (3)$$

We argue that Condition (3) can be relaxed to (2), since in our application of transitive reduction there is no necessity to preserve a path $u \Rightarrow_t v$ between two nodes u and v if no edge $u \to_t v$ (i.e. neither a direct nor an indirect effect of u on v) could be deduced from the experiments. However, as long as we consider acyclic graphs both definitions will nevertheless lead to the same result because then (3) follows from (2).

The example in Figure 1b shows that accounting for the edge signs avoids removing edges that cannot be explained: in contrast to Figure 1a (unsigned perturbation graph) the edge A$\to_-$B is kept because the path A$\to_+$C$\to_+$B cannot explain the negative sign of this edge.

2.5.2 Transitive reduction in signed and weighted acyclic graphs As explained in Section 1, 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 P_{TR} should be minimal and satisfy

$$\begin{aligned} S^+_{TR}(u,v) < \alpha w \text{ for all removed positive edges } u \to_{+,w} v \text{ in } P \text{ and} \\ S^-_{TR}(u,v) < \alpha w \text{ for all removed negative edges } u \to_{-,w} v \text{ in } P, \end{aligned} \quad (4)$$

with positive confidence factor α discussed below. For this purpose, we now consider explicitly the edge weights based on conditional correlation as derived in Section 2.4. As in the previous section, we compute the shortest path lengths S^+ and S^- in P. For quantifying the overall weight (length) of a path, we use MAX-metric, i.e. an influence path is as good as its 'weakest' edge having the largest weight and thus the lowest association. In acyclic graphs, we can again use the double label algorithm adapted for MAX-metric.

In order to fulfill (4), the transitive reduction step has to be modified as follows: we remove an edge $u \to_{s,w} v$ if we can find a successor $z \neq v$ of u such that an edge $u \to_{q,c} z$ and a path $z \Rightarrow_{t,d} v$ exist fulfilling the sign condition $q \cdot t = s$ and now additionally the weight condition $\max(c,d) < \alpha \cdot w$. The positive factor α controls the overall association strength a path must have in order to explain a given edge. Normally, one will choose a value close to one (we use 0.95) but one may also prefer smaller values, demanding significantly larger associations in all edges of a path to explain an edge. In the extreme case $\alpha = 0$, we have $P_{TR} = P$. If $\alpha > 1$ one would allow edges in a path to have lower associations than of the edge the path explains. With $\alpha = inf$ Condition (4) coincides with Condition (2) and we were thus back at transitive reduction in unweighted graphs. Again, having an acyclic graph ensures first that the augmented path $u \to_{q,c} z \Rightarrow_{t,d} v$ yielding $u \Rightarrow_{s,\max(c,d)} v$ is elementary, i.e. $z \Rightarrow_{t,d} v$ does not contain edge $u \to_{q,c} z$, and is thus a valid explanation for $u \to_{s,w} v$ 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$\to_{-,0.3}$D is retained because the path A$\to_{-,0.6}$B$\to_{+,0.5}$D has length 0.6 and is thus not a valid explanation when choosing $\alpha < 1$ (but it would be with $\alpha > 2$).

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 $u \to v$ was removed only if two consecutive edges $u \to z \to v$ explain it.

2.5.3 Transitive reduction in signed and weighted cyclic graphs We now discuss the most general case where the perturbation graph may contain

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 (Lapaugh 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 exact 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 $C\rightarrow_{+,0.3}D\rightarrow_{-,0.4}C$. The key question is whether we consider the negative non-elementary path (walk) $A\rightarrow_{+,0.2}C\rightarrow_{+,0.3}D\rightarrow_{-,0.4}C\rightarrow_{+,0.3}D\rightarrow_{+,0.3}B$ as a valid explanation for the negative influence $A\rightarrow_{-,0.6}B$ we observe when perturbing A. With $\alpha<1$, sign and length of this walk would actually allow that. Tresch *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; Tresch *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 activation level of B. From system theory (Maurya *et al*., 2003), one can prove that the graph without this edge cannot show a decrease in B upon constitutive over-expression of A if we measure the *initial response* or the *steady-state response* in B. The initial response in a network is governed by the sign of the elementary paths and since removal of edge $A\rightarrow_{-,0.6}B$ would imply that only a positive elementary path from A to B remains the initial response would be positive in B (simply speaking, the effect of the positive path cannot be overtaken by the effect of the negative feedback induced by this path when looking at the initial response in B). Also in steady state, B cannot exhibit a decreased activity (compared to unperturbed wild-type) if the negative edge from A to B is removed. If only positive elementary paths from A to B exist, a negative feedback can induce an opposite effect in steady state only in conjunction with other structural requirements including positive feedbacks (Maurya *et al*., 2003). Albeit a negative effect in B might be observed transiently, we generally consider non-elementary paths containing a negative cycle as not sufficient for explaining an edge; only elementary paths with appropriate sign and weight are accepted. The negative edge from A to B is, therefore, kept in Figure 1d.

A third problem that may arise in cyclic graphs is non-uniqueness. An advantage of our approach is that edge weights eliminate many possible sources of non-uniqueness, in particular those related to positive cycles. Figure 1e depicts an unweighted perturbation graph containing a positive cycle. The positive edge from A to B could be explained by the positive path $A\rightarrow_{+}C\rightarrow_{+}B$. On the other hand, the positive edge from A to C could be explained by the positive path $A\rightarrow_{+}B\rightarrow_{+}C$. Methods based on unweighted perturbation graphs as in Albert *et al*. (2007) will thus remove one of both edges and keep the other. The choice depends on the edge processing order. With additional information on association strengths (edge weights) a unique solution can often be found with $\alpha<1$ as shown in Figure 1f: we would remove the edge from A to B as it can be explained by the positive path from A to C via B whose overall length (in MAX-metric) is shorter than that of the edge whereas the edge from A to C would be kept.

However, even with edge weights non-uniqueness may occur as illustrated in Figure 1g. In a first step, we may remove edge $A\rightarrow_{+,0.8}C$ (with $\alpha=0.95$ explainable by path $A\rightarrow_{+,0.5}B\rightarrow_{+,0.6}C$ or, alternatively, by $A\rightarrow_{+,0.35}D\rightarrow_{+,0.4}B\rightarrow_{+,0.6}C$). In a second step, we may either remove edge $A\rightarrow_{+,0.5}B$ (explainable by $A\rightarrow_{+,0.35}D\rightarrow_{+,0.4}B$) or edge $D\rightarrow_{+,0.4}B$ (explainable by $D\rightarrow_{-,0.3}E\rightarrow_{+,0.2}C\rightarrow_{-,0.25}B$). We can only remove one of both and then have to stop pruning because otherwise no explanation for the removed edge $A\rightarrow_{+,0.8}C$ 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 $u\rightarrow_{s,w}v$ is one where we can find a successor $z\neq v$ of u such that an edge $u\rightarrow_{q,c}z$ and a path $z\Rightarrow_{t,d}v$ exist fulfilling the sign condition $q\cdot t=s$ and the weight condition $\max(c,d)<\alpha\cdot w$. In contrast to acyclic networks, it may happen that the augmented path $u\rightarrow_{q,c}z\Rightarrow_{t,d}v$ is not elementary because the path $z\Rightarrow_{t,d}v$ 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 $A\rightarrow_{+,0.8}C$ and then $A\rightarrow_{+,0.5}B$ and we have to keep $D\rightarrow_{+,0.4}B$. This example illustrates also a fourth issue that we have to account for in cyclic perturbation graphs: in principle, $D\rightarrow_{+,0.4}B$ could be explained by $D\rightarrow_{-,0.3}E\rightarrow_{+,0.2}C\rightarrow_{-,0.25}B$. However, as mentioned before, removal is not allowed because then influence $A\rightarrow_{+,0.8}C$ 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 $A\rightarrow_{+,0.8}C$ is not explainable anymore by the path $A\rightarrow_{+,0.35}D\rightarrow_{+,0.4}B\rightarrow_{+,0.6}C$ if we removed edge $D\rightarrow_{+,0.4}B$. The explanation of the latter, $D\rightarrow_{-,0.3}E\rightarrow_{+,0.2}C\rightarrow_{-,0.25}B$, cannot be embedded in the path $A\rightarrow_{+,0.35}D\rightarrow_{+,0.4}B\rightarrow_{+,0.6}C$ 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_I^+ and S_I^- 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 recalculation 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 $\alpha<1$ 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 $\alpha=inf$. 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* (Klamt *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 $z \to v$ is removed if there is a (elementary or non-elementary) path $z \Rightarrow v$ of the same sign whose overall probability is larger than the probability of the edge $z \to v$. Actually, SOS pruning distinguishes two probabilities, one for having a positive edge (p_+) and one for having a negative edge (p_-) and the sign s 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 100 000 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. $\vartheta = 0.005$ 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 (pAUPR) 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 varies 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

DREAM4-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 (6.23e-111)	<5 s
Unweighted perturbation graph + NET-SYNTHESIS	67	9650	74	109	0.856 (1.98e-32)	0.394 (7.96e-93)	<5 s
Perturbation graph + SOS pruning	97	9524	200	79	0.869 (2.78e-34)	0.465 (2.27e-110)	<5 s
Perturbation graph + TRANSWESD	97	9562	162	79	0.870 (1.88e-34)	0.490 (1.97e-116)	full: 55 s; appr.: <5 s (0 errors)
NETWORK 2 (100 nodes, 249 edges)							
Perturbation graph	98	9371	280	151	0.779 (2.96e-39)	0.333 (7.20e-143)	<5 s
Unweighted perturbation graph + NET-SYNTHESIS	51	9572	79	198	0.765 (7.70e-36)	0.257 (1.55e-103)	<5 s
Perturbation graph + SOS pruning	94	9396	255	155	0.775 (3.19e-38)	0.329 (1.54 e-141)	<5 s
Perturbation graph + TRANSWESD	86	9442	209	163	0.773 (8.77e-38)	0.327 (6.07e-140)	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 (8.8e-51)	0.311 (4..24e-75)	<5 s
Perturbation graph + TRANSWESD	82	9512	193	113	0.844 (2.84e-51)	0.326 (7.38e-79)	full: >5 h; 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)	full: 23 min; appr: <5 s (0 errors)
NETWORK 5 (100 nodes, 193 edges)							
Perturbation graph	68	9238	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 (1.92e-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 (4.13e-67)	0.442 (6.82e-102)	<5 s
Unweighted perturbation graph + NET-SYNTHESIS	83	9660	47	110	0.910 (4.50e-60)	0.456 (3.09e-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 $\alpha > 1$ 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.

ACKNOWLEDGEMENTS

We thank Regina Samaga and Axel von Kamp for commenting on the manuscript and the DREAM4 organizers for creating the *in silico* challenge and providing data and gold standards.

Funding: German Federal Ministry of Education and Research [HepatoSys, Virtual Liver; FORSYS-Centre MaCS (Magdeburg Centre for Systems Biology)]; Ministry of Education and Research of Saxony-Anhalt (Research Center 'Dynamical Systems in Process Engineering and Biomedicine').

Conflict of Interest: none declared.

REFERENCES

Aho,A.V. *et al.* (1972) The transitive reduction of a directed graph. *SIAM J. Comput.*, **1**, 131.

Akutsu,T. *et al.* (2003) Identification of genetic networks by strategic gene disruptions and gene overexpressions under a Boolean model. *Theor. Comput. Sci.*, **298**, 235–251.

Albert,R. *et al.* (2007) A novel method for signal transduction network inference from indirect experimental evidence. *J. Comput. Biol.*, **14**, 927–949.

Berman,P. *et al.* (2009) Approximating transitive reductions for directed networks. *Algorithms Data Struct.*, **5664**, 74–85.

Daub,C.O. *et al.* (2004) Estimating mutual information using B-spline functions—an improved 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.

Durzinsky,M. *et al.* (2008) Automatic reconstruction of molecular and genetic networks from discrete time series data. *Biosystems*, **93**, 181–190.

Gardner,T.S. and Faith,J.J. (2005) Reverse-engineering transcriptional control networks. *Phys. Life Rev.*, **2**, 65–88.

Hansen,P. (1984) Shortest paths in signed graphs. *Ann. Discrete Math.*, **19**, 201–214.

Hecker,M. *et al.* (2009) Gene regulatory network inference: data integration in dynamic models—a review. *Biosystems*, **96**, 86–103.

Kachalo,S. *et al.* (2008) 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 Bioinformatics*, **10**, 181.

Lapaugh,A.S. and Papadimitriou,C.H. (1984) The even-path problem for graphs and digraphs. *Networks*, **14**, 507–513.

Marbach,D. *et al.* (2009) Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *J. Comput. Biol.*, **16**, 229–239.

Margolin,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.

Moyles,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**, 216.

Oberhardt,M.A. *et al.* (2009) Applications of genome-scale metabolic reconstructions. *Mol. Syst. Biol.*, **5**, 320.

Rice,J.J. *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**, 331.

Shen-Orr,S.S. *et al.* (2002) Network motifs in the transcriptional regulation network of Escherichia coli. *Nat. Genet.*, **31**, 64–68.

Stolovitzky,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.

Stolovitzky,G. *et al.* (2009) Lessons from the DREAM2 Challenges. *Ann. N. Y. Acad. Sci.*, **1158**, 159–195.

Tresch,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 *n* gene perturbations in fewer than $n(2)$ easy steps. *Bioinformatics*, **17**, 1183–1197.

Yip,K.Y. *et al.* (2010) Improved reconstruction of in silico gene regulatory networks by integrating knockout and perturbation data. *PLoS One*, **5**, e8121.