Version of Record: https://www.sciencedirect.com/science/article/pii/S0304397519301914
Manuscript_272e097de770a8a894741deac091708f

Temporal Matching ⋆

Julien Baste [a] Binh-Minh Bui-Xuan [a] Antoine Roux [a,b]

[a] *Laboratoire d'Informatique de Paris 6 (LIP6), Centre National de la Recherche Scientifique (CNRS), Sorbonne Université (SU UPMC).*

[b] *Thales Communications & Security, Thales Group.*

[julien.baste,buixuan,antoine.roux]@lip6.fr

Abstract

A link stream is a sequence of pairs of the form $(t, \{u, v\})$, where $t \in \mathbb{N}$ represents a time instant and $u \neq v$. Given an integer γ, the γ-edge between vertices u and v, starting at time t, is the set of temporally consecutive edges defined by $\{(t', \{u, v\}) \mid t' \in [\![t, t + \gamma - 1]\!]\}$. We introduce the notion of temporal matching of a link stream to be an independent γ-edge set belonging to the link stream. We show that the problem of computing a temporal matching of maximum size is NP-hard as soon as $\gamma > 1$. We depict a kernelization algorithm parameterized by the solution size for the problem. As a byproduct we also give a 2-approximation algorithm.

Both our 2-approximation and kernelization algorithms are implemented and confronted to link streams collected from real world graph data. We observe that finding temporal matchings is a sensitive question when mining our data from such a perspective as: managing peer-working when any pair of peers X and Y are to collaborate over a period of one month, at an average rate of at least two email exchanges every week. We furthermore design a link stream generating process by mimicking the behaviour of a random moving group of particles under natural simulation, and confront our algorithms to these generated instances of link streams. All the implementations are open source.

Key words: graph, parameterized algorithm, link stream, open source code

⋆ Part of the results reported in this paper were presented at *CTW'18*. Links to the source code and the GUI of the link stream generator:
`https://github.com/antoinedimitriroux/Temporal-Matching-in-Link-Streams`
`https://antoinedimitriroux.github.io`
For financial support, we are grateful to: *Thales Communications & Security*, project TCS.DJ.2015-432; *Agence Nationale de la Recherche Technique*, project 2016.0097; *Centre National de la Recherche Scientifique*, project INS2I.GraphGPU.

© 2019 published by Elsevier. This manuscript is made available under the CC BY NC user license
https://creativecommons.org/licenses/by-nc/4.0/

1 Introduction

The problem of finding a maximum matching is a fundamental and well studied question. It consists in finding a maximum independent edge set of a given graph. It can be solved in polynomial time by the well known Edmonds algorithm [8]. On the theoretical side, Edmonds result plays a primary role in combinatorial optimization. This is not only because it made a major historical impact in pointing out the polytope structure of a graph problem, but also because this result has marked the beginning of a long and fruitful list of matching algorithms all having polynomial worst case time complexity. We can cite in this sense the connection between MATCHING and matrix multiplication which was exploited for algorithm design [4,16].

A lot of research effort has been put in investigating variants of MATCHING as well. For instance, a tricky structural analysis helps in devising a linear time procedure for finding popular matchings [1]. Albeit it must be under some conditions called fairness, the fact that a linear time algorithm exists for this kind of popular matching helps in better understanding the underlying discrete structure of matchings. Surprisingly, while being a well-known and fundamentally polynomial algorithmic problem, MATCHING has lately attracted research interest in the parameterized areas of algorithmic as well [9,12]. Here, the overall effort has been put in reducing the polynomial time complexity to linear time, by means of factorising bits of the time complexity to depend on another parameter of the input instance rather than its size.

On the practical side, MATCHING is a convenient formalism to approach task management problems. For instance, in a bipartite graph where one vertex set represents chores and the other vertex set represents executors, each having the ability to execute a (different) subset of chores, MATCHING models the question of maximising the number of chores that can be executed. This problem has been intensively investigated under the setting of streaming inputs, where unpredictable arrivals of executors must be affected to chores in real time, see e.g. [14,19]. In this topic, a careful randomized study [7] not only provides a streaming algorithm achieving competitive approximation ratios, but it also gives the upper bound of extra information the streaming algorithm requires, and, particularly, a tricky proof of a lower bound of extra information one need to use in order to achieve the previously said approximation ratio. More generally, in an arbitrary graph representing compatible coworkers, MATCHING models the concern of maximising the overall workload when work must be done by compatible pairs. This problem has recently been investigated from a heuristics point of view [6], as well as under a quantitative comparison of the greedy approach on large input [20].

From the perspective of mining data collected from human activities, how-

ever, the input graph should be taken under the light of the time dimension: graph edges are time stamped edges. They come ordered by the time instants where they are recorded. We call this kind of data a link stream, in the sense of [11,18]. The most natural illustration of such thing is web logs, where any single line of log includes a field under time format. Phone calls between individuals are also time dependent information, so are email exchanges. Other kind of time dependent interaction could arise from peer programming management in IT best practices too. For instance, let us consider a human resource platform where collaborators register for the coming trimester the time intervals where they are unavailable for work. For simplicity we discretize these time intervals by days, from 1 to 90. Besides, the collaborators also communicate via keywords the hard skills in which they are efficient. Let us consider that a task must be processed by two skilled collaborators over at least $\gamma = 5$ five consecutive days before delivery. By human limitations, a collaborator will only process a given maximum number of tasks at a time. Under these conditions, given a potentially infinite number of tasks to process, how many deliveries can be made for the coming trimester? How continuously can delivery be? How dense can peer working be versus individualistic task processing? Fundamentally, *how to quantify the concession in term of continuous delivery in favoring peer programming over individualistic behaviour?* While not fully answering to these questions, we propose to make one step toward this kind of reflection by formalizing the notion of timed collaboration, and show how to compute it.

A link stream L is a triple $L = (T, V, E)$ where E is a sequence of pairs of the form $(t, \{u, v\})$, with $\{u, v\} \in \binom{V}{2}$ being an edge in the sense of classical loopless undirected simple graphs, and $t \in T \subseteq \mathbb{N}$ an integer representing a discretized time instant. If every pair $(t, \{u, v\})$ in L satisfies $t = t_0$ for some fixed t_0, then we say that link stream L is a graph. Given an integer γ, a time instant t, and two distinct vertices u and v, we define the γ-edge between u and v starting at time t as the set $\{(t', \{u, v\}) \mid t' \in [\![t, t + \gamma - 1]\!]\}$. A temporal vertex is a pair (t, u), representing vertex $u \in V$ at time $t \in T$. We say that a γ-edge Γ contains a temporal vertex (t, u) if there exists a vertex $v \in V$ such that $(t, \{u, v\}) \in \Gamma$. Two γ-edges are independent if there is no temporal vertex that is contained in both of them. Finally, a γ-matching of link stream L is a set of pairwise independent γ-edges where each γ-edge contains exclusively edges from L. We define γ-MATCHING as the problem of computing a maximum γ-matching from a given input link stream. We believe that problems involving γ-edges with $\gamma = 1$ somewhat are rooted in classical graph theory, whereas γ-edges for $\gamma > 1$ intrinsically model temporal interactions. For instance, when $\gamma = 1$, this problem can be solved by a slight extension of previously mentioned Edmonds algorithm [8]. In recent studies under a temporal perspective, the problem when $\gamma = 1$ has also been considered along with additional conditions in the computed γ-matching, making it NP-hard [2,13]. In this paper, we focus on γ-edges with a non-trivial duration, that is, when $\gamma > 1$.

Unfortunately, γ-MATCHING turns out to be NP-hard for $\gamma > 1$. We subsequently address the question of pre-processing, in polynomial time, an input instance of γ-MATCHING, in order to reduce it to an equivalent instance of smaller size, in the sense of kernelization algorithms introduced in [5]. We show that γ-MATCHING when parameterized by the solution size admits a quadratic kernel. On the way to do so, we also point out a simple way to produce a 2-approximation algorithm for γ-MATCHING.

We try to comprehend our result from a practical point of view. From this perspective we design a link stream generating process [1] by mimicking the behaviour of a random moving group of particles, using natural simulation: velocity, friction, and random walk. The generating process helps us in unit testing our implementations on small generated inputs, as well as in stress testing our implementations on large inputs mimicking natural movements.

Both our 2-approximation and kernelization algorithms are implemented [2] and confronted, not only to our generated link streams, but also to two particular sets of link streams collected from real world graph data. These raw datasets are first cleaned by a procedure that we call time-compression, and argue the need for it right below. In one dataset the link streams have been built by time-compression over exchanges collected from the Enron emailing network [10]. The other dataset is built by time-compression over a recording of 2×80 minute Rollerblade touring in Paris [17].

The reason for us to time-compress the raw data is because, therein, the (machine recorded) consecutive time stamps can happen quite instantaneously for human standards. This leaves no chance for a γ-edge to exist, as soon as $\gamma > 1$. For instance with the Enron emails, by ISO8601 time stamps are discretized down to the order of seconds. However, there is absolutely no chance for that individuals X and Y exchange two different emails in any two consecutive seconds in the whole duration of the experiment: there is simply no time to read the first email and type a reply the following second. For Enron we usually merge up the time stamps to the order of half a week: if an email is received, read, thought over, eventually replied within 3.5 days, then we consider there is collaboration within that period of time. From this perspective, we compress our raw data by edge contraction over the time dimension, formally as follows. For any link stream $L = (T, V, E)$ and $1 < \delta < |T|$, we define the *δ-compression* $L_\delta = (T_\delta, V_\delta, E_\delta)$ as $V_\delta = V$, $T_\delta = [\![\frac{\min T}{\delta}, \frac{\max T}{\delta}]\!]$, and

$$E_\delta = \{(t, \{u, v\}) \mid \exists t' \in T : \delta t \leq t' < \delta(t+1) \wedge (t', \{u, v\}) \in E\}.$$

[1] Direct link to the GUI of the generator:
`https://antoinedimitriroux.github.io`
[2] The source code is available at
`https://github.com/antoinedimitriroux/Temporal-Matching-in-Link-Streams`

After running our implementation on the two datasets, we make three observations. First, we believe that solving γ-MATCHING is not easy as soon as we time-compress the raw data with "human-understandable" values of δ and γ. Second, on small values of γ, we observe that the kernelization algorithm helps in reducing the input link stream to an equivalent instance of size approximatively $10 - 20\%$ the size of the original input. This gives measurable evidence of performance for our preprocessing by kernelization. Our third observation is very marginal. Note beforehand from definition that the 2-approximation algorithm produces a lower bound for γ-MATCHING, which is at least half the optimal value. Moreover, note also that the kernelization algorithm gives a naive upper bound for γ-MATCHING by simply counting the number of γ-edges present in the kernel. Our third observation from the numerical analysis is that these upper bound and lower bound nearly meet on some areas in the datasets. Even though they remain extremely marginal, these cases point out that, sometimes, a kernelization algorithm can also provide a numerical proof of optimality of the result found by a (greedy) 2-approximation. This hints at the usefulness beyond theoretical considerations of γ-MATCHING kernelization. Moreover, our kernelization runtime is under ten seconds for inputs where the input size is some hundreds thousand and the parameter is some thousands. Fixed parameter tractable (FPT) paradigm in general, and kernelization in particular, would never be numerically helpful if the complexity analysis hides a big function of the parameter in the Landau notation. Luckily, we use simple algorithmic processes for our kernelization.

The paper is organised as follows. We first introduce the notion of temporal matching in Section 2. In Section 3, we present our algorithmic tools in order to obtain our main result, the kernelization algorithm: it is presented in Section 4. In Section 5, we present our numerical analysis. We close the paper with concluding remarks and directions for further research.

2 Temporal matching

Unless otherwise stated, graphs in this paper are simple, undirected and loopless graphs. We denote by $\mathbb{N}$ the set of non negative integer. Given two integers p and q, we denote by $[\![p, q]\!]$ the set $\{r \in \mathbb{N} \mid p \leq r \leq q\}$. A *link stream* L is a triple (T, V, E) such that $T \subseteq \mathbb{N}$ is an interval, V is a set, and $E \subseteq T \times \binom{V}{2}$. The link stream can be seen as an extension of graphs. Indeed, a graph is a link stream where $|T| = 1$. The elements of V are called *vertices* and the elements of E are called *(timed) edges*. A *temporal vertex* of L is a pair (t, u) such that $t \in T$ and $u \in V$.

Given an integer γ, a γ-edge between two vertices u and v at time t, denoted $\Gamma_\gamma(t, u, v)$, is the set $\{(t', \{u, v\}) \mid t' \in [\![t, t + \gamma - 1]\!]\}$. We say that a γ-edge

Γ contains a temporal vertex (t, u) if there exists a vertex $v \in V$ such that $(t, \{u, v\}) \in \Gamma$. We say that two γ-edges are independent if there is no temporal vertex that is contained in both of them. A γ-matching $\mathcal{M}$ of a link stream L is a set of pairwise independent γ-edges. We say that a γ-edge Γ is incident with a vertex $u \in V$ if there exist a vertex $v \in V$ and an integer $t \in T$ such that $\Gamma = \Gamma_\gamma(t, u, v)$. We say that an edge $e \in E$ is in a γ-matching $\mathcal{M}$ if there exists $\Gamma \in \mathcal{M}$ such that $e \in \Gamma$.

We focus on the following problem.

γ-MATCHING
Input: A link stream L and an integer k.
Output: A γ-matching of L of size k or a correct answer that such a set does not exist.

Theorem 1 γ-MATCHING *is NP-hard for* $\gamma > 1$.

Proof: We prove the NP-completeness of the decision version of γ-MATCHING by a reduction from 3-SAT, that is well known to be NP-complete. Let φ be a formula with n variables $x_1, \ldots, x_n$ and m clauses $C_0, \ldots, C_{m-1}$ such that each clauses is of size at most 3. Without loss of generality, we assume that a clause does not contain twice the same variable. We call X the set containing the n variables and $\mathcal{C}$ the set containing the m clauses.

We define the link stream $L = (T, V, E)$ in the following way:

- $T = [\![0, (m+1)\gamma - 1]\!]$.
- $V = \{x^-, x^=, x^+ \mid x \in X\} \cup \{x_t^{++}, x_t^{--} \mid x \in X, t \in [\![0, m-1]\!]\} \cup \{c\}$
- $E = E_{var} \cup E_{cla}$ where:

$$\begin{aligned}
E_{var} = &\{(t, \{x^=, x^+\}), (t, \{x^=, x^-\}) \mid t \in [\![0, (m+1)\gamma - 1]\!], x \in X\} \\
&\cup \{(t, \{x^+, x_i^{++}\}) \mid t \in [\![i\gamma + 1, (i+1)\gamma]\!], i \in [\![0, m-1]\!], x \in X\} \\
&\cup \{(t, \{x^-, x_i^{--}\}) \mid t \in [\![i\gamma + 1, (i+1)\gamma]\!], i \in [\![0, m-1]\!], x \in X\} \\
E_{cla} = &\{(t, \{c, x_i^{++}\}) \mid t \in [\![i\gamma + 1, (i+1)\gamma]\!], i \in [\![0, m-1]\!], x \in X, \\
&\qquad x \text{ appears positively in } C_i\} \\
&\cup \{(t, \{c, x_i^{--}\}) \mid t \in [\![i\gamma + 1, (i+1)\gamma]\!], i \in [\![0, m-1]\!], x \in X, \\
&\qquad x \text{ appears negatively in } C_i\}.
\end{aligned}$$

We depict in Figure 1 the link stream build for $\gamma = 3$ and $\varphi = (w \vee \overline{x} \vee y) \wedge (w \vee x \vee \overline{z})$.

We show that there is an assignments of the variables that satisfies φ if and only if L contains a γ-matching of size $(2m+1)n + m$.

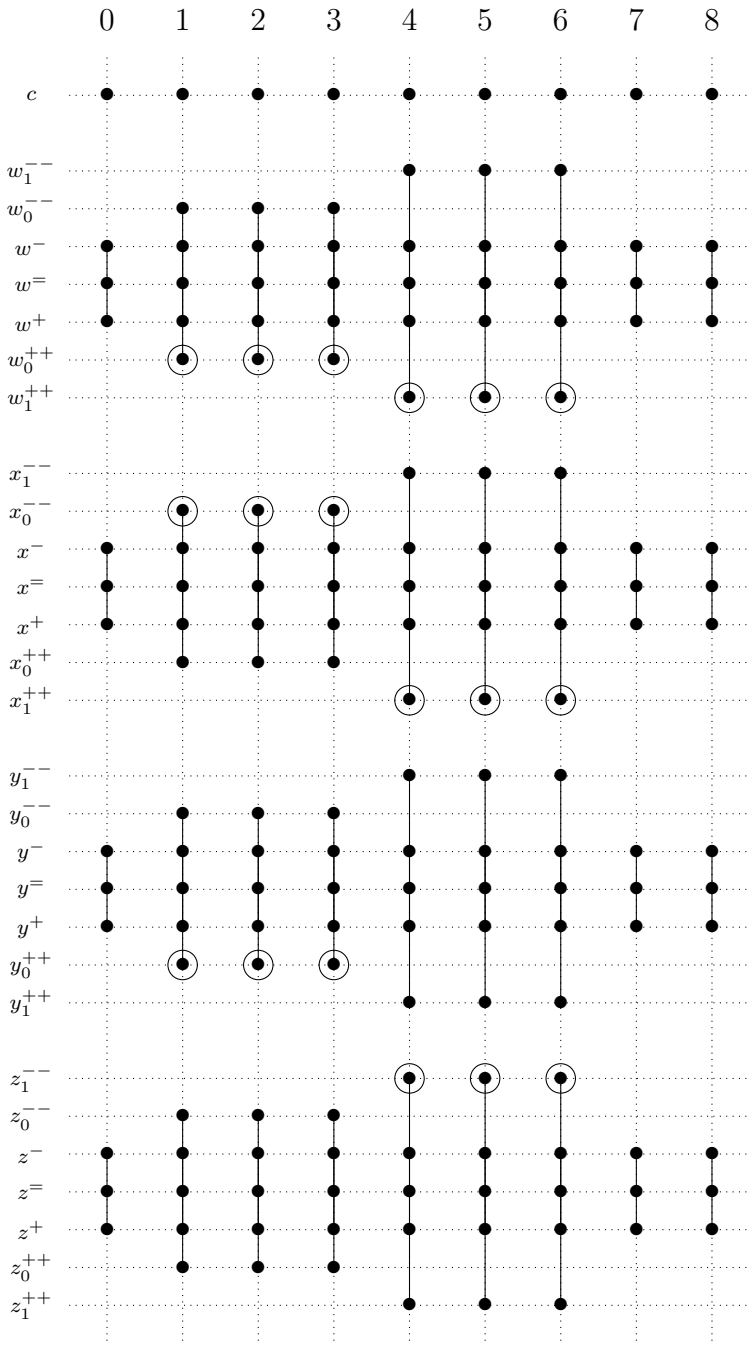


Fig. 1. The constructed linkstream L when $\varphi = (w \vee \overline{x} \vee y) \wedge (w \vee x \vee \overline{z})$ and $\gamma = 3$. Here $T = [\![0, 8]\!]$ and the edge $(t, \{u, v\})$ of L are depicted by an edge following the vertical line corresponding to time t going from the horizontal line corresponding to u to the horizontal line corresponding to v. For readability, the edges incident with c are not drown. Instead, we have circled the vertices that are neighbors of c at each specific time.

Intuitively, the edge between $(0, \{x^=, x^+\})$ and $(0, \{x^=, x^-\})$ that is in the requested γ-matching determines if the variable x is set to true or false. Moreover, the size of the requested γ-matching will ensure that if the edge $(0, \{x^=, x^+\})$ (resp. $(0, \{x^=, x^-\})$) is in the γ-matching, then every edge $(t, \{x^=, x^+\})$ (resp. $(t, \{x^=, x^-\})$), $t \in [\![0, (m+1)\gamma - 1]\!]$ and every edge $(t, \{x^-, x_i^{--}\})$ (resp. $(t, \{x^+, x_i^{++}\})$), $t \in [\![1, m\gamma]\!]\}$, $i = \left\lfloor \frac{t-1}{\gamma} \right\rfloor$, are in the γ-matching as well. Finally, during the time interval $[\![i\gamma + 1, (i+1)\gamma]\!]$, we will certify that the clause C_i is satisfied.

First assume that φ is satisfiable. Let ψ be a satisfying assignment of φ and let $\chi : \mathcal{C} \to V$ be a function that, for each clause C_i, $i \in [\![0, m-1]\!]$, arbitrary chooses a variable $x \in X$, such that the assignment of x given by ψ satisfies C_i, and returns x_i^{++} (resp. x_i^{--}) if $\psi(x) = \texttt{true}$ (resp. $\psi(x) = \texttt{false}$). Let

$$\begin{aligned}
\mathcal{M} = &\{\Gamma_\gamma(i \cdot \gamma, x^=, x^+) \mid x \in X, \psi(x) = \texttt{true}, i \in [\![0, m]\!]\} \\
&\cup \{\Gamma_\gamma(i \cdot \gamma, x^=, x^-) \mid x \in X, \psi(x) = \texttt{false}, i \in [\![0, m]\!]\} \\
&\cup \{\Gamma_\gamma(i \cdot \gamma + 1, x^-, x_i^{--}) \mid x \in X, \psi(x) = \texttt{true}, i \in [\![1, m]\!]\} \\
&\cup \{\Gamma_\gamma(i \cdot \gamma + 1, x^+, x_i^{++}) \mid x \in X, \psi(x) = \texttt{false}, i \in [\![1, m]\!]\} \\
&\cup \{\Gamma_\gamma(i \cdot \gamma + 1, c, \chi(C_t)) \mid i \in [\![1, m]\!]\}.
\end{aligned}$$

One can verify that $\mathcal{M}$ is a γ-matching of L of size $(2m+1)n + m$.

Assume now that L contains a γ-matching $\mathcal{M}$ of size $(2m+1)n + m$. We use several claim in order to construct a satisfying assignment of φ.

Claim 1 *For each $x \in X$, $\mathcal{M}$ contains at most $m+1$ γ-edges incident with x^+ (resp. x^-).*

Proof: This result follows from the fact that $T = [\![0, (m+1)\gamma - 1]\!]$ is of size $(m+1)\gamma$ and so, cannot be divided into $m+2$ pairwise disjoint sets of size γ. □

Claim 2 *Given $x \in X$, if $\Gamma_\gamma(0, x^=, x^+) \notin \mathcal{M}$ (resp. $\Gamma_\gamma(0, x^=, x^-) \notin \mathcal{M}$), then $\mathcal{M}$ contains at most m γ-edges incident with x^+ (resp. x^-).*

Proof: As $\Gamma_\gamma(0, x^=, x^+)$ is the only γ-edge of L that contains the edge $e_x^0 = (0, \{x^=, x^+\})$, this implies that the edge e_x^0 is not contained in any γ-edge of $\mathcal{M}$. So the γ-edge of $\mathcal{M}$ that are incident with x^+ are constraint to exist in the time interval $I = [\![1, (m+1)\gamma - 1]\!]$ that is of size $(m+1)\gamma - 1$. Thus, I cannot be divided in $m+1$ pairwise disjoint sets of size γ. The claim follows. □

Claim 3 *$\mathcal{M}$ contains exactly m γ-edges incident with c and contains exactly*

$2m+1$ *γ-edges incident with x^+ or x^-, for each $x \in X$.*

Proof: As $\mathcal{M}$ is a γ-matching, then for each $x \in X$, $\Gamma_\gamma(0, x^=, x^+) \notin \mathcal{M}$ or $\Gamma_\gamma(0, x^=, x^-) \notin \mathcal{M}$. So Claim 1 and Claim 2 implie that for each $x \in X$, $\mathcal{M}$ contains at most $2m+1$ γ-edges incident with x^+ or x^-. Moreover, by construction $\mathcal{M}$ can contains at most m γ-edges incident with c and L does not contains any edge of the form $(t, \{x^+, y^-\})$, $(t, \{x^+, y^+\})$, $(t, \{x^-, y^-\})$, $(t, \{x^+, c\})$, or $(t, \{x^-, c\})$, for any $x, y \in X$. Thus the budget is tight. The claim follows. □

Note that by construction, if $\mathcal{M}$ contains a γ-edge incident with x_i^{++} for some $x \in X$ and $i \in [\![0, m-1]\!]$, then this γ-edge has to be either $\Gamma_\gamma(i\gamma+1, c, x_i^{++})$ or $\Gamma_\gamma(i\gamma+1, x^+, x_i^{++})$. Moreover Claim 4 give us some information in the case where $\Gamma_\gamma(i\gamma+1, x^+, x_i^{++}) \in \mathcal{M}$

Claim 4 *Given $x \in X$, if $\mathcal{M}$ contains a γ-edge $\Gamma_\gamma(i\gamma+1, x^+, x_i^{++})$ (resp. $\Gamma_\gamma(i\gamma+1, x^-, x_i^{--})$) for some $i \in [\![0, m-1]\!]$, then $\mathcal{M}$ contains at most m γ-edges incident with x^+ (resp. x^-).*

Proof: Let $x \in X$ and let i be the first value such that $\Gamma_\gamma(i\gamma+1, x^+, x_i^{++}) \in \mathcal{M}$. As γ does not divide $i\gamma+1$, this implies that, in the interval $[\![0, i\gamma]\!]$, at least one edge $e_x^t = (t, \{x^=, x^+\})$, $t \in [\![0, i\gamma]\!]$ is not in $\mathcal{M}$. So the γ-edges of $\mathcal{M}$ that are incident with x^+ are constraint to exist in the time interval $I = T \setminus t$ that is of size $(m+1)\gamma - 1$. Thus, I cannot be divided in $m+1$ pairwise disjoint sets of size γ. The claim follows. □

Let $x \in X$. Using Claim 3, we know that $\mathcal{M}$ contains exactly $2m+1$ γ-edges incident with x^+ or x^-. By the pigeonhole principle, we know that for x^+ or x^-, say x^+, $\mathcal{M}$ contains exactly $m+1$ γ-edges incident with x^+. By Claim 4, this implies that $\{\Gamma_\gamma(i\gamma, x^=, x^+) \mid i \in [\![0, m]\!]\} \subseteq \mathcal{M}$. Thus, as $\mathcal{M}$ is a γ-matching that contains m γ-edges incident with x^-, this also implies that $\{\Gamma_\gamma(i\gamma+1, x^-, x_i^{--}) \mid i \in [\![0, m-1]\!]\} \subseteq \mathcal{M}$.

For each variable $x \in X$, we set x to `true` (resp. `false`) if $\Gamma_\gamma(0, x^=, x^+) \in \mathcal{M}$ (resp. $\Gamma_\gamma(0, x^=, x^-) \in \mathcal{M}$). Let φ be the so obtained assignment. Let $i \in [\![0, m-1]\!]$. We know that there exists $x \in X$ such that either $\Gamma_\gamma(i\gamma+1, c, x_i^{++}) \in \mathcal{M}$ or $\Gamma_\gamma(i\gamma+1, c, x_i^{--}) \in \mathcal{M}$. Let fix this $x \in X$ and assume that $\Gamma_\gamma(i\gamma+1, c, x_i^{++}) \in \mathcal{M}$, meaning that x appears positively in C_i. This implies that $\Gamma_\gamma(i\gamma+1, x^+, x^{++}) \notin \mathcal{M}$, so that $\{\Gamma_\gamma(i'\gamma, x^=, x^+) \mid i' \in [\![0, m]\!]\} \subseteq \mathcal{M}$. Thus, x is set to `true` by φ and x satisfies C_i. This concludes the proof. □

3 Approximation algorithm

In classical graph theory, it is folklore that any maximal matching is also a 2-approximation of a maximum matching, see e.g. [3, Exercice 35.4]. Fortunately enough, it is roughly the same situation with link streams. Precisely, in this section, we adopt the greedy approach –finding a maximal γ-matching– in order to provide a 2−approximation algorithm for γ-MATCHING.

Let $L = (T, V, E)$ be a link stream. Let $\mathcal{P}$ be the set of all γ-edges of L. Note that these γ-edges are not independent from each other, on the contrary, they highly overlap. Let $\preceq$ be an arbitrary total ordering on the elements of $\mathcal{P}$ such that given for any two elements of $\mathcal{P}$, $\Gamma_1 = \Gamma_\gamma(t_1, u_1, v_1)$ and $\Gamma_2 = \Gamma_\gamma(t_2, u_2, v_2)$ such that $t_1 < t_2$, we have $\Gamma_1 \preceq \Gamma_2$.

We denote by $\mathcal{A}$ the following greedy algorithm. The algorithm starts with $\mathcal{M} = \emptyset$, $\mathcal{Q} = \mathcal{P}$, and a function $\rho : V \times T \to \{0, 1\}$ such that for each $(t, v) \in T \times V$, $\rho(t, v) = 0$. The purpose of ρ is to keep track of the temporal vertices that are contained in a γ-edge of $\mathcal{M}$. As long as $\mathcal{Q}$ is not empty, the algorithm selects Γ, the γ-edge of $\mathcal{Q}$ that is minimum for $\preceq$, and removes it from $\mathcal{Q}$. Let K be the set of the 2γ temporal vertices that are contained in Γ. If, for each $(t, v) \in K$, $\rho(t, v) = 0$, then the algorithm adds Γ to $\mathcal{M}$, otherwise it does nothing at this step. For each $(t, v) \in K$, it sets $\rho(t, v)$ to 1 and repeats. If $\mathcal{Q} = \emptyset$, it returns $\mathcal{M}$.

As $\mathcal{P}$ can be determined in a sorted way in time $\mathcal{O}(m)$, this algorithm runs in time $\mathcal{O}(n\tau + m)$, where $\tau = |T|$, $n = |V|$, $m = |E|$, and where γ is a constant hidden in the $\mathcal{O}$.

Given a γ-matching $\mathcal{M}$, we define the *bottom temporal vertices* of $\mathcal{M}$, denoted by $\texttt{bot}(\mathcal{M})$, as the set $\{(t+\gamma-1, u), (t+\gamma-1, v) \mid \Gamma_\gamma(t, u, v) \in \mathcal{M}\}$. Lemma 1 shows the crucial role of the bottom temporal vertices of the matchings returned by $\mathcal{A}$.

Lemma 1 *Let γ be a positive integer, let L be a link stream, and let $\mathcal{M}$ be a γ-matching returned by $\mathcal{A}$ when applied to L. If $\mathcal{M}'$ is a γ-matching of L, then every γ-edge of $\mathcal{M}'$ contains, at least, one temporal vertex of $\texttt{bot}(\mathcal{M})$.*

Proof: First, note that any γ-edge of $\mathcal{M}$ contains two temporal vertices of $\texttt{bot}(\mathcal{M})$, and so, at least one. Let Γ' be a γ-edge of $\mathcal{M}'$ that is not in $\mathcal{M}$. Let $\mathcal{M}^* \subseteq \mathcal{M}$ be the set of every γ-edge Γ^* of $\mathcal{M}$ such that there exists a temporal vertex (t, u) that is contained in both Γ' and Γ^*. Assume that $\Gamma' = \Gamma_\gamma(t, u, v)$. If there exists $\Gamma^* \in \mathcal{M}$ such that $\Gamma^* = \Gamma_\gamma(t', u, v')$ and $t' \leq t$, then we have that $(t' + \gamma - 1, u) \in \texttt{bot}(\mathcal{M})$ is contained in Γ'. Otherwise, we have that for each $\Gamma^* \in \mathcal{M}^*$ such that $\Gamma^* = \Gamma_\gamma(t', u, v')$, $t' > t$. This is not possible by construction of $\mathcal{A}$. This concludes the proof. □

Lemma 1 plays a cornerstone role in the proof of subsequent Theorem 2. As a byproduct, we also obtain the following result.

Corollary 1 *$\mathcal{A}$ is a 2-approximation of the γ-MATCHING problem.*

Proof: Let L be the input link stream. Let $\mathcal{M}$ be a solution returned by the algorithm $\mathcal{A}$ when applied to L, and let $\mathcal{M}'$ be a γ-matching of L. As $|\texttt{bot}(\mathcal{M})| = 2|\mathcal{M}|$, two γ-edges of $\mathcal{M}'$ cannot contains the same temporal vertex, and, by Lemma 1, every γ-edge of $\mathcal{M}'$ contains at least one element of $\texttt{bot}(\mathcal{M})$, we obtained that $|\mathcal{M}'| \leq 2|\mathcal{M}|$. □

4 Kernelization algorithm

Problem γ-MATCHING has a *kernel* if there exist a computable function $f : \mathbb{N} \to \mathbb{N}$ and a polynomial time algorithm $\mathcal{A}$ which takes as input an instance (L, k) of γ-MATCHING and produces an instance (L', k') such that: $k' \leq k$; $|L'| \leq f(k)$; and (L', k') yields a positive answer for γ-MATCHING if and only if (L, k) yields a positive answer for γ-MATCHING. In this case, algorithm $\mathcal{A}$ is called a *kernelization algorithm* for γ-MATCHING [5].

We now show a kernelization algorithm for γ-MATCHING by a direct pruning process based on Lemma 1. For convenience, let us say that a γ-edge Γ is incident to a temporal vertex (t, u) when there exists vertex $v \neq u$ such that $(t, \{u, v\}) \in \Gamma$. The main idea is as follows. First, we compute the set S of all bottom temporal vertices of a γ-matching produced by previously defined algorithm $\mathcal{A}$. Then, we prune the original instance by only keeping edges that belong to a γ-edge incident to a temporal vertex of S. More precisely, we prove the following result.

Theorem 2 *There exists a polynomial-time algorithm that for each instance (L, k), either returns a true instance which correctly correspond to the fact that L contains a γ-matching of size k, or returns an equivalence instance (L', k) such that the number of edges of L' is $2(k-1)(2k-1)\gamma^2$.*

Proof: Let $L = (T, V, E)$ be a link stream and k be an integer. We first run the algorithm $\mathcal{A}$ on L. Let $\mathcal{M}$ be the γ-matching outputed by the algorithm and let $\ell = |\mathcal{M}|$. If $\ell \geq k$, then we already have a solution and then return a true instance. If $\ell < \frac{k}{2}$, then, by Corollary 1, we know that the instance does not contains a γ-matching of size k, and then we return a false instance. We now assume that $\frac{k}{2} \leq \ell < k$.

Lemma 1 justifies that we are now focusing on the temporal vertices of $\texttt{bot}(\mathcal{M})$ in order to find the requested kernel. We construct a set $\mathcal{P}$ of γ-edges and we

show that any edge e, that is not in a γ-edge of $\mathcal{P}$, is useless when looking for a γ-matching of size k. For each $(t,u) \in \texttt{bot}(\mathcal{M})$, and for each t' such that $\max(0, t-\gamma+1) \leq t' \leq t$, we consider the set $\mathcal{S}(t',u)$ of every γ-edge, existing in L, with the form $\Gamma_\gamma(t',u,v)$ with $v \in V$. If the set $\mathcal{S}(t',u)$ is of size at most $2k-1$, we add every element of $\mathcal{S}(t',u)$ to $\mathcal{P}$. Otherwise, we select $2k-1$ elements of $\mathcal{S}(t',u)$ that we add to $\mathcal{P}$. In both cases, we denote by $\mathcal{S}'(t',u)$ the set of elements of $\mathcal{S}(t',u)$ that we have added to $\mathcal{P}$. This finish the construction of $\mathcal{P}$. As $|\texttt{bot}(\mathcal{M})| = 2\ell$ and for each element of $\texttt{bot}(\mathcal{M})$ we have added at most $(2k-1)\gamma$ γ-edges to $\mathcal{P}$, we have that $|\mathcal{P}| \leq 2\ell(2k-1)\gamma \leq 2(k-1)(2k-1)\gamma$.

We now prove that if L contains a γ-matching $\mathcal{M}'$ of size k, then it also contains a γ-matching $\mathcal{M}''$ of size k such that $\mathcal{M}'' \subseteq \mathcal{P}$. Let $\mathcal{M}'$ be a γ-matching of L of size k such that $p = |\mathcal{M}' \setminus \mathcal{P}|$ is minimum. We have to prove that $p = 0$. Assume that $p \geq 1$. Let Γ be a γ-edge in $\mathcal{M}' \setminus \mathcal{P}$. Let (t,u) be a temporal vertex of $\texttt{bot}(\mathcal{M})$ that is contained in Γ. We know by Lemma 1 that this temporal vertex exists. Assume that $\Gamma = \Gamma_\gamma(t',u,v)$ for some $v \in V$ and some t' such that $\max(0, t-\gamma+1) \leq t' \leq t$. As $\Gamma \notin \mathcal{P}$, we have that $\Gamma \in \mathcal{S}(t',u) \setminus \mathcal{S}'(t',u)$, and so $|\mathcal{S}'(t',u)| = 2k-1$. Let $N_{\mathcal{S}'}(t',u)$ be the set of vertices w of $V \setminus \{u\}$ such that a γ-edge of $\mathcal{S}'(t',u)$ is incident to w. As $\mathcal{M}' \setminus \{\Gamma\}$ is of size $k-1$, the γ-edges that it contains can be incident to at most $2k-2$ vertices. This means that there exists $w \in N_{\mathcal{S}'}(t',u)$ such that no γ-edge of $\mathcal{M}' \setminus \{\Gamma\}$ is incident to w. Thus $(\mathcal{M}' \setminus \{\Gamma\}) \cup \{\Gamma_\gamma(t',u,w)\}$ is a γ-matching of size k. As $\Gamma \notin \mathcal{P}$ and $\Gamma_\gamma(t',u,v) \in \mathcal{P}$, this contradicts the fact that p is minimum.

We now can define the link stream $L' = (T, V, E')$ such that $E' = \{e \in E \mid \exists \Gamma \in \mathcal{P} : e \in \Gamma\}$. As $|\mathcal{P}| \leq 2(k-1)(2k-1)\gamma$ and every element of $\mathcal{P}$ is a γ-edge, we have that $|E'| \leq 2(k-1)(2k-1)\gamma^2$. The theorem follows. □

5 Experimental result

For easy diffusion, both our 2-approximation and kernelization algorithms are implemented in Java and JavaScript [3]. Experiments are run on a standard laptop clocking at 3,1 Ghz with DDR3 16Go memory.

[3] The source code is available at
`https://github.com/antoinedimitriroux/Temporal-Matching-in-Link-Streams`

5.1 Dataset

We carried out our experiments on two main types of datasets: those that are randomly generated [4]; and those that are collected from human activities.

Artificially generated link streams, and stress test:
In order to generate random sets of link stream instances, we adopt the more realistic point of view of random geometric graphs, rather than the classically theoretic Erdös-Rényi model, as follows. Let $\mathcal{S}$ be a 2D Euclidian space. We define a particle as a point in space $\mathcal{S}$. Every particle is given along with a radius representing the maximum communication distance it can have with another particle. Thus, the particle together with its range define a disk in space $\mathcal{S}$. Let $\mathcal{P}$ be a set of particles, given along with the same value of radius. We partition set $\mathcal{P}$ into n parts, $\mathcal{P} = P_1 \cup P_2 \cup \cdots \cup P_n$, of roughly equal size. We will construct a link stream $L = (T, V, E)$ as follows. Let $V = \{P_1, P_2, \ldots, P_n\}$. At time zero, let $E_0 = \{(0, \{P_i, P_j\}) \mid \exists a \in P_i \wedge b \in P_j,$ *the distance between a and b is less than their radius*$\}$. In other words, if there are at least two particles under communication range $a \in P_i, b \in P_j$ of different groups $P_i \neq P_j$ (at time zero), then, we consider there is a (zero-timed) edge between P_i and P_j. Every particle has a velocity that is defined as follows. First, the velocity of a particle at time t is a fraction of the velocity at time $t-1$ of that particle (friction). Second, all velocity vectors are subject to a small random additional factor (random walk factor modeling the wind condition). Finally, we truncate every velocity vector in order to insure that the norm of the vector is lower than a given maximum particle speed (physical limits). We then let the system evolve during a given laps of time, that we also refer to as T. At every time instant $t \in T$, we define, similarly as before, the t-timed edge set $E_t = \{(t, \{P_i, P_j\}) \mid \exists a \in P_i \wedge b \in P_j,$ *the distance between a and b is less than their radius*$\}$. Finally, we define our link stream as $E = \bigcup_{t \in T} E_t$.

Roughly, increasing any of the three parameters which are defined by the particle radius, the cardinality of $\mathcal{P}$, and the maximum particle speed, results in the same effect on the generated link stream, that is, to produce a dense link stream. The generated data allows us to unit test our code, and especially to verify that approximation and kernelization runtime is sound on large input. For instance, with inputs containing hundreds of thousand timed edges, our runtime is under ten seconds, cf. Fig. 2.

Interestingly, in the left chart of Fig. 2, we note for some input with a large number of timed edges and no γ-edges that the runtime can be very quick. We also discuss this phenomenon on real world dataset, in below Fig. 3 and 4. For

[4] Direct link to the GUI of the generator:
`https://antoinedimitriroux.github.io`

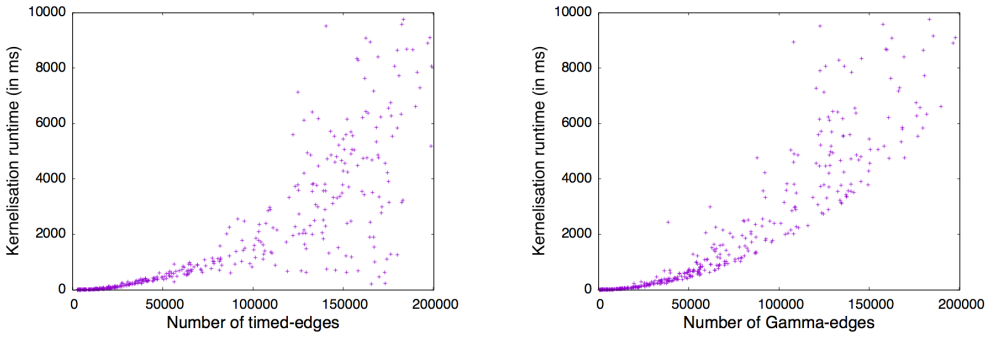


Fig. 2. Stress-test on generated input, with $\gamma = 5$. Though the number of γ-edges can be extremely high compared to timed edges (cf. overlapping γ-edges), we parametrised the generator so that they look the same in the left chart and the right chart. This is by no means a general property.

this reason, it is much more interesting to examine the runtime as a function of the number of γ-edges of the input, cf. the right chart in Fig. 2. For easy eye-comparison, we tried to parametrise the generator in a way that most of the generated instances have roughly the same number of timed edges and γ-edges. For instance, most instances with less than 100000 timed edges also have roughly the same number of γ-edges. However, the situation is more random for instances having between 100000 and 200000 timed edges.

Real world datasets, with cleaning methodology:
We also confront the implementations of our algorithms to two particular link streams collected from real world graph data. In one dataset the link stream is built from emailing information collected from the Enron company [10]. The other dataset has been built by analysing a recording of 2×80 minute Rollerblade touring in Paris [17]. Because of the long duration of these two experiments, the number of vertices (under two hundred persons in both cases) is negligible when comparing to the number of temporal vertices (nearly one million for Rollernet). For a link stream $L = (T, V, E)$, we mostly compare $|T|$ with $|E|$ to get a glimpse on the density of the links. For a complete view of $|T|$, $|V|$, and $|E|$ in the datasets, we refer the reader to Fig. 7. Furthermore, we noticed with our raw datasets that the instants where some timed edge is present can be very sparse, leaving no chance for a γ-edge to exist as soon as $\gamma > 1$. For instance, in the Enron experiment (Fig. 3), we can see that there are no pair of employees who keep sharing 1 mail per hour during 24 hours, probably due to inactivity at night. We will, for this reason, time-compress our raw datasets by the following process.

Definition 1 (Data cleaning by time-compression) For any link stream $L = (T, V, E)$ and for any $1 < \delta < |T|$, we define the *δ-compression* $L_\delta = (T_\delta, V_\delta, E_\delta)$ as $V_\delta = V$, $T_\delta = [\![\frac{\min T}{\delta}, \frac{\max T}{\delta}]\!]$, and

$$E_\delta = \{(t, \{u, v\}) \mid \exists t' \in T : \delta t \leq t' < \delta(t+1) \wedge (t', \{u, v\}) \in E\}.$$

δ	$\lvert T\rvert$	$\lvert E\rvert$	γ	$\lvert\gamma_E\rvert$
$3600s = 1h$	27300	21959	24	0
$7200s = 2h$	13650	20962	12	0
$10800s = 3h$	9100	20284	8	0
$14400s = 4h$	6825	19732	6	16
$21600s = 6h$	4550	19071	4	69
$28800s = 8h$	3413	18402	3	335
$43200s = 12h$	2275	17610	2	2667

Fig. 3. Enron dataset: number of timed edges and γ-edges after time-compression. Values are taken such that $\delta * \gamma = 24hours$. In particular, we observe that Enron employee will not continuously share 1 mail per hour during 24 hours, since the company is closed at night. When compared to the number $|T|$ of time instants, the number $|V|$ of vertices is very small (under two hundred) and not presented here.

δ	$\lvert T\rvert$	$\lvert E\rvert$	γ	$\lvert\gamma_E\rvert$
$1s$	9977	403834	7200	0
$5s$	1996	127401	1240	0
15	666	77989	480	0
$30s$	333	60919	240	0
$60s = 1m$	167	45469	120	0
$300s = 5m$	34	22484	24	51
$600s = 10m$	17	15808	12	357
$1200s = 20m$	9	10735	6	1893
$1800s = 30m$	6	8324	4	2745
$3600s = 1h$	3	5000	2	3094

Fig. 4. Rollernet dataset: number of timed edges and γ-edges after time-compression. Values are taken such that $\delta * \gamma = 7200s = 2hours$. In particular, we observe that every person in the Rollernet experiment has been away from another person for at least 1 minutes during 2 hours. When compared to the number $|T|$ of time instants, the number $|V|$ of vertices is very small (under on hundred) and not presented here.

Fig. 3 and 4 show that parameters δ and γ have an important influence on the number of $\gamma-$edges. One can also notice from Fig. 3 and 4 that the compression process generally breaks down the number $|E|$ of timed edges in the dataset. However, we stress that it is not necessarily the case: Fig. 5 exemplifies two different time-compressions of the same original link stream, respectively with $\delta = 3$ and $\delta = 4$, where it is possible to obtain more edges even if we have a larger δ. Nonetheless, the usual effect of time-compression is to drastically reduce the number of timed edges. On Enron and Rollernet datasets, we need to ensure that time-compression does not result in empty inputs. Luckily, for sensible values of δ, e.g. $\frac{1}{2}$ week for Enron or 15 minutes for Rollernet, there is still a large number of timed edges after δ-compression, cf. Fig. 6. We give in the subsequent Fig. 7 and 8 the runtime of our algorithm on random pieces of the two Enron and Rollernet datasets, where we observe that our runtime is very quick.

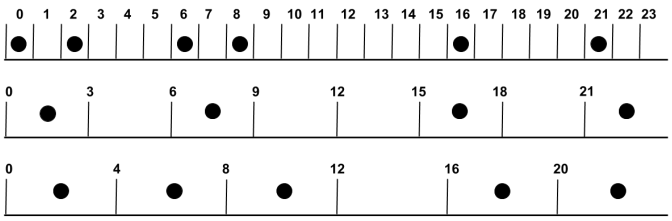


Fig. 5. Time-compression with $\delta = 3$ and $\delta = 4$, resulting in 4 and 5 timed edges.

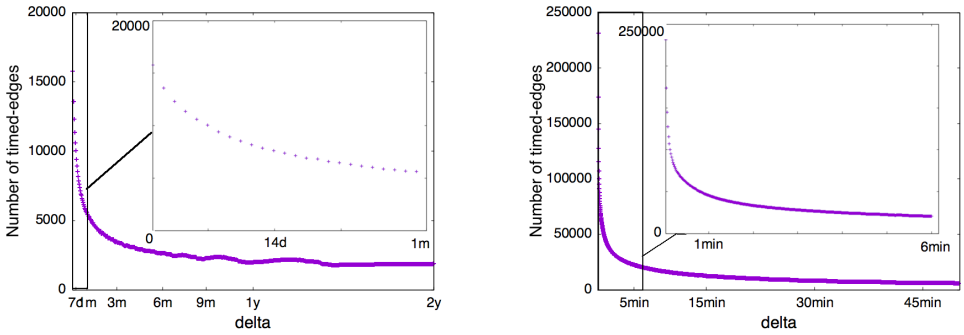


Fig. 6. Enron dataset (left) and Rollernet dataset (right): remaining timed edges after δ-compression. For instance, with $\delta = \frac{1}{2}$ week, the number of timed edges after δ-compression Enron is over ten thousand. With $\delta = 15$ minutes, the number of timed edges after δ-compression Rollernet is also over ten thousand. We conclude that for sensible values of δ, our time compression process does not break down the input to an empty instance.

5.2 Hypothesis

We theorise three hypotheses. For each hypothesis we run experiment on the above described datasets, and expose our results in the next Subsection 5.3. We discuss and conclude our numerical analysis in the subsequent and last Subsection 5.4 of current Section 5.

Hypothesis 1. Consistence of the formalism:
We would like to verify that γ-MATCHING is non-trivial on human values for δ and γ. For instance, we suppose when mining emails that collaborating during a month at a rate of at least two emails per week (round trip) is sensibly human values. When mining proximity records of 2×80 minute Rollerblade touring Paris, we consider that collaborating during 80 minutes at a rate of one visit every quarter hour (water/snack supplying) is sensibly human values.

Hypothesis 2. Kernelization quality:
We reckon that, beside the hidden constants under the Landau notation, a space reduction from $O(n)$ to $O(k^2)$, when $k = O(\sqrt{n})$ is also meaningless. Unfortunately, when parameterizing by the size of the solution as we do in this paper, one very usually results in the situation where k is numerically in the

Link-Stream$_\delta$	$\|V\|$	$\|T\|$	$\|E\|$	$\|\gamma_E\|$	appr(s)	kern(s)	total(s)
$Enron_{1hour}$	150	27300	21959	1991	0.010	0.397	0.408
$Enron_{3hours}$	150	9100	20284	2695	0.018	0.694	0.696
$Enron_{1day}$	150	1138	16224	4416	0.007	1.76	1.774
$Enron_{3days}$	150	380	12868	4644	0.007	1.932	1.939
$Enron_{7days}$	150	163	10028	4812	0.005	1.173	1.179
$Enron_{30days}$	150	38	5573	2917	0.001	0.147	0.149
$Enron_{90days}$	150	13	3480	1650	6.6E-4	0.029	0.030
$Rollernet_{1min}$	61	167	45469	24009	0.098	1.696	1.794
$Rollernet_{2mins}$	61	84	33304	17089	0.047	0.561	0.609
$Rollernet_{5mins}$	61	34	22484	13346	0.018	0.140	0.158
$Rollernet_{15mins}$	61	12	12410	8544	0.005	0.044	0.050
$Rollernet_{30mins}$	61	6	8324	5979	0.005	0.032	0.038
$Rollernet_{1hour}$	61	3	5000	3094	0.001	0.007	0.008
Generated	10	50	684	83	4.4E-4	0.001	0.001
Generated	10	100	1384	136	4.8E-4	7.4E-4	0.001
Generated	10	200	2906	322	4.1E-4	0.002	0.002
Generated	20	50	2599	705	0.001	0.004	0.006
Generated	20	100	5326	1508	0.003	0.016	0.020
Generated	20	200	10667	2998	0.004	0.030	0.035
Generated	50	50	15842	6534	0.005	0.034	0.040
Generated	50	100	31113	12876	0.018	0.125	0.144
Generated	50	200	63032	26495	0.054	0.426	0.480
Generated	100	50	53665	32235	0.093	0.280	0.374
Generated	100	100	107524	65145	0.342	1.054	1.396
Generated	100	200	214728	130371	1.437	4.277	5.713

Fig. 7. Runtime required by our 2-approximation algorithm, kernelization algorithm, and the total process. Values are taken for $\gamma = 2$. The parameter k for the kernelization algorithm is the size of the solution found by the 2-approximation algorithm. The (rawly recorded) runtime is very quick, hence, probably subject to many noises.

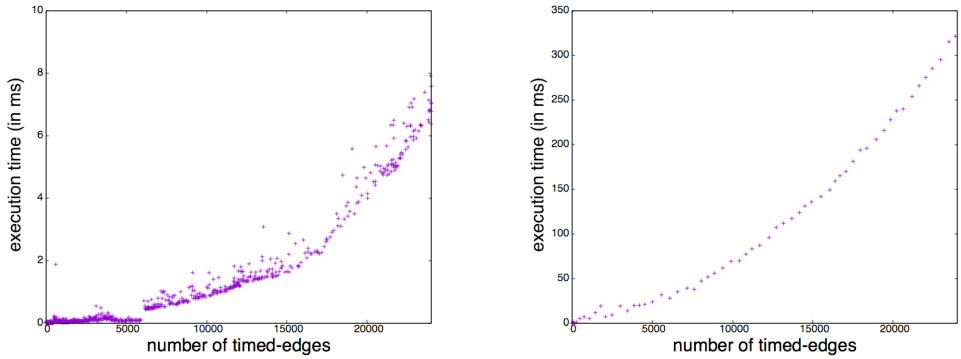


Fig. 8. Enron dataset (left) and Rollernet dataset (right): runtime of our algorithms in function of the number of timed edges, with $\gamma = 2$. The parameter k for kernelization is the size of the 2-approximation result. The (rawly recorded) runtime is probably subject to many noises. We rather refer to Fig. 2 for evaluating performance. Each dot in current Fig. 8 is obtained by first truncating the raw input with varying maximum value of time instants, then, δ-compressing the so-obtained link streams with $\delta = 100$. The only observation we make with this figure is that, on real world dataset, our combined runtime for both the 2-approximation and the kernelization algorithms is very quick.

order of $\sqrt{n}$. This is particularly true for our study of temporal matchings on Enron and Rollernet. For this reason, our second hypothesis is that, in addition to a theoretical guarantee of reduction from $O(n)$ to $O(k^2)$ space complexity, solving γ-MATCHING can numerically benefit from the kernelization algorithm described in Theorem 2, at least on well-chosen and humanly sensible intervals of δ and γ.

Hypothesis 3. Approximation quality:
We stress that MATCHING in a classical graph is polynomial. Unfortunately, γ-MATCHING in a link stream is NP-hard. However, in practice, a lot of NP-complete problems are not difficult on datasets arising from human activities. What's more, some such problems can be solved near-optimally by simple algorithms such as by a random or greedy approach, or a mix of both approaches, even on arbitrary inputs. A popular example is COLORING [15]. Accordingly, our last hypothesis is that, in practice, finding an optimal γ-matching need not to be difficult. Moreover, we hypothesise that the greedy 2-approximation described in Lemma 1 can produce near-optimal γ-matching on real world dataset, as well as artificial datasets that mimic real word datasets.

5.3 Result

Both our 2-approximation and kernelization algorithms are implemented and confronted to the above mentioned datasets.

Observations w.r.t. Hypothesis 1. Consistence of the formalism:
Results are given in Fig. 9. We observe on Enron dataset with $\delta \approx \frac{1}{2}$ week that, after δ-compression, the number of γ-edges for γ varying from 2 to 10 is: over 500 for $\gamma = 10$; over 1000 for $\gamma = 6$; and over 4500 for $\gamma = 2$. Moreover, we will see in the next paragraph that our numerical analysis can only find γ-matchings of size approximately one fifth of the previous number. We also tried other techniques to improve the size of the γ-matchings but failed in finding substantial difference. With the Enron dataset, we believe that γ-MATCHING for $\gamma \approx 10$ is a tricky question when the time-compression rate is $\delta \approx \frac{1}{2}$ week. These values translate the fact that any pair of collaborators in the γ-matching necessarily keep exchanging emails together continuously for one month, at a rate of at least one email per week and in average at least two emails every week.

Observations w.r.t. Hypothesis 2. Kernelization quality:
Results are given in Fig. 10. The parameter k for the kernelization algorithm is the size of the solution found by the 2-approximation algorithm. We observe that on well chosen intervals of δ and γ, kernelization reduces the input size down to under twenty per cent. This is particularly true for $\gamma \approx 20, 30$ with

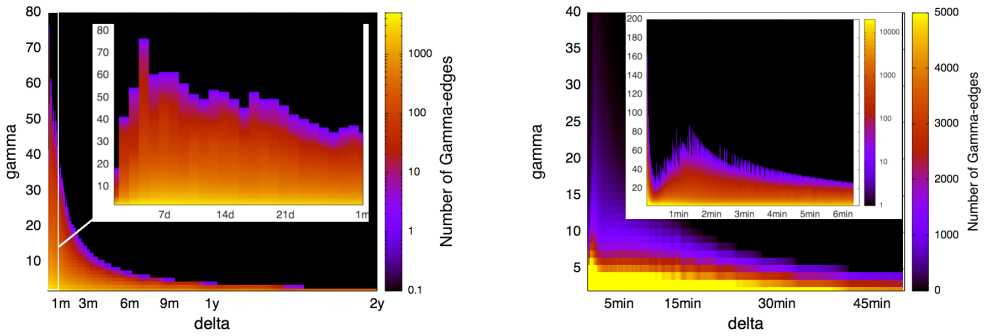


Fig. 9. Enron dataset (left) and Rollernet dataset (right): number of γ-edges after δ-compression, for varying values of δ and γ.

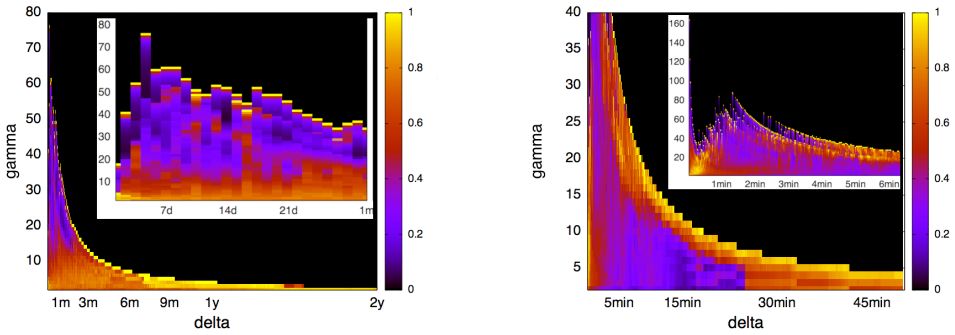


Fig. 10. Enron dataset (left) and Rollernet dataset (right): ratio obtained by dividing the number of γ-edges in the kernelization output by the number of γ-edges in the kernelization input (which is obtained after δ-compression). Here, the darker is the better. The parameter k for the kernelization algorithm is the size of the solution found by the 2-approximation algorithm.

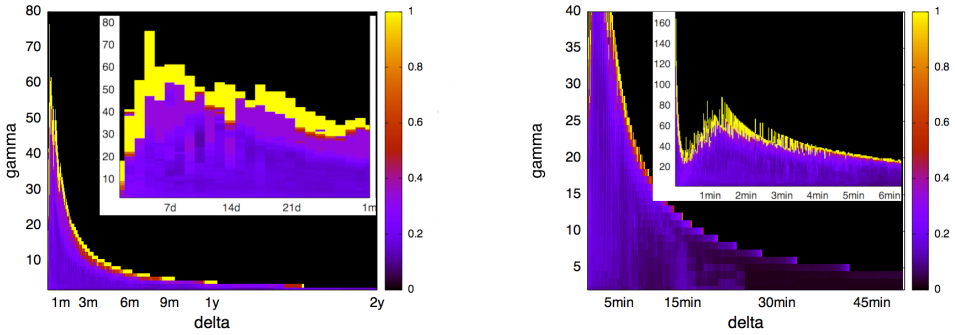


Fig. 11. Enron dataset (left) and Rollernet dataset (right): ratio obtained by dividing the 2-approximation output by the number of γ-edges in the kernelization output. Here, the brighter is the better: the approximation solves γ-MATCHING optimally on regions where the ratio is 100%, which is denoted by the yellow color. The parameter k for the kernelization algorithm is the size of the solution found by the 2-approximation algorithm.

$\delta \leq 2$ month on Enron; and $\gamma \leq 10$ with $\delta \leq 20$ minutes for Rollernet. We observe on these values that the kernelization algorithm reduces the input link stream to an equivalent instance of size smaller than 20% the size of the original input, and sometimes under 10%. We conclude that Hypothesis 2 is sound.

Observations w.r.t. Hypothesis 3. Approximation quality:
Results are given in Fig. 11. The parameter k for the kernelization algorithm is the size of the solution found by the 2-approximation algorithm. We notice from definition that the 2-approximation algorithm produces a lower bound for γ-MATCHING, which is at least half the optimal value. Moreover, the kernelization algorithm gives a naive upper bound for γ-MATCHING by simply counting the number of γ-edges present in the kernel. We observe for tangent areas in Fig. 11 that these two upper and lower bounds meet. This means that the 2-approximation outputs an optimal solution for γ-MATCHING on these areas. However, we observe that for most parts of our dataset, Hypothesis 3 is not confirmed. At this state, our experiments w.r.t. Hypothesis 3 fail in giving any clue for a conclusion. Further experiments must be done in order to clarify this question. Our feeling, however, is that Hypothesis 3 is generally false. We conjecture that the 2 approximation factor is far from optimal.

5.4 Discussion

Our experiment results are optimistic about the numerical usefulness of kernelization in finding temporal matching. At the same time, they also point out several questions. While our experiments allow us to observe that:

- preprocessing an instance of γ-MATCHING by a greedy process, and then kernelization as described in Theroem 2, seems to be sound;
- the preprocessing is very quick on real world input;
- the preprocessing is robust versus stress testing on large inputs using common laptop: below ten seconds on input of hundreds thousand timed edges;

they also testify that works still need to be done for further investigating γ-MATCHING, especially that:

- the optimisation problem is NP-hard;
- numerical proofs of optimality of the 2-approximation are only available for very marginal bits of data in the Enron and Rollernet datasets;
- while it is true that the kernelization algorithm helps in reducing the input down to $10-20$% for interesting mining parameters on Enron and Rollernet datasets, we still do not know how then to find a γ-matching of the kernel that is better than the output of the 2-approximation;
- in particular, we do not know if the approximation factor can be improved.

6 Conclusion and perspectives

We introduce the notion of temporal matching in a link stream. Unfortunately, the problem of computing a temporal matching of maximum size, called γ-MATCHING, turns out to be NP-hard. We then show a kernelization algorithm for γ-MATCHING parameterized by the size of the solution. Our process produces quadratic kernels. On the way to obtaining the kernelization algorithm, we also provide a 2-approximation algorithm for γ-MATCHING. We believe that the same techniques extend to a large class of hitting set problems in link streams.

Acknowledgements: We are grateful to Clémence Magnien for helpful pointers and valuable pieces of advice. We are grateful to the anonymous referees for helpful comments which greatly improve the paper. In particular, we are grateful to the referee who coined the name of time-compression, whose usage greatly improves the paper. For financial support, we are grateful to: *Thales Communications & Security*, project TCS.DJ.2015-432; *Agence Nationale de la Recherche Technique*, project 2016.0097; *Centre National de la Recherche Scientifique*, project INS2I.GraphGPU.

References

[1] D. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4):1030–1045, 2007.

[2] E. Bampis, B. Escoffier, M. Lampis, and V. T. Paschos. Multistage matchings. In *16th Scandinavian Symposium and Workshops on Algorithm Theory*, volume 101 of *LIPIcs*, pages 7:1–7:13, 2018.

[3] T. H. Cormen, C. E. Leiserson, R. L Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 1989.

[4] M. Cygan, H. N. Gabow, and P. Sankowski. Algorithmic applications of Baur-Strassen's theorem: Shortest cycles, diameter, and matchings. *Journal of the ACM*, 62(4):28:1–28:30, 2015.

[5] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[6] F. Dufossé, K. Kaya, I. Panagiotas, and B. Uçar. Approximation algorithms for maximum matchings in undirected graphs. In *SIAM Workshop on Combinatorial Scientific Computing*, 2018. `https://hal.archives-ouvertes.fr/hal-01740403`.

[7] C. Dürr, C. Konrad, and M. P. Renault. On the Power of Advice and Randomization for Online Bipartite Matching. In *24th Annual European Symposium on Algorithms*, volume 57 of *LIPIcs*, pages 37:1–37:16, 2016.

[8] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[9] F. V. Fomin, D. Lokshtanov, S. Saurabh, M. Pilipczuk, and M. Wrochna. Fully polynomial-time parameterized computations for graphs and matrices of low treewidth. *ACM Transactions on Algorithms*, 14(3):34:1–34:45, 2018.

[10] B. Klimt and Y. Yang. Introducing the Enron Corpus. In *CEAS*, 2004.

[11] M. Latapy, T. Viard, and C. Magnien. Stream graphs and link streams for the modeling of interactions over time. 2017. `https://arxiv.org/abs/1710.04073`.

[12] G. B. Mertzios, A. Nichterlein, and R. Niedermeier. The power of linear-time data reduction for maximum matching. In *42nd International Symposium on Mathematical Foundations of Computer Science*, volume 83 of *LIPIcs*, pages 46:1–46:14, 2017.

[13] O. Michail and P. G. Spirakis. Traveling salesman problems in temporal graphs. *Theoretical Computer Science*, 634:1–23, 2016.

[14] S. Miyazaki. On the advice complexity of online bipartite matching and online stable marriage. *Information Processing Letters*, 114(12):714–717, 2014.

[15] M. Molloy and B. Reed. *Graph Colouring and the Probabilistic Method*. Springer, 2002.

[16] M. Mucha and P. Sankowski. Maximum matchings via Gaussian elimination. In *45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 248–255, 2004.

[17] P.-U. Tournoux, J. Leguay, F. Benbadis, V. Conan, M. D. De Amorim, and J. Whitbeck. The Accordion Phenomenon: Analysis, Characterization, and Impact on DTN routing. In *28th IEEE Conference on Computer Communications*, 2009.

[18] T. Viard, M. Latapy, and C. Magnien. Computing maximal cliques in link streams. *Theoretical Computer Science*, 609:245–252, 2016.

[19] Y. Wang and S. C.-W. Wong. Two-sided Online Bipartite Matching and Vertex Cover: Beating the Greedy Algorithm. In *42nd International Colloquium on Automata, Languages, and Programming*, volume 9134 of *LNCS*, pages 1070–1081, 2015.

[20] S. Wøhlk and G. Laporte. Computational comparison of several greedy algorithms for the minimum cost perfect matching problem on large graphs. *Computers and Operations Research*, 87(C):107–113, 2017.