Fast Subexponential Algorithm for Non-Local Problems on Graphs of Bounded Genus

Presented by Harry Zisopoulos

November 29, 2012

Motivation

- Many computational problems on graphs are $\mathbb{NP}$-hard. Still, exact solutions are sometimes required.
- Subexponential algorithms are a significant improvement over exponential ones.
  - Toy example : For $n = 100$
    - $2^{\sqrt{n}} = 2^{10} = 1.024 * 10^3$
    - $2^n = 2^{100} = 1.2676506 * 10^{30}$

Presentation's topic

- We will focus on :
  - **Non-local graph problems**: You must **always** consider the entire graph in order to solve the problem. It is not possible to solve it in a local "neighborhood" of the graph.
  - E.g. HAMILTONIAN CYCLE, TRAVELLING SALESMAN PROBLEM , LONGEST CYCLE , ...
- Main result: A $2^{O(\sqrt{n})}$ time algorithm , provided the graph can be embedded on the torus.
- Extension: A $2^{O(\sqrt{n})}$ time algorithm , provided the graph has *bounded genus*.

Roadmap

1. Introduce the necessary tools
   - 1.1 Dynamic programming
   - 1.2 Topological graph theory
   - 1.3 Branch decomposition
2. Illustrate proof for HAMILTONIAN CYCLE on torus (*main focus*)
3. Generalization to graphs of bounded genus.

Beginning our journey

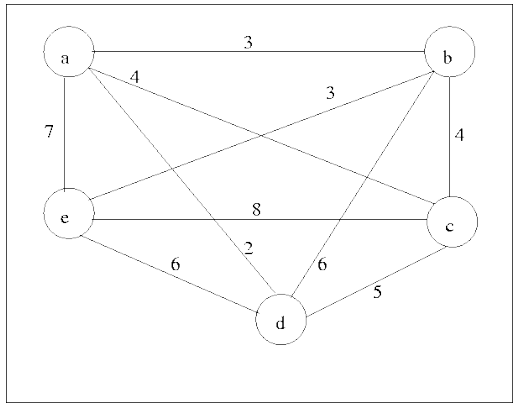
1. Introduce the necessary concepts
    1.1 Dynamic programming

Dynamic programming

- Algorithm design technique. We solve a problem by breaking down to subproblems of smaller size, then combine the solutions.
- Applied when the subproblem size is equal to the original problem size, minus a constant.
- How it works: The same subproblems occur many times. By storing the solutions their solutions the first time, we can avoid costly computation later on.

An example of dynamic programming(1/2)

- TSP: Given a collection of cities and the distance between each pair of them, find the shortest path that visits all the cities and returns to your starting point.



  - Brute force: $O(n!)$
  - Dynamic programming (Held-Karp 1962): $O(n^2 2^n)$
  - Suppose that you want to visit all the capitals of the world.
  - It would make sense to first consider subsets of capitals that are nearby (e.g. in the same continent), and then solve the problem for each subset separately. Then, you can find the shortest connection between those individuals routes to combine them.
  - Can we generalize this?

An example of dynamic programming (2/2)

- Let V be the cities you want to visit. Then for every subset A of V and for every node $t \in A$, compute the shortest path that starts at s, runs through every vertex of A and ends at t.
- This path has length:
  $P(A, t) = \min_{x \in A \setminus \{t\}} P(A \setminus \{t\}, x) + d(x, t)$
- Pick t that minimizes $P(A, t)$.
- Note that you can compute this value for a subset A by re-using your computation for those subsets that have smaller size.

Dynamic Programming Requirements

- Works great on combinatorial optimization problems.
- However, there are two requirements:
  - Principle of Optimality: An optimal solution to a problem instance can be obtained by combining optimal solutions of the subproblems.
    - e.g. Finding the shortest path between two vertices.
    - But not the longest simple path
  - Overlapping subproblems: When solving the problem, the same subproblems should appear many times. Otherwise, there is no gain from storing the solutions.
- Note: If your approach doesn't satisfy the conditions, it doesn't necessarily mean any approach won't.

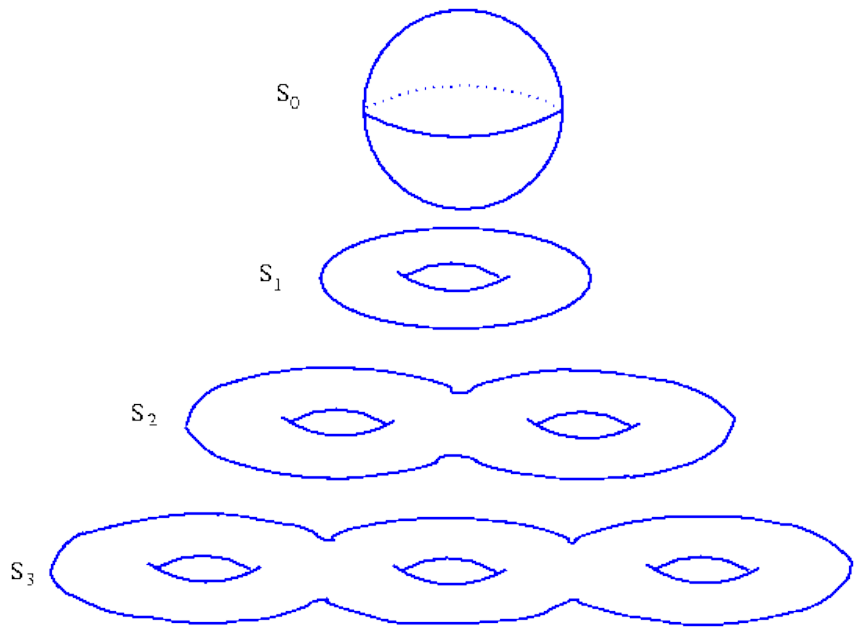
Roadmap

1. Introduce the necessary concepts
   - 1.1 Dynamic programming
   - 1.2 Topological graph theory

Surfaces

- We consider surfaces Σ that are compact, connected 2-manifolds without boundary.
- This definition is not as complex as it looks.
- Surfaces with the same genus are homomorphic. For example,

- $\mathbb{S}_0$ denotes the sphere.
- $\mathbb{S}_1$ denotes the torus.



Surface orientability

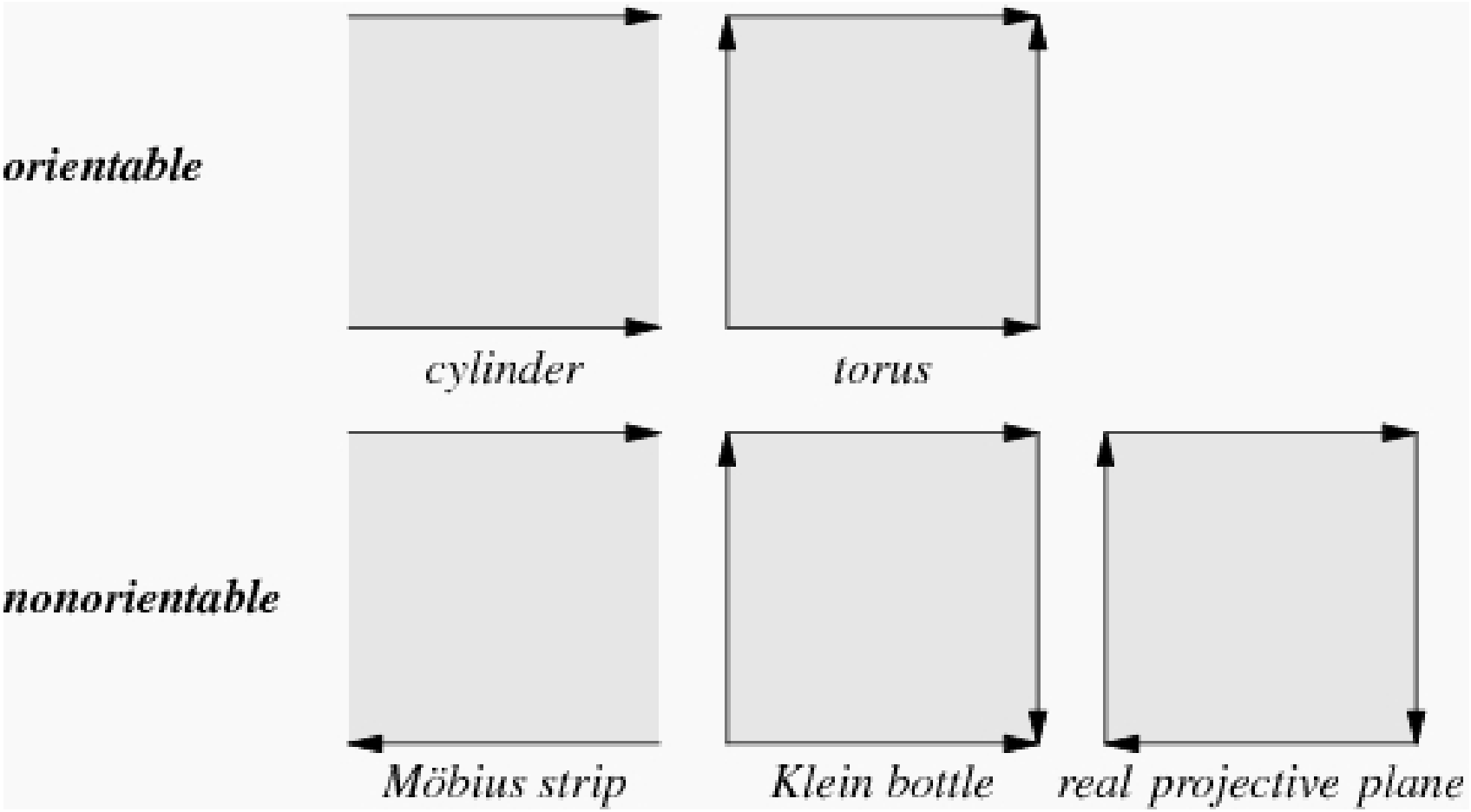


Figure: Orientable and non-orientable surfaces

A non-orientable surface

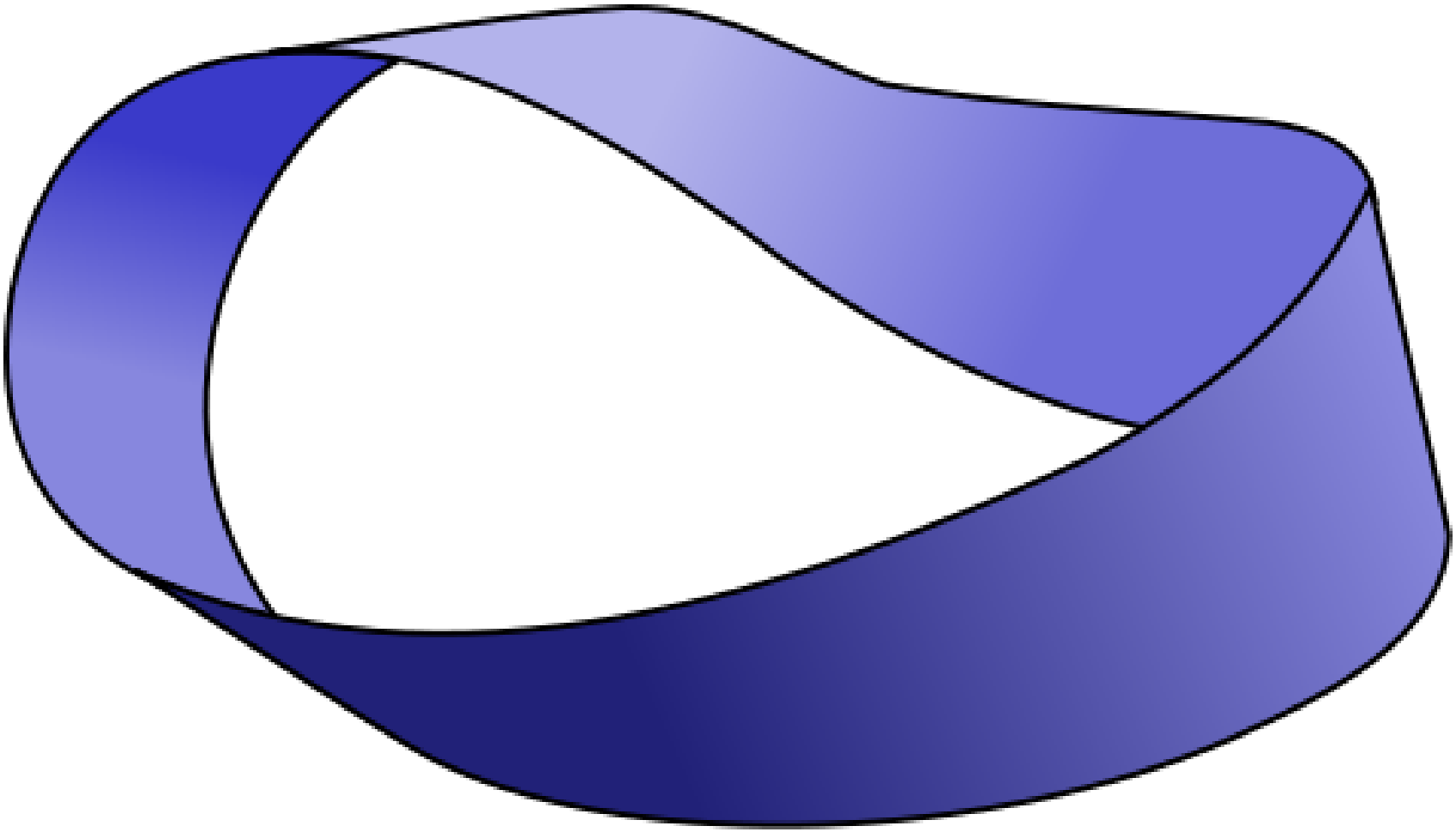
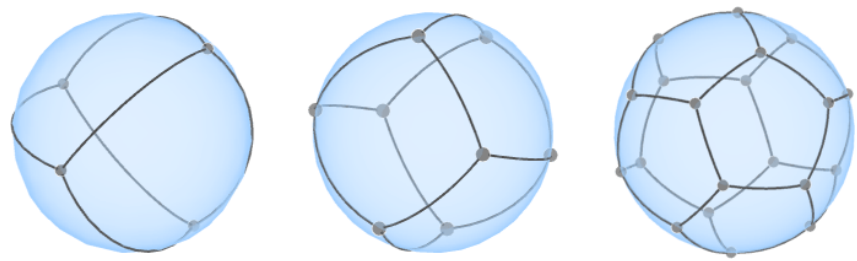


Figure: The Möbius Strip

Graphs on Surfaces

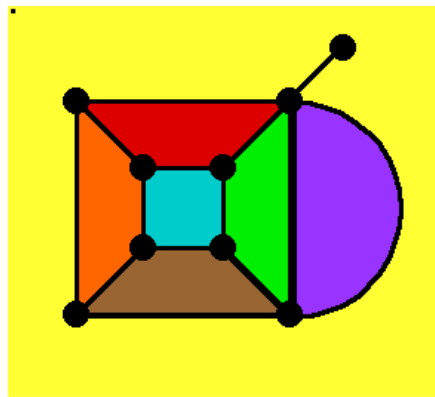
- A Σ-embedded graph G is a graph that can be "drawn" on the surface Σ, such that edges only meet at vertices.
- In general, this is not true for all drawings of G.

- Planar graphs are Σ_0-embedded.



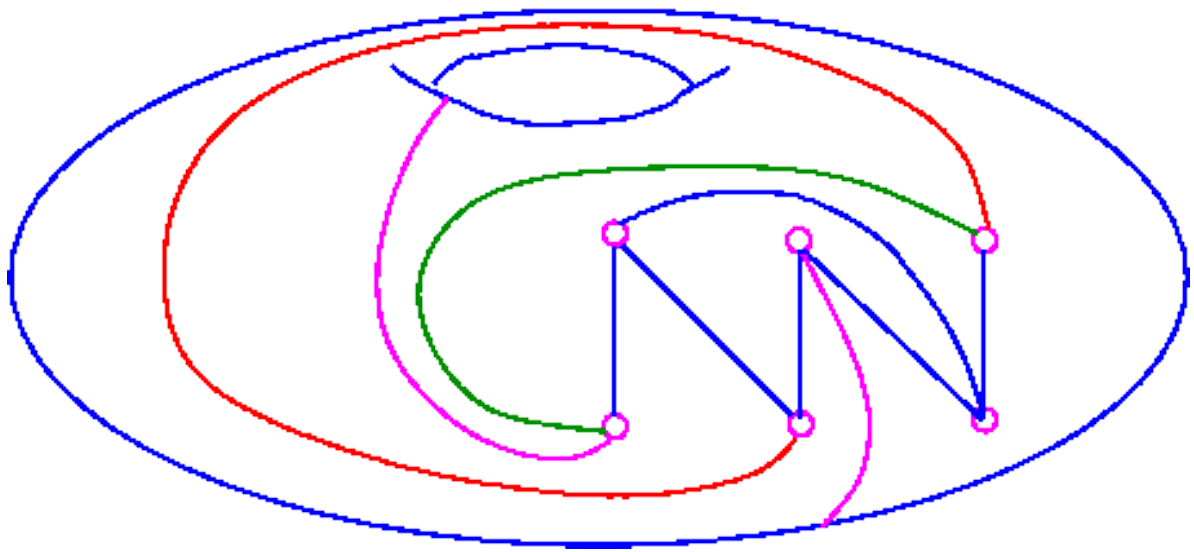
- We will not distinguish between a graph and its drawing.

- We consider 2-cell embeddings $\Longrightarrow$ Faces are homomorphic to the open disk.
- This includes the external face!



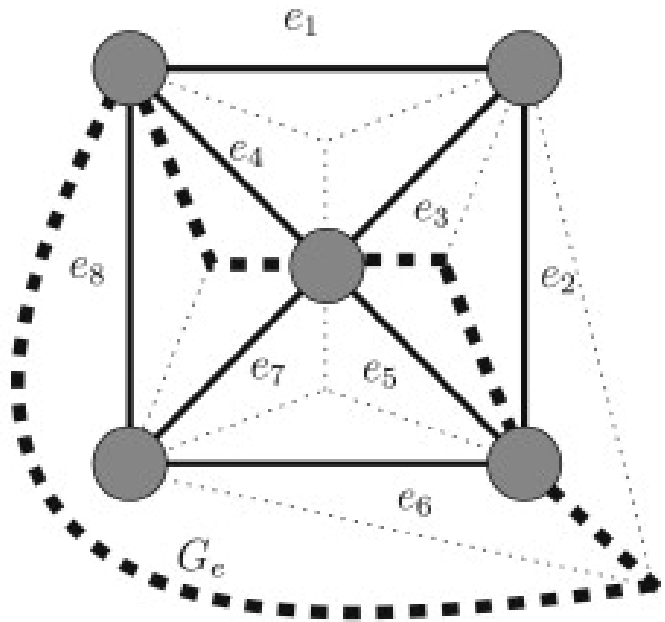
Graph Genus

- With each graph G we associate the following quantities:
  - The orientable genus $g(G)$ is equal to the minimum n s.t. the graph G can be embedded in an orientable surface of orientable genus n.
  - The nonorientable genus $\tilde{g}(G)$ is equal to the minimum n s.t. the graph G can be embedded in a nonorientable surface of non-orientable genus n.
  - The Euler Genus $eg(G) = \min\{2g(G), \tilde{g}\ (G)\}$



Nooses

- An O-arc is a subset of Σ *homomorphic* to a circle.
- A subset of Σ is called G-normal if it meets G only on points that are vertices.
- An O-arc that is G-normal is called a *noose*. The length of a noose N is denoted by $|N|$.

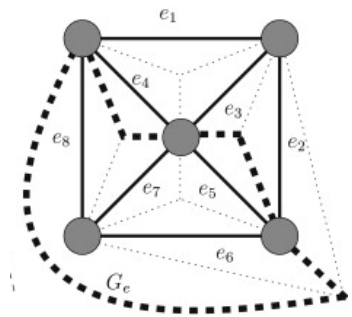


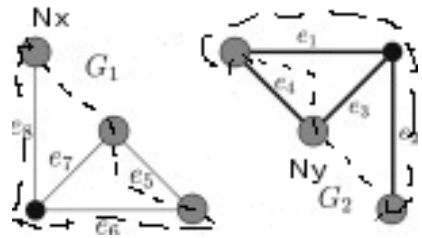
- A noose that "visits" any region only once is called **tight**.
- Let $\Sigma \neq \mathbb{S}_0$. The *representativity* $rep(G)$ of a graph is the length of the smallest noncontractible noose in Σ.
  - Measures how dense a graph is embedded on the surface.

Cutting along a noose

- Given a graph G and a noncontractible tight noose N, we obtain a graph G' obtained by "cutting along" N as follows:

1. For every vertex N meets, it partitions the vertex's neighbors into two sets, N_1 and N_2.
2. We remove this vertex v and we add two new vertices v_1 and v_2.
3. We connect vertex v_1 with the neighbors in N_1 and v_2 with vertices in N_2.





- The vertices v_1 and v_2 for each $v \in N$ define two nooses, N_X and N_Y respectively. We call N_X,N_Y **cut-nooses**.

Two important results

Proposition (1)

There exists a poly-time algorithm that, given a Σ-embedded graph G, $\Sigma \neq \mathbb{S}_0$, finds a noncontractible tight noose of minimum size.

Proposition (2)

Let G be a Σ-embedded graph, $\Sigma \neq \mathbb{S}_0$. Let G' be the graph obtained from G by cutting along a noncontractible tight noose N on G. One of the following is true:

- G' can be embedded in a surface Σ' with $eg(\Sigma') < eg(\Sigma)$.
- G' is the disjoint union of graphs G_1 and G_2 that can be embedded in surfaces Σ_1 and Σ_2 s.t. $eg(\Sigma) = eg(\Sigma_1) + eg(\Sigma_2)$ and $eg(\Sigma_i) > 0$, $i = 1, 2$.

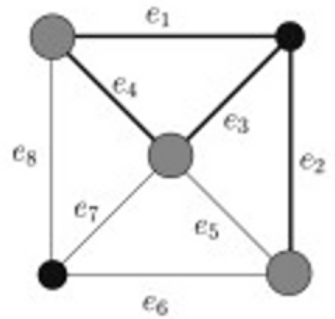
- In both cases, the result of this operation are graphs with **strictly smaller** genus.

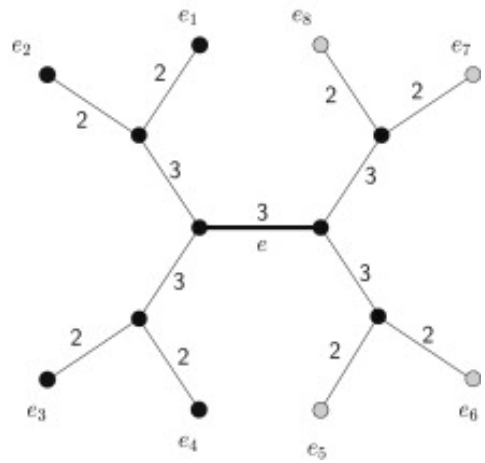
Roadmap

1. Introduce the necessary tools
   - 1.1 Dynamic programming
   - 1.2 Topological graph theory
   - 1.3 Branch decomposition

Branch decomposition

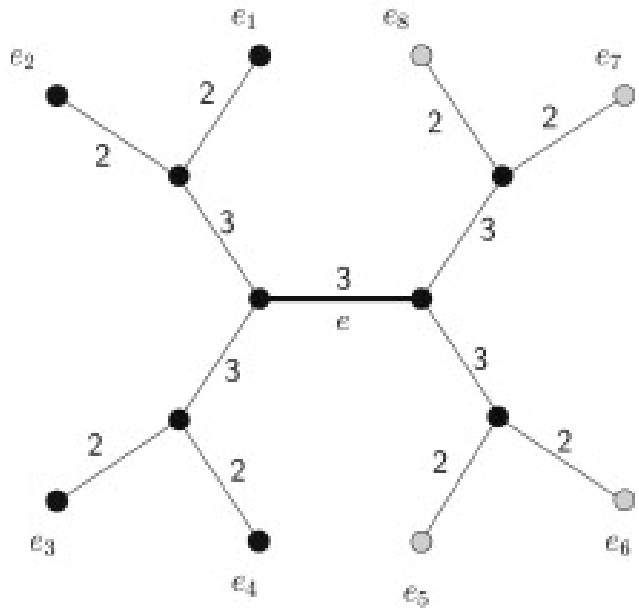
- A useful tool for speeding up computations.
- It provides a structured way of considering subsets of edges.
- A branch decomposition of a graph G is a tuple $< T, \mu >$, where:
  - T is a tree where all inner vertices have degree 3.
  - μ is a bijection from the set of leaves of T to the set of edges of G, $E(G)$.





Branchwidth

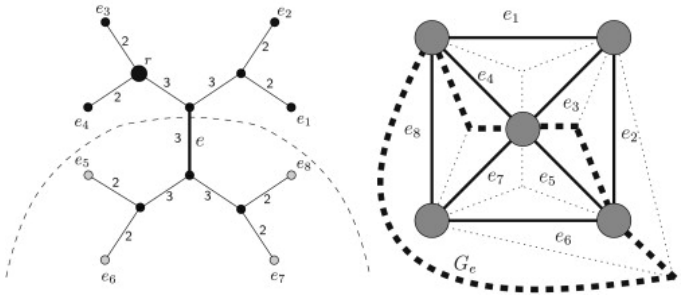
- Each edge $e \in T$, partitions the sets of edges G in two sets,A_e and B_e.
- We define the **middle set** of e, as $mid(e) = V(A_e) \cap V(B_e)$.



- The **width** of a branch decomposition is $\max_{e \in E(T)} |mid(e)|$.
- The branchwidth of G,$bw(G)$, is the **minimum** width over all branch decompositions of G.

Sphere cut decomposition

- A branch decomposition such that for every $e \in E(T)$, the vertices of $mid(e)$ define a tight noose.
- This noose defines two subgraphs, which share only vertices of $mid(e)$.
- We can visit the vertices of $mid(e)$ in order, if we traverse the noose clockwise.



One more important result

Proposition (3)

Let G be a connected $\mathbb{S}_0$-embedded graph without vertices of degree one. There exists a sc-decomposition of G of width **bw(G)**. *Moreover, such a branch decomposition can be constructed in time $O(n^3)$.*

- Minimum width branch decomposition!
- Can be constructed efficiently!
- Actually a sphere-cut decomposition, so for each edge of the decomposition, we have a tight noose on the graph!

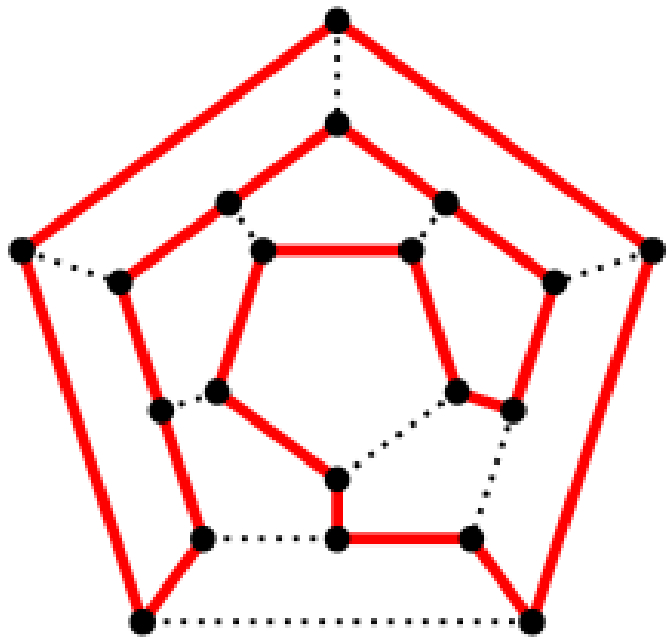
Roadmap

1. Introduce the necessary tools
   - 1.1 Dynamic programming
   - 1.2 Topological graph theory
   - 1.3 Branch decomposition
2. Illustrate proof for HAMILTONIAN CYCLE on torus (*main focus*)

Identifying the problem

- Goal: Solve the HAMILTONIAN CYCLE problem in $2^{O(\sqrt{n})}$ time for graphs that can be embedded on the torus ($\mathbb{S}_1$).

- HAMILTONIAN CYCLE: Given a graph G, is there a cycle that visits each vertex exactly once?
- Special case of TSP, where all distances are 1.
- We can apply dynamic programming.



Solving the problem

- Goal: Solve the HAMILTONIAN CYCLE problem in $2^{O(\sqrt{n})}$ time for graphs that can be embedded on the torus ($\mathbb{S}_1$).
- Intuition: On planar graphs, we can use dynamic programming and sc-decompositions to obtain such an algorithm. This does not apply to surfaces of higher genus (Proposition 3).
- Solution: Let G be a $\mathbb{S}_1$ graph.Cut along a tight noose N to obtain a graph G' that can be embedded on $\mathbb{S}_0$ (Proposition 2).
- Identify the components of a Hamiltonian cycle in G'. Use them to reconstruct the Hamiltonian cycle in G.

Relaxed Hamiltonian Sets

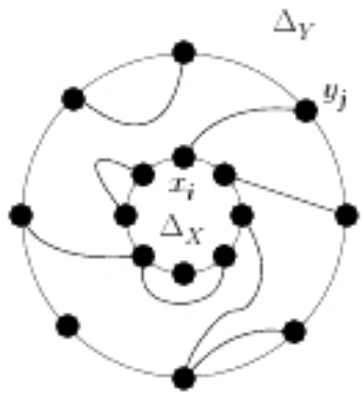
- A **cut of a Hamiltonian cycle** C in G along a tight noose N is a set of disjoint paths in G'.
- Let N_X and N_Y be the cut-nooses obtained from cutting G along N.Also let $x_i \in N_X$ and $y_i \in N_Y$ be duplicated vertices of the same vertex in N.
- We call a set of disjoint paths **P** in G' relaxed Hamiltonian if:
  1. Every path has its endpoints in N_X and N_Y.
  2. Vertex x_i is an endpoint of some path P iff y_i is an endpoint of a path $P' \neq P$.
  3. If one of x_i,y_i is an inner vertex of a path, the other one is not in any path.
  4. Every vertex of $G' \backslash (N_X \cup N_Y)$ is in some path.

From path sets to cycle

- A cut of a Hamiltonian cycle in G is a relaxed Hamiltonian set in G'.
- Do all relaxed Hamiltonian set in G' form a Hamiltonian cycle in G?
- No, but we can check for a single relaxed Hamiltonian set if it does in linear time.
- So, if we check them all either we find a cut of a Hamiltonian cycle in G' or we can say that there is no Hamiltonian cycle in G!
- How many relaxed Hamiltonian sets are there?

Equivalent sets of disjoint paths

- We are mainly interested in the vertices that are part of the cut-nooses, as long as every other vertex belongs to a path. We can make this requirement precise.
- Two sets of disjoint paths $\mathbf{P} = (P_1, \ldots, P_k)$ and $\mathbf{P'} = (P'_1, \ldots P'_k)$ are equivalent if for all i the paths P_i and P'_i have the same endpoints and all inner vertices in $\mathbf{P}$ are also inner vertices in $\mathbf{P'}$.

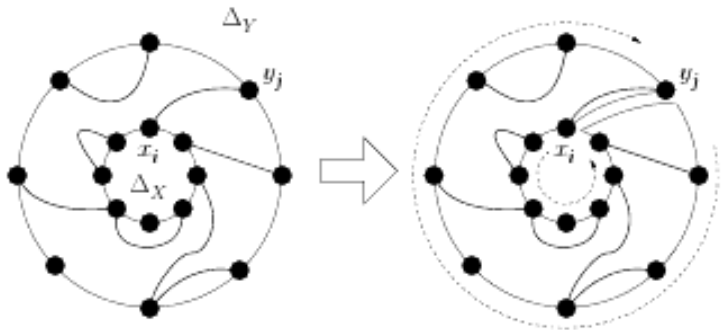


Enumerating relaxed Hamiltonian sets

Lemma (1)

The number of **different** *equivalence classes of relaxed Hamiltonian sets in G' is $O(2^{3k})$, where k is the length of the tight noose N.*

- We can order the vertices of N_X and N_Y.
- The paths from N_X to N_Y do not cross.

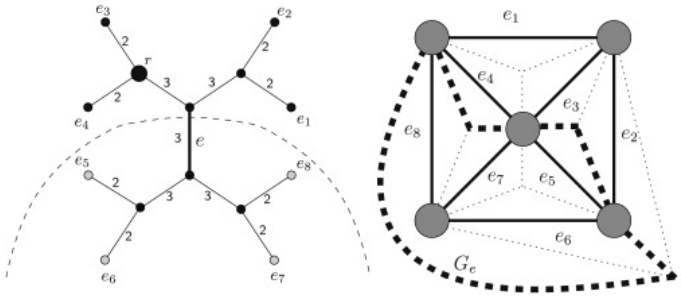


  - We can also have paths from a cut-noose to itself.
  - If a vertex of a cut-noose is an inner vertex in a path, its correspondent vertex in the other cut-noose is not in any path.

Reconstructing the Hamiltonian cycle

- A candidate K of an equivalence class of relaxed Hamiltonian sets is a set of paths with vertices only in $N_X \cup N_Y$, satisfying the first 3 conditions of the definition of a relaxed Hamiltonian set.
  - We are missing the vertices in $G' \backslash (N_X \cup N_Y)$ that would belong to some path.
- If we could find a relaxed Hamiltonian set $\mathbf{P}$ that is equivalent to K, we could use $\mathbf{P}$ to reconstruct the Hamiltonian cycle.

Finding a relaxed Hamiltonian set **P**



- We use a **rooted** sc-decomposition of G', $< T, \mu >$ and we perform dynamic programming over middle sets, in a bottom-up fashion.
- Each middle set defines a tight noose O_e in G'. Let G_e be the component of G' that is bounded by O_e.
  - All paths of $\mathbf{P} \cap G_e$ start and end in O_e and $G_e \cap (N_x \cup N_y)$.
  - For each G_e, we keep track of the equivalence classes of sets of disjoint paths with endpoints in O_e.
  - When we reach the root, we will obtain the equivalence class of the candidate K. Then we can reconstruct $\mathbf{P}$ by traversing T from the root to the leaves.

Preliminaries for running time

Lemma (2)

For each candidate K and a given sc-decomposition of G' of width ℓ, the running time of the dynamic programming step of the algorithm for K is $O(2^{\ell} n^{O(1)})$.

Lemma (3)

Let G be a $\mathbb{S}_1$-embedded graph on n vertices and G' the planar graph obtained by cutting along a noncontractible tight noose. Then $bw(G') = O(\sqrt{n})$.

Lemma (4)

Let G be a $\mathbb{S}_1$-embedded graph. Then $rep(G) = O(\sqrt{n})$.

Running time

1. Cutting along a tight noose can be done in polynomial time (Proposition 1)
2. The number of all candidates of relaxed Hamiltonian sets is $O(2^{3k})n^{O(1)}$, by Lemma 1, where k is the length of the noose of minimum size obtained.
3. We can construct an optimal sc-decomposition in polynomial time.
4. Dynamic programming on a single candidate takes $O(2^{\ell}n^{O(1)})$ time.
5. The total running time is $\mathbf{O(2^{\ell+3k}n^{O(1)})}$.
6. By applying lemmas 3 and 4, we get that the total running time of the algorithm is $2^{O\sqrt{n}}n^{O(1)}$.

Roadmap

1. Introduce the necessary tools
   - 1.1 Dynamic programming
   - 1.2 Topological graph theory
   - 1.3 Branch decomposition
2. Illustrate proof for HAMILTONIAN CYCLE on torus (*main focus*)
3. Generalization to graphs of bounded genus.

Generalization to bounded genus

- The same running time, $2^{O(\sqrt{n})}$ can be achieved for graphs with bounded genus.
- However, several modifications are needed:
  1. We cut along a noncontractible tight noose many times, so we get a set of cut-nooses that we have to handle carefully.
  2. The definition of a relaxed Hamiltonian set of paths alters accordingly.
  3. Similar bounds can be proven for the branchwidth of G', the number of different equivalence classes of relaxed Hamiltonian sets and the representativity of G.
  4. Other technical issues.
- Bonus: The same technique can be used to design parameterized algorithms as well!

Concluding remarks

- The technique is used to solve **non-local** problems on graphs of bounded genus.
- The first step is to "planarize" these graphs by cutting along tight nooses. We also define a **more general** problem on **planar graphs** that is related to the original problem.
- We can then use planar graph techniques, such as **dynamic programming** in **sphere-cut decompositions** to solve the problem efficiently.
- We obtain algorithms with running time $2^{O(\sqrt{n})}$ for graphs of bounded genus.
- This line of research is very active and there is a variety of more recent results.

Sources used (1/2)

- I would like to thank the respective owners of the images I used in the slides. The images were used for educational purposes and they were retrieved from the following links:

1. TSP Example: "http://lcm.csa.iisc.ernet.in/dsa/node187.html" , Lecture notes for "Algorithms and Data Structures" , Y. Narahari, Indian Institute of Science
2. Surfaces examples and $K_{3,3}$ on the torus: "http://www.personal.kent.edu/~rmuhamma/GraphTheory/MyGraphTheory/embedding.htm" , Lecture notes for "Algorithmic Graph Theory" , Rashin Bin Muhammad , Kent State University
3. Mobius strip: "http://www.wpclipart.com/signs_symbol/optical_illusions/illusions_2/Mobius_Strip.png.html"

Sources used (2/2)

4. Graph faces : http://www.math.lsa.umich.edu/mmss/coursesONLINE/graph/graph5/" , Dale Winter , University of Michigan
5. Sphere embedded graphs: "http://11011110.livejournal.com/141365.html" , Blog of David Eppstein
6. Hamiltonian cycle example: "http://en.wikipedia.org/wiki/File:Hamiltonian_path.svg" , Wikipedia.
7. Many figures were taken from the paper "Subexponential parameterized algorithms" that can be found on "http://www.sciencedirect.com/science/article/pii/S1574013708000063"