Matchings and the Switch Chain

Martin Dyer

University of Leeds

Simons Institute
Berkeley

June 7th, 2017

Joint work with Haiko Müller

Perfect matchings

A *matching* in a graph $G = (V, E)$ is a set of independent edges.

A *perfect matching* in an n-vertex graph G is a set of $n/2$ independent edges. Clearly n must be even.

Deciding if G has a perfect matching, and finding one if it does, is in P (Edmonds, 1965), but counting the number of perfect matchings *exactly* is known to be #P-complete (Valiant, 1979).

If G is *bipartite*, the number of perfect matchings in G is called the 0-1 *permanent*, and this case remains #P-complete.

Perfect matchings

A *matching* in a graph $G = (V, E)$ is a set of independent edges.

A *perfect matching* in an n-vertex graph G is a set of $n/2$ independent edges. Clearly n must be even.

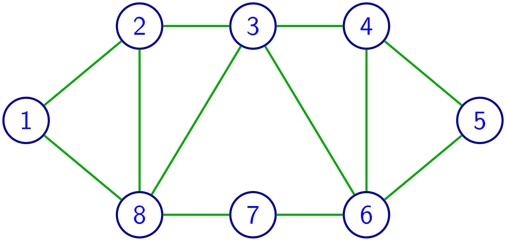
Deciding if G has a perfect matching, and finding one if it does, is in P (Edmonds, 1965), but counting the number of perfect matchings *exactly* is known to be #P-complete (Valiant, 1979).

If G is *bipartite*, the number of perfect matchings in G is called the 0-1 *permanent*, and this case remains #P-complete.

Perfect matchings

A *matching* in a graph $G = (V, E)$ is a set of independent edges.

A *perfect matching* in an n-vertex graph G is a set of $n/2$ independent edges. Clearly n must be even.



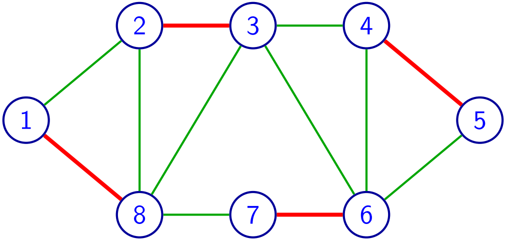
Deciding if G has a perfect matching, and finding one if it does, is in P (Edmonds, 1965), but counting the number of perfect matchings *exactly* is known to be #P-complete (Valiant, 1979).

If G is *bipartite*, the number of perfect matchings in G is called the 0-1 *permanent*, and this case remains #P-complete.

Perfect matchings

A *matching* in a graph $G = (V, E)$ is a set of independent edges.

A *perfect matching* in an n-vertex graph G is a set of $n/2$ independent edges. Clearly n must be even.



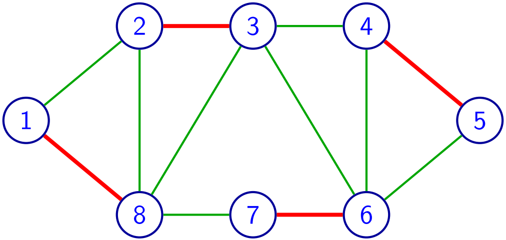
Deciding if G has a perfect matching, and finding one if it does, is in P (Edmonds, 1965), but counting the number of perfect matchings *exactly* is known to be #P-complete (Valiant, 1979).

If G is *bipartite*, the number of perfect matchings in G is called the 0-1 *permanent*, and this case remains #P-complete.

Perfect matchings

A *matching* in a graph $G = (V, E)$ is a set of independent edges.

A *perfect matching* in an n-vertex graph G is a set of $n/2$ independent edges. Clearly n must be even.



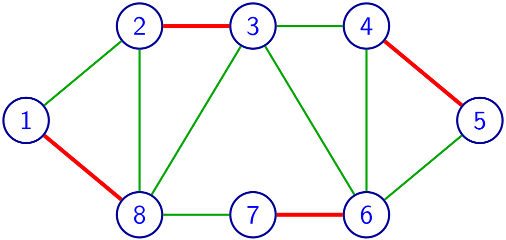
Deciding if G has a perfect matching, and finding one if it does, is in P (Edmonds, 1965), but counting the number of perfect matchings *exactly* is known to be #P-complete (Valiant, 1979).

If G is *bipartite*, the number of perfect matchings in G is called the 0-1 *permanent*, and this case remains #P-complete.

Perfect matchings

A *matching* in a graph $G = (V, E)$ is a set of independent edges.

A *perfect matching* in an n-vertex graph G is a set of $n/2$ independent edges. Clearly n must be even.



Deciding if G has a perfect matching, and finding one if it does, is in P (Edmonds, 1965), but counting the number of perfect matchings *exactly* is known to be #P-complete (Valiant, 1979).

If G is *bipartite*, the number of perfect matchings in G is called the 0-1 *permanent*, and this case remains #P-complete.

Approximate counting

Jerrum, Valiant & Vazirani (1986) showed that *approximate* counting and sampling almost uniformly at random are equivalent for *self-reducible* problems, such as counting perfect matchings.

Jerrum & Sinclair (1989) used a Markov chain on perfect and *near-perfect* matchings ($n/2 - 1$ edges) to approximate the number of perfect matchings. The chain converges in polynomial time (rapid mixing) provided there are not exponentially more near-perfect than perfect matchings. (G is *P-stable*).

Jerrum, Sinclair & Vigoda (2004) showed that the permanent can be *approximated* in polynomial time, settling the question completely for bipartite graphs. Their algorithm involves running a sequence of Markov chains multiple times. The running time is $O(n^7 \log^4 n)$ (Bezáková, Štefankovič, Vazirani & Vigoda, 2008).

For general nonbipartite graphs, the question remains open.

Approximate counting

Jerrum, Valiant & Vazirani (1986) showed that *approximate* counting and sampling almost uniformly at random are equivalent for *self-reducible* problems, such as counting perfect matchings.

Jerrum & Sinclair (1989) used a Markov chain on perfect and *near-perfect* matchings ($n/2 - 1$ edges) to approximate the number of perfect matchings. The chain converges in polynomial time (rapid mixing) provided there are not exponentially more near-perfect than perfect matchings. (G is *P-stable*).

Jerrum, Sinclair & Vigoda (2004) showed that the permanent can be *approximated* in polynomial time, settling the question completely for bipartite graphs. Their algorithm involves running a sequence of Markov chains multiple times. The running time is $O(n^7 \log^4 n)$ (Bezáková, Štefankovič, Vazirani & Vigoda, 2008).

For general nonbipartite graphs, the question remains open.

Approximate counting

Jerrum, Valiant & Vazirani (1986) showed that *approximate* counting and sampling almost uniformly at random are equivalent for *self-reducible* problems, such as counting perfect matchings.

Jerrum & Sinclair (1989) used a Markov chain on perfect and *near-perfect* matchings ($n/2 - 1$ edges) to approximate the number of perfect matchings. The chain converges in polynomial time (rapid mixing) provided there are not exponentially more near-perfect than perfect matchings. (G is *P-stable*).

Jerrum, Sinclair & Vigoda (2004) showed that the permanent can be *approximated* in polynomial time, settling the question completely for bipartite graphs. Their algorithm involves running a sequence of Markov chains multiple times. The running time is $O(n^7 \log^4 n)$ (Bezáková, Štefankovič, Vazirani & Vigoda, 2008).

For general nonbipartite graphs, the question remains open.

Approximate counting

Jerrum, Valiant & Vazirani (1986) showed that *approximate* counting and sampling almost uniformly at random are equivalent for *self-reducible* problems, such as counting perfect matchings.

Jerrum & Sinclair (1989) used a Markov chain on perfect and *near-perfect* matchings ($n/2 - 1$ edges) to approximate the number of perfect matchings. The chain converges in polynomial time (rapid mixing) provided there are not exponentially more near-perfect than perfect matchings. (G is *P-stable*).

Jerrum, Sinclair & Vigoda (2004) showed that the permanent can be *approximated* in polynomial time, settling the question completely for bipartite graphs. Their algorithm involves running a sequence of Markov chains multiple times. The running time is $O(n^7 \log^4 n)$ (Bezáková, Štefankovič, Vazirani & Vigoda, 2008).

For general nonbipartite graphs, the question remains open.

The switch Markov chain

Diaconis, Graham & Holmes (2001) proposed a simple Markov chain for sampling perfect matchings almost uniformly at random.

Let the matching at time t be M_t.

Switch chain

(1) Set $t \leftarrow 0$, and find any perfect matching M_0 in G.

(2) Choose $v, v' \in V$, uniformly at random. Let $u, u' \in V$ be such that uv, $u'v' \in M_t$.

(3) If $u'v$, $uv' \in E$, set $M_{t+1} \leftarrow \{u'v, uv'\} \cup M_t \setminus \{uv, u'v'\}$.

(4) Otherwise, set $M_{t+1} \leftarrow M_t$.

(5) Set $t \leftarrow t + 1$. If $t < t_{\max}$, repeat from (2). Otherwise, stop.

The chain involves switching two matchings edges in a 4-cycle for two non-matching edges.

There is clearly a question about the *ergodicity* of the chain, before considering its mixing time.

The switch Markov chain

Diaconis, Graham & Holmes (2001) proposed a simple Markov chain for sampling perfect matchings almost uniformly at random.

Let the matching at time t be M_t.

Switch chain

(1) Set $t \leftarrow 0$, and find any perfect matching M_0 in G.

(2) Choose $v, v' \in V$, uniformly at random. Let $u, u' \in V$ be such that $uv,\ u'v' \in M_t$.

(3) If $u'v,\ uv' \in E$, set $M_{t+1} \leftarrow \{u'v, uv'\} \cup M_t \setminus \{uv, u'v'\}$.

(4) Otherwise, set $M_{t+1} \leftarrow M_t$.

(5) Set $t \leftarrow t + 1$. If $t < t_{\text{max}}$, repeat from (2). Otherwise, stop.

The chain involves switching two matchings edges in a 4-cycle for two non-matching edges.

There is clearly a question about the *ergodicity* of the chain, before considering its mixing time.

The switch Markov chain

Diaconis, Graham & Holmes (2001) proposed a simple Markov chain for sampling perfect matchings almost uniformly at random.

Let the matching at time t be M_t.

Switch chain

(1) Set $t \leftarrow 0$, and find any perfect matching M_0 in G.

(2) Choose $v, v' \in V$, uniformly at random. Let $u, u' \in V$ be such that $uv,\ u'v' \in M_t$.

(3) If $u'v,\ uv' \in E$, set $M_{t+1} \leftarrow \{u'v, uv'\} \cup M_t \setminus \{uv, u'v'\}$.

(4) Otherwise, set $M_{t+1} \leftarrow M_t$.

(5) Set $t \leftarrow t + 1$. If $t < t_{\mathrm{max}}$, repeat from (2). Otherwise, stop.

The chain involves switching two matchings edges in a 4-cycle for two non-matching edges.

There is clearly a question about the *ergodicity* of the chain, before considering its mixing time.

The switch Markov chain

Diaconis, Graham & Holmes (2001) proposed a simple Markov chain for sampling perfect matchings almost uniformly at random.

Let the matching at time t be M_t.

Switch chain

(1) Set $t \leftarrow 0$, and find any perfect matching M_0 in G.

(2) Choose $v, v' \in V$, uniformly at random. Let $u, u' \in V$ be such that uv, $u'v' \in M_t$.

(3) If $u'v$, $uv' \in E$, set $M_{t+1} \leftarrow \{u'v, uv'\} \cup M_t \setminus \{uv, u'v'\}$.

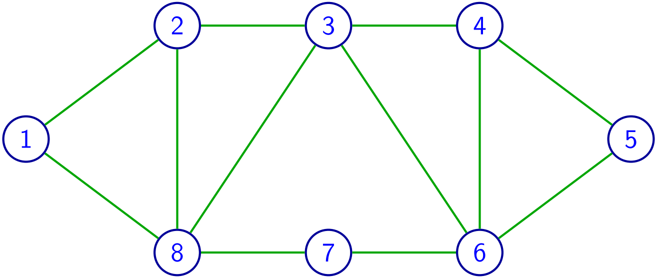
(4) Otherwise, set $M_{t+1} \leftarrow M_t$.

(5) Set $t \leftarrow t + 1$. If $t < t_{\max}$, repeat from (2). Otherwise, stop.

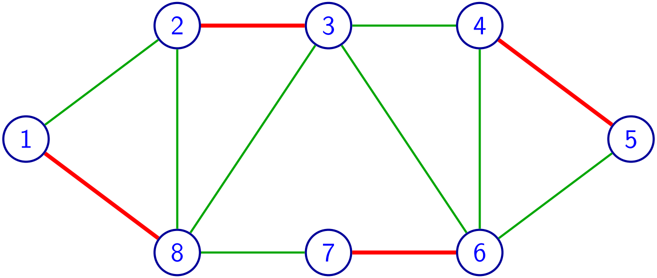
The chain involves switching two matchings edges in a 4-cycle for two non-matching edges.

There is clearly a question about the *ergodicity* of the chain, before considering its mixing time.

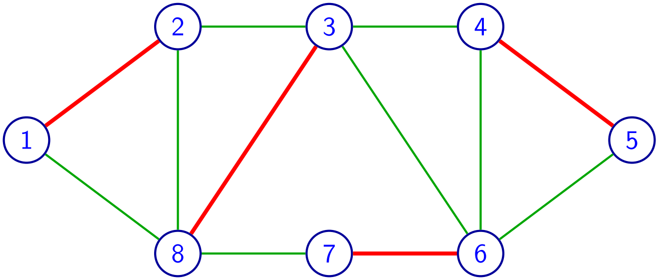
Example



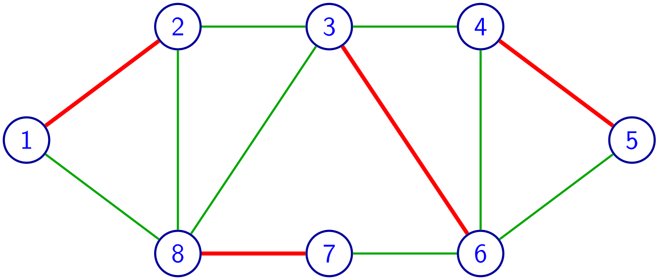
Example



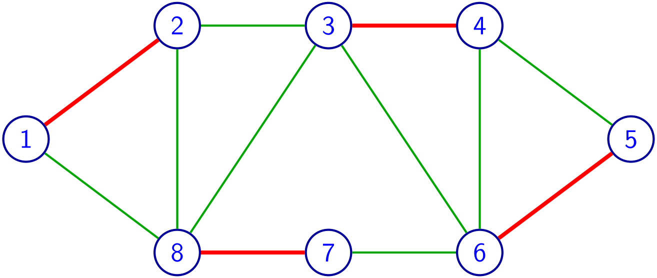
Example



Example

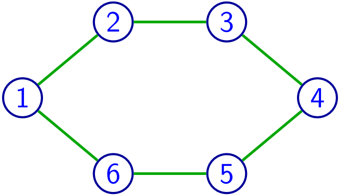


Example



Ergodicity

Diaconis, Graham & Holmes observed that this chain is not ergodic in general. They gave a simple bipartite example

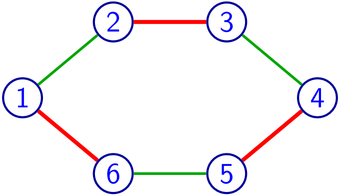


This graph has two perfect matchings, but the chain cannot move between them, because the graph is a chordless 6-cycle.

A *bipartite* graph with no chordless cycle of length greater than 4 is called *chordal bipartite*. D, Jerrum & Müller (2016) showed that the switch chain is ergodic for this class, and that any two matchings are connected by a sequence of at most $n/2$ matchings.

Ergodicity

Diaconis, Graham & Holmes observed that this chain is not ergodic in general. They gave a simple bipartite example

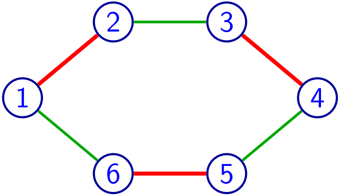


This graph has two perfect matchings, but the chain cannot move between them, because the graph is a chordless 6-cycle.

A *bipartite* graph with no chordless cycle of length greater than 4 is called *chordal bipartite*. D, Jerrum & Müller (2016) showed that the switch chain is ergodic for this class, and that any two matchings are connected by a sequence of at most $n/2$ matchings.

Ergodicity

Diaconis, Graham & Holmes observed that this chain is not ergodic in general. They gave a simple bipartite example

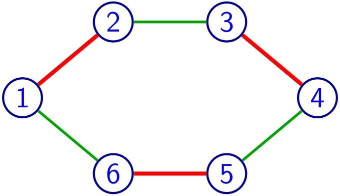


This graph has two perfect matchings, but the chain cannot move between them, because the graph is a chordless 6-cycle.

A *bipartite* graph with no chordless cycle of length greater than 4 is called *chordal bipartite*. D, Jerrum & Müller (2016) showed that the switch chain is ergodic for this class, and that any two matchings are connected by a sequence of at most $n/2$ matchings.

Ergodicity

Diaconis, Graham & Holmes observed that this chain is not ergodic in general. They gave a simple bipartite example



This graph has two perfect matchings, but the chain cannot move between them, because the graph is a chordless 6-cycle.

A *bipartite* graph with no chordless cycle of length greater than 4 is called *chordal bipartite*. D, Jerrum & Müller (2016) showed that the switch chain is ergodic for this class, and that any two matchings are connected by a sequence of at most $n/2$ matchings.

When is the switch chain ergodic?

The general question is: Given a graph G, is the switch chain ergodic on G?

Settling the complexity of this question is difficult, since we have no polynomial bound on the length of the sequence of switches connecting two matchings. If we had such a bound, the question would be within the second level of the polynomial hierarchy. In fact, we can only put the problem in PSPACE. However, our best lower bound on the length of the sequence is only $\Omega(n^2)$.

For this reason, and also to have self-reducibility, we restrict attention to *hereditary* classes of graphs. These are lasses for which every (vertex) induced subgraph of a graph in the class is also in the class.

D, Jerrum & Müller (2016) showed that chordal bipartite graphs form the largest hereditary class of bipartite graphs for which the switch chain is ergodic.

When is the switch chain ergodic?

The general question is: Given a graph G, is the switch chain ergodic on G?

Settling the complexity of this question is difficult, since we have no polynomial bound on the length of the sequence of switches connecting two matchings. If we had such a bound, the question would be within the second level of the polynomial hierarchy. In fact, we can only put the problem in PSPACE. However, our best lower bound on the length of the sequence is only $\Omega(n^2)$.

For this reason, and also to have self-reducibility, we restrict attention to *hereditary* classes of graphs. These are lasses for which every (vertex) induced subgraph of a graph in the class is also in the class.

D, Jerrum & Müller (2016) showed that chordal bipartite graphs form the largest hereditary class of bipartite graphs for which the switch chain is ergodic.

When is the switch chain ergodic?

The general question is: Given a graph G, is the switch chain ergodic on G?

Settling the complexity of this question is difficult, since we have no polynomial bound on the length of the sequence of switches connecting two matchings. If we had such a bound, the question would be within the second level of the polynomial hierarchy. In fact, we can only put the problem in PSPACE. However, our best lower bound on the length of the sequence is only $\Omega(n^2)$.

For this reason, and also to have self-reducibility, we restrict attention to *hereditary* classes of graphs. These are lasses for which every (vertex) induced subgraph of a graph in the class is also in the class.

D, Jerrum & Müller (2016) showed that chordal bipartite graphs form the largest hereditary class of bipartite graphs for which the switch chain is ergodic.

When is the switch chain ergodic?

The general question is: Given a graph G, is the switch chain ergodic on G?

Settling the complexity of this question is difficult, since we have no polynomial bound on the length of the sequence of switches connecting two matchings. If we had such a bound, the question would be within the second level of the polynomial hierarchy. In fact, we can only put the problem in PSPACE. However, our best lower bound on the length of the sequence is only $\Omega(n^2)$.

For this reason, and also to have self-reducibility, we restrict attention to *hereditary* classes of graphs. These are lasses for which every (vertex) induced subgraph of a graph in the class is also in the class.

D, Jerrum & Müller (2016) showed that chordal bipartite graphs form the largest hereditary class of bipartite graphs for which the switch chain is ergodic.

Nonbipartite graphs

D & Müller (2017) show that the largest hereditary class for which the switch chain is ergodic is a class of which we call *switchable*. To define this we need the following definitions for a graph G.

A *chord* of a cycle C is an edge $vw \in E \setminus C$. If C is an even cycle, it is an *odd* chord if v and w are joined by and odd-length path on C, otherwise an *even* chord. Note that there are two paths, but both are odd or even. An odd chord divides an even cycle C into two even cycles, sharing an edge. Even and odd chords are not defined for odd cycles.

odd chord

even chord

Nonbipartite graphs

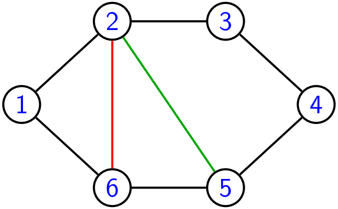
D & Müller (2017) show that the largest hereditary class for which the switch chain is ergodic is a class of which we call *switchable*. To define this we need the following definitions for a graph G.

A *chord* of a cycle C is an edge $vw \in E \setminus C$. If C is an even cycle, it is an *odd* chord if v and w are joined by and odd-length path on C, otherwise an *even* chord. Note that there are two paths, but both are odd or even. An odd chord divides an even cycle C into two even cycles, sharing an edge. Even and odd chords are not defined for odd cycles.

Nonbipartite graphs

D & Müller (2017) show that the largest hereditary class for which the switch chain is ergodic is a class of which we call *switchable*. To define this we need the following definitions for a graph G.

A *chord* of a cycle C is an edge $vw \in E \setminus C$. If C is an even cycle, it is an *odd* chord if v and w are joined by and odd-length path on C, otherwise an *even* chord. Note that there are two paths, but both are odd or even. An odd chord divides an even cycle C into two even cycles, sharing an edge. Even and odd chords are not defined for odd cycles.



odd chord

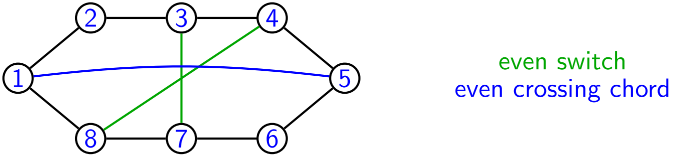
even chord

Odd chordal and switchable graphs

Odd chordal graphs are the hereditary class such that every even cycle has an odd chord. The switch chain is ergodic on this class, but it is not the largest hereditary class.

Two edges of an even cycle have the same *parity* if they are separated by an odd number of edges on the cycle. A *legal switch* is a 4-cycle with two chords and two cycle edges of equal parity.

An *even switch* is a legal switch with even chords. A *crossing chord* of a switch is a chord with end vertices separated by the switch.



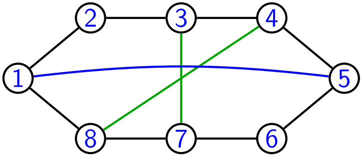
Switchable graphs are the class such that every even cycle has an odd chord or an even switch with a crossing chord. This is the largest hereditary class for which the switch chain is ergodic.

Odd chordal and switchable graphs

Odd chordal graphs are the hereditary class such that every even cycle has an odd chord. The switch chain is ergodic on this class, but it is not the largest hereditary class.

Two edges of an even cycle have the same *parity* if they are separated by an odd number of edges on the cycle. A *legal switch* is a 4-cycle with two chords and two cycle edges of equal parity.

An *even switch* is a legal switch with even chords. A *crossing chord* of a switch is a chord with end vertices separated by the switch.



even switch
even crossing chord

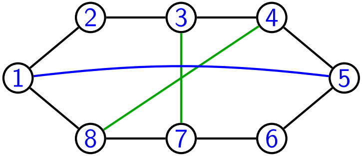
Switchable graphs are the class such that every even cycle has an odd chord or an even switch with a crossing chord. This is the largest hereditary class for which the switch chain is ergodic.

Odd chordal and switchable graphs

Odd chordal graphs are the hereditary class such that every even cycle has an odd chord. The switch chain is ergodic on this class, but it is not the largest hereditary class.

Two edges of an even cycle have the same *parity* if they are separated by an odd number of edges on the cycle. A *legal switch* is a 4-cycle with two chords and two cycle edges of equal parity.

An *even switch* is a legal switch with even chords. A *crossing chord* of a switch is a chord with end vertices separated by the switch.



even switch
even crossing chord

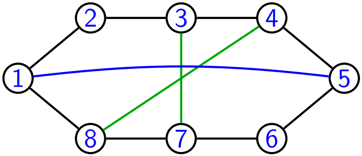
Switchable graphs are the class such that every even cycle has an odd chord or an even switch with a crossing chord. This is the largest hereditary class for which the switch chain is ergodic.

Odd chordal and switchable graphs

Odd chordal graphs are the hereditary class such that every even cycle has an odd chord. The switch chain is ergodic on this class, but it is not the largest hereditary class.

Two edges of an even cycle have the same *parity* if they are separated by an odd number of edges on the cycle. A *legal switch* is a 4-cycle with two chords and two cycle edges of equal parity.

An *even switch* is a legal switch with even chords. A *crossing chord* of a switch is a chord with end vertices separated by the switch.



even switch
even crossing chord

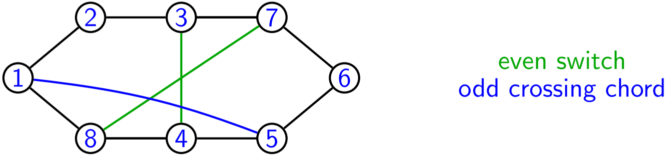
Switchable graphs are the class such that every even cycle has an odd chord or an even switch with a crossing chord. This is the largest hereditary class for which the switch chain is ergodic.

Odd chordal and switchable graphs

Odd chordal graphs are the hereditary class such that every even cycle has an odd chord. The switch chain is ergodic on this class, but it is not the largest hereditary class.

Two edges of an even cycle have the same *parity* if they are separated by an odd number of edges on the cycle. A *legal switch* is a 4-cycle with two chords and two cycle edges of equal parity.

An *even switch* is a legal switch with even chords. A *crossing chord* of a switch is a chord with end vertices separated by the switch.



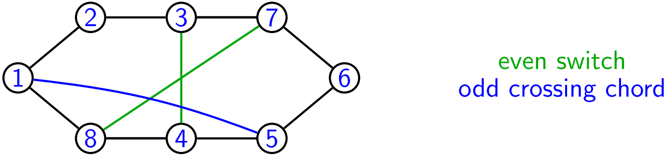
Switchable graphs are the class such that every even cycle has an odd chord or an even switch with a crossing chord. This is the largest hereditary class for which the switch chain is ergodic.

Odd chordal and switchable graphs

Odd chordal graphs are the hereditary class such that every even cycle has an odd chord. The switch chain is ergodic on this class, but it is not the largest hereditary class.

Two edges of an even cycle have the same *parity* if they are separated by an odd number of edges on the cycle. A *legal switch* is a 4-cycle with two chords and two cycle edges of equal parity.

An *even switch* is a legal switch with even chords. A *crossing chord* of a switch is a chord with end vertices separated by the switch.



Switchable graphs are the class such that every even cycle has an odd chord or an even switch with a crossing chord. This is the largest hereditary class for which the switch chain is ergodic.

Monotone graphs

D, Jerrum & Müller (2016) proved rapid mixing of the switch chain for a certain class of chordal bipartite graphs. These graphs have a permutation of their vertex sets so that the 1's in the rows of the biadjacency matrix form intervals with the leftmost and rightmost 1's having nondecreasing order, giving a "staircase" presentation.

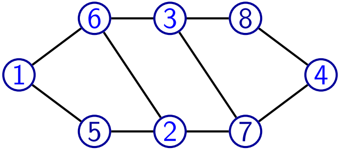
	5	6	7	8
1	1	1	0	0
2	1	1	1	0
3	0	1	1	1
4	0	0	1	1

Diaconis, Graham & Holmes (2001) called these *monotone* graphs, though they originally appeared in Spinrad, Brandstädt & Stewart (1987) in the guise of *bipartite permutation* graphs.

These graphs are not P-stable in general, so the rapid mixing result is essentially different from Jerrum & Sinclair's (1989) algorithm.

Monotone graphs

D, Jerrum & Müller (2016) proved rapid mixing of the switch chain for a certain class of chordal bipartite graphs. These graphs have a permutation of their vertex sets so that the 1's in the rows of the biadjacency matrix form intervals with the leftmost and rightmost 1's having nondecreasing order, giving a "staircase" presentation.



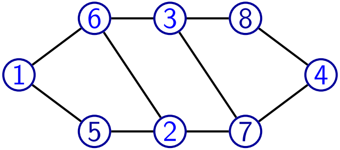
	5	6	7	8
1	1	1	0	0
2	1	1	1	0
3	0	1	1	1
4	0	0	1	1

Diaconis, Graham & Holmes (2001) called these *monotone* graphs, though they originally appeared in Spinrad, Brandstädt & Stewart (1987) in the guise of *bipartite permutation* graphs.

These graphs are not P-stable in general, so the rapid mixing result is essentially different from Jerrum & Sinclair's (1989) algorithm.

Monotone graphs

D, Jerrum & Müller (2016) proved rapid mixing of the switch chain for a certain class of chordal bipartite graphs. These graphs have a permutation of their vertex sets so that the 1's in the rows of the biadjacency matrix form intervals with the leftmost and rightmost 1's having nondecreasing order, giving a "staircase" presentation.



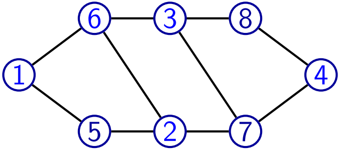
	5	6	7	8
1	1	1	0	0
2	1	1	1	0
3	0	1	1	1
4	0	0	1	1

Diaconis, Graham & Holmes (2001) called these *monotone* graphs, though they originally appeared in Spinrad, Brandstädt & Stewart (1987) in the guise of *bipartite permutation* graphs.

These graphs are not P-stable in general, so the rapid mixing result is essentially different from Jerrum & Sinclair's (1989) algorithm.

Monotone graphs

D, Jerrum & Müller (2016) proved rapid mixing of the switch chain for a certain class of chordal bipartite graphs. These graphs have a permutation of their vertex sets so that the 1's in the rows of the biadjacency matrix form intervals with the leftmost and rightmost 1's having nondecreasing order, giving a "staircase" presentation.



	5	6	7	8
1	1	1	0	0
2	1	1	1	0
3	0	1	1	1
4	0	0	1	1

Diaconis, Graham & Holmes (2001) called these *monotone* graphs, though they originally appeared in Spinrad, Brandstädt & Stewart (1987) in the guise of *bipartite permutation* graphs.

These graphs are not P-stable in general, so the rapid mixing result is essentially different from Jerrum & Sinclair's (1989) algorithm.

Quasiclasses

We considered how far the proof technique used in D, Jerrum & Müller (2016) can be extended to nonbipartite graphs. This led us to the following definitions.

If $G = (V, E)$ is a graph, let $L, R \subseteq V$ be such that $L \cup R = V$ and $L \cap R = \emptyset$, and let $G[L{:}R]$ denote the bipartite graph with vertex bipartition L, R, and edge set $\{vw \in E : v \in L, w \in R\}$.

Let $\mathcal{C}$ be a class of bipartite graphs. Then we will define the class quasi-$\mathcal{C}$ as follows: G is in quasi-$\mathcal{C}$ if $G[L{:}R] \in \mathcal{C}$ for *all* bipartitions L, R of V. This seems a demanding definition, but in fact quasi-$\mathcal{C}$ is larger than $\mathcal{C}$ for most cases of interest. If $\mathcal{C}$ is hereditary and closed under disjoint union, then so is quasi-$\mathcal{C}$, and $\mathcal{C} \subseteq$ quasi-$\mathcal{C}$.

The motivation for this definition is that techniques for bipartite graphs can often be lifted to the corresponding quasiclass.

Quasiclasses

We considered how far the proof technique used in D, Jerrum & Müller (2016) can be extended to nonbipartite graphs. This led us to the following definitions.

If $G = (V, E)$ is a graph, let $L, R \subseteq V$ be such that $L \cup R = V$ and $L \cap R = \emptyset$, and let $G[L{:}R]$ denote the bipartite graph with vertex bipartition L, R, and edge set $\{vw \in E : v \in L, w \in R\}$.

Let $\mathcal{C}$ be a class of bipartite graphs. Then we will define the class quasi-$\mathcal{C}$ as follows: G is in quasi-$\mathcal{C}$ if $G[L{:}R] \in \mathcal{C}$ for *all* bipartitions L, R of V. This seems a demanding definition, but in fact quasi-$\mathcal{C}$ is larger than $\mathcal{C}$ for most cases of interest. If $\mathcal{C}$ is hereditary and closed under disjoint union, then so is quasi-$\mathcal{C}$, and $\mathcal{C} \subseteq$ quasi-$\mathcal{C}$.

The motivation for this definition is that techniques for bipartite graphs can often be lifted to the corresponding quasiclass.

Quasiclasses

We considered how far the proof technique used in D, Jerrum & Müller (2016) can be extended to nonbipartite graphs. This led us to the following definitions.

If $G = (V, E)$ is a graph, let $L, R \subseteq V$ be such that $L \cup R = V$ and $L \cap R = \emptyset$, and let $G[L{:}R]$ denote the bipartite graph with vertex bipartition L, R, and edge set $\{vw \in E : v \in L, w \in R\}$.

Let $\mathcal{C}$ be a class of bipartite graphs. Then we will define the class quasi-$\mathcal{C}$ as follows: G is in quasi-$\mathcal{C}$ if $G[L{:}R] \in \mathcal{C}$ for *all* bipartitions L, R of V. This seems a demanding definition, but in fact quasi-$\mathcal{C}$ is larger than $\mathcal{C}$ for most cases of interest. If $\mathcal{C}$ is hereditary and closed under disjoint union, then so is quasi-$\mathcal{C}$, and $\mathcal{C} \subseteq$ quasi-$\mathcal{C}$.

The motivation for this definition is that techniques for bipartite graphs can often be lifted to the corresponding quasiclass.

Quasiclasses

We considered how far the proof technique used in D, Jerrum & Müller (2016) can be extended to nonbipartite graphs. This led us to the following definitions.

If $G = (V, E)$ is a graph, let $L, R \subseteq V$ be such that $L \cup R = V$ and $L \cap R = \emptyset$, and let $G[L{:}R]$ denote the bipartite graph with vertex bipartition L, R, and edge set $\{vw \in E : v \in L, w \in R\}$.

Let $\mathcal{C}$ be a class of bipartite graphs. Then we will define the class quasi-$\mathcal{C}$ as follows: G is in quasi-$\mathcal{C}$ if $G[L{:}R] \in \mathcal{C}$ for *all* bipartitions L, R of V. This seems a demanding definition, but in fact quasi-$\mathcal{C}$ is larger than $\mathcal{C}$ for most cases of interest. If $\mathcal{C}$ is hereditary and closed under disjoint union, then so is quasi-$\mathcal{C}$, and $\mathcal{C} \subseteq$ quasi-$\mathcal{C}$.

The motivation for this definition is that techniques for bipartite graphs can often be lifted to the corresponding quasiclass.

Quasiclasses

We considered how far the proof technique used in D, Jerrum & Müller (2016) can be extended to nonbipartite graphs. This led us to the following definitions.

If $G = (V, E)$ is a graph, let $L, R \subseteq V$ be such that $L \cup R = V$ and $L \cap R = \emptyset$, and let $G[L{:}R]$ denote the bipartite graph with vertex bipartition L, R, and edge set $\{vw \in E : v \in L, w \in R\}$.

Let $\mathcal{C}$ be a class of bipartite graphs. Then we will define the class quasi-$\mathcal{C}$ as follows: G is in quasi-$\mathcal{C}$ if $G[L{:}R] \in \mathcal{C}$ for *all* bipartitions L, R of V. This seems a demanding definition, but in fact quasi-$\mathcal{C}$ is larger than $\mathcal{C}$ for most cases of interest. If $\mathcal{C}$ is hereditary and closed under disjoint union, then so is quasi-$\mathcal{C}$, and $\mathcal{C} \subseteq$ quasi-$\mathcal{C}$.

The motivation for this definition is that techniques for bipartite graphs can often be lifted to the corresponding quasiclass.

Quasimonotone graphs

If $\mathcal{C}$ is the class of monotone graphs, quasi-$\mathcal{C}$ is the class of *quasimonotone* graphs. It is easy to lift the canonical paths analysis of D, Jerrum & Müller to prove rapid mixing for this class.

The class includes monotone graphs, but also *unit interval* graphs. These are intersection graphs of sets of intervals $v_i = [x_i, x_i + 1]$ ($i \in [n]$)on the real line. Thus $G = (V, E)$, with $V = \{v_i : i \in [n]\}$ and $v_i v_j \in E$ if and only if $i \neq j$ and $v_i \cap v_j \neq \emptyset$.

However, the class is larger than the union of these two classes. For example, the graph below is quasimonotone.

Quasimonotone graphs

If $\mathcal{C}$ is the class of monotone graphs, quasi-$\mathcal{C}$ is the class of *quasimonotone* graphs. It is easy to lift the canonical paths analysis of D, Jerrum & Müller to prove rapid mixing for this class.

The class includes monotone graphs, but also *unit interval* graphs. These are intersection graphs of sets of intervals $v_i = [x_i, x_i + 1]$ ($i \in [n]$)on the real line. Thus $G = (V, E)$, with $V = \{v_i : i \in [n]\}$ and $v_i v_j \in E$ if and only if $i \neq j$ and $v_i \cap v_j \neq \emptyset$.

However, the class is larger than the union of these two classes. For example, the graph below is quasimonotone.

Quasimonotone graphs

If $\mathcal{C}$ is the class of monotone graphs, quasi-$\mathcal{C}$ is the class of *quasimonotone* graphs. It is easy to lift the canonical paths analysis of D, Jerrum & Müller to prove rapid mixing for this class.

The class includes monotone graphs, but also *unit interval* graphs. These are intersection graphs of sets of intervals $v_i = [x_i, x_i + 1]$ ($i \in [n]$)on the real line. Thus $G = (V, E)$, with $V = \{v_i : i \in [n]\}$ and $v_i v_j \in E$ if and only if $i \neq j$ and $v_i \cap v_j \neq \emptyset$.

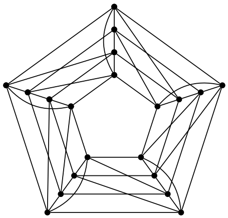
However, the class is larger than the union of these two classes. For example, the graph below is quasimonotone.

Quasimonotone graphs

If $\mathcal{C}$ is the class of monotone graphs, quasi-$\mathcal{C}$ is the class of *quasimonotone* graphs. It is easy to lift the canonical paths analysis of D, Jerrum & Müller to prove rapid mixing for this class.

The class includes monotone graphs, but also *unit interval* graphs. These are intersection graphs of sets of intervals $v_i = [x_i, x_i + 1]$ ($i \in [n]$)on the real line. Thus $G = (V, E)$, with $V = \{v_i : i \in [n]\}$ and $v_i v_j \in E$ if and only if $i \neq j$ and $v_i \cap v_j \neq \emptyset$.

However, the class is larger than the union of these two classes. For example, the graph below is quasimonotone.



Recognition

An important question now arises. For a given graph G, can we decide efficiently whether it is in some specified class?

Quasi-chordal bipartite graphs are precisely the class of odd chordal graphs defined above. So, since monotone graphs are chordal bipartite, quasimonotone graphs are odd chordal.

Chordal bipartite graphs have linear time recognition, but this implies nothing for the quasiclass. Currently we do not know how recognise odd chordal graphs in polynomial time. If we could recognise odd chordal graphs efficiently, we could recognise quasimonotone graphs efficiently, simply by using a small set of forbidden subgraphs. Currently we cannot do this.

Nevertheless, we can recognise quasimonotone graphs in polynomial time. The algorithm is rather complicated, so we cannot describe it here.

Recognition

An important question now arises. For a given graph G, can we decide efficiently whether it is in some specified class?

Quasi-chordal bipartite graphs are precisely the class of odd chordal graphs defined above. So, since monotone graphs are chordal bipartite, quasimonotone graphs are odd chordal.

Chordal bipartite graphs have linear time recognition, but this implies nothing for the quasiclass. Currently we do not know how recognise odd chordal graphs in polynomial time. If we could recognise odd chordal graphs efficiently, we could recognise quasimonotone graphs efficiently, simply by using a small set of forbidden subgraphs. Currently we cannot do this.

Nevertheless, we can recognise quasimonotone graphs in polynomial time. The algorithm is rather complicated, so we cannot describe it here.

Recognition

An important question now arises. For a given graph G, can we decide efficiently whether it is in some specified class?

Quasi-chordal bipartite graphs are precisely the class of odd chordal graphs defined above. So, since monotone graphs are chordal bipartite, quasimonotone graphs are odd chordal.

Chordal bipartite graphs have linear time recognition, but this implies nothing for the quasiclass. Currently we do not know how recognise odd chordal graphs in polynomial time. If we could recognise odd chordal graphs efficiently, we could recognise quasimonotone graphs efficiently, simply by using a small set of forbidden subgraphs. Currently we cannot do this.

Nevertheless, we can recognise quasimonotone graphs in polynomial time. The algorithm is rather complicated, so we cannot describe it here.

Recognition

An important question now arises. For a given graph G, can we decide efficiently whether it is in some specified class?

Quasi-chordal bipartite graphs are precisely the class of odd chordal graphs defined above. So, since monotone graphs are chordal bipartite, quasimonotone graphs are odd chordal.

Chordal bipartite graphs have linear time recognition, but this implies nothing for the quasiclass. Currently we do not know how recognise odd chordal graphs in polynomial time. If we could recognise odd chordal graphs efficiently, we could recognise quasimonotone graphs efficiently, simply by using a small set of forbidden subgraphs. Currently we cannot do this.

Nevertheless, we can recognise quasimonotone graphs in polynomial time. The algorithm is rather complicated, so we cannot describe it here.

Recognition

An important question now arises. For a given graph G, can we decide efficiently whether it is in some specified class?

Quasi-chordal bipartite graphs are precisely the class of odd chordal graphs defined above. So, since monotone graphs are chordal bipartite, quasimonotone graphs are odd chordal.

Chordal bipartite graphs have linear time recognition, but this implies nothing for the quasiclass. Currently we do not know how recognise odd chordal graphs in polynomial time. If we could recognise odd chordal graphs efficiently, we could recognise quasimonotone graphs efficiently, simply by using a small set of forbidden subgraphs. Currently we cannot do this.

Nevertheless, we can recognise quasimonotone graphs in polynomial time. The algorithm is rather complicated, so we cannot describe it here.

Thank you!