Finding path motifs in temporal graphs using algebraic fingerprints

Suhas Thejaswi

Department of Computer Science,
Aalto University
suhas.thejaswi@aalto.fi

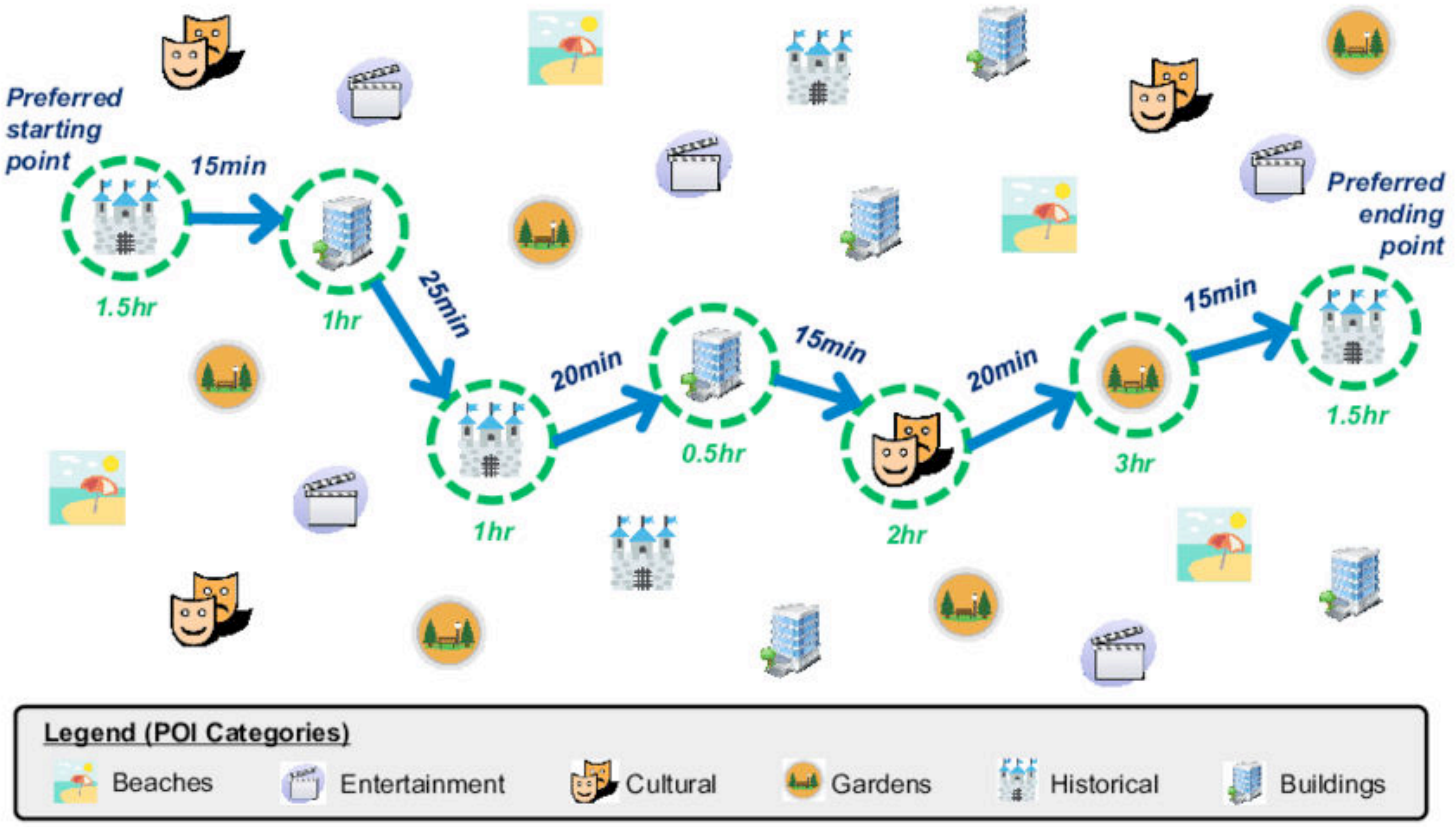
Juho Lauri

Helsinki, Finland
juho.lauri@gmail.com

Aristides Gionis

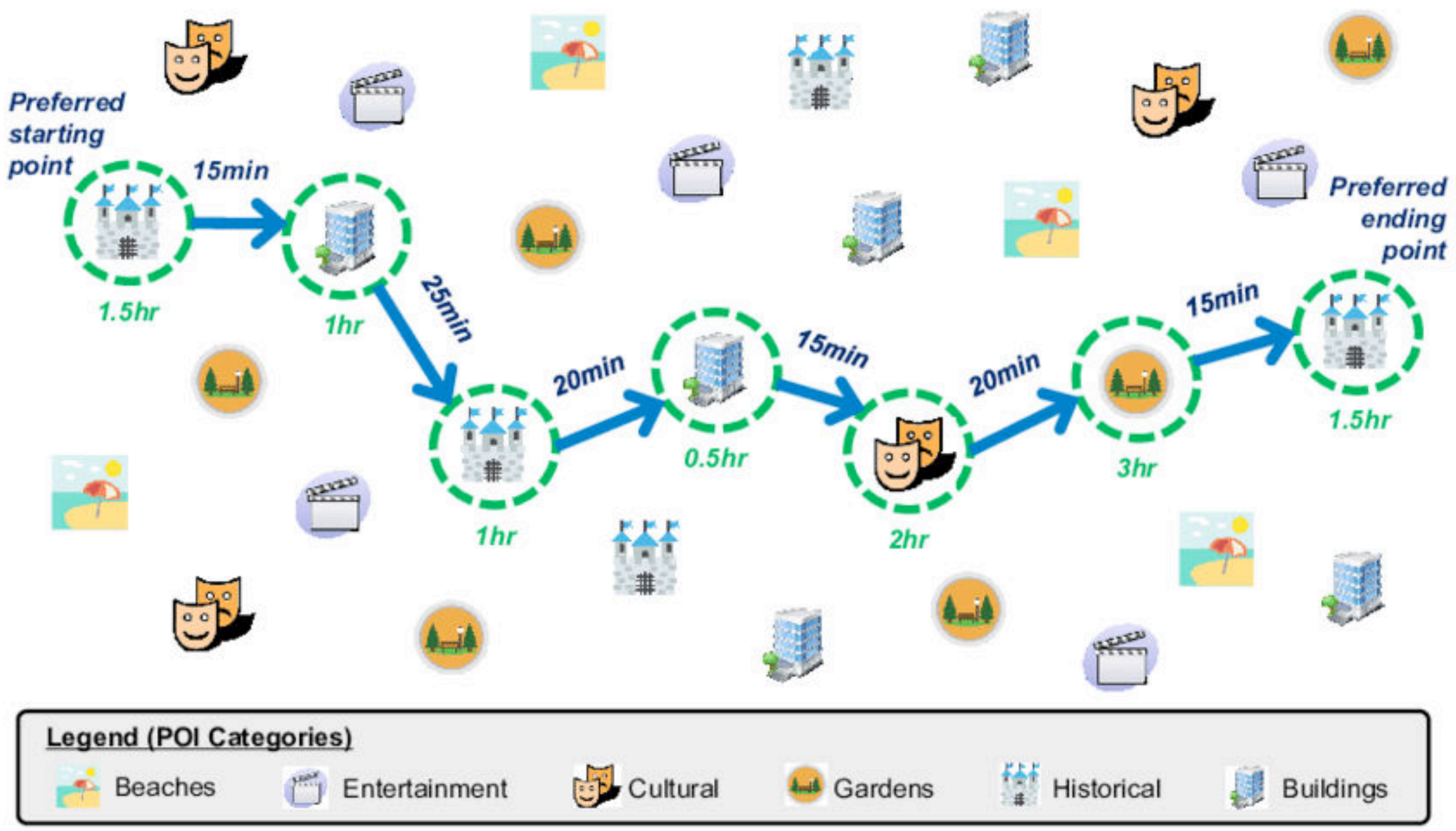
Division of Theoretical Computer Science,
KTH Royal Institute of Technology
argioni@kth.se

motivation



- given a transport network
- nodes are places
- edges are connections
- *source* — starting point
- destination — ending point
- find a short path between source and destination

motivation



- additional requirements
- point of interests
  3 - historical places
  1 - cultural place
  1 - garden / park
  2 - buildings (restaurants)
- *minimum time* to spend at each POI
- transportation links exist at discrete timestamps
- find a travel itinerary

outline

- preliminaries
- problem statement
- algebraic fingerprinting
- solving path-motif problem
- algorithmic results
- experimental results

temporal graph

a temporal graph G is a tuple (V, E), where V is a set of vertices and $E \subseteq V \times V \times [t]$ is set of edges.

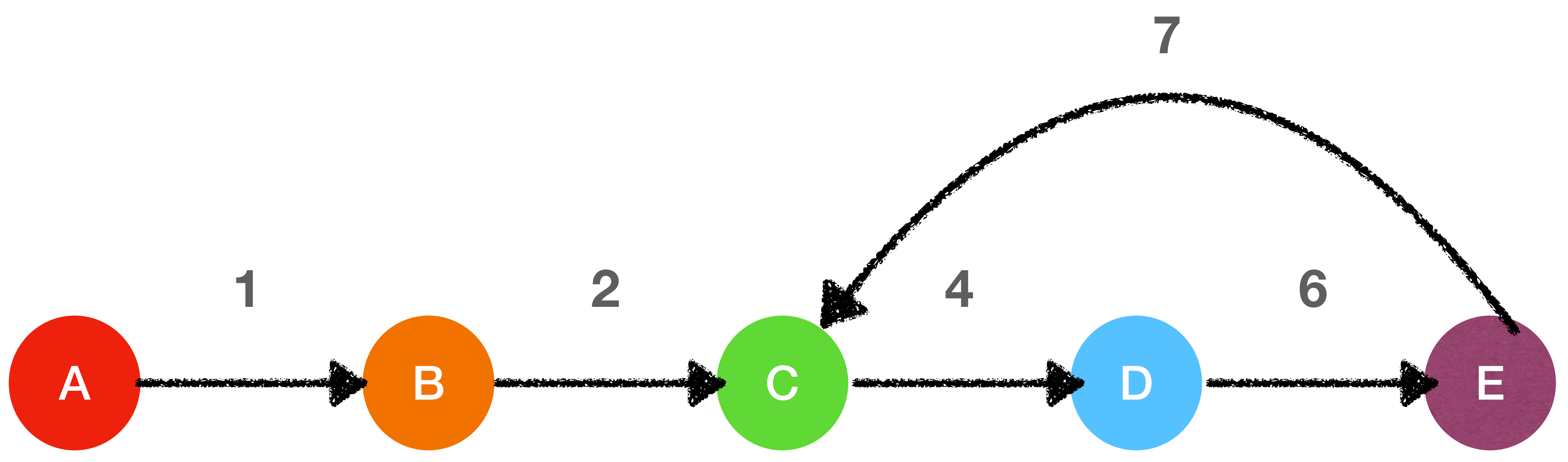
a temporal edge is a tuple (u, v, i), where $u, v \in V$ and $i \in [t]$

temporal walk

given a temporal graph $G = (V, E, \tau)$, a temporal walk

$$W = u_A \; e_{AB,1} \; u_B \; e_{BC,2} \; u_C \; e_{CD,4} \; u_D \; e_{DE,6} \; u_E \; e_{EC,7} \; u_C$$

is an *alternating sequence* of vertices and edges, such that the timestamps of consecutive edges are *(strictly) increasing*



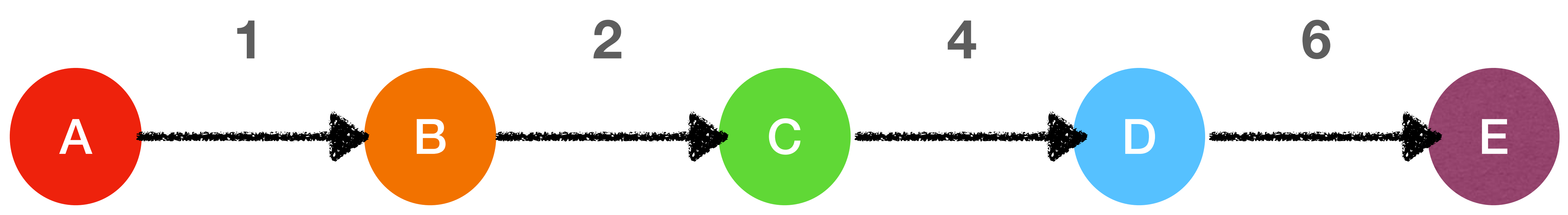
time-respecting (temporal) walk

temporal path

given a temporal graph $G = (V, E, \tau)$, a temporal path

$$P = u_A \ e_{AB,1} \ u_B \ e_{BC,2} \ u_C \ e_{CD,4} \ u_D \ e_{DE,6} \ u_E$$

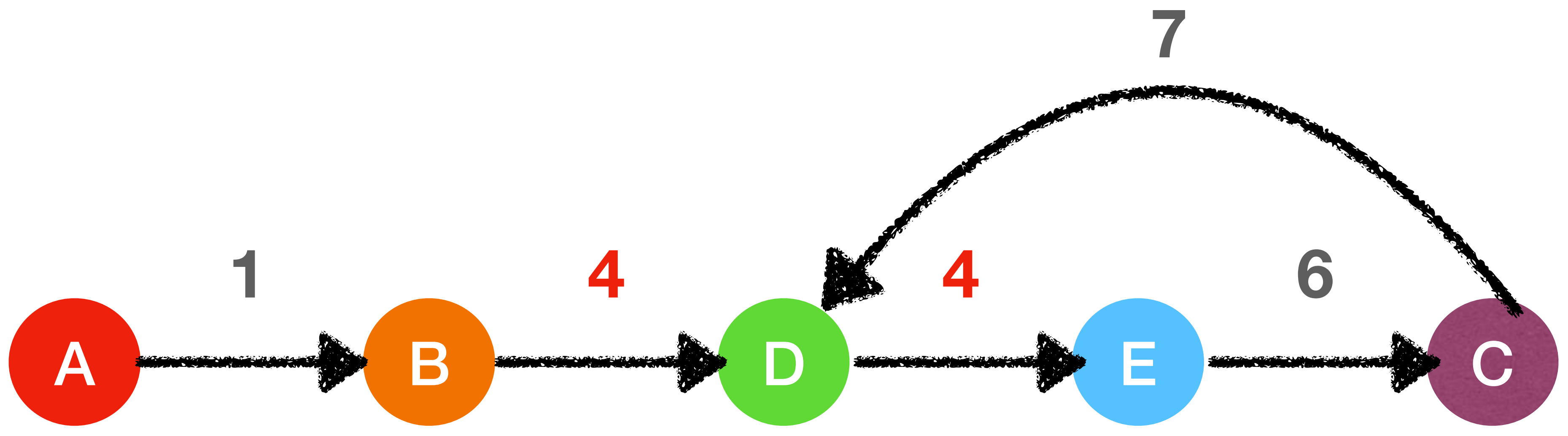
is an alternating sequence of vertices and edges, such that the timestamps on consecutive edges are *(strictly) increasing* and vertices are *not repeated*



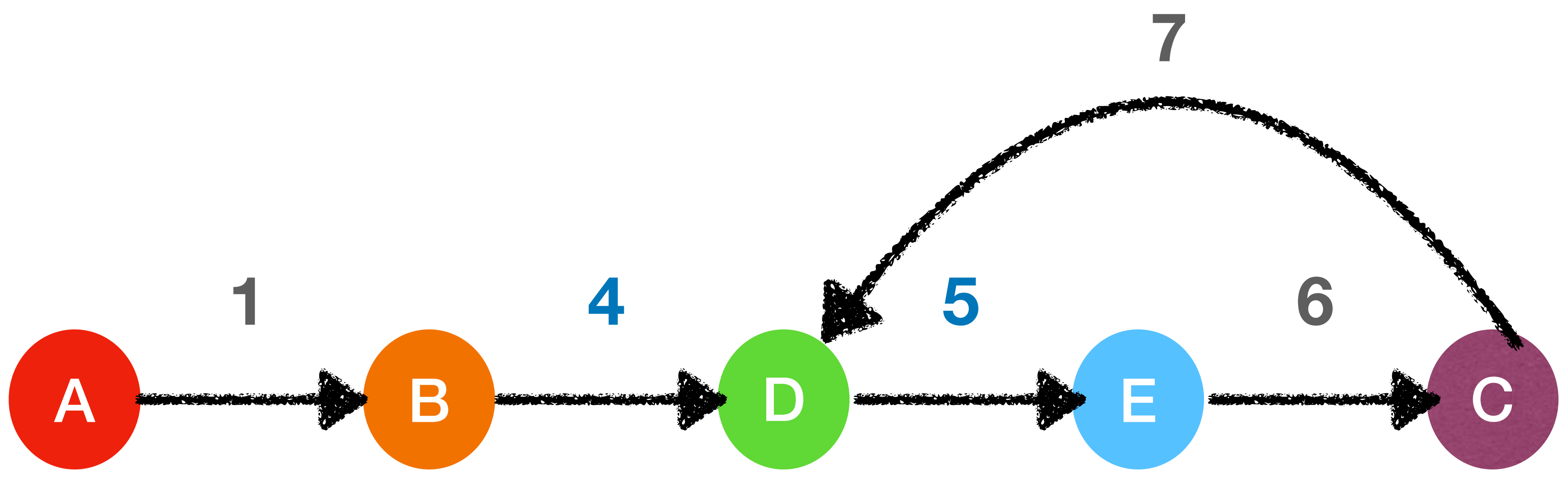
time-respecting (temporal) path

* temporal path and walk variants with non-decreasing timestamps are also studied

strict and non-strict temporal walks



non-strict temporal walks



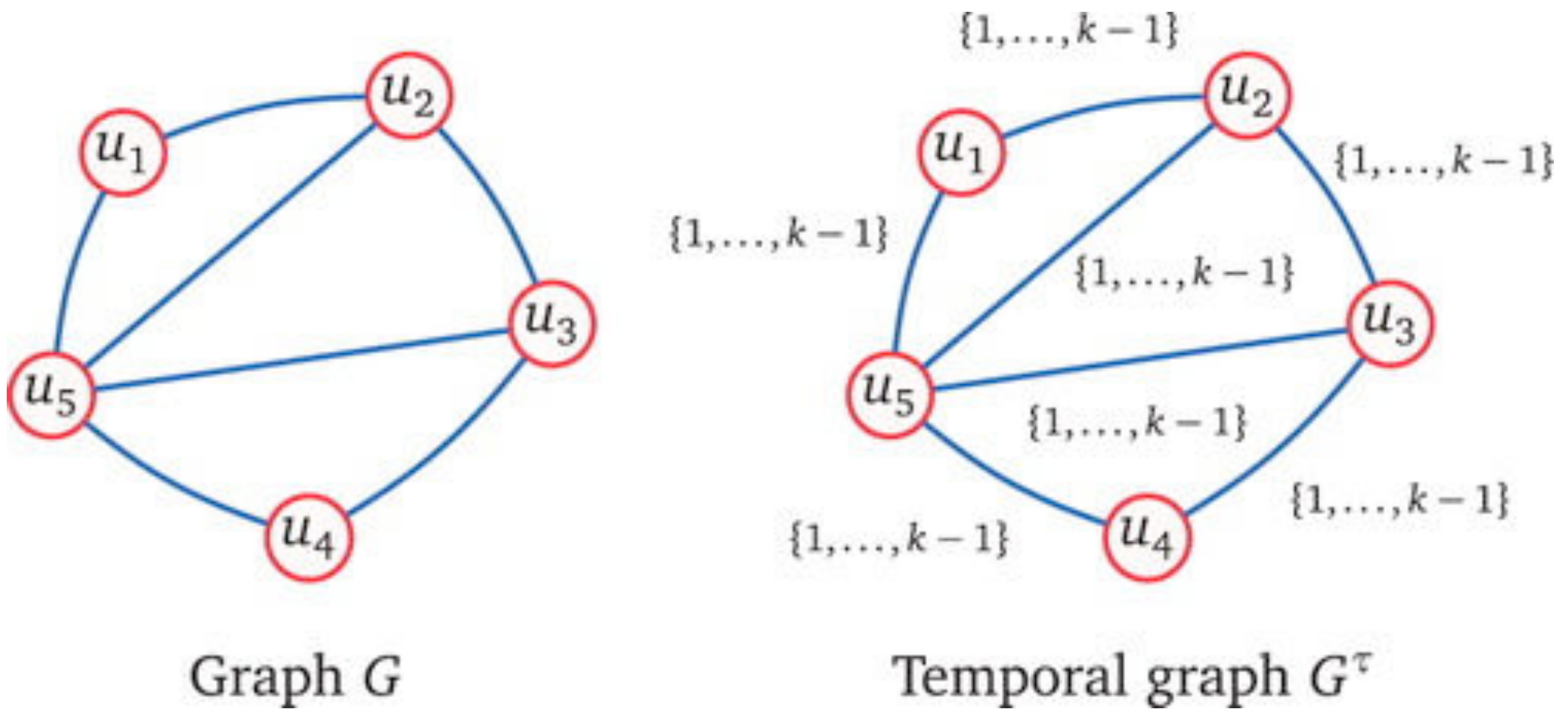
strict temporal walks

- strict walks have increasing timestamps on consecutive edges
- non-strict temporal walks have non-increasing timestamps on consecutive edges
- generalises to paths as well
- our technique works for both settings

temporal path problems

$k-$path problem in temporal graphs

input: given a temporal graph $G = (V, E, \tau)$, and an integer $k \leq |V|$
question: is there a temporal path of length $k - 1$?

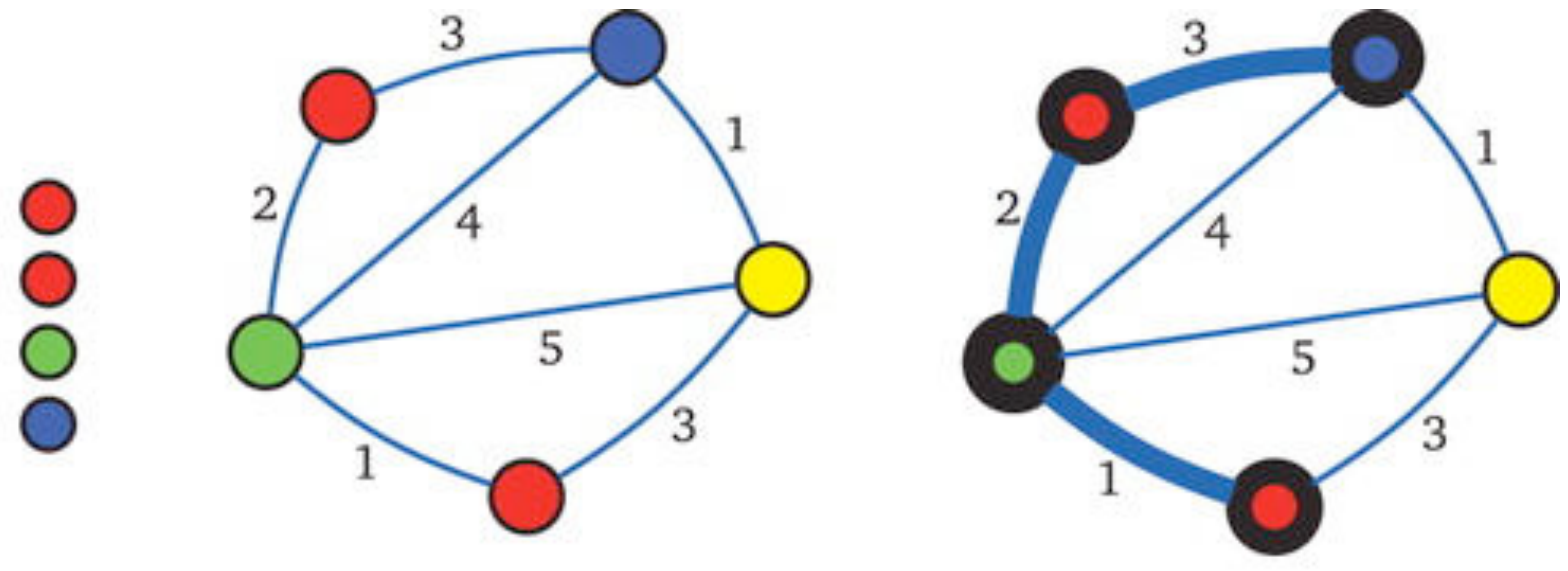


problem is *NP-hard*, hardness follows from k-path problem in static graphs

path motif problem in temporal graphs

input: a temporal graph $G = (V, E)$,
a colouring function $c : V \to [q]$, and
a multiset $M \subseteq [q]$ of colours, $|M| = k$

question: is there a temporal path P such that
the vertex colours of P agree with M?

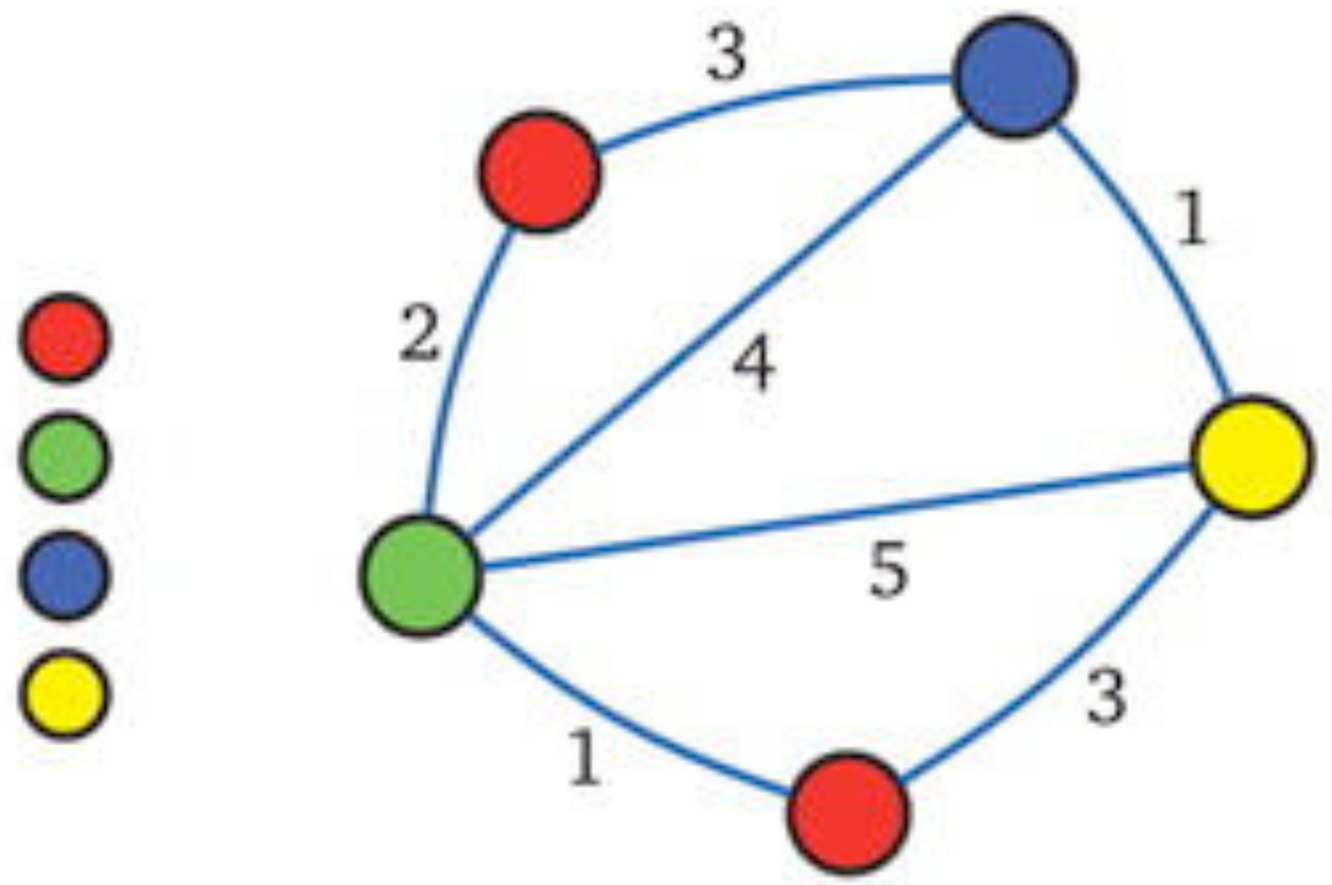


A motif query and a temporal graph

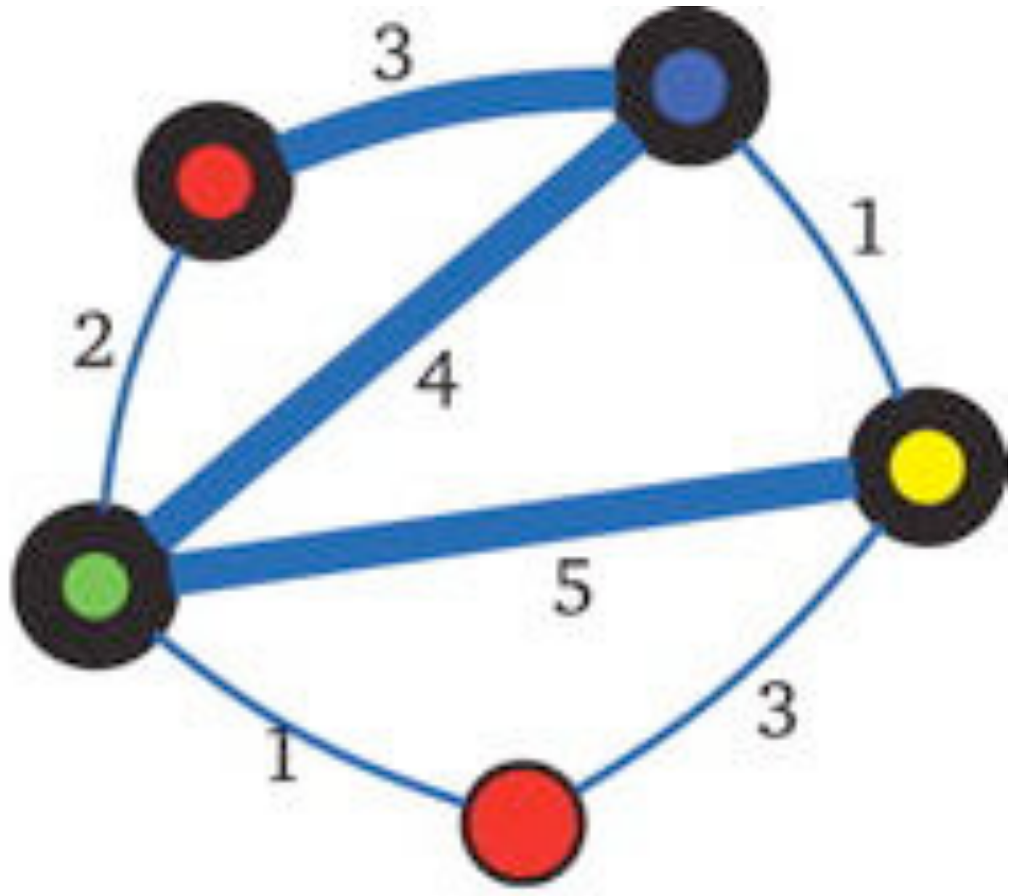
A PathMotif

rainbow path problem in temporal graphs

input: a temporal graph $G = (V, E, \tau)$,
a colouring function $c : V \to [q]$
question: is there a temporal path of length $k - 1$ such that the vertex colours are different?

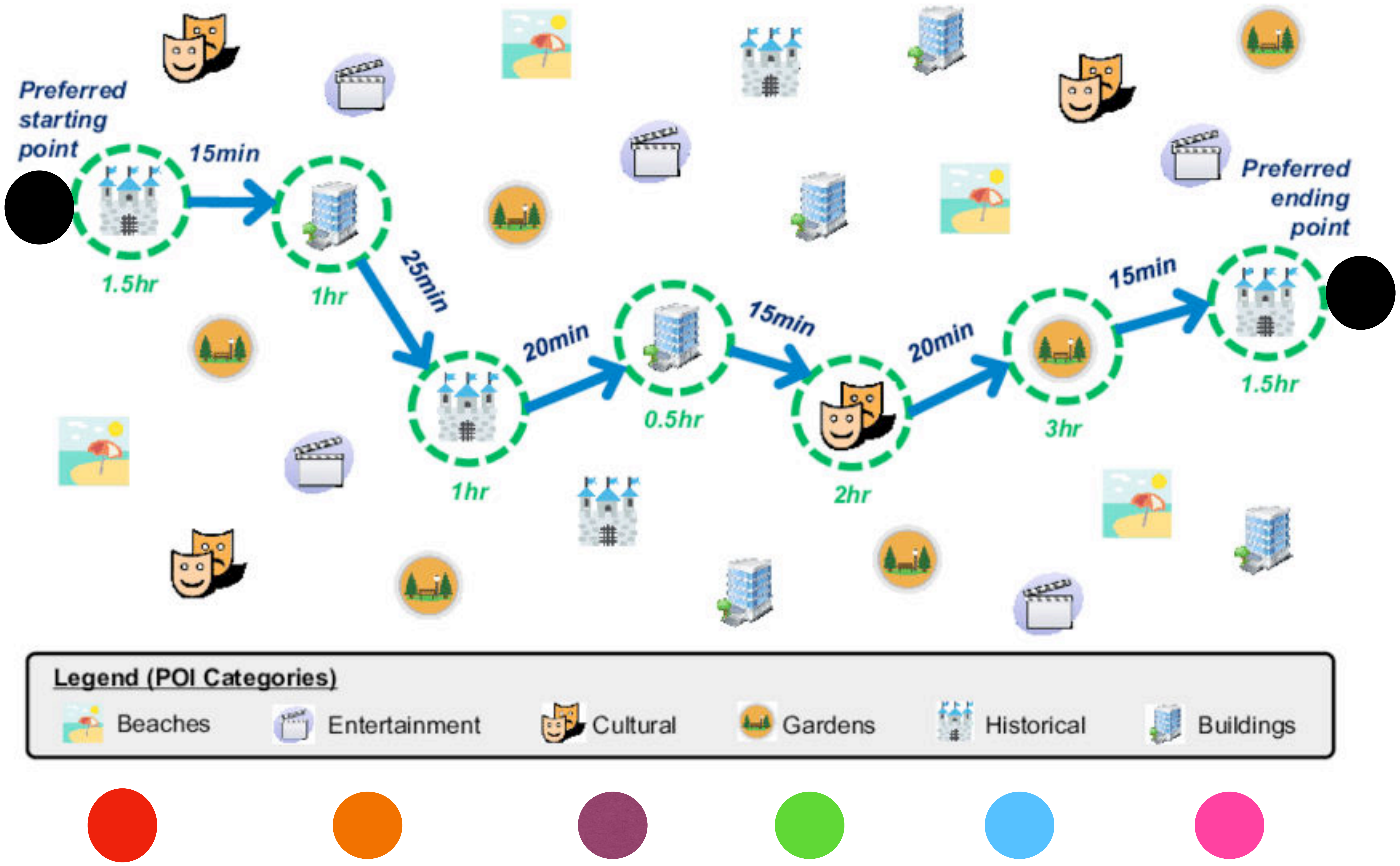


A motif query and a temporal graph



A RainbowPath

motivation



- additional requirements
- point of interests
  3 - historical places
  1 - cultural place
  1 - garden
  2 - buildings (restaurants)
- motif M
- find a path agreeing colours in M

algorithm

preliminaries

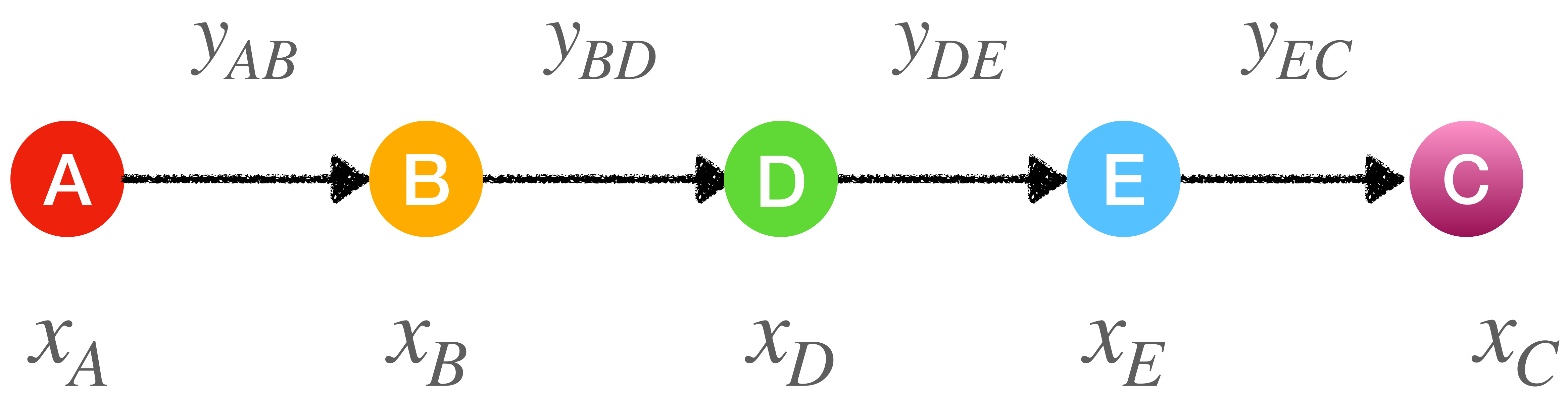
- let $\mathscr{P}$ be a polynomial where each monomial M is of the form $x_1^{f_1} x_2^{f_2} \ldots x_n^{f_n}$
- M is multilinear if $f_1, f_2, \ldots, f_n \in \{0,1\}$, i.e, no variable is repeated
- for example:

  - $\mathscr{P} = x_1 x_2^2 x_{n-1}^{10} x_n^5 + x_3 x_5 x_{n-2} x_n + x_4 x_5 x_6^2 x_{10}$ is a polynomial,

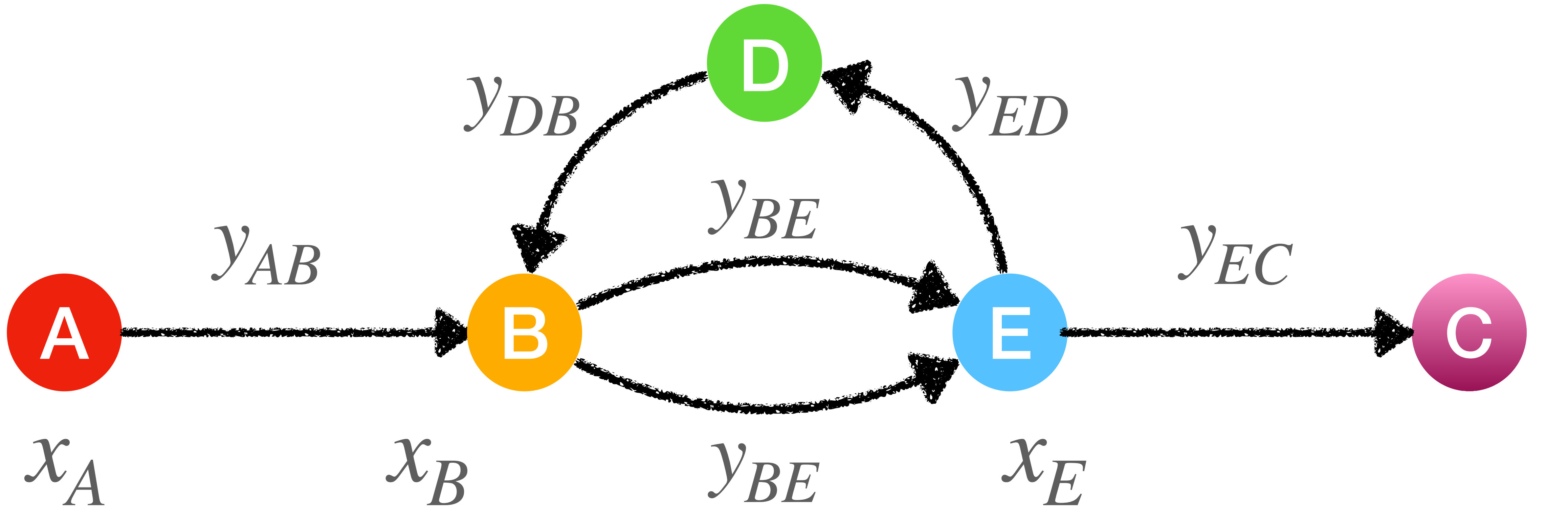
  - monomials are $x_1 x_2^2 x_{n-1}^{10} x_n^5$, $x_3 x_5 x_{n-2} x_n$ and $x_4 x_5 x_6^2 x_{10}$

- multilinear monomial: $x_3 x_5 x_{n-2} x_n$
- not a multilinear monomial: $x_4 x_5 x_6^2 x_{10}$, $x_1 x_2^2 x_{n-1}^{10} x_n^5$

algebraic fingerprinting

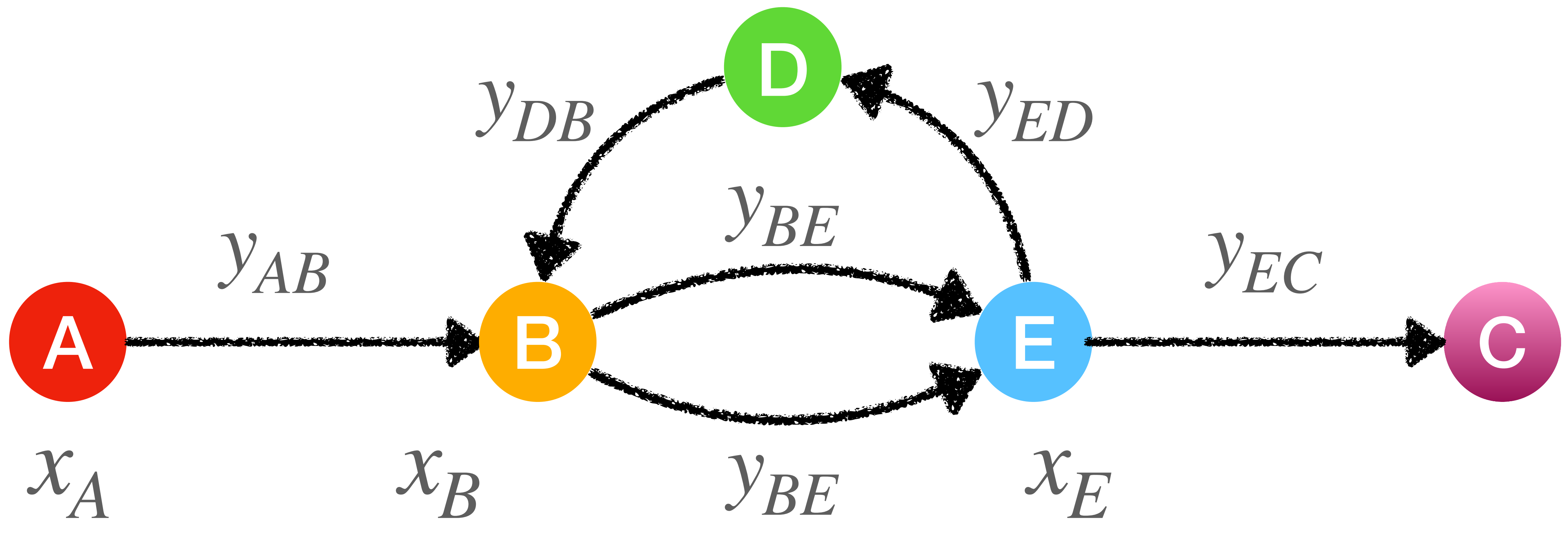


$$x_A\ y_{AB}\ x_B\ y_{BD}\ x_D\ y_{DE}\ x_E\ y_{EC}\ \ x_C$$



$$x_A\ y_{AB}\ x_B\ y_{BE}\ x_E\ y_{ED}\ x_D\ y_{DB}\ x_B\ y_{BE}\ x_E\ y_{EC}\ x_C$$

- represent vertices and edges using variables
- encode a path/walk as a monomial
- assign values to variables at random — Galois field 2^b
- evaluate the monomial (field multiplication)
- If variables are not repeated — evaluates to a non-zero term



algebraic fingerprinting

$$x_A\, y_{AB}\, x_B\, y_{BE}\, x_E\, y_{ED}\, x_D\, y_{DB}\, x_B\, y_{BE}\, x_E\, y_{EC}\, x_C$$

x_A	0	1	0	1	0	0	1	1	1	0	1	1	1	0	0	0
y_{AB}	0	0	1	1	1	0	1	0	0	0	0	1	1	1	0	1
x_B	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1
y_{BE}	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
x_E	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0	0
y_{ED}	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1
x_D	0	0	1	1	0	1	0	0	1	0	1	1	1	1	0	0
y_{DB}	1	0	1	0	0	0	0	1	1	1	0	0	1	0	1	1
x_B	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	1
y_{BE}	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
x_E	1	1	0	0	0	0	0	1	1	1	1	1	1	1	0	0
y_{EC}	0	0	0	0	0	1	1	1	1	0	1	0	1	1	0	0
x_C	1	1	1	0	1	0	1	0	1	0	0	0	0	0	1	1

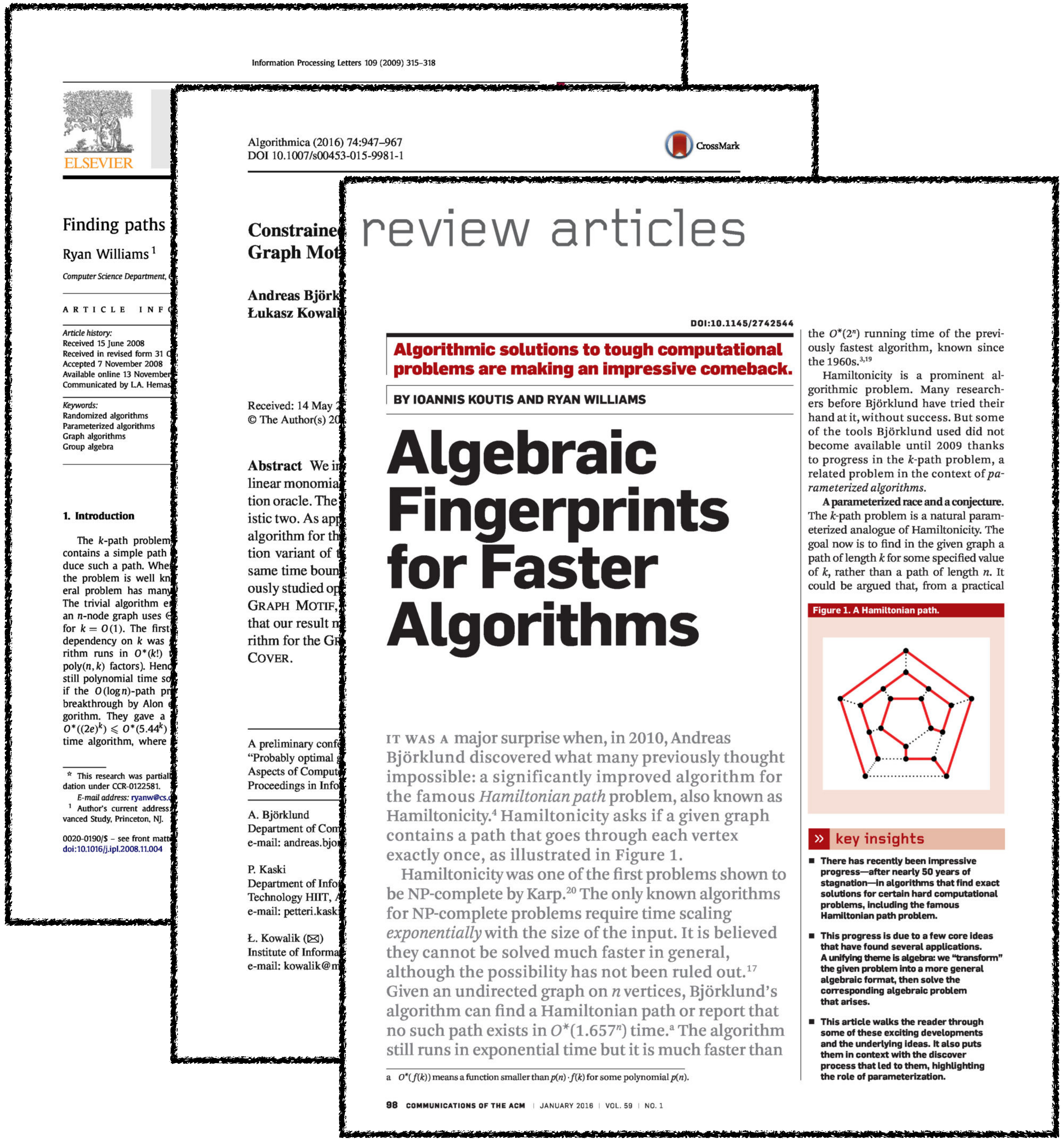
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- evaluates to zero term — if at least one variable is repeated
- encode all walks in the graph as a polynomial
- repeat evaluation with 2^{ℓ} random assignments
- false-negative probability is $\dfrac{2\ell - 1}{2^b}$
- no false-positives

* ℓ is number of vertices in path/walk

algorithm overview

- given a problem instance (say a graph) and a pattern to find (say a path)
- encode the problem as a polynomial such that there exists a multilinear monomial if and only if the desired pattern is present
- evaluate the polynomial using random substitutions
- if one of the substitutions evaluate to a non-zero term, then the desired pattern is present in the graph
- repeat the random substitution for 2^ℓ iterations


Information Processing Letters 109 (2009) 315–318

ELSEVIER

Finding paths

Ryan Williams [1]

Computer Science Department,

ARTICLE INFO

Article history:
Received 15 June 2008
Received in revised form 31
Accepted 7 November 2008
Available online 13 November
Communicated by L.A. Hemas

Keywords:
Randomized algorithms
Parameterized algorithms
Graph algorithms
Group algebra

1. Introduction

The k-path problem contains a simple path duce such a path. When the problem is well kn eral problem has many The trivial algorithm e an n-node graph uses for $k = O(1)$. The first dependency on k was rithm runs in $O^*(k!)$ poly(n,k) factors). Henc still polynomial time so if the $O(\log n)$-path pr breakthrough by Alon gorithm. They gave a $O^*((2e)^k) \leqslant O^*(5.44^k)$ time algorithm, where

* This research was partial dation under CCR-0122581.
E-mail address: ryanw@cs.
[1] Author's current address vanced Study, Princeton, NJ.

0020-0190/$ – see front matt
doi:10.1016/j.ipl.2008.11.004

Algorithmica (2016) 74:947–967
DOI 10.1007/s00453-015-9981-1

CrossMark

Constrained
Graph Mot

Andreas Björk
Łukasz Kowali

Received: 14 May 2
© The Author(s) 20

Abstract We in linear monomia tion oracle. The istic two. As app algorithm for th tion variant of t same time boun ously studied op Graph Motif, that our result m rithm for the Gr Cover.

A preliminary conf "Probably optimal Aspects of Comput Proceedings in Info

A. Björklund
Department of Com
e-mail: andreas.bjor

P. Kaski
Department of Info
Technology HIIT, A
e-mail: petteri.kask

Ł. Kowalik (✉)
Institute of Informa
e-mail: kowalik@m

review articles

DOI:10.1145/2742544

Algorithmic solutions to tough computational problems are making an impressive comeback.

BY IOANNIS KOUTIS AND RYAN WILLIAMS

Algebraic Fingerprints for Faster Algorithms

IT WAS A major surprise when, in 2010, Andreas Björklund discovered what many previously thought impossible: a significantly improved algorithm for the famous *Hamiltonian path* problem, also known as Hamiltonicity.[4] Hamiltonicity asks if a given graph contains a path that goes through each vertex exactly once, as illustrated in Figure 1.

Hamiltonicity was one of the first problems shown to be NP-complete by Karp.[20] The only known algorithms for NP-complete problems require time scaling *exponentially* with the size of the input. It is believed they cannot be solved much faster in general, although the possibility has not been ruled out.[17] Given an undirected graph on n vertices, Björklund's algorithm can find a Hamiltonian path or report that no such path exists in $O^*(1.657^n)$ time.[a] The algorithm still runs in exponential time but it is much faster than the $O^*(2^n)$ running time of the previously fastest algorithm, known since the 1960s.[3,19]

Hamiltonicity is a prominent algorithmic problem. Many researchers before Björklund have tried their hand at it, without success. But some of the tools Björklund used did not become available until 2009 thanks to progress in the k-path problem, a related problem in the context of *parameterized algorithms.*

A parameterized race and a conjecture. The k-path problem is a natural parameterized analogue of Hamiltonicity. The goal now is to find in the given graph a path of length k for some specified value of k, rather than a path of length n. It could be argued that, from a practical

Figure 1. A Hamiltonian path.

» key insights

- There has recently been impressive progress—after nearly 50 years of stagnation—in algorithms that find exact solutions for certain hard computational problems, including the famous Hamiltonian path problem.
- This progress is due to a few core ideas that have found several applications. A unifying theme is algebra: we "transform" the given problem into a more general algebraic format, then solve the corresponding algebraic problem that arises.
- This article walks the reader through some of these exciting developments and the underlying ideas. It also puts them in context with the discover process that led to them, highlighting the role of parameterization.

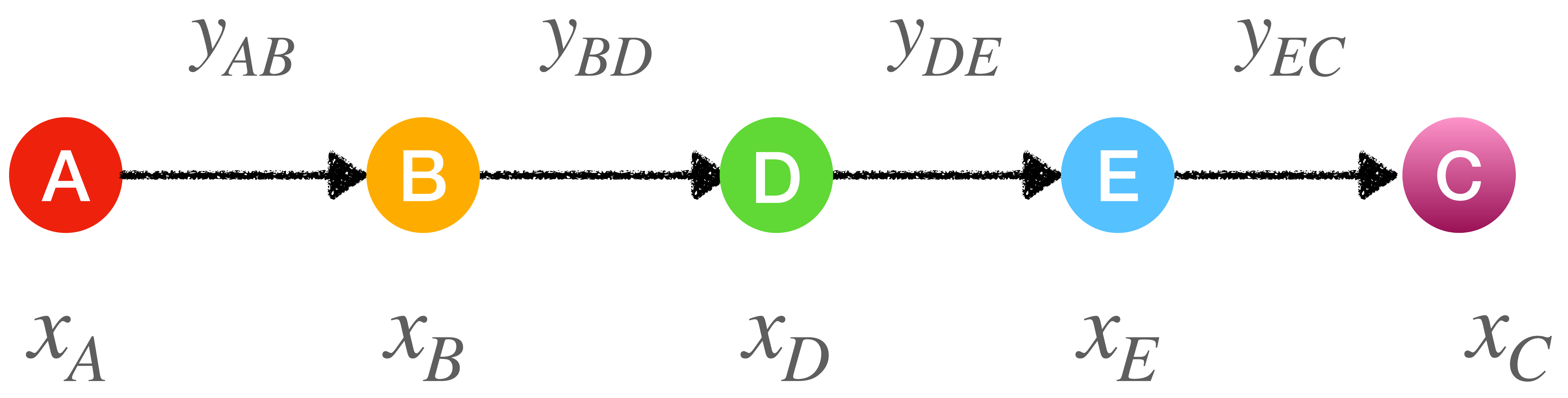
a $O^*(f(k))$ means a function smaller than $p(n) \cdot f(k)$ for some polynomial $p(n)$.

98 COMMUNICATIONS OF THE ACM | JANUARY 2016 | VOL. 59 | NO. 1

algebraic fingerprinting

- represent paths and walks as monomials
- detect if a monomial has a repeated variable
- theoretically best-known results for graphs problems such as Hamiltonian path, k-path, graph motifs, …
- theoretically best-known results for temporal graph problems such as colourful path, path motifs, restless path, …

algorithm



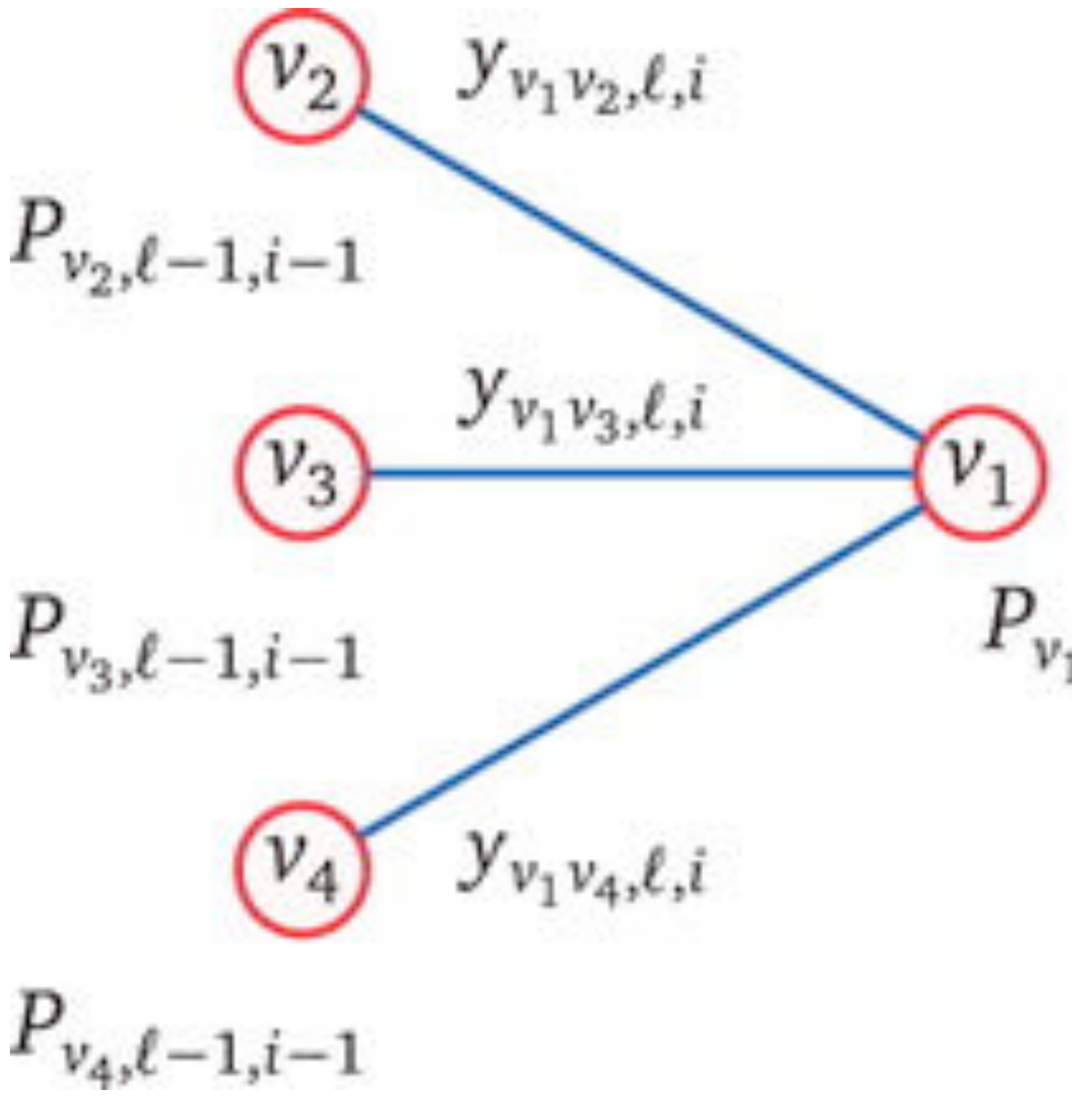
$$x_A \; y_{AB} \; x_B \; y_{BD} \; x_D \; y_{DE} \; x_E \; y_{EC} \; x_C$$

path-length : $k - 1$ (4)

monomial-size : $2k - 1$ (9)

- generate all walks of length $k - 1$ using a polynomial encoding P
- check if there exists a multilinear monomial of size $2k - 1$ in P
- evaluate P with random substitution for the variables
- there exists a path if and only if there exists a multilinear monomial

generating temporal walks

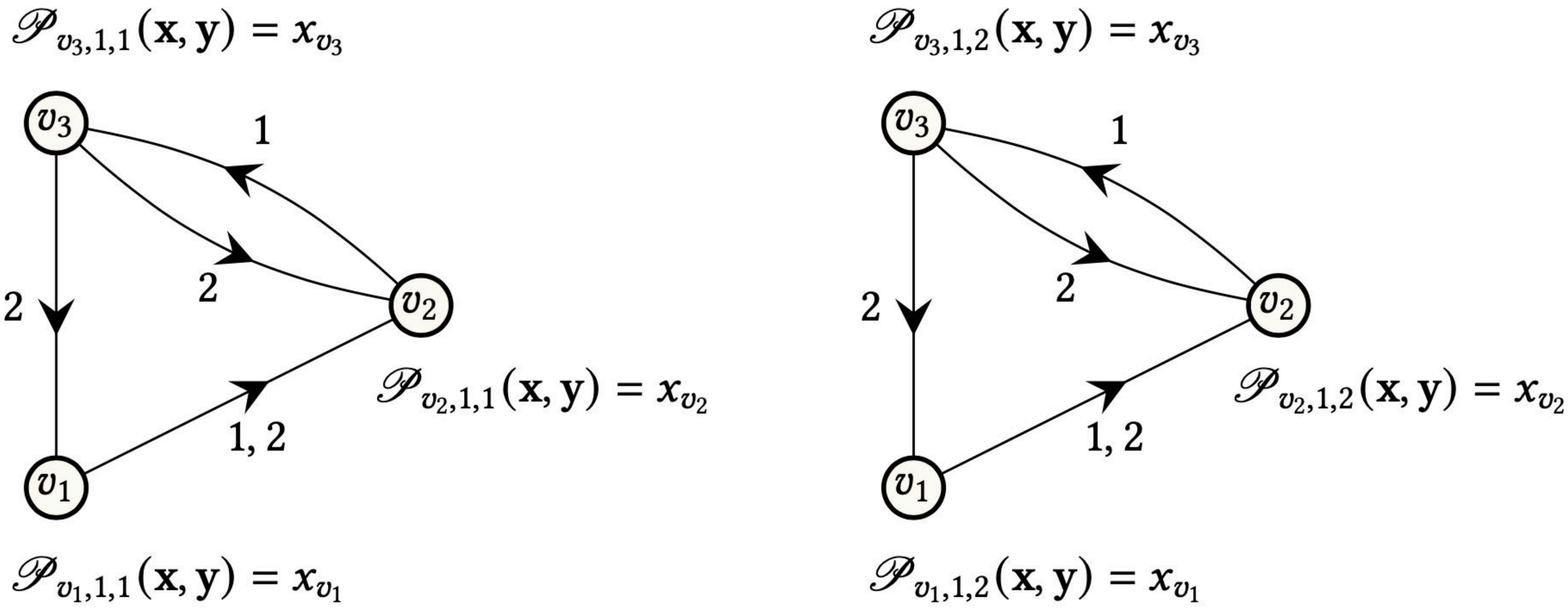


$$\begin{aligned} P_{v_1,\ell,i} = {} & x_{v_1} y_{v_1 v_2,\ell-1,i} P_{v_2,\ell-1,i-1} + \\ & x_{v_1} y_{v_1 v_3,\ell-1,i} P_{v_3,\ell-1,i-1} + \\ & x_{v_1} y_{v_1 v_4,\ell-1,i} P_{v_4,\ell-1,i-1} + P_{v_1,\ell,i-1} \end{aligned}$$

$$P_{u,\ell,i} = x_u \sum_{v \in N_i(u)} y_{uv,\ell-1,i} P_{v,\ell-1,i-1} + P_{u,\ell,i-1}$$

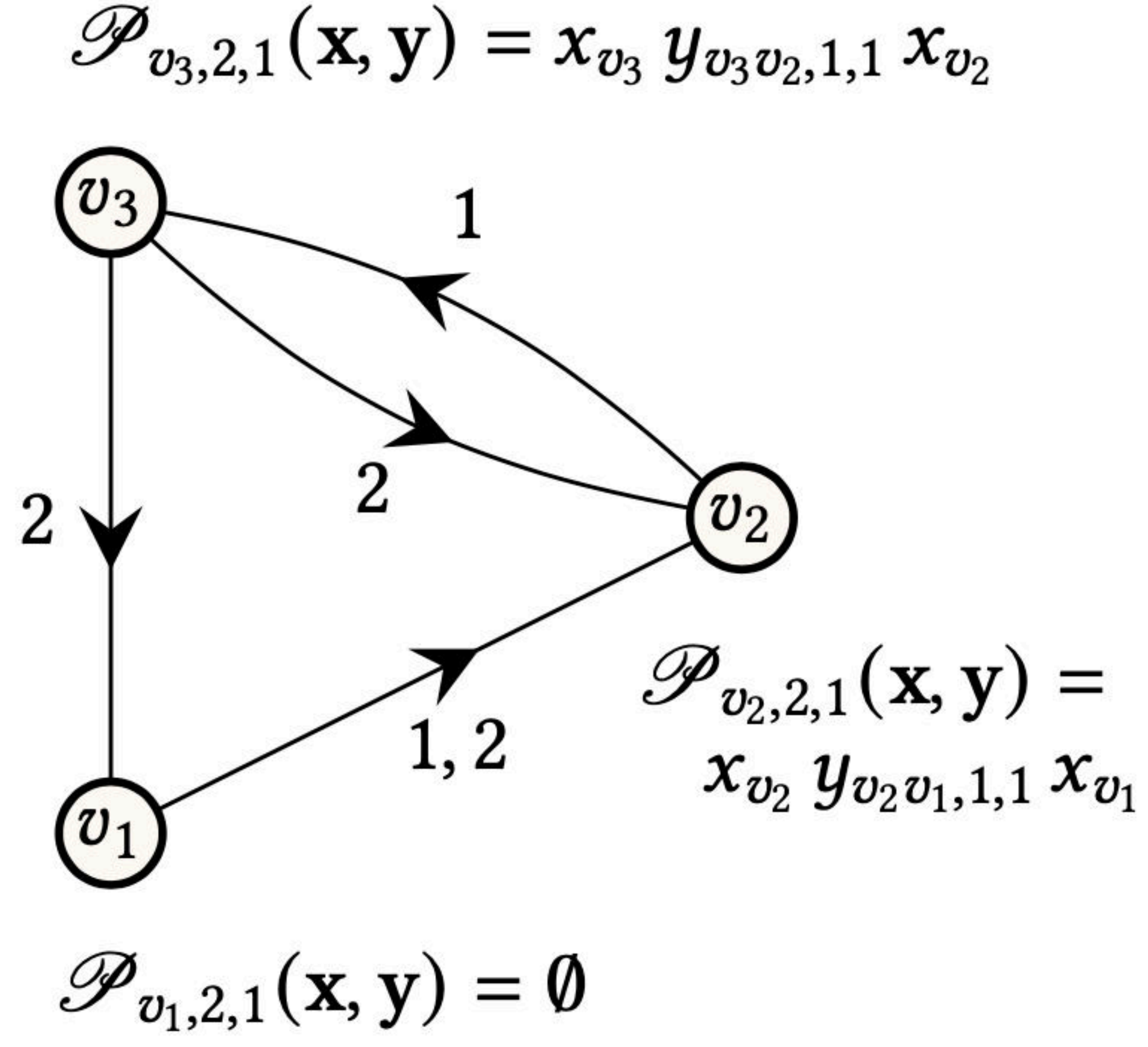
- $\mathscr{P}_{u,\ell,i}$ encoding of all walks ending at vertex u, length $\ell - 1$, and at latest time i
- v_1, v_2 are neighbours of u at time i
- only walk to u if we have reached v_1, v_2 at latest time i
- x_u : variable for vertex u
- $y_{vu,\ell,j}$: variable for edge (v, u, j) at position ℓ in the walk

generating temporal walks (length = 0)

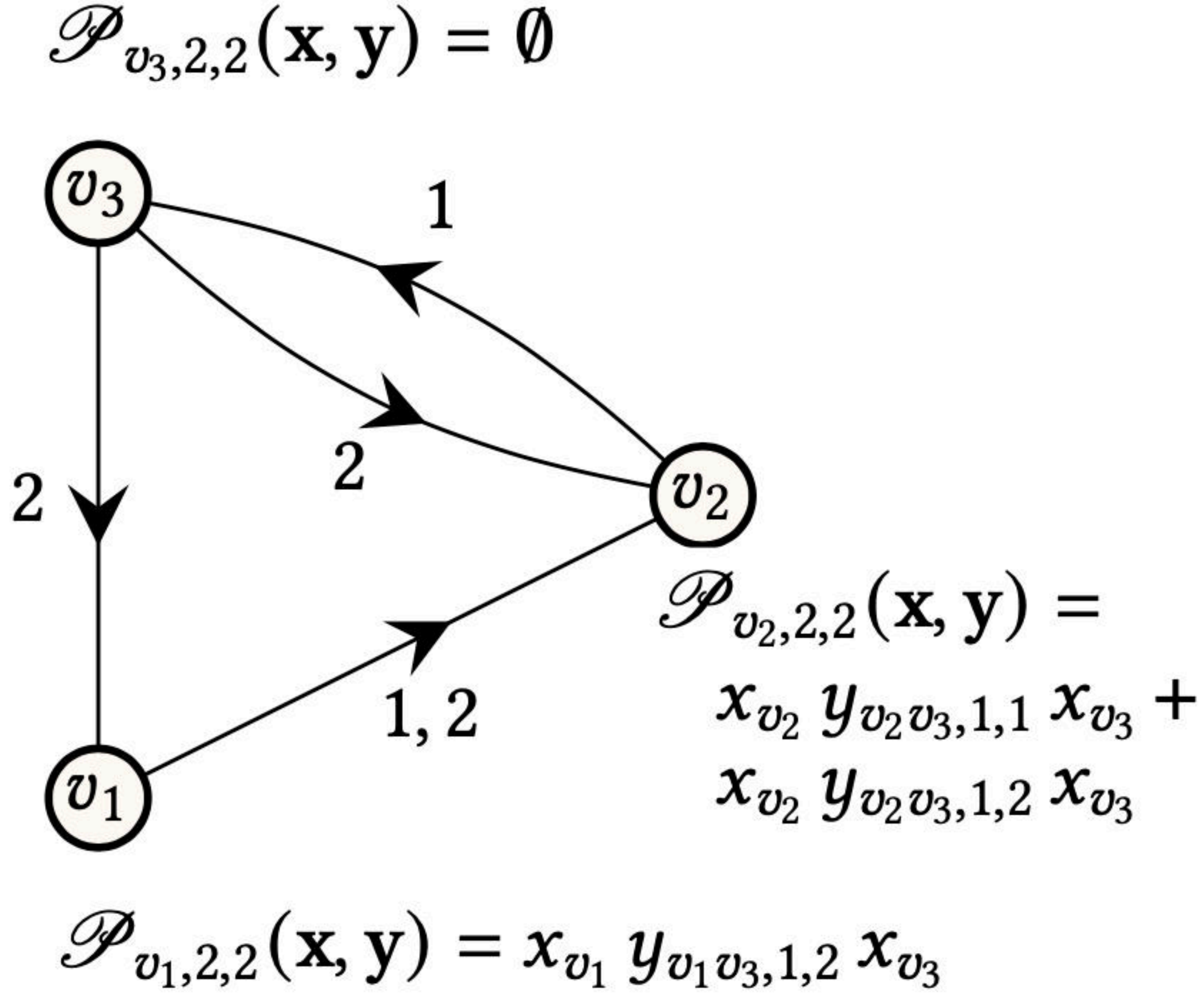


(a) $\ell = 1, i = 1$ (b) $\ell = 1, i = 2$

generating temporal walks (length = 1)

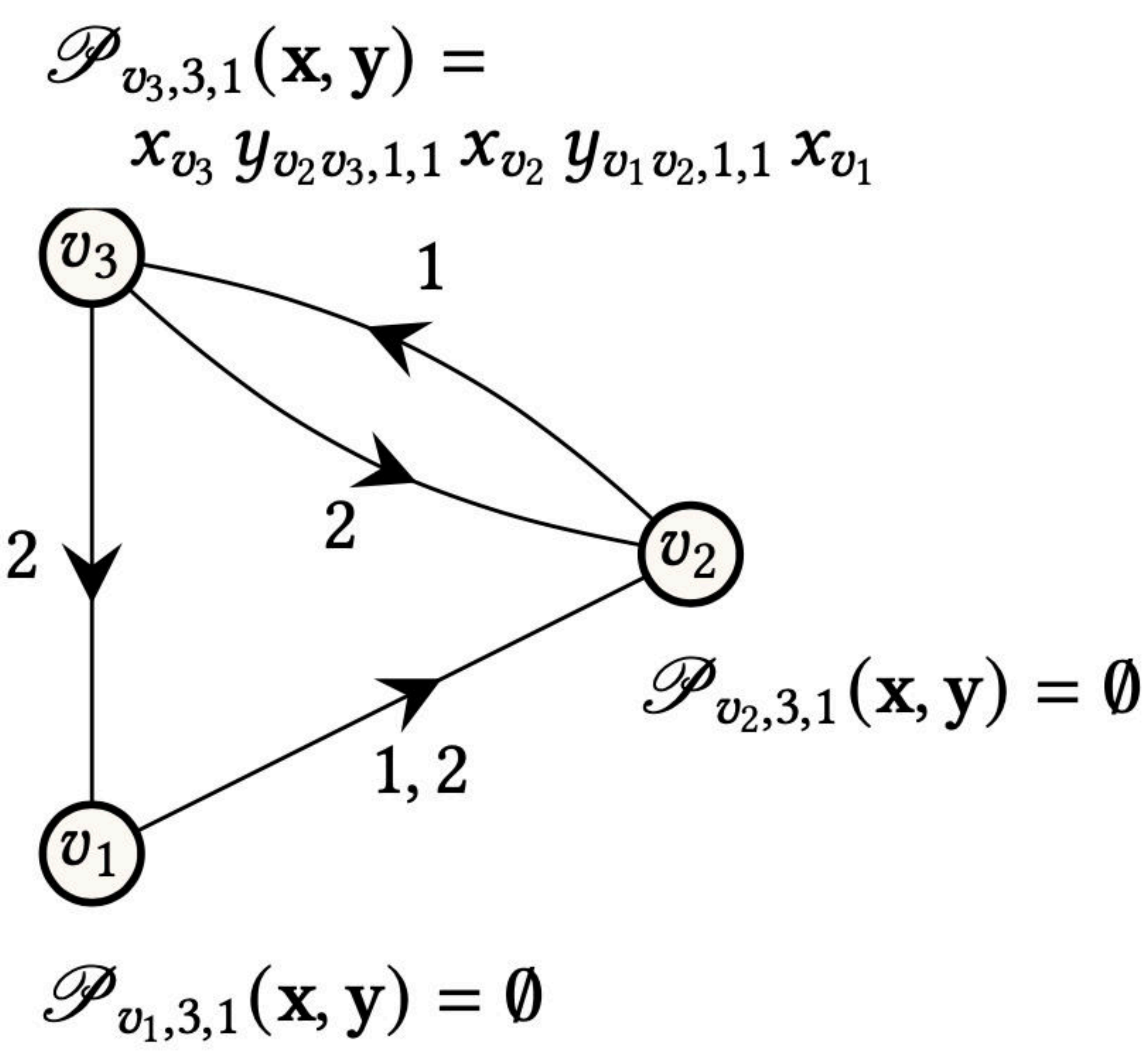


(c) $\ell = 2, i = 1$

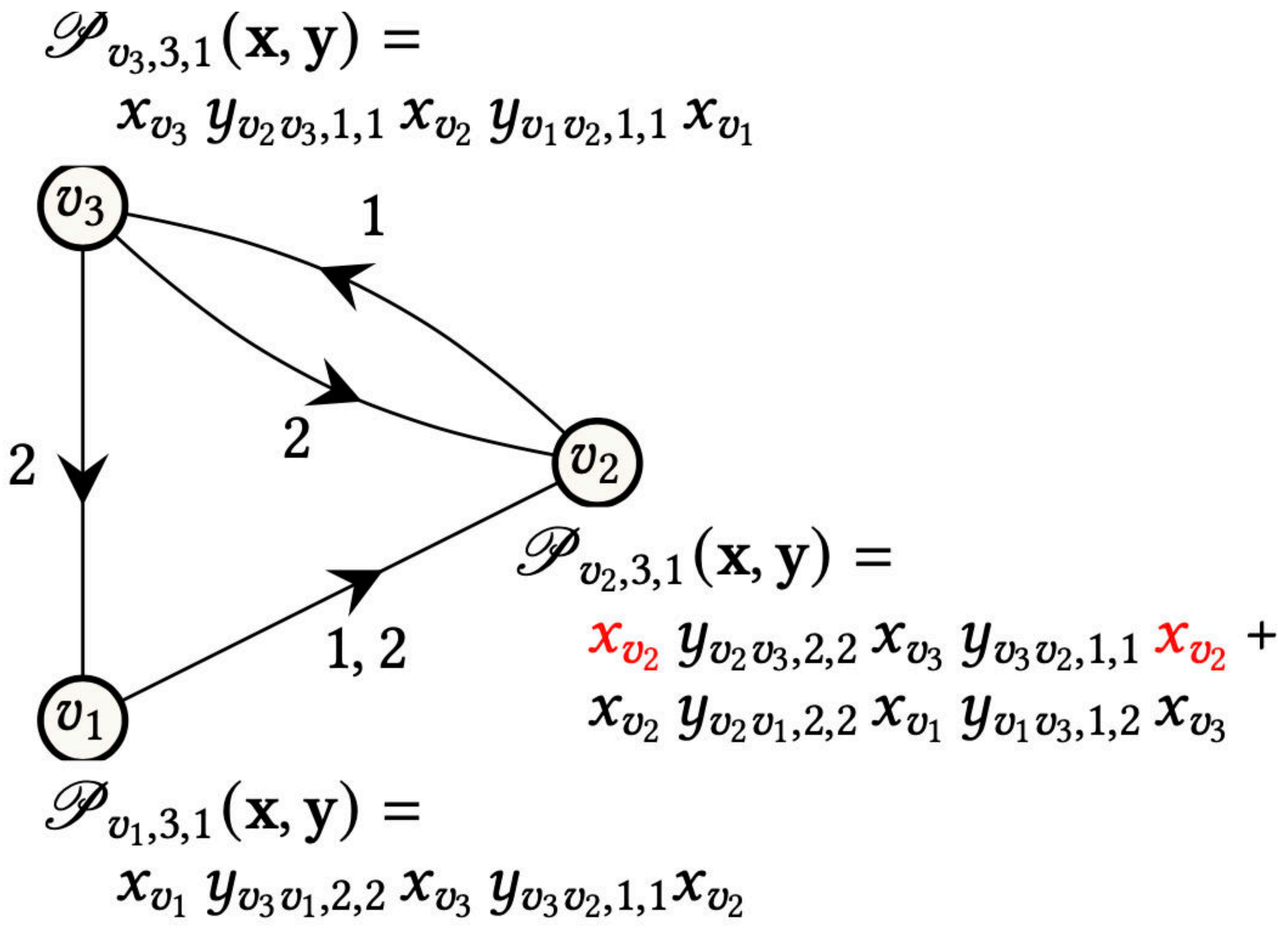


(d) $\ell = 2, i = 2$

generating temporal walks (length = 2)

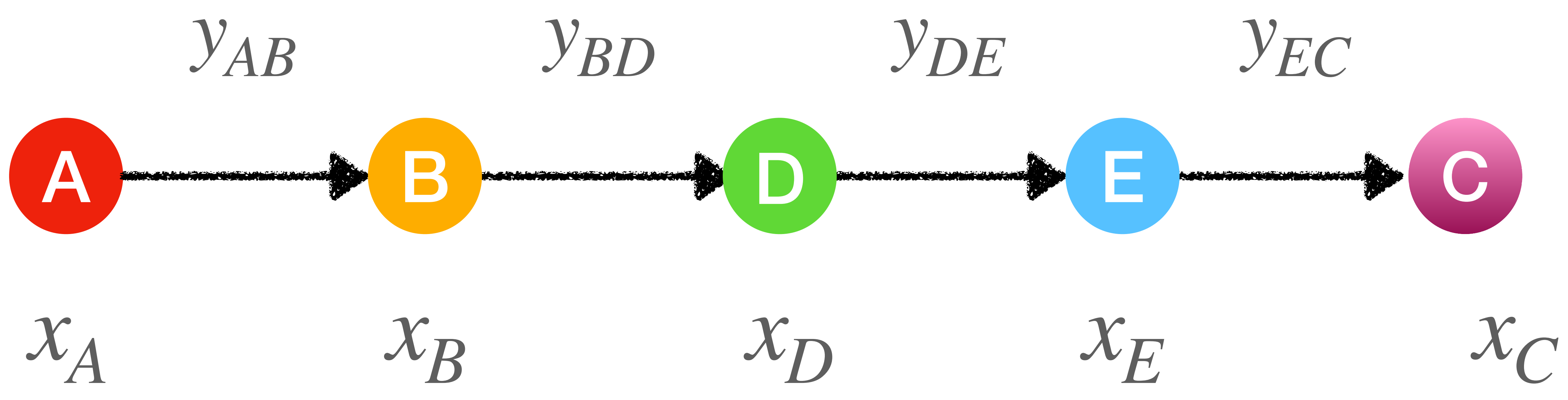


(e) $\ell = 3, i = 1$



(f) $\ell = 3, i = 2$

algorithm



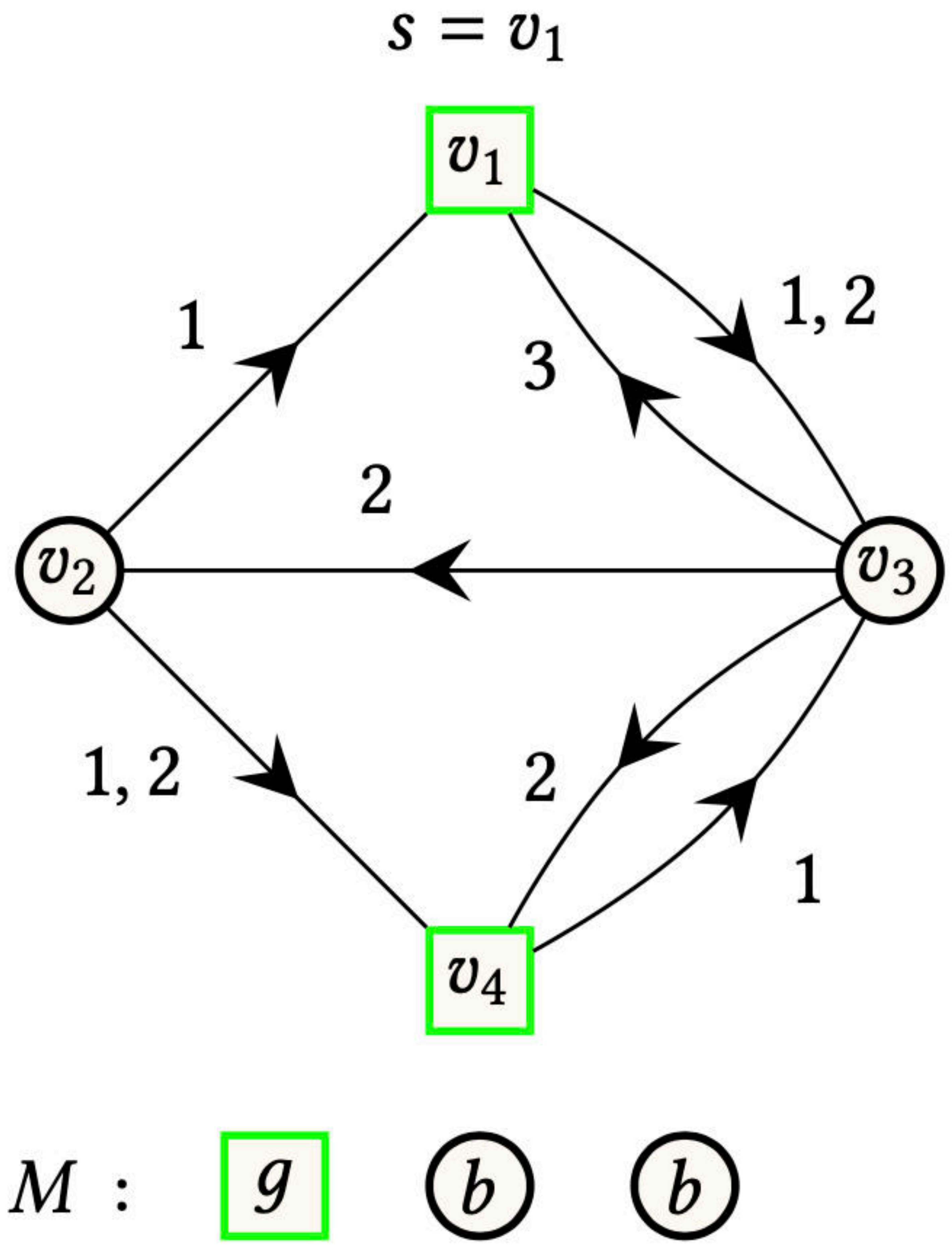
$x_A \; y_{AB} \; x_B \; y_{BD} \; x_D \; y_{DE} \; x_E \; y_{EC} \; x_C$

path-length : $k - 1$ (4)

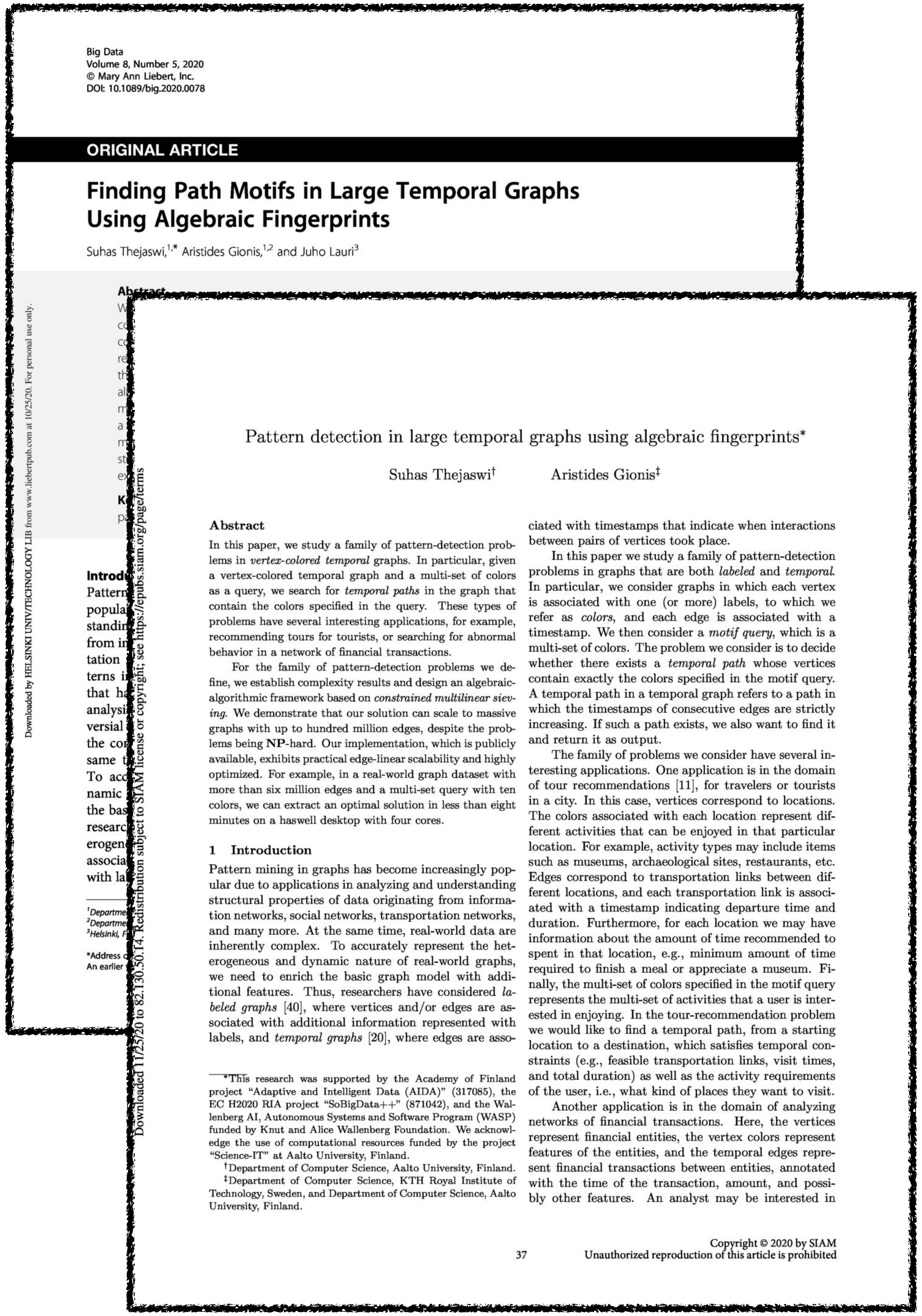
monomial-size : $2k - 1$ (9)

- generate all restless walks of length $k - 1$ using a polynomial encoding P
- check if there exists a multilinear monomial of size $2k - 1$ in P
- evaluate P with random substitution for the variables
- there exists a restless path if and only if there exists a multilinear monomial

vertex color constraints



- given a graph and a multiset of colors
- similarly, we can introduce new variable to restrict the color on vertices
- generate a polynomial encoding of walks
- evaluate the polynomial to check if there exists a path which agree with multiset of colors in M
- constrained-multilinear sieving
- Bjorklund et al. (STACS 2013, Algorithmica)



Big Data
Volume 8, Number 5, 2020
© Mary Ann Liebert, Inc.
DOI: 10.1089/big.2020.0078

ORIGINAL ARTICLE

Finding Path Motifs in Large Temporal Graphs Using Algebraic Fingerprints

Suhas Thejaswi,[1,*] Aristides Gionis,[1,2] and Juho Lauri[3]

Pattern detection in large temporal graphs using algebraic fingerprints*

Suhas Thejaswi† Aristides Gionis‡

Abstract

In this paper, we study a family of pattern-detection problems in *vertex-colored temporal* graphs. In particular, given a vertex-colored temporal graph and a multi-set of colors as a query, we search for *temporal paths* in the graph that contain the colors specified in the query. These types of problems have several interesting applications, for example, recommending tours for tourists, or searching for abnormal behavior in a network of financial transactions.

For the family of pattern-detection problems we define, we establish complexity results and design an algebraic-algorithmic framework based on *constrained multilinear sieving*. We demonstrate that our solution can scale to massive graphs with up to hundred million edges, despite the problems being **NP**-hard. Our implementation, which is publicly available, exhibits practical edge-linear scalability and highly optimized. For example, in a real-world graph dataset with more than six million edges and a multi-set query with ten colors, we can extract an optimal solution in less than eight minutes on a haswell desktop with four cores.

1 Introduction

Pattern mining in graphs has become increasingly popular due to applications in analyzing and understanding structural properties of data originating from information networks, social networks, transportation networks, and many more. At the same time, real-world data are inherently complex. To accurately represent the heterogeneous and dynamic nature of real-world graphs, we need to enrich the basic graph model with additional features. Thus, researchers have considered *labeled graphs* [40], where vertices and/or edges are associated with additional information represented with labels, and *temporal graphs* [20], where edges are associated with timestamps that indicate when interactions between pairs of vertices took place.

In this paper we study a family of pattern-detection problems in graphs that are both *labeled* and *temporal*. In particular, we consider graphs in which each vertex is associated with one (or more) labels, to which we refer as *colors*, and each edge is associated with a timestamp. We then consider a *motif query*, which is a multi-set of colors. The problem we consider is to decide whether there exists a *temporal path* whose vertices contain exactly the colors specified in the motif query. A temporal path in a temporal graph refers to a path in which the timestamps of consecutive edges are strictly increasing. If such a path exists, we also want to find it and return it as output.

The family of problems we consider have several interesting applications. One application is in the domain of tour recommendations [11], for travelers or tourists in a city. In this case, vertices correspond to locations. The colors associated with each location represent different activities that can be enjoyed in that particular location. For example, activity types may include items such as museums, archaeological sites, restaurants, etc. Edges correspond to transportation links between different locations, and each transportation link is associated with a timestamp indicating departure time and duration. Furthermore, for each location we may have information about the amount of time recommended to spent in that location, e.g., minimum amount of time required to finish a meal or appreciate a museum. Finally, the multi-set of colors specified in the motif query represents the multi-set of activities that a user is interested in enjoying. In the tour-recommendation problem we would like to find a temporal path, from a starting location to a destination, which satisfies temporal constraints (e.g., feasible transportation links, visit times, and total duration) as well as the activity requirements of the user, i.e., what kind of places they want to visit.

Another application is in the domain of analyzing networks of financial transactions. Here, the vertices represent financial entities, the vertex colors represent features of the entities, and the temporal edges represent financial transactions between entities, annotated with the time of the transaction, amount, and possibly other features. An analyst may be interested in

*This research was supported by the Academy of Finland project "Adaptive and Intelligent Data (AIDA)" (317085), the EC H2020 RIA project "SoBigData++" (871042), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation. We acknowledge the use of computational resources funded by the project "Science-IT" at Aalto University, Finland.

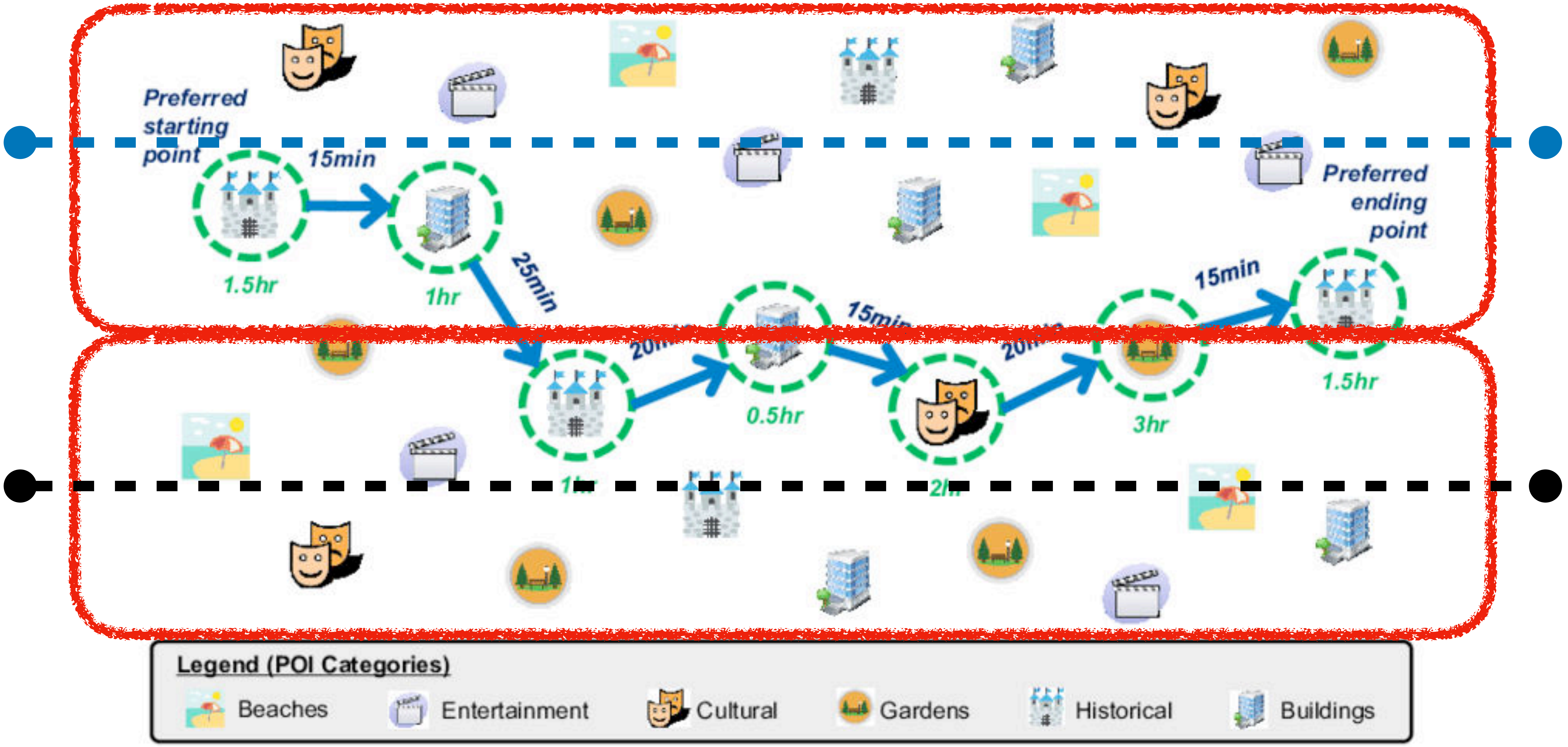
†Department of Computer Science, Aalto University, Finland.
‡Department of Computer Science, KTH Royal Institute of Technology, Sweden, and Department of Computer Science, Aalto University, Finland.

overview of results

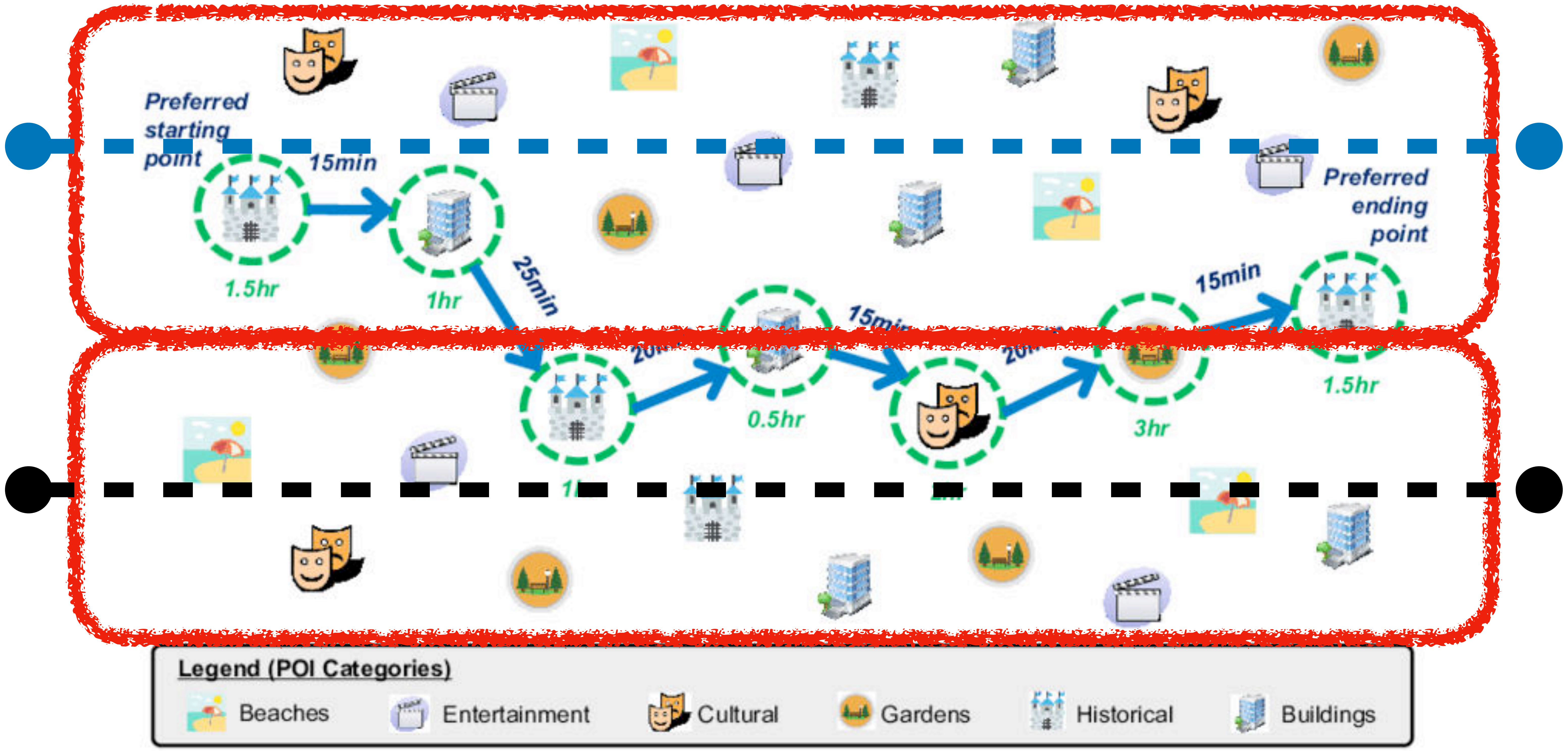
problem	complexity
k-temppath	$O(2^k(nt+m))$
pathmotif	$O(2^k(nt+m))$
colorfulpath	$O(2^k(nt+m))$
(s,d)-colorfulpath	$O(2^k(nt+m))$
rainbowpath	$O(q^k 2^k(nt+m))$
EC-temppath	$O(2^k(nt+m))$
EC-pathmotif	$O(2^k(nt+m))$
VC-pathmotif	$O(2^k(nt+m))$

n - num of vertices t - max timestamp
m - num of edges k - length of path/walk

extracting a solution



extracting a solution



- Bjorklund, Kaski and Kowalik (ESA 2016)
- recursively divide the graph and search for a pattern
- $O(k \log n)$ queries
- we improve this to exactly k queries (Thejaswi et al. 2021)

vertex localisation

- static underlying graph — backbone network obtained by ignoring edge timestamps
- fact : there exists a temporal path if there exists a path in static underlying graph (vice versa might not be true)
- build a sieve to evaluate find all vertices which are incident to at least one match
- difficult to give a theoretical bound on the size of the underlying graph

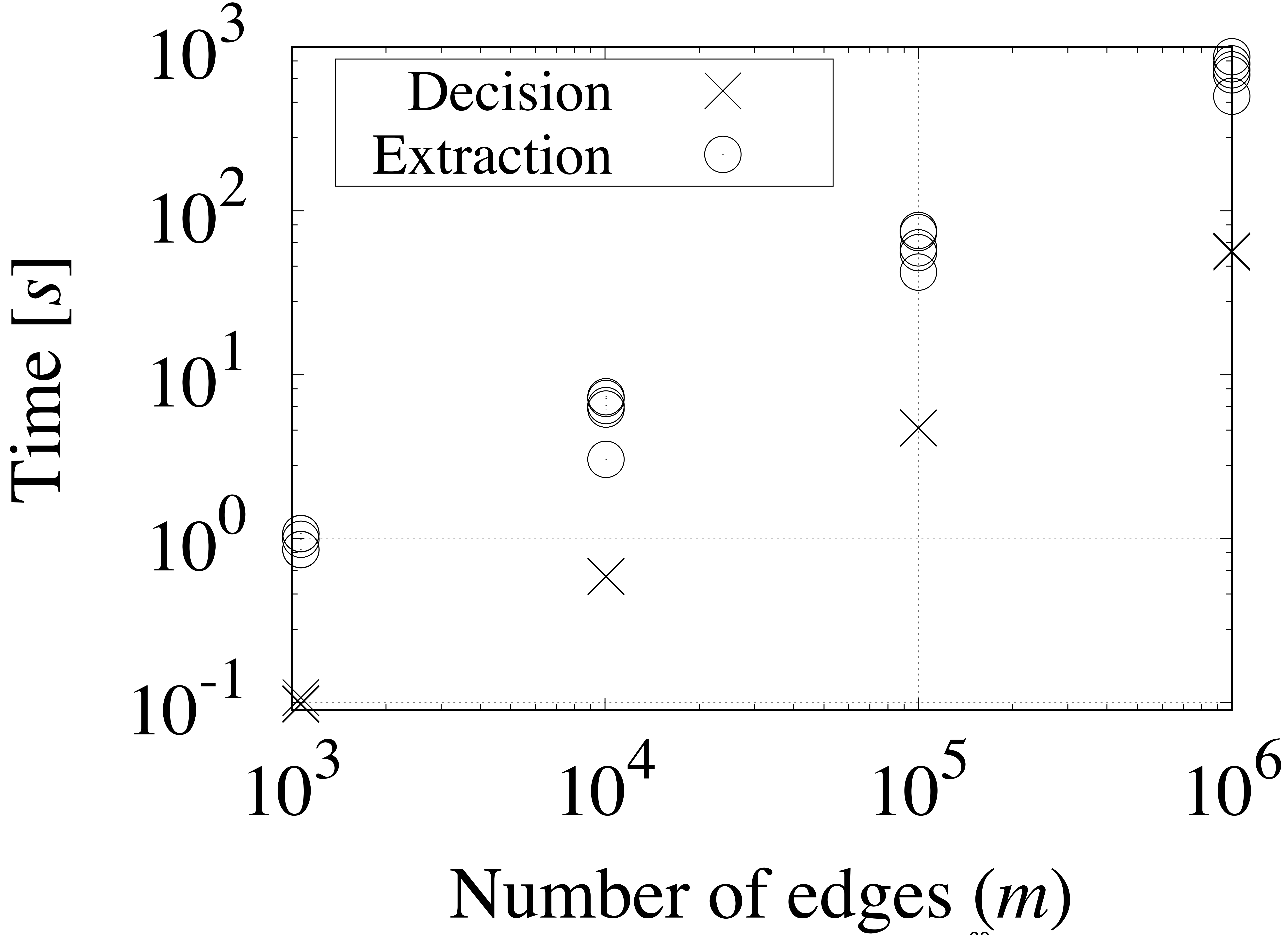
experiments

experimental results

- datasets
  - *transportation networks* from Helsinki and Madrid
    up to 36 million edges, 8 thousand vertices, 1400 timestamps
  - *temporal graphs* from SNAP
    up to 800 thousand edges, 130 thousand vertices, 100 thousand timestamps
  - synthetic graphs
    d-regular and power-law graphs using graph generator

- hardware
  - *workstation*
    4-core Haswell CPU with 16 GB main memory
  - *computenode*
    24-core Haswell CPU with 128 GB main memory

edge linear scaling (m)



d-regular graphs

$d = 20$ (fixed)

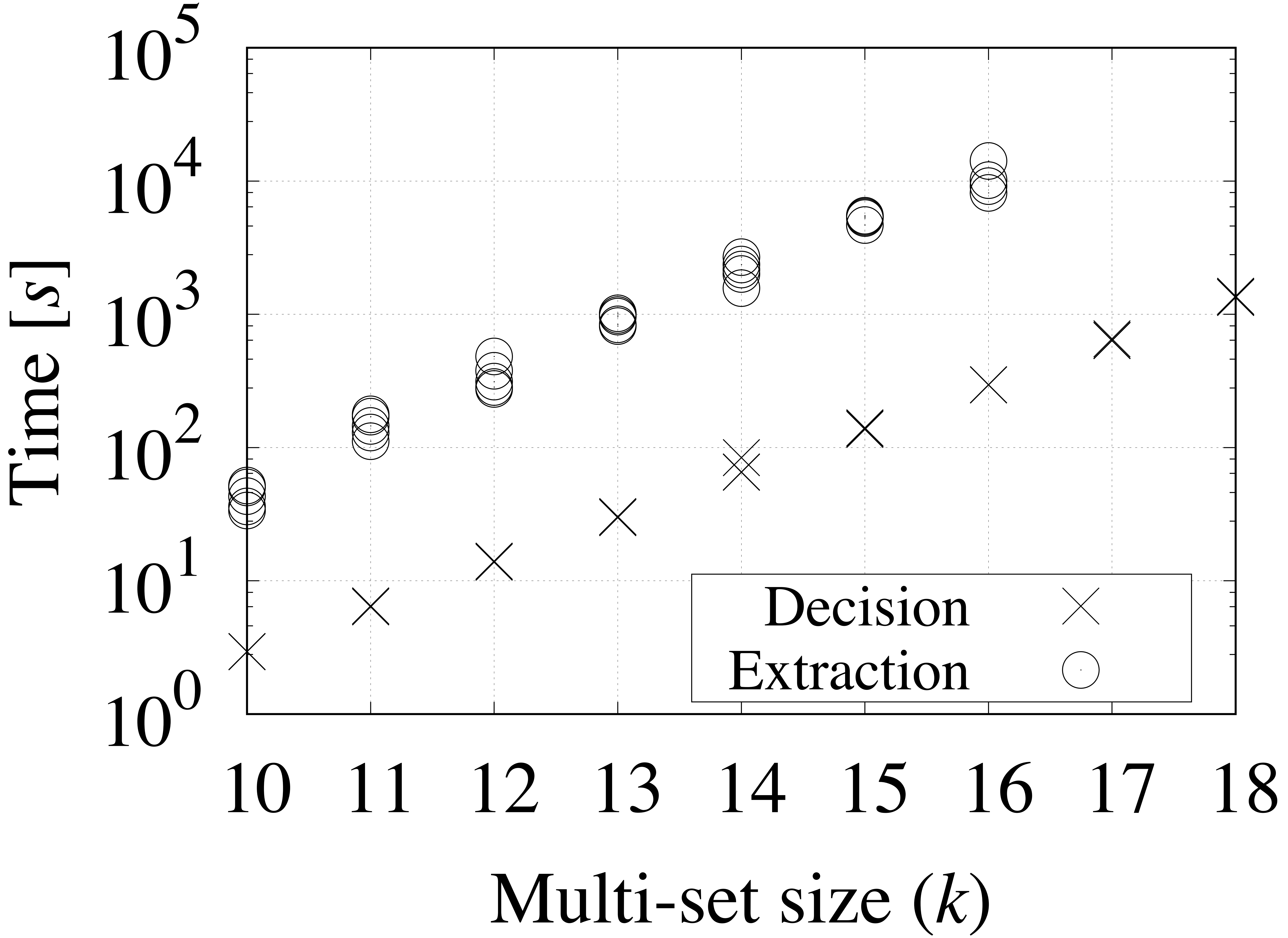
$t = 100$ (fixed)

$k = 8$ (fixed)

workstation

* decision — decide existence
extraction — extract a solution

multiset exponential scaling (k)



d-regular graphs

$d = 20$ (fixed)

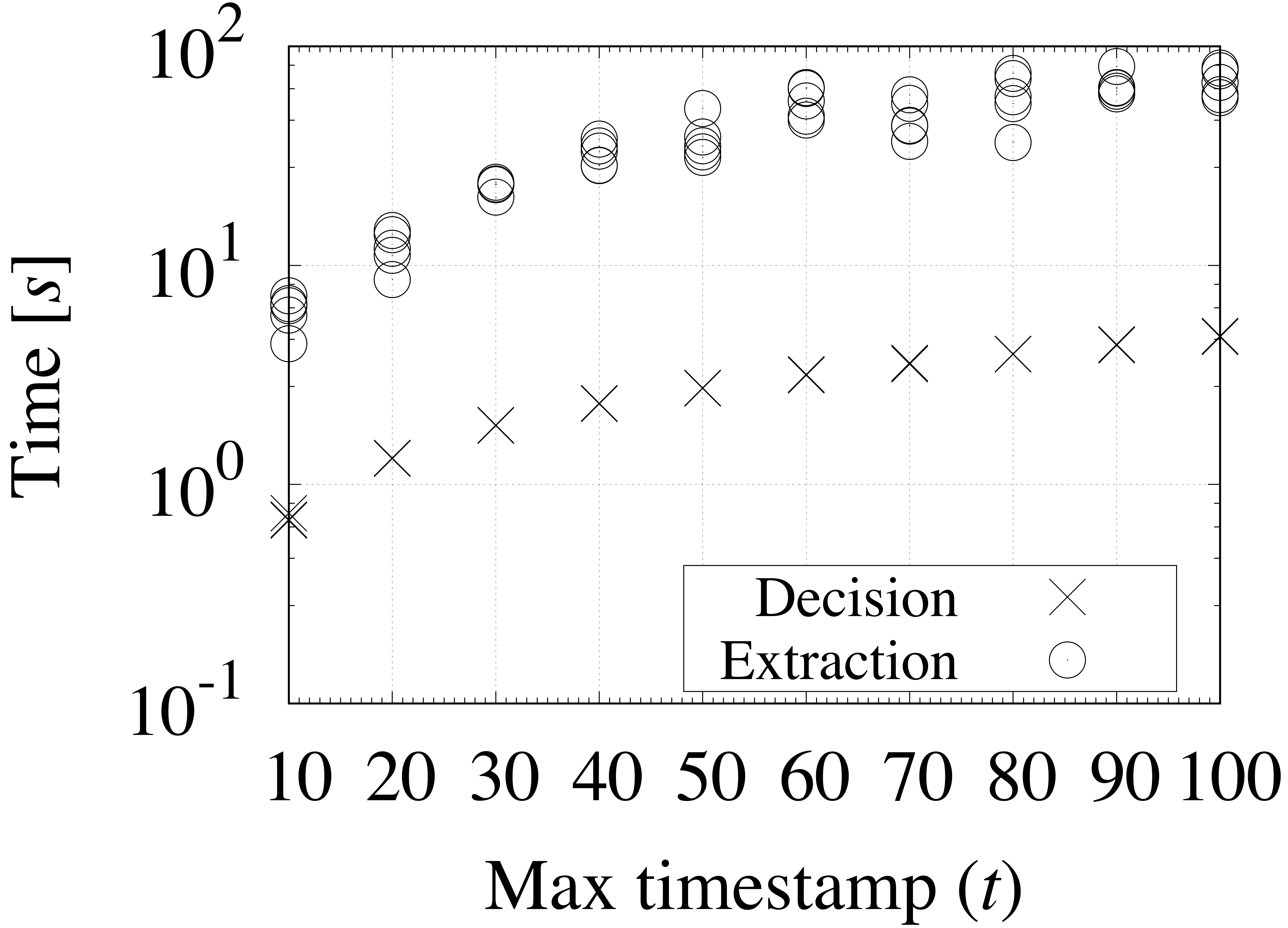
$t = 100$ (fixed)

$n = 10^3$ (fixed)

workstation

* decision — decide existence
extraction — extract a solution

max timestamp scaling (t)



d-regular graphs

$d = 20$ (fixed)

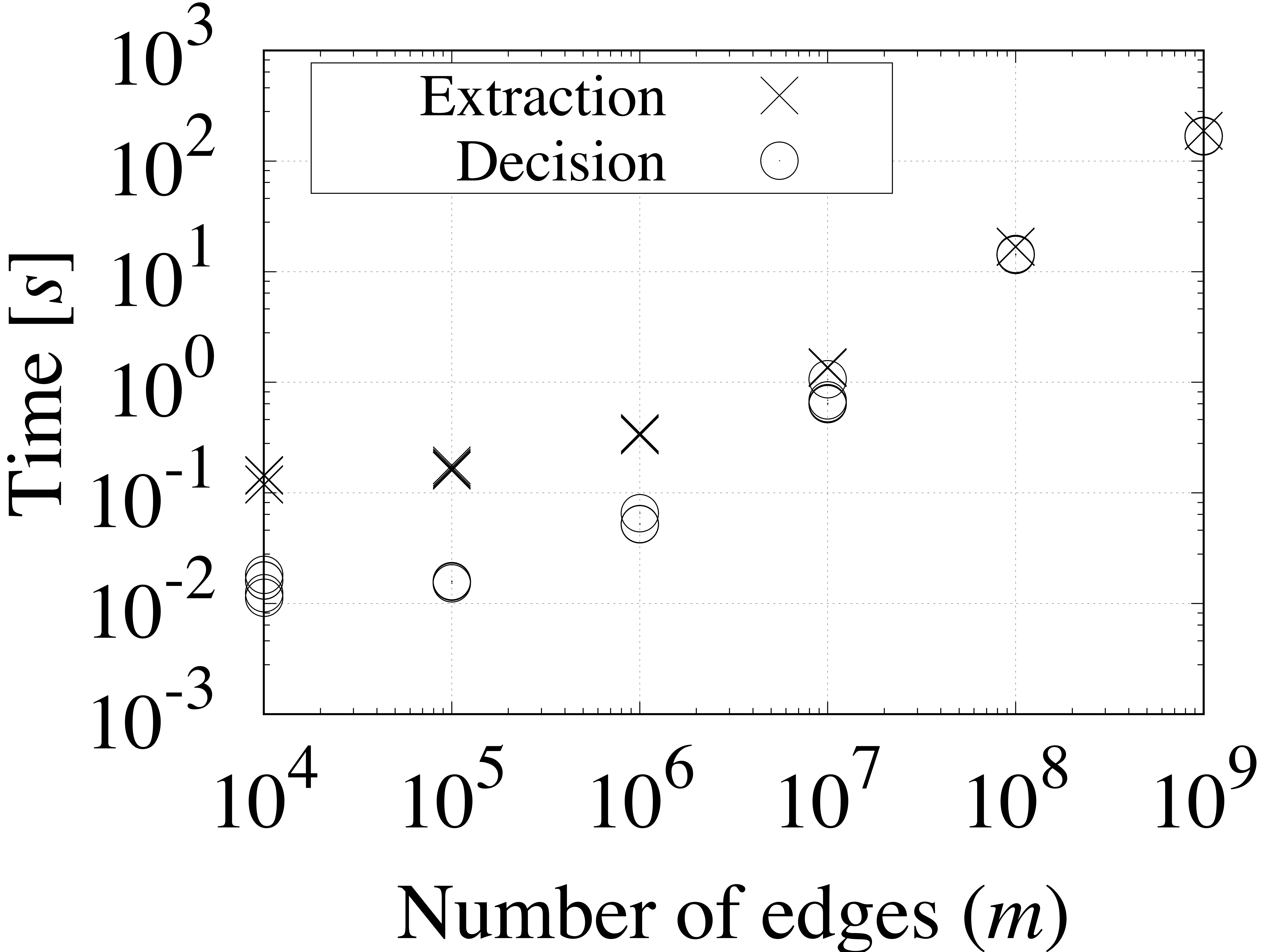
$k = 8$ (fixed)

$n = 10^4$ (fixed)

workstation

* decision — decide existence
extraction — extract a solution

scaling to one billion edges (m)



d-regular graphs

$d = 200$ (fixed)

$k = 5$ (fixed)

$t = 200$ (fixed)

computenode

* decision — decide existence
extraction — extract a solution

real-world datasets

Dataset	n	m	t	$k=5$		$k=10$	
				Base	Alg	Base	Alg
Tram(M)	70	35144	1265	1.37	0.24	1337.98	28.05
Train(M)	91	43677	1181	40.01	0.25	–	24.12
Bus(M)	4597	2254993	1440	6337.89	1.27	–	278.91
IU-bus(M)	7543	1495055	1440	744.79	1.30	–	325.51
Bus(H)	7959	6403785	1440	–	1.67	–	444.66
Metro(M)	467	37565706	1440	–	12.87	–	98.69

* Base — baseline (extraction)
Alg — algebraic fingerprinting (extraction)

real-world datasets

Dataset	n	m	t	*No vloc (seconds)*	*Vloc (seconds)*	*Speedup*	*Memory (GB)*
Bitcoin alpha	3783	24,190	1647	0.69	0.36	1.9	0.10
Madrid tram	70	35,139	1265	0.20	0.12	1.7	0.00
Bitcoin otc	5881	35,596	31,467	22.19	13.27	1.7	2.95
DNC emails	1891	39,268	19,383	4.63	2.83	1.6	0.58
Madrid train	91	43,672	1181	0.19	0.12	1.7	0.00
College msg	1899	58,975	35,913	12.52	7.14	1.8	1.11
Chess	7301	64,962	100	0.12	0.10	1.1	0.01
Elections	7118	103,679	98,026	85.85	53.32	1.6	11.40
Emails EU core	986	327,228	139 649	44.15	23.93	1.8	2.26
Epinions	131,828	841,376	939	5.31	4.63	1.1	1.97
Madrid interurban bus	7543	1,495,050	1440	1.44	1.11	1.3	0.22
Madrid bus	4597	2,254,988	1440	1.77	1.40	1.3	0.21
Helsinki bus	7959	6,403,780	1440	3.50	3.52	1.0	0.41
Madrid metro	467	37,565,706	1195	12.87	12.87	1.0	1.76

summary of algorithmic results

Problem	Hardness	Time complexity	Space complexity
k-TEMPPATH	**NP**-complete (Lemma 5.1)	$\mathcal{O}(2^k k(nt+m))$	$\mathcal{O}(nt)$
PATHMOTIF	**NP**-complete (Lemma 5.2)	$\mathcal{O}(2^k k(nt+m))$	$\mathcal{O}(nt)$
COLORFULPATH	**NP**-complete (Lemma 5.3)	$\mathcal{O}(2^k k(nt+m))$	$\mathcal{O}(nt)$
(s,d)-COLORFULPATH	**NP**-complete (Lemma 5.4)	$\mathcal{O}(2^k k(nt+m))$	$\mathcal{O}(nt)$
RAINBOWPATH	**NP**-complete (Lemma 5.5)	$\mathcal{O}(q^k 2^k k(nt+m))$	$\mathcal{O}(nt)$
EC-TEMPPATH	**NP**-complete (Lemma 5.6)	$\mathcal{O}(2^k(nk+m))$	$\mathcal{O}(n)$
EC-PATHMOTIF	**NP**-complete (Lemma 5.7)	$\mathcal{O}(2^k(nk+m))$	$\mathcal{O}(n)$
VC-PATHMOTIF	**NP**-complete (Lemma 5.8)	$\mathcal{O}(2^k k(nt+m))$	$\mathcal{O}(nt)$
VC-COLORFULPATH	Polynomial	$\mathcal{O}(mt)$	$\mathcal{O}(nt)$

references

- Thejaswi, S. and Gionis, A., 2020. *Pattern detection in large temporal graphs using algebraic fingerprints.* In *Proceedings of the 2020 SIAM International Conference on Data Mining* (pp. 37-45).
- Thejaswi, S., Gionis, A. and Lauri, J., 2020. *Finding path motifs in large temporal graphs using algebraic fingerprints*. *Big Data*, *8*(5), pp.335-362. Special issue — Best of SIAM Data Mining 2020
- Thejaswi, S. Lauri, J. and Gionis, A., 2020. *Restless reachability problems in temporal graphs. arXiv preprint arXiv:2010.08423*

- source code — *https://github.com/suhastheju*

— thank you