PREMER: parallel reverse engineering of biological networks with information theory

Alejandro F. Villaverde[1,2,4], Kolja Becker[3], and Julio R. Banga[4]

[1] Department of Systems & Control Engineering, University of Vigo, Galiza, Spain
[2] Centre for Biological Engineering, University of Minho, Braga, Portugal
[3] Modelling of Biological Networks, Institute of Molecular Biology gGmbH, Mainz, Germany
[4] Bioprocess Engineering Group, Instituto de Investigacións Mariñas (IIM-CSIC), Vigo, Galiza, Spain
afvillaverde@iim.csic.es

Abstract. A common approach for reverse engineering biological networks from data is to deduce the existence of interactions among nodes from information theoretic measures. Estimating these quantities in a multidimensional space is computationally demanding for large datasets. This hampers the application of elaborate algorithms – which are crucial for discarding spurious interactions and determining causal relationships – to large-scale network inference problems. To alleviate this issue we have developed PREMER, a software tool which can automatically run in parallel and sequential environments, thanks to its implementation of OpenMP directives. It recovers network topology and estimates the strength and causality of interactions using information theoretic criteria, and allowing the incorporation of prior knowledge. A preprocessing module takes care of imputing missing data and correcting outliers if needed. PREMER (`https://sites.google.com/site/premertoolbox/`) runs on Windows, Linux and OSX, it is implemented in Matlab/Octave and Fortran 90, and it does not require any commercial software.

Keywords: Network inference, Information theory, Parallel computing

1 Introduction

Many biological systems can be meaningfully represented as networks, that is, as a set of nodes (variables) connected by links (interactions). In the context of cellular networks the nodes are molecular entities such as genes, transcription factors, proteins, metabolites, and so on [7]. The network inference problem consists of learning the interconnection structure of the nodes, using as data the values of the variables (e.g. their expression levels or concentrations) at different situations and/or time instants. The concept of mutual information [12] can be used as a statistical measure for estimating the strength of the (possibly nonlinear) relations among nodes from a dataset. Indirect interactions, which take place when an entity A exerts an influence in C by means of an intermediate

entity B (i.e. $A \to B \to C$), are difficult to detect, because a spurious interaction may be deduced (not only $A \to B$ and $B \to C$, but also $A \to C$). The difficulty of discriminating between them increases when dealing with higher-order interactions, involving four or more entities. Although a few methods can cope with this issue, their application to large-scale problems is computationally costly [6], especially when dealing with time-series data. One such method, MIDER [13], is a general purpose network inference tool which takes into account time delays. It distinguishes between direct and indirect interactions using entropy reduction [9] and assigns directionality to the predicted links using transfer entropy [11]. It is implemented in Matlab, a widely used programming environment which nevertheless has some drawbacks, mainly (i) the need of buying commercial licenses, and (ii) low computational efficiency compared to other languages.

Here we present PREMER (Parallel Reverse Engineering with Mutual information & Entropy Reduction), a tool that overcomes these issues. It includes an advanced Fortran 90 implementation of the MIDER procedures, which allows for faster computations than Matlab. Additionally, the use of OpenMP directives enable it to run seamlessly in parallel environments, thus allowing for further speedups in performance. Results obtained on different datasets show that PREMER can be orders of magnitude faster than MIDER. Additionally, PREMER's Matlab code is fully compatible with the free Octave environment. Furthermore, PREMER offers two important additional capabilities. One is the ability to take prior knowledge into account, allowing to specify if a particular interaction is known to be non-existent. This is of particular importance in applications such as gene regulatory network (GRN) inference, where only a subset of the genes — the transcription factors, TFs — can regulate other genes. The second one is the ability to handle datasets with missing values and/or outliers, using statistical techniques to impute new values which are coherent with the latent structure of the data. PREMER's work-flow is depicted in Figure 2. More details about the methodology are given in the supplementary information (user's manual).

2 Implementation and availability

PREMER is provided as a set of Matlab/Octave scripts and an executable file which carries out the core computations. It has a number of options, which can be tuned by editing the main file, `runPremer`. Executable files are provided for Windows, Linux and OSX, and also as source code in Fortran (F90), which can be compiled to run on any operating system. The executable can also be invoked from the command line, thus avoiding the need for Matlab/Octave. A key feature of PREMER is its ability to run sequentially or in parallel. Parallelization has been implemented using OpenMP directives [3] and is entirely transparent to the user, who only needs to specify the number of threads in the main file. Mutual information and multidimensional entropies are estimated using an adaptive partitioning algorithm inspired in [2]. The PREMER toolbox is released under the free and open source GNU GPLv3. It is available at `https://sites.google.com/site/premertoolbox/`. Its use does not require any commercial software.

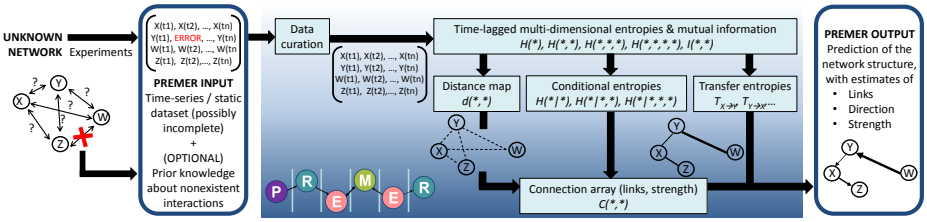

Fig. 1. Work-flow of the PREMER algorithm. First, a data curation module imputes missing data [4] and detects and corrects outliers, thus allowing the use of faulty datasets. Then PREMER calculates the distance between every possible pair of variables $d(X, Y)$ for several time delays. To this end it estimates the entropies of all variables $H(*)$, as well as the joint entropies $H(*, *)$ and the mutual information $I(*, *)$ of all pairs of variables. The user can choose to estimate also the multi-dimensional joint entropies of 3 and 4 variables ($H(*, *, *)$, $H(*, *, *, *)$), in order to use them in the subsequent entropy reduction step. The aim of this step is to determine whether all the variation in a variable Y can be explained by the variation in another variable X or, more generally, in a set of variables $\mathbf{X}$ [9]. By iterating through cycles of adding a variable X that reduces $H(Y|X, \mathbf{X})$ until no further reductions are obtained, the entropy reduction step yields the complete set of variables that control the variation in Y. Finally, directions are assigned to the links using transfer entropy, $T_{X\to Y}$, a non-symmetric measure of causality [11] calculated from time-lagged conditional entropies.

3 Selected Experimental Results

We tested PREMER on the same set of seven benchmark problems that was used for assessing the performance of MIDER. It has been shown elsewhere [13] that MIDER performs well compared with other state-of-the-art methods in terms of precision and recall of the inferred networks. We found that PREMER predicts the same networks as MIDER (in examples without missing data or prior information) achieving large reductions in computation times, as shown in Fig. 2. Panel A plots the accelerations obtained with PREMER's sequential implementation (i.e. using only one processor) with respect to MIDER. The most computationally costly problems give rise to the largest speed-ups: for example, for benchmark B7 with 3 entropy reduction rounds the computation time decreases from 42 hours to roughly 1.5 hours. This improvement is obtained using a single processor; additional speed-ups can be achieved in a parallel environment, as shown in panel B. The combined effect of code acceleration and parallel speed-up results in very significant reductions in computation time. For example, using a current 12-core desktop PC (hardware detailed in the caption of Fig. 2), PREMER runs up to 170 times faster than MIDER.

By going through several entropy reduction rounds it is possible to discover additional links, but in this process errors may appear: since the accuracy of every network inference method is limited by the information content of the data, some of the extra links can in fact be false positives. Therefore in many

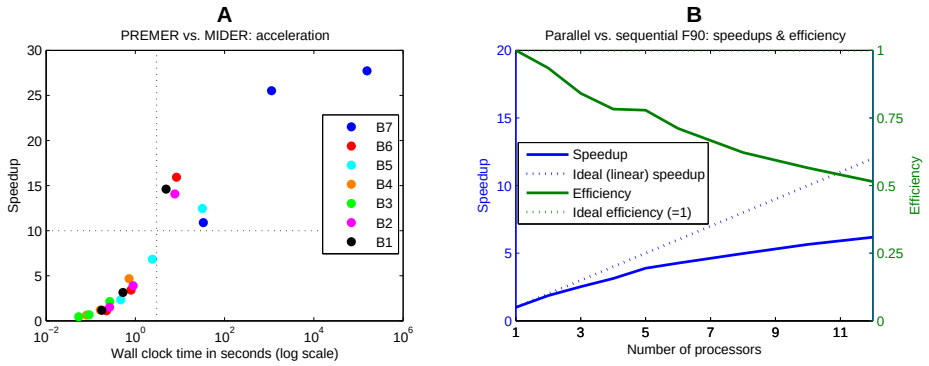

Fig. 2. [A]: Accelerations achieved by PREMER w.r.t. MIDER, for benchmarks B1–B7 of [13]. For every benchmark three points are plotted, depending on the number of entropy reduction rounds performed: 1, 2, or 3. **[B]:** Speed-up and efficiency of the parallel vs sequential versions of PREMER (benchmark B7, 3 entropy reduction rounds). Results obtained in a multi-core PC running Windows 7 64-bit with 16 GB RAM and 12 cores, 2 processors/core, Intel Xeon 2.30 GHz.

cases there is a trade-off between precision and recall: increasing the number of entropy reduction rounds leads to increased recall and decreased precision, and vice versa. Table 1 shows this trade-off for the average of the seven benchmark problems considered in [13].

Table 1. Trade-offs between precision and recall for different numbers of entropy reduction rounds. The values shown are the averages of the seven benchmark problems (B1–B7) considered in [13].

Entropy reduction rounds	1	2	3
Average precision (B1–B7)	0.7676	0.6958	0.6311
Average recall (B1–B7)	0.5267	0.5676	0.5819

Finally, we illustrate the performance improvement that can be obtained by taking prior knowledge into account. With this aim we create a benchmark network with GeneNetWeaver [10] consisting of 18 genes, out of which only 11 are considered transcription factors, and we generate time course data of the expression of each gene at 24 different time points. We evaluate the performance of PREMER using two different modes of including information (removing interactions *a priori* or *a posteriori*) and we compare it to other network inference methods such as ARACNE, CLR, MRNET, MRNET Backward (MRNETB) (available in the R package MINET [8]), Inferelator [1], and GENIE3 [5]. We report the corresponding values of AUROC (Area Under Receiver Operating Characteristic) and AUPR (Area Under the Precision-Recall curve) of each method

in Fig. 3.e–f. In panel (e) the set of regulators is assumed to be unknown (no prior knowledge), and consequently none of the interactions can be excluded. Among all the methods, PREMER achieves the highest score in both AUROC and AUPR. Panel (f) shows that the performance of all methods increases when prior knowledge is taken into account. Since the methods in the MINET package do not allow for excluding interactions *a priori*, they are evaluated by removing these interactions *a posteriori*. As for PREMER, in order to show the difference between both options we test both PREMER (post) and PREMER (prior), in which the interactions known to be non-existent are removed respectively *a posteriori* and *a priori*. It can be seen that excluding interactions *a posteriori* already increases the performance of all methods. However, excluding interactions *a priori* results in a further improvement of PREMER, as shown by the fact that PREMER (prior) is outperforming PREMER (post), where interactions are removed *a posteriori*, both in terms of AUROC and AUPR (Fig. 3.f).

Therefore, we conclude that (i) removing interactions based on prior knowledge is a way of increasing the performance of network inference methods, and that (ii) the improvement is bigger if this information is incorporated before network inference (*a priori*) instead of as a post-processing step (*a posteriori*). This is the solution adopted in PREMER.

4 Conclusions

PREMER is an open-source, multi-platform network inference tool based on information theory. It predicts the existence of network links, estimates their relative strength and direction, and provides a visual representation of the inferred system. It can take prior knowledge about the non-existence of specific interactions into account, which improves the quality of the network reconstructions. It also features a data preprocessing step which enables the use of datasets with missing values and/or outliers. PREMER is freely available as a Matlab/Octave toolbox. Core computations are performed in F90, achieving large speed-ups which can be increased further if working on a parallel environment. PREMER is geared towards ease of use, requiring minimum input from the user.

Acknowledgements

AFV acknowledges funding from the Galician government (Xunta de Galiza) through the I2C fellowship ED481B2014/133-0. KB was supported by the German Federal Ministry of Research and Education (BMBF, OncoPath consortium). JRB acknowledges funding from the Spanish government (MINECO) and the European Regional Development Fund (ERDF) through the project "SYNBIOFACTORY" (grant number DPI2014-55276-C5-2-R). This project has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 686282 (CanPathPro). We thank David R. Penas and David Henriques for assistance with the implementation.

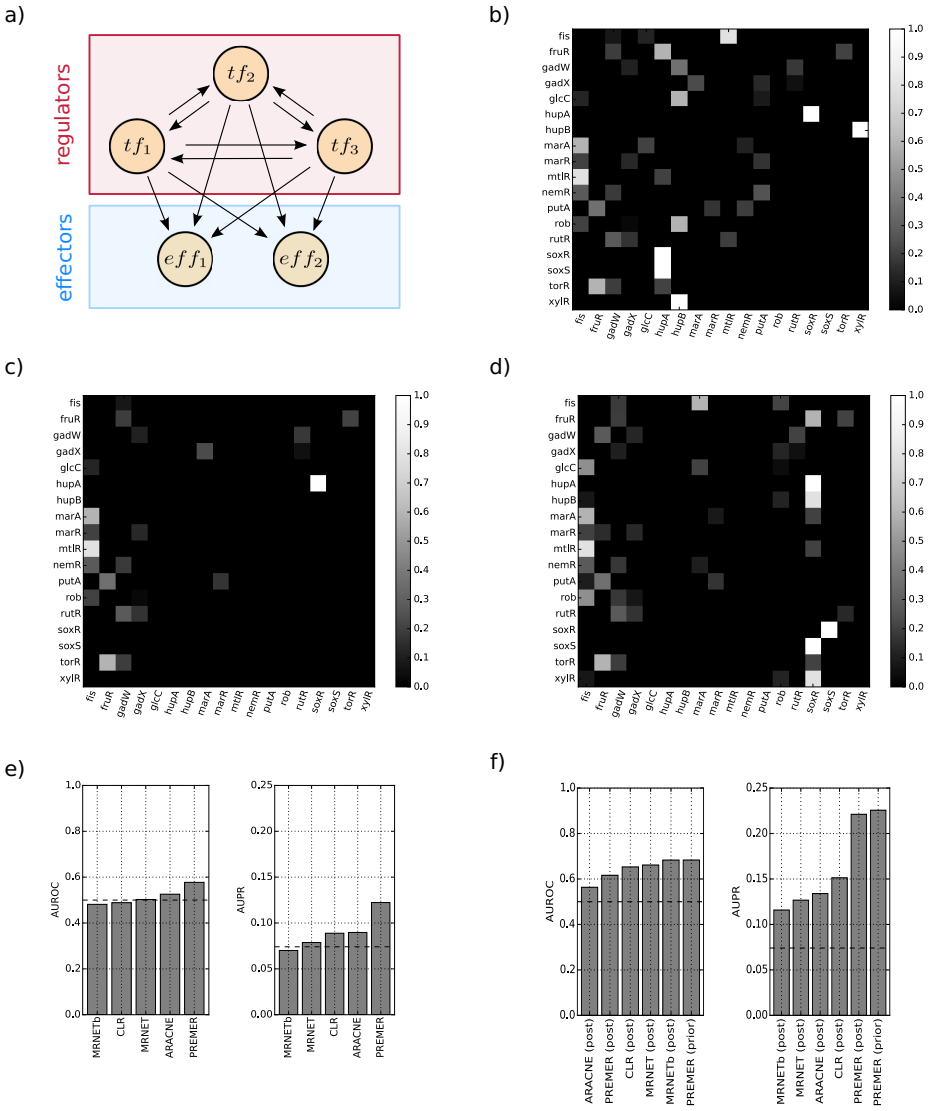

Fig. 3. Incorporating prior knowledge into network inference algorithms: **a)** Reasoning behind excluding interactions from gene regulatory networks. Only transcription factors effectively serve as regulators in the network, hence interactions from the effector genes to transcription factors or to other effectors can be excluded. **b)** Regulatory interaction matrix returned by PREMER without incorporating prior knowledge. The heat map scale represents the strength of the predicted interaction (0 = no interaction, 1 = strongly predicted interaction). **c)** Regulatory interaction matrix returned by PREMER incorporating prior knowledge by removing excluded interactions *a posteriori*. **d)** Regulatory interaction matrix returned by PREMER incorporating prior knowledge by removing excluded interactions *a priori*. **e)** Comparison of several network inference methods without incorporating prior knowledge. **f)** Comparison of network inference methods incorporating prior knowledge either *a posteriori* (post) or *a priori* (prior). In (e) and (f) horizontal dashed lines indicate the theoretical performance of a random classifier.

References

1. R. Bonneau, D. J. Reiss, P. Shannon, M. Facciotti, L. Hood, N. S. Baliga, and V. Thorsson. The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome Biol*, 7(5):R36, 2006.
2. C. Cellucci, A. Albano, and P. Rapp. Statistical validation of mutual information calculations: comparison of alternative numerical algorithms. *Phys. Rev. E: Stat., Nonlinear, Soft Matter Phys.*, 71(6):066208, 2005.
3. L. Dagum and R. Menon. OpenMP: an industry standard API for shared-memory programming. *IEEE Comput. Sci. Eng.*, 5(1):46–55, 1998.
4. A. Folch-Fortuny, A. F. Villaverde, A. Ferrer, and J. R. Banga. Enabling network inference methods to handle missing data and outliers. *BMC Bioinform.*, 16(1): 283, 2015.
5. V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, Y. Saeys, and P. Geurts. Inferring regulatory networks from expression data using tree-based methods. *PLOS ONE*, 5(9):e12776, 2010.
6. I. Jang, A. Margolin, and A. Califano. hARACNe: improving the accuracy of regulatory model reverse engineering via higher-order data processing inequality tests. *Interface Focus*, 3(4):20130011, 2013.
7. N. Le Novère. Quantitative and logic modelling of molecular and gene networks. *Nature Reviews Genetics*, 2015.
8. P. Meyer, F. Lafitte, and G. Bontempi. MINET: A R/Bioconductor package for inferring large transcriptional networks using mutual information. *BMC Bioinform.*, 9(1):461, 2008.
9. M. Samoilov, A. Arkin, and J. Ross. On the deduction of chemical reaction pathways from measurements of time series of concentrations. *Chaos*, 11(1):108–114, 2001.
10. T. Schaffter, D. Marbach, and D. Floreano. GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*, 27(16):2263–2270, 2011.
11. T. Schreiber. Measuring information transfer. *Phys. Rev. Lett.*, 85(2):461, 2000.
12. C. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.
13. A. F. Villaverde, J. Ross, F. Morán, and J. R. Banga. MIDER: network inference with mutual information distance and entropy reduction. *PLOS ONE*, 9(5):e96732, 2014.