Temporal Matching on Geometric Graph Data⋆

Timothe Picavet[1], Ngoc-Trung Nguyen[2], and Binh-Minh Bui-Xuan[3]

[1] ENS Lyon, `timothe.picavet@ens-lyon.fr`
[2] HCM University of Education, `trungnn@hcmue.edu.vn`
[3] LIP6 (CNRS – Sorbonne Université), `buixuan@lip6.fr`

Abstract. Temporal graphs are the modeling of pairwise and historical interaction in recordings of a dataset. A temporal matching formalizes the planning of pair working sessions of a required duration. We depict algorithms finding temporal matchings maximizing the total workload, by an exact algorithm and an approximation. The exact algorithm is a dynamic programming solving the general case in $O^*((\gamma + 1)^n)$ time, where n is the number of vertices, γ represents the desired duration of each pair working session, and O^* only focuses on exponential factors. When the input data is embedded in an Euclidean space, called geometric data, our approximation is based on a new notion of temporal velocity. We revise a known notion of static density [van Leeuwen, 2009] and result in a polynomial time approximation scheme for temporal geometric graphs of bounded density. We confront our implementations to known opensource implementation[4].

Keywords: temporal matching · geometric graph · PTAS

1 Introduction

Data collected from automated processes come ordered by the time instants when they are recorded. Graphs in this context appear in several variants: link streams [18], time varying [7], temporal [14] or evolving graphs [6]. These structures occur in the study of transportation timetables [9,15,16], navigation programs [26], email exchanges [17], proximity interactions [27], and many other types of dataset [28]. Therein, a pair working session is a repeated interaction of two vertices over a certain amount of time. Pair working helps in optimizing global parameters such as total fuel consumption when co-sailing with Fello'fly [2], or code reliability when running XP agile projects [1].

The total workload of pair working is captured in the notion of a temporal matching [4]. Given an integer γ, we define formally problem γ-MATCHING in the subsequent section; informally, it consists in finding a maximum cardinality set of compatible pair working sessions, each to be recorded in at least γ consecutive timestamps in a historical dataset of graphs.

---

[4] Our source code is available at https://github.com/Talessseed/Temporal-matching-of-historical-and-geometric-graphs

When $\gamma = 1$, the problem can be reduced to (classical) static MATCHING, which consists in computing a maximum independent edge set of a static graph. It can be solved in polynomial time by many well known algorithms [8,13,23], heuristics [11], greedy approximations for large input [31]; as well as in an on-line algorithmic context [12,22,30]. It is very intriguing to know whether these enthusiastic results extend to the non-static case of γ-MATCHING.

Unfortunately, very little positive results are known for the temporal case. Even when restricting the input to be a path at each instant, one can very naturally obtain a grid-like structure by folding out a temporal graph instance over the time dimension. On these structures, careful polynomial and parameterized reductions allow to obtain very good hardness results, see *e.g.* [3,21] and the extensive bibliography therein. Likewise, γ-MATCHING is NP-hard as soon as $\gamma > 1$ [4], even on very restricted input instances [20]. To the best of our knowledge, most notable positive results for γ-MATCHING are: a fixed parameter tractable (FPT [10]) algorithm parameterized by the matching number of the union graph[5] [20], an implemented kernelization producing quadratic kernels [4,5] (in the sense of FPT algorithms), and an implemented 2-approximation from a greedy approach [5].

Our paper addresses the following question: *Would there be fast algorithms computing γ-MATCHING on data recorded from human activities? Can they be implemented?* Human data are not artificial, yet very naturally captured by a geometric graph: an embedding of a vertex set into a Euclidean space, along with a real number representing the threshold below which an edge exists between two vertex-points, see *e.g.* [19]. The formalism is especially useful in transportation and social networks where geometric proximity implies higher probability of successful routing, resp. social relationship [24].

Theoretical contribution. In order to obtain good runtime, we consider *natural behaviour* of embedded vertex-points. The main crux is to carefully examine a notion of partial derivative, called velocity. This parameter helps ruling out unrealistic leaps of a vertex-point from one recorded instant to another. We revisit, ubiquitously, the parameter control used in [29] which is related to the (static) density of vertex-points. Then, we present and implement a PTAS for temporal geometric graphs of bounded velocity and density.

Numerical comparison to previous works. We compare our result to previously known works. The FPT algorithm given in [20] has not been implemented. In particular, part of this algorithm relies on complex algorithmic results in matroid theory. We skip the corresponding analysis. The kernelization implementation [4,5] helps in reducing the input data, but not in solving the reduced instance. On instances where we can afford the runtime, we use it as preprocessing step for the PTAS, exactly the same way done for the greedy implementation [5]. Essentially, the PTAS is compared to the greedy implementation. Our

---

[5] If a temporal graph is considered to be a sequence $(G_t)_{t \in T}$ of graphs over the same vertex set, then the edge set of the union graph is the union of all edge sets of G_t for all t.

numerical results are in favour of the latter, which finds temporal matching of size $\approx 10\%$ bigger than the PTAS on generated geometric datasets. Since the theoretical approximation factor of the greedy algorithm is 2 [4], which is much worse than the theoretical ratio of the PTAS on our datasets, these experiments raise the question whether both implementations perform badly, or the greedy approximation factor is near optimal on geometric data.

We devise and implement an optimal solution for the general case of γ-Matching terminating in reasonable time on parts of our datasets, that is, in $O^*((\gamma + 1)^n)$ time where n is the number of vertices and O^* only focuses on exponential factors. The PTAS and the greedy experimental approximation ratios are then determined, which average at 1.02-approximation from optimal.

We present in Section 2 the formal framework of problem γ-Matching. Section 3 presents a PTAS solution for the case of temporal geometric graphs of bounded velocity and density. Section 4 presents a FPT solution for the general case. Due to space restriction, properties marked with $(\star)$ are given without proof, and only the essential numerical experiments are summarised in Section 5.

2 Pair working sessions in historical graph data

Every graph $G = (V, E)$ in this paper is simple, loopless and undirected. We also note $V(G) = V$ and $E(G) = E$. When, and only when, $u \neq v \in V$, we abusively note $uv = vu = \{u, v\} \in \binom{V}{2}$ the *edge* between u and v.

Graph data collected over a duration of time are formalized as a triple $L = (T, V, E)$, called *link stream*, such that $T \subseteq \mathbb{N}$ is an interval, V a finite set of *vertices*, and E a lexicographically ordered subset of $T \times \binom{V}{2}$ called *recorded edges*. We also note $T(L) = T$, $V(L) = V$ and $E(L) = E$.

Pairwise collaborations over a duration are defined as γ-edges, with γ an integer. A γ-*edge* $\Gamma \subseteq E(L)$ is a subset of γ consecutive edges recorded in $E(L)$, namely $\Gamma = \{(i, uv) \in E(L) : t \leq i < t + \gamma\}$ for $t \in T(L)$ and $u \neq v \in V(L)$. We also note such γ-edge $E_\gamma(t, uv)$.

We note $E_\gamma(L)$ the set of all γ-edges of L. Two γ-edges $\Gamma, \Gamma' \in E_\gamma(L)$ are *dependent* if there exist instant i and vertices $u \neq v$, $u \neq w$, such that $(i, uv) \in \Gamma$ and $(i, uw) \in \Gamma'$; the two γ-edges are *independent* otherwise. In planning pair working sessions, a conflict-free planning is called a γ-*matching*, and defined as a set of pairwise independent γ-edges. The following problem is NP-hard for $\gamma > 1$ [4], even on very restricted classes of link streams [20].

Problem γ-Matching:
Input: a link stream L
Output: a γ-matching of maximum cardinality in L.

Geometric model: Let L be a link stream. The subgraph G_t of L induced at time $t \in T(L)$ is defined as $V(G_t) = V(L)$ and $E(G_t) = \{uv : (t, uv) \in E(L)\}$. For $d \in \mathbb{N}$, graph G is a *unit ball graph* if there exists a point set $\{\mathbf{c}_v : v \in V(G)\} \subseteq \mathbb{R}^d$, called set of *centers*, such that $E(G) = \{uv : u \neq v \wedge \|\mathbf{c}_u - \mathbf{c}_v\| \leq 1\}$. Link stream L is a *unit ball stream* if the subgraph of L induced at any time

$t \in T(L)$ is a unit ball graph. In this case, we denote the center of vertex v at time t in L as $\mathbf{c}_v(t)$. L has *velocity* ν if $\|\mathbf{c}_v(t+1) - \mathbf{c}_v(t)\| \leq \nu$ for every $t \in T \setminus \{\max(T)\}$ and $v \in V(L)$. We also refer to balls as *intervals* when $d = 1$ and *disks* when $d = 2$.

Line graph extrapolation: γ-MATCHING links itself to MAXIMUMINDEPEN-DENTSET on the following input. The γ-*line graph* L_γ of a link stream L is defined as $V(L_\gamma) = E_\gamma(L)$ and $E(L_\gamma) = \{\{\Gamma, \Gamma'\} : \Gamma$ and Γ' are dependent γ-edges$\}$. By definition, solving γ-MATCHING on a link stream is equivalent to solving MAXIMUMINDEPENDENTSET on its γ-line graph.

3   Approximation for unit ball streams

In this section, we use velocity and extend van Leeuwen approximation [29, Theorem 6.3.8, page 74] for MAXIMUMINDEPENDENTSET on unit disk graphs to the γ-line graph of a unit ball stream L. Since the γ-line graph is not necessarily a unit ball graph, our main idea is to examine the middle of the two vertex-points of every γ-edge $\Gamma \in E_\gamma(L)$: the middle point can not vary much because of velocity. Corollary 1 below is crucial to our approach.

For a γ-edge $\Gamma = E_\gamma(t, uv) \in E_\gamma(L)$ between $u \neq v \in V(L)$ starting at time $t \in T(L)$, the *(middle) center* $\mathbf{c}_\Gamma$ is defined as the middle point of the centers of u and v recorded at the starting time t of the γ-edge, $\mathbf{c}_\Gamma = \frac{1}{2} \cdot (\mathbf{c}_u(t) + \mathbf{c}_v(t))$. Using velocity of the centers, which can only move $\gamma - 1$ times while maintaining the existence of Γ, we refer to the *normalized center* of Γ as $\overline{\mathbf{c}_\Gamma} = \frac{1}{1+(\gamma-1)\nu} \cdot \mathbf{c}_\Gamma$.

Proposition 1. *In a unit ball stream, if Γ and Γ' are dependent γ-edges, then* $\|\overline{\mathbf{c}_\Gamma} - \overline{\mathbf{c}_{\Gamma'}}\| \leq 1$.

Proof. Let $\Gamma = E_\gamma(t, uv)$ and $\Gamma' = E_\gamma(t', u'v')$. Because of dependency, we suppose w.l.o.g. that $u = u'$, and $t \leq t' \leq t + \gamma - 1$. We note from Euclidean triangular inequality that $\|\mathbf{c}_\Gamma - \mathbf{c}_{\Gamma'}\| \leq \|\mathbf{c}_\Gamma - \mathbf{c}_u(t)\| + \|\mathbf{c}_u(t) - \mathbf{c}_u(t')\| + \|\mathbf{c}_u(t') - \mathbf{c}_{\Gamma'}\|$. Now, $\|\mathbf{c}_\Gamma - \mathbf{c}_u(t)\| \leq \frac{1}{2}$ because $\|\mathbf{c}_u(t) - \mathbf{c}_v(t)\| \leq 1$. Since $u = u'$, we deduce likewise that $\|\mathbf{c}_u(t') - \mathbf{c}_{\Gamma'}\| \leq \frac{1}{2}$. Finally, $\|\mathbf{c}_u(t) - \mathbf{c}_u(t')\| = \left\| \sum_{i=t}^{t'-1} \big(\mathbf{c}_u(i+1) - \mathbf{c}_u(i) \big) \right\| \leq \sum_{i=t}^{t'-1} \|\mathbf{c}_u(i+1) - \mathbf{c}_u(i)\| \leq (t' - t)\nu \leq (\gamma - 1)\nu$. $\square$

Since the normalized centers are uniquely computed from their starting instant, this is also a fast checking method for independent γ-edges. We refer to the unit ball graph having as geometric model the set of normalized centers of all γ-edges of L the *normalized γ-line graph* of L.

Corollary 1. *The normalized γ-line graph of a unit ball stream is a unit ball graph having the γ-line graph as partial subgraph. Any independent set of the γ-line graph is also an independent set of its normalized graph*

We now adapt the notion of density [29] to the normalized γ-line graph of a unit ball stream L of dimension d. Let $A \subseteq E_\gamma(L)$. We refer to the set of all γ-edges of A starting at time $t \in T(L)$ as $A_t = \{E_\gamma(t, uv) \in A : u \neq v \in V(L)\}\}$. The *thickness of A* is the maximum cardinality of such a set, taken over every possible starting time, that is, $\theta(A) = \max\{|A_t| : t \in T(L)\}$.

In the sequel, all cubes are axis-aligned cubes. For a unit d-cube $\mathbf{U} \subseteq \mathbb{R}^d$, we denote $A_\mathbf{U} = \{\Gamma \in A : \overline{\mathbf{c}_\Gamma} \in \mathbf{U}\}$. The *density of A* is the maximum thickness of such a set, taken over every possible unit d-cube, that is, $\rho(A) = \max\{\theta(A_\mathbf{U}) : \mathbf{U} \subseteq \mathbb{R}^d \text{ a unit } d\text{-cube}\}$. The *density of L* is $\rho(L) = \rho(E_\gamma(L))$.

We will describe in Lemma 1 a decrementing process for the Euclidean space dimension. Informally, this is a partial density relaxing the first dimension of the unit d-cubes to infinite unit cuboids. For a unit $(d-1)$-cube $\mathbf{H} \subseteq \mathbb{R}^{(d-1)}$, we denote $A_\mathbf{H} = \{\Gamma \in A : \overline{\mathbf{c}_\Gamma} \in \mathbb{R} \times \mathbf{H}\}$ (we replace the unit d-cube $\mathbf{U}$ in the definition of density by the infinite unit cuboid $\mathbb{R} \times \mathbf{H}$). The *partial density of A (w.r.t. the first dimension)* is the maximum thickness of such a set, taken over every possible infinite unit cuboids, that is, $\partial\rho(A) = \max\{\theta(A_\mathbf{H}) : \mathbf{H} \subseteq \mathbb{R}^{(d-1)} \text{ a unit } (d-1)\text{-cube}\}$. We observe when $d = 1$ that the partial density is the thickness.

For a γ-edge $\Gamma \in E_\gamma(L)$, let $\overline{x_\Gamma}$ denote the projection of $\overline{\mathbf{c}_\Gamma}$ on the first dimension, that is, $\overline{\mathbf{c}_\Gamma} = (\overline{x_\Gamma}, \dots)$. A *decomposition path X* is a set of scalars ordered increasingly, $X = \{x_1, x_2, \dots, x_{|X|}\}$. We define the *incomplete partition of A by X (w.r.t. the first dimension)*, noted $P_X(A) = (P_0(A), P_1(A), \dots, P_{|X|}(A))$, as follows. Firstly, $P_0(A) = \{\Gamma \in A : \overline{x_\Gamma} < x_1 - \frac{1}{2}\}$. For $0 < i < |X|$, $P_i(A) = \{\Gamma \in A : x_i + \frac{1}{2} \leq \overline{x_\Gamma} < x_{i+1} - \frac{1}{2}\}$. Finally, $P_{|X|}(A) = \{\Gamma \in A : x_{|X|} + \frac{1}{2} \leq \overline{x_\Gamma}\}$.

Basically, this is a partition of the Euclidean space into $|X| + 1$ parts called slab decomposition [29]. $P_X(A)$ corresponds to the γ-edges inside each part, while those incident to the boundaries are removed. From Proposition 1, two γ-edges of different parts are independent. Hence, $P_X(A)$ can also be seen as a partial decomposition tree for the pathwidth [25] of the (normalized) γ-line graph of L. In the rest of this section, we describe a way to compute such a set X.

Lemma 1. *Let L be a link stream with density ρ. Let $E_\gamma = E_\gamma(L)$ and $m_\gamma = |E_\gamma|$. Let f_L be a big enough integer, $f_L \geq \rho$. Then, we can compute in time polynomial in m_γ a decomposition path $X = \{x_1, x_2, \dots, x_{|X|}\}$, in such a way that the incomplete partition $P_X(E_\gamma)$ satisfies:*

- $P_0(E_\gamma) = P_{|X|}(E_\gamma) = \emptyset$,
- $\forall 0 < i < |X| - 1$, $f_L \leq \partial\rho(P_i(E_\gamma)) < f_L + \rho$,
- $0 \leq \partial\rho(P_{|X|-1}(E_\gamma)) < f_L + \rho$,
- $x_{|X|} - x_{|X|-1} \geq \frac{f_L}{\rho}$.

Proof. We would like to scan the γ-edges of E_γ in such a way to only increase the partial density. The main crux is to process greedily along the same x-axis w.r.t. which the partial density is defined: informally, the infinite unit cuboids $\mathbb{R} \times \mathbf{H}$ with $\mathbf{H}$ a unit $(d-1)$-cube can be seen as FIFO strips along this x-axis.

Formally, sort $E_\gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_{m_\gamma}\}$ so that $\overline{x_{\Gamma_1}} \leq \overline{x_{\Gamma_2}} \leq \cdots \leq \overline{x_{\Gamma_{m_\gamma}}}$. In the following, i and P are auxiliary variables containing an index and a set of centers, respectively. Initialize $i \leftarrow 1$, $P \leftarrow \emptyset$, define $x_i = \overline{x_{\Gamma_1}}$ minus one unit, and increment i. For all $\Gamma \in E_\gamma$ in increasing order, if the partial density $\partial\rho(P)$ is strictly less than f_L, add Γ to P along with all other Γ' with $\overline{x_\Gamma} = \overline{x_{\Gamma'}}$, skipping the partial density check for these Γ'. We call this step (**Add**). Else, create a new boundary by defining $x_i = \overline{x_\Gamma}$, emptying P, and incrementing i. At the end of the iteration process, define x_i to be the last seen $\overline{x_\Gamma}$ plus one unit (in order to avoid coinciding with the previous x_i). Finally, increment i again and define x_i to be an arbitrarily big number so that it satisfies the last item of Lemma 1. Partial density checks can be done in polynomial time in m_γ because of Procedure 1 described below. Hence, the overall process is polynomial in m_γ.

All parts $P_i(E_\gamma)$ defined by the computed x_i's have a partial density of at least f_L, except for the first and the last two parts. The only thing left to prove is $\partial\rho(P_i(E_\gamma)) < f_L + \rho$. By contradiction suppose that the partial density exceeds that number on some part P_i. This could only happen after adding a set A of γ-edges in some step (**Add**). Then, we must have $\partial\rho(A) > \rho$ because adding γ-edges along the x-axis can only increase the partial density w.r.t. that axis. Let $\mathbf{H}$ be the unit $(d-1)$-cube such that $\partial\rho(A) = \theta(A_\mathbf{H})$. Consider then the unit cube $\mathbf{U}$ defined by $\mathbf{U} = \left[\overline{x_\Gamma} - \frac{1}{2}, \overline{x_\Gamma} + \frac{1}{2}\right] \times \mathbf{H}$, with $\Gamma \in A$. It holds $E_{\gamma\mathbf{U}} \supseteq A_\mathbf{H}$. Hence, $\rho = \rho(E_\gamma) \geq \theta(E_{\gamma\mathbf{U}}) \geq \theta(A_\mathbf{H}) = \partial\rho(A) > \rho$. Contradiction. $\qquad\square$

Procedure 1: Procedure to calculate the density of a set A of γ-edges

---

1 $\rho \leftarrow 0$

    // Each C_i is a normalized center of a γ-edge of A

2 **for** $(C^1, C^2, \ldots, C^d) \in \{\overline{\mathbf{c}_\Gamma} : \Gamma \in A\}^d$ **do**

        // We consider the unit hypercube $\mathbf{H}$ with C_i^i as its i-th lowest coordinate

3 $\mathbf{H} \leftarrow [C_1^1, C_1^1 + 1] \times \cdots \times [C_d^d, C_d^d + 1]$

4 $\rho \leftarrow \max\{\rho, \theta(A_\mathbf{H})\}$

5 **return** ρ

Due to space restriction, the proof of properties marked with $(\star)$ is omitted.

Lemma 2 $(\star)$. *Keeping the same notations as in the hypothesis of Lemma 1, we have $x_{i+1} - x_i \geq \frac{f_L}{\rho}$ for any $1 \leq i < |X|$.*

Lemma 3 $(\star)$. *Keeping the same notations as in the hypothesis of Lemma 1, let $k \in \mathbb{N}$ be such that $k \leq \left\lfloor \frac{f_L}{\rho} \right\rfloor$. Let $l = |X|$. For $s \in [\![0, k-1]\!]$, we define $(P_0(s), P_1(s), \ldots, P_l(s)) = P_{\{x+s : x \in X\}}(A)$. Then, we have $\partial\rho(P_i(s)) < 2f_L$ for $i \in [\![0, l]\!]$ and $s \in [\![0, k-1]\!]$.*

Lemma 4 $(\star)$. *Keeping the same notations as in the hypothesis of previous Lemma 3, suppose that for every $s \in [\![0, k-1]\!]$ and $i \in [\![0, l]\!]$, $M_i(s)$ is a*

r-approximation of γ-Matching on $P_i(s)$. Let $M(s) = \cup_{i\in[\![0,l]\!]} M_i(s)$. Then, $\gamma MM^\sim = \max\limits_{s\in[\![0,k-1]\!]}\{M(s)\}$ is a $r\left(1-\frac{1}{k}\right)$-approximation of γ-Matching on L.

We will now show an exact algorithm to compute the base case $(d = 1)$ of the approximation. It is also a correct algorithm for arbitrary link streams.

Algorithm 1 (Exact algorithm for the base case of the PTAS, on input an arbitrary link stream)
On input any link stream L, we note $E_\gamma(t)$ the set of all γ-edges starting at time $t \in T(L)$, $E_\gamma(t) = \{E_\gamma(t, uv) \in E_\gamma(L) : u \neq v \in V(L)\}$. By convention, we note $E_\gamma(t) = \emptyset$ for $t \notin T(L)$. We proceed by dynamic programming and store in $M(t, S_1, S_2, \ldots, S_{\gamma-1})$ a γ-matching $\mathcal{M}$ of maximum cardinality of the restriction of L to time instants between 0 and $t + \gamma - 2$ where we have for $1 \leq i \leq \gamma - 1$ that $\mathcal{M} \cap E_\gamma(t - 1 + i) = S_i$.
If $\mathcal{T} = S_1 \cup S_2 \cup \cdots \cup S_{\gamma-1}$ is a γ-matching, we have the following formulae:

- *$M(0, S_1, S_2, \ldots, S_{\gamma-1}) = \mathcal{T}$*
- *$M(t+1, S_1, S_2, \ldots, S_{\gamma-1}) = S_{\gamma-1} \cup \max\limits_{\substack{S_0 \subseteq E_\gamma(t),\\ S_0 \cup \mathcal{T} \text{ is a}\\ \gamma-matching}} \{M(t, S_0, S_1, S_2, \ldots, S_{\gamma-2})\}$*

If $\mathcal{T} = S_1 \cup S_2 \cup \cdots \cup S_{\gamma-1}$ is not a γ-matching, we let $M(t, S_1, S_2, \ldots, S_{\gamma-1}) = \emptyset$. After sequentially filling table M by increasing t, we output the value stored in $M(\max(T(L)), \emptyset, \ldots, \emptyset)$. A python implementation is available[6].

Lemma 5. *On input any link stream $L = (T, V, E)$ with γ-edge set $E_\gamma = E_\gamma(L)$, Algorithm 1 computes an optimal solution for γ-Matching on L in time $O(t_{max}\gamma^2\theta(E_\gamma)2^{\gamma\theta(E_\gamma)})$, where $t_{max} = \max(T(L))$.*

Proof. We proceed by induction on t. Let $P(\mathcal{M}, t, S_1, S_2, \ldots, S_{\gamma-1})$ denote the fact that $\mathcal{M}$ is a γ-matching of maximum cardinality of the restriction of L to time instants between 0 and $t + \gamma - 2$ such that $\mathcal{M} \cap E_\gamma(t - 1 + i) = S_i$ for every $1 \leq i \leq \gamma - 1$.

Firstly, $M(0, S_1, S_2, \ldots, S_{\gamma-1}) = S_1 \cup S_2 \cup \cdots \cup S_{\gamma-1}$ because $\forall i \in [\![1, \gamma - 1]\!]$, $\mathcal{M} \cap E_\gamma(i - 1) = S_i$.

Secondly, let $t \in T(L)$, $S_i \subseteq E_\gamma(t + i)$ for each $1 \leq i \leq \gamma - 1$, and suppose that the formula is correct for $t - 1$. For convenience, let $\mathcal{T} = S_1 \cup S_2 \cup \cdots \cup S_{\gamma-1}$. We suppose that $\mathcal{T}$ is a γ-matching. Otherwise, it's trivial. Moreover, let $\mathcal{M}$ be such that $P(\mathcal{M}, t, S_1, S_2, \ldots, S_{\gamma-1})$ is satisfied and $S = \{M(t - 1, S_0, S_1, S_2, \ldots, S_{\gamma-2}) : S_0 \subseteq E_\gamma(t - 1) \wedge S_0 \cup \mathcal{T} \text{ is a } \gamma - \text{matching}\}$. We will show that $|\mathcal{M}| = |M(t, S_1, S_2, \ldots, S_{\gamma-1})|$.

$\geq$ We claim that S only contains γ-matchings of the restriction of L to time instants between 0 and $t + \gamma - 3$, where every $\mathcal{M}' = M(t - 1, S_0, S_1, \ldots, S_{\gamma-2}) \in S$ is a γ-matching such that $\mathcal{M}' \cap E_\gamma(t + i) = S_i$ for every $0 \leq i \leq \gamma - 2$. This is

---
[6] It is implemented in function `base_case` in https://github.com/Talessseed/Temporal-matching-of-historical-and-geometric-graphs/blob/master/approx.py

entirely deduced from the induction hypothesis. Thus, because every $\mathcal{M}' \in S$ has only edges living in times between 0 and $t+\gamma-3$, $M(t, S_1, S_2, \ldots, S_{\gamma-1}) \cap E_\gamma(t+\gamma-2) = S_{\gamma-1}$. Moreover, $M(t, S_1, S_2, \ldots, S_{\gamma-1})$ is a γ-matching with only edges living in times between 0 and $t+\gamma-2$. Therefore by definition of $\mathcal{M}$, $|\mathcal{M}| \geq |M(t, S_1, S_2, \ldots, S_{\gamma-1})|$.

$\leq$ Let $S_0 = \mathcal{M} \cap E_\gamma(t-1)$. We can write $\mathcal{M} = S_{\gamma-1} \cup \mathcal{M}'$ such that $P(\mathcal{M}', t-1, S_0, S_1, \ldots, S_{\gamma-2})$ is satisfied. Note by induction hypothesis that we have $|\mathcal{M}'| = |M(t-1, S_0, S_1, \ldots, S_{\gamma-2})|$. Hence, it follows from the definition of $M(t, S_1, S_2, \ldots, S_{\gamma-1})$ that $|\mathcal{M}| = |S_{\gamma-1} \cup M(t-1, S_0, S_1, \ldots, S_{\gamma-2})| \leq |M(t, S_1, S_2, \ldots, S_{\gamma-1})|$.

We now address complexity issues. Checking if a set is a γ-matching can be done by Procedure 2. Note that $|\mathcal{M}| \leq \gamma\,\theta(E_\gamma)$. Hence, Procedure 2 terminates in time $O(\gamma^2 \theta(E_\gamma))$. Algorithm 1 iterates over t and the S_i's. Their number is exactly $t_{max} 2^{\gamma \theta(E_\gamma)}$.

Procedure 2: Procedure to check if $\mathcal{M}$ is a γ-matching

---

1   seen $\leftarrow \emptyset$
2   **for** $\Gamma = E_\gamma(t, uv) \in \mathcal{M}$ **do**
3   **for** $t' \in [\![t, t+\gamma-1]\!]$ **do**
4   **if** $(t', u) \in$ seen $\vee$ $(t', v) \in$ seen **then**
5   **return** false
6   seen $\leftarrow$ seen $\cup \{(t', u), (t', v)\}$
7   **return** true

Algorithm 2 (Approximation for γ-MATCHING on unit ball streams)
We keep the same notations as in the hypothesis of Lemma 4. If $d = 1$, we return the output of Algorithm 1. Otherwise, we compute the sets $M_i(s)$ with recursive calls on L but with positions in $\mathbb{R}^{d-1}$: we remove the x dimension by projecting every $\mathbf{c}_\Gamma$ on the hyperplane with equation $(x = 0)$: the input $\mathbf{c}_\Gamma = (x_\Gamma, y_\Gamma, z_\Gamma, \ldots)$ is replaced with $\mathbf{c}_\Gamma \leftarrow (y_\Gamma, z_\Gamma, \ldots)$. We return set $\gamma MM^\sim$ as defined in Lemma 4.

We stress on the use of variable $f_L \geq \rho$. Basically, if we call the approximation algorithm on a link stream with positions in $\mathbb{R}^p$, our algorithm will also use a similar value $f_L \leftarrow f_{p,L}$ with $f_{p,L} \geq \rho$. We define $f_{p,L} = q^{d-p+1} 2^{d-p-1} \frac{\log(m_\gamma)}{\gamma}$ in order to obtain the following result.

Theorem 1. *Algorithm 2 is polynomial in m_γ and is a PTAS for γ-MATCHING on unit ball streams of bounded velocity and density ρ embedded in an d-dimension space. More precisely, for any $q \geq 2\rho\gamma$, a γ-matching with approximation ratio $\left(1 - \frac{1}{\left\lfloor \frac{q \log(m_\gamma)}{2\rho\gamma} \right\rfloor}\right) \left(1 - \frac{1}{\lfloor q \rfloor}\right)^{d-1}$ can be computed in time $O^*(q^d 2^{d-1} m_\gamma^{q^d 2^{d-1}})$, where O^* only retains exponential factors.*

Procedure 3: Exact algorithm for the base case of the PTAS ($d = 1$)

---

1 Compute $E_\gamma(1), E_\gamma(2), \ldots, E_\gamma(\max(T(L)))$
2 Initialize M as $\emptyset$ for all elements
3 **for** $S_i \subseteq E_\gamma(i)$ *with* $1 \le i \le \gamma - 1$ **do**
4   **if** $S_1 \cup S_2 \cup \cdots \cup S_{\gamma-1}$ *is a γ-matching* **then**
5 $M(0, S_1, S_2, \ldots, S_{\gamma-1}) \leftarrow S_1 \cup S_2 \cup \cdots \cup S_{\gamma-1}$
6 **for** $1 \le t \le \max(T(L)) - \gamma + 1$ **do**
7   **for** $S_i \subseteq E_\gamma(t + i)$ *with* $1 \le i \le \gamma - 1$ **do**
8 $\mathcal{T} \leftarrow S_1 \cup S_2 \cup \cdots \cup S_{\gamma-1}$
9   **for** $S' \subseteq E_\gamma(t)$ **do**
10   **if** $S' \cup \mathcal{T}$ *is a γ-matching* **then**
11 $\mathcal{N} \leftarrow S_{\gamma-1} \cup M(t, S', S_1, \ldots, S_{\gamma-2})$
12 $M(t + 1, S_1, \ldots, S_{\gamma-1}) \leftarrow \max\{M(t + 1, S_1, \ldots, S_{\gamma-1}), \mathcal{N}\}$
13 $t \leftarrow \max\{-1, \max(T(L)) - \gamma + 1\}$
14 $\mathcal{M} \leftarrow \emptyset$
15 **for** $S_i \subseteq E_\gamma(t + i)$ *with* $1 \le i \le \gamma - 1$ **do**
16 $\mathcal{M} \leftarrow \max\{\mathcal{M}, M(t + 1, S_1, S_2, \ldots, S_{\gamma-1})\}$
17 **return** $\mathcal{M}$

Proof. First, we must verify our assumptions on the new transformed vertices each time we do the reduction of dimension. To do so, we need to show that if two γ-edges have their transformed normalized centers at distance strictly greater than 1, they must be independent. We prove this by induction on the dimension p. In dimension d, this is proven following the bounded velocity of the unit ball stream, *cf.* Proposition 1. If the dimension p is strictly smaller than d, we suppose we have proven what we want in dimension $p + 1$. Let Γ and Γ' be two independent γ-edges with normalized centers in dimension $p + 1$ $(\overline{x_\Gamma}, \overline{y_\Gamma}, \overline{z_\Gamma}, \ldots)$ and $(\overline{x_{\Gamma'}}, \overline{y_{\Gamma'}}, \overline{z_{\Gamma'}}, \ldots)$ respectively. We suppose that $\|(\overline{y_\Gamma}, \overline{z_\Gamma}, \ldots) - (\overline{y_{\Gamma'}}, \overline{z_{\Gamma'}}, \ldots)\| > 1$ in $\mathbb{R}^p$. But then, it also holds that $\|(\overline{x_\Gamma}, \overline{y_\Gamma}, \overline{z_\Gamma}, \ldots) - (\overline{x_{\Gamma'}}, \overline{y_{\Gamma'}}, \overline{z_{\Gamma'}}, \ldots)\| > 1$ in $\mathbb{R}^{p+1}$. This contradicts the induction hypothesis.

We call the algorithm on a link stream L with positions in $\mathbb{R}^d$ initially. L has m_γ γ-edges. Let ρ_p $(p \le d)$ be the maximum density of link stream the algorithm processes with positions in $\mathbb{R}^p$. We claim that $\rho_p \le 2f_{p+1,L} = q^{d-p}2^{d-p-1}\frac{\log(m_\gamma)}{\gamma}$ for $p < d$. Indeed, if the algorithm processes a link stream with positions in $\mathbb{R}^p$, then for each $M_i(s)$ considered, we have that $\partial\rho(M_i(s)) \le \rho_{p-1}$, because the partial density is actually a density where one dimension is forgotten. We conclude with Lemma 3. The conditions on $f_{p,L}$ are also satisfied for $p < d$. Indeed, $f_{p,L} = 2qf_{p+1,L} \ge q\rho_p \ge \rho_p$.

We first address the approximation ratio. We suppose w.l.o.g. that $\log m_\gamma \ge 1$. Recall in Lemma 3 that k was chosen to satisfy $k \le \left\lfloor \frac{f_L}{\rho} \right\rfloor$. For a link stream L' with positions in $\mathbb{R}^p$, we choose for the algorithm $k = \left\lfloor \frac{f_{p,L}}{\rho'} \right\rfloor$ where ρ' is the density of L'. Therefore, when the algorithm is called on a link stream with positions in $\mathbb{R}^p$, we have for $p < d$ that $k = \left\lfloor \frac{f_{p,L}}{\rho'} \right\rfloor \ge \left\lfloor \frac{f_{p,L}}{\rho_p} \right\rfloor \ge \left\lfloor \frac{f_{p,L}}{2f_{p+1,L}} \right\rfloor \ge \lfloor q \rfloor$.

Hence, $1 - \frac{1}{k} \geq 1 - \frac{1}{\lfloor q \rfloor} \xrightarrow[q \to +\infty]{} 1$. Combining this with Lemma 4 implies:

$$\left(1 - \frac{1}{\left\lfloor \frac{f_{d,L}}{\rho} \right\rfloor}\right) \prod_{p=1}^{d-1} \left(1 - \frac{1}{\left\lfloor \frac{f_{p,L}}{\rho_p} \right\rfloor}\right)$$

$$\geq \left(1 - \frac{1}{\left\lfloor \frac{q \log(m_\gamma)}{2\rho\gamma} \right\rfloor}\right) \prod_{p=1}^{d-1} \left(1 - \frac{1}{\lfloor q \rfloor}\right)$$

$$= \left(1 - \frac{1}{\left\lfloor \frac{q \log(m_\gamma)}{2\rho\gamma} \right\rfloor}\right) \left(1 - \frac{1}{\lfloor q \rfloor}\right)^{d-1} \xrightarrow[q \to +\infty]{} 1 \text{ if } q \geq 2\rho\gamma \text{ and } \log m_\gamma \geq 1.$$

Finally, we show that Algorithm 2 is polynomial when q is fixed. Since the computations when removing a dimension are polynomial in m_γ, it is left to prove that Algorithm 1 also solves the base case in time polynomial in m_γ. We note that $\theta(E_\gamma) \leq \rho_0 \leq q^d 2^{d-1} \frac{\log(m_\gamma)}{\gamma}$, and combine it with Lemma 5 to obtain:

$$O\left(t_{max} \gamma^2 \theta(E_\gamma) 2^{\gamma\theta(E_\gamma)}\right)$$

$$= O\left(t_{max} \gamma^2 q^d 2^{d-1} \frac{\log(m_\gamma)}{\gamma} 2^{\gamma q^d 2^{d-1} \frac{\log(m_\gamma)}{\gamma}}\right)$$

$$= O\left(t_{max} \gamma q^d 2^{d-1} \log(m_\gamma) 2^{q^d 2^{d-1} \log(m_\gamma)}\right)$$

$$= O\left(t_{max} \gamma q^d 2^{d-1} \log(m_\gamma) m_\gamma^{q^d 2^{d-1}}\right)$$

Notice that we can suppose w.l.o.g that each frame of γ consecutive time instants contains a time-vertex itself contained in a γ-edge. Indeed, if that is not the case, we can delete time instants where no γ-edges exist, without making two γ-edges that were independent dependent. Hence, $t_{max} = O(\gamma m_\gamma)$, implying the algorithm is polynomial in m_γ when q is fixed. Whence, the algorithm is a PTAS of bounded ρ for all q such that $q \geq 2\rho\gamma$.   □

4   Exact algorithm for arbitrary link streams

For later use in the numerical analysis, we need to find an optimal solution for γ-MATCHING. This section presents an FPT solution for γ-MATCHING parameterized by the vertex number. Without being pushy about the time complexity "in the big O", we are demanding on good runtime performance. Our implementation performs well because with temporal graphs, the vertex number is a very small FPT parameter compared to the more popularly used size of the output.

We shall store in $M(t, A_1, A_2, \ldots, A_\gamma)$, for every $t \in [\![1, t_{max} - \gamma + 1]\!]$, a maximum γ-matching of the restriction of link stream L to time instants between 0 and $t + \gamma - 1$, where we remove all timed edges adjacent to the time-vertices $(t + i - 1, u)$ for all $u \in A_i$. Intuitively, A_i is the set of time vertices at time

$t + i - 1$ that are endpoints of already used γ-edges beginning at $t + i - 1$. The recursion for M is:

- $M(-1, A_1, A_2, \ldots, A_\gamma) = \emptyset$
- $M(t, A_1, A_2, \ldots, A_\gamma) = \max\bigg(\bigg\{M(t-1, \emptyset, A_1, A_2, \ldots, A_{\gamma-1})\bigg\} \cup$

$$\bigg\{\{\Gamma\} \cup M(t, A_1 \cup \{u,v\}, A_2, \ldots, A_\gamma) : \Gamma = E_\gamma(t, u, v) \subseteq L \wedge u, v \notin \bigcup_{i=1}^{\gamma} A_i\bigg\}\bigg).$$

Lemma 6 ($\star$). *Keeping the same notations introduced before, $M(t_{max} - \gamma + 1, \emptyset, \ldots, \emptyset)$ is a maximum γ-matching of L.*

Theorem 2 ($\star$). *On any n-vertex, m-recorded-edge link stream L with γ-edge number $m_\gamma = |E_\gamma(L)|$, γ-MATCHING can be solved in time $O(m + n^2 + m_\gamma(\gamma+1)^n)$ by a dynamic programming filling the above described table M. At the end of the process, $M(t_{max} - \gamma + 1, \emptyset, \ldots, \emptyset)$ stores a maximum γ-matching of L, where $t_{max} = \max(T(L))$.*

5 Numerical analysis

We implement[7] in Python 3 both the PTAS described in Section 3 and the DP described in Section 4. We compare their numerical results with the JavaScript implementation of the greedy approximation[8] given in [5]. In the latter reference, the authors use an arbitrary total ordering to sort the γ-edge set, then step by step try to add each γ-edge to the matching if it does not conflict with the others that are already present in the matching.

Our experiments are run on a standard laptop with i5 6300HQ 4 cores @2.3GHz and 8Gb memory @2133Mhz. We place automatic timeouts so that the computation stops after spending 100 (one hundred) seconds on an instance. In general, the greedy implementation returns instantaneously, while the PTAS and the DP take much more time. All our computations add up to ≈ 400 hours CPU time over the 4 cores. In what follows, we totally skip the discussion about computing time, and solely focus on approximation ratio.

5.1 PTAS vs. greedy

Theoretically, the greedy implementation is a 2-approximation [4]. The theoretical approximation ratio of the PTAS is as follows: it is 1.27 for unit interval streams of velocity 5 and density 5; and is 1.38 for unit disk streams of velocity 2 and density 4. Both values are very far from the theoretical ratio of the greedy implementation. Accordingly, we would like to confirm that information

---

[7] Our source code is available at https://github.com/Talessseed/Temporal-matching-of-historical-and-geometric-graphs

[8] The source code of [5] is available at
https://github.com/antoinedimitriroux/Temporal-Matching-in-Link-Streams
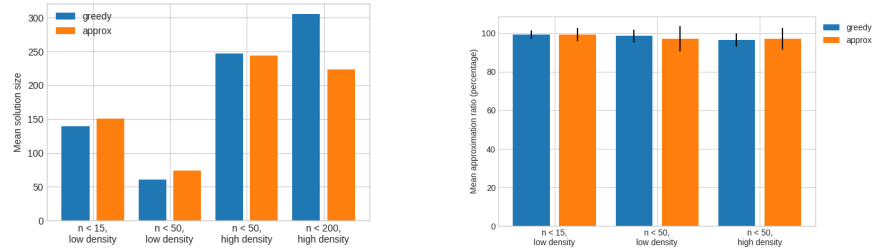
Fig. 1. (Left) Mean of the outputted values of PTAS vs. greedy on 4 generated datasets of unit ball streams; (Right) Mean and standard deviation of the approximation ratio of PTAS vs. greedy, when compared to the optimal values, on $\approx 90\%$ of the datasets in the left figure. In one of the 4 datasets, we do not have any reliable approximation ratio because the optimal computation runs out of time on most instances.

on artificially generated data. In the following, we choose the value of q to be the biggest integer from $\left\lceil \frac{2\gamma\rho}{\log(m_\gamma)} \right\rceil$ to $\left\lceil \frac{2\gamma\rho}{\log(m_\gamma)} \right\rceil + 10$ that does not result in a timeout.

Hypothesis 1: PTAS finds better solution than greedy on unit ball streams of well chosen velocity and density. Our methodology is to generate four different datasets, *cf.* Figure 1 (left). We run both implementations on these, and record the outputted γ-matching size.

Discussion: Hypothesis 1 is not confirmed in our setting. Our experimental results are in favour of the greedy implementation, which performs $\approx 10\%$ better than the PTAS, especially when the density is high. This is surprising in the sense that very good conditions for PTAS are met: low dimension of the Euclidean space (good runtime), controlled velocity and density (good approximation ratio), and varying number of vertices (to rule out the help of kernelization [4,5], at least on parts of the dataset). While it does not completely refute *Hypothesis 1* in other settings, our numerical analysis implies at least two possibilities: either both implementations find solutions half the size of the optimal values, or their solutions are near optimal (or they are somewhere in between).

Hypothesis 2: both approximation factors in previous experiment are close to optimal. We address the dynamic programming explained in Section 4. Since the runtime of its implementation is very long, we can only obtain the answer for $\approx 90\%$ of the datasets described in the above experiment. Our results are presented in Figure 1 (right).

Discussion: likely validation of Hypothesis 2. Our experimental results show that PTAS and greedy find solutions that are more than $\approx 95\%$ of the optimal values, with a deviation less than $\approx 5\%$ of the mean, on $\approx 90\%$ of the unit ball streams presented in the previous experiment. We conclude that Hypothesis 2 seems to be valid on our generated input.

6 Conclusion and perspectives

We introduce the notion of velocity in a temporal geometric graph. Revisiting van Leeuwen theorem on static geometric graphs of bounded density [29, Theorem 6.3.8, page 74], we extend their PTAS to temporal geometric graphs of bounded density and bounded temporal velocity. Our study case is γ-MATCHING [4], a temporal version of (static) graph matching [13]. Implementation works show that a known greedy implementation [5] finds better approximated solutions by a factor of $\approx 10\%$, when compared to the PTAS. Theoretically, the approximation factor is 2 for the greedy algorithm and between 1.27 and 1.38 for the PTAS on our datasets. This raises the question whether the greedy factor 2 is tight on temporal geometric graphs. As a byproduct for obtaining parts of the above numerical analysis, we devise a simple dynamic programming formula solving optimally the general case in FPT time parameterized by the number of vertices. Since vertices are in small number compared to recordings of edges in a temporal graph, our dynamic programming could be of independent practical interest.

References

1. XP. http://www.extremeprogramming.org.
2. Airbus Industrie. Fello'fly demonstrator. Dubai Airshow 2019.
3. E.C. Akrida, G.B. Mertzios, P.G. Spirakis, and V. Zamaraev. Temporal vertex cover with a sliding time window. *Journal of Computer and System Sciences*, 107:108–123, 2020.
4. J. Baste and B.-M. Bui-Xuan. Temporal matching in link stream: kernel and approximation. In *16th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*, 2018.
5. J. Baste, B.-M. Bui-Xuan, and A. Roux. Temporal matching. *Theoretical Computer Science*, 806:184–196, 2020.
6. B.-M. Bui-Xuan, A. Ferreira, and A. Jarry. Computing shortest, fastest, and foremost journeys in dynamic networks. *International Journal of Foundations of Computer Science*, 14(2):267–285, 2003.
7. A. Casteigts, P. Flocchini, E. Godard, N. Santoro, and M. Yamashita. Expressivity of time-varying graphs. In *19th International Symposium on Fundamentals of Computation Theory*, pages 95–106, 2013.
8. 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.
9. J. Dibbelt, T. Pajor, B. Strasser, and D. Wagner. Connection scan algorithm. *ACM Journal of Experimental Algorithmics*, 23, 2018.
10. R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer, 1999.
11. 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.

12. 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.
13. J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
14. T. Erlebach, M. Hoffmann, and F. Kammer. On Temporal Graph Exploration. In *42nd International Colloquium on Automata, Languages, and Programming*, volume 9134 of *LNCS*, pages 444–455, 2015.
15. L. Foschini, J. Hershberger, and S. Suri. On the complexity of time-dependent shortest paths. *Algorithmica*, 68(4):1075–1097, 2014.
16. D. Kempe, J. Kleinberg, and A. Kumar. Connectivity and inference problems for temporal networks. *Journal of Computer and System Sciences*, 64(4):820–842, 2002.
17. B. Klimt and Y. Yang. Introducing the Enron Corpus. In *CEAS*, 2004.
18. M. Latapy, T. Viard, and C. Magnien. Stream graphs and link streams for the modeling of interactions over time. *Social Network Analysis and Mining*, 8(61), 2018.
19. T.A. McKee and F.R. McMorris. *Topics in Intersection Graph Theory*. SIAM Monographs on Discrete Mathematics and Applications, 1999.
20. G.B. Mertzios, H. Molter, R. Niedermeier, V. Zamaraev, and P. Zschoche. Computing Maximum Matchings in Temporal Graphs. In *37th International Symposium on Theoretical Aspects of Computer Science*, volume 154 of *LIPIcs*, pages 27:1–27:14, 2020.
21. G.B. Mertzios and P.G. Spirakis. Strong bounds for evolution in networks. *Journal of Computer and System Sciences*, 97:60–82, 2018.
22. S. Miyazaki. On the advice complexity of online bipartite matching and online stable marriage. *Information Processing Letters*, 114(12):714–717, 2014.
23. M. Mucha and P. Sankowski. Maximum matchings via Gaussian elimination. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 248–255, 2004.
24. M. Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.
25. N. Robertson and P.D. Seymour. Graph minors. I. Excluding a forest. *Journal of Combinatorial Theory, Series B*, 35(1):39–61, 1983.
26. F.J. Ros, P.M. Ruiz, and I. Stojmenovic. Acknowledgment-based broadcast protocol for reliable and efficient data dissemination in vehicular ad-hoc networks. *IEEE Transactions on Mobile Computing*, 11(1):33–46, 2012.
27. 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.
28. I. Tsalouchidou, R. Baeza-Yates, F. Bonchi, K. Liao, and T. Sellis. Temporal betweenness centrality in dynamic graphs. *International Journal of Data Science and Analytics*, pages 1–16, 2019.
29. E.J. van Leeuwen. *Optimization and Approximation on Systems of Geometric Objects*. PhD thesis, Utrecht University, 2009.
30. 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.
31. 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.