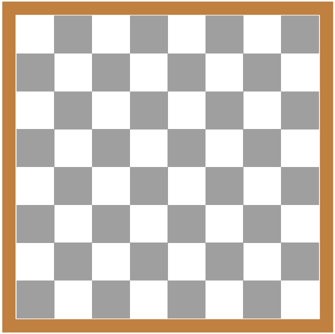
Counting perfect matchings in planar graphs

Ashley Montanaro

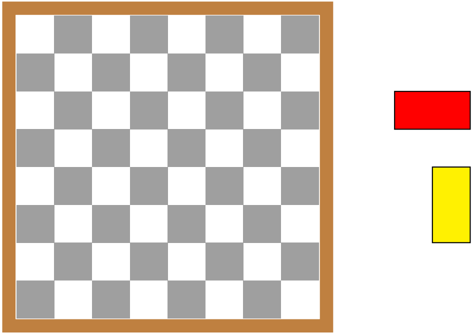
Department of Computer Science,
University of Bristol

December 11, 2009

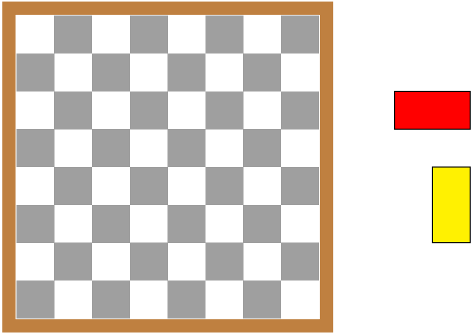
Chess boards and dominoes



Chess boards and dominoes

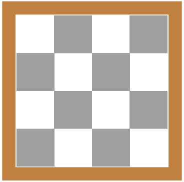


Chess boards and dominoes

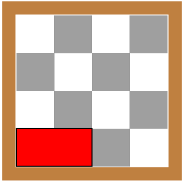


How many ways are there to cover the chess board with dominoes?

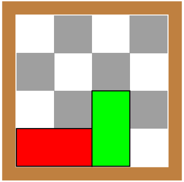
Chess boards and dominoes



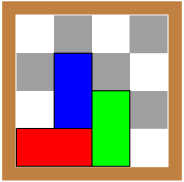
Chess boards and dominoes



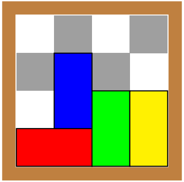
Chess boards and dominoes



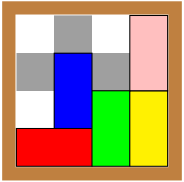
Chess boards and dominoes



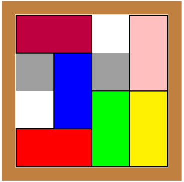
Chess boards and dominoes



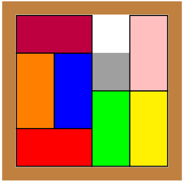
Chess boards and dominoes



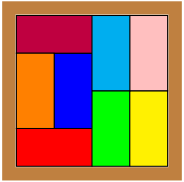
Chess boards and dominoes



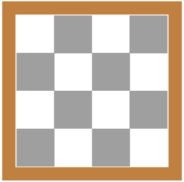
Chess boards and dominoes



Chess boards and dominoes

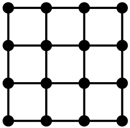


Chess boards and dominoes



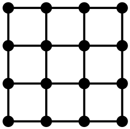
This is an instance of a more general problem. We turn the board into a graph by replacing the squares with vertices, putting an edge between adjacent squares.

Chess boards and dominoes



This is an instance of a more general problem. We turn the board into a graph by replacing the squares with vertices, putting an edge between adjacent squares.

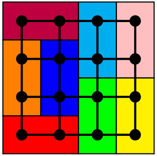
Chess boards and dominoes



This is an instance of a more general problem. We turn the board into a graph by replacing the squares with vertices, putting an edge between adjacent squares.

Then putting dominoes on the board corresponds to selecting edges from the graph such that no two edges share a vertex.

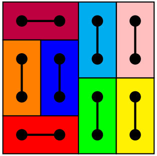
Chess boards and dominoes



This is an instance of a more general problem. We turn the board into a graph by replacing the squares with vertices, putting an edge between adjacent squares.

Then putting dominoes on the board corresponds to selecting edges from the graph such that no two edges share a vertex.

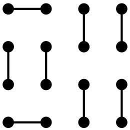
Chess boards and dominoes



This is an instance of a more general problem. We turn the board into a graph by replacing the squares with vertices, putting an edge between adjacent squares.

Then putting dominoes on the board corresponds to selecting edges from the graph such that no two edges share a vertex.

Chess boards and dominoes



This is an instance of a more general problem. We turn the board into a graph by replacing the squares with vertices, putting an edge between adjacent squares.

Then putting dominoes on the board corresponds to selecting edges from the graph such that no two edges share a vertex.

A covering of the board is known as a perfect matching.

Matchings

More formally, we have:

Definition

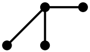
Given a graph $G = (V, E)$, a matching M in G is a set of pairwise non-adjacent edges. M is said to be perfect if every vertex of G is included in M.

Matchings

More formally, we have:

Definition

Given a graph $G = (V, E)$, a matching M in G is a set of pairwise non-adjacent edges. M is said to be perfect if every vertex of G is included in M.

- Of course, G can only have a perfect matching if $|V|$ is even.
- Not every graph with an even number of vertices has a perfect matching, e.g. consider 
- The number of perfect matchings can be exponential in the number of vertices.

The complexity of perfect matchings

There are many questions we might want to ask about perfect matchings:

1. Can we find one efficiently?

The complexity of perfect matchings

There are many questions we might want to ask about perfect matchings:

1. Can we find one efficiently?
   Yes! A (version of a) famous algorithm of Jack Edmonds finds a perfect matching, if it exists, in $O(\sqrt{|V|}|E|)$ time.

The complexity of perfect matchings

There are many questions we might want to ask about perfect matchings:

1. Can we find one efficiently?
   Yes! A (version of a) famous algorithm of Jack Edmonds finds a perfect matching, if it exists, in $O(\sqrt{|V|}|E|)$ time.

2. Can we count the number of perfect matchings efficiently?

The complexity of perfect matchings

There are many questions we might want to ask about perfect matchings:

1. Can we find one efficiently?
   Yes! A (version of a) famous algorithm of Jack Edmonds finds a perfect matching, if it exists, in $O(\sqrt{|V|}|E|)$ time.

2. Can we count the number of perfect matchings efficiently?
   No! (Probably.) Counting the number of perfect matchings in a general graph has been shown to be #P-complete (much harder than NP-complete).

The complexity of perfect matchings

There are many questions we might want to ask about perfect matchings:

1. Can we find one efficiently?
   Yes! A (version of a) famous algorithm of Jack Edmonds finds a perfect matching, if it exists, in $O(\sqrt{|V|}|E|)$ time.

2. Can we count the number of perfect matchings efficiently?
   No! (Probably.) Counting the number of perfect matchings in a general graph has been shown to be #P-complete (much harder than NP-complete).

3. So are there any special cases we can deal with?

The complexity of perfect matchings

There are many questions we might want to ask about perfect matchings:

1. Can we find one efficiently?
   Yes! A (version of a) famous algorithm of Jack Edmonds finds a perfect matching, if it exists, in $O(\sqrt{|V|}|E|)$ time.

2. Can we count the number of perfect matchings efficiently?
   No! (Probably.) Counting the number of perfect matchings in a general graph has been shown to be #P-complete (much harder than NP-complete).

3. So are there any special cases we can deal with?
   Yes! This lecture: an efficient algorithm for counting the number of perfect matchings in a planar graph.

Planar graphs

Definition

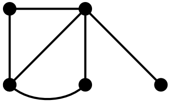
A graph is said to be planar if it can be drawn in the 2D plane in such a way that its edges intersect only at its vertices.

Planar graphs

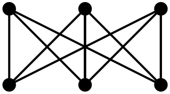
Definition

A graph is said to be planar if it can be drawn in the 2D plane in such a way that its edges intersect only at its vertices.

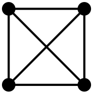
For example:



Planar



Not planar



Planar

Many graphs that occur in real-world applications are planar.

Counting perfect matchings in planar graphs

We start by making the problem more mathematically tractable.

- Let $G = (V, E)$ be a graph on n vertices, where n is even. Define $A_{ij} = 1 \Leftrightarrow (i, j) \in E$ (A is the adjacency matrix of G).

Counting perfect matchings in planar graphs

We start by making the problem more mathematically tractable.

- Let $G = (V, E)$ be a graph on n vertices, where n is even. Define $A_{ij} = 1 \Leftrightarrow (i, j) \in E$ (A is the adjacency matrix of G).
- Define $PM(n)$ to be the set of partitions of n elements into pairs. (e.g. $PM(4) = \{[\{1, 2\}, \{3, 4\}], [\{1, 3\}, \{2, 4\}], [\{1, 4\}, \{2, 3\}]\}$)

Counting perfect matchings in planar graphs

We start by making the problem more mathematically tractable.

- Let $G = (V, E)$ be a graph on n vertices, where n is even. Define $A_{ij} = 1 \Leftrightarrow (i, j) \in E$ (A is the adjacency matrix of G).
- Define $PM(n)$ to be the set of partitions of n elements into pairs. (e.g. $PM(4) = \{[\{1, 2\}, \{3, 4\}], [\{1, 3\}, \{2, 4\}], [\{1, 4\}, \{2, 3\}]\}$)
- Each element of $PM(n)$ can be thought of as a permutation of the integers between 1 and n, and gives a potential perfect matching of G.

Counting perfect matchings in planar graphs

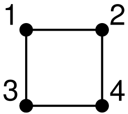
We start by making the problem more mathematically tractable.

- Let $G = (V, E)$ be a graph on n vertices, where n is even. Define $A_{ij} = 1 \Leftrightarrow (i,j) \in E$ (A is the adjacency matrix of G).
- Define $PM(n)$ to be the set of partitions of n elements into pairs. (e.g. $PM(4) = \{[\{1,2\},\{3,4\}],[\{1,3\},\{2,4\}],[\{1,4\},\{2,3\}]\}$)
- Each element of $PM(n)$ can be thought of as a permutation of the integers between 1 and n, and gives a potential perfect matching of G.
- So we want to compute the following quantity:

$$\text{PerfMatch}(G) = \sum_{M \in PM(n)} \prod_{(i,j) \in M} A_{ij}.$$

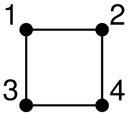
A simple example

$G =$



$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

A simple example

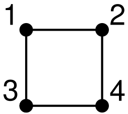
$G =$ 

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$$PM(4) = \{[\{1,2\},\{3,4\}],\ [\{1,3\},\{2,4\}],\ [\{1,4\},\{2,3\}]\}.$$

$$\text{PerfMatch}(G) \;=\; \sum_{M \in PM(n)} \prod_{(i,j) \in M} A_{ij}$$

A simple example

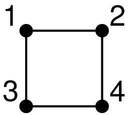
$G =$ 

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$$PM(4) = \{[\{1,2\},\{3,4\}],\ [\{1,3\},\{2,4\}],\ [\{1,4\},\{2,3\}]\}.$$

$$\begin{aligned} \text{PerfMatch}(G) &= \sum_{M \in PM(n)} \prod_{(i,j) \in M} A_{ij} \\ &= A_{12}A_{34} + A_{13}A_{24} + A_{14}A_{23} \end{aligned}$$

A simple example

$G =$ 

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

$$PM(4) = \{[\{1,2\},\{3,4\}],\ [\{1,3\},\{2,4\}],\ [\{1,4\},\{2,3\}]\}.$$

$$\begin{aligned} \text{PerfMatch}(G) &= \sum_{M \in PM(n)} \prod_{(i,j) \in M} A_{ij} \\ &= A_{12}A_{34} + A_{13}A_{24} + A_{14}A_{23} \\ &= 2. \end{aligned}$$

Pfaffians

We will try to compute PerfMatch(G) using Pfaffians ("perfect matchings with signs").

Definition

The Pfaffian Pf(A) of an $n \times n$ matrix A is defined as

$$\mathrm{Pf}(A) = \sum_{M \in PM(n)} \mathrm{sgn}(M) \prod_{(i,j) \in M} A_{ij},$$

where sgn(M) is the sign of M as a permutation of n elements.

Pfaffians

We will try to compute PerfMatch(G) using Pfaffians ("perfect matchings with signs").

Definition

The Pfaffian Pf(A) of an $n \times n$ matrix A is defined as

$$\mathrm{Pf}(A) = \sum_{M \in PM(n)} \mathrm{sgn}(M) \prod_{(i,j) \in M} A_{ij},$$

where sgn(M) is the sign of M as a permutation of n elements.

Recall that the sign of a permutation σ is 1 if σ contains an even number of transpositions (exchanges of 2 elements), and -1 if σ contains an odd number of transpositions.

For example, $\mathrm{sgn}((2, 1, 4, 3)) = 1$, $\mathrm{sgn}((3, 2, 1, 4)) = -1$.

Why think about Pfaffians?

Theorem (Muir, 1882)

Let A be a skew-symmetric matrix ($A_{ij} = -A_{ji}$). Then $\mathrm{Pf}(A)^2 = \det(A)$, where $\det(A)$ is the determinant of A.

Why think about Pfaffians?

Theorem (Muir, 1882)

Let A be a skew-symmetric matrix ($A_{ij} = -A_{ji}$). Then $\mathrm{Pf}(A)^2 = \det(A)$, where $\det(A)$ is the determinant of A.

- The determinant of an $n \times n$ matrix can be computed in $O(n^3)$ operations (or fewer).

Why think about Pfaffians?

Theorem (Muir, 1882)

Let A be a skew-symmetric matrix ($A_{ij} = -A_{ji}$). Then $\mathrm{Pf}(A)^2 = \det(A)$, where $\det(A)$ is the determinant of A.

- The determinant of an $n \times n$ matrix can be computed in $O(n^3)$ operations (or fewer).
- So the Pfaffian of a skew-symmetric matrix can be computed efficiently, up to a sign (despite the fact that it is a sum over exponentially many things).

Why think about Pfaffians?

Theorem (Muir, 1882)

Let A be a skew-symmetric matrix ($A_{ij} = -A_{ji}$). Then $\mathrm{Pf}(A)^2 = \det(A)$, where $\det(A)$ is the determinant of A.

- The determinant of an $n \times n$ matrix can be computed in $O(n^3)$ operations (or fewer).
- So the Pfaffian of a skew-symmetric matrix can be computed efficiently, up to a sign (despite the fact that it is a sum over exponentially many things).
- So, if we can find some skew-symmetric matrix A such that $\mathrm{Pf}(A) = \pm\mathrm{PerfMatch}(G)$, we can compute $\mathrm{PerfMatch}(G)$ efficiently!

Turning this idea into an algorithm

1. We produce a directed graph G' from G by orienting each edge of G in some direction.

Turning this idea into an algorithm

1. We produce a directed graph G' from G by orienting each edge of G in some direction.
2. This gives us a skew-symmetric adjacency matrix A' ($A'_{ij} = 1$ if there is an edge $i \to j$; $A'_{ij} = -1$ if there is an edge $j \to i$).

Turning this idea into an algorithm

1. We produce a directed graph G' from G by orienting each edge of G in some direction.
2. This gives us a skew-symmetric adjacency matrix A' ($A'_{ij} = 1$ if there is an edge $i \to j$; $A'_{ij} = -1$ if there is an edge $j \to i$).
3. We want $\mathrm{Pf}(A') = \pm\mathrm{PerfMatch}(G)$, i.e. all the terms in the Pfaffian to have the same sign.

Turning this idea into an algorithm

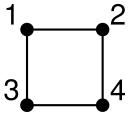
1. We produce a directed graph G' from G by orienting each edge of G in some direction.
2. This gives us a skew-symmetric adjacency matrix A' ($A'_{ij} = 1$ if there is an edge $i \to j$; $A'_{ij} = -1$ if there is an edge $j \to i$).
3. We want $\mathrm{Pf}(A') = \pm\mathrm{PerfMatch}(G)$, i.e. all the terms in the Pfaffian to have the same sign.

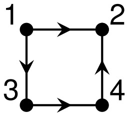
This will be the case when, for all $M \in PM(n)$ such that M is a perfect matching of G,

$$\prod_{(i,j)\in M} A'_{ij} = \mathrm{sgn}(M) \cdot s,$$

for some $s = \pm 1$, which is the same for all M. If this holds, G' is said to be a Pfaffian orientation of G.

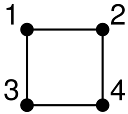
Example

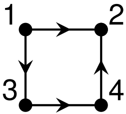
$G =$ 

$G' =$ 

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad A' = \begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \end{pmatrix}$$

Example

$G =$ 

$G' =$ 

$$A = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix} \quad A' = \begin{pmatrix} 0 & 1 & 1 & 0 \\ -1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \end{pmatrix}$$

- The two perfect matchings of G are $\{(1,2),(3,4)\}$ and $\{(1,3),(2,4)\}$.
- $A'_{12}A'_{34} = 1 = \text{sgn}((1,2,3,4))$; $A'_{13}A'_{24} = -1 = \text{sgn}((1,3,2,4))$.
- Therefore G' is a Pfaffian orientation of G.
- It can be verified that $\det(A') = 4 = \text{Pf}(A')^2$.

Finding Pfaffian orientations

Theorem (Kasteleyn, 1963)

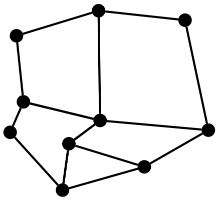
Every planar graph has a Pfaffian orientation. Such an orientation can be found in polynomial time.

Finding Pfaffian orientations

Theorem (Kasteleyn, 1963)

Every planar graph has a Pfaffian orientation. Such an orientation can be found in polynomial time.

The algorithm to do this uses an interpretation of planar graphs as a mesh of faces. For example:

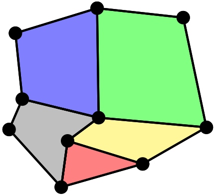


Finding Pfaffian orientations

Theorem (Kasteleyn, 1963)

Every planar graph has a Pfaffian orientation. Such an orientation can be found in polynomial time.

The algorithm to do this uses an interpretation of planar graphs as a mesh of faces. For example:



Each coloured component above is called a face of G.

Faces and Pfaffian orientations

The algorithm is based on the following result.

Theorem (Kasteleyn, 1963)

Let G be a planar graph. Then (a) G can be oriented efficiently so that each face has an odd number of lines oriented clockwise, and (b) this is a Pfaffian orientation of G.

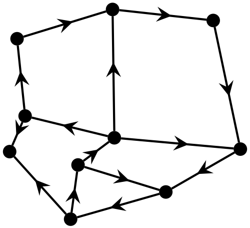
Faces and Pfaffian orientations

The algorithm is based on the following result.

Theorem (Kasteleyn, 1963)

Let G be a planar graph. Then (a) G can be oriented efficiently so that each face has an odd number of lines oriented clockwise, and (b) this is a Pfaffian orientation of G.

An example of such an orientation:



Proving part (b) of Kasteleyn's theorem

Part (b) is based on the following lemma (proof omitted).

Lemma

Let G be a graph and G' be an orientation of G. Then G' is a Pfaffian orientation if every nice cycle in G is oddly oriented in G'.

Proving part (b) of Kasteleyn's theorem

Part (b) is based on the following lemma (proof omitted).

Lemma

Let G be a graph and G' be an orientation of G. Then G' is a Pfaffian orientation if every nice cycle in G is oddly oriented in G'.

- A nice cycle C is an even cycle such that, if C were removed, G would still have a perfect matching.
- C is oddly oriented if there are an odd number of edges in C going in each direction.

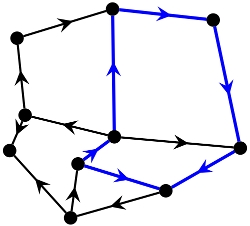
Proving part (b) of Kasteleyn's theorem

Part (b) is based on the following lemma (proof omitted).

Lemma

Let G be a graph and G' be an orientation of G. Then G' is a Pfaffian orientation if every nice cycle in G is oddly oriented in G'.

- A nice cycle C is an even cycle such that, if C were removed, G would still have a perfect matching.
- C is oddly oriented if there are an odd number of edges in C going in each direction.



Proving part (b) of Kasteleyn's theorem

By the previous lemma, it suffices to show the following result.

Lemma

Let G be a planar graph. If G is oriented so that each face has an odd number of lines oriented clockwise, then every nice cycle in G is oddly oriented.

Proving part (b) of Kasteleyn's theorem

By the previous lemma, it suffices to show the following result.

Lemma

Let G be a planar graph. If G is oriented so that each face has an odd number of lines oriented clockwise, then every nice cycle in G is oddly oriented.

We will need the following version of Euler's formula:

Euler's formula

For any cycle C, $e = v + f - 1$, where e is the number of edges inside C, v is the number of vertices inside C, and f is the number of faces inside C.

(Proof: exercise.)

Proof of Lemma

- Let C be a nice cycle, let c_i be the number of clockwise lines on the boundary of face i in C, and c be the number of clockwise lines on C.

Proof of Lemma

- Let C be a nice cycle, let c_i be the number of clockwise lines on the boundary of face i in C, and c be the number of clockwise lines on C.
- We oriented each face to have an odd number of clockwise lines, so $c_i \equiv 1 \mod 2$, so $f \equiv \sum_{i=1}^{f} c_i \mod 2$.

Proof of Lemma

- Let C be a nice cycle, let c_i be the number of clockwise lines on the boundary of face i in C, and c be the number of clockwise lines on C.
- We oriented each face to have an odd number of clockwise lines, so $c_i \equiv 1 \mod 2$, so $f \equiv \sum_{i=1}^{f} c_i \mod 2$.
- But also $\sum_{i=1}^{f} c_i = c + e$ (each interior line is counted as clockwise once).

Proof of Lemma

- Let C be a nice cycle, let c_i be the number of clockwise lines on the boundary of face i in C, and c be the number of clockwise lines on C.
- We oriented each face to have an odd number of clockwise lines, so $c_i \equiv 1 \mod 2$, so $f \equiv \sum_{i=1}^{f} c_i \mod 2$.
- But also $\sum_{i=1}^{f} c_i = c + e$ (each interior line is counted as clockwise once).
- So $f \equiv c + (v + f - 1) \mod 2$, so $c \equiv (v - 1) \mod 2$.

Proof of Lemma

- Let C be a nice cycle, let c_i be the number of clockwise lines on the boundary of face i in C, and c be the number of clockwise lines on C.
- We oriented each face to have an odd number of clockwise lines, so $c_i \equiv 1 \mod 2$, so $f \equiv \sum_{i=1}^{f} c_i \mod 2$.
- But also $\sum_{i=1}^{f} c_i = c + e$ (each interior line is counted as clockwise once).
- So $f \equiv c + (v + f - 1) \mod 2$, so $c \equiv (v - 1) \mod 2$.
- But $v \equiv 0 \mod 2$, as C is a nice cycle.

Proof of Lemma

- Let C be a nice cycle, let c_i be the number of clockwise lines on the boundary of face i in C, and c be the number of clockwise lines on C.
- We oriented each face to have an odd number of clockwise lines, so $c_i \equiv 1 \mod 2$, so $f \equiv \sum_{i=1}^{f} c_i \mod 2$.
- But also $\sum_{i=1}^{f} c_i = c + e$ (each interior line is counted as clockwise once).
- So $f \equiv c + (v + f - 1) \mod 2$, so $c \equiv (v - 1) \mod 2$.
- But $v \equiv 0 \mod 2$, as C is a nice cycle.
- So C, and hence every nice cycle, is oddly oriented.

Recap

We have shown that:

Recap

We have shown that:

1. To count the number of perfect matchings in a graph G, it suffices to find a Pfaffian orientation of G.

Recap

We have shown that:

1. To count the number of perfect matchings in a graph G, it suffices to find a Pfaffian orientation of G.
2. To find a Pfaffian orientation of a planar graph G, it suffices to orient G so that each face has an odd number of lines oriented clockwise.

Recap

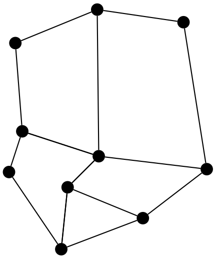
We have shown that:

1. To count the number of perfect matchings in a graph G, it suffices to find a Pfaffian orientation of G.

2. To find a Pfaffian orientation of a planar graph G, it suffices to orient G so that each face has an odd number of lines oriented clockwise.

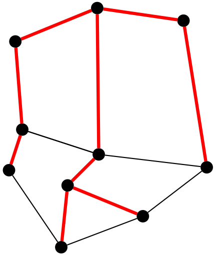
Remaining step: An efficient algorithm to orient a planar graph so that each face has an odd number of lines oriented clockwise.

Finding a Pfaffian orientation



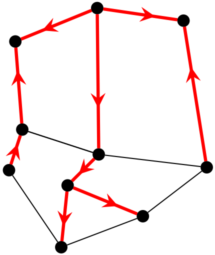
Finding a Pfaffian orientation

1. Find a spanning tree for G. Call this tree T_1.



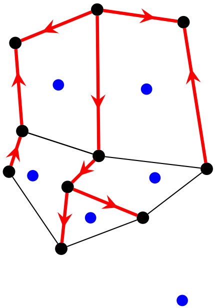
Finding a Pfaffian orientation

1. Find a spanning tree for G. Call this tree T_1.
2. Orient the edges contained in T_1 arbitrarily.



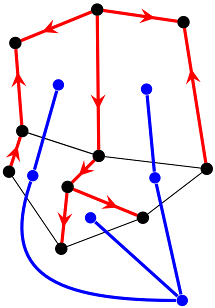
Finding a Pfaffian orientation

1. Find a spanning tree for G. Call this tree T_1.
2. Orient the edges contained in T_1 arbitrarily.
3. Construct a second tree T_2, whose vertices are the faces of G. Put an edge between faces that share an edge that's not in T_1.



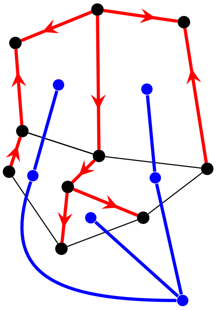
Finding a Pfaffian orientation

1. Find a spanning tree for G. Call this tree T_1.
2. Orient the edges contained in T_1 arbitrarily.
3. Construct a second tree T_2, whose vertices are the faces of G. Put an edge between faces that share an edge that's not in T_1.



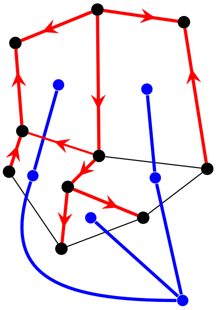
Finding a Pfaffian orientation

1. Find a spanning tree for G. Call this tree T_1.
2. Orient the edges contained in T_1 arbitrarily.
3. Construct a second tree T_2, whose vertices are the faces of G. Put an edge between faces that share an edge that's not in T_1.
4. Starting with the leaves of T_2, orient these edges of G such that each face has an odd number of lines oriented clockwise.



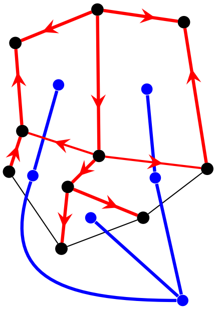
Finding a Pfaffian orientation

1. Find a spanning tree for G. Call this tree T_1.
2. Orient the edges contained in T_1 arbitrarily.
3. Construct a second tree T_2, whose vertices are the faces of G. Put an edge between faces that share an edge that's not in T_1.
4. Starting with the leaves of T_2, orient these edges of G such that each face has an odd number of lines oriented clockwise.



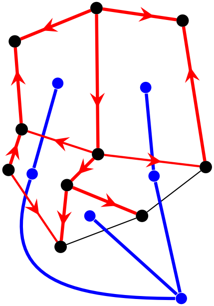
Finding a Pfaffian orientation

1. Find a spanning tree for G. Call this tree T_1.
2. Orient the edges contained in T_1 arbitrarily.
3. Construct a second tree T_2, whose vertices are the faces of G. Put an edge between faces that share an edge that's not in T_1.
4. Starting with the leaves of T_2, orient these edges of G such that each face has an odd number of lines oriented clockwise.



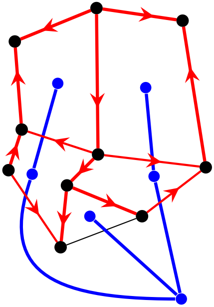
Finding a Pfaffian orientation

1. Find a spanning tree for G. Call this tree T_1.
2. Orient the edges contained in T_1 arbitrarily.
3. Construct a second tree T_2, whose vertices are the faces of G. Put an edge between faces that share an edge that's not in T_1.
4. Starting with the leaves of T_2, orient these edges of G such that each face has an odd number of lines oriented clockwise.



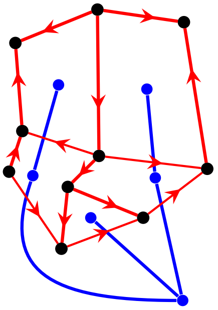
Finding a Pfaffian orientation

1. Find a spanning tree for G. Call this tree T_1.
2. Orient the edges contained in T_1 arbitrarily.
3. Construct a second tree T_2, whose vertices are the faces of G. Put an edge between faces that share an edge that's not in T_1.
4. Starting with the leaves of T_2, orient these edges of G such that each face has an odd number of lines oriented clockwise.



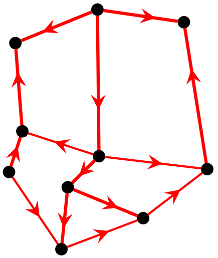
Finding a Pfaffian orientation

1. Find a spanning tree for G. Call this tree T_1.
2. Orient the edges contained in T_1 arbitrarily.
3. Construct a second tree T_2, whose vertices are the faces of G. Put an edge between faces that share an edge that's not in T_1.
4. Starting with the leaves of T_2, orient these edges of G such that each face has an odd number of lines oriented clockwise.



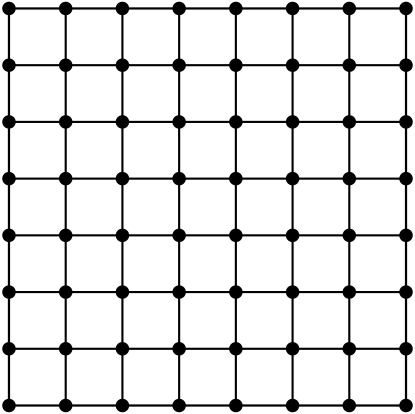
Finding a Pfaffian orientation

1. Find a spanning tree for G. Call this tree T_1.
2. Orient the edges contained in T_1 arbitrarily.
3. Construct a second tree T_2, whose vertices are the faces of G. Put an edge between faces that share an edge that's not in T_1.
4. Starting with the leaves of T_2, orient these edges of G such that each face has an odd number of lines oriented clockwise.



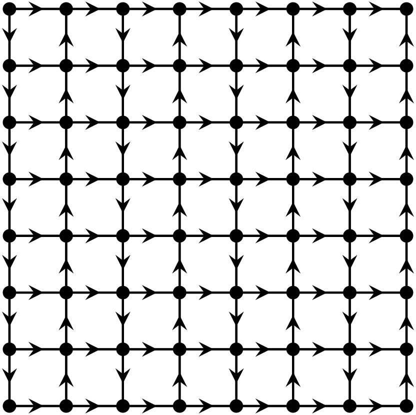
We are left with a Pfaffian orientation of G.

Back to chess boards (aka lattice graphs)



How many perfect matchings does this graph have?

Back to chess boards (aka lattice graphs)



How many perfect matchings does this graph have?

The number of perfect matchings on lattice graphs

- This graph has a high degree of symmetry, and this can be used to calculate the Pfaffian exactly.

The number of perfect matchings on lattice graphs

- This graph has a high degree of symmetry, and this can be used to calculate the Pfaffian exactly.

- For example, a 4×4 lattice graph turns out to have 36 perfect matchings, while an 8×8 graph has $12,988,816$.

The number of perfect matchings on lattice graphs

- This graph has a high degree of symmetry, and this can be used to calculate the Pfaffian exactly.
- For example, a 4×4 lattice graph turns out to have 36 perfect matchings, while an 8×8 graph has $12,988,816$.
- Asymptotically, an $m \times n$ graph has about $(1.339)^{mn}$ perfect matchings.

The number of perfect matchings on lattice graphs

- This graph has a high degree of symmetry, and this can be used to calculate the Pfaffian exactly.
- For example, a 4×4 lattice graph turns out to have 36 perfect matchings, while an 8×8 graph has $12,988,816$.
- Asymptotically, an $m \times n$ graph has about $(1.339)^{mn}$ perfect matchings.
- This result has applications to statistical physics and chemistry – the number of perfect matchings of this graph tells us about the energy of systems where molecules are arranged in a lattice.

Conclusion

- We can count the number of perfect matchings in planar graphs, even though there can be exponentially many of them.

- This is despite the same problem being probably very hard for general graphs.

- The proof brings together many different ideas and it's almost magical that it works.

Further reading

- “Paths, trees and flowers” by Jack Edmonds (1965).
- “Pfaffian” on Wikipedia.
- “Matching theory”, book by Lovàsz and Plummer.
- “Dimer statistics and phase transitions”, P. W. Kasteleyn (1963).
- “Great algorithms” lecture notes by Richard Karp.

Thanks and Merry Christmas!