The Tale of Two Clocks

Zvi Lotker[1]

[1]*Department of Communication Systems Engineering, Ben Gurion University, Israel*

Abstract—**The main question that this paper addresses is how to identify critical events in the evolution of a social network. The paper uses ideas from psychology about time perception. It is well known that time flows differently in different emotional situations. Equipped with this idea, this paper studies the relationship between two clocks. As opposed to standard synchronization, where everything is done in order to force clocks to agree on the time, the paper embraces the discrepancy between the clocks.**

This paper presents a standard model where two natural clocks exists simultaneously: the *event clock* C_e and the *weighted clock* C_w. As the paper shows, using the drift between those two clocks is useful to understand the dynamics in social networks. The main claim is that the drift between different clocks points to a critical event in the evolution of the social network, similar to time perception in psychology.

In order to demonstrate this claim, plays by William Shakespeare were used, and from them two clocks were created: the *"word time"*, which is the weighted clock, and the *"response time"*, which is the event clock. The paper will introduce the concept of a *single clock drift*. A play can have many, or a single clock drift events. It is shown that in the single clock drift plays, the beginning of the drift points to a critical event in the play. The results are compared with the "standard common" opinion.

1. Introduction

An important observation, first made by Karl Ernst Von Baer in the 19th century, shows that time flows differently in different emotional situations. For example, when going through a life threatening experience such as bungee jumping, time moves slowly for the jumper. These results were later established through psychology and neuroscience research, and were explained by our adapted *Time perception.* There are many well known conditions that relate to the phenomenon of the shrinking and the expanding of time, such as age, sickness, drugs etc. In the core of these phenomena is our biological clock which adapt its relative pace to its environment. For a recent review on the subject of psychology of time see [4].

This paper studies the question of detecting time perceptions in social networks. It asks what is time in social networks? How can we measure it? The standard tool to measure time are clocks, therefore, instead of studying time, this paper will concentrate on clocks. So, rephrasing the question, this paper will study: what is a clock, or to be precise, what is a clock in social networks?

The paper raises the possibility whether there are actually many time perceptions, and many ways to measure them. Once the possibility of different clocks exist, we are forced to answer questions concerning the clocks: how to compare clocks? How to determine if two clocks are equivalent or different?

Time is a central theme in human thoughts, and usually, there is more than one definition of time. Time allows us to look at and study changes. Unlike other disciplines, physics starts with the existence of space-time. Based on that, we can measure time in our daily lives, and enjoy the illusion of the universality of time.

But when we try to force this "time" on the evolution of social networks, it doesn't always make sense. Physical time is not always relevant to social networks, where time flows differently. Consider the spreading of Bin-Laden's capture through Twitter: in a matter of seconds the news appeared all over the net, via billions of tweets. Watching Twitter in standard physical time would have missed it.

In mathematics there are two different models of time: *continuous-time* and *discrete-time*. Continuous-time is used to describe smooth and continuous change.

The domain of discrete-time is used to describe atomic and discrete events, where an event cannot be split into a small, sub-event. In principle, every evolutionary structure has its own measure of time. According to this, the way we measure time is influenced by the structure of what is measured. The existence of many important quantities/concepts in a social network, that can be understood better in the context of evolution, suggests that there are different ways to measure time, and therefore many different clocks.

A useful guideline to understand how time flows is to ask: what changes? When trying to capture the meaning of time in social networks, the first observation is that time is discrete. According to the guiding principle, what determines the property of time is what changes in the structure. The change of the structure in social networks is adding or deleting a vertex/edge, which are atomic events. These events are called the atomic-graph events. Since the atomic-graph events are discrete, the time is discrete. Therefore, one way to measure the time in social networks is the stream of atomic-graph events.

author is supported by FSMP, and is hosted at IRIF, and by the Israel Science Foundation (grant 1549/13)

IEEE/ACM ASONAM 2016, August 18-21 2016, San Francisco, CA, USA
978-1-5090-2846-7/16/$31.00 ©2016 IEEE

We cannot hope to achieve a deeper insight into evolution of social networks without a narrative. There is no history without a story. The richest vein of narratives is art, and in particular theatre. Time in theatre is complex, due to the medium itself. Time in theatre is divided into several different time dimensions, as will be explained through the play *Hamlet*. Time of the text, which is dense i.e., what happens from act I to the end of the play. This time can be flexible, and doesn't have to be chronological. There is the time of the show, or the actual time of the event, about 3-5 hours when performing *Hamlet*. There is the historic time, or the time the play takes place in (the 16th century in *Hamlet*, more or less). Inside the time of the text (or the plot) there is the time of the actual plot, but also the time of the story, which can be longer: Hamlet's story starts when Hamlet's father was assassinated by his uncle, long before the play begins.

This paper will use two clocks in order to detect time perception, since time perception must be relative to another clock. The two clocks are: the time of the plot (of what happens between the characters)- which is the *event clock*, and the *weighted clock*, which is an approximation of what is performed on stage. While the event clock is the reflection of the structure of conversation in the play, the weighted clock captures the complexity of what is being said. When a character discusses a complex idea the weighted clock "ticks" much faster than the event clock. On the other hand, when it is time for action (implying danger, comic or tragic) then the event clock "ticks" faster than the weighted clock, meaning lots of short responses, with few words. The event clock is discrete, since its atomic building blocks are the *responses*, which cannot be broken into smaller pieces and still retain the same properties. The weighted clock is also discrete since it counts words. It is an approximation to the stage-time.

It seems that the event clock and weighted clock are unique to plays. But they can be found everywhere, especially in social networks. For example, consider emails: the event clock is advanced by 1 for every email that is sent, and the weighted clock advances by the number of words in each email. Emails with more words usually contain more information. A similar example can be applied to any text-oriented social network, such as Facebook, LinkedIn, Twitter etc.

The remainder of this paper is organized as follows: in section 3 the model is introduced and the definition of clocks is given. Section 4 shows how to compare different clocks. Section 5 defines clock drift and shows how to compute it. In section 6 the results are compared to the "standard common" opinion, represented by CliffsNotes which are student's study guides.

2. Related Work

The idea of examining social networks as a dynamic system is not new (see [9]). In [1] Réka and László show that a small change in the evolution of the social network can lead to a major difference in the network's structure. Recently, several papers considered the dynamics of social networks [14, 16, 5]. In general, modeling dynamic networks requires some assumptions about the evolution of the network over time. Perhaps, the most general case is to assume that the graph can change without any restrictions, and it is controlled by some, unlimited, all-powerful adversary. This model is usually called the *Evolving Graph* (see [3]). However, when using an all powerful adversary, usually, the task of designing an efficient algorithm becomes too complex. To overcome this, several authors have suggested to restrict the power of the adversary to either a Markovian evolving graph model (see [3]), or edge evolving Markovian evolving graph model (see [7]).

Rydberg-Cox [17] analyzed the linguistic dependency of Greek classic plays and texts, including the *Iliad* and the *Odyssey*, using the graph of who spoke to whom and social network tools. Stiller et al. [18] measured the diameter of the social network in ten Shakespeare plays, and showed that the social networks in Shakespeare's plays indeed have small world properties. These authors justify their evaluation of social network algorithms on plays by the view that effective plays must reflect human society. In [13] the author uses the Shakespeare's *Julias Ceaser* play to prove the soundness of the voting algorithm. The idea was to construct a social network from the play, run the voting algorithm and compare the result of the algorithm to the narrative of the play.

Usually, when people have two different clocks, they wish them to show the same time, this is called *clock synchronization*. A plethora of papers have been published on clock synchronization. One of the corner stone papers is [11], by Leslie Lamport. One book recently published on clock synchronization is [20]. Another survey paper is [19]. However, the current paper takes the opposite approach: while research in clock synchronization tries to synchronize clocks as best as possible, this study is interested in the time when the clocks are at their most un-synchronized positions.

3. The Model and Notation

The first step in defining the model is to have some notation. Let $\tau_s < \tau_e$ be real numbers. A close interval is $[\tau_s, \tau_e] = \{x \in \mathbb{R} : \tau_s \leq x \leq \tau_e\}$, an open interval is $(\tau_s, \tau_e) = \{x \in \mathbb{R} : \tau_s < x < \tau_e\}$. The subscript s stands for the start of the interval, and the subscript e stands for the end of the interval.

A directed graph $G(V, E)$ is a pair of sets; the set $V \neq \emptyset$ of nodes is non-empty, and the set $E \subseteq V \times V$. The set E is the collection of all directed edges. Each directed edge $e = (u, v) \in E$ is an ordered pair $v, u \in V$ of nodes. We say that a directed edge (u, v) starts from node u, which is called the source of the edge and ends at node v, which is called the target of the edge.

An evolving network is a sequence of graphs $(G_1(V_1, E_1), G_2(V_2, E_2), ..., G_n(V_n, E_n))$. Evolving graphs are designed to model dynamic networks in a compact way, in which availability and capabilities of links change over time. When the changes are limited to only adding edges, the model is called *Accumulated Links Stream*. An example for accumulated links stream is the relation of "who talk after whom" in a given play. Here direct edges represent the relation between two characters. En edge is added between two characters who spoke at the same scene, one after another. Edges arrive one after another, according to the chronology of the play. Note that in this case edges do not disappear. Since the main focus of this paper is time and clocks, it is more convenient to define the mathematical object as a sequence. Therefore, first define a space which will contain the elements of the sequence, and then define the relevant sequence.

In order to describe a *link stream*, it is needed to have a set of nodes, denoted by V. Let $\mathcal{E} = V \times V$ be the set of all possible edges over V. The space $\mathcal{L} = \mathcal{E} \times \mathbb{R}$ will contain the elements of the link stream. The elements of $\mathcal{L}$ are called *Arriving links*. Let the function $T : \mathcal{L} \to \mathbb{R}$ be projection function which gives to each element in L it's arrival time i.e. $T(l_i) = t_i$.

Definition 1. *Any finite sequence of arriving links $L = (l_1, ..., l_n)$ s.t. $l_i \in \mathcal{L}$ is a* link stream *if for all $i = 1, ..., n-1$, the elements of the sequence are sorted, i.e., $T(l_i) < T(l_{i+1})$.*

This paper borrows the definition of a weighted evolving graph from [12]. The next step is to define *weighted link stream*. Again, first define the space which contains the sequence. Let $\mathcal{L}^w = \mathcal{L} \times \mathbb{R}$ be the space of the weighted link stream elements. The elements of $\mathcal{L}^w$ are called *arriving weighted links*. Again, we use the function $T^w : \mathcal{L}^w \to \mathbb{R}$ to give the arrival time t_i of the edge e_i in the sequence L_w. Denote as $W : \mathcal{L}^w \to \mathbb{R}$ be the projection function which give each element in L^w its weight, i.e., $W(l_i^w) = w_i$.

Definition 2. *Any finite sequence of arriving weighted links $L_w = (l_1^w, ..., l_n^w)$ s.t. $l_i^w \in \mathcal{L}^w$ is a* weighted link stream *if for all $i = 1, ..., n-1$, the elements of the sequence are sorted, i.e., $T^w(l_i) < T^w(l_{i+1})$.*

3.1. Definition of clock and some properties

The first step when studying two clocks is to verify their symmetry. Obviously, Euclidean time enjoys Euclidean symmetry: translation and rotation are invariant of time (ignore the relativity phenomenon in the context of social networks). Another symmetry concerning comparison of clocks, is swapping between clocks. This means that any comparable function γ of two clocks $\gamma(C_1, C_2)$ has to satisfy the swapping action, i.e, $\gamma(C_1, C_2) = \gamma(C_2, C_1)$.

The main tool used in this paper is computing when different clocks are drifting from one another. Therefore, a very general definition of a clock is used. Following the assumption that time exists and flows constantly in one direction, and that a clock is a device which measures time; the basic requirement from a clock is that it will always advance through time. Therefore, the paper considers the following definition of a clock.

Definition 3. *A* clock *C is a strictly monotonic bijective continuous function from the standard time interval $[\tau_s, \tau_e]$ to the measured time interval $[\mu_s, \mu_e]$, i.e., $C : [\tau_s, \tau_e] \to [\mu_s, \mu_e]$. The domain of a clock is called the* standard time*, and the range of a clock is called the* measure time.

Note that according to the definition of clocks, $C(\tau_s) = \mu_s$ and $C(\tau_e) = \mu_e$. The *standard clock* is defined as the identity function. Notice that, since the clock is a strictly monotonic bijective continuous function, each clock has its own inverse function, which is also strictly monotonic continuous, and therefore also a clock. Sometimes it is useful to define the reduce clocks.

Definition 4. *Let $C : [\tau_s, \tau_e] \to [\mu_s, \mu_e]$ be a clocks, and $[\tau_1, \tau_2] \subset [\tau_s, \tau_e]$ be a sub-interval. A* reduce clock *$C|_{\tau_1}^{\tau_2}$ is a clock defined on the sub-interval $[\tau_2, \tau_2]$ s.t. for all $t \in [\tau_2, \tau_2]$ $C|_{\tau_1}^{\tau_2}(t) = C(t)$.*

A *discrete clock* is a discrete monotonic increasing function over the real numbers, i.e., $C : A \to \mathbb{R}$, where A is a discrete subset of real numbers. Note that each discrete clock can be *extended to a clock* in many ways. One way to do so is by using linear interpolation. This extension will be used when a discrete clock is mentioned.

Definition 5. *Let C be a discrete clock, defined on the discrete set A. Assume that all of the elements of A are sorted, and denote them by $a_1, a_2, ..., a_n \in A$ s.t $a_i < a_{i+1}$. A time tick sequence $\{\Delta_i\}_{i=0}^{n-1}$ is the amount of measure time that it takes for a clock to jump between two consecutive time points, in the set A, i.e., $\Delta_i = C(a_{i+1}) - C(a_i)$, for all $i = 1, ..., n-1$. The first element of the sequence Δ_0 has a special definition: $\Delta_0 = C(a_1)$.*

Note that once the time tick sequence is given, one can compute its clock at the time a_i easily, according to the following formula: $C(a_i) = \sum_{j=0}^{i-1} \Delta_j$. Again, once a discrete clock is defined, it is extended to a clock through linear interpolation.

Given a link stream L_w the discrete event clock C_e is defined on the set of all arriving times $\mathcal{A} = \{T^w(l_i^w) \in \mathbb{R} : l_i^w \in \mathcal{L}^w\}$, and it is equal to the number of edges arriving up to time $t \in \mathcal{A}$, in the link stream L^w, i.e.,

$$C_e(t) = |\{l_i^w \in \mathcal{L}^w : T(l_i^w) \leq t\}|.$$

Now define the discrete weighted clock C_w, as the accumulated weight of the edges arriving up to time t, in the evolving graph,

$$C_w(t) = \sum_{i \in T(t)} W(l_i^w) \tag{1}$$

where $T(t) = \{i : T(l_i^w) \leq t\}$, for all $t \in \mathcal{A}$. The C_e, C_w discrete clocks are extended to C_e, C_w clocks through linear interpolation.

Definition 6. *The two clocks C_1, C_2 are* linear *with respect to each other, if there exists a two linear transformation $L_i(t) = a_i t + b_i, i = 1, 2$, s.t. $C_1(L_1(t)) = L_2(C_2(t))$ are equal as a function of measure time. In this case we will call them* equivalent*, and we denote this by $C_1 \sim C_2$*

Note that Both functions L_1, L_2 have an inverse function L_1^{-1}, L_2^{-1}. The above definition uses two different linear functions: L_1, L_2. The purpose of L_1 is to rescale and translate the standard time. The purpose of L_2 is to rescale and translate the measure time. Note that the linear relation between two clocks is reflexive, i.e., C is in a linear relation with respect to itself. The linear relation is symmetric, i.e. if C_1 is linear with respect to C_2. Since the clocks C_1, C_2 are strictly monotonic, and the linear function is reversible, it follows that C_2 is linear with respect to C_1. Lastly, the linear relation is transitive for the same reason, and therefore it is a equivalent relation.

Definition 7. *A clock C would be called* normal *if its domain and range are the unit interval i.e., $C : [0,1] \to [0,1]$*

Using the linear equivalent relation, one can normalize any clock. To do so it is useful to consider some general operators acting on the clock. The first operator is used to rescale the standard time, and the second operator is used to rescale the measured time of a clock. Formally, denote:

$$\nu_1(t,a,b) = a + (b-a)t,$$

Now the rescaling of the measured time of the clock C is given by:

$$\nu_2(C(t),a,b) = \frac{C(t) - C(a)}{C(b) - C(a)}.$$

Lemma 8. *For any clock C there is a unique normalized clock $N(C) \sim C$.*

Proof. There are two steps in the construction of the clock $N(C)$. First, shift and normalize the standard time interval $[\tau_s, \tau_e]$ into the unit standard time interval. Denote the result of this process by the clock C_s. The second step is to normalize the measure time to the unit interval. The combined clocks are $N(C)$.

The function $\nu_1(t, \tau_s, \tau_e)$ normalizes the standard time. Clearly, $\nu_1(t, \tau_s, \tau_e)$ maps the time interval $[0,1]$ to the standard time interval $[\tau_s, \tau_e]$. Therefore, the clock C_s can be defined as the rescaling clock of the standard time C_s as follows:

$$C_s(t) = C\left(\nu_1(t)\right) = C\left(\tau_s + t(\tau_e - \tau_s)\right).$$

Note that since the domain of clock C is the time interval $[\tau_s, \tau_e]$, the domain of the clock C_s is the unit interval. Moreover, since the function $\tau_s + t(\tau_e - \tau_s)$ is linear, it follows that both clocks C and C_s are equivalent. Now the clock domain of C_s is the unit interval. The next step is to normalize the measure time of clock C_s. This is done using the following function:

$$N(C)(t) = \nu_2(C_s(t), 0, 1) = \frac{C_s(t) - C_s(0)}{C_s(1) - C_s(0)}.$$

This clock is equivalent to the clock C_s. Notice that the clock $N(C)$ is a normalized clock. Since clocks are monotone, continues and $\tau_s < \tau_e$, both functions ν_1, ν_2 are reversible, and therefore the clock $N(C)$ is unique. $\square$

Using Lemma 8, all clocks are assumed to be normalized. If the reader has an example for an un-normalized clock, it is recommended that she normalize it. Note that the action of normalizing clocks can take two clocks that are synchronized at a sub-interval, and un-synchronize them at this sub-interval (see Figure 1). However, the main interest is to find the beginning of the drift, and therefore, normalizing and second moment are useful and legitimate tools since they emphasize this point in time.

Consider the example in Figure 1, where C_s is the normalized standard clock. The general clock C_g is synchronize with the standard clock, in the time interval $[0, 0.5]$. At standard time $t = 0.5$ the general clock C_g runs twice as fast as the standard time $[0.5, 1]$. Note that the normalized clock $N(C_g)$ is not synchronize with the standard C_s in the unit interval, see the middle diagram in Figure 1. Notice that moving from left to right in the diagrams, the beginning β of the drift is amplified, and therefore it is easier to detect the β in the second moment diagram.

However, the purpose of normalization is *not* to synchronize clocks, but to discover times when the clocks start to be "un-synchronize". More on this in 5.

4. Time diagrams

One major problem regarding two different clocks C_1, C_2 is how to compare them. The natural thing to do when comparing two clocks is to draw them as a function of each other. This can be done easily, using the inverse function of these clocks C_1^{-1}, C_2^{-1}, respectively. Note that a clock is a monotonic increasing function, therefore, the functions C_1^{-1}, C_2^{-1} exist. The time diagram of C_1 with respect to C_2 is the function $C_2(C_1^{-}1)$, while the time diagram of C_2 with respect to C_1 is the function $C_1(C_2^{-}1)$.

Another useful diagram is the *Second Moment Clocks Diagram*. This diagram captures the "variance" of one clock with respect to the other clock. The idea is to remove the first clock from the second clock, after normalizing them both. This technique is borrowed from stochastic processes. Formally, the x-coordinate is the first clock, while the y-coordinate is the difference between the clocks, i.e., $y(t) = C_2(t) - C_1(t)$, assuming both clocks are normalized. This means that in the second moment clocks diagram the function starts at 0 and ends at 0, since $C_1(0) = C_2(0) = 0$, and $C_1(1) = C_2(1) = 1$. Therefore, $C_1(0) - C_2(0) = C_1(1) = C_2(1) = 0$. The advantage of the second moment clocks diagram is that by removing the first clock from the second clock, the scale actually changes from a $[0,1]$ interval to the deviation of the second clock from the first, which is usually much smaller then the full interval. Therefore, the difference between the clocks is clearer!

An example of a clock diagram can be seen in Figure 1, where the x-coordinate is measuring the standard time, and the y-coordinate measures time according to a general clock C_g. The normalized diagram is shown in the middle of Figure 1. In the right diagram is shown the second

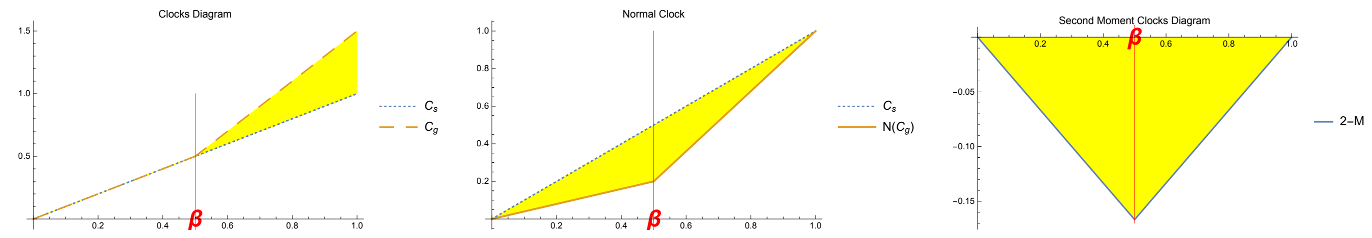

Figure 1: Three different clocks diagrams: on the left it is shown the diagram between C_s and C_g. The middle diagram is the the normalized diagram between the clocks $N(C_s)$ and $N(C_g)$. The right diagram is the second moment diagram. The yellow areas indicates the drift between the clocks.

moment clocks diagram of the same two clocks. Again, x-coordinate represents the time of the first clock, while the y-coordinate represents the difference between the clocks, i.e., $y = C_2(t) - C_1(t)$.

One can use the clocks diagram for several purposes. First, when using this diagram it is easy to compare the clocks. Second, you can easily determine which clock runs faster at a specific time. The same information can also be seen in the second moment clocks diagram.

Definition 9. *Two clocks, C_1, C_2, defined on the the interval $[\tau_s, \tau_e]$, have a* clocks drift *with respect to each other, if they are not equivalent on the set $[\tau_s, \tau_e]$. A clocks drift is denoted by $C_1 \not\sim C_2$.*

5. Single Clock Drift

Consider the following example: two clocks start at the same speed. Then one of the clocks changes its speed in the middle of the interval, and maintains its new speed until the end, while the other clock does not change at all. This scenario can be seen in Figure 1. Following this example, the definition of a single Single Clock Drift is:

Definition 10. *Let C_1, C_2 be two clocks defined on the same standard time interval $[\tau_s, \tau_e]$. Assume $\beta \in (\tau_s, \tau_e)$. The Clocks C_1, C_2 have a single clock drift at time β, which is denoted by $C_1 \underset{\beta}{\simeq} C_2$ if $C_1|_{\tau_s}^{\beta} \sim C_2|_{\tau_s}^{\beta}$, and $C_1|_{\beta}^{\tau_e} \sim C_2|_{\beta}^{\tau_e}$, and $C_1 \not\sim C_2$.*

Note that this $\underset{\beta}{\simeq}$ is an equivalent relation for a fixed given β, since for a given β the relation $\underset{\beta}{\simeq}$ is reflexive, symmetric and transitive. Sometimes β is called the *beginning of the drift.*

Lemma 11. *Assume $C_1 \underset{\beta}{\simeq} C_2$ have a single clock drift. The normalized clocks $N(C_1) \underset{\beta'}{\simeq} N(C_2)$ have a single clock drift on the standard time interval $[0, 1]$, and the beginning of the drift is $\beta' := \nu_1^{-1}(\beta, \tau_s, \tau_e) = \frac{\beta - \tau_s}{\tau_e - \tau_s}$.*

Proof. According to Lemma 8 the normalized clocks are

$$N(C_i)(t) = \frac{C_{s,i}(t) - C_i(\tau_s)}{C_i(\tau_e) - C_i(\tau_s)}.$$

Now from $C_1 \underset{\beta}{\simeq} C_2$ it follows that there exist $a_1, b_1, a_2, b_2 \in \mathbb{R}$ s.t. $a_1, a_2 > 0$ and

$$C_2(t) = \begin{cases} a_1 C_1(t) + b_1 & \text{for } t \in [\tau_s, \beta] \\ a_2 C_1(t) + b_2 & \text{for } t \in [\beta, \tau_e] \end{cases} \tag{2}$$

By substituting the definition of $N(C_1)(t)$ and $N(C_2)(t)$ in the linear two piecewise (2) it follows that $N(C_1)|_0^{\beta'} \sim N(C_2)|_0^{\beta'}$ and $N(C_1)|_{\beta'}^1 \sim N(C_2)|_{\beta'}^1$. Now $N(C_1) \not\sim N(C_2)$, follows from the fact that $C_1 \not\sim C_2$, which is equivalent to $\det\left(\begin{pmatrix} a_1 & b_1 \\ a_2 & b_2 \end{pmatrix}\right) \neq 0$

□

5.1. Finding a Single Clock Drift Using Dynamic Programming Algorithm

In many cases, the single clock drift can explain the time diagram. One needs to find the best single clock drift among all possible single clock drifts.

One way to find a single clock drift is through dynamic programming. After formulating the dynamic programming, the best single clock drift can be found by solving it with standard tools. However, since the dynamic programming is simple, it is possible to get a combinatorial algorithm that runs in linear time.

Since by assumption, there is a single clock drift, the clocks diagram can be represented using two straight lines. After normalization, denote those lines by $l_1(x) = a_1 x$ and $l_2(x) = a_2 x + 1 - a_2$. Now, the object function is the sum of the L_2 norm, from the time diagram to those lines, i.e.:

$$\begin{aligned} F_0(k, a_1, a_2) &= \sum_{i=1}^{k} \left(C_1(t_i) - l_1(C_2(t_i))\right)^2 + \\ &+ \sum_{j=k+1}^{n} \left(C_2(t_i) - l_2(C_1(t_i))\right)^2 . \end{aligned}$$

The next two constraints simply say that clocks are monotonic functions, and therefore, linear clocks must have a slope that is a positive number. The last constraint that needs to be satisfied is that clocks are continues bijective functions, and therefore the start of the drift needs to be between t_k and t_{k+1} i.e.,: $t_k \leq \frac{a_2 - 1}{a_2 - a_1} \leq t_{k+1}$. To conclude, the full formalization of the dynamic programming is given below:

$$
\begin{aligned}
DP(C_1, C_2) = &\underset{a_1,a_2,k}{\text{minimize}} F_0(k, a_1, a_2)\\
&\text{subject to}\\
&a_i \geq 0,\ i = 1, 2\\
&t_k \leq \frac{a_2 - 1}{a_2 - a_1} \leq t_{k+1}
\end{aligned}
\tag{3}
$$

Note that since it is assumed that clocks are strictly monotonic it follows that $DP(C_1, C_2) = DP(C_2, C_1)$, and so the swapping between clocks is preserved.

If one wishes to solve this problem efficiently, it is possible to use the same idea from [2]: reduce the mathematical dynamic programming formulation into several different cases, where each case can be computed in a constant time amortizing. Therefore, according to [2] the time complexity to solve the dynamic programing is $O(n)$ where n is the number of ticks of the standard clock.

Denote by a_1^*, a_2^* the value of the variables a_1, a_2 in the optimal solution of the problem (3). If the line $y = a_1 t$ intersects with the line $y = a_2^* t - (1 - a_2^*)$, denote the standard time in the intersection point by $\widehat{\beta}_{DP} = \frac{a_2^*-1}{a_2^*-a_1^*}$. This is the estimator of the dynamic programming to the beginning of the clock drift, in the case of a single clock drift.

5.2. The Gap algorithm of Two Clocks

One important property of a single drift between clocks is their maximal gap. The idea behind this is as follows: assuming that the first clock is perfect, i.e., that it is the standard time, a natural question is how much has the second clock drifted from the first clock. Clearly, this is captured by $C_2 - C_1$. Note that if they are fully synchronized and normalized, then the difference will be 0.

Before computing the size of the drift between the two clocks, one needs to remove the Euclidean symmetry. This is done with the normalized procedure which is described in 8. Define the function:

$$g(C_1, C_2) := \max_{t_1 \in [0,1]} C_2(t_1) - C_1(t_1) - \min_{t_2 \in [0,1]} C_2(t_2) - C_1(t_2)$$

Denote the $\widehat{\beta}_{g_M} = \arg\max_{t_1 \in [0,1]} C_2(t_1) - C_1(t_1)$, and $\widehat{\beta}_{g_m} = \arg\min_{t_2 \in [0,1]} C_2(t_2) - C_1(t_2)$.

In the definition of the gap function, the first clock was assumed to be standard, while the other was not. If the order is reversed, so that the second clock is the standard time, then the drift will remain the same. This is shown by the following Lemma.

Lemma 12.

$$g(C_1, C_2) = g(C_2, C_1)$$

Proof.

$$
\begin{aligned}
g(C_1, C_2) &= \max_{t_1} C_2(t_1) - C_1(t_1) - \min_{t_2} C_2(t_2) - C_1(t_2)\\
&= \max_{t_2} C_1(t_2) - C_2(t_2) - \min_{t_1} C_1(t_1) - C_2(t_1)\\
&= g(C_2, C_1)
\end{aligned}
$$

□

The next theorem shows that when two clocks have a single clock drift with respect to each other, both the max gap and the dynamic programming algorithm point to the same point in time.

Theorem 13. *Let $C_1 \underset{\beta}{\simeq} C_2$ be normalized clocks that have a single clock drift in the time interval $[0, 1]$ where β is the beginning of the drift. One of the alternatives holds:*

1) *If $C_1(\beta) < C_2(\beta)$, then $\widehat{\beta}_{g_M} = \widehat{\beta}_{DP} = \beta$.*
2) *If $C_2(\beta) < C_1(\beta)$, then $\widehat{\beta}_{g_m} = \widehat{\beta}_{DP} = \beta$.*

Proof. Since C_1, C_2 are normalized and have a single clock drift in the interval $[0, 1]$ it follows that there exist two positive values $a_1, a_2 \in \mathbb{R}$ i.e., $a_1 > 0, a_2 > 0$, s.t. $\det\left(\begin{pmatrix} a_1 & 0 \\ a_2 & 1 - a_2 \end{pmatrix}\right) \neq 0$ and,

$$C_2(t) = \begin{cases} a_1 C_1(t) & \text{for } t \in [0, \beta] \\ a_2 C_1(t) + 1 - a_2 & \text{for } t \in [\beta, 1] \end{cases} \tag{4}$$

Let F denote the time function converting the first clock C_1 to the second C_2, i.e., $C_2(t) = F(C_1(t))$. From equation (4) it follows that

$$F(t) = \begin{cases} a_1 t & t \leq \beta \\ a_2 t + (1 - a_2) & t > \beta \end{cases}$$

where the first linear piece is $y = a_1 t$, and the second linear piece is $y = a_2 t + (1 - a_2)$. The connection between a_1, a_2 and β is

$$\beta = \frac{a_2 - 1}{a_2 - a_1}$$

And therefore the second moment diagram is:

$$M_2(C_1, C_2) = \begin{cases} (a_1 - 1)\, t & t \leq \beta \\ (1 + t)(a_2 - 1) & t > \beta \end{cases}$$

Consider the case (1) of the theorem: $C_1(\beta) < C_2(\beta)$. It follows that $a_1 > 1$ and $0 < a_2 < 1$, therefore, the second moment diagram achieves its max at β so that: $\widehat{\beta}_{g_M} = \beta$.

Now consider the dynamic programming. The target function of 3 is always bigger then 0. The function F is a feasible solution to (3), and the value of the target function, when substituting the value of F in (3), is equal to 0. It follows that this is the optimal solution. Therefore, $a_1^* = a_1$, $a_2^* = a_2$.

The estimator of the dynamic programming to the beginning of the drift is the time the lines $y = a_1 t$, and $y = a_2 t + (1 - a_2)$ intersect, which is equal to β. So, it follows that $\widehat{\beta}_{DP} = \widehat{\beta}$.

To get the second case, just replace $\widehat{\beta}_{g_M}$ with $\widehat{\beta}_{g_m}$, in the previous proof. □

6. Comparing clocks in social networks

The main claim in this paper is that the beginning of a single clock drift between two clocks points to a critical event in the social network. This claim is problematic. First, it is not clear what the clocks measure, and what is the connection between what the clocks measure and the event itself. Another problem is how to verify that the critical event happened at the beginning of the drift.

In order to answer the latter, this paper validates the claim that the algorithms identify a critical event by examining the algorithm on plays. In order to overcome the first problem, the clocks must be global, so they can identify critical global events in the relevant network, and therefore, the weighted C_w and event C_e clocks were used.

6.1. Generating clocks from plays

An important question left unanswered is how to generate the links stream from plays. As defined, a weighted links stream is a sequence of elements $(l_i^w = (((v_i, u_i), t_i), w_i)_i)_{i=1}^n$ each belonging to $l_i^w \in \mathcal{L}^w$ for more detailed see section 2. In this paper it is assumed that v_i is the character in the play, who spoke at time t_i. The destination u_i is the character that spoke immediately after v_i, unless the scene ended, in which case the last character to speak in the scene points to itself, i.e. "self loop".

There are two ways to measure time: the event clock, and the weighted clock. Perhaps the most natural way to measure time in a play is the event clock. In the event clock, the clock is advanced by 1 every time a character speaks, i.e. $t_i = i$. The weighted time in the scope of a play is the approximation of the stage-time, which is a complicated task, depending on many changing variables. Therefore, instead of measuring the stage-time, it was estimated by counting the number of words a character spoke. The underlying assumption is that each word, regardless of its length and meaning, takes about the same amount of time to be spoken. The clock of this time is the sum of the weights w_i where w_i = # of words in the i-th response. For more details see equation (1).

6.2. Comparing the Different Clocks in Plays

The previous section developed a methodology to identify and compute a single clock drift between two clocks. In order to examine the main claim of this paper, the algorithm was tested on plays by William Shakespeare. Most plays don't have a *single* clock drift. Note that when considering a long enough period of time there will be many clock drifts. As in real life, there are many exciting events, and therefore many clock drifts. This paper focuses on plays that have one single clock drift.

In plays, the smallest component is the scene, therefore, the solution of both algorithms will be given according to the scene, and not according to the response number.

The plays examined are *Julius Caesar*, *Othello* and *Romeo and Juliet*. A links stream for each play was generated. Using the links stream, the dynamic programming (3) was solved, and the quality of the solution is represented in Figure 2 Another useful measurement for the beginning of the drift is the *second moment time diagram*, which is shown also in Figure 2. Also, the beginning of the drift was calculated, and the results will be elaborated in the next few paragraphs.

In *Julius Caesar*, both methods point to act III, scene 2, as the beginning of the drift. Specifically, the dynamic programming and the maximum gap both claim that the single clock drift starts in act III, scene 2, in Julius Caesar's funeral, where the opinion of the people shift against the conspirators. See figure 2, left column, first row. Both estimators $\widehat{\beta}_{g_M}$,$\widehat{\beta}_{DP}$ coincide. Below is the social network of who spoke after whom in act III, scene 2 of *Julius Caesar*, which is the critical event in the play. According to pagerank algorithm Anthony is the central character in this scene.

CliffsNotes supports the results of the algorithms in *Julius Caesar*. Citing CliffsNotes: "The exact climax of *Julius Caesar* has been debated for years. Some feel that the actual murder of Caesar is the climax, whereas others contend that the turning point occurs when Antony turns the crowd against the conspirators." (See [10] page 113).

In *Othello*, both methods point to act III, scene 3 as the beginning of the drift, where Iago seeds the suspicion of Othello in Cassio and Desdemona, which will divert the action of the play towards revenge of the cuckold husband. The minimum point to the climax of the play is in act IV, scene 2, where Othello has killed Desdemona but learns that she was faithful and that Iago betrayed him, causing him to take his own life. See Figure 2, middle column. Although the algorithms points to different responses in the scene, they still agree on the scene.

Again, CliffsNotes supports the results of the algorithms in *Othello*. According to CliffsNotes, this: "This scene, often called the 'temptation scene', is the most important scene in the entire play and one of the most well-known scenes in all drama. In it, Iago speaks carefully and at length with Othello and plants the seeds of suspicion and jealousy which eventually bring about the tragic events of the play...". See [6] page 43.

In *Romeo and Juliet*, both methods point to act III, scene 1 as the beginning of the drift. As theorem 13 dictate, in this case the dynamic programming coincide with the minimum point of the gap $\widehat{\beta}_{g_m}$, see Figure 2, right column. In the first row both methods point to the same event in the play. The second moment diagram shows that the time drift is more complicated than a single clock drift. However, a single clock drift is a good approximation for the second moment time diagram. In this example, the gap and the dynamic programming points to act III, scene 1; in which the Prince reduces Romeo's death sentence to banishment, thus sealing his and Juliet's fates, a critical event. The minimum gap method points to act III, scene 1; which is close to the dynamic programming result. This scene is the turning point of the entire play from comedy to tragedy. The maximum

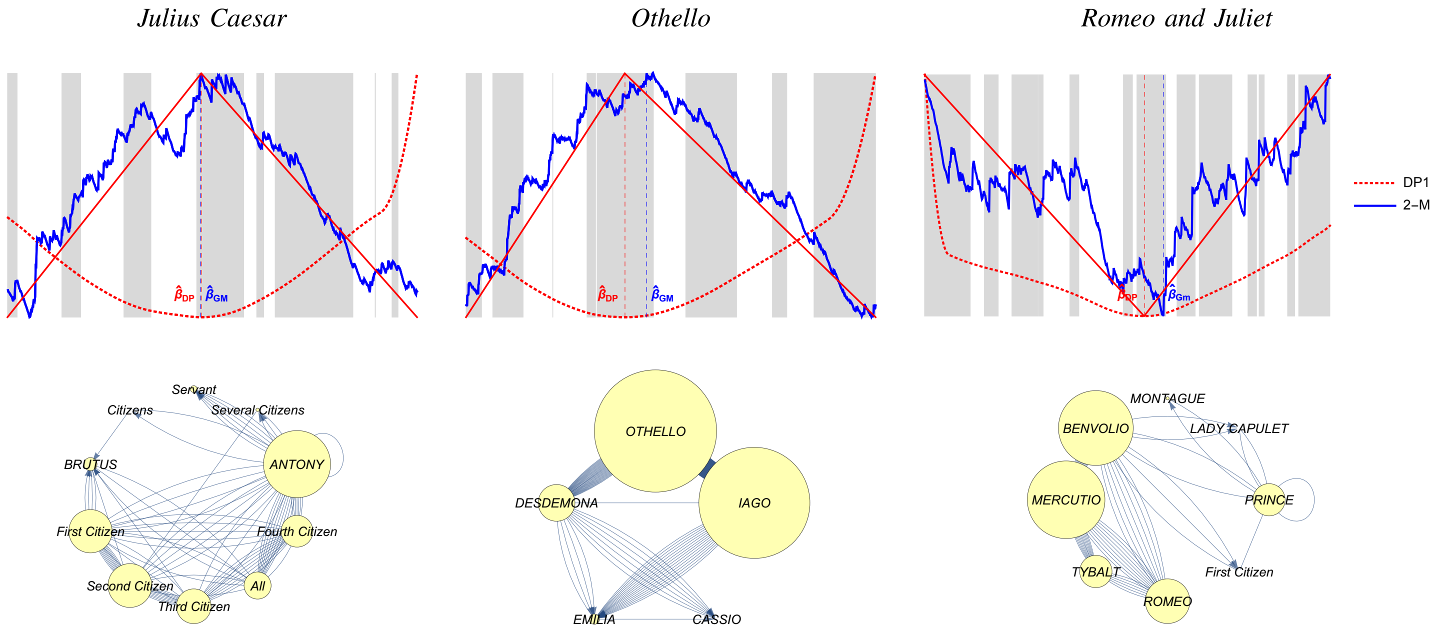

Figure 2: The first row is the results of the algorithms. The red lines is the dynamic programing, the blue lines is the second moment diagram. The two vertical dash-lines represent the estimation of the beginning of the drift: red for dynamic programing, blue for the gap. The gray/white areas mark the odd/even scenes. The second row is the critical scene's social networks: left is scene III.2 of *Julius Caesar*, middle is scene III.3 in *Othello*, right is scene III.1 in *Romeo and Juliet*.

gap points to act V, scene 3; where the Prince concludes the terrible deaths of Romeo and Juliet.

Again, CliffsNotes supports the results of the algorithms in *Romeo and Juliet*, as stated by CliffsNotes: ”The play reaches a dramatic crescendo as Romeo and Juliets private world clashes with the public feud with tragic consequences. Mercutios death is the catalyst for the tragic turn the play takes from this point onward”. See [8] page 51.

7. Conclusion

The main motivation for this paper was to develop tools that can identify critical events in an evolving social network. Ideas from psychology on time perception in humans, especially those of Karl Ernst Von Baer were used, in order to develop time deriving algorithms that point to critical time in the evolution of a social network. Two clocks were generated from a weighted evolving graph: the event clock and the weighted clock. The drift between the clocks was used to identify a critical time in the weighted social network. This follows the paradigm of the seminal work by Von Baer, who showed the psychological clock drift happened at life threatening times.

The dynamic programming and the gap algorithms were tested on plays by William Shakespeare, and both coincide with each other, succeeding in finding the critical points in all the examined plays, whenever there is a single clock drift, as the theorem showed. This was compared to CliffsNotes, which agreed with the critical point in the plays found by the algorithms. From a computational point of view, both algorithms are linear, however, the gap is much simpler to implement then the dynamic programming.

Perhaps somewhat surprisingly, the algorithm uses very little information. It only uses who is speaking, and how many words they say. This can be explained in the spirit of Albert Mehrabian, who's study on verbal and non-verbal communication which is known as the *7%-38%-55%* rule of thumb see [15] . Apparently, there is a lot of information found in the structure of who is speaking and how much.

One idea that was proven to be powerful is the study of two different clocks, within the context of social networks. The event and the weighted clocks are universal in weighted evolving social networks. And thus, the algorithms developed in this paper have many applications. The idea of multi-clocks is a novel one, but much more work is needed in order to bring it to its full potential.

References

[1] Réka Albert and Albert-László Barabási. Topology of evolving networks: local events and universality. *Physical review letters*, 85(24):5234, 2000.

[2] Boris Aronov, Tetsuo Asano, Naoki Katoh, Kurt Mehlhorn, and Takeshi Tokuyama. Polyline fitting of planar points under min-sum criteria. *Int. J. Comput. Geometry Appl.*, 16(2-3):97–116, 2006.

[3] Chen Avin, Michal Koucký, and Zvi Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 121–132, 2008.

[4] Richard A Block and Simon Grondin. Timing and time perception: A selective review and commentary on recent reviews. *Frontiers in psychology*, 5, 2014.

[5] Dan Braha and Yaneer Bar-Yam. Time-dependent complex networks: Dynamic centrality, dynamic motifs, and cycles of social interactions. In *Adaptive Networks*, pages 39–50. Springer, 2009.

[6] Gary K Carey and Helen McCulloch. *CliffsNotes on Shakespeares Othello*. Houghton Mifflin Harcourt, 2002.

[7] Andrea E. F. Clementi, Claudio Macci, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Flooding time in edge-markovian dynamic graphs. In *Proceedings of the Twenty-Seventh Annual ACM, PODC 2008, Toronto, Canada, August 18-21, 2008*, pages 213–222, 2008.

[8] Annaliese F Connolly. *CliffsNotes on Shakespeare's Romeo and Juliet (Cliffsnotes Literature)*. Houghton Mifflin Harcourt, 2000.

[9] Patrick Doreian and Frans N Stokman. The dynamics and evolution of social networks. *Evolution of social networks*, pages 1–17, 1997.

[10] Martha Perry James E Vickers. *CliffsNotes on Shakespeare's Julius Caesar (Cliffsnotes Literature Guides)*. Houghton Mifflin Harcourt, 2000.

[11] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, 1978.

[12] Wei Liu, Andrey Kan, Jeffrey Chan, James Bailey, Christopher Leckie, Jian Pei, and Ramamohanarao Kotagiri. On compressing weighted time-evolving graphs. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 2319–2322. ACM, 2012.

[13] Zvi Lotker. Voting algorithm in the play julius caesar. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 848–855, 2015.

[14] Clémence Magnien and Fabien Tarissan. Time evolution of the importance of nodes in dynamic networks. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, pages 1200–1207. ACM, 2015.

[15] Albert Mehrabian. *Nonverbal communication*. Transaction Publishers, 1972.

[16] Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. Graph metrics for temporal networks. In *Temporal Networks*, pages 15–40. Springer, 2013.

[17] Jeff Rydberg-Cox. Social networks and the language of greek tragedy. In *Journal of the Chicago Colloquium on Digital Humanities and Computer Science*, volume 1, 2011.

[18] James Stiller, Daniel Nettle, and Robin IM Dunbar. The small world of shakespeares plays. *Human Nature*, 14(4):397–408, 2003.

[19] Bharath Sundararaman, Ugo Buy, and Ajay D Kshemkalyani. Clock synchronization for wireless sensor networks: a survey. *Ad Hoc Networks*, 3(3):281–323, 2005.

[20] Yik-Chung Wu, Qasim Chaudhari, and Erchin Serpedin. Clock synchronization of wireless sensor networks. *Signal Processing Magazine, IEEE*, 28(1):124–138, 2011.