Generation of graphs

Bipartite graphs – unique perfect matching.

In this section, we assume $G = (V, E)$ bipartite connected graph. The following theorem states that if $G$ has unique perfect matching, then some vertices must have degree one.

**Lemma 1.** Let $G$ be a bipartite graph with bipartition $\{X, Y\}$ and unique perfect matching $M$. Then, there are vertices $x \in X$ and $y \in Y$ with degree one.

Generation from Directed Acyclic Graphs.

We illustrate an algorithm that generates any bipartite graph with unique perfect matching. The method consists in three main steps.

1. **Directed Acyclic Graph.** We start from a Directed Acyclic Graph (DAG) $G_1 = (V_1, E_1)$ with $V_1 = \{v_1, \ldots, v_n\}$. As $G_1$ is acyclic, there exist source $v \in V_1$ and sink $w \in V_1$. In general there may be multiple sources and sinks.

2. **Bipartite Directed Acyclic Graph.** From $G_1$, we build a Bipartite DAG $G_2 = (V_2, E_2)$ in the following way.

   - For each vertex $v \in V_1$ add two vertices $v, v' \in V_2$. Assume that each vertex $v$ and the corresponding vertex $v'$ are well defined.
   
   - $E_2$ is determined as follows. For each $v, v'$ add the directed edge $(v', v) \in E_2$. For each edge $(u, v) \in E_1$ add an edge from $u'$ to $v$ in $E_2$.

   $G_2$ is bipartite with bipartitions $\{v_1, \ldots, v_n\}$ and $\{v'_1, \ldots, v'_n\}$. Each source $v \in G_1$ corresponds to a source $v' \in G_2$ with degree one – the only directed edge adjacent to $v'$ is $(v', v)$. Each sink $w \in G_1$ corresponds to a sink $w \in G_2$ – the only directed edge adjacent to $w$ is $(w', w)$.

3. **Undirected Bipartite Graph with unique perfect matching.** Get $G = (V, E)$ from $G_2$ simply removing the directions of edges in $E_2$. 
Graphs tested in the experiments.

A DAG generates a bipartite graph with unique perfect matching. The unique perfect matching is the set of edges \((v, v')\) for all \(v \in X\). Figure 1 shows a DAG and the corresponding bipartite graph with unique perfect matching. We consider the following parameters for the DAG:

- the number of layers (or the height) of the DAG;
- the number of nodes in each layer of the DAG;
- the average in-degree of the nodes in the DAG that are not sources.

We force no sinks in layers below the top one (because each sink of the DAG translates into a degree-one node in the bipartite graph). Beyond that constraint, the links in the DAG are randomly generated, given the average in-degree. Instead, the bipartite graph is determined in a unique way from the DAG.

Figure 1(a) shows the example of a DAG with 6 layers, two nodes in levels 1 and 6, and one node in levels 2, 3, 4, 5, and average in-degree of 2. The short notation for this graph is 2-1-1-1-2(2), with the following meaning: first the number of nodes in each level from 1 to 6, then the average in-degree in brackets.

The following table contains the DAGs used to generate the bipartite graphs with unique perfect matching that are tested in the experiments. The nomenclature is the one described above. The DAGs are ordered according to the expected difficulty of the distributed matching game on the resulting bipartite graphs. We expect that the game difficulty is increased by the following: higher average in-degree of the DAG, lower number of sinks/sources in the DAG, longer paths in the DAG.

<table>
<thead>
<tr>
<th>Easier</th>
<th>2-4-2(2), 3-2-3(2), 3-1-1-3(2), 1-6-1(2), 2-2-2-2(2), 2-1-1-1-1-2(2), 2-4-2(3), 3-2-3(3), 3-1-1-3(3).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1-3-3-1(2), 1-2-2-2-1(2), 1-1-4-1-1(2), 1-2-1-1-2-1(2), 1-1-1-1-1-1-1-1(2).</td>
</tr>
<tr>
<td>Less easy</td>
<td>1-3-3-1(3), 1-2-2-2-1(3), 1-1-4-1-1(3), 1-2-1-1-2-1(3), 1-1-1-1-1-1-1-1(3).</td>
</tr>
</tbody>
</table>

Bipartite graphs – multiple perfect matching.

In this section we show how to generate bipartite graphs with multiple perfect matching. The method is based on the generation of bipartite graphs with
unique perfect matching from a DAG, presented in the previous section. Indeed, a bipartite graph with multiple perfect matching is obtained from a directed cyclic graph (DCG) that is obtained adding directed links to a DAG.

**Generation from DCG and number of perfect matchings.**

Consider a DCG $G' = (V_1, E')$. Clearly, $G'$ can be obtained from a DAG $G_1 = (V_1, E_1)$ by adding directed links from nodes in higher layers to nodes in lower layers. The bipartite graph $G = (V, E)$ is obtained as follows – we do not consider the intermediate step of the directed bipartite graphs.

- For each vertex $v \in V_1$ add two vertices $v, v' \in V$. Assume that each

Figure 1: Bipartite graph generated from a DAG. Node $n$ in the DAG corresponds to node $2n - 1$ in the bipartite graph.
vertex $v$ and the corresponding vertex $v'$ are well defined.

- $E$ is determined as follows. For each $v, v' \in V$ add edge $(v', v) \in E$. For each edge $(u, v) \in E'$ add an edge $(u', v) \in E$.

The unique perfect matching on the bipartite graph obtained from the corresponding DAG is still a perfect matching on the bipartite graph $G$ obtained from the DCG.

The DAG has a certain number of cycles, that we denote as $C_1, \ldots, C_M$. We say that two cycles are disjoint if they have no edge in common. Each cycle $C_i$ (of length $l_i$) in the DCG corresponds to a cycle $C'_i$ (of length $2l_i$) in the bipartite graph $G$. If $C_1$ and $C_2$ are disjoint then $C'_1$ and $C'_2$ are also disjoint.

We know characterize the number of perfect matchings in the bipartite graph with respect to the number of cycles in the DCG. Orient the edges $(v, v')$ for all $v \in X$ in one direction – they are the edges of the unique perfect matching in the bipartite graph obtained from the starting DAG. Orient all remaining edges of $G$ in the opposite direction.

Each cycle $C'_i$ in $G$ (corresponding to cycle $C_i$ in the DCG) is an alternating cycle of even length. The nodes in $C'_i$ can be perfectly matched in two ways. If $C'_1$ and $C'_2$ are disjoint cycles, then the nodes in $C'_1$ and the nodes in $C'_2$ can be matched in two ways each, for a total of four possibilities.

For every subset of disjoint cycles $\{C'_1, \ldots, C'_M\} \subseteq \{C'_1, \ldots, C'_M\}$ there is a corresponding perfect matching on $G$. Note that the empty set of cycles corresponds to the perfect matching on the graph generated from the DAG.

**Graphs to test in the experiments.**

We aim to study the effects cycles have on the distributed matching game. In particular we separately investigate the effect of short disjoint cycles and long non-disjoint cycles.

**Short disjoint cycles.** Starting from a DAG we consider the following three DCGs, obtained forming short disjoint cycles.

1. Add two disjoint two-nodes cycles to the DAG, reducing the number of sinks and sources by two – remind that for Theorem 1 there are always a sink and a source in the DAG. Choose the cycles in a way that it is possible to add two more disjoint two-nodes cycles. Call this
graph $DCG_1$ and the resulting bipartite graph $G_1$. $G_1$ has four perfect matchings.

2. Add two disjoint two-nodes cycles to the DAG, trying not to reduce the number of sinks and sources. Choose the cycles in a way that it is possible to add two more disjoint two-nodes cycles. In particular, we choose two cycles that are disjoint from the ones of $DCG_1$. Call this graph $DCG_2$ and the resulting bipartite graph $G_2$. $G_2$ has four perfect matchings.

3. Add four disjoint two-nodes cycles to the DAG. In particular we consider the cycles in $DCG_1$ and $DCG_2$. Call this graph $DCG_3$ and the resulting bipartite graph $G_3$. $G_3$ has sixteen perfect matchings.

**Long cycles.** Starting from a DAG we consider the following two DCGs, obtained by adding long and non-disjoint cycles.

4. Add a link that creates a long cycle as well as other cycles, some of which are non-disjoint. In particular, we choose a link from a sink to a source. Call this graph $DCG_4$ and the resulting bipartite graph $G_4$. The number of perfect matchings on $G_4$ equals the number of all possible sets of disjoint cycles on $DCG_4$.

5. Add a further long link that creates a new long cycle and other non-disjoint cycles. Call this graph $DCG_5$ and the resulting bipartite graph $G_5$. The number of perfect matchings on $G_5$ still depends on the cycles, and each perfect matching on $G_4$ is a perfect matching also on $G_5$.

**Experiments.** We saw how from a DAG we generate the bipartite graph $G$. Then, adding directed links to the DAG we obtain five distinct directed graphs, and then the bipartite graphs $G_1, \ldots, G_5$ with multiple perfect matching.

We generate the DCGs starting from the following DAGs from the previous section: 2-4-2(3), 2-2-2-2(2), 1-3-3-1(2), 1-2-2-2-1(2), 1-1-1-1-1-1-1-1(2).

**General graphs – unique perfect matching.**

We begin this section with two useful facts.
Lemma 2. If a graph has unique perfect matching, then there exists a bridge edge.

Let $G = (V, E)$ be a graph with unique perfect matching: clearly $|V|$ is even. Given $u, v \in V$ such that $(u, v) \in E$ is a bridge, let $G_u, G_v \subset G$ be the connected components obtained from $G$ deleting $(u, v)$ and respectively containing $u$ and $v$. $|G_u|$ and $|G_v|$ are the cardinalities of $V(G_u)$ and $V(G_v)$.

Lemma 3. Let $G = (V, E)$ be a graph with unique perfect matching, and let $(u, v)$ be a bridge edge. If $|G_u|$ and $|G_v|$ are odd then $(u, v) \in \mathcal{M}$. If $|G_u|$ and $|G_v|$ are even then $(u, v) \notin \mathcal{M}$.

Theorem 2 and Theorem 3 suggest both a method to find the unique perfect matching on a graph, and a recursive method to generate graphs with unique perfect matching. Here, we focus on the latter.

**Generation by bridge criterion.**

Let $n$ be an even number and let $V = \{v_1, \ldots, v_n\}$. Suppose we want to generate a connected graph with vertices $V$ and unique perfect matching. For Theorem 2 there must be a bridge. In the generation of the graph, we decide where the bridge is – in particular we decide the numbers $n_1$ and $n_2$ of vertices there are on each side of the bridge. We denote the ends of the bridge by $v_1$ and $v_2$, and add an edge between them. Let $V_1$ be the set of $n_1$ vertices on the first side of the bridge (including $v_1$) and $V_2$ be the set of $n_2 = n - n_1$ vertices on the other side of the bridge (including $v_2$). We distinguish two cases.

1. $n_1$ and $n_2$ are **even**: the bridge is not part of the perfect matching.
   Run the generation process separately on the sets $V_1$ and $V_2$ – generate two graphs with respectively $n_1$ and $n_2$ nodes, unique perfect matching, and connected by the bridge $(v_1, v_2)$.

2. $n_1$ and $n_2$ are **odd**: the bridge $(v_1, v_2)$ is part of the perfect matching.
   Let $V'_1 = V_1 \setminus \{v_1\}$ and $V'_2 = V_2 \setminus \{v_2\}$. Note that $V'_1$ and $V'_2$ have even number of vertices. Divide $V'_1$ in $l_1 \leq |V'_1|/2$ sets $V'_{1,1}, \ldots, V'_{1,l_1}$ of even number of nodes each. Run the generation process separately on the $l_1$ sets of vertices, and generate $l$ graphs $G'_{1,1}, \ldots, G'_{1,l_1}$ with unique perfect matching. Each graph is then connected to $v_1$ with at least one
link – the links can be chosen arbitrarily and are not part of the perfect matching as $(v_1, v_2)$ is guaranteed to be a bridge contained in the perfect matching. Proceed similarly with the vertices in $V'_2$, generating graphs $G_{2,1}, \ldots, G_{2,l_2}$ with unique perfect matching and connected to $v_2$.

The recursive construction ends with a connected graph with unique perfect matching. Note that the construction ends because finally each vertex ends in a one-node set.

**Implementation.** Consider a set $V = \{v_1, \ldots, v_n\}$ of even number $n$ of nodes. Let $E$ be a $n \times n$ matrix with all zero elements. Decide that there is a bridge between nodes $v_a$ and $v_b$. Set $e_{a,b} = e_{b,a} = 1$. Suppose bridge $(v_a, v_b)$ divides $V$ in sets $V_a = \{v_a, v_{i_1}, \ldots, v_{i_k}\}$ and $V_b = V \setminus V_a = \{v_b, v_{j_1}, \ldots, v_{j_l}\}$.

As usual we distinguish two cases.

- $|V_a|$ and $|V_b|$ even. The bridge $(v_a, v_b)$ is not part of the perfect matching. Run the generation process separately on sets $V_a$ and $V_b$, getting the matrices $E(V_a, V_a)$ and $E(V_b, V_b)$ as outputs.

- $|V_a|$ and $|V_b|$ odd. The bridge $(v_a, v_b)$ is part of the perfect matching. Any arbitrary number of edges in $\{v_{i_1}, \ldots, v_{i_k}\}$ can be connected to $v_a$ (setting the corresponding elements of $E(V_a, V_a)$ to 1), and an arbitrary number of edges in $\{v_{j_1}, \ldots, v_{j_l}\}$ can be connected to $v_b$. Run the generation process separately on sets $U_a = V_a \setminus \{v_a\}$ and $U_b = V_b \setminus \{v_b\}$ that have even cardinality, getting the matrices $E(U_a, U_a)$ and $E(U_b, U_b)$ as outputs.

All the recursions stop when they get to all one-node sets. At the end of the generation process, $E$ is the adjacency matrix of a graph $G$ with unique perfect matching.

**Graphs to test in the experiments.**

In the previous section we observed how we can increase the graph’s connectivity only when we divide an even set of vertices into two odd sets, i.e. by adding edges to the ends of a matching bridge. In the generation of the experiment’s graphs we always divide even sets into odd sets, because this allows us to get both higher and lower connectivity.

The generation method presented above offers great variability in the generation of the graphs. We try to limit this variability, and in particular we focus our attention on the following.
1. The cardinalities $n_1, n_2$ of the two sets the vertices are divided into, during the first step of the generation. This choice identifies the position of a bridge (or of the bridge) of the graph. In all other steps, we divide $n$ vertices into sets of cardinality $n - 1$ and 1.

2. As previously explained, when a bridge $(u, v)$ divides $G$ into $G_u$ and $G_v$ off odd cardinality, the set $U_v = V(G_v) - \{v\}$ ($U_u = V(G_u) - \{u\}$) can be divided in different subsets of even cardinality, each connected to $v$ ($u$). We consider two cases. In the first, $U_v$ is never divided into different sets. In the second, at the first step $U_v$ is divided into two sets – while not dividing $U_u$.

We consider the following choices of 16-node graphs.

A. At the first step, place the bridge dividing sets of 15 and 1 vertices – with no loss of generality the bridge is (1, 2). In all next steps divide $n$ vertices into sets of $n - 1$ and 1. At all steps, $U_v$ is never divided into different sets.

B. At the first step, place the bridge dividing sets of 9 and 7 vertices – with no loss of generality the bridge is (9, 10). In all next steps divide $n$ vertices into sets of $n - 1$ and 1. At all steps, $U_v$ is never divided into different sets.

C. At the first step, place the bridge dividing sets of 13 and 3 vertices – with no loss of generality the bridge is (13, 14). Moreover, $U_{13}$ is divided into two sets of six nodes each. In all the next steps, divide $n$ vertices into sets of $n - 1$ and 1, and never divide $U_v$. We further decide that nodes 14, 15, 16 form a 3-node clique.

D. At the first step, place the bridge dividing sets of 13 and 3 vertices – with no loss of generality the bridge is (13, 14). Moreover, $U_{13}$ is divided into three sets of four nodes each. In all the next steps, divide $n$ vertices into sets of $n - 1$ and 1, and never divide $U_v$. We further decide that nodes 14, 15, 16 form a 3-node clique.

Based on each of these three choices, we can vary the connectivity of the graph when we connect vertices with the ends of a bridge. In particular, we consider a connectivity ratio $r$ that expresses the fraction of nodes that are connected to a bridge end (among all nodes that in the current step can be connected to it). We consider the following values for $r$. 

8
- $r = 1$: always add the maximum number of connections, given the choice $A, B, C, D$ for the bridge. In the case of choice $A$, the maximum number of edges in a graph with unique perfect matching is achieved (refer to section ??). Note that, given the bridge choice at each step, $r = 1$ gives a unique possible output graph.

- $r = 1/2$: at each step, randomly choose half of all the possible connections to the bridge end, with the constraint not to get degree-1 nodes (one degree-1 node is of course allowed in case $A$). Note that, given the bridge choice at each step, $r = 1/2$ allows to generate many graphs. So, we generate two graphs for each of the choices $A, B, C$, and one for choice $D$.

- $r = 1/4$: at each step, randomly choose one fourth of all the possible connections to the bridge end, with the constraint not to get degree-1 nodes (one degree-1 node is of course allowed in case $A$). Note that, given the bridge choice at each step, $r = 1/4$ allows to generate many graphs. So, we generate two graphs for each of the choices $A, B$, and one for choice $C$.

**Graph names.** The graph name has the form $sXcYrZ.N$. $X$ is the index of the (lower numbered) end of the first bridge. $Y$ is the number of sets $U_v$ is divided into after the first step. $Z$ is the value of $100 \times r$. $N$ is an increasing number to order graphs of the same class. We consider the following graphs.

- $s15c1r100_1, s15c1r50_1, s15c1r50_2, s15c1r25_1, s15c1r25_2$ for choice $A$.
- $s9c1r100_1, s9c1r50_1, s9c1r50_2, s9c1r25_1, s9c1r25_2$ for choice $B$.
- $s13c2r100_1, s13c2r50_1, s13c2r50_2, s13c2r25_1$ for choice $C$.
- $s13c3r100_1, s13c3r50_1$ for choice $D$.

**General graphs – multiple perfect matching.**

Given a graph $G = (V, E)$ and a matching $M \subseteq E$ on $G$, a path $P$ is alternating with respect to $M$ if it alternates matching edges with non-matching
edges. An alternating cycle $C$ is similarly defined. Obviously, an alternating cycle has even length. The following results states that what identifies the unicity (multiplicity) of the perfect matching is the existence (absence) of alternating cycle.

**Lemma 4.** Let $M$ be a perfect matching on graph $G$.

Then, $G$ has unique perfect matching if and only if it contains no cycle that is alternating with respect to $M$.

Equivalently, $G$ has multiple matching if and only if there exists a cycle that is alternating with respect to $M$.

Lemma 4 suggests a criterion to generate general graphs with multiple perfect matching.

- Generate a graph $G$ with unique perfect matching $M$.
- Add edges to $G$ that create alternating cycles with respect to $M$. The resulting graph $G'$ has multiple perfect matching.

**Graphs to test in the experiments.**

For each graph $G$ with unique perfect matching considered in the previous section, we generate a corresponding graph $G'$ with multiple perfect matching by adding edges that create alternating cycles.

**Small World networks**

We generated 3 SW networks with the function `watts_strogatz_graph(n, k, p)` of the NetworkX Python package, where $n = 24$ is the number of nodes, $k = 4$ is the number of nearest neighbors each node is connected to, and $p = 0.1$ is the rewiring parameter of each edges. We made sure that each generated networks had at least a perfect matching.

**Preferential Attachment networks**

We generated 3 PA networks with the function `barabasi_albert_graph(n, m)` of the NetworkX Python package, where $n = 24$ is the number of nodes, $m = 2$ is the number of edges to attach from a new node to existing nodes. We made sure that each generated networks had at least a perfect matching.