Building Graphs at Scale via Sequence of Edges: Model and Generation Algorithms

Yu Liu, Lei Zou, and Zhewei Wei

Abstract—Real-world graphs exhibit many interesting properties that differentiate them from random graphs, which have been extensively studied for the past decades. For various proposed generative models, a majority of them build the graph by sequentially adding each node and the attached edges. However, the growth of many real-world graphs, such as social networks, is naturally modeled by the sequential insertion of edges. Unfortunately, to the best of our knowledge, no generative model has been proposed to reveal this process.

We propose the first sequence-of-edges model, denoted as *temporal preferential attachment (TPA)*. It relies on *preferential attachment (PA)*, one of the most influential mechanisms to generate scale-free graphs, and takes time-decay effect and node fitness into consideration. Empirical analysis demonstrates that our model preserves several key properties of the real-world graphs, including both the properties observed from the snapshot graphs (e.g., power-law distribution) and temporal properties observed from the graph generation process (e.g., shrinking diameter). Meanwhile, our model is sufficiently general to accommodate several forms of time decay and fitness distributions. Then, we design two efficient algorithms that generate TPA graphs with billions of edges in several minutes.

Index Terms—Complex graph model, sequence-of-edges graph, temporal preferential attachment, massive graph generation.

1 Introduction

Graphs are widely used to model the relationships between objects in various applications, such as websites [1], social networks [2] and knowledge graphs [3]. Some of the real-world graphs, such as social networks and graph streams, exhibit structural properties that are fundamentally different from those of random graphs, e.g., the Erdős and Rényi's graphs [4]. Tracing back to the pioneering work [5] in early 20th century, considerable research has been devoted to the study of the properties of real-world graphs, with new observations and understandings continuously arising in the past ten years. These findings not only further our understanding of graph theory, but also change the way we design graph algorithms and systems. For example, the fact that the degrees of real-world graphs follow heavy-tailed distribution facilitates a number of efficient graph algorithms [6], [7]. On the other hand, the same property poses new challenges for graph-parallel systems [8].

The study of structural properties on real-world graphs can be broadly divided into three categories. First, there exists many early work [9], [10], [11] that focus on making observations for various structural properties on real-world graphs. Second, based on these observations, a significant amount of research [12], [13], [14], [15] tries to propose complex graph models that explain the observed properties. Finally, to incorporate the growing need of large synthetic graphs [16], [17], a few recent work [18], [19], [20], [21] focuses on designing efficient algorithms to generate synthetic graphs that look like real-world graphs. State-of-the-art algorithms can generate a billion-edge graph on a commodity machine [21], or a trillion-sized graph in the distributed environment [20]. This not only avoids the privacy concerns of real-world data, but also facilitates the evaluation of algorithms and systems for large graphs.

• *Y. Liu and L. Zou are with Peking University, Beijing 100871, China. E-mail: {dokiliu,zoulei}@pku.edu.cn.*
• *Z. Wei is with Renmin University of China, Beijing 100872, China. E-mail: zhewei@ruc.edu.cn. The corresponding author of this work is Lei Zou.*

Motivations. As pointed by [13], growth is the very important property of real-world graphs. Nonetheless, graphs generated by various applications exhibit fundamentally different growth patterns. For citation networks and Wikipedia graphs, they evolve in a *node-centric* way, i.e., the graph expands by additions of nodes (e.g., papers, entities) and the attached edges. We formally define this process as follows.

Definition 1 (*Sequence-of-nodes* graph)**.** *The generation process of a sequence-of-nodes graph G is defined as $G = (G_1, \ldots, G_k, G_{k+1}, \ldots)$, where*

$$G_{k+1}.V = G_k.V \cup \{v_{k+1}\},$$
$$G_{k+1}.E = G_k.E \cup \{(v_{k+1}, u), u \in S_{k+1}\}.$$

Here v_{k+1} denotes a new node not belonging to G_k, G_k denotes a snapshot graph, and $G_k.V$ (resp. $G_k.E$) represents the set of nodes (resp. edges) of G_k. For $k \in [1, \infty)$, the graph expands by adding node v_{k+1} to the current graph G_k with a set of attached edges. Denote by S_{k+1} the set of the other endpoint of each edge, it holds that $S_{k+1} \subseteq G_k.V$.

In general, we do not exclude the possibility of adding edges between existing nodes upon inserting v_{k+1}. To this end, the set of inserted edges are represented by $\{(v_{k+1}, u), u \in S_{k+1}\} \cup \{(u_{i_1}, u_{j_1}), \ldots, (u_{i_l}, u_{j_l})\}$, where u_is and u_js belong to $G_k.V$. We refer to the former (resp. latter) set of edges as *external* (resp. *internal*) edges.

On the other hand, graphs such as social networks and communication networks can not be properly modeled by the above process, as they all grow in an *edge-centric* way. For example, social network grows by building new edges (e.g., friendship, the follow relationship) between nodes (i.e., users). However, each newly established interaction does not necessarily involve the addition of new users. In fact, interaction among existing users evolves over time and composes a large fraction of edges in the network. Another example is the graphs that can be represented by a sequence of time-stamped edges, such as graph streams. It is natural to model these graphs by a *sequence-of-edges* manner.

Definition 2 (*Sequence-of-edges* graph)**.** *The generation process of a sequence-of-edges graph is defined as* $G = (G_1, \ldots, G_k, G_{k+1}, \ldots)$*, where*

$$G_{k+1}.V = G_k.V \cup \{u_{k+1}\} \cup \{v_{k+1}\},$$
$$G_{k+1}.E = G_k.E \cup \{(u_{k+1}, v_{k+1})\}.$$

For $k \in [1, \infty)$*, the graph grows by adding an edge* (u_{k+1}, v_{k+1}) *to the current graph* G_k*. Note that the two endpoints* u_{k+1} *and* v_{k+1} *do not have to be nodes in* $G_k.V$*, and can be newly added ones.*

Unfortunately, very few works have put their emphasis on the sequence-of-edges modeling of real-world graphs, if they exist. The classical Barabási-Albert (BA) model [13] is regarded as one of the most influential sequence-of-nodes models among many other preferential attachment-based ones [22], [23]. Meanwhile, a few prevalent graph generation mechanisms, such as the recursive graph model [24], [25] and the hyperbolic unit disk model [14], need to fix the number of nodes (and edges) in advance as the Erdős-Rényi (ER) model [4] and thus ignore the growth process. Although follow-up literature [26] improves the dynamicity to some extent, they are inherently unsuitable to model temporal or streaming graphs. Instead, we aim to propose a sequence-of-edges model by non-trivially integrating several key ingredients of the PA-based model, which is powerful enough to explain the well-recognized properties of real-world graphs, and yet simple enough so we can design highly scalable algorithms that generate large synthetic datasets.

Contributions. Our principal contribution in this paper is summarized as follows.

- We propose temporal preferential attachment (TPA), the first sequence-of-edges generative model that captures the edge-centric growth over a wide range of real-world graphs, such as social networks and communication networks. Our model relies on the well-studied ingredients including degree-based preferential attachment [13], [27], the time-decay effect [22], [28], and node fitness [23], [29], and we non-trivially integrate them into the sequence-of-edges framework. Besides, our model is sufficiently general to accommodate various forms of aging and fitness. We also observe the existence of the TPA mechanism via empirical analysis of two real-world graphs.
- We design two efficient generation algorithms for the TPA model, by integrating the idea of logarithmic binning and state-of-the-art generation techniques for BA graph [21], [30]. We theoretically prove that our algorithm has asymptotically lower or the same time and space complexity compared to the algorithms for the simple BA model. Moreover, our algorithms can be naturally applied to several recent preferential attachment-based models with aging [22], [23], [29]. We believe this is the very first work to investigate the fast generation of these models.
- We conduct extensive empirical analysis to study the properties of our TPA model and the efficiency of the generation algorithm. Experimental results demonstrate that the TPA model preserves several critical properties of real-world graphs, such as power-law distribution and shrinking diameter [12]. We also conduct parameter sensitivity analysis to show the robustness and generality of our model. Finally, our generation algorithms manage to build billion-edge graphs within several minutes, which is comparable to state of the art of the BA model.

TABLE 1
Table of notations.

Notation	Description
G	graph represented by a sequence of time-stamped edges $e_1, e_2, \ldots$
G_t	induced subgraph of G by edges of timestamp $\leq t$
$e = (u, v, t(e))$	a time-stamped edge
$d(v)$	the degree of a node v
$t(v), t(e)$	the time of a node v or edge e
t	current time
γ	power law exponent [31]
$f(\cdot), g(\cdot)$	the degree-based (resp. temporal) PA function
β_d, β_t	the degree-based (resp. temporal) PA parameter
$h(v)$	fitness of node v
$\mathcal{D}_h$	the fitness distribution, parameterized by β_f and h_{max}

Table 1 lists notations that are frequently used in the remainder of the paper.

2 Related Work

There exists an abundance of existing work from observation of real-world graph phenomenon to the generation of massive synthetic graphs. We categorize them as follows.

2.1 Complex Graph Models

2.1.1 Classical preferential attachment-based models

Preferential attachment (PA) [13] is arguably one of the most influential mechanisms to generate graphs of heavy-tailed distribution due to its succinctness and interpretability. Generally speaking, when a new node is added, preferential attachment determines how to connect edges from it to existing nodes based on their attractiveness. Based on the assumption, [13] proposes the Barabási-Albert (BA) model which generates random power-law graphs (referred to as BA graphs) using the following two steps.

Step 1. Start with a small graph of m_0 *vertices generated randomly.*

Step 2. At every time step, add a new vertex with m *edges connecting to* m *distinct vertices already present in the graph. For each connection the selection of the existing vertex is governed by the following equation:*

$$\Pr[v \text{ is attached}] = \frac{d(v)}{\sum_w d(w)}. \tag{1}$$

Here the summation takes consideration of all existing nodes in the graph. It has been proved that the BA model generates power-law graphs with exponent $\gamma = 3$, i.e., $\Pr[d(v) = k] \propto k^{-3}$.

Despite its simplicity, the popular BA model has several limits. First, the graph does not contain nodes of degree less than m as opposed to most real-world graphs, where a large portion of the nodes have smaller degrees (e.g., one or two). Therefore, a few follow-up work [32], [33] proposes PA-based models with a random number of initial edges. In particular, when a new node v is inserted to the graph, the number of attached edges follows Poisson distribution [33] (i.e., $\Pr[d_{init}(v) = k] \propto \frac{\lambda^k e^{-k}}{k!}$) or the power law [32] (i.e., $\Pr[d_{init}(v) = k] \propto k^{-\gamma_{init}}$). Second, [27] considers more generalized PA functions and proposes the notion of *non-linear* preferential attachment:

$$\Pr[v \text{ is attached}] = \frac{d(v)^{\beta_d}}{\sum_w d(w)^{\beta_d}}, \tag{2}$$

where β_d is referred to as the preferential attachment exponent. However, they prove that the power-law distribution is found only when $\beta_d = 1$. When $\beta_d > 1$, a single node connects to nearly all other nodes; while for $\beta_d < 1$, the degree distribution changes to stretched exponential.

For the above models, preferential attachment is only applied for the addition of nodes. More precisely, since one endpoint of each edge is the new node and only the other one is chosen preferentially, we say the PA mechanism is *one-sided*. On the other hand, [34] discusses the *two-sided* PA for the addition of internal edges between two existing nodes v_i and v_j. They also consider the transitivity, which denotes the effect of the common neighbors of v_i and v_j:

$$\Pr[v_i \text{ and } v_j \text{ build a new edge}] \propto \tag{3}$$
$$f(d(v_i)) \cdot f(d(v_j)) \cdot f'(|N(v_i) \cap N(v_j))|). \tag{4}$$

Here $f(\cdot)$ (and $f'(\cdot)$) represents the degree-based PA function.

2.1.2 PA-based models with aging and fitness

The aging phenomenon has been observed in graphs such as citation networks and analyzed by some existing literature [35], [36], [37]. In particular, [22], [28] integrate the time-decay effect[1] to the PA mechanism. For example, a few literature assumes the PA function as follows:

$$\Pr[v \text{ is attached}] \propto f(d(v)) \cdot g(\Delta t(v)), \tag{5}$$

where $f(d(v))$ is a generalized PA function based on node degree, while $g(\cdot)$ describes the time-decay effect that a node is less attractive as time goes on, which can be power law [22] (i.e., $g(\Delta t(v)) = \Delta t(v)^{-\beta_t}$) or exponential [28] (i.e., $g(\Delta t(v)) = e^{-\beta_t \Delta t(v)}$). Note that the definition of time decay depends on the assumption that the graph is generated via the sequential addition of nodes and edges. For real-world graphs, let t denote the time at present, then $\Delta t(v)$ can be defined as $t - t(v)$, where $t(v)$ is the timestamp at which node v first appears. For synthetic graphs, following almost all previous works, the nodes are numbered from 1 to n according to their insertion order. Then, at time t, we have $\Delta t(v_i) = t - i$. Also note that for simplicity of notation, we omit the subscript t for node degree and time decay throughout the paper.

Furthermore, recent studies [23], [29], [38] also consider *node fitness* to model the intrinsic differences between nodes that can not be simply represented by degree-based and temporal PA. Formally, the preferential probability is defined as follows:

$$\Pr[v \text{ is attached}] \propto f(d(v)) \cdot g(\Delta t(v)) \cdot h(v), \tag{6}$$

where $h(v)$ is the fitness of node v, a constant sampled from some specific distribution (e.g., exponential) upon node insertion. We refer to the model as *dynamic preferential attachment (DPA)*.

1. We will use these two terms interchangeably.

TABLE 2
Representative PA-based models and their function forms. (Specifications for different function forms are as follows. Power law: $f(x) = x^{-a}$; Exponential: $f(x) = e^{-ax}$; Poisson: $f(x) = a^x e^{-a}/x!$; Log-normal: $f(x) = e^{-a\log^2(x+1)}$, where a is the function parameter.)

Model	PA function	Aging form	Fitness
BA [13]	$f(d(v))$ (linear)	N/A	N/A
[27]	$f(d(v))$ (non-linear)	N/A	N/A
[22]	$f(d(v))g(\Delta t(v))$	Power law	N/A
[28]		Exponential	N/A
[29]	$f(d(v))g(\Delta t(v))h(v)$	Log-normal	(Fitted)
[38]		Power law/Exponential	Exponential
DPA [23]		Log-normal	(General)

Existing study [23] demonstrates that the power-law distribution can be restored by introducing fitness to strengthen the *old-get-richer* effect. Otherwise, the time decay implies old nodes should be less attractive, resulting in exponential degree distribution. We list in Table 2 the representative PA-based models, along with their concrete forms of the PA function and the fitness settings. Due to the theoretical result of [27], most of them assume linear preference based on degree, i.e., $f(d(v)) = a \cdot d(v) + b$. Following most existing literature, we only discuss the undirected graph models for simplicity, and the conclusions can be easily extended to its directed version.

2.1.3 Graph models based on other mechanisms

Except for preferential attachment, numerous complex graph models have been proposed according to various presumed or observed generation mechanisms, such as the recursive graph models [24], [25], the hyperbolic unit disk model [14], models with mixed strategies [12], [15], and static model [39] which takes the degree distribution as given, to name a few. Nonetheless, not all of them focus on the generative process, i.e., the sequential insertion of nodes and edges. For example, the recursive models [24], [25] are based on matrix products and thus do not focus on the growth of network over time. It is not straightforward to modify them to model the behaviors of real-world graphs related to the growth process.

2.2 Observations of Real-world Graphs

2.2.1 Properties of real-world graphs

We classify the properties observed in real-world graphs as the *static properties* and the *temporal properties* (or *dynamic properties*). We will use these two terms interchangeably.

Static properties. The properties observed by only one snapshot graph are referred to as static properties. For example, various distributions related to graph structures have been observed to approximately obey the power law, including the distribution of node degree [40], singular values and first singular vectors [24], the size of connected components [10], and etc. The most well-studied property is perhaps the power-law distribution of node degrees. First observed by [41] and re-discovered by [13], this property is extensively investigated by follow-up work, such as [9] and [31]. It states that on real-world graphs, the number of nodes of degree k is (approximately) in inverse proportion to k^γ, where $\gamma \in (1, \infty)$ denotes the *power-law exponent*.

Temporal properties. If some property is observed from a temporal graph that is defined by a sequence of edges or a sequence of snapshot graphs, it is called the temporal properties. Several properties of temporal graphs have been revealed, such as edge densification [12] and shrinking diameter. As the graph grows larger, edge densification states that the average degree of the graph will increase, while shrinking diameter suggests that the effective diameter will continuously decrease and finally reach a stable value. Another important temporal property is the aging phenomenon, which has been extensively investigated by existing literature. Intuitively, it says that the old nodes (i.e., node appears early in the graph) tend to be less attractive as the graph expands.

2.2.2 Growth mechanism of the real-world graphs

Among the proposed graph generation mechanisms to build complex graph models, several of them have been observed in real-world graphs, such as preferential attachment and the communities-within-communities [42] phenomenon. In particular, preferential attachment has been extensively investigated [43], [44] and its existence in many real-world graphs have been proved. Also note that in early literature of the PA-based models [13], [27], preferential attachment is *assumed* rather than observed. We briefly describe the observation procedure as follows. Let G_k be a snapshot graph, e.g., consisting all edges in the temporal graph whose timestamp is less than t. Then, we check a sequence of edges inserted after t, and for each edge connecting to nodes in G_k, we record the degree of that node. Then, we get a histogram to approximately reveal the node preference w.r.t. degree.

2.2.3 Measurement of graph statistics

Graph statistics such as average degree, effective diameter, and clustering coefficient [11] reveal the structural properties of a graph. KONECT [45] lists a variety of graphs statistics for real-world graphs. In particular, the power-law exponent γ is perhaps the most important statistics to model the skewed distribution of graphs. Noticing that the curve fitting method on the log-log plot of degree distribution is biased and lacks theoretical guarantee, Newman [31] first proposes a maximum likelihood estimation (MLE) method to compute γ. Follow-up work [46], [47] presents the testing method to judge if the distribution is more similar to power law than others, e.g, the log-normal distribution.

As for the estimation of the PA exponent β_d, early work either relies on curve fitting [43], [44] or presumes the form of the PA function (e.g., as polynomial [44], [48], [49]). Pham [50] and Inoue [34] lately propose the maximum likelihood-based method to estimate β_d. This method works very well for the observation of the one-sided degree-based preferential attachment, but suffers from efficiency problems when applied to more complicated PA functions (e.g., two-sided PA) and on large graphs.

2.3 Generative Algorithms for Scale-free Graphs

Recently, progress has been made on the fast generation of large synthetic graphs. In particular, [30] proposes the stochastic acceptance (SA) algorithm based on rejection sampling. [21] studies the generation of billion-edge BA graphs using a commodity machine. They propose ROLL, an efficient in-memory generation algorithm based on two optimizations. First, nodes of the same degree are placed into buckets. Therefore, the preferential node selection can be implemented as bucket selection followed by random sampling inside a bucket. Second, they introduce a binary search tree to speed up inter-bucket selection. On graphs larger than millions of nodes, ROLL is at least one order of magnitude faster than existing techniques such as SA [30]. Besides, a few works [17], [19] study preferential attachment in the parallel setting, while [51] discusses the approximate generation of BA graphs. However, all known algorithms are designed for the PA-based models where the PA functions are one-sided and only depend on node degree. It is not clear how these algorithms can be extended to PA-based models with aging while maintaining the efficiency. To the best of our knowledge, the more recent PA-based models [23], [28], [29], [38] still have the problem of scalable generation.

Except the PA-based models, a line of works [18], [20], [26], [52] consider the generation of massive graphs for recursive and hyperbolic unit disk models. Another closely related topic is graph upscaling [53], [54], which expands a given real-world graphs by some specific mechanism such as preferential attachment other than generate synthetic graphs from scratch. Nevertheless, they still model the graph generation in the sequence-of-nodes manner, and do not focus on the temporal effect in the growth process.

3 TPA: OUR SEQUENCE-OF-EDGES MODEL

3.1 Model Specification

In this section, we propose our temporal preferential attachment (TPA) model, aiming at reflecting the edge-centric growth of real-world networks. We formally describe the TPA model as follows.
Step 1. Start with a small random graph (e.g., ER graph) G_k, which consists k vertices $\{v_1, \ldots, v_k\}$. Note that the subscript k represents the number of nodes in the current graph. Place one virtual node *v_{k+1} outside of G_k. For simplicity, we can set $k = 1$.*
Step 2. At each time, add one edge $e = (u_1, u_2)$ between nodes $\{v_1, \ldots, v_k, v_{k+1}\}$.

- *For each $v \in \{v_1, \ldots, v_k\}$, the node preference, denoted by $tpa(v)$, is computed by a general PA function, which will be illustrated in Section 3.2. The preference of v_{k+1} is a constant $\alpha \in [1, \infty)$ given as model parameter.*
- *Both endpoints u_1 and u_2 of e is chosen in proportional to the node preference. As long as a self-loop is formed ($u_1 = u_2$), we re-sample u_2.*
- *If v_{k+1} is chosen as one endpoint of e, add it (and e) to the current graph, and place virtual node v_{k+2}. Otherwise, we only add the edge to the graph.*

We have the following specifications for the proposed model. First, the above process generates an undirected graph. Since the direction of the edge is, to a large extent, independent of the generation process, following previous works [13], [22], [23], [28], [29], we do not focus on the edge direction. Second, in the baseline model, we only place *one* virtual node each time. One may extend the model by placing a set virtual nodes, however, note that it is sufficient to consider only two cases, i.e., *the number of virtual nodes is either one or two*. This is because edges come sequantially, and each edge has exact two endpoints. To be precise, assume the preference of each virtual node is equal and computed by a general function $a(\cdot)$. Therefore, setting n_v virtual nodes is equivalent to setting only one virtual node of preference $n_v \cdot a(\cdot)$. For now, we do not discuss the case that contains two virtual nodes, where the new edge may connect them and we generate a graph with many connected components.

Note that we assume the preference of the virtual node be a constant, for the following reasons. Except for the simplicity, it reflects the fact that as the graph grows, the percentage of internal edges becomes larger, which naturally leads to edge densification and shrinking diameter. We do not exclude the possibility of more complicated choices, such as setting the preference as a slowly (e.g., sub-linear) increasing function of the graph size, but that is beyond the scope of this paper.

3.2 The Attachment Function

We demonstrate how to compute the preference of the existing nodes. Indeed, we adopt a general function as in [23], [29], [38]:

$$tpa(v) = f(d(v)) \cdot g(\Delta t(v)) \cdot h(v), \tag{7}$$

which consists three independent parts.

The degree-based PA function $f(\cdot)$ is a monotonic increasing function of node degree. By default, we set it as the power-law (i.e., polynomial) function:

$$f(d(v)) = d(v)^{\beta_d}, \tag{8}$$

where β_d is the preferential attachment exponent [27], [43], [44], satisfying that $\beta_d \in [0, \infty)$.

Similarly, the temporal PA function $g(\cdot)$ is a monotonic decreasing function of node age. The function has a general form and can be power law, exponential or log-normal decay. (Please refer to Table 2 for their definitions.) The function parameter $\beta_t \in [0, \infty)$ is referred to as the temporal preferential attachment exponent. We assume nodes are numbered $v_1, \ldots, v_k, \ldots$ by their insertion order. By supposing *nodes are inserted sequentially and at a steady rate*, which is implied by existing sequence-of-nodes models, the time decay of node v_k at time t is $t - k$. We use this assumption by default. Note that another option is to assume *edges come at a steady rate*, e.g., for temporal and streaming graphs. In this case, $t(v_k)$ is the number of edges in G_{k-1} when it is added to the graph, and $\Delta t(v_k)$ can be defined accordingly.

Finally, we use a general distribution $\mathcal{D}_h$ to generate node fitness, which can be power-law ($\Pr[h(v) = k] \propto k^{-\beta_f}$), exponential ($\Pr[h(v) = k] \propto e^{-k\beta_f}$) or Poisson distributed ($\Pr[h(v) = k] \propto {\beta_f}^k e^{-\beta_f}/k!$), where $\beta_f \in [0, \infty)$ is the model parameter and $h(v)$ is the fitness of v. We use another parameter h_{max} to limit the upper bound of node fitness, i.e., $h(v) \in \{1, \ldots, h_{max}\}$ for each node v. Note that we assume the node fitness $h(v)$ is an integer so that it can be efficiently generated. By default, we use the Poisson distributed fitness, because it is less skewed and condensation [55] is not obvious.

Remarks. Compared to the one-sided DPA model, we only need one extra parameter (i.e., α) to control the graph density. Nonetheless, our model is sufficiently general to accommodate several forms of $f, g,$ and h. As will be demonstrated in the empirical analysis, many combinations with a wide range of parameter values result in graphs exhibiting power-law distribution and several other static and temporal properties.

We also briefly explain why other prominent graph generation mechanisms can not be trivially extended to model temporal graphs. Consider the recursive graph models [24], [25] and the dynamic hyperbolic model [26]. Suppose the graph now contains n nodes $v_1, \ldots, v_n$. Inserting v_{n+1} to the graph means adding one row and one column to the matrix for recursive graph models, while for [26] the node is put to the hyperbolic space. Then, whether v_{n+1} has an edge to each existing node $v_i (i \in [1, n])$ is determined immediately by the underlying generation mechanism. Moreover, to check all possible edges, the cost is prohibitive for sizeable graphs.

3.3 Generation Algorithm

We describe our baseline graph generation algorithm for the TPA model. We refer to the algorithm as *TPA-U-RW*, where "U" denotes that the algorithm generates undirected graphs. For now, we first ignore node fitness, i.e., *assuming that $h(v)$ follows the uniform distribution*. We will discuss later how to integrate fitness to the generation algorithm.

Algorithm 1: Basic algorithm for the TPA model (*TPA-U-RW*)

```
Input: $n$, the degree-based PA function $f(\cdot)$ (and $\beta_d$), the temporal PA
       function $g(\cdot)$ (and $\beta_t$), $\alpha$
Output: An undirected graph $G_n$ of $n$ nodes
1  Initialize $G_1$ with a single node $v_1$;
2  $i = 1$;
3  while $i < n$ do
4      while true do
5          do
6              $fromNode \leftarrow$ RW-Select($G_{i-1}, f(\cdot), g(\cdot), \alpha$);
7              $toNode \leftarrow$ RW-Select($G_{i-1}, f(\cdot), g(\cdot), \alpha$);
8          while $fromNode == toNode$;
9          if $fromNode == v_i$ or $toNode == v_i$ then
10             $G_i \leftarrow G_{i-1} \cup v_i$ (and the corresponding edge) ;
11             break;
12         else
13             insert edge $(fromNode, toNode)$ into $G_{i-1}$;
14     $i++$;
15 return $G_n$;
```

Given the number of nodes n, the degree-based PA function $f(\cdot)$ and parameter β_d, the temporal PA function $g(\cdot)$ and parameter β_t, and the preference α for virtual nodes, *TPA-U-RW* generates an undirected TPA graph of n nodes. Here we follow the paradigm of previous PA-based models [13], [22], [23], which takes the number of nodes as input. In contrast, our algorithm can also takes the number of edges as input and generates a graph with unfixed number of nodes. However, these two configurations are equivalent when considering the properties of the generated graph. For similar reasons, by default we assume the nodes are numbered $1, \ldots n$ and come at a steady rate. Therefore, we have $t(v) = v$.

The intuition of the algorithm is straightforward: given a set of nodes with corresponding preferences, we can implement preferential node selection by the roulette wheel. The pseudo-code is illustrated in Algorithm 1. We initialize the graph with one single vertex v_1, denoted as G_1 (Line 1). Then, a sequence of edges is inserted into the current graph. For each edge, its two endpoints ($fromNode$ and $toNode$) are determined by roulette wheel selection, a procedure denoted as **RW-Select** (Lines 6-7). We eliminate self-loops in Line 8. Recall that the algorithm incorporates node insertion by placing a virtual node of preference α in roulette wheel selection. The algorithm terminates when n nodes have been inserted. Notice that $G_1, G_2, ..., G_n$ only represents different stages of the generated graph, rather than n different graph instances.

Procedure **RW-Select** chooses a node v as an endpoint according to the TPA attachment function, with pseudo-code shown in Algorithm 2. It implements the straightforward roulette wheel selection. First, it sums the preference of all nodes given the current graph G_k and the preference of the virtual node (Line 1). Note that the concrete forms and the corresponding parameters for degree-based and temporal PA functions are given as input. Then it randomly generates a floating point number $r \in [0, Sum)$ (Line 2) and finds the smallest i such that $\sum_{j=1}^{i} \frac{f(d(j))}{g(\Delta t(j))} \geq r$ and $i \leq k$ (Lines 5-10). Otherwise, a new node v_{k+1} is returned as default (Line 4). It can be proved that the selection probability of $v_i, i \in [1, k+1]$ is proportional to $tpa(v_i)$ defined in Equation 7.

Indeed, algorithm *TPA-U-RW* generates a multi-edged graph, as defined by our TPA model. To be precise, two nodes u and v

Algorithm 2: RW-Select

```
Input: current graph $G_k$, $f(\cdot)$ (and $\beta_d$), $g(\cdot)$ (and $\beta_t$), and $\alpha$;
Output: A randomly chosen node $v$ by our attachment function,
        $v \in [1, k+1]$
1  $Sum \leftarrow \sum_{u=1}^{k} \frac{f(d(u))}{g(\Delta t(u))} + \alpha$;
2  Uniformly generate a random number $r \in [0, Sum)$;
3  Initialize $\sigma \leftarrow 0$;
4  $chosenNode \leftarrow k+1$;
5  for $i = 1$ to $k$ do
6      if $\sigma + \frac{f(d(i))}{g(\Delta t(i))} \geq r$ then
7          $chosenNode \leftarrow i$;
8          break;
9      else
10         $\sigma \leftarrow \sigma + \frac{f(d(i))}{g(\Delta t(i))}$;
11 return $chosenNode$;
```

can be connected by multiple edges $e_1, \ldots, e_k$ inserted at different timestamp. This is a reasonable assumption for temporal graphs and graph streams. To generate a simple graph, we have two choices: (1) Remove all duplicated edges if the timestamp can be ignored to the application; or (2) We store all edges in the memory (e.g., as adjacency lists). Once an chosen edge already exists, we repeat the edge selection process (Lines 6-7 of Algorithm 1). It turns out that duplicated edges do not affect the graph properties qualitatively.

Generating TPA graph with varied node fitness. Note that the fitness value of each node v can be easily integrated to Lines 1,6,&10 of Algorithm 2. We assign a constant value $h(v)$ to node v upon its insertion, following the distribution $\mathcal{D}_h$. To speed up this process, we employ the Alias method, which takes $O(1)$ time for each node. The procedure is demonstrated in Algorithm 3.

Indeed, if we assume the degree-based PA function is power-law (i.e., polynomial), which is often the case, the node fitness can be processed together with the degree-based PA function. Suppose the node fitness follows some distribution $\mathcal{D}_h = \{p_1, \ldots, p_{h_{max}}\}$ where $p_i = \Pr[h(v) = i]$. We demonstrate that the product of degree-based PA function and fitness can be rewritten as

$$f(d(v)) \cdot h(v) = f(d(v) \cdot h'(v)), \tag{9}$$

where $h'(v)$ is the transformed fitness following another distribution $\mathcal{D}'_h$. Specifically, since $f(d(v)) = d(v)^{\beta_d}$, we have $h(v) = h'(v)^{\beta}_{d}$, and therefore $h'(v)$ follows distribution $\{p'_1, \ldots, p'_{h'_{max}}\}$ where $h'_{max} = \lceil h_{max}^{1/\beta_d} \rceil$ and $p'_i \propto p_i^{1/\beta_d}$. Let $w(v) = d(v) \cdot h'(v)$ be the weight of node v. Then, it is sufficient to replace $d(v)$ by $w(v)$ in Algorithm 2. When node v receives an edge, we increase its weight by $h'(v)$. As we will see in the next section, this facilitates the design of efficient generation algorithms.

Time complexity. It is easy to see that the complexity of Algorithm 1 is $O(nm)$, where n (resp. m) denotes the number of nodes (resp. edges) of the generated graph[2]. To be precise, each edge insertion invokes **RW-Select** twice, and the time complexity of **RW-Select** is $O(n)$.

Space complexity. For each node $v \in \{1, 2, ..., n\}$, we only need to store $d(v)$. Recall that $t(v) = v$ is recorded implicitly. Therefore the space cost of *TPA-U-RW* is $O(n)$.

2. Careful readers may notice that Algorithm 1 incurs an extra cost by avoiding self-loops. In practice, very few self-loops are generated thus the cost is negligible.

Algorithm 3: GenFitness

```
Input: $\mathcal{D}_h$, $\beta_f$, $h_{max}$, and node $v$
Output: The fitness value $h(v)$ for node $v$
1  Let $p_i$ denote $\Pr[h(v) = i]$, which can be computed by $\mathcal{D}_h$ (and with
   $\beta_f$), $\forall i \in \{1, \ldots, h_{max}\}$;
2  Construct Alias structure $\mathcal{A}$ for probability distribution
   $\{p_1, \ldots, p_{h_{max}}\}$;
3  Sample $k$ with probability $p_k$ and set $h(v) \leftarrow k$;
4  return $h(v)$;
```

4 Fast Generation Algorithm

4.1 Rationale

For practical use, it is essential to generate large-sized graphs in TPA model efficiently. Unfortunately, the time complexity of *TPA-U-RW* is $O(nm)$, which prevents it from massive graph generation. In practice, it takes nearly one hour to generate a TPA graph of 100,000 nodes with an average degree $\bar{d} = 10$.

Despite the existing optimization techniques [21], [30] for BA graph generation, the introduction of time decay inherently prevents a trivial application for the TPA model. For instance, the stochastic acceptance method [30] relies on maintaining the maximum degree at all times, which is trivial for BA graphs. As for the TPA model, after the insertion of a new node (or edge), all nodes should have their time updated, and the maximum TPA score has to be computed from scratch. [21] proposes a tree structure combined with *bucketing* in that for BA model, all nodes of identical degrees have the same preference. Obviously, it does not hold for TPA graphs. To alleviate the side effect of aging, we introduce *logarithmic binning*, with little modification on the definition of node time. Precisely, we place nodes into a sequence of bins, called T-Bucket, satisfying that (1)no more than two T-Buckets have the same bucket size, defined as the number of nodes inside; (2)a larger bucket is merged from two buckets each with half of its size; and (3)older nodes are placed in T-Bucket of larger size. The bucket list is maintained during graph generation. We explain it by the example in Fig. 1. Assume there have been already 10 nodes placed in 5 T-Buckets, as shown in Fig. 1(a). At time $t = 11$, a new node 11 is added (Fig. 1(b)). At present, there exist three T-Buckets of size 1. Thus, we merge the older two containing 9 and 10 respectively into one bucket with its size doubled (Fig. 1(c)). This could result in a cascading merge of larger T-Buckets, as illustrated in Fig. 1(d), until condition (1) is satisfied.

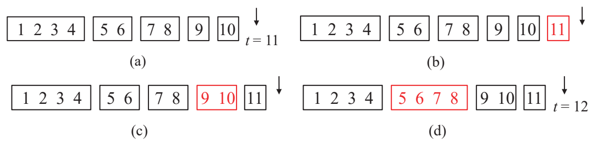


Fig. 1. Merging logarithmic bins.

To accommodate logarithmic binning, we slightly modify the definition of $\Delta t(v)$ in the TPA model. Recall that $\Delta t(v) = t - t(v)$, where $t(v) = v$ is the node id numbered from 1 to n. Denote by $\Delta t'(v)$ the size of T-Bucket to which v belongs. We have the following lemma.

Lemma 1. *Given a TPA graph G with nodes stored in a list of T-Buckets, for any $v \in V$ it satisfies that*

$$\Delta t(v)/4 < \Delta t'(v) < \Delta t(v), \tag{10}$$

where $\Delta t'(v) = |TB(v)|$, and $TB(v)$ is the T-Bucket containing v.

Proof. First, note that the size of any T-Bucket is some power of 2. Another observation is that, if a T-Bucket of size 2^i exists, there must be at least one bucket of size 2^j, for each $j \in [0, i)$. W.l.o.g. we assume v is in this T-bucket. The number of nodes coming after v in G is at least $\sum_{j=0}^{i-1} 2^j = 2^i - 1$, and at most $2\sum_{j=0}^{i} 2^j - 1 = 2^{i+2} - 3$, since placing one more node incurs cascading merging, which will put v into a bucket of size 2^{i+1}. □

Next, we will show how to improve the generation efficiency of the TPA model by incorporating logarithmic binning and optimizations for degree-based PA models.

4.2 The *TPA-U-SA* algorithm

According to the analysis above, to improve the efficiency of Algorithm 1, we should accelerate the node selection phase for each edge insertion. With the idea of logarithmic binning, every node selection by the TPA function can be decomposed into two steps, i.e., the *inter-bucket* selection which chooses a T-Bucket by its weight (i.e., the sum of preference of nodes inside), followed by the *intra-bucket* selection to finally choose a node. In this section, we propose the *TPA-U-SA* algorithm, which exploits the properties of the TPA model to speed up intra-bucket selection.

In [30], Lipowski and Lipowska proposed the stochastic acceptance (SA) algorithm. Generally speaking, given a set of events with probability distribution $p_1, p_2, ..., p_n$, to select an event based on the probabilities, first it selects an event uniformly at random, say event i. Second, with probability p_i/p_{max} it accepts the selection, or else the process is repeated until this condition is satisfied. The number of repetition is $O(p_{max}/\bar{p})$ in expectation, where $p_{max} = \max\{p_1, p_2, ..., p_n\}$ and $\bar{p} = \sum_{i=1}^{n} p_i/n$. It has been proved as state-of-the-art for generating small-sized BA graphs[3]. For instance, it is the most efficient algorithm to generate a BA graph of $n \leq 10^5$ and $\bar{d} = 2$.

Our TPA model favors the idea of stochastic acceptance (SA) because it always contains lots of T-Buckets of small size. Precisely, let τ be the threshold that SA can efficiently implement random selection from any power law distribution of smaller size. For a TPA graph of size n, there are at least $\log_2\tau$ T-Buckets of size less than τ, and at most $\log_2 n - \log_2\tau$ buckets of larger size. Even with $\tau = 10^5$, more than half T-Buckets should be optimized by SA selection for a billion-node graph. Secondly, for TPA graph, older and larger T-Buckets have considerably higher average degree than the newer and smaller buckets. As a consequence, the degree (i.e., probability) distribution inside each T-Bucket is less skewed. This could speed up the SA algorithm inside a T-Bucket.

The above discussion indicates that the SA algorithm for the TPA model could be more efficient than for the BA model. Our algorithm, denoted by *TPA-U-SA*, is based on the modified version of Algorithm 1. We list the modifications in Algorithm 4. First, we need a list to store all the T-Buckets with their corresponding weights. Specifically, the weight of T-Bucket TB is defined as the sum of TPA score for all nodes in TB: $w(TB) = \sum_{v \in TB} tpa(v)$. Denote by $tBucketList$ the list of T-Buckets, with newly inserted T-Bucket in front of the list. We maintain $tBucketList$ after each node insertion, by merging some buckets when necessary.

3. Another simple but quite efficient algorithm is as follows. First choose an edge randomly, followed by choosing one of its endpoints for attachment. We do not consider this method for two reasons. First, it has to store all edges in memory with space cost $O(m)$, whereas other methods only need $O(n)$ space. Second and more importantly, this method can not be extended to non-linear or temporal preferential attachment, which is common in real-world graphs.

Algorithm 4: Fast generation algorithm for TPA-U model

```
1. Insert after the 1st line of Alg. 1:
Initialize $tBucketList = \emptyset$ and add $TB_1$ with node $v_1$;
2. Insert after the 10th line of Alg. 1:
Initialize T-Bucket $TB_i$ with node $v_i$, add it to the front of
  $tBucketList$;
mergeTBucket($tBucketList$);
```

Algorithm 5: SA-Select

```
Input: $tBucketList$ of current graph $G_k$, $f(\cdot)$ (and $\beta_d$), $g(\cdot)$ (and
       $\beta_t$), and $\alpha$;
Output: A randomly chosen node $v$ by temporal preferential
        attachment, $v \in \{v_1, ..., v_{k+1}\}$
1  Uniformly generate a random number $r \in [0, 1)$;
2  if $r < \alpha/(\alpha + \sum_{TB \in tBucketList} w(TB))$ then
3      $chosenNode = v_{k+1}$;
4  else
5      Randomly select a T-Bucket $TB_i$ from $tBucketList$ according to
         its weight $w(TB_i)$, by roulette wheel and starting from the tail;
6      while true do
7          Select a node $v$ from $TB_i$ uniformly at random;
8          Uniformly generate a random number $r \in [0, 1)$;
9          if $r < f(d(v))/\max_{u \in TB_i} f(d(u))$ then
10             $chosenNode = v$;
11             break;
12 return $chosenNode$;
```

We replace the roulette wheel node selection (Algorithm 2) to the SA-based algorithm (**SA-Select**) shown in Algorithm 5, which has the same input and output. Also note that for simplicity, we do not include node fitness; it can be processed the same way as in Alg. 1. We first check if the virtual node is selected (Lines 1-3). If not, the algorithm selects a T-Bucket by roulette wheel (Line 5). We traverse *$tBucketList$ from the tail to the front*, since typically T-Buckets at the tail have larger weights and thus higher selection probability. Then, the intra-bucket node selection is implemented by stochastic acceptance (Lines 6-11) based on the degree factor, because all nodes in the same T-Bucket are considered having the identical timestamp.

At last, we specify the procedure for merging T-Buckets, i.e., **mergeTBuckets** in Alg. 4. Each time a new T-Bucket (of size one) is inserted, we check iteratively from the *front* of $tBucketList$ to see if there exist three consecutive buckets of the same size and merge the latter two. Otherwise, the procedure stops. Correctness can be easily proved by induction.

Space complexity. Note that $tBucketList$ can be efficiently implemented by an array of length n, while each T-Bucket records its start and end index. We denote by $D\text{-}Array$ for the data structure, which incurs $O(n)$ space cost.

Time complexity. We analyze the computation cost of the *TPA-U-SA* algorithm by focusing on the procedure **SA-Select**, which will be invoked $O(m)$ times for a TPA graph of m edges. We also consider the cost of bucket merging. Formally, the cost of *TPA-U-SA* is

$$\mathcal{C}(\textit{TPA-U-SA}) = \sum_{i=1}^{m} \left(\mathcal{C}_{inter}(G_{n_i,i}) + \mathcal{C}_{intra}(G_{n_i,i})\right) + \mathcal{C}_{merge},$$

where $\mathcal{C}_{inter}(G_{n_i,i})$ (resp. $\mathcal{C}_{intra}(G_{n_i,i})$) denotes the cost of one inter-bucket (resp. intra-bucket) selection on a graph of n_i nodes and i edges, while $\mathcal{C}_{merge}$ denotes the cost incurred by **mergeTBuckets** during the generation process. We use the following two lemmas to bound the cost of inter- and intra-bucket selection, respectively.

Lemma 2. *The expected cost of inter-bucket selection is asymptotically $O(1)$.*

Proof. To simplify the analysis, we consider the case that no buckets have the same size, and for the general case, the proof is analogous. It is reasonable to assume that the cost is non-decreasing w.r.t. graph size, thus we analyze its upper bound by considering node selection from the final graph G_n. Furthermore, let $c_1 = \max\{\frac{w(TB_i)}{w(TB_{i+1})}\}$ for $i \in [1,k)$ and $k = tBucketList.size$, whereas $w(TB_i)$ denotes the weight of the i-th T-Bucket. Note that c_1 is a small constant less than 1. By assuming the roulette wheel selection starts at the *tail* of $tBucketList$, the expected cost for inter-bucket selection is

$$\begin{aligned}\mathrm{E}[\mathcal{C}_{inter}] &= \sum_{i=1}^{k}(k+1-i)\cdot\Pr[TB_i \text{ is selected}]\\ &= \sum_{i=1}^{k}(k+1-i)\cdot\Pr[TB_k \text{ is selected}]\cdot\frac{w(TB_i)}{w(TB_k)}\\ &\le \sum_{i=1}^{k}(k+1-i)\cdot {c_1}^{k-i} = \frac{1-c_1^k}{(1-c_1)^2}-\frac{kc_i^k}{1-c_1}=O(1).\end{aligned}$$

□

Lemma 3. *The expected cost of intra-bucket selection, i.e. the amortized cost of SA-Select on a graph of n nodes is $O(T(\frac{n}{2}))$, where $T(x)$ denotes the cost of SA algorithm for a BA graph (denoted as BA-SA) of x nodes.*

Proof. Let $T(x)$ denote the cost of node selection procedure inside a T-Bucket of size x. Let $c_2 = \max\{\frac{T(2^i)}{T(2^{i+1})}\}, i \in [1,k)$, which can be taken as a constant smaller than 1. We have

$$\begin{aligned}\mathrm{E}[\mathcal{C}_{intra}] &= \sum_{i=1}^{k} T(|TB_i|)\cdot\Pr[TB_i \text{ is selected}]\\ &\le \sum_{i=1}^{k} T(\frac{n}{2^{k+1-i}})\cdot {c_1}^{k-i}\cdot\Pr[TB_k \text{ is selected}]\\ &\le \sum_{i=0}^{k-1} c_1^i\cdot T(\frac{n}{2^{i+1}}) \le \sum_{i=0}^{k-1} c_1^i\cdot c_2^i\cdot T(\frac{n}{2})\\ &= \frac{1-c_1^k c_2^k}{1-c_1c_2}T(\frac{n}{2}) = O(T(\frac{n}{2})).\end{aligned}$$

□

Finally, notice that merging two T-Buckets costs $O(1)$ time with the $D\text{-}Array$ implementation, because we only need to update the start and end positions of $D\text{-}Array$, along with the weight of two corresponding T-Buckets. The number of merging operation is $O(\sum_{l=1}^{\log n} l) = O(\log^2 n)$, since each cascading merge generates a new T-Bucket with cost $O(l)$ where $l = tBucketList.size$ and $l = O(\log n)$. Therefore the cost of intra-bucket selection dominates the others, and our *TPA-U-SA* algorithm is asymptotically more efficient than the *BA-SA* algorithm.

Theorem 1. *Our TPA-U-SA algorithm is asymptotically more efficient than the SA algorithm for BA model in generating graphs of the same size.*

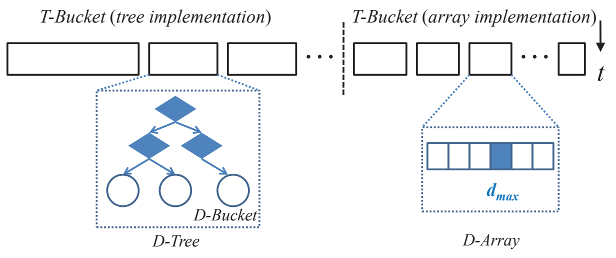

T-Bucket (tree implementation)
T-Bucket (array implementation)
t
D-Bucket
D-Tree
d_{max}
D-Array

Fig. 2. Data structure for the *TPA-U-Hybrid* algorithm.

Algorithm 6: Hybrid-Select

```
Input: $tBucketList$ of current graph $G_k$, $f(\cdot)$ (and $\beta_d$), $g(\cdot)$ (and
       $\beta_t$), $\alpha$, and switching threshold $\tau$
Output: A randomly chosen node $v$ by temporal preferential
        attachment, $v \in [1, k+1]$
1  Uniformly generate a random number $r \in [0, 1)$;
2  Lines 1-3 of Alg. 5;
3  else
4      Randomly select a T-Bucket $TB_i$ according to its weight $w(TB_i)$
         from $tBucketList$ and by roulette wheel;
5      if $TB_i.size \le \tau$ then
6          Lines 6-11 of Alg. 5;
7      else
8          $tNode \leftarrow TB_i.tree.root$;
9          while $tNode$ is not leaf do
10             Uniformly generate a random number $r \in [0, 1)$;
11             if $r < tNode.left/(tNode.left + tNode.right)$ then
12                 $tNode \leftarrow tNode.left$;
13             else
14                 $tNode \leftarrow tNode.right$;
15         Randomly select a node $v$ from vertices in $tNode$ and set
             $chosenNode = v$;
16 return $chosenNode$;
```

4.3 The *TPA-U-Hybrid* algorithm

The efficiency of the *TPA-U-SA* algorithm can still be improved when generating graphs with tens of millions of nodes. According to [21], ROLL-tree based on bucketing and tree search outperforms SA by a significant margin for generating dense and sizeable BA graphs. We integrate similar ideas into node selection inside a T-Bucket, only for large-sized T-Buckets. The basic idea is illustrated in Fig. 2. When the size of a T-Bucket exceeds the switching threshold τ, we change its implementation from $D\text{-}Array$ to a tree structure denoted as $D\text{-}Tree$. Each leaf node of $D\text{-}Tree$ contains a D-Bucket which stores all nodes of some specific degree inside the T-Bucket. The binary search tree aims to speed up inter D-Bucket selection, whereas inside D-Bucket nodes are chosen uniformly at random.

Algorithm 6 demonstrates our **Hybrid-Select** procedure, a substitution for **RW-Select** and **SA-Select**. It differs from **SA-Select** only when choosing a node from some T-Bucket of size $> \tau$ (Lines 8-15 of Algorithm 2). We search the tree according to the weights of tree nodes. For leaf tree nodes containing a D-Bucket for degree d, its weight is defined by the number of graph nodes inside multiplied by $f(d)$; while for internal tree nodes, the weight is the sum of those from its children.

The procedure **mergeTBucket** (shown in Alg. 7) also needs to be modified accordingly to include three cases: (1)two $D\text{-}Array$s merging into one, (2)two $D\text{-}Array$s merging into a new $D\text{-}Tree$, and (3)merging of two $D\text{-}Tree$s. As discussed before, case (1) can be implemented in $O(1)$ time. Both case (2) and (3) incur

Algorithm 7: mergeTBucket

```
Input: $tBucketList$ of current graph $G_k$
Output: merged $tBucketList$, satisfying that the number of
    T-Bucket with size $s$ is at most 2, for any $s$
1  for $i = 1$ to $tBucketList.size - 2$ do
2      if $|TB_i| == |TB_{i+1}| == |TB_{i+2}|$ then
3          if $|TB_i| * 2 \le \tau$ then
4              merge $TB_{i+1}$ into $TB_{i+2}$, by updating the start
                 and end position of $D\text{-}Array$ of $TB_{i+2}$;
5              Remove $TB_{i+1}$ from $tBucketList$;
6          else if $|TB_i| \le \tau$ and $|TB_i| * 2 > \tau$ then
7              Construct a list of D-Buckets from $TB_{i+1}$ and
                 $TB_{i+2}$;
8              Construct $TB_{new}$ by D-Buckets, implemented by
                 tree ;
9              replace $TB_{i+2}$ by $TB_{new}$ and remove $TB_{i+1}$;
10         else
11             Merge all D-Buckets in $TB_{i+1}$ into $TB_{i+2}$;
12             Remove $TB_{i+1}$ from $tBucketList$;
13 return $tBucketList$;
```

data transfer from $D\text{-}Array$ (or $D\text{-}Tree$) to $D\text{-}Tree$; we bound the total data transfer by the following lemma.

Lemma 4. *For* TPA-U-Hybrid*, the total amount of data transfer in procedure* **mergeTBucket** *is* $O(n\log\log n)$.

Proof. Consider the case where all T-Buckets are implemented as trees (the upper bound). For a T-Bucket containing 2^i nodes, let $N(2^i)$ denote the total amount of data transfer upon its construction. Since it is merged from two T-Buckets of size 2^{i-1}, we have the following recursive equation.

$$N(2^i) = O(2^{i-1}) + 2N(2^{i-1}).$$

Note that in the merging process, only one T-Bucket moves all data to the other, and the cost is $O(2^{i-1})$. Since $N(1) = O(1)$, we have $N(2^i) = O(2^{i-1}\log i)$. Combining the fact that $i = O(\log n)$, the amount of data transfer in generating the graph is bounded by

$$\sum_i O(2^{i-1}\log i) < \sum_i O(2^{i-1}\log\log n) = O(n\log\log n) \sim O(n).$$

□

Space complexity. Compared to *TPA-U-SA*, for any T-Bucket implemented as $D\text{-}Tree$, the extra space cost for each tree node is $O(1)$, and the number of tree nodes is precisely twice the number of D-Buckets, i.e., distinct $d(v)$ in this T-Bucket. For graphs of skewed degree distributions, this extra cost is negligible compared to $O(n)$ cost to store the degree info. Consequently, the asymptotical space complexity is also $O(n)$, which is scalable to massive graphs.

Remarks. Note that in this section, we assume that nodes come at a steady rate, i.e., $t(v) = v$, as implicitly implied by most PA-based models. The above two algorithms can be directly applied to the case that node time $t(v)$ is defined by the number of existing edges, i.e., edges come at a steady rate. More precisely, we use the logarithmic bins to store the edges that come sequentially. In practice, we do not have to retain each edges in memory; it is sufficient to store the corresponding new nodes in the same data structures described above.

On the other hand, our fast generation algorithms can be naturally extended to generate PA-based models with time decay and node fitness [22], [23], [28], [29], [38], which can be implemented by the same data structures and the procedures **SA-Select** and **Hybrid-Select**. As far as we know, this is the first work to consider the efficient generation of these complex models.

5 EXPERIMENTS

This section experimentally evaluates our proposed models and generation algorithms. We first empirically evaluate the properties of TPA graphs, with comparison to other PA-based models. We also observe the existence of temporal preferential attachment in real-world networks. Then, we demonstrate the efficiency of our fast generation algorithms. Our code is available at https://github.com/pkumod/TemporalPreferentialAttachment.

5.1 Properties of TPA graphs

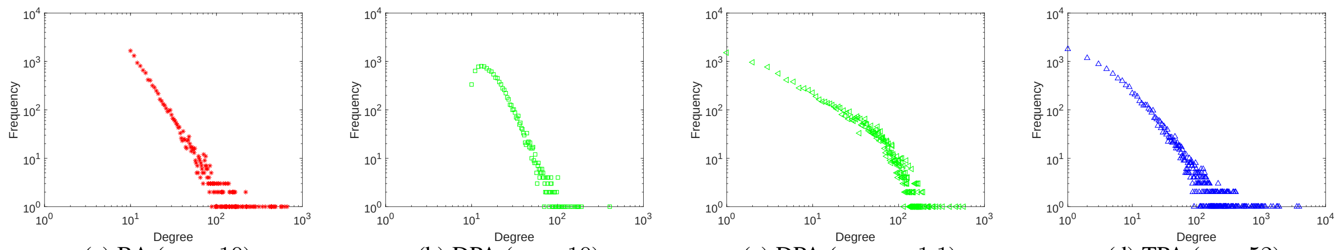
We empirically study the properties of synthetic graphs generated by the TPA model and take BA [13] and a generalized version of DPA [23], [29], [38] for comparison. We evaluate the degree distribution, as well as two other temporal graph properties, i.e., effective diameter over time and the temporal distribution of node degree.

5.1.1 Degree distribution

We generate four synthetic graphs according to the BA, DPA and TPA model with $|V| = 10,000$ and $|E| = 100,000$, and plot their degree distributions in Figure 3. In particular, for the BA model, we set the parameter $m = 10$ (Figure 3(a)). For the DPA model, we first generate the graph according to [23], [29], [38] (Figure 3(b)), with linear degree-based preferential attachment ($\beta_d = 1$), power law TPA function with $\beta_t = 0.8$, and set the default node fitness as Poisson distribution with $\beta_f = 5$ and $h_{max} = 30$. We set $m = 10$ as for the BA model. By default, we assume $t(v) = v$ for $v \in \{1, \dots, n\}$ to compute the time decay. Note that the baseline DPA model has the same problem with BA, i.e., the minimal degree in the graph equals m. We observe that it is easy to integrate the DPA model with random initial degrees [32], [33]. We set the initial degrees follow power law with an extra parameter γ_{init}, and tune the value of γ_{init} so that the number of edges is $100,000$ (Figure 3(c)). For our model, we use the same function form and parameters as the DPA model (except m and γ_{init}), and tune virtual node preference α so that $|E| = 100,000$ (Figure 3(d)).

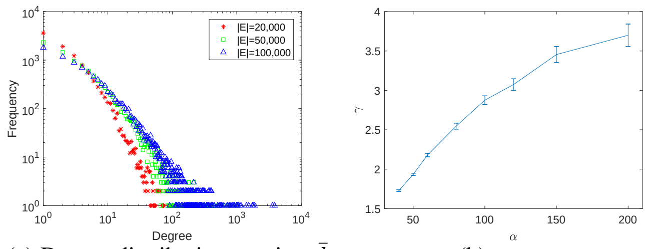
We have the following conclusions. The degree distribution of all compared graph models follow power law (approximately). The BA graph and the baseline DPA graph have a power-law exponent close to 3, which is independent of graph density. This is a little counter-intuitive because dense graph always leads to more heavy-tailed distribution. For the DPA model with random initial degrees, we have to set γ_{init} as 1.1 to generate the graph. Moreover, the degree distribution exhibits secondary power law, which implies that one-sided TPA will result in insufficient large-degree nodes. Here we do not adopt the Poisson distribution [33] for initialization of node degrees, because in practice the head of the distribution significantly deviates from power law while the exponent of the tail stays unchanged. In comparison, our model fits the power law quite well.

Parameter sensitivity. Next, we conduct parameter sensitivity analysis to demonstrate the robustness of our model. We show that the virtual node preference α has a great impact on the density of the graph as well as the power law exponent γ. We fix the parameters as in Figure 3 except α, and generate graph varying



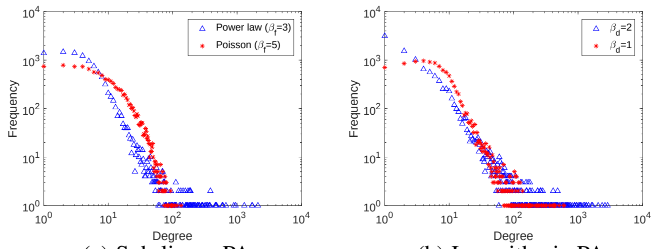
(a) BA ($m = 10$) (b) DPA ($m = 10$) (c) DPA ($\gamma_{init} = 1.1$) (d) TPA ($\alpha = 52$)

Fig. 3. Degree distribution of BA, DPA, and TPA graphs



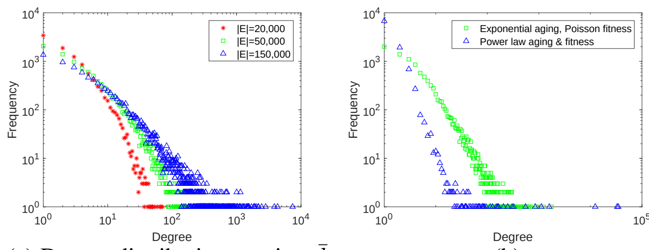
(a) Degree distribution varying $\bar{d}$ (b) γ vs. α

Fig. 4. Effect of virtual node preference α on γ.



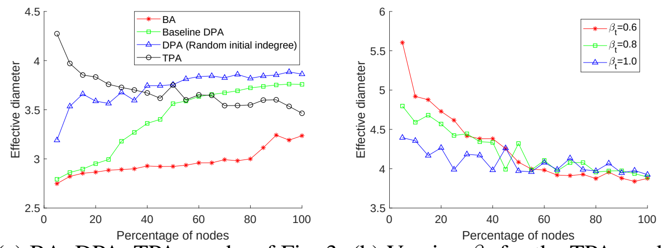
(a) Sub-linear PA (b) Logarithmic PA

Fig. 6. Sub-linear and logarithmic degree-based PA.



(a) Degree distribution varying $\bar{d}$ (b)

Fig. 5. Varying the function form of aging and fitness.



(a) BA, DPA, TPA graphs of Fig. 3 (b) Varying β_t for the TPA model

Fig. 7. Effective diameter of BA, DPA, and TPA graphs.

average degrees (Figure 4(a)). We also plot the fitted γ using maximum likelihood estimation [31] and the standard error by varying α (Figure 4(b)). It turns out that our model is more flexible in generating power-law graphs with large or small density.

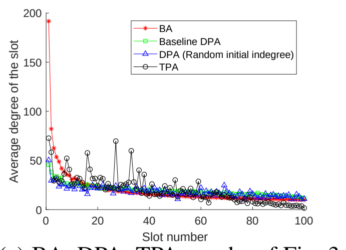
We then vary the function forms of aging and node fitness to generate TPA graphs. In Figure 5(a), we generate graphs with log-normal aging function ($\beta_t = 0.1$). In Figure 5(b), we generate graphs of $|E| = 50,000$ and set the aging function as exponential ($\beta_t = 0.1$) and Poisson fitness ($\beta_f = 1$), or power law functions for aging ($\beta_t = 0.8$) and fitness ($\beta_f = 3$). It show that more skewed fitness (the blue plot) leads to condensation, i.e., the graph contains many super-nodes.

At last, we set the degree-based PA function as sub-linear with $\beta_d = 0.6$, power-law aging ($\beta_t = 0.8$) and vary the fitness function (Fig. 6(a)). In this case, if the fitness is the less skewed Poisson distribution with $\beta_f = 5$, the degree distribution is exponential. The power-law distribution is restored when setting fitness as power law ($\beta_f = 3$). We further assume $f(\cdot)$ as the logarithmic function, i.e., $f(d(v)) = \log(d(v) + 1)^{\beta_d}$, and set $\beta_d = 1$ or $\beta_d = 2$ in Fig. 6(b). With more skewed fitness distribution as in Fig. 6(a), the degree distributions also follow power law in the tail.

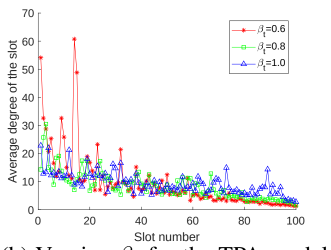
Remarks. Empirically, to generate graphs with heavy-tailed distribution, the values of β_d and β_t have to be close. If β_d is large and β_t is small (e.g., $\beta_t = 0$), unlike the one-sided model, the TPA model always tends to insert internal edges, so the node set grows extremely slow and the graph becomes very dense; if β_t is too large, the graph will not contain nodes of large degree. We leave theoretical analysis and more extensive investigation of these parameters as future work.

5.1.2 *Effective diameter*

We use the graphs generated for Fig. 3 and plot the 90-percentile effective diameter over time in Fig. 7(a), with the x-axis being the percentage of nodes of the whole graph (i.e., $|G_t.V|/|G.V| \times 100\%$). Both the BA graph and the two DPA graphs have increasing diameters. As for the TPA graph, the effective diameter continuously shrinks with time. To explain this, note that in the generation of the BA graph, preferential attachment favors connection between newly inserted nodes and nodes of high degree, which are those inserted at the very beginning. Consequently, the diameter is quite small from the very start and continuously increases with graph size. Although the DPA model explicitly includes the aging phenomenon, it does not consider new links between existing edges. Moreover, the effective diameter of the initial graph (e.g., with first 20% of the nodes) is small for BA and DPA model. On the other hand, the TPA model favors three types of edges according to the two-sided temporal



(a) BA, DPA, TPA graphs of Fig. 3



(b) Varying β_t for the TPA model

Fig. 8. Node degree vs. birth time.

preferential attachment: (a) edge between two old and highly-connected nodes; (b) edge between two new nodes; and (c) edge between one old node and one new node. This is because both an old node of a high degree and a new node with a relatively lower degree can have a high preference. Intuitively, case (a) and (b) results in the shrinking of diameter, whereas edges of case (c) enlarge it. Also, the initial diameter of the TPA graph is much larger than that of other model, because at the beginning many new nodes but fewer edges are inserted, and new nodes tend to connect with each other, resulting in a sparse graph of large diameter.

Parameter sensitivity. Intuitively, parameter β_t mainly controls the shrinking speed of the effective diameter, because as β_t increases, new nodes become more attractive. In Fig. 7(b), we vary the aging parameter β_t and fix all other parameters the same as Fig. 7(a) for TPA graphs. It shows that our model can flexibly adjust the effective diameter over time.

5.1.3 Node degree vs. birth time

We investigate the relation of node degree $d(v)$ vs. node birth time $t(v)$ for the PA-based models, including BA, DPA, and TPA. Again, the tested graphs are those generated for Fig. 3. We set $t(v) = v$ for simplicity. Then, we split the nodes into 100 slots according to their insertion time, where the i-th slot contains nodes numbered from $\frac{|G.V|}{100} \cdot (i-1)+1$ to $\frac{|G.V|}{100} \cdot i$, and plot the average degree of nodes in each slot against the slot number (Fig. 8(a)). For both the BA and two DPA graphs, the average degree per slot continuously decreases. Meanwhile, the average degree of the first few slots are significantly larger, indicating the *old-get-richer* phenomenon. In comparison, for the plot of the TPA graph, there exist several ”spikes”, and this does not necessarily appear at the very beginning. The spikes implies that some super-nodes exist in that slot. Therefore, the newer nodes still have a chance to be popular (i.e., connected by many nodes), which is in accordance with the real-world scenarios.

Parameter sensitivity. We also plot in Fig. 8(b) the effect of β_t. (We only vary this parameter compared to the TPA graph in Fig. 8(a).) Note that as β_t increases, it is more probable that spikes appear in slots with large numbers. To generate larger spikes, we recommend to use more skewed node fitness distributions.

5.2 Observation of TPA in Real-world Graphs

We adopt two real-world datasets to empirically validate the temporal preferential attachment, as shown in Table 3. Both datasets are obtained from [56]. Each dataset represents a *temporal* graph, i.e., a sequence of edges, with each edge containing a *from* node, a *to* node and its timestamp. We preprocess the graphs by sorting the edges in the ascending order of their corresponding timestamps.

Denote by G_t the induced subgraph by all edges of timestamp no larger than t. We select a set of edges E_{test} after time t, such that $|E_{test}| \ll |G_t.E|$ and check the edges that have at least one endpoint in $G_t.V$. According to the TPA mechanism, the probability of two nodes u and v connected by a new edge should be roughly proportional to

$$\frac{f(d(u))f(d(v))g(\Delta t(u))g(\Delta t(v))}{\sum_{x,y\in G_t.V} f(d(x))f(d(y))g(\Delta t(x))g(\Delta t(y))}. \tag{11}$$

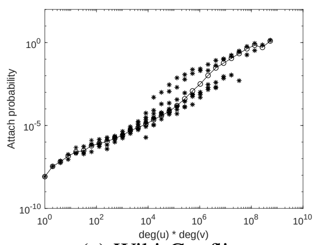
We omit the random disturbance of node fitness here, and employ the observation method of [43]: we count the number of new edges between nodes of degree d_1, age Δt_1 and nodes of degree d_2, age Δt_2. If TPA holds, the count value should be proportional to

$$\sum_{\substack{u,v\in G_t,\\ d(u)=d_1,\Delta t(u)=\Delta t_1,\\ d(v)=d_2,\Delta t(v)=\Delta t_2}} f(d_1)f(d_2)g(\Delta t_1)g(\Delta t_2). \tag{12}$$

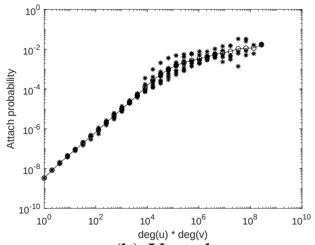
To reduce fluctuations, we plot the histogram (attachment probability vs. degree or time decay) by normalizing all degrees and time decays in $[2^i, 2^{i+1})$ to 2^i. We measure the effect of degree and time decay independently. The results are demonstrated in Fig 9 & 10. We conclude that node preference has positive correlation with node degree, and negative correlation with node age. Even the concrete form of the attachment function is hard to derive, this is not a big problem because of the generality of our model.

TABLE 3
Real-world temporal datasets.

Dataset	**Edge Type**	n	m
Wiki-Conflicts	undirected, multiple	116,836	5,835,570
Youtube	undirected, simple	3,223,585	18,750,748

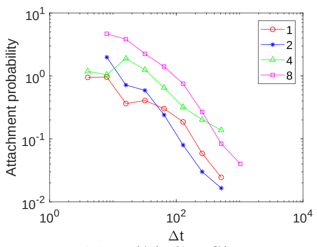


(a) Wiki-Conflicts

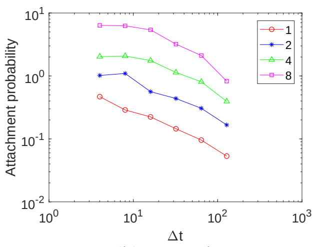


(b) Youtube

Fig. 9. Observation of *two-sided* degree-based PA. Line with circles is the plot by taking $d(u) * d(v)$ as the argument in Eq. 11, whereas the asterisks represent the approximation by viewing $d(u)$ and $d(v)$ as two independent arguments. For example, assume $\beta_d = 1$. Two pairs of nodes with equal value of degree product (e.g., 1 times 8 vs. 2 times 4) should have same probability of receiving an internal link, though in practice their probabilities differ slightly. (G_t contains 75% edges.)



(a) Wiki-Conflicts



(b) Youtube

Fig. 10. Attachment probability (unnormalized) vs. Δt, observed from nodes of degree 1, 2, 4, and 8 respectively.

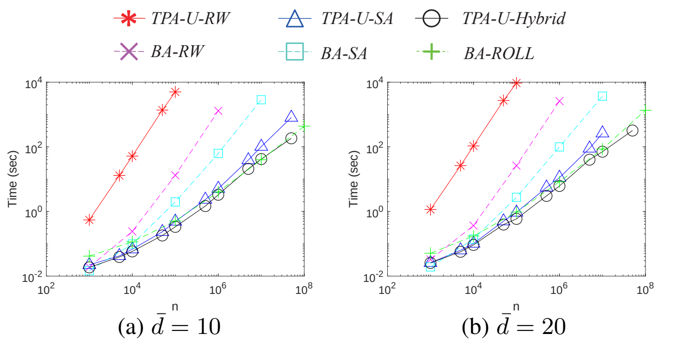


Fig. 11. Scalability of *TPA-U-RW*, *TPA-U-SA*, and *TPA-U-Hybrid*.

Remarks. We only report the results for nodes of degree 1, 2, 4, and 8 in Fig. 10. This is because there are very few nodes of large degree, and fewer are connected by new edges during a small sliding window of edges E_{test}, therefore the plot fluctuates significantly. Note that despite the high preference of each large-degree node, the number of such nodes dominates the value of Eq. 12. We admit this is a shortcoming of the heuristic observation method [43]; however, the recently proposed methods [34], [50] based on maximum likelihood estimation have scalability issues for complex PA functions or large graphs. Generally, they maximize the following likelihood:

$$\Pi_t P(m(t), n(t)|G_{t-1}, \theta(t))P(G_t|G_{t-1}, m(t), n(t), \mathcal{A}), \tag{13}$$

where the first term describes how many new nodes ($n(t)$) and edges ($m(t)$) are added at time t which is governed by parameters $\theta(t)$, while the second term represents the preferential attachment procedure ($\mathcal{A}$ denotes the PA function). For sequence-of-nodes models, $\theta(t) = \emptyset$; while for TPA model, $\theta(t) = \{\alpha\}$. Hence, our model is fundamentally different from the sequence-of-nodes models, and the corresponding measurement technique is still an open problem.

5.3 Evaluation of the Generation Algorithms

To demonstrate the efficiency of our generation algorithm, we compare with the state-of-the-art algorithms for the BA model [21]. Note that the BA model is the simplest among the PA-based models, therefore it has the fastest generation speed. We get the codes from [21], and implement ours in Java and compile with JRE 1.8.0_171. Experiments are conducted on a machine with a Xeon(R) E5-2640@2.60GHz CPU and 96GB memory, and for all experiments, we report the average of 10 runs. Finally, we conduct parameter sensitivity analysis for the TPA model.

We evaluate the efficiency of our three algorithms for the TPA-U model, i.e., *TPA-U-RW*, *TPA-U-SA*, and *TPA-U-Hybrid*. As an example, we set the parameters as: linear degree-based PA function $f(d(v)) = d(v)$ (i.e., $\beta_d = 1.0$), power-law time-decay function $g(\Delta t(v))$ with $\beta_t = 0.8$, and Poisson fitness distribution with $\beta_f = 5.0$ and $h_{max} = 30$. Note that the parameter setting is the same as in Section 5.1 (including Fig. 3, 4, 7 (a) & 8 (a)). We tune the value of virtual node preference α to generate undirected TPA graphs varying node size from 1,000 to 50 million and $\bar{d} \in \{10, 20\}$. Fig. 11 shows the scalability of our three algorithms. For comparison, we also plot the running time of simple roulette wheel (*BA-RW*), stochastic acceptance (*BA-SA*), and ROLL-tree (*BA-ROLL*) for the BA model.

We have the following conclusions. First, *TPA-U-RW* is slow as expected. Moreover, it is more than two orders of magnitude

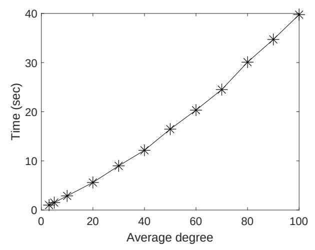


Fig. 12. Generation speed vs. $\bar{d}$.

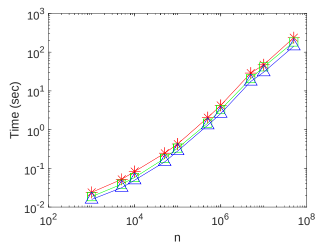


Fig. 13. Generation speed vs. β_t.

slower than *BA-RW* for BA graphs. To explain this, note that the computation of node preference under the TPA model is slower than fetching node degree. Besides, more roulette wheel selection is conducted because of the two-sided temporal preferential attachment mechanism. Last but not least, as we have shown, early inserted nodes have a significantly large degree for BA graphs, thus in practice, the roulette wheel selection should terminate earlier for *BA-RW*. Second, consistent with the analysis in Section 4.2, *TPA-U-SA* is significantly faster than *BA-SA*. For instance, *TPA-U-SA* generates a million-sized graph in less than five seconds, while for *BA-SA*, it takes 26 seconds. This phenomenon is more evident as graph size increases. Finally, the generation speed of *TPA-U-Hybrid* closely matches *BA-ROLL*, which is the state-of-the-art for preferential attachment based models. For the generation of dense graphs, the performance is even slightly better. To generate a TPA graph of 50 million nodes and average degree 20, *TPA-U-Hybrid* only needs about five minutes.

Parameter Sensitivity. We first evaluate our *TPA-U-Hybrid* algorithm by generating graphs with different average degrees. We generate TPA graphs of one million nodes and vary $\bar{d} \in \{10, 20, ..., 100\}$, using the same model parameter setting as Fig. 11. The generation time is shown in Fig. 12. It can be seen that the generation speed w.r.t. average degree is nearly linear, which proves the scalability of *TPA-U-Hybrid*.

Next, we investigate the effect of β_t on generation speed. Intuitively, when β_t decreases, the old nodes tend to have higher node preference. Therefore, the weight distribution of the T-Buckets is more skewed, and we need to visit the buckets of large size more times. We fix other parameters as in Fig. 11, and vary $\beta_t \in \{0.6, 0.8, 1.0\}$ and α (Fig. 13). Decreasing β_t indeed results in the degradation in performance a little bit, but overall the generation efficiency is not affected. Also note that for aging functions that decay faster, such as exponential or log-normal aging, our algorithm can achieve even better performance.

6 Conclusion and Future Work

This paper presents *Temporal Preferential Attachment (TPA)*, the first sequence-of-edges model to reveal the edge-centric growth of some real-world graphs. Our model adopts the preferential attachment mechanism, and non-trivially integrates several ingredients into the sequence-of-edges framework, including two-sided attachment, the time-decay effect and node fitness. Nonetheless, we prove that our model is more robust and flexible than the traditional PA-based models relying on sequence-of-nodes modeling. We further propose two efficient generation algorithms that can be applied to both our model and several PA-based models with aging. Empirical analysis shows that the TPA model preserves several key properties of real-world graphs such as power-law distribution and shrinking diameter, while our generation algorithms are highly scalable. For future work, we plan to investigate TPA graph generation in the distributed environment. Another interesting problem is to mathematically study the relationship between the TPA mechanism and the power law exponent.

Acknowledgement. This work was supported by NSFC under grants 61932001,61961130390 and U20A20174. This work was also partially supported by Beijing Academy of Artificial Intelligence (BAAI). The corresponding author of this work is Lei Zou (zoulei@pku.edu.cn).

REFERENCES

[1] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford InfoLab, 1999.

[2] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*, vol. 8. Cambridge university press, 1994.

[3] A. Singhal, "Introducing the knowledge graph: things, not strings," *Official google blog*, vol. 5, 2012.

[4] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, no. 1, pp. 17–60, 1960.

[5] M. O. Lorenz, "Methods of measuring the concentration of wealth," *JASA*, vol. 9, no. 70, pp. 209–219, 1905.

[6] C. Gkantsidis, M. Mihail, and A. Saberi, "Conductance and congestion in power law graphs," in *SIGMETRICS*, vol. 31, pp. 148–159, ACM, 2003.

[7] P. Brach, M. Cygan, J. Lacki, and P. Sankowski, "Algorithmic complexity of power law networks," in *SODA*, pp. 1306–1325, 2016.

[8] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, "Powergraph: Distributed graph-parallel computation on natural graphs," in *OSDI*, pp. 17–30, 2012.

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *SIGCOMM*, vol. 29, pp. 251–262, ACM, 1999.

[10] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer networks*, vol. 33, no. 1-6, pp. 309–320, 2000.

[11] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-worldnetworks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[12] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: densification laws, shrinking diameters and possible explanations," in *SIGKDD*, pp. 177–187, ACM, 2005.

[13] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[14] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguná, "Hyperbolic geometry of complex networks," *PRE*, vol. 82, no. 3, p. 036106, 2010.

[15] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal, "Stochastic models for the web graph," in *FOCS*, pp. 57–65, IEEE, 2000.

[16] J. Leskovec, "Dynamics of large networks," 2008.

[17] M. Alam, M. Khan, and M. Marathe, "Distributed-memory parallel algorithms for generating massive scale-free networks using preferential attachment model," in *International Conference on High Performance Computing*, 2013.

[18] M. Penschuck, "Generating practical random hyperbolic graphs in near-linear time and with sub-linear memory," in *SEA*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[19] A. Yoo and K. Henderson, "Parallel generation of massive scale-free graphs," *arXiv preprint*, 2010.

[20] H. Park and M.-S. Kim, "Trilliong: A trillion-scale synthetic graph generator using a recursive vector model," in *SIGMOD*, pp. 913–928, ACM, 2017.

[21] A. Hadian, S. Nobari, B. Minaei-Bidgoli, and Q. Qu, "Roll: Fast in-memory generation of gigantic scale-free networks," in *SIGMOD*, pp. 1829–1842, ACM, 2016.

[22] S. N. Dorogovtsev and J. F. F. Mendes, "Evolution of networks with aging of sites," *PRE*, vol. 62, no. 2, p. 1842, 2000.

[23] A. Garavaglia, R. V. D. Hofstad, and G. Woeginger, "The dynamics of power laws: Fitness and aging in preferential attachment trees," *Journal of Statistical Physics*, vol. 168, no. 6, pp. 1137–1179, 2017.

[24] D. Chakrabarti, Y. Zhan, and C. Faloutsos, "R-mat: A recursive model for graph mining," in *ICDM*, pp. 442–446, SIAM, 2004.

[25] J. Leskovec, D. Chakrabarti, J. Kleinberg, and C. Faloutsos, "Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication," in *PKDD*, pp. 133–145, Springer, 2005.

[26] M. von Looz, C. L. Staudt, H. Meyerhenke, and R. Prutkin, *Fast generation of dynamic complex networks with underlying hyperbolic geometry*. KIT, Fakultät für Informatik, 2014.

[27] P. L. Krapivsky, . Redner, S., and . Leyvraz, F., "Connectivity of growing random networks," *PRL*, vol. 85, no. 21, pp. 4629–32, 2000.

[28] M. Medo, G. Cimini, and S. Gualdi, "Temporal effects in the growth of networks," *Physical review letters*, vol. 107, no. 23, p. 238701, 2011.

[29] D. Wang, C. Song, and A.-L. Barabási, "Quantifying long-term scientific impact," *Science*, vol. 342, no. 6154, pp. 127–132, 2013.

[30] A. Lipowski and D. Lipowska, "Roulette-wheel selection via stochastic acceptance," *Physica A*, vol. 391, no. 6, pp. 2193–2196, 2012.

[31] M. E. Newman, "Power laws, pareto distributions and zipf's law," *Contemporary physics*, vol. 46, no. 5, pp. 323–351, 2005.

[32] M. Deijfen, H. V. D. Esker, R. V. D. Hofstad, and G. Hooghiemstra, "A preferential attachment model with random initial degrees," *Arkiv F?r Matematik*, vol. 47, no. 1, pp. 41–72, 2009.

[33] P. Sheridan, Y. Yagahara, and H. Shimodaira, "A preferential attachment model with poisson growth for scale-free networks," *Annals of the institute of statistical mathematics*, vol. 60, no. 4, pp. 747–761, 2008.

[34] M. Inoue, T. Pham, and H. Shimodaira, "Joint estimation of non-parametric transitivity and preferential attachment functions in scientific co-authorship networks," *Journal of Informetrics*, vol. 14, no. 3, p. 101042, 2020.

[35] L. A. Adamic and B. A. Huberman, "Power-law distribution of the world wide web," *Science*, vol. 287, no. 5461, pp. 2115–2115, 2000.

[36] S. Redner, "Citation statistics from 110 years of physical review," *arXiv preprint physics/0506056*, 2005.

[37] M. E. Newman, "The first-mover advantage in scientific publication," *EPL (Europhysics Letters)*, vol. 86, no. 6, p. 68001, 2009.

[38] M. S. Mariani, M. Medo, and Y.-C. Zhang, "Ranking nodes in growing networks: When pagerank fails," *Scientific reports*, vol. 5, p. 16181, 2015.

[39] W. Aiello, F. Chung, and L. Lu, "A random graph model for power law graphs," *Experimental Mathematics*, vol. 10, no. 1, pp. 53–66, 2001.

[40] A. Clauset, C. R. Shalizi, and M. E. Newman, "Power-law distributions in empirical data," *SIAM review*, vol. 51, no. 4, pp. 661–703, 2009.

[41] D. J. D. S. Price, "Networks of scientific papers," *Science*, pp. 510–515, 1965.

[42] M. E. Newman, "Communities, modules and large-scale structure in networks," *Nature physics*, vol. 8, no. 1, pp. 25–31, 2012.

[43] H. Jeong, Z. Néda, and A.-L. Barabási, "Measuring preferential attachment in evolving networks," *EPL*, vol. 61, no. 4, p. 567, 2003.

[44] J. Kunegis, M. Blattner, and C. Moser, "Preferential attachment in online networks: Measurement and explanations," in *WebSci*, pp. 205–214, ACM, 2013.

[45] J. Kunegis, "Konect: the koblenz network collection," in *WWW*, pp. 1343–1350, ACM, 2013.

[46] M. L. Goldstein, S. A. Morris, and G. G. Yen, "Problems with fitting to the power-law distribution," *The European Physical Journal B-Condensed Matter and Complex Systems*, vol. 41, no. 2, pp. 255–258, 2004.

[47] A. D. Broido and A. Clauset, "Scale-free networks are rare," *Nature communications*, vol. 10, no. 1, pp. 1–10, 2019.

[48] C. P. Massen and J. P. Doye, "Preferential attachment during the evolution of a potential energy landscape," *The Journal of chemical physics*, vol. 127, no. 11, p. 114306, 2007.

[49] P. Sheridan, Y. Yagahara, and H. Shimodaira, "Measuring preferential attachment in growing networks with missing-timelines using markov chain monte carlo," *Physica A: Statistical Mechanics and its Applications*, vol. 391, no. 20, pp. 5031–5040, 2012.

[50] T. Pham, P. Sheridan, and H. Shimodaira, "Pafit: A statistical method for measuring preferential attachment in temporal complex networks," *PloS one*, vol. 10, no. 9, p. e0137796, 2015.

[51] Y. C. Lo, H. C. Lai, C. T. Li, and S. D. Lin, "Mining and generating large-scaled social networks via mapreduce," *Social Network Analysis & Mining*, vol. 3, no. 4, pp. 1449–1469, 2013.

[52] R. Aldecoa, C. Orsini, and D. Krioukov, "Hyperbolic graph generator," *CPC*, vol. 196, pp. 492–496, 2015.

[53] J. Zhang and Y. Tay, "Gscaler: Synthetically scaling a given graph.," in *EDBT*, vol. 16, pp. 53–64, 2016.

[54] H. Park and M.-S. Kim, "Evograph: an effective and efficient graph upscaling method for preserving graph properties," in *SIGKDD*, pp. 2051–2059, ACM, 2018.

[55] G. Bianconi and A.-L. Barabási, "Bose-einstein condensation in complex networks," *Physical review letters*, vol. 86, no. 24, p. 5632, 2001.

[56] http://konect.cc/.

Yu Liu received the BS degree in computer science from Shandong University, in 2011, and received the M.Eng and PhD degrees from School of Information, Renmin University, in 2014 and 2018 respectively. He is now a postdoctoral fellow at Peking University. His research interests include complex graph models and approximate graph algorithms.

Lei Zou received the BS and PhD degrees in computer science from the Huazhong University of Science and Technology (HUST), in 2003 and 2009, respectively. Now, he is a professor with Wangxuan Institute of Computer Technology, Peking University. His research interests include graph database and semantic data management.

Zhewei Wei received the PhD degree in computer science and engineering from the Hong Kong University of Science and Technology. He is now a professor with the Renmin University of China. His research interests include algorithms for massive data, streaming algorithms, and graph algorithms.