Discovering a Causal DAG for genes via the Mappa Mundi algorithm

Robert R. Tucci
tucci@ar-tiste.com

March 16, 2025

Abstract

This paper can be viewed as an application and further refinement of the Mappa Mundi (MM) algorithm, an algorithm for which software and a paper are available at github. The MM algorithm can be used to extract causal DAGs from chronologically ordered data that is either in sentences text or tabular numeric form. In this paper, we describe how to apply it to finding what are called Gene Regulatory Networks (GRN), Autoregulon Nets and Network Motifs in the Genomics and Systems Biology literature.

1 Introduction

This paper can be viewed as an application and further refinement of the Mappa Mundi (MM) algorithm. In this paper, we describe how to apply it to finding what are called Gene Regulatory Networks (GRN), autoregulon (AR) nets and Network Motifs in the Genomics and Systems Biology literature (Ref.[1]).

The MM algorithm was first proposed in Ref.[4] for DAG Extraction From Text (DEFT). In Ref.[4], it was used to extract causal DAGs from 3 P.G. Wodehouse short stories and the scripts of 3 PiXar movies. In general, the MM algorithm can extract causal DAGs from 2 or more text files, as long as each of those text files recounts actions in chronological order. So it will work with time stamped lab notebooks or medical records, but it won't work with textbooks or fiction with time travel or flashback shenanigans.

After Ref.[4], the MM algorithm was applied in Ref.[3] to extracting causal DAGs from FitBit times series data.

In this paper, we use the MM algorithm to extract DAGs from time series data for concentrations of gene expressions and transcription factors. The DAGs we obtain are called GRN in the Genomics literature. GRN are a special case of AR networks. AR nets are discussed in the chapter entitled "Autoregulon Networks (Network Motifs)" of my book Ref.[2]. AR nets are a special case of Dynamical systems (DS). DS are discussed in the chapter entitled "Dynamical Systems" of my book Ref.[2].

2 Comparing 2 TS Records

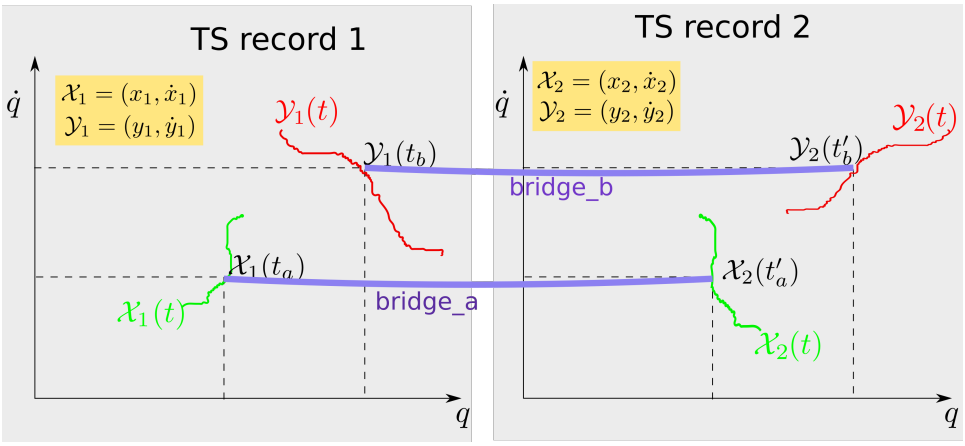


Figure 1: Causal bridges a and b spanning phase planes for TS records 1 and 2.

We will use the term **time series (TS) record** to refer to a data file such as a spreadsheet with the first column giving time increasing downwards and additional columns giving the values of q_i (a state variable) and $\dot{q}_i$ (time derivative of q_i) for $i = 1, 2, \ldots N$ at the time indicated by the first column. Any q_i or $\dot{q}_i$ is called a **state variable**. The multidimensional space $(q_i)_{i=0}^N$ is called **configuration space**. The multidimensional space $(q_i, \dot{q}_i)_{i=0}^N$ is called **phase space**. A two dimensional space $(q_i, \dot{q}_i)$ for any i is called a **phase plane**.

A TS record may contain initially only a column for time and columns for configuration space, if the change in time between rows is small. In that case we can subtract two consecutive q_i readings and divide by the difference in time to obtain the $\dot{q}_i$ columns.

In this paper, each q_i represents either a **transcription factor (TF)** concentration, or a **gene expression (GE)** concentration.[1]

Suppose x and y are any two q_i. For $\xi \in \mathcal{A}_{xy} = \{x \to y, y \to x\}$, let

n_{acc}^{ξ}: number of arrows accepted, initially zero

[1]The terms "transcription factor" and "gene expression" are both defined in the AR networks chapter of my free book Ref.[2]

n^{ξ}_{rej}: number of arrows rejected, initially zero
$N^{\xi} = n^{\xi}_{acc} + n^{\xi}_{rej}$: number of arrows detected
$p^{\xi}_{acc} = \frac{n^{\xi}_{acc}}{n^{\xi}_{acc}+n^{\xi}_{rej}}$: probability of causal arrow, initially zero.
N^*: threshold value for N^{ξ}
p^*_{acc}: threshold value for p^{ξ}_{acc}
The MM algorithm for finding an AR net from 2 TS records, consists of the following steps:

1. **Compare 2 TS records and score arrows**

   Consider Fig.1. In that figure, suppose the two ends of bridge a are approximately equal: $\mathcal{X}_1(t_a) \approx \mathcal{X}_2(t'_a)$ and the two ends of bridge b are approximately equal too: $\mathcal{Y}_1(t_b) \approx \mathcal{Y}_2(t'_b)$. [2]

   At the very least, one must store (unless they are the default value zero) the current values of n^{ξ}_{acc} and n^{ξ}_{rec} for $\xi \in \mathcal{A}_{xy}$, where x and y are any two q_i.

   - if $t_a < t_b$ and $t'_a < t'_b$ (bridges are parallel in time)[3]

   $$\begin{cases} n^{x\to y}_{acc} ++ \\ N^{x\to y} ++ \end{cases} \tag{1}$$

   - if $t_a > t_b$ and $t'_a > t'_b$ (bridges are parallel in time)

   $$\begin{cases} n^{y\to x}_{acc} ++ \\ N^{y\to x} ++ \end{cases} \tag{2}$$

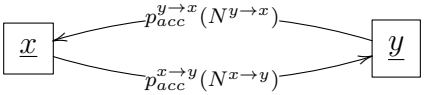
   - if $t_a < t_b$ and $t'_a > t'_b$ (bridges are crossing in time)

   $$\begin{cases} n^{x\to y}_{rej} ++ \\ N^{x\to y} ++ \end{cases} \tag{3}$$

   - if $t_a > t_b$ and $t'_a < t'_b$ (bridges are crossing in time)

   $$\begin{cases} n^{y\to x}_{rej} ++ \\ N^{y\to x} ++ \end{cases} \tag{4}$$

2. **Draw DAG**



(5)

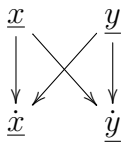
[2] By $\mathcal{X} \approx \mathcal{Y}$ we mean that both of these vectors are inside the same small bin or open ball of size given by a pre-specified precision.
[3] $x++$ means add 1 to x

If x and y are any two q_i, draw an arrow from autoregulon $\boxed{\underline{x}}$ to autoregulon $\boxed{\underline{x}}$ iff both $p_{acc}^{x\to y} > p^*$ and $N^{x\to y} > N^*$ are true.

Likewise, draw an arrow from autoregulon $\boxed{\underline{y}}$ to autoregulon $\boxed{\underline{x}}$ iff both $p_{acc}^{y\to x} > p^*$ and $N^{y\to x} > N^*$ are true.

When drawing an arrow, write the values of p_{acc}^{ξ} and N^{ξ} over the arrow, where $\xi \in \mathcal{A}_{xy}$. See Eq.(5) where this is done with variables. Do it with the values of those variables instead.

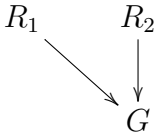
At first glance, Eq.(5) doesn't look like a DAG, because it has a cycle and DAGs are, by definition, acyclic. But Eq.(5) does indeed represent a DAG because, as explained in the AR net chapter of Ref.[2], Eq.(5) represents this net:



(6)

which is acyclic.

If R_1 and R_2 are two TS records and G is the DAG obtained by following the MM algorithm presented above, then one can represent that TS algorithm diagrammatically by



(7)

In the next section, we will try to define the merging of more than two records into a single DAG.

3 Comparing More Than 2 TS Records

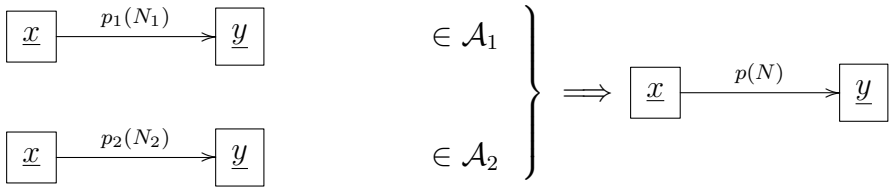
3.1 Merging two or more DAGs into one DAG

Suppose G_1 and G_2 are two DAGs that we wish to merge. Let $\mathcal{A}_i$ be the set of arrows of DAG G_i for $i = 1, 2$.

An autoregulon $\boxed{\underline{x}}$ in a DAG G stores as "cargo" an ordered set $\mathbb{X} = [\mathcal{X}(t_1), \mathcal{X}(t_2), \mathcal{X}(t_3), \ldots]$ where $t_1 < t_2 < t_3 < \ldots$. An autoregulon $\boxed{\underline{x}_1}$ in DAG G_1 is equal to an autoregulon $\boxed{\underline{x}_2}$ in DAG G_2 if $\mathbb{X}_1$ and $\mathbb{X}_2$ have at least one $\mathcal{X}(t_i)$ in common.

If AR nodes $\boxed{\underline{x}_1}$ and $\boxed{\underline{x}_2}$ are merged, the merged node should store both $\mathbb{X}_1$ and $\mathbb{X}_2$.

- merging doubly overlapping arrows (overlap in head and tail)



(8)

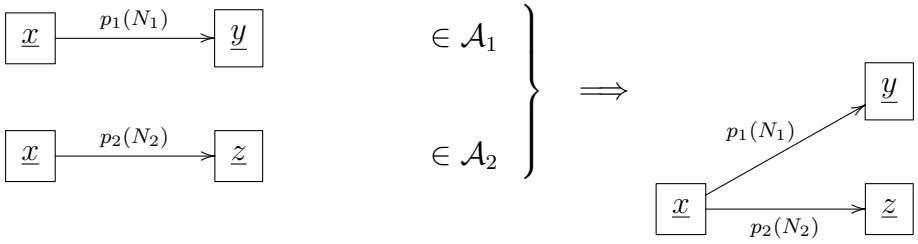
where

$$N = N_1 + N_2 \tag{9}$$

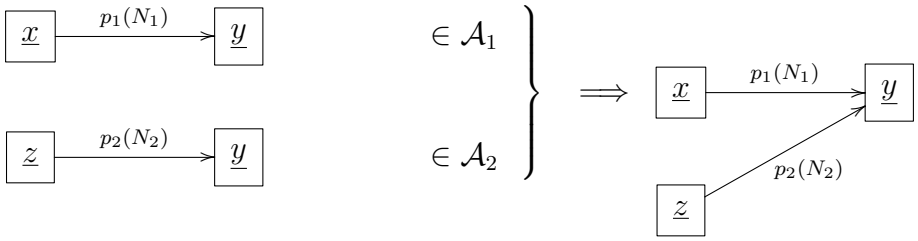
and

$$p = \frac{\sum_{i=1}^{2} n_{acc,i}}{\sum_{i=1}^{2}(n_{acc,i} + n_{rej,i})} = \frac{p_1 N_1 + p_2 N_2}{N_1 + N_2} = p_1 \frac{N_1}{N} + p_2 \frac{N_2}{N} \tag{10}$$

- merging singly overlapping arrows (overlap in head or tail but not both)

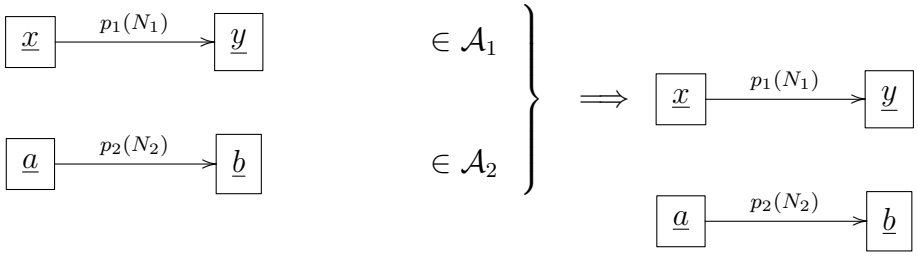


(11)



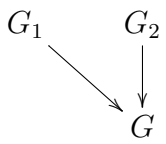
(12)

- merging non-overlapping arrows



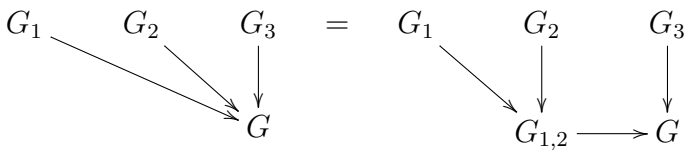
(13)

If G_1 and G_2 are two DAGs and G is the DAG obtained by following the DAG merging algorithm presented above, then one can represent that DAG merging algorithm diagrammatically by



(14)

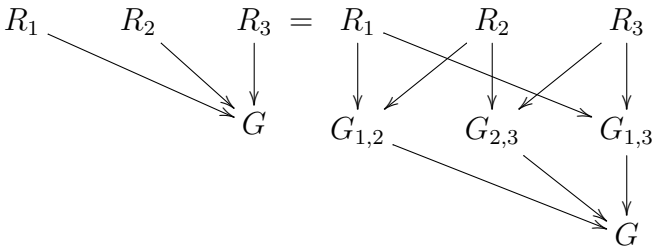
Once one defines the meaning of Eq.(14), one can use it as a building block to define the merging of more than 2 DAGs into a single DAG. For example, we can define the merging of 3 DAGs as follows



(15)

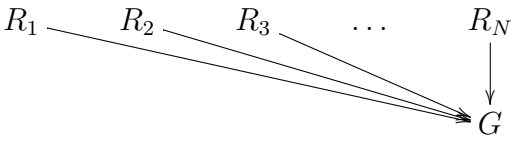
3.2 Extracting single DAG from more than 2 TS records

Once we define by Eq.(15) the merging of 3 DAGs into a single DAG, one can define the extraction of a single DAG from 3 TS records, as follows.



(16)

Suppose we have N records $R_1, R_2, \ldots, R_N$. Eq.(16) can be generalized to define

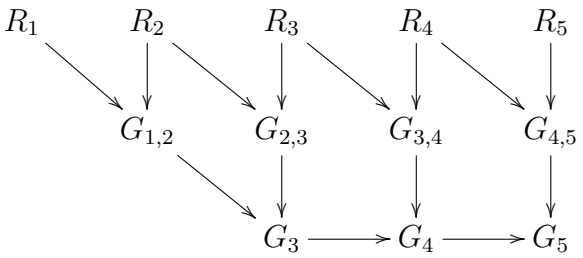


(17)

The generalization extracts a DAG from every pair of records $\{R_i, R_j\}$ where $1 \leq i < j \leq N$, and merges all the extracted DAGs into a single one.

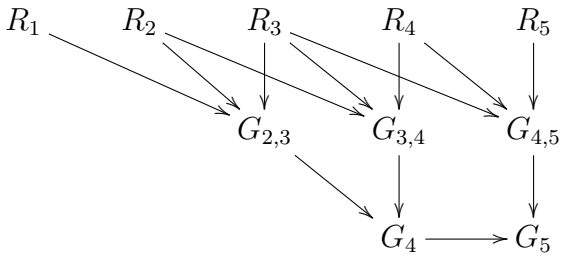
Eqs. (16) and (17) involve the comparison of $\binom{N}{2}$ record pairs. For large N, performing that many record comparisons is untenable. So we need a less laborious algorithm (even if less informative too) for extracting a single DAG from N records.

One family of such extraction algorithms is the following. Order the records R_i so that i denotes the time at which they are presented to us. If R_i is the current record, one can extract a DAG from R_i and the previous record R_{i-1} as follows



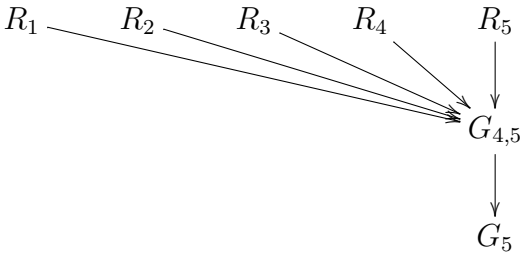
(18)

Alternatively, if one is willing to do more work, one can extract a DAG from R_i and the previous 2 records R_{i-1} and R_{i-2}, as follows



(19)

And so on. If we assume perfect memory of all past records, then we get the following, which is the same as Eq.(17).



(20)

References

[1] Uri Alon. *An introduction to systems biology: design principles of biological circuits*. Chapman and Hall/CRC, 2019.

[2] Robert R. Tucci. Bayesuvius (free book). `https://github.com/rrtucci/Bayesuvius`.

[3] Robert R. Tucci. CausalFitbit (software and paper). `https://github.com/rrtucci/CausalFitbit`.

[4] Robert R. Tucci. Mappa Mundi (software and paper). `https://github.com/rrtucci/mappa_mundi`.