Safe algorithms:

Dealing with multiple solutions in practice

Alexandru Tomescu, Department of Computer Science, University of Helsinki
PI of Graph Algorithms Team / Algorithmic Bioinformatics Research Group

UNIVERSITY OF HELSINKI
FACULTY OF SCIENCE

Outline

- Motivation from computational biology
- Previous related notions
- Safety
- Simple examples of techniques for safe algorithms
- Conclusions

The Human Genome Project

"The most important biomedical research undertaking of the 20th Century"

February 2001

- *Carried out from 1990–2003, it was one of the most ambitious and important scientific endeavors in human history.*
- *The [...] cost for the Human Genome Project was $3 billion [...].*
- *Assemble[d] interdisciplinary groups from across the world, involving experts in engineering, biology, and computer science [...].*
  - https://www.genome.gov/about-genomics/educational-resources/fact-sheets/human-genome-project

NATURE BIOTECHNOLOGY VOLUME 29 NUMBER 11 NOVEMBER 2011

How to apply de Bruijn graphs to genome assembly

Phillip E C Compeau, Pavel A Pevzner & Glenn Tesler

A mathematical concept known as a de Bruijn graph turns the formidable challenge of assembling a contiguous genome from billions of short sequencing reads into a tractable computational problem.

The development of algorithmic ideas for next-generation sequencing can be traced back 300 years to the Prussian city of Königsberg (present-day Kaliningrad, Russia), where seven bridges joined the four parts of the city located on opposing banks of the Pregel River and two river islands (**Fig. 1a**). At the time, Königsberg's residents enjoyed strolling through their city, and they wondered if every part of the city could be visited by walking across each of the seven bridges exactly once and returning to one's starting location. The solution came in 1735, when the great mathematician Leonhard Euler[1] made a

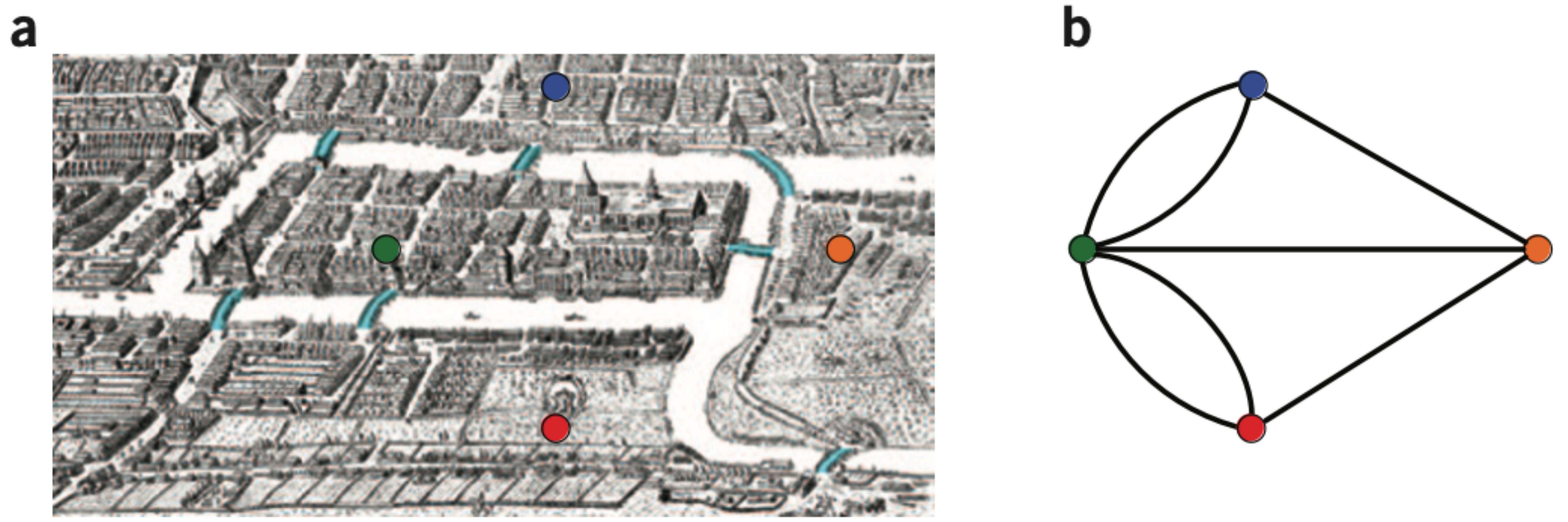
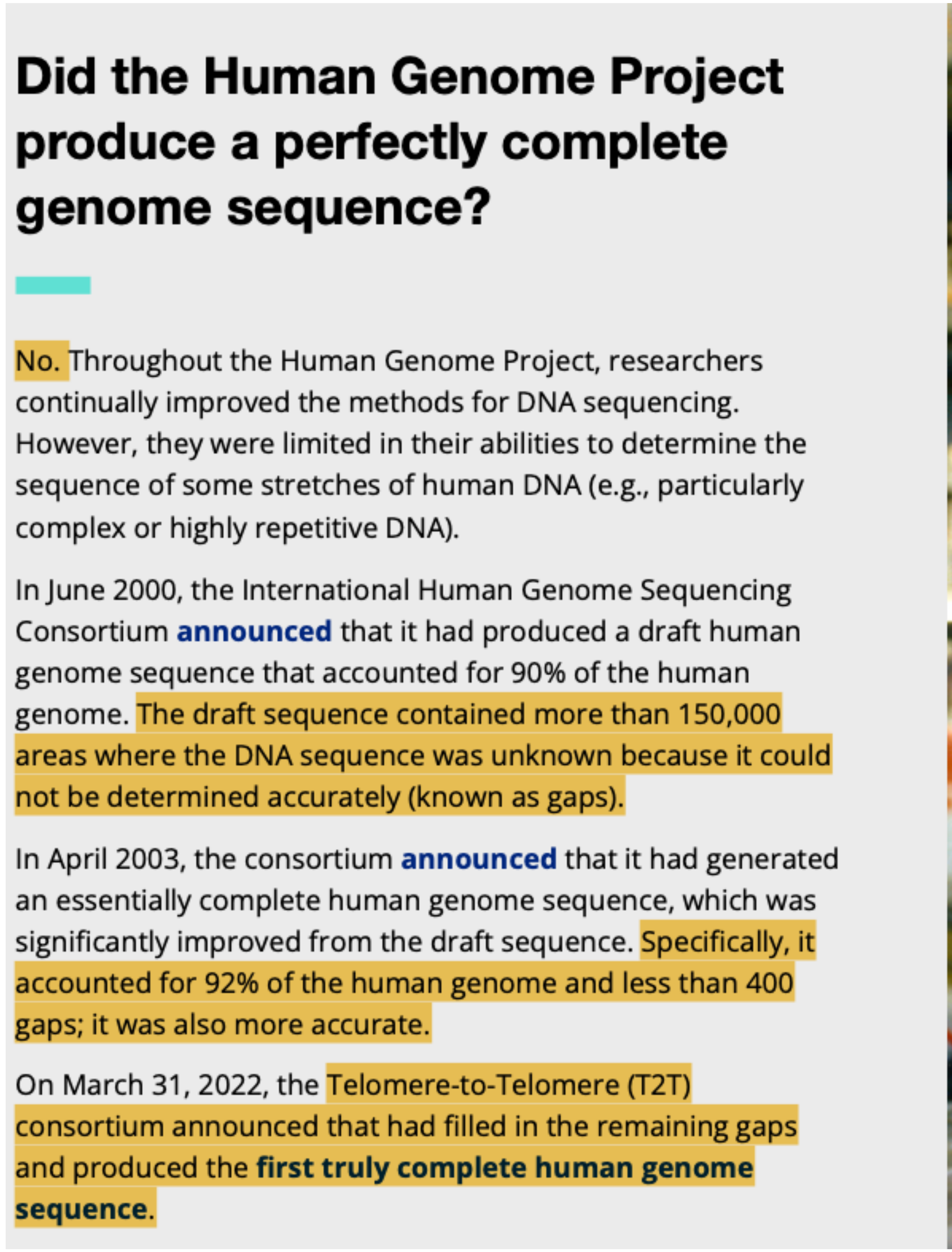


Figure 1 Bridges of Königsberg problem. (**a**) A map of old Königsberg, in which each area of the city is labeled with a different color point. (**b**) The Königsberg Bridge graph, formed by representing each of four land areas as a node and each of the city's seven bridges as an edge.



Did the Human Genome Project produce a perfectly complete genome sequence?

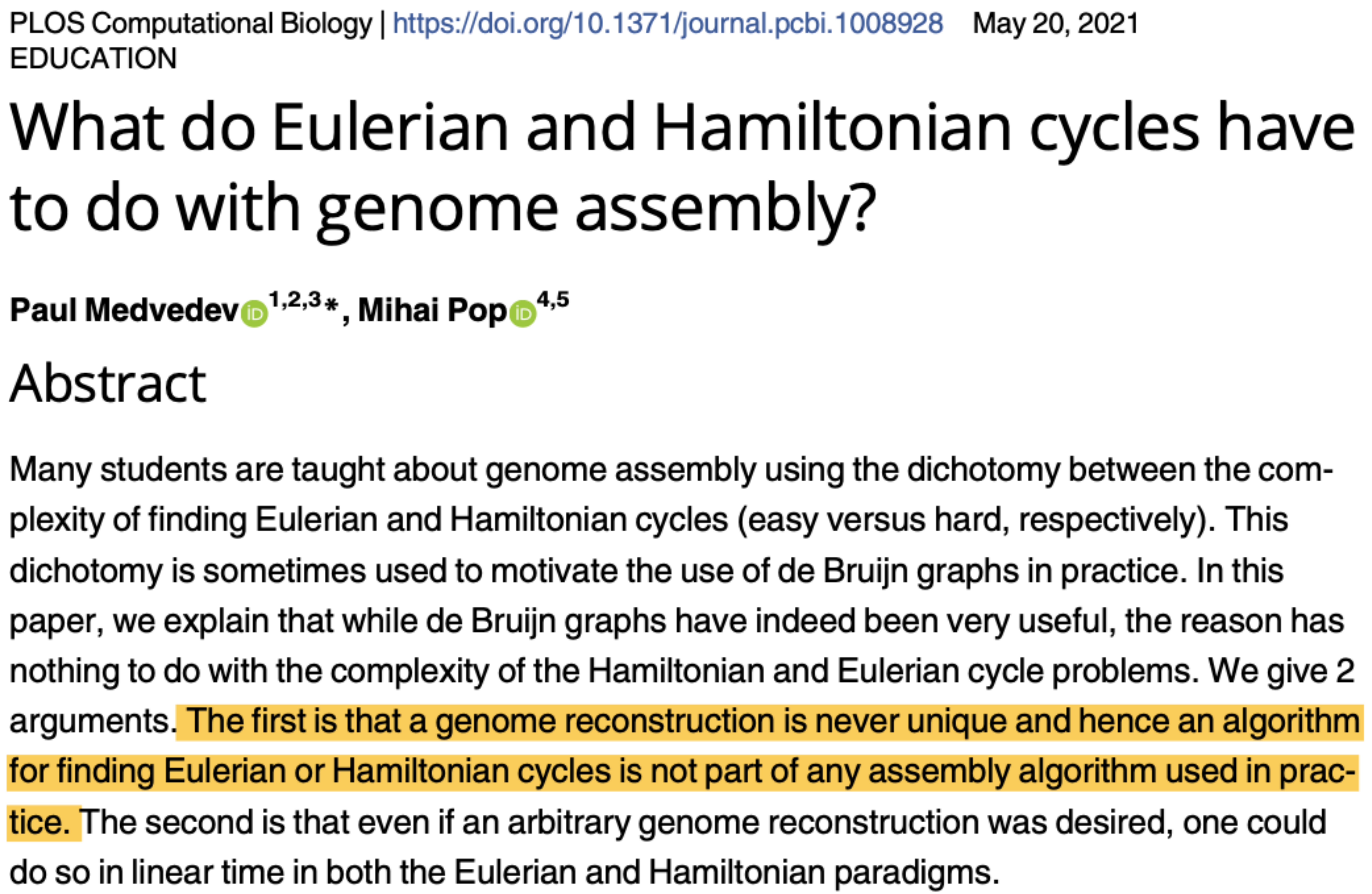
No. Throughout the Human Genome Project, researchers continually improved the methods for DNA sequencing. However, they were limited in their abilities to determine the sequence of some stretches of human DNA (e.g., particularly complex or highly repetitive DNA).

In June 2000, the International Human Genome Sequencing Consortium **announced** that it had produced a draft human genome sequence that accounted for 90% of the human genome. The draft sequence contained more than 150,000 areas where the DNA sequence was unknown because it could not be determined accurately (known as gaps).

In April 2003, the consortium **announced** that it had generated an essentially complete human genome sequence, which was significantly improved from the draft sequence. Specifically, it accounted for 92% of the human genome and less than 400 gaps; it was also more accurate.

On March 31, 2022, the Telomere-to-Telomere (T2T) consortium announced that had filled in the remaining gaps and produced the **first truly complete human genome sequence**.

https://www.genome.gov/about-genomics/educational-resources/fact-sheets/human-genome-project



PLOS Computational Biology | https://doi.org/10.1371/journal.pcbi.1008928 May 20, 2021
EDUCATION

What do Eulerian and Hamiltonian cycles have to do with genome assembly?

Paul Medvedev[1,2,3]*, **Mihai Pop**[4,5]

Abstract

Many students are taught about genome assembly using the dichotomy between the complexity of finding Eulerian and Hamiltonian cycles (easy versus hard, respectively). This dichotomy is sometimes used to motivate the use of de Bruijn graphs in practice. In this paper, we explain that while de Bruijn graphs have indeed been very useful, the reason has nothing to do with the complexity of the Hamiltonian and Eulerian cycle problems. We give 2 arguments. The first is that a genome reconstruction is never unique and hence an algorithm for finding Eulerian or Hamiltonian cycles is not part of any assembly algorithm used in practice. The second is that even if an arbitrary genome reconstruction was desired, one could do so in linear time in both the Eulerian and Hamiltonian paradigms.

Enumeration

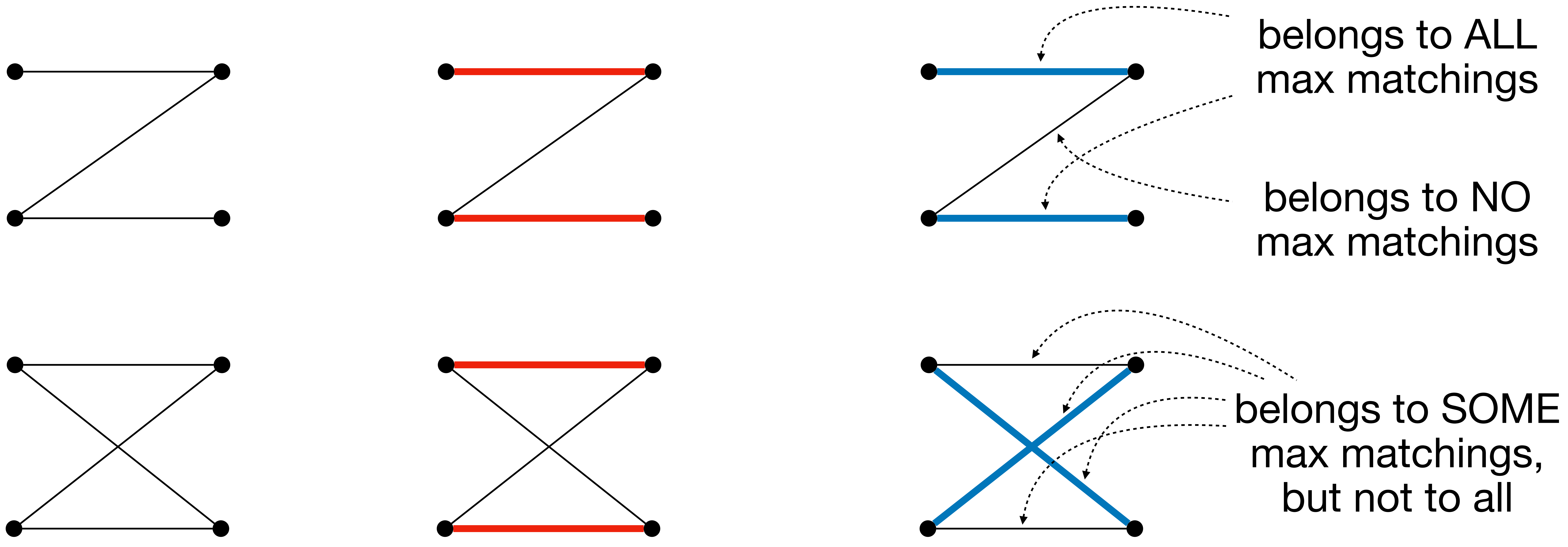
- Counting (#P-complete problems), constant-time delay, poly-time delay, etc
    - Carl Kingsford et al. "Assembly complexity of prokaryotic genomes using short reads", BMC Bioinformatics 11:21, 2010

- *k*-best enumeration
    - David Eppstein, "*k*-best enumeration." arXiv preprint arXiv:1412.5075 (2014):

      Many real-world problems are only approximately modeled by mathematical formulations. [...] Once such a list of candidates is generated, one can apply more sophisticated quality criteria, wait for data to become available to choose among them, or present them all to human decision-makers. On the other hand, the exponential growth of the solution space for many combinatorial optimization problems would make it infeasible to list all possible candidate solutions; instead, some filtering is needed.

    - Enumerate *k*-shortest s-t paths
        - See e.g. Yen's algorithm: https://en.wikipedia.org/wiki/Yen's_algorithm
          $O(k|V|(|E| + |V|\log|V|))$

All, none, some


belongs to ALL max matchings
belongs to NO max matchings
belongs to SOME max matchings, but not to all

Bipartite graph G Maximum matching 1 Maximum matching 2

Persistency

- For a bipartite graph $G = (V, E)$, the partition $(E_{all}, E_{none}, E_{some})$ is the <u>persistency partition of E for maximum cardinality matchings:</u>
  - Computable in time $O(|V||E|)$ (or faster)
  - Marie-Christine Costa "Persistency in maximum cardinality bipartite matchings." Operations Research Letters 15.3 (1994): 143-149.
- For a graph $G = (V, E)$, let $(V_{all}, V_{none}, V_{some})$ be the <u>persistency partition of V for maximum independent sets:</u>
  - Computing (V_{all}, V_{none}) is NP-hard
  - Peter Hammer et al. "Vertices belonging to all or to no maximum stable sets of a graph." SIAM Journal on Algebraic Discrete Methods 3.4 (1982): 511-522.

Backbones for Boolean satisfiability

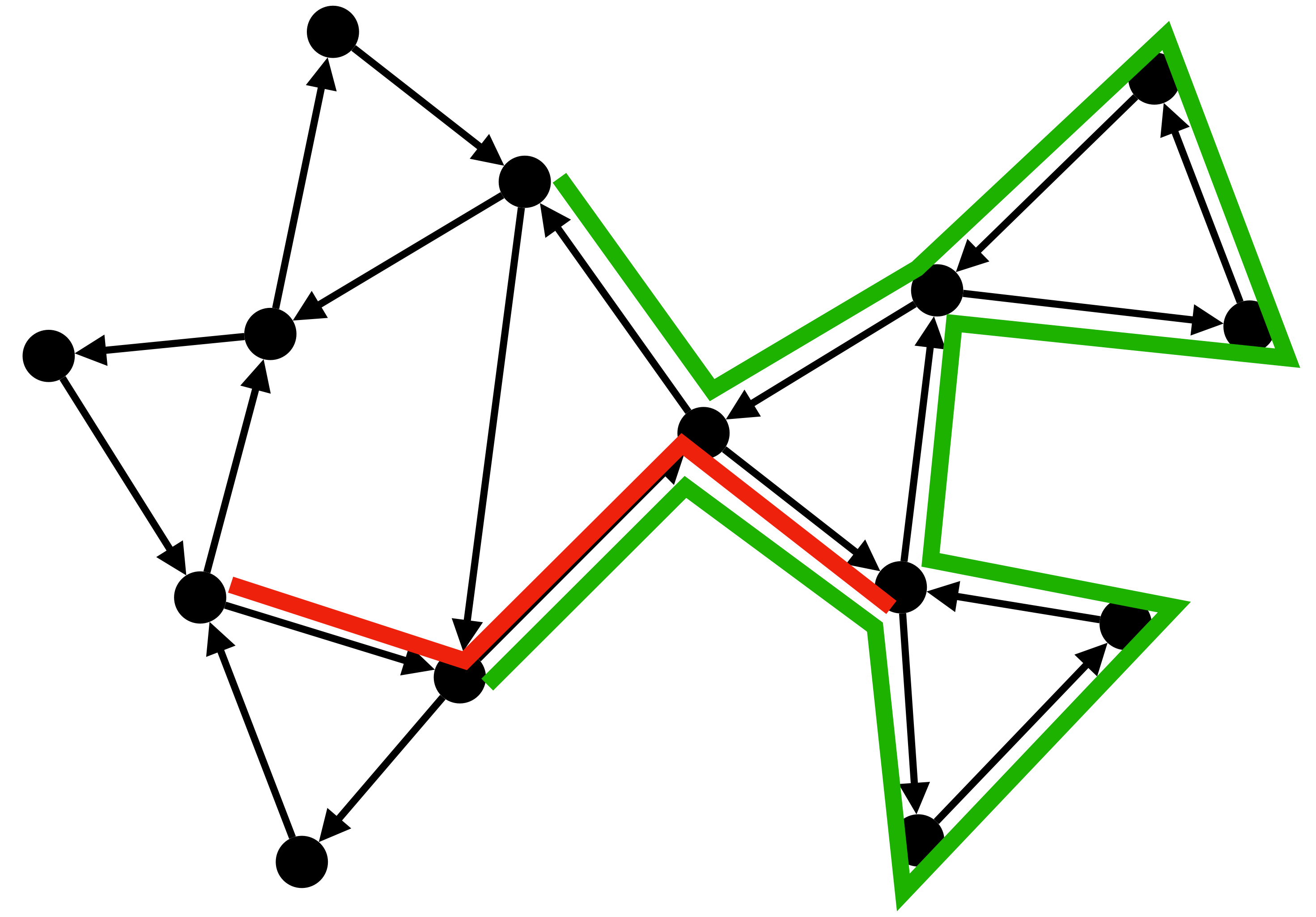
- For a propositional formula φ, the backbone of φ is its set of variables that are set to TRUE in every assignment satisfying φ
- Alessandro Previti et al. "On computing generalized backbones." 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2017:

  [...] backbone variables of SAT instances encoding real-world problem instances can have various kinds of meaningful interpretations [...] backbones can represent faults in integrated circuits
- Mikoláš Janota et al. "Algorithms for computing backbones of propositional formulae." AI Communications 28.2 (2015) 161-177:

  The concept of backbone appears under various terms. For instance, inadmissible and necessary variables [16], bound literals [15], fixed assignments [37], units [21], or frozen variables [1].

Safety: contiguous persistency

- Consider Eulerian cycles (closed walks covering every edge exactly once)
  - Nothing to do: every edge is persistent, every vertex is persistent

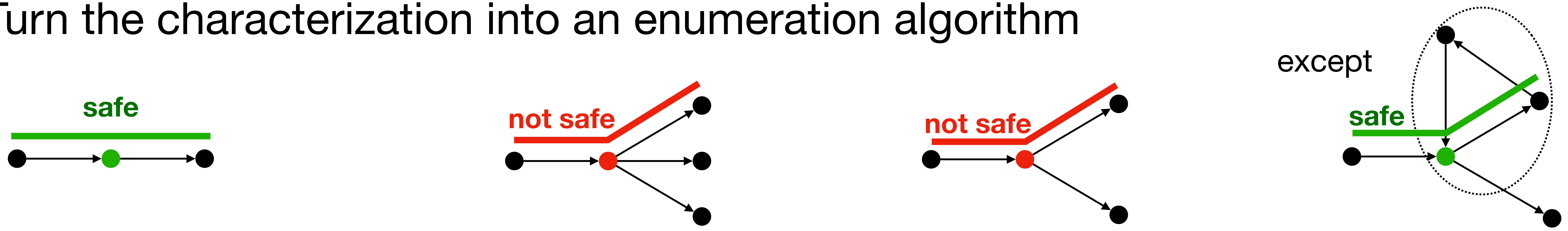


- Let P be a problem on a graph $G = (V, E)$ whose solutions are certain sets $\mathscr{W}$ of walks in G
- We say that a walk W is a safe walk for P on G if for every solution $\mathscr{W}$, W is a subwalk of some walk in $\mathscr{W}$
- Enumerate all maximal safe walks for P on G

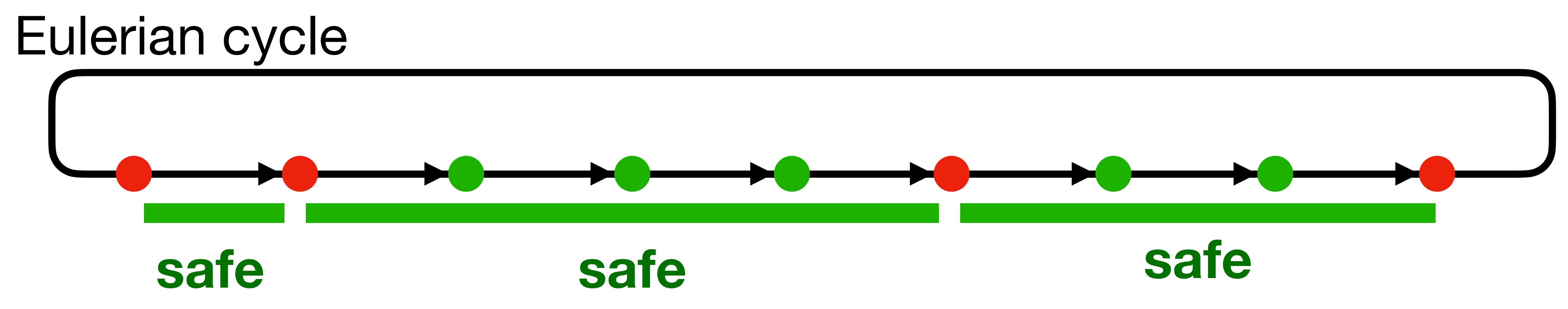
- Safe algorithm for P: an algorithm outputting only safe walks for P
- Safe and complete algorithm for P: an algorithm outputting all (maximal) safe walks for P

A general approach to compute safe walks

1. Characterize safe walks
2. Turn the characterization into an enumeration algorithm



- A walk W is safe for Eulerian cycles on G if every internal node v of W either:
    1. has degree 1, or
    2. has degree 2 and is a cut node of the undirected graph underlying G



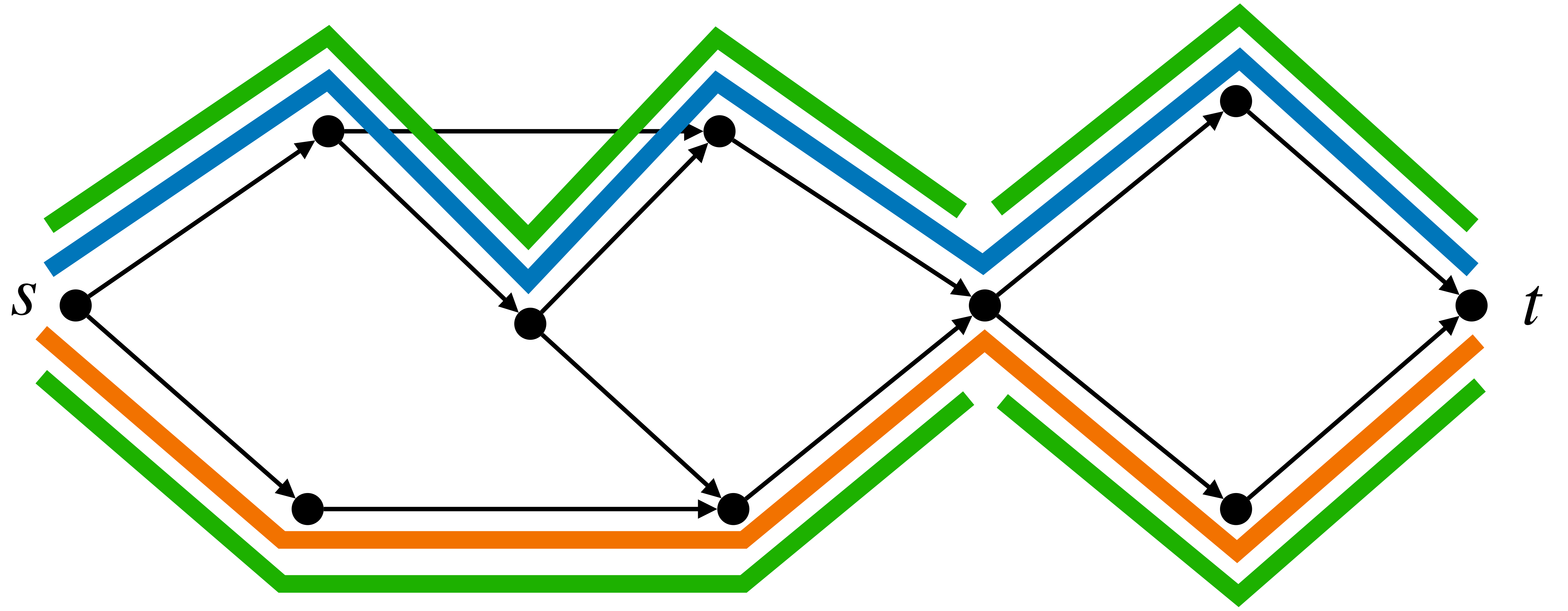
- Finding all maximal safe walks for Eulerian cycles: $O(|E|)$ time
    - Nidia Obscura Acosta, A. T. "Simplicity in Eulerian circuits: Uniqueness and safety." Information Processing Letters 183 (2024): 106421

A general approach to detect safety

- "Hide-and-test" for edges in E_{all} for maximum bipartite matchings
  - Compute **`max-matching-size`**(G)
  - **For** each edge $e \in E$:
    - **If** **`max-matching-size`**$(G \backslash e)$ < **`max-matching-size`**(G):
    - **Then** add e to E_{all}
- Complexity: $O(|E| \cdot \text{max-matching(G)})$
  - Recall: can do in $O(|V||E|)$ (or faster) [Costa, 1994]

Hide-and-test for safety: example

- Given a directed acyclic graph $G = (V, E)$, $s, t \in V$, we say that paths $P_1, \ldots, P_k$ are a <u>minimum s-t path cover</u> if:
    - every P_i is an s-t path
    - every $v \in V$ belongs to at least one P_i
    - k is minimum

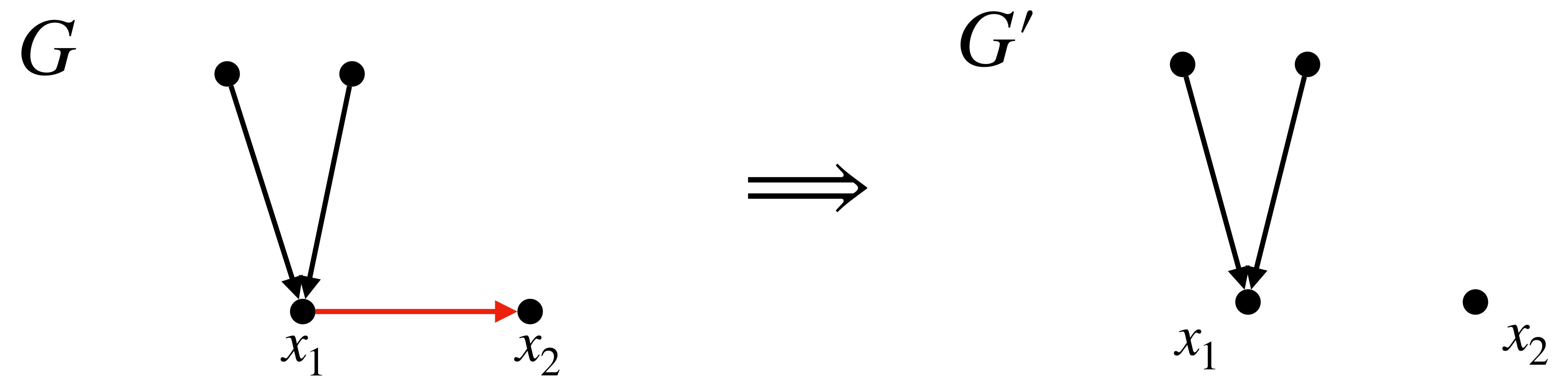


- Compute all maximal safe paths
    - Manuel Cáceres et al. "Safety in multi-assembly via paths appearing in all path covers of a DAG", IEEE/ACM TCBB 2022

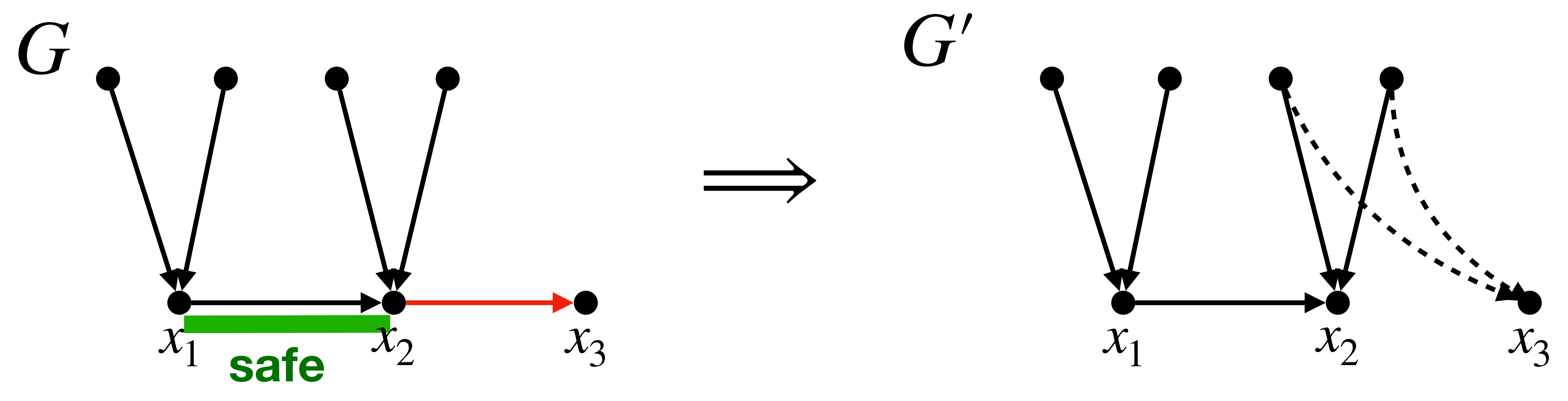
How to hide paths?

Length-1 safe paths:

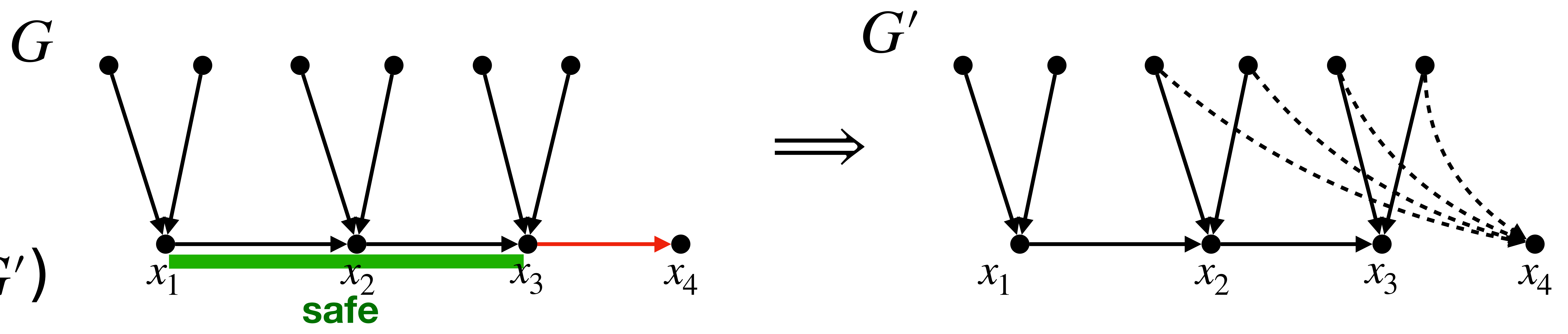
(x_1, x_2) unsafe iff
mpc-size(G)=**mpc-size**(G')



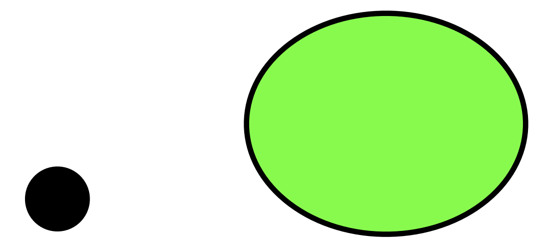
Length-2 safe paths:
(x_1, x_2, x_3) unsafe iff
mpc-size(G)=**mpc-size**(G')



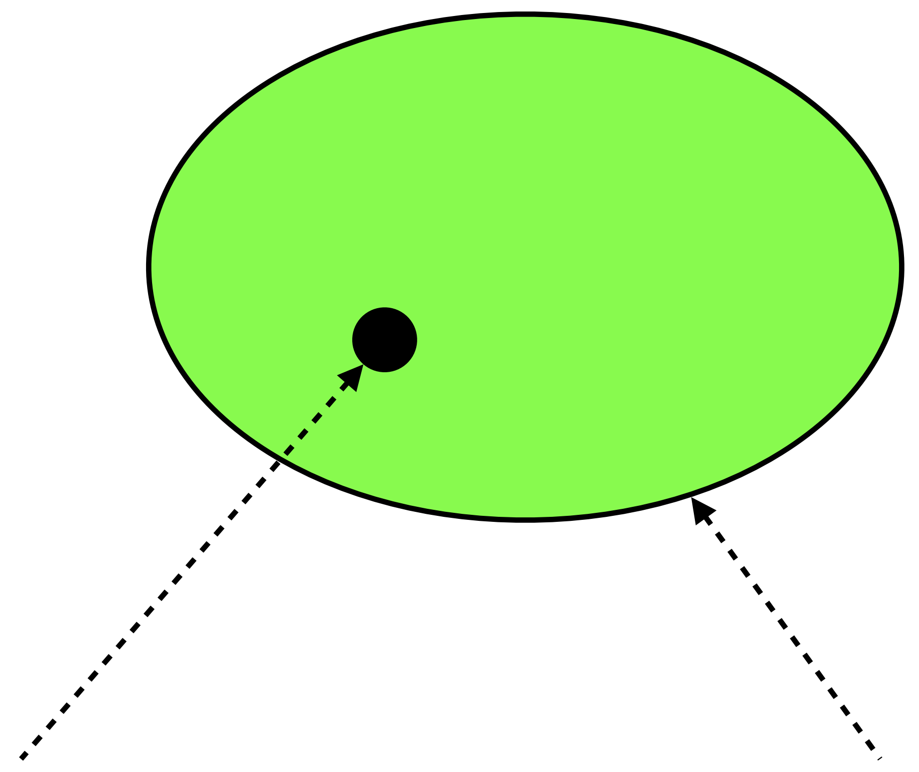
Length-3 safe paths:
(x_1, x_2, x_3, x_4) unsafe iff
mpc-size(G)=**mpc-size**(G')



When are safe walks correct in practice



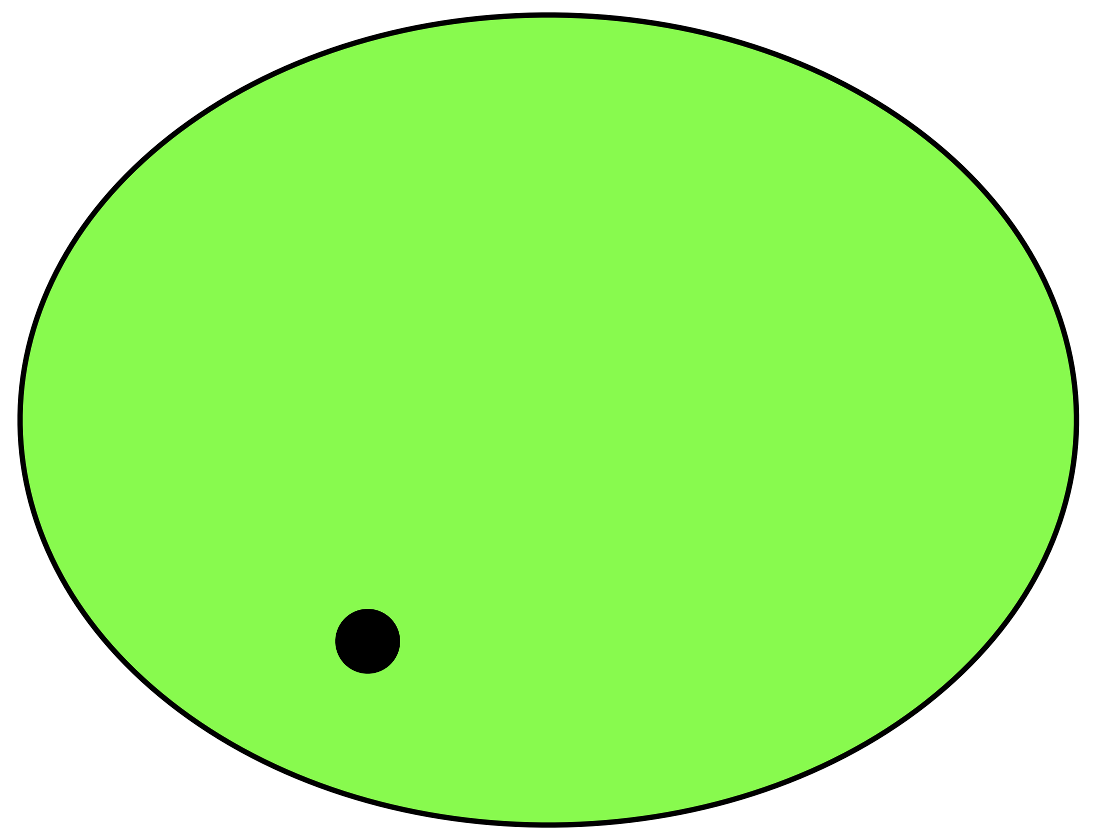
smaller solution space
⇒
risk of the true solution falling outside



true unknown solution

possible problem solutions

safe walks are also part of the true unknown solution



larger solution space
⇒
smaller safe walks

- Eulerian cycle
- Closed walk covering every edge at least once
  - Massimo Cairo et al. "An Optimal O(nm) Algorithm for Enumerating All Walks Common to All Closed Edge-covering Walks of a Graph", CPM 2019, ACM TALG 2019
  - Massimo Cairo et al. "Genome Assembly, from Practice to Theory: Safe, Complete and Linear-Time", ICALP 2021, ACM TALG 2023

- Hamiltonian cycle ∈ NP-complete
- Closed walk covering every node at least once ∈ P
  - Massimo Cairo et al. "Cut Paths and Their Remainder Structure, with Applications." STACS 2023
  - A.T., Paul Medvedev, "Safe and complete contig assembly via omnitigs", RECOMB 2016

Conclusions

Be safe when you output!

- For problems about discovering some object:
  - Safety is a practical way of dealing with multiple solutions
  - Safety can lead to interesting computational problems
- Open theoretical problems (meta-theorems):
  - Is the "safety version" of any problem in P also in P?
    - Might be much simpler for persistency
  - Is the "safety version" of any NP-hard problem also NP-hard?
    - Safety seems harder than detecting uniqueness