Construction of Directed 2K Graphs

Bálint Tillman
University of California, Irvine
tillmanb@uci.edu

Athina Markopoulou
University of California, Irvine
athina@uci.edu

Carter T. Butts
University of California, Irvine
buttsc@uci.edu

Minas Gjoka*
University of California, Irvine
minas.gjoka@gmail.com

ABSTRACT

We study the problem of generating synthetic graphs that resemble real-world directed graphs in terms of their degree correlations. In order to capture degree correlation specifically for directed graphs, we define *directed 2K (D2K)* as those graphs with a given directed degree sequence (DDS) and a given target joint degree and attribute matrix (JDAM). We provide necessary and sufficient conditions for a target D2K to be realizable and we design an efficient algorithm that generates graph realizations with exactly the target D2K. We apply our algorithm to generate synthetic graphs that target real-world directed graphs (such as Twitter), and we demonstrate its benefits compared to state-of-the-art construction algorithms.

CCS CONCEPTS

•Mathematics of computing →Graph algorithms;

KEYWORDS

Directed graphs; graph realizations; construction algorithms

1 INTRODUCTION

It is often desirable to generate (or "construct") synthetic graphs that resemble real-world networks w.r.t. certain properties of interest. For example, researchers often want to simulate a process on a realistic network topology and they may not have access to a real-world network; or they may want to generate several different realizations of graphs of interest. In this paper, we focus specifically on directed graphs that appear in many application scenarios including, but not limited to, online social networks such as Twitter (*e.g.,* referring to the following, re-tweeting or actual communication among users).

There is a large body of work, in classic literature [11],[23],[22],[31], as well as more recently (dK-series [27],[28], PAM [12]), on generating realizations of undirected graphs that exhibit exactly some target structural properties such as a given degree distribution or a given joint degree matrix. In this paper, we adopt the dK-series framework [27],[28], which provides an elegant way to trade off accuracy (in terms of graph properties) for complexity (in generating graph realizations). Construction of dK-graphs is well understood (*i.e.,* efficient algorithms and realizability conditions are known) for 1K (graphs with a given degree distribution) and 2K (graphs with a given joint degree matrix). For $d > 2$ (which is necessary for capturing the clustering exhibited in social networks), we recently proved that the problem is NP-hard [8] and we also developed efficient heuristics [20]. In contrast, construction is not well-understood for directed graphs: results are known for construction of graphs with a target directed degree sequence [18], [17], but the notion of directed degree correlation, or directed dK-series for $d \geq 2$, has not been previously addressed.

In this paper, we address this problem. We define two notions of degree correlation in directed 2K graphs: primarily *directed 2K (D2K),* and secondarily its special case *D2Km.* D2K includes the notion of directed degree sequence (DDS) and builds on an old trick (mapping directed graphs to bipartite undirected graphs) to also express degree-correlation via a joint degree-attribute matrix (JDAM) for the bipartite graph. This problem definition lends itself naturally to techniques we previously developed for *undirected 2K* [20], which we exploit to develop (i) necessary and sufficient realizability conditions and (ii) an efficient algorithm that constructs graph realizations with the exact target D2K. Our D2K approach advances the state-of-the-art in modeling and simulation of realistic directed graphs, especially in the context of online social networks.

The outline of the rest of the paper is as follows. Section 2 summarizes related work. Section 3 defines the Directed 2K problem (D2K and its special case D2Km). Section 4 provides realizability conditions for D2K and an efficient algorithm for constructing such realizations. Section 5 applies the algorithm to construct directed 2K graphs that resemble real world graphs, and demonstrates the advantages of our approach compared to state-of-the-art approaches. Section 6 concludes the paper.

2 RELATED WORK

We adopt the systematic framework of dK-series [27], which was introduced to characterize the properties of an *undirected* graph using a series of probability distributions specifying all degree correlations within d-sized, simple, and connected subgraphs of a given graph G. In this framework, higher values of d capture progressively more properties of G at the cost of more complex representation of the probability distribution. The dK-series exhibit two desired properties: inclusion (a dK distribution includes all properties defined by any d'K distribution, $\forall d' < d$ and convergence (nK, where $n = |V|$ specifies the entire graph, within isomorphism).

*M. Gjoka was with UCI when this work was conducted. He is currently with Google.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

KDD'17, August 13–17, 2017, Halifax, NS, Canada.
© 2017 ACM. 978-1-4503-4887-4/17/08…$15.00
DOI: http://dx.doi.org/10.1145/3097983.3098119

We are interested in graph construction approaches that produce *simple* graphs that exhibit an *exact* target distribution; this is different from the stochastic approach presented by [10] or the configuration model in [1].

0K Construction. 0K describes graphs with prescribed number of nodes and edges. This notion translates to simple Erdős-Rényi (ER) graphs with fixed number of edges. There is a simple extension of ER graphs to generate not just undirected but directed graphs as well, which we will use in our evaluation.

1K Construction. Degree sequences are equivalent to 1K as defined in the dK-series. Because degree sequences have been studied since the 1950s, we only focus on the most relevant results. The realizability conditions for degree sequences were given by the Erdős-Gallai theorem [11], and first algorithm to produce a single realization by Havel-Hakimi [23],[22]. More recently, importance sampling algorithms were proposed in [5] and [7].

2K Construction. A Joint Degree Matrix (JDM) is given by the number of edges between nodes of degree i (V_i) and j (V_j) as in [2]:

$$JDM(i,j) = \sum_{u \in V_i} \sum_{v \in V_j} 1_{\{(u,v) \in E\}} \tag{1}$$

Realizability conditions for undirected 2K were provided in [2]. Algorithms for generating realizations of a target JDM were provided in [6], [20] and [30]. The algorithms presented in [2] and [30] are designed to only produce restricted realizations with an additional property called Balanced Degree Invariant (BDI). In [6] and [20], the algorithms have non-zero probability to produce any realization of a 2K distribution. An importance sampling algorithm was introduced in [4].

In prior work [20], we defined the notion of Joint Degree and Attribute Matrix (JDAM), which extends JDM for graphs with a single node attribute. It captures the number of edges between nodes of degree i, attribute value p ($V_{\{i,p\}}$) and degree j, attribute value q ($V_{\{i,q\}}$). Known results can be easily applied from the JDM problem including sampling.

$$JDAM(\{i,p\},\{j,q\}) = \sum_{u \in V_{\{i,p\}}} \sum_{v \in V_{\{j,q\}}} 1_{\{(u,v) \in E\}} \tag{2}$$

The space of simple graph realizations of 1K distributions is connected over double edge swaps preserving degrees [31] and a similar result was shown in [6] or [2] for 2K with double edge swaps that preserve degrees and the joint degree matrix. These swaps allow the use of MCMC to generate approximate probability samples for 1K and 2K. However, fast mixing for the MCMC has not been proved in general, only for special classes of realizations [16] [15].

dK, $d > 2$ Construction. While algorithms of known time complexity exist for $d \leq 2$, Monte Carlo Markov Chain (MCMC) approaches are typically used for $d > 2$. Several attempts were made to find polynomial time algorithms to produce 3K graphs [27] or 2K realizations with prescribed (degree-dependent) clustering coefficient [19],[20] and [28], but we recently proved that even the realizability check for these inputs is NP-Complete in [8].

Annotated graph construction was proposed in [9] that considered degree correlations, however the proposed construction method will generate graphs with self-loops or multi-edges initially. An additional step removes these extra edges to make the graph simple and finally the largest connected component of the graph is returned as the constructed realization.

Beyond the undirected dK-series framework [27], we discuss related work for bipartite and directed degree sequences in the following section, as part of our description of directed dK-series.

3 DIRECTED 2K PROBLEM DEFINITION

In this paper, we follow the previous taxonomy, but we define the dK-series for directed graphs with given properties.

Directed 0K. As mentioned in Section 2, it is trivial to extend ER model for directed graphs. In addition, we consider the UMAN model [24], which captures the number of mutual, asymmetric, and null dyads in a graph. One can think about UMAN as 0K distribution with fixed numbers of mutual and unreciprocated edges.

Directed 1K. In an undirected graph G, a node v has degree $d(v)$, and the degree sequence is simply:

$$DS = \{d_1, d_2, ...d_{|V|}\} \tag{3}$$

In a directed graph, a node v has both in and out degree and the *directed degree sequence* can be expressed in the following way. An example is shown on Fig. 1-top left.

$$DDS = \{(d_v^{in}, d_v^{out}), v \in V\} \tag{4}$$

It is well known from Gale's work [18], that any directed graph can be mapped 1-1 to an undirected bipartite graph, where each node v of the directed graph is split in two nodes v_{in} and v_{out}, and the undirected edges across the two (in and out) partitions of the bipartite graph correspond to the directed edges in the directed graph. A self loop (v,v) in the directed graph corresponds to a "non-chord" (v_{in}, v_{out}) in the bipartite graph, and is shown in dashed line on Fig. 1-left-bottom.

Construction algorithms are known for a bipartite degree sequence with [18], or without non-chords, and therefore for the corresponding directed graphs with or without [17] self-loops, respectively. More recently, an importance sampling algorithm was provided for D1K in [25].

Directed 2K. Our goal in this paper is to go beyond just directed degree sequence and capture degree correlation, and there are two ways to go about it.

D2Km:[1] **Joint (Directed) Degree Matrix** $JDM((d_i^{in}, d_i^{out}), (d_j^{in}, d_j^{out}))$.

One way is to work directly with the directed graph, see Fig. 1-top right. We partition nodes by both their in and out degrees (d_v^{in}, d_v^{out}), and we can define the joint degree matrix to capture the number of edges $JDM((d_i^{in}, d_i^{out}), (d_j^{in}, d_j^{out}))$, between nodes with (d_i^{in}, d_i^{out}) and (d_j^{in}, d_j^{out}). This is shown on the top of Fig. 1. This is a natural extension of the JDM in the undirected case and captures a restrictive notion of degree correlation. However, there is no previously known algorithm that can provably generate graph realizations with this target JDM. For example, a naive extension of our own 2K construction algorithm [20], from undirected to directed graphs, does not work all the time, although it is still a good heuristic for sparse graphs.

[1]Although presented first, D2Km is actually a special (more restrictive) case of D2K (directed degree correlation) and stands for "D2K modified".

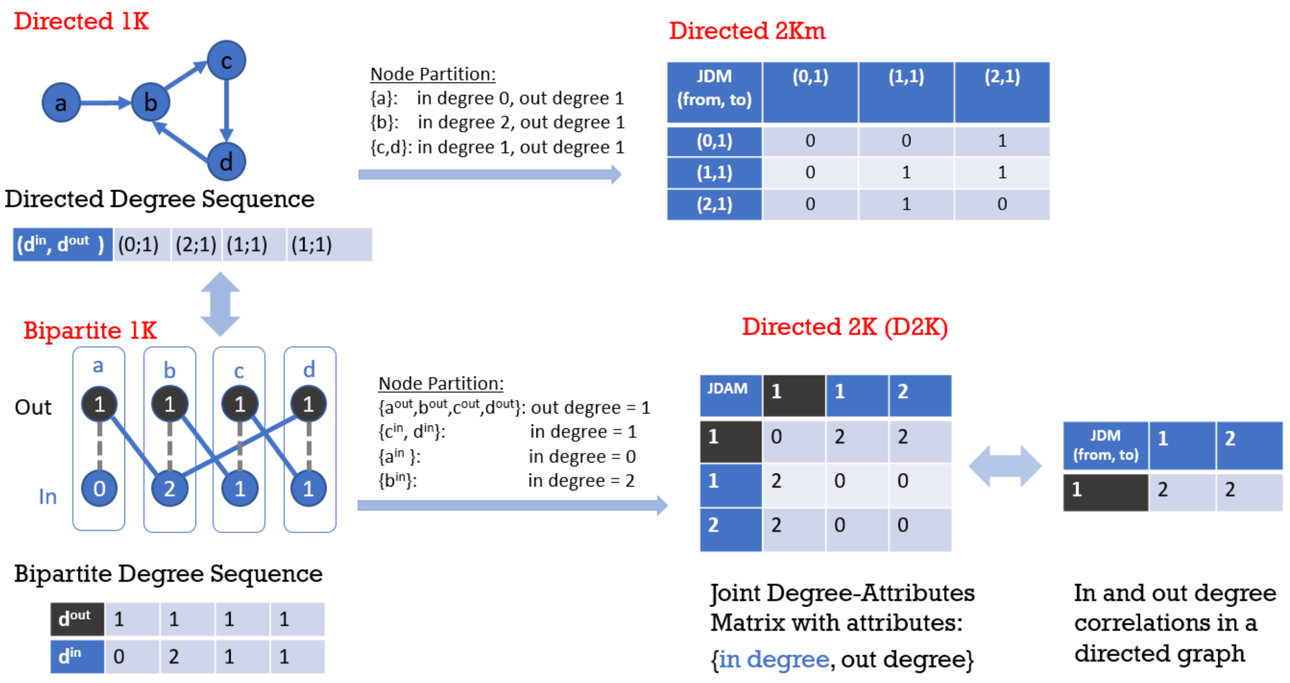
Directed 1K
a
b
c
d
Directed Degree Sequence
(d^in, d^out) (0;1) (2;1) (1;1) (1;1)
Node Partition:
{a}: in degree 0, out degree 1
{b}: in degree 2, out degree 1
{c,d}: in degree 1, out degree 1
Directed 2Km
JDM (from, to) (0,1) (1,1) (2,1)
(0,1) 0 0 1
(1,1) 0 1 1
(2,1) 0 1 0
Bipartite 1K
Out
In
Bipartite Degree Sequence
d^out 1 1 1 1
d^in 0 2 1 1
Node Partition:
{a^out,b^out,c^out,d^out}: out degree = 1
{c^in, d^in}: in degree = 1
{a^in }: in degree = 0
{b^in}: in degree = 2
Directed 2K (D2K)
JDAM 1 1 2
1 0 2 2
1 2 0 0
2 2 0 0
Joint Degree-Attributes Matrix with attributes:
{in degree, out degree}
JDM (from, to) 1 2
1 2 2
In and out degree correlations in a directed graph

Figure 1: Defining Directed 2K, to capture degree correlations in a directed graph. Top left, Directed 1K: Directed graph with a given degree sequence (DDS). Bottom left, Bipartite 1K: Mapping of the previous to a Bipartite undirected graph with a given bipartite degree sequence; non-chords in the bipartite graph (shown in dashed line) correspond to self-loops in the directed graph. Bottom right, Directed 2K (D2K): Joint-Degree-Attribute Matrix (JDAM), where nodes of the bipartite graph are partitioned by their degree-and-(in or out) attribute or equivalently the in and out degree correlations of the directed graph. Top right: Directed 2Km: Joint Degree Matrix (JDM) for directed graphs, where nodes are partitioned according to their (in degree, out degree).

D2K: Joint (Directed) Degree-Attribute Matrix *JDAM*(*degree*, *in or out*).

Another natural approach would be to simply consider the degree correlations between in and out degrees in a directed graph, as shown in Fig.1-bottom rightmost matrix. Alternatively it is possible to work with the equivalent representation of a directed graph as an undirected bipartite graph without non-chords (Fig. 1-bottom left), and define degree correlations there. We partition in and out nodes by their degree, essentially considering that nodes in the bipartite graph can have an attribute that takes two values, "in" or "out." We can now define degree correlation using the Joint Degree-Attribute Matrix (JDAM, which we first defined in [20]), as shown on Fig. 1 - bottom right. This leads to a JDAM with two attribute values, such that $\forall i, j = 1, \ldots, d_{max}$ degrees and $p \in \{in, out\}$ attribute values $JDAM(\{i, p\}, \{j, p\}) = 0$, *i.e.,* because the bipartite graph has no edges between two "in" or "'out" nodes. Furthermore, the number of non-chords will be noted as $f(\{i, p\}, \{j, q\})$, where $i, j \in \{1, \ldots, d_{max}\}$ and $p \neq q \in \{in, out\}$; f can be computed by passing through the directed degree sequence once and counting the number entries with in-degree i and out-degree j.

This notion of *bipartite JDAM* has all the properties known for *JDM* and *JDAM*, since it is a special case of *JDAM* that we first defined in [20]. This includes sufficient and necessary conditions for realizability, construction algorithms, existence of Balanced Degree Invariant realizations, importance sampling algorithm extensions from *JDM*, connectivity of space of realizations over *JDAM* preserving double-edge swaps and MCMC properties. However, we have to show for D2K, that the non-chords described by the directed degree sequence can be added as well.

Relation of the two problems. An overview of the problems of interest is provided on Fig. 1. One difference between the two 2K problems is that D2Km provides a more restrictive notion of degree correlation than D2K since it partitions nodes by two numbers (d^{in}, d^{out})) vs. one number d_v^{in} or d_v^{out}. D2Km can essentially be obtained as a special case of D2K by further partitioning nodes with the same d^{in} by their out degree as well. Therefore, D2Km can be solved by the same algorithm that solves D2K. For the rest of the paper, we will consider the D2K problem.

Directed 2K (D2K) Problems: Given targets $JDAM^{\odot}$ *and* $DDS^{\odot}$:

- **Realizability:** Decide whether this D2K is realizable, *i.e.,* whether there exist graphs with the target properties.
- **Construction:** Design an algorithm that constructs (*i.e.,* generates) at least one such graph realization.
- **Sampling:** Sample, *e.g.,* uniformly, from the space of all graph realizations with the target D2K.

4 REALIZABILITY AND ALGORITHM

In this section, we take as input the two target properties, namely the target $\text{JDAM}^{\odot}(\{i,p\},\{j,q\})$ with two attribute values and the $\text{DDS}^{\odot}$, and construct a directed 2K-graph with N nodes that exhibits exactly these target properties. In this section, we use the bipartite representation of directed graphs as in Figure 1. This enables us to simplify our algorithm description and show the relation to our previous construction algorithm for undirected graphs [20]. More specifically, we remove directionality of the edges and only handle a partition with non-chords. Since the bipartite representation is equivalent to a directed graph, the following algorithm could be rephrased to construct directed graphs without the use of the bipartite representation.

Recall that in the D2K definition, nodes are partitioned into K parts $V_k, k = 1...K$, according to the distinct $d^{in} = i$ or $d^{out} = j$ they exhibit and $\text{JDAM}(\{i,p\},\{j,q\})$ is indexed accordingly. For example, on Fig. 1 bottom-right, each node belongs to one of four parts $V_{\{0,in\}} = \{v \in V : d^{in} = 0\}$, $V_{\{1,out\}} = \{v \in V : d^{out} = 1\}$, $V_{\{1,in\}} = \{v \in V : d^{in} = 1\}$, $V_{\{2,in\}} = \{v \in V : d^{in} = 2\}$, and the JDAM is 3x3 (by removing rows and columns corresponding to any $V_{\{0,p\}}$, since there are no edges using these parts of any partition).

4.1 Realizability

Not all target properties are realizable (or "graphical"): there does not always exist at least one simple directed graph with those exact properties. Necessary and sufficient conditions for a target D2K, *i.e.,* $\text{JDAM}^{\odot}(\{i,p\},\{j,q\})$ and $\text{DDS}^{\odot}$, to be realizable are the following.

I $\forall i,j,p : JDAM(\{i,p\},\{j,p\}) = 0$

II $\forall i,j,p,q,$ if $JDAM(\{i,p\},\{j,q\}) > 0,$ $JDAM(\{i,p\},\{j,q\}) + f(\{i,p\},\{j,q\}) \leq |V_{\{i,p\}}| \cdot |V_{\{j,q\}}|$

III $\forall i,p : |V_{\{i,p\}}| = \sum_{\{j,q\}} \frac{JDAM(\{i,p\},\{j,q\})}{i} = |\{(d_v^{in}, d_v^{out}),$

$v \in DDS | (d_v^{in} = i \wedge p = \text{"}in\text{''}) \vee (d_v^{out} = i \wedge p = \text{"}out\text{''})\}|$

These are generalizations of the conditions for an undirected JDM, JDAM to be realizable, and they are clearly necessary. The first condition states that every realization of the target JDAM is bipartite, i.e., there should be no edges between two nodes both in "in" or "out" parts. The second condition considers edges between two ("in" and "out") parts and states that the number of edges defined by the $JDAM(\{i,p\},\{j,p\})$ plus the number of non-chords ($f(\{i,p\},\{j,q\})$) should not exceed the total number of edges possible in a complete bipartite graph across the two parts. This can be seen in the example of Fig. 1: given higher $JDAM(2, \text{"}in\text{''}, 1, \text{"}out\text{''}) = 4$ would result using a non-chord or even higher values would create multi-edges (with same number of nodes). The last condition ensures that the target JDAM and the target DDS are consistent: the number of nodes with in (or out) degree i should be the same whether computed using the JDAM or the DDS. In the example of Fig. 1: $|V_{\{1,\text{"}in\text{''}\}}| = 2$ since there are two nodes in DDS with in degree 1 and $\sum_{\{j,q\}} \frac{JDAM(\{1,\text{"}in\text{''}\},\{j,q\})}{1} = 2$. The DDS also trivially ensures that the number of nodes is going to be an integer.

Necessity of these conditions for simple graph construction are trivial. Sufficiency is established via the constructive proof of the algorithm presented next.

4.2 Algorithm

Algorithm 1 can generate a simple directed graph for given $DDS^{\odot}$, $JDAM^{\odot}$ if the inputs are realizable.[2]

```
Algorithm 1: D2K
Input: $DDS^{\odot}$, $JDAM^{\odot}$
Initialization:
a: Create G with nodes, partition, stubs using $DDS^{\odot}$
b: Add non-chords to G using $DDS^{\odot}$
Add Edges:
1: for every pair ($\{i,p\},\{j,q\}$) of partition:
2:     while $JDAM(\{i,p\},\{j,q\}) < JDAM^{\odot}(\{i,p\},\{j,q\})$
3:         Pick any nodes $u$ (from $V\{i,p\}$), $v$ (from $V\{j,q\}$)
             s.t. $(u,v)$ is not a non-chord or existing edge
4:         if $u$ does not have free stubs:
5:             $u'$: node in $V\{i,p\}$ with free stubs
6:             neighbor switch for $u$ using $u'$
                   if neighbor switch fails, $u := u'$
7:         if $v$ does not have free stubs:
8:             $v'$: node in $V\{j,q\}$ with free stubs
9:             neighbor switch for $v$ using $v'$
                   if neighbor switch fails, $v := v'$
10:        add edge between $(u,v)$
11:        $JDAM(\{i,p\},\{j,q\})$ ++; $JDAM(\{j,q\},\{i,p\})$ ++;
Transform bipartite G to directed graph
Output: simple directed graph
```

First, we create a set of nodes V, where $|V| = 2 \cdot |\text{DDS}|$, we assign stubs to each node and partition nodes, as specified in the *target directed degree sequence* $\text{DDS}^{\odot}$. The stubs are originally free, *i.e.,* they are only connected to one node. We also initialize all entries of JDAM to 0 and the non-chords between nodes according to $\text{DDS}^{\odot}$. Then the algorithm proceeds by connecting two nodes (one from "in" and one from "out" side), thus adding one edge (u,v) at a time, that (i) are not previously connected to each other (ii) do not have a non-chord between them (to avoid self loops) and (iii) for whom the corresponding entry in the JDAM has not reached its target. The challenge lies in showing that the algorithm will always be able to make progress, by adding one edge at a time, until all entries of the JDAM reach their target, when the algorithm terminates. Indeed, there may be cases (2-5 in Fig.3), where u, v or both do not have free stubs. Even in those cases, however, we will be able to perform JDAM-preserving edge rewirings (called neighbor switch [20]: remove a neighbor t of v such that t is not a neighbor of v' and add edge (t,v')) to free stubs and then add the edge (u,v) (cases 2-3); or we will be able to add another edge (u',v), (u,v') (case 4); or (u',v') (case 5a) in Fig.3. Next, we prove that this is, indeed, always the case.

[2]Here, we follow the style of proof and algorithm for construction of undirected graphs [20]. In addition, other results for JDM construction could be extended for directed 2K as well, such as the proof in [6].

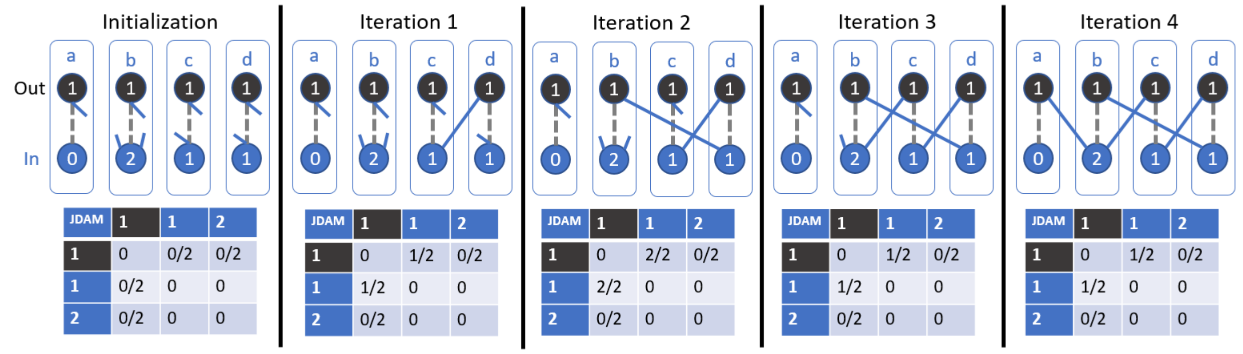

Figure 2: Example of running Algorithm 1 to construct a graph realization with the target D2K specified in Fig. 1. The algorithm starts by initializing a bipartite graph with non-chords and stubs (initialization). In each iteration, one new edge is added by connecting two stubs. Depending on the case, a local edge rewiring may need to be performed first, or a new pair of nodes may be chosen, before the edge can be added. In this example, the algorithm terminates after four iterations by generating a realization that is different from Fig. 1 (the directed cycle is reoriented).

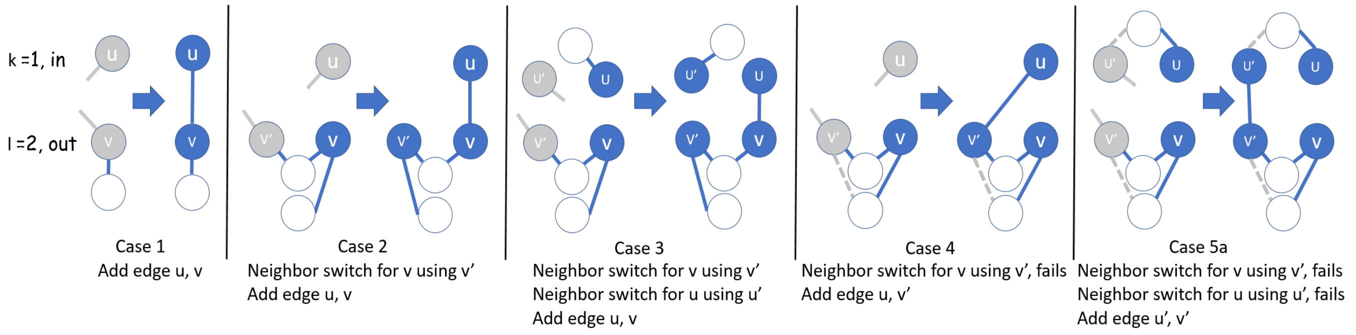
k =1, in
l =2, out
u
v
u'
v'
Case 1
Add edge u, v
Case 2
Neighbor switch for v using v'
Add edge u, v
Case 3
Neighbor switch for v using v'
Neighbor switch for u using u'
Add edge u, v
Case 4
Neighbor switch for v using v', fails
Add edge u, v'
Case 5a
Neighbor switch for v using v', fails
Neighbor switch for u using u', fails
Add edge u', v'

Figure 3: Some of the possible cases, while attempting to add edge (u, v) in Algorithm 1.

Example. In Fig. 2, we show an example of running Algorithm 1 to construct graphs with the target D2K specified in Fig.1. Depending on how the algorithm chooses (u, v) pairs, cases 2-5a can appear. For example, in iteration 3, if we try to add edge (c^{out}, b^{in}), we can simply add the edge. However, (d^{out}, b^{in}) would be another possible choice and c^{out} a candidate for neighbor switch, but the neighbor switch is not possible due to non-chord and this results in a Case 4; the algorithm proceeds by adding (c^{out}, b^{in}) edge, without additional rewiring.

Proof. While the construction has not reached the target, there is a pair $\{i, p\}$,$\{j, q\}$ s.t. $JDAM(\{i, p\}, \{j, q\}) < JDAM^{\odot}(\{i, p\}, \{j, q\})$. Condition II guarantees that two nodes, (u, v), can always be chosen to add an edge, *i.e.,* there is no edge or non-chord between them (Alg. 1, this is line 3). Condition III ensures that at least one node exists with a free stub in $V_{\{i,p\}}$ and $V_{\{j,q\}}$. For more details please see Lemma 2 and 3 in [20].

Next, we show that every iteration can proceed by adding a new edge to the graph (*i.e.,* we will not get stuck). The cases are depicted on Fig. 3.

Case 1. Add a new edge between two nodes with free stubs, no local rewiring needed.

Case 2. Add a new edge between a node v without free stubs and a node u with free stubs where neighbor switch is possible for v without using any non-chords.

Case 3. Add a new edge between two nodes without free stubs where neighbor switches are possible for both nodes without using any non-chords.

Case 4. Add a new edge between a node v without free stubs and a node u with free stubs (or without free stubs where neighbor switch is possible) where neighbor switch is not possible for v using v' without using any non-chords. In this case v' has the same neighbors as v except the one for which it has an assigned non-chord. In this case v' is not connected to u and it is possible to add $\{v', u\}$ edge ($\{v', u\}$ is clearly not an edge since then u would be also connected to v or v could have done a neighbor switch).

Case 5. Add a new edge between two nodes (u, v) w/out free stubs, where neither can do a neighbor switch with u' and v' respectively. We have to break this case into two subcases, based on whether two nodes u', v' w/out free stubs form a non-chord or not.

Case 5a. u', v' is not a non-chord. This means that we can add a new edge between u', v'. It is easy to see that u', v' edge is not already present, because otherwise u and v could have performed a neighbor switch.

Case 5b. u', v' is a non-chord. This case is not possible when u, v are not able to perform neighbor switches at the same time. Without loss of generality, let's say that u connects to all the neighbors of u' and node v'. This means that no neighbor switch is available for u. If we want to construct v such that it can't perform a neighbor switch with v', we need v to connect all of the neighbors of v'; however, this would include u as well, and clearly that edge doesn't exist. Contradiction.

This concludes our proof and shows that the algorithm will terminate and will generate a bipartite graph after adding $\sum \frac{JDAM^{\odot}(i,j)}{2} = |E|$ edges. Finally, the bipartite graph can be mapped to the corresponding directed graph, as depicted on Fig.1. □

Running Time. The time complexity of the above algorithm is $O(|E| \cdot d_{max})$. In each iteration of the `while` loop, one edge is always added, until we add all $|E|$ edges. However, we have to consider how much time it takes to pick u, v nodes. There could be neighbor switches that remove previously added edges or add edges between the two parts. If we naively looked up node pairs, it would become an issue for dense graphs. A simple solution is to keep track of $JDAM^{\odot}(\{i,p\},\{j,q\}) - JDAM(\{i,p\},\{j,q\})$ many node pairs where edges can be added in a set P. For every pair of $\{i,p\},\{j,q\}$, it is possible to initialize P by passing through $O(JDAM^{\odot}(\{i,p\},\{j,q\}) + f(\{i,p\},\{j,q\}))$ node pairs. A new (u,v) node pair can simply be picked as a random element from P. If a neighbor switch for $u \in V_{\{i,p\}}$ (and similarly to v), rewires a neighbor $t \in V_{\{j,q\}}$, then $P = P \setminus \{u',t\} \cup \{u,t\}$ maintains available node pairs in P. Note: $\{u',t\}$ might not be in P. This ensures that $|P| \geq JDAM^{\odot}(\{i,p\},\{j,q\}) - JDAM(\{i,p\},\{j,q\})$. These simple set operations can be done in constant time, and building P takes $O(E+V)$ time over all partition class pairs. Finally we remove (u,v) from P, which could be different from the starting pair if Case 4 or 5a occurs.

It is also possible to keep track of nodes with free stubs in a queue for each part of the partition. Once a node has no free stubs, it will remain so, except *during* neighbor switches. This allows selection of candidates for neighbor switches, or new edges when neighbor switches are not possible, in constant time. However, it still takes $O(d_{max})$ to check whether a node with free stubs is a good candidate for neighbor switch, because the sets of neighbors can be almost the same length, which takes linear time in the size of sets. In the worst case, there is a possibility for at most two neighbor switches per new edge, hence the running time is $O(|E| \cdot d_{max})$.

The directed graph can be constructed from the bipartite representation by collapsing nodes with non-chords and assigning directions to edges appropriately, this takes $O(E+V)$ time.

4.3 Space of Realizations

The algorithm can produce any realization of a realizable D2K, with a non-zero probability. Recall that the order in which the algorithm adds edges is unspecified. Considering all possible edge permutations as the order in which to add the edges, the permutations where no neighbor switch is required correspond to all the possible realizations.[3] Unfortunately, the remaining orderings are difficult

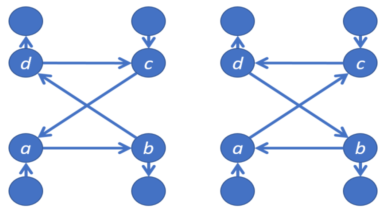

Figure 4: Two realizations with the same degree sequence and JDAM. There is no JDAM preserving double-edge swap that would not use any self-loops and the triangular C_6 [13] swaps are not preserving JDAM. This example shows that the edges along the directed four-cycle must change their direction simultaneously. Therefore, the known swaps that were sufficient for undirected 2K or directed degree sequences are not sufficient for preserving the directed 2K.

to characterize, thus the current algorithm cannot sample uniformly from all realizations with a target D2K during construction.

Another way to sample from the space of graph realizations with a target D2K is by edge rewiring, *after* constructing one such realization. This method is typically used by MCMC approaches that transform one realization to another by rewiring edges so as to present the target properties. On the positive side, D2K is a special case of an undirected JDAM, and thus inherits the property that JDAM realizations are connected via 2K-preserving double-edge swaps [6],[2] if non-chords are allowed (equivalently, self-loops in directed graphs). On the negative side, we cannot use the known swaps to sample from the space of simple directed graphs.

It is known that the space of simple realizations of directed degree sequences is connected over double edge swaps, that preserve (in and out) degrees, and triangular C_6 swaps [13]. Triangular C_6 swaps are necessary in some cases where the difference between two realization is the orientation of a directed three-cycle: in this case, the orientation of the cycle has to be reversed in a single step. The sufficiency of these two types of swaps was shown in [13]. The necessity of these swaps also carries over to (simple) directed 2K realizations. However, Fig. 4 shows a counterexample: a directed four-cycle where the classic swaps are not sufficient to transform one realization to the other, thus requiring a more complex swap. It remains an open question whether tight upper bounds can be derived on the swap size for Directed 2K realizations. [4]

5 EVALUATION ON REAL-WORLD GRAPHS

5.1 Datasets

We have used examples of directed graphs for our experiments from SNAP [26]: p2p-Gnutella08, Wiki-Vote, AS-Caida, Twitter. We have chosen these networks in order to represent several different sizes and generating processes for directed graphs.[5] We removed

[3]In [20], it was shown that given JDMs for every non-isomorphic undirected seven node graphs, the algorithm was indeed able to generate every realization.

[4]There are possibly other cases where more complex swaps are necessary and include more edges at once, for example larger directed cycles with specific in/out degree order. In this paper, we do not provide tight upper bounds on the number of self-loops or the size of swaps required, but we do emphasize that no multi-edges are required and the number of self-loops are of course bounded by $|N|$.

[5]While generating graphs of larger sizes would be feasible using Algorithm 1, computing all the properties mentioned in 5.2 would not be practical.

Table 1: Graphs

Name	#Nodes	#Edges	Generation (sec)
p2p-Gnutella08	6,301	20,777	0.258
Wiki-Vote	7,115	103,689	0.941
AS-Caida	26,475	57,582	0.923
Twitter	81,306	1,768,135	21.292

self-loops or multi-edges from these graphs, since our goal is to produce simple graphs and this step ensures that the measured inputs from these graphs will be realizable using different directed graph construction methods (0K, UMAN, 1K, 2K, 2Km).

AS-Caida has edge labels according to the relationship between two ASes (peer, sibling, provider, customer), however provider and customer edges describe the same relation from the opposite point of view. We have modified the AS-Caida network by removing customer relations between ASes. The affect of the mutual edges - which would be present using both provider and customer relations - will be visited again in our discussion. In [9], this graph was also considered with the relations included, however during our construction, we only use directed dK-series as described before in Section 3.

A summary of the final graphs used in our experiments is shown in Table 1. We also report the average time to generate realizations using D2K for these example networks on a laptop with Intel Core i7 6700HQ processor. Interestingly, the time to generate realizations based on Wiki-Vote and AS-Caida are similar, although the former has twice the edges; this is because the AS-Caida graph needs a larger number of neighbor switches and time to transform the bipartite representation to a directed graph.

5.2 Properties

In our results, we evaluate the performance of D2K graph generators in terms of properties associated with directed graphs. While the size of the generated graphs are maintained (number of nodes and edges), there are many other properties one could investigate. First, we use Degree Distributions and Degree Correlations. These are the ones expected to be exactly matched by definition. In addition, we consider additional properties such as shortest paths, spectrum, strongly connected components, betweenness centrality, and k-core distribution. Finally, to measure how some of the local structures are preserved, we use dyad and triad censuses, dyad-wise shared partners, average neighbor degree, and expansion.

The dyad census counts the different configurations for every pair of nodes: "mutual" - edges in both direction, "asymmetric" - edge only in one direction and "null" - no edge present. The triad census counts the non-isomorphic configuration for every triplet of nodes. A complete list of configurations and naming conventions can be found in [24]. Configurations are identified by three numbers (mutual, asymmetric, and null counts) and a letter in case of different non-isomorphic configuration with the same number of edges. For example "003" is a triplet of nodes where none of the edges are present, "030C" is a directed three-cycle and "300" is a triplet of nodes where all directed edges are present.

Shared partners for pairs of nodes can be defined in three ways for directed graphs: using independent two-paths, using shared outgoing neighbors or using shared incoming neighbors between pairs of nodes [29]. Dyad-wise shared partners (DSP) count node pairs by the number of shared partners appearing in a network.

Average neighbor degree captures the average degree of a node's neighbors, and we split this property for in - and out degrees. Similarly, we refer to *expansion* for directed graphs as the ratio of the first hop and second hop neighborhoods' sizes going out, or coming in to a node. These properties capture similar aspects of a network, but expansion excludes any mutual edges or edges between nodes in the first hop neighbors.

Some of the above properties are also used by Orsini et. al. [28] to study the convergence of dK-series over different types of undirected networks. We have included some properties that are more natural for directed graphs, such as the triad census. The computation of the graph properties of interest is available as part of NetworkX [21] or trivial to implement using the above description.

5.3 Results

We compare realizations generated by Directed ER (D0K), UMAN, Directed Degree Sequence (D1K), Directed 2K, Directed 2Km with the corresponding target properties captured on input graph (G). We use 20 random instances for every construction method and then average our results for each specific property. Due to space constraints, we provide detailed results only for the Twitter graph in Figure 5, 6 and 7 and a brief overview of our observations over different input graphs.

Targeted properties. Fig. 5 shows that Directed Degree Distributions and Degree Correlations are captured by D2K, D2Km as expected by definition. This shows that our implementation is correct. On the other hand, D0K, D1K and UMAN capture Degree Correlations poorly, thus D2K graphs have a possibility to capture other properties more accurately than D0K or D1K.

Dyad Census is not well captured for Twitter, as we can see in Figure 6. However, there are order of magnitude improvements in the number of mutual edges between D2Km (123,040.4) and D2K (3,628.7), D1K (2,155.95) or D0K (233.05). Of course, UMAN preserves this property by definition.

Triad Census is surprisingly well captured by UMAN, the reason being the exact match for the Dyad Census in the previous point. On the other hand, a convergence can be seen between dK-series generators with significant improvements in dense triadic structures from D1K to D2K and from D2K to D2Km.

Betweenness Centrality has no significant improvements after matching degree distributions with D1K in Twitter; other examples reached target closer with D1K. Interestingly UMAN performs almost identically to D0K, even though the number of mutual edges is significantly different.

Shortest Path Distribution has slow convergence to target across different methods, but the average shortest path is shorter than the observed in G.

Strongly Connected Components (SCC) are not well captured by any of the dK-series generators and they tend to produce realizations with a single giant SCC and many one-node components without any intermediate sizes of SCCs.

K-Core Distribution is best captured by D2Km, and there is a small improvement from D1K to D2K using Twitter. However,

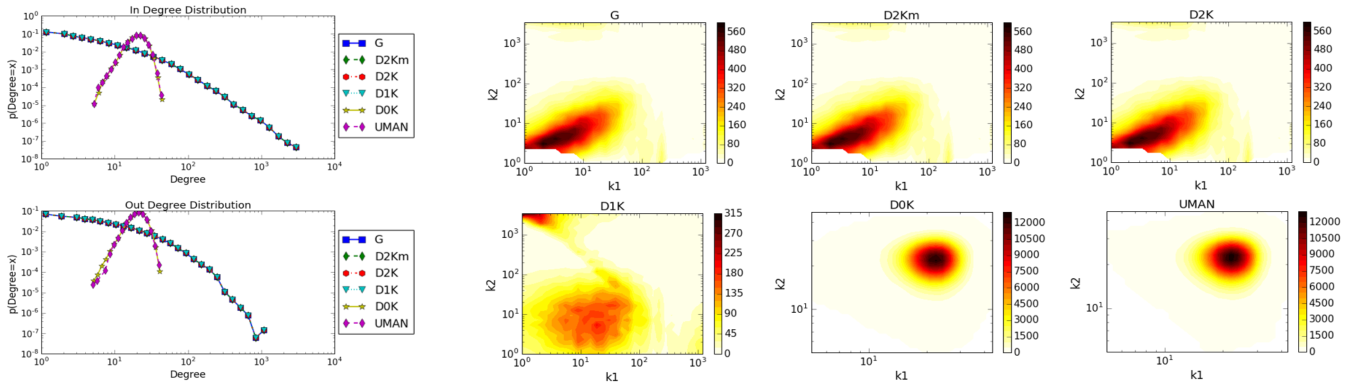

Figure 5: Results for Twitter graph: Directed Degree Distribution and Degree Correlation

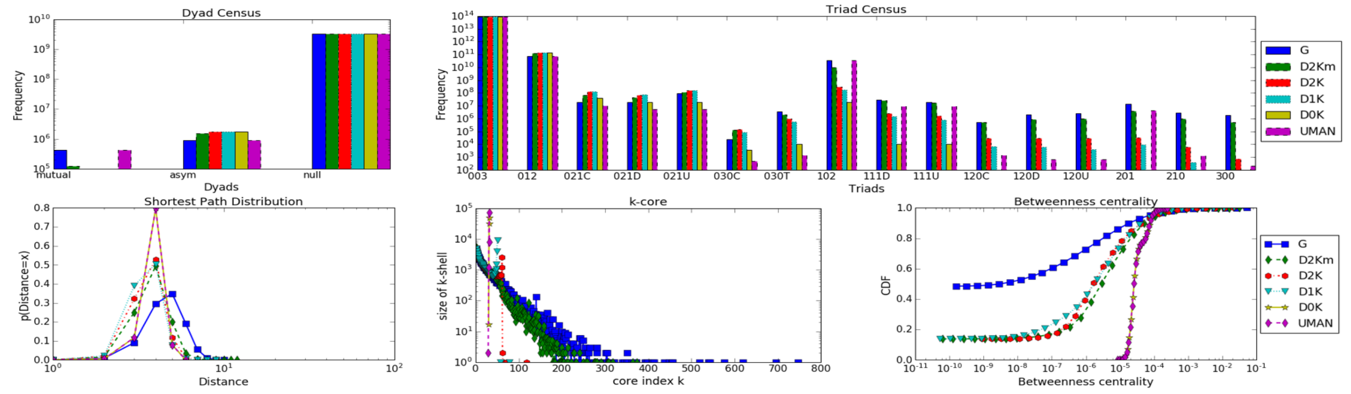

Figure 6: Results for Twitter graph: Dyad-, Triad Census, Shortest Path Distribution, K-core distribution, Betweenness Centrality

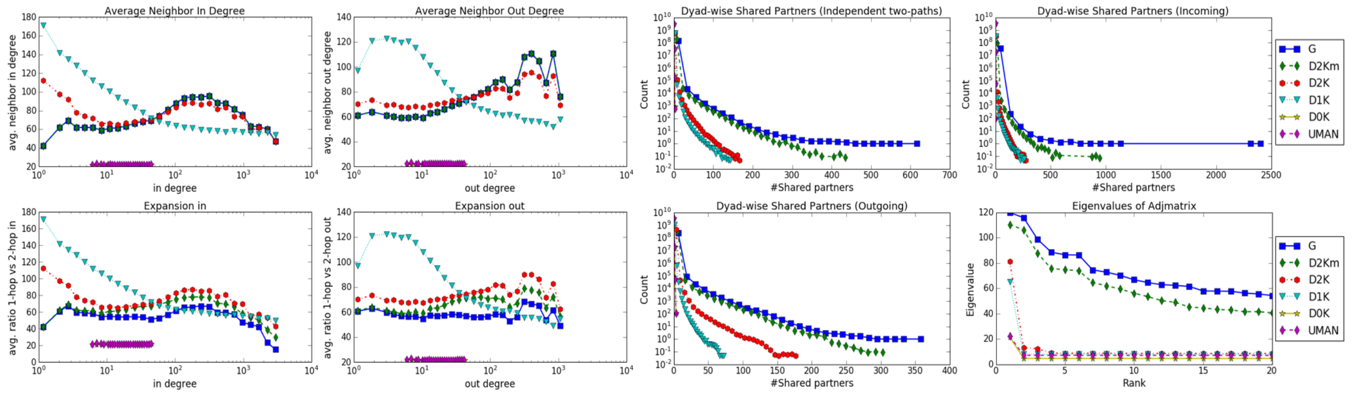

Figure 7: Results for Twitter graph: Expansion, Average Neighbor Degree, DSP and top 20 Eigenvalues

the dense core using D1K or D2K is almost an order of magnitude lower core index.

Eigenvalues of the Twitter graph are again best targeted by D2Km. There is a difference between leading eigenvalues in graph realizations of the other methods but starting at the second eigenvalue the difference between D1K and D2K quickly decreases.

Dyad-wise Shared Partners follow similar trends to other properties, such that D2Km is significantly more accurate than D1K and D2K. D2K improves over D1K in terms of ”outgoing shared partners” but that improvement decreases at ”independent two-paths” and disappears at ”incoming shared partners”.

Table 2: Summary of results: showing improvements by targeting more properties. Labels: "." - no improvement, "-" - decreased accuracy, "+" - increased accuracy, "Exact" - targeted by construction.

Property	UMAN→D1K	D1K→D2K	D2K→D2Km
Degree Distribution	Exact	Exact	Exact
Degree Correlation	+	Exact	Exact
Dyad Census	-	+	+
Triad Census	+	+	+
Betweenness Centrality	+	.	.
Shortest Path Distribution	+	+	+
Eigenvalues	+	+	+
DSP	+	+	+
Expansion	+	+	+
Avg. Neighbor degrees	+	+	Exact
S. Connected Components	.	.	.
K-Core Distribution	+	.	+

Expansion is best approximated by D2Km. D2Km also matches **Average Neighbor Degree** exactly if marginalized by degrees as in Figure 7. D2K also follows the general shape of these distributions but includes larger error, while D1K has systematic difference compared to G.

Table 2 gives an overview of how well network properties are targeted by the different dK graph construction methods for the remaining networks. The Twitter network showcased most of our general findings, but individually some of these networks have characteristics that makes them different from Twitter, *e.g.* p2p-Gnutella08 does not contain any mutual edges. The most interesting question is whether D2K or D2Km capture network properties more accurately compared to D1K. The answer is yes in most cases, but it might not be a significant improvement in targeting certain properties. Local structures are generally better captured by D2K and even more precisely for D2Km, and global properties can be matched moderately better depending on the original network.

5.4 Discussion

We have used several construction algorithms to generate random graphs and we have observed better matching of many properties for higher d in the directed dK-series. However, certain properties are not well captured by these methods, such as the Strongly Connected Components. For some networks D1K could be a good choice, since it captures many network properties. However, we have shown that even in those cases properties related to first hop neighbors are not expected to be captured.

While for most metrics the degree labeled construction (D1K, D2K, D2Km) performs reasonably well, it is important to note that these methods create a low number of mutual edges. On one side, UMAN generates graphs with prescribed number of mutual edges and otherwise ER random graph-like structures. On the other hand, for larger (sparse) networks the degree labeled construction only achieves a fraction of the target number of mutual edges. A solution to this problem would be to generate D2K graphs with a number of mutual edges. It is possible to design heuristics for this problem but exact solutions might be difficult to achieve. In addition, we could consider for D2Km two matrices: one describing asymmetric and another for mutual edges between nodes with given (in, out) degrees.

We have also shown that finer partitioning of nodes, by considering attribute values in addition to their degree, can help to better describe graphs by their mixing matrix. D2Km is a very specific partition, that preserves average neighbor degrees for directed graphs, which is a property that is given by 2K in the undirected case. Graphs could be exactly fixed in the limit of the number of parts in a partition; however, in our example this is not the case. While there are parts of the partition with only a single node in them, most of their edges go to other parts of the partition with multiple nodes. This results in a chance to construct distinct realizations with minimum number of fixed edges across different realizations. It is an interesting question, how different partitions would affect the number of realizations. In prior work, it has been shown how to approximate the number of realizations for a wide class of degree sequences [3]. To the best of our knowledge, the number of realizations for undirected 2K has not been characterized. However, the relation between 1K, 2K and 3K realizations has been shown by the number of possible edge rewirings in [27], which gives an intuition on how more restrictive models shrink the space of realizations. This remains an open question for future work for both undirected and directed JDM, JDAM construction.

The hardness of degree labeled construction for undirected graphs have been shown recently. The realizability problem of undirected 2K with fixed number of triangles, 3K [8], and second order degree sequences (degree and number of two hop neighbors) [14] are NP-Complete. A relaxation of JDM partitions called "PAM" [12] is believed to be NP-Complete and only solved for special cases. More restrictive models (D2K with additional target properties) are likely to lead to NP-Complete problems; however, for any partition of nodes, construction is possible as long as the target JDAM is realizable.

6 CONCLUSION

We have proposed a new approach for directed graph construction by considering in and out degree correlations in addition to directed degree sequences (directed 2K or D2K). We built a framework for directed graphs similar to the dK-series. We could generate bipartite graphs with prescribed degree correlations using the Joint Degree and Attribute Matrix construction algorithms. We used this property of JDAM construction and defined directed 2K as the combination of JDAM and a directed degree sequence (representing non-chords). To solve our D2K problem, we provide the necessary and sufficient conditions for realizability of such inputs and a simple, efficient algorithm to generate simple realizations (without self-loops) as well. The uniform sampling from the space of these realizations remains an open problem, as discussed in section 4.3.

In addition to directed 2K, we have defined D2Km, a special case using additional node attributes. D2Km provides a more restricted notion of directed 2K, that exactly captures average neighbor degree of nodes marginalized by degree. In our experiments, we have shown convergence for degree labeled directed graph construction similar to the undirected case [28]: it shows that degree correlations capture more information about graph structure and enable us to generate graphs which more closely resemble real-life networks.

Overall, our D2K approach advances the state-of-the-art in modeling and simulation of realistic directed graphs, especially in the context of online social networks where degree correlations are important.

ACKNOWLEDGMENTS

This work was supported by NSF Award 1526736. A. Markopoulou, B. Tillman and M. Gjoka were members of CPCC and Networked Systems at UC Irvine.

REFERENCES

[1] William Aiello, Fan Chung, and Linyuan Lu. 2000. A random graph model for massive graphs. In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*. Acm, 171–180.

[2] Georgios Amanatidis, Bradley Green, and Milena Mihail. 2015. Graphic realizations of joint-degree matrices. *arXiv preprint arXiv:1509.07076* (2015).

[3] Alexander Barvinok and John A Hartigan. 2013. The number of graphs and a random graph with a given degree sequence. *Random Structures & Algorithms* 42, 3 (2013), 301–348.

[4] Kevin E Bassler, Charo I Del Genio, Péter L Erdős, István Miklós, and Zoltán Toroczkai. 2015. Exact sampling of graphs with prescribed degree correlations. *New Journal of Physics* 17, 8 (2015), 083052.

[5] Joseph Blitzstein and Persi Diaconis. 2011. A sequential importance sampling algorithm for generating random graphs with prescribed degrees. *Internet Mathematics* 6, 4 (2011), 489–522.

[6] Éva Czabarka, Aaron Dutle, Péter L Erdős, and István Miklós. 2015. On realizations of a joint degree matrix. *Discrete Applied Mathematics* 181 (2015), 283–288.

[7] Charo I Del Genio, Hyunju Kim, Zoltán Toroczkai, and Kevin E Bassler. 2010. Efficient and exact sampling of simple graphs with given arbitrary degree sequence. *PloS one* 5, 4 (2010), e10012.

[8] William Devanny, David Eppstein, and Bálint Tillman. 2016. The computational hardness of dK-series. In *NetSci 2016*.

[9] Xenofontas Dimitropoulos, Dmitri Krioukov, Amin Vahdat, and George Riley. 2009. Graph Annotations in Modeling Complex Network Topologies. *ACM Trans. Model. Comput. Simul.* 19, 4, Article 17 (Nov. 2009), 29 pages. DOI: http://dx.doi.org/10.1145/1596519.1596522

[10] SN Dorogovtsev. 2003. Networks with desired correlations. *arXiv preprint cond-mat/0308336* (2003).

[11] P Erdős and T Gallai. 1960. Gráfok előírt fokú pontokkal. *Mat. Lapok* 11 (1960), 264–274.

[12] Péter L Erdős, Stephen G Hartke, Leo van Iersel, and István Miklós. 2015. Graph realizations constrained by skeleton graphs. *arXiv preprint arXiv:1508.00542* (2015).

[13] Péter L Erdős, Zoltán Király, and István Miklós. 2013. On the swap-distances of different realizations of a graphical degree sequence. *Combinatorics, Probability and Computing* 22, 03 (2013), 366–383.

[14] Péter L Erdős and István Miklós. 2016. Not all simple looking degree sequence problems are easy. *arXiv preprint arXiv:1606.00730* (2016).

[15] Péter L Erdos, István Miklós, and Zoltán Toroczkai. 2015. A decomposition based proof for fast mixing of a Markov chain over balanced realizations of a joint degree matrix. *SIAM Journal on Discrete Mathematics* 29, 1 (2015), 481–499.

[16] Péter L Erdős, István Miklós, and Zoltán Toroczkai. 2016. New classes of degree sequences with fast mixing swap Markov chain sampling. *arXiv preprint arXiv:1601.08224* (2016).

[17] Delbert Ray Fulkerson and others. 1960. Zero-one matrices with zero trace. *Pacific J. Math* 10, 3 (1960), 831–836.

[18] David Gale and others. 1957. A theorem on flows in networks. *Pacific J. Math* 7, 2 (1957), 1073–1082.

[19] Minas Gjoka, Maciej Kurant, and Athina Markopoulou. 2013. 2.5 k-graphs: from sampling to generation. In *INFOCOM, 2013 Proceedings IEEE*. IEEE, 1968–1976.

[20] Minas Gjoka, Bálint Tillman, and Athina Markopoulou. 2015. Construction of simple graphs with a target joint degree matrix and beyond. In *2015 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 1553–1561.

[21] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*. Pasadena, CA USA, 11–15.

[22] S Louis Hakimi. 1962. On realizability of a set of integers as degrees of the vertices of a linear graph. I. *J. Soc. Indust. Appl. Math.* 10, 3 (1962), 496–506.

[23] Václav Havel. 1955. Poznámka o existenci konecnych grafu. *Časopis pro pěstování matematiky* 80, 4 (1955), 477–480.

[24] Paul W Holland and Samuel Leinhardt. 1976. Local structure in social networks. *Sociological methodology* 7 (1976), 1–45.

[25] Hyunju Kim, Charo I Del Genio, Kevin E Bassler, and Zoltán Toroczkai. 2012. Constructing and sampling directed graphs with given degree sequences. *New Journal of Physics* 14, 2 (2012), 023012.

[26] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data. (June 2014).

[27] Priya Mahadevan, Dmitri Krioukov, Kevin Fall, and Amin Vahdat. 2006. Systematic topology analysis and generation using degree correlations. In *ACM SIGCOMM Computer Communication Review*, Vol. 36. ACM, 135–146.

[28] Chiara Orsini, Marija M Dankulov, Pol Colomer-de Simón, Almerima Jamakovic, Priya Mahadevan, Amin Vahdat, Kevin E Bassler, Zoltán Toroczkai, Marián Boguñá, Guido Caldarelli, and others. 2015. Quantifying randomness in real networks. *Nature communications* 6 (2015).

[29] Tom A. B. Snijders, Philippa E. Pattison, Garry L. Robins, and Mark S. Handcock. 2006. New Specifications for Exponential Random Graph Models. *Sociological Methodology* 36 (2006), 99–154.

[30] Isabelle Stanton and Ali Pinar. 2012. Constructing and sampling graphs with a prescribed joint degree distribution. *Journal of Experimental Algorithmics (JEA)* 17 (2012), 3–5.

[31] R Taylor. 1980. *Constrained switchings in graphs*. University of Melbourne, Department of Mathematics.