36th Annual Symposium on Combinatorial Pattern Matching

CPM 2025, June 17–19, 2025, Milan, Italy

Edited by

Paola Bonizzoni

Veli Mäkinen

Editors

Paola Bonizzoni
University of Milano-Bicocca, Milan, Italy
paola.bonizzoni@unimib.it

Veli Mäkinen
University of Helsinki, Finland
veli.makinen@helsinki.fi

ACM Classification 2012
Theory of computation → Algorithm design techniques; Theory of computation → Data compression; Theory of Computation → Design and analysis of algorithms; Theory of computation → Fixed parameter tractability; Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Pattern matching; Theory of computation → Problems, reductions and completeness; Theory of computation → Sorting and searching; Theory of computation → Theory and algorithms for application domains; Mathematics of computing → Discrete mathematics; Mathematics of computing → Combinatorics on words; Mathematics of computing → Combinatoric problems; Applied computing → Computational biology

ISBN 978-3-95977-369-0

Published online and open access by
Schloss Dagstuhl – Leibniz-Zentrum für Informatik GmbH, Dagstuhl Publishing, Saarbrücken/Wadern, Germany. Online available at https://www.dagstuhl.de/dagpub/978-3-95977-369-0.

Publication date
June, 2025

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at https://portal.dnb.de.

Digital Object Identifier: 10.4230/LIPIcs.CPM.2025.0

ISBN 978-3-95977-369-0 **ISSN 1868-8969** **https://www.dagstuhl.de/lipics**

LIPIcs – Leibniz International Proceedings in Informatics

LIPIcs is a series of high-quality conference proceedings across all fields in informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Regular Papers

36th Annual Symposium on Combinatorial Pattern Matching (CPM 2025).
Editors: Paola Bonizzoni and Veli Mäkinen
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Preface

The Annual Symposium on Combinatorial Pattern Matching (CPM) has a tradition spanning over 35 years and is considered the leading conference for the Stringology research community. The objective of the annual CPM meetings is to provide an international forum for research in combinatorial pattern matching and related applications such as computational biology, data compression and data mining, coding, information retrieval, natural language processing, and image processing (i.e. 2D strings).

This volume contains the papers presented at the 36th Annual Symposium on Combinatorial Pattern Matching (CPM 2025) held on June 17–19, 2025 in Milan, Italy. The conference program includes 28 contributed papers and three invited talks, by

- Inge Li Gørtz (Technical University of Denmark, Denmark),
- Nicola Prezza (Ca' Foscari University of Venice, Italy)
- Kunihiko Sadakane (The University of Tokyo, Japan)

For the seventh time, CPM includes the "Highlights of CPM" special session for presenting the highlights of recent developments in combinatorial pattern matching. In this seventh installment, we selected as highlight papers "Almost Linear Size Edit Distance Sketch" by Michal Koucký and Michael Saks, presented at STOC 2024, and "Prokrustean Graph: A substring index for rapid k-mer size analysis" by Adam Park and David Koslicki, presented at RECOMB 2025.

The contributed papers for CPM 2025 were selected out of 70 submissions, corresponding to an acceptance ratio of 40%. This edition used for the first time a two-round system. We selected 14 papers in each round. There were 8 resubmissions in the second round of papers rejected in the first round. One paper in the second round continued the work of a first-round accepted paper, and these appear now as parts A and B in the proceedings. Each paper received at least three reviews. We thank the members of the Program Committee and all the additional external subreviewers, who are listed below, for their hard, invaluable, and collaborative effort that resulted in an excellent scientific program. We also thank the CPM Steering Committee for their support and advice.

The CPM 2025 was co-located with a StringMasters workshop held on June 16 and June 20, 2024 in Milan. The StringMasters workshop was organized by Zsuzsanna Lipták and Golnaz Badkobeh. Preceding these events, a two-day summer school was held in Milan. The school was organized by Giulia Bernardini, Gabriele Fici and Raffella Rizzi, and provided lectures by Hideo Bannai (Kyushu University, Japan), Jonas Ellert (École Normale Supérieure, France) and Giulia Punzi (University of Pisa, Italy). We thank the Organizing Committee chaired by Gianluca Della Vedova and the organizers of the co-located events for the arrangements enabling a memorable event.

The Annual Symposium on Combinatorial Pattern Matching started in 1990, and has since then taken place every year. Previous CPM meetings were held in Paris, London (UK), Tucson, Padova, Asilomar, Helsinki, Laguna Beach, Aarhus, Piscataway, Warwick, Montreal, Jerusalem, Fukuoka, Morelia, Istanbul, Jeju, Barcelona, London (Ontario, Canada), Pisa, Lille, New York, Palermo, Helsinki, Bad Herrenalb, Moscow, Ischia, Tel Aviv, Warsaw, Qingdao, Pisa, Copenhagen (on-line), Wrocław, Prague, Marne-la-Vallée, and Fukuoka. From 1992 to the 2015 meeting, all proceedings were published in the LNCS (Lecture Notes in Computer Science) series. Since 2016, the CPM proceedings have appeared in the

36th Annual Symposium on Combinatorial Pattern Matching (CPM 2025).
Editors: Paola Bonizzoni and Veli Mäkinen
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

LIPIcs (Leibniz International Proceedings in Informatics) series, as volume 54 (CPM 2016), 78 (CPM 2017), 105 (CPM 2018), 128 (CPM 2019), 161 (CPM 2020), 191 (CPM 2021), 223 (CPM 2022), 259 (CPM 2023), and 296 (CPM 2024). The entire submission and review process was carried out using the EasyChair conference system.

Paola Bonizzoni and Veli Mäkinen
CPM 2025 Program Committee Chairs

Program Committee

- Amihood Amir, Bar Ilan University, Israel
- Hideo Bannai, Institute of Science Tokyo, Japan
- Giulia Bernardini, University of Trieste, Italy
- Paola Bonizzoni, University of Milano-Bicocca, Italy, co-chair
- Manuel Cáceres, Aalto University, Finland
- Bastien Cazaux, University of Lille, CNRS, France
- Panagiotis Charalampopoulos, Birkbeck, University of London, UK
- Nadia El-Mabrouk, University of Montreal, Canada
- Johannes Fischer, TU Dortmund, Germany
- Travis Gagie, Dalhousie University, Canada
- Pawel Gawrychowski, University of Wroclaw, Poland
- Jan Holub, Czech Technical University in Prague, Czech Republic
- Shunsuke Inenaga, Kyushu University, Japan
- Dominik Kempa, Stony Brook University, USA
- Tomasz Kociumaka, Max Planck Institute for Informatics, Germany
- Gregory Kucherov, CNRS/Gustave Eiffel University, France
- Avivit Levy, Shenkar College of Engineering, Design and Art, Israel
- Veli Mäkinen, University of Helsinki, Finland, co-chair
- Sabrina Mantaci, Università di Palermo, Italy
- Gonzalo Navarro, University of Chile, Chile
- Solon Pissis, CWI, Netherlands
- Cinzia Pizzi, University of Padova, Italy
- Giulia Punzi, University of Pisa, Italy
- Svetlana Puzynina, Saint Petersburg State University, Russia
- Rajeev Raman, University of Leicester, UK
- Joe Sawada, University of Guelph, Canada
- Teresa Anna Steiner, Technical University of Denmark, Denmark
- Jens Stoye, University of Bielefeld, Germany
- Philip Wellnitz, National Institute of Informatics, Japan

36th Annual Symposium on Combinatorial Pattern Matching (CPM 2025).
Editors: Paola Bonizzoni and Veli Mäkinen
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

List of External Reviewers

Alejandro Pacheco
Alexander Valyuzhenich
Andrea Frosini
Anuran Maity
B. Riva Shalom
Bartlomiej Dudek
Brian Riccardi
Daniel Gabric
Davide Cozzi
Dennis Wong
Diego Arroyuelo
Dmitry Kosolobov
Dominik Freydenberger
Dominik Köppl
Dora Giammarresi
Egor Gorbachev
Estéban Gabory
Filippo Mignosi
Francisco Olivares
Gabriel Bathie
Gabriele Fici
Gianluca Della Vedova
Giovanni Buzzega
Giulio Ermanno Pibiri
Hector Ferrada
Hiroki Shibata
Itai Boneh
Jakub Radoszewski
Jannik Olbrich
Jarkko Kari
Jarno Alanko
Jonas Ellert
Jouni Sirén
Katsuhisa Yamanaka
Kazuhiro Kurita
Knut Reinert
László Kozma
Leonard Bohnenkämper
Luca Parmigiani
Marinella Sciortino
Markus Schmid
Markus Whiteland
Massimo Equi
Mathilde Girard
Mattéo Delabre
Michael Itzhaki
Nikita Sivukhin
Pascal Caron
Ragnar Groot Koerkamp
Robert Mercas
Rocco Ascone
Roland Wittler
Sebastian Schmidt
Sergey Kirgizov
Shoshana Marcus
Simon R. Tarnow
Simone Faro
Stefan Siemer
Sung-Hwan Kim
Takuya Mieno
Tatiana Starikovskaya
Tizian Schulz
Tomohiro I
Vincent Vajnovszki
Wiktor Zuba
Yoshifumi Sakai

36th Annual Symposium on Combinatorial Pattern Matching (CPM 2025).
Editors: Paola Bonizzoni and Veli Mäkinen
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Representing Paths in Digraphs

Riccardo Dondi
Università degli Studi di Bergamo, Italy

Alexandru Popa
Department of Computer Science, University of Bucharest, Romania

Abstract

In this contribution we consider two combinatorial problems related to graph string matching, motivated by recent approaches in computational genomics. Given a DAG where each node is labeled by a symbol, the problems aim to find a path in the DAG whose nodes contain all (or the maximum number of) symbols of the alphabet. We introduce a decision problem, Σ-Representing Path, that asks whether there exists a path that contains all the symbols of the alphabet, and an optimization problem, called Maximum Representing Path, that asks for a path that contains the maximum number of symbols. We analyze the complexity of the problems, showing the NP-completeness of Σ-Representing Path when each symbol labels at most three nodes in the DAG, and showing the APX-hardness of Maximum Representing Path when each symbol labels at most two nodes in the DAG. We complement the first result by giving a polynomial-time algorithm for Σ-Representing Path when each symbol labels at most two nodes in the DAG. Then we investigate the parameterized complexity of the two problems when the DAG has a limited distance from a set of disjoint paths and we show that both problems are W[1]-hard for this parameter. We consider the approximation of Maximum Representing Path, giving an approximation algorithm of factor $\sqrt{OPT}$, where OPT is the value of an optimal solution of the problem. We also show that Maximum Representing Path cannot be approximated within factor $\frac{e}{e-1} - \alpha$, for any constant $\alpha > 0$, unless $NP \subseteq DTIME(|V|^{O(\log \log |V|)})$ (V is the set of nodes of the DAG).

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms; Theory of computation → Design and analysis of algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases Graph String Matching, Computational Complexity, Parameterized Complexity, Algorithms

Digital Object Identifier 10.4230/LIPIcs.CPM.2025.1

1 Introduction

Graph string matching and approximate matching have been widely investigated due to their application in the context of pattern matching of hypertexts [1, 2, 20, 17] and panegenome analysis [19, 25, 5]. The problems consider a query string s and a directed graph D, whose nodes are labeled with symbols or strings. A path or a walk in D is associated with a string that is obtained by concatenating the symbols or the strings on the path (or walk) nodes. The problems ask whether there exists a path or a walk that matches, or approximately matches, the query string. In exact matching the two strings have to be identical, while in approximated matching edit operations may be applied on the query string or on the node labels, in order to obtain two identical strings, and such edit operations have to be minimized.

Both matching and approximate matching can be solved in polynomial time if the input graph is a Directed Acyclic Graph (DAG) [17]. When the input digraph admits cycles the exact matching is solvable in polynomial time [1, 2, 20, 18, 13] and conditional lower bounds [8] show that improving the algorithms known in the literature is unlikely. The

approximate matching is solvable in polynomial time when edit operations are applied only in the query string [13] and it is instead NP-hard when the edit operations may be applied on node labels [2], also for binary alphabet [13, 6].

A second research direction that has been recently investigated in sequence analysis, in the context of computational genomics, is the quest for subsequences of a given string that contain all symbols of the alphabet (and that have other specific properties) [21, 7, 15, 3, 16]. The property that each symbol of the alphabet is contained in a subsequence of a given string s is applied in [21] looking for a *run subsequence* of maximum length, where a run subsequence contains at least one substring for each symbol of the alphabet. Another approach is considered in [15, 16] and it looks whether there exists a subsequence of s that consists of substrings of repeated symbols that have length at least two, with the constraint that the subsequence contains each symbol of the alphabet. Both problems are NP-hard [21, 16] and results on their tractability have been given [21, 7, 3, 15, 16].

In this contribution, we introduce new problems that aim to integrate the two aforementioned approaches. On the one hand we consider a DAG having nodes labeled by symbols over an alphabet Σ, since this allows us to represent variants of a sequence as in graph string matching. On the other hand we look for a path that contains all (or the maximum number of) symbols of Σ, in a similar way to the second approach. If there exists a path in the DAG that contains all the symbols of the alphabet, we call this a *representing path*. We introduce a decision problem, called Σ-Representing Path, where we ask whether there exists a representing path in a DAG D. Since in some cases a representing path may not exist, we consider an optimization variant, called Maximum Representing Path, where we look for a path in D that represents the maximum number of symbols in Σ, that is the string associated with the path contains the maximum number of symbols of Σ. Next, we summarize our results.

We start in Section 2 by presenting some concepts and by formally defining the two problems, Σ-Representing Path and Maximum Representing Path. Then we investigate the complexity of the problems and we prove in Section 3 that Σ-Representing Path is NP-complete even when each symbol labels at most three nodes of the input DAG, and Maximum Representing Path is APX-hard when each symbol labels at most two nodes of the DAG. Moreover, in both aforementioned cases, the input DAG has also the maximum degree bounded by three. In Section 3 we prove also a lower bound on the approximation: Maximum Representing Path cannot be approximated with factor $\frac{e}{e-1} - \alpha$, for any constant $\alpha > 0$, unless $NP \subseteq DTIME(|V|^{O(\log\log|V|)})$, where V is the set of nodes of the input DAG. We complement the first hardness result by showing in Section 4 that Σ-Representing Path can be solved in polynomial time when each symbol labels at most two nodes of D.

Then we study in Section 5 how the complexity of the two problems is influenced by the structure of D. Observe that if D consists of a set of disjoint paths (except for the source and the target nodes of D), then the two problems are easy to solve by inspecting each path independently. We show that Σ-Representing Path and Maximum Representing Path are W[1]-hard for parameter distance to disjoint paths. Finally, in Section 6 we present an approximation algorithm for the Maximum Representing Path problem of factor $\sqrt{OPT}$, where OPT is the value of an optimal solution of Maximum Representing Path. We conclude the paper in Section 7 pointing out some future directions. Some of the proofs are not included due to page limit (marked with (*)).

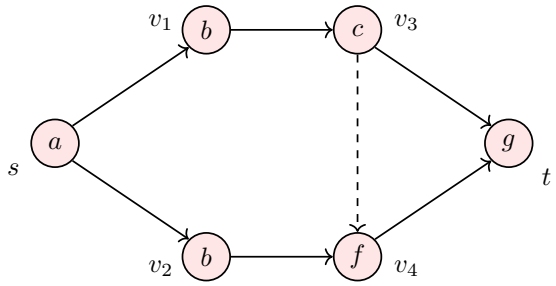

Figure 1 A DAG having labeled nodes (labels are inside each node, node names outside). If all the arcs (including the dashed arc between v_3 and v_4) are in D there exists an $s-t$ path that is Σ-representing: $sv_1v_3v_4t$. If the dashed arc is not in the graph, there is no Σ-representing path in D.

2 Preliminaries

We introduce some notation. For a natural number $n \in \mathbb{N}$, we denote $[n] = \{1, \ldots, n\}$. A Directed Acyclic Graph (DAG) $D = (V, A)$ is a directed graph consisting of a set V of nodes, a set $A = \{(u, v) : u, v \in V\}$ of arcs, such that there is no directed cycle in D. Given a finite alphabet Σ and a DAG $D = (V, A)$, we define a labeling function λ that associates a symbol with each node of D, that is, $\lambda : V \to \Sigma$.

A path p in D is a sequence $v_{p,1} \ldots v_{p,z}$ ($v_{p,i}$ represents the i-th node of path p) of adjacent distinct nodes of V, that is $(v_{p,i}, v_{p,i+1}) \in A$ for $i \in [z-1]$ and $v_{p,i} \neq v_{p,j}$, for each $i, j \in [z]$ with $i \neq j$. The path is called a $v_{p,1} - v_{p,z}$ path, since it starts from $v_{p,1}$ and ends in $v_{p,z}$. The set $\Sigma(p)$ of symbols represented by p is defined as follows:

$$\Sigma(p) = \{c \in \Sigma : \lambda(v_{p,i}) = c, \text{ for some } i \in [z]\}.$$

A path p in D (labeled by λ) is Σ-*representing* if $\Sigma(p) = \Sigma$ (an example is given in Fig. 1); we denote the nodes of the path as $V(p)$. When a path p contains a node labeled by symbol $c \in \Sigma$, we say that p *covers* c. We denote the length of p (the number of nodes in p) by $|p|$. A longest path in a graph is a path having maximum length. Finding a longest path in a graph is an NP-hard problem and even hard to approximate [14], but in DAGs it can be computed in linear time [22]. Now, we define the first problem we study in this paper (we assume that the labeling λ is surjective, so each symbol in Σ is associated with at least one node of the input graph D).

▶ **Problem 1.** *(Σ-Representing Path)*
Input: *A DAG $D = (V, A)$, with a source node $s \in V$, a target node $t \in V$, a labeling $\lambda : V \to \Sigma$.*
Output: *Is there an $s - t$-path in D that is Σ-representing?*

Since there are cases where a DAG does not contain an $s - t$-path in D that is Σ-representing (see the example in Fig. 1), we consider a second problem, where we look for an $s - t$-path in D that contains the maximum number of distinct symbols of Σ.

▶ **Problem 2.** *(Maximum Representing Path)*
Input: *A DAG $D = (V, A)$, with a source node $s \in V$, a target node $t \in V$, a labeling $\lambda : V \to \Sigma$.*
Output: *An $s - t$-path in D that covers the maximum number of symbols in Σ.*

Given a DAG $D = (V, A)$ and a node $v \in V$, the degree of v is the number of arcs incoming to v or outgoing from v, that is

$deg(v) = |\{(u, v) \in A\} \cup \{(v, u) \in A\}|$.

The transitive closure of D is a graph $D' = (V, A')$ where A' is defined as follows:

$A' = \{(u, v) : u, v \in V, u \neq v$ and there is a path from u to v in $D\}$.

Given two DAGs $D_1 = (V_1, A_1)$ and $D_2 = (V_2, A_2)$, with $V_1 \cap V_2 = \emptyset$, having source nodes s_1, s_2, respectively, and target nodes t_1, t_2, respectively, the *concatenation* of D_1 and D_2 is a DAG $D = (V_1 \cup V_2, A_1 \cup A_2 \cup \{(t_1, s_2)\})$. Informally D is obtained by adding an arc from t_1 to s_2. The definition can be easily extended to more than two DAGs, specifying the order of concatenation. Note that the definition of concatenation holds also for paths.

3 Hardness

In this section we present hardness results for the Σ-Representing Path and Maximum Representing Path problems. In Subsection 3.1 we show the NP-completeness of the Σ-Representing Path problem even if when each symbol labels at most three nodes of D, whose degree is bounded by three. In Subsection 3.2 we show two hardness of approximation results for the Maximum Representing Path problem: first we show that the problem is APX-hard even if each symbol labels at most three nodes of D, whose degree is bounded by three; then we show a stronger result on arbitrary instances, namely that Maximum Representing Path cannot be approximated within a factor of $\frac{e}{e-1} - \alpha$, for any constant $\alpha > 0$, unless $NP \subseteq DTIME(|V|^{O(\log\log|V|)})$.

3.1 NP-completeness

In this subsection we prove that the Σ-Representing Path problem is NP-complete via a reduction from 3-SAT, a classical NP-complete problem (see, e.g., [11]). Our reduction holds even on restricted instances of the problems when each symbol labels at most three nodes of the input DAG D and each node has degree bounded by three.

We recall that 3-SAT, given a formula ϕ in conjunctive normal form over a set of variables X, where each clause is a disjunction of three literals (a variable or its negation), asks for an assignment of the variable set X so that each clause of ϕ is satisfied.

▶ **Theorem 1.** *The Σ-Representing Path problem is NP-complete, even in the case when (1) each symbol labels at most three nodes of the DAG, that is, $\forall c \in \Sigma, |v \in V : \lambda(v) = c| \leq 3$, and (2) the degree of each node is bounded by three.*

Proof. First, observe that the Σ-Representing Path problem is in the class NP since, given an $s - t$ path, we can verify in polynomial time if the path contains each symbol of Σ.

Given an instance of the 3-SAT problem, that is a Boolean formula ϕ with n variables and m clauses, denote the n variables of the formula ϕ as $x_1, x_2, \ldots, x_n$ and the m clauses as $C_1, C_2, \ldots, C_m$. We construct an instance of Σ-Representing Path, that is, a labeled DAG $(D = (V, A), \lambda)$, as follows (see an example in Fig. 2). First, assume without loss of generality that no variable x_i appears only negated or nonnegated – otherwise said, every variable has at least one negated and one nonnegated occurrence in the formula ϕ. If there exists such a variable, we can simply assign it to *True* (if it is nonnegated) or to *False* (if it is negated) and remove all the clauses that contain the respective variable.

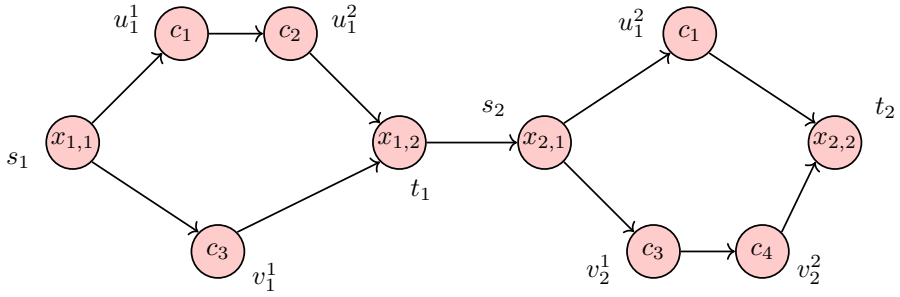

Figure 2 A sketch of the DAG computed by the reduction from 3-SAT. We consider x_1 to appear nonnegated in clauses C_1, C_2 and negated in clause C_3; x_2 to appear nonnegated in clause C_1 and negated in clauses C_3 and C_4.

- **The alphabet.** For each clause C_i, $i \in [m]$, we add a corresponding symbol c_i to the alphabet Σ. Moreover, for each variable x_i, $i \in [n]$, we add symbols $x_{i,1}$, $x_{i,2}$ to Σ.
- **The node set.** For each variable x_i, $i \in [n]$, we add the following nodes in the node set. Assume that x_i appears in k clauses, $k_1 \geq 1$ times nonnegated and $k_2 \geq 1$ times negated. Thus, $k = k_1 + k_2$.
  We first add to the node set two nodes: s_i, t_i. Then, we add nodes $u_i^1, u_i^2, \ldots, u_i^{k_1}$ and nodes $v_i^1, v_i^2, \ldots, v_i^{k_2}$. The source node node of D is $s = s_1$ and the target node of D is $t = t_n$. As for the labeling, $\lambda(s_i) = x_{i,1}$, $\lambda(t_i) = x_{i,2}$; assuming x_i, $i \in [k_1]$, has the j-th nonnegated appearance in clause C_h then $\lambda(u_i^j) = c_h$; assuming x_i, $i \in [k_2]$, has the j-th negated appearance in clause C_h then $\lambda(v_i^j) = c_h$.
- **The arc set.** For each $i \in [n]$ we add the following arcs:
  - Arcs outgoing from s_i: (s_i, u_i^1), (s_i, v_i^1)
  - Arcs incoming in t_i: $(u_i^{k_1}, t_i)$, $(v_i^{k_2}, t_i)$
  - Arcs to connect two subgraphs associated with x_i and x_{i+1}, where $i \in [n-1]$: (t_i, s_{i+1})
  - Arcs (u_i^j, u_i^{j+1}), with $j \in [k_1 - 1]$, and (v_i^j, v_i^{j+1}), with $j \in [k_2 - 1]$

We show now that our reduction is correct. More precisely, we show that (1) each symbol labels at most three nodes of D, (2) the degree of D is bounded by three and (3) the formula ϕ has a satisfying assignment, if and only if, there is an $s-t$-path in D that contains every symbol of Σ. Consider (1) and note that each symbol $x_{i,1}$, $x_{i,2}$, $i \in [n]$, labels exactly one node of D. Moreover, since each clause consists of three literals, then each of the symbols c_i, $i \in [m]$, labels exactly three nodes of D.

Consider (2). The nodes of D having degree larger than two, are possibly s_i and t_i, $i \in [n]$. Each s_i has indegree at most one and outdegree two, since the arcs outgoing from s_i are (s_i, u_i^1) and (s_i, v_i^1), thus $deg(s_i) \leq 3$. Each t_i has outdegree at most one and indegree two, since the arcs incoming to t_i are $(u_i^{k_1}, t_i)$ and $(v_i^{k_2}, v_i^1)$. Hence $deg(t_i) \leq 3$.

Now, we prove (3). First, given a satisfying assignment of ϕ, we select the path in D as follows. For each $i \in [n]$, if x_i is *True*, then in the subgraph corresponding to the variable x_i, we take the path $s_i u_i^1 u_i^2 \ldots u_i^{k_1} t_i$ thus, covering the symbols $x_{i,1}$, $x_{i,2}$, and $c_1, c_2, \ldots, c_{k_1}$, associated with clauses where x_i appears nonnegated; otherwise, if x_i is assigned to *False*, we choose the path $s_i v_i^1 v_i^2 \ldots v_i^{k_2} t_i$, thus covering the labels $x_{i,1}$, $x_{i,2}$, and $c_1, c_2, \ldots, c_{k_2}$, associated with clauses where x_i appears negated. Between two subgraphs the path contains arc (t_i, s_{i+1}), $i \in [n-1]$.

Observe that we take the path that covers precisely the symbols corresponding to the clauses satisfied by x_i. Observe that the symbols $x_{i,1}$, $x_{i,2}$ that are associated with variables are always covered. Since the formula ϕ is satisfied by the assignment, the path obtained by concatenating the subpaths also covers all the symbols in the alphabet.

Conversely, assume that we are given an $s-t$-path in D that covers all the symbols in Σ. If the $s-t$-path contains the subpath $s_i u_i^1 u_i^2 \dots u_i^{k_1} t_i$, $i \in [n]$, then we set x_i to *True*, otherwise we set x_i to *False*. The crucial observation that was also mentioned in the first part of the reduction is that the symbols covered in the subgraph associated with variable x_i correspond to the clauses satisfied by the assignment to x_i. Since all the labels are covered, the assignment produced satisfies all the clauses. ◀

3.2 Hardness of Approximation

The previous result (Theorem 1) can be extended to Maximum Representing Path. By reducing from Max 2-SAT, which is known to be APX-hard [12], we show that Maximum Representing Path is APX-hard even when each symbol labels at most two nodes of the input DAG and the degree of each node is bounded by three.

▶ **Corollary 2** (*). *The Maximum Representing Path problem is APX-hard, even in the case when (1) each symbol labels at most two nodes of the input DAG, that is,* $\forall c \in \Sigma, |v \in V : \lambda(v) = c| \leq 2$*, and (2) the degree of each node is bounded by three.*

We now present the second inapproximability result, namely an approximation preserving reduction from the Max k-Cover problem (defined next), that cannot be approximated within factor $\frac{e}{e-1} - \varepsilon$, for any constant $\varepsilon > 0$, unless $NP \subseteq DTIME(|U|^{O(\log\log|U|)})$.

▶ **Problem 3.** *(Max k-Cover)*
Input: *A collection of n sets $\mathcal{S} = \{S_1, S_2, \dots S_n\}$ over a universe U and an integer k.*
Output: *k sets from $\mathcal{S}$ whose union have the largest cardinality.*

Now, we present our second inapproximability result.

▶ **Theorem 3** (*). *The Maximum Representing Path problem cannot be approximated within a factor of $\frac{e}{e-1} - \alpha$, for any constant $\alpha > 0$, unless $NP \subseteq DTIME(|V|^{O(\log\log|V|)})$.*

Proof. (Sketch) We present an approximation preserving reduction from Max k-Cover to Maximum Representing Path. Given an input instance $(\mathcal{S}, k)$ of Max k-Cover, we construct the following instance of Maximum Representing Path. First, the set of nodes of the DAG D is defined as follows:

- We add $k+1$ nodes $v_1, v_2, \dots v_{k+1}$, where $s = v_1$ and $t = v_{k+1}$, labeled with the same symbol a.
- For each element $x \in U$, define a path $p(x)$ of length $|U|$, and each node of $p(x)$ is labeled with a distinct symbol x_i, $i \in [|U|]$.
- Between two nodes v_j, v_{j+1}, $j \in [k]$, add $|\mathcal{S}|$ paths, each one associated with a set S_i (denoted by $p(S_i)$); each path associated with S_i consists of the concatenation of paths $p(x_{i,1}), \cdots, p(x_{i,z})$, where $x_{i,1}, \dots, x_{i,z}$ are the elements in set S_i.

Given a solution $S_1^*, \dots, S_k^*$ of Max k-Cover on instance $(U, \mathcal{S})$ such that $\bigcup_{i=1}^{k} |S_i^*| = h$, we can compute in polynomial time a solution of Maximum Representing Path on instance D that covers $|U|h + 1$ symbols, by defining, between each two nodes v_j and v_{j+1}, $j \in [k]$, path $p(S_j^*)$. For the other direction, given a solution p of Maximum Representing Path on instance D that covers $|U|h + 1$ symbols, we can compute in polynomial time a solution $S_1^*, \dots, S_k^*$ of Max k-Cover on instance $(U, \mathcal{S})$, by defining S_j^* as the set corresponding to the path between nodes v_j and v_{j+1}, $j \in [k]$. ◀

4 A Polynomial Time Algorithm for Σ-Representing Path when Each Symbol Labels at most Two Nodes

In this section we present a polynomial time exact algorithm (Algorithm 1) for the Σ-Representing Path problem when each symbol in Σ labels at most two nodes of the input DAG D. We first introduce the notion of *compatible nodes*.

▶ **Definition 4.** *We say that two nodes $u_1, u_2 \in V$ are* compatible *if there exists an $s-t$-path that contains both nodes, that is either a path $s \dots u_1 \dots u_2 \dots t$ or $s \dots u_2 \dots u_1 \dots t$. Otherwise, the nodes are called* incompatible.

Algorithm 1 reduces the input instance of the Σ-Representing Path problem when each symbol labels at most two nodes to an instance of 2-SAT, which is know to be polynomial time solvable [24, 4].

Algorithm 1 A polynomial time exact algorithm for the Σ-Representing Path problem where each symbol labels at most two nodes of D.

Input: A labeled DAG $(D = (V, A), \lambda)$, a source node s and a target node t.

1. Construct a 2-SAT Boolean formula ϕ that has $|V|$ variables as follows:
   a. The formula ϕ has a variable x_u for each node $u \in V$.
   b. For every two nodes $u, v \in V$ that are incompatible we add the clause $(\overline{x_u} \vee \overline{x_v})$.
   c. For every symbol $c \in \Sigma$, let $u, v \in V$, with $u \neq v$, such that $\lambda(u) = \lambda(v) = c$; we add the clause $(x_u \vee x_v)$.
2. Decide in polynomial time if ϕ is satisfiable using the algorithm from [4]. Output YES, if ϕ is satisfiable and output NO, otherwise.

▶ **Theorem 5** (*). *The Σ-Representing Path problem is solvable in $O(|V||A|)$ time in the case when each symbol labels at most two nodes of D, that is, $\forall c \in \Sigma, |u \in V : \lambda(u) = c| \leq 2$.*

5 Distance from Disjoint Paths

The Σ-Representing Path and the Maximum Representing Path problems are trivial if the DAG, after the removal of s and t, consists of a set of disjoint paths. Here we consider the two problems when parameterized by distance to disjoint paths and we show that Σ-Representing Path and Maximum Representing Path are W[1]-hard for this parameter. The *distance to disjoint paths* is defined as the minimum number of nodes to be removed from a graph (in this case D) such that the resulting graph consists of a set of node disjoint paths (except possibly for s and t). We prove the hardness results by giving a parameterized reduction from the Multicolored Clique problem, defined as follows.

▶ **Problem 4.** *(Multicolored Clique)*
Input: *An undirected graph $G = (W, E)$, whose nodes are partitioned into color classes $W = W_1 \uplus W_2 \cdots \uplus W_k$ and each edge in E connects two nodes that are in a different color class.*
Output: *Is there a clique in G that contains exactly one node for each color class?*

A clique in G that contains exactly one node for each color class is called a *multicolored clique*. Given an instance $G = (W, E)$ of the Multicolored Clique problem (recall that the partition of W in color classes is given in input), we construct a corresponding instance

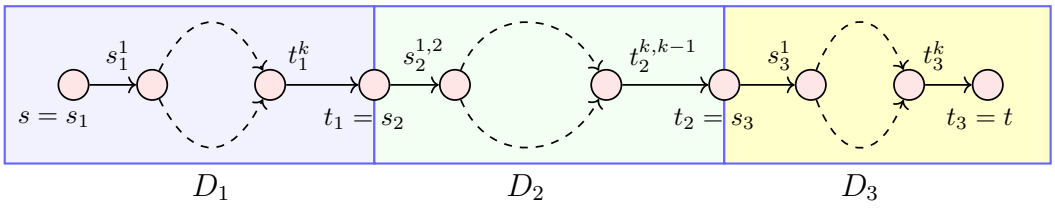

Figure 3 The structure of graph D, computed by the reduction from Multicolored Clique to Σ-Representing Path (dashed arcs represent DAGs between two nodes).

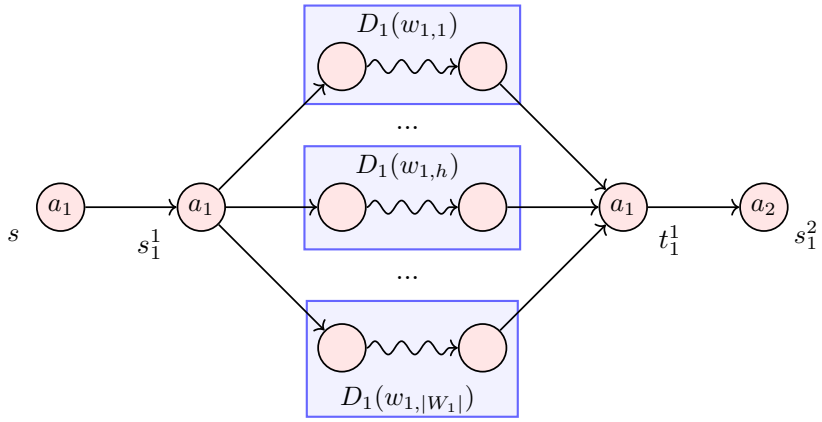

Figure 4 A sketch of the DAG D_1^1 (we include also s, the souruce of D, and s_1^2) computed by the reduction from Multicolored Clique to Σ-Representing Path.

(D, λ) of Σ-Representing Path. The DAG D consists of three subgraphs, that share some nodes (see Fig. 3 for a representation of the structure of D): (1) a DAG D_1 with source node $s_1 = s$ and target node t_1, (2) A DAG D_2 with source node $s_2 = t_1$ and target node t_2, and (3) A DAG D_3 with source node $s_3 = t_2$ and target node $t_3 = t$.

We first give an informal description of the reduction and then we present it formally. D_1 and D_2 encode a multicolored clique (symbols not covered by a path in D_1 and D_2 represent nodes and edges of a multicolored clique), D_3 enables to cover all the symbols not selected in D_1 and D_2 (assuming a path selected in D_1 and D_2 contains all the symbols except those encoding a multicolored clique).

Consider $G = (W, E)$, where $W = \{w_{1,1}, \ldots, w_{k,|W_k|}\}$, and $w_{i,h}$, $i \in [k]$ and $h \in [|W_i|]$, represents the h-th node of color class W_i (we assume that the nodes in each color class have some ordering). We start by giving an informal description of D_1, D_2 and D_3.

The DAG D_1 consists of k concatenated DAGs $D_1^1, \ldots, D_1^k$ (see Fig. 4 for a sketch of D_1^1), where each D_1^i, $i \in [k]$, is associated with a color class W_i. Each D_1^i, $i \in [k]$, encodes the selection of exactly one node of W_i and it contains a path, denoted by $D_1(w_{i,h})$, for each node $w_{i,h} \in W_i$, with $h \in [|W_i|]$. D_1^i has a source node s_1^i and a target node t_1^i. Node s_1^i has arcs to the first node of each path $D_1(w_{i,h})$. Each $D_1(w_{i,h})$ contains the symbols associated with nodes in W_i, except for the symbols associated with $w_{i,h}$.

The DAG D_2 consists of $k(k-1)$ concatenated DAGs $D_2^{1,2}, \ldots, D_2^{k,k-1}$ (see Fig. 5 for a sketch of $D_2^{1,2}$), where each $D_2^{i,j}$, $i, j \in [k]$, $i \neq j$, is associated with edges connecting nodes of color class W_i and color class W_j. DAG $D_2^{i,j}$ has a source node $s_2^{i,j}$ and a target node $t_2^{i,j}$.

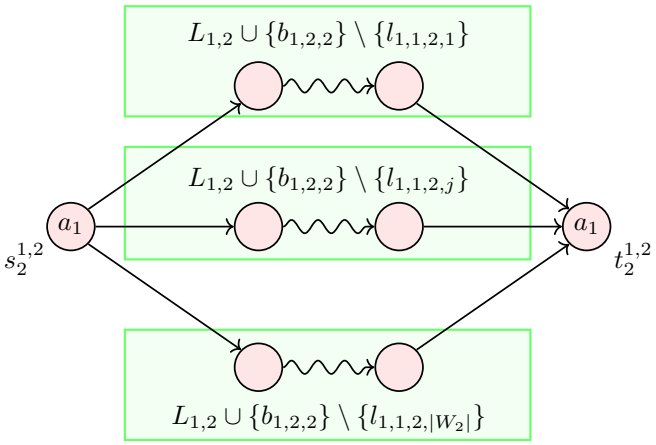

Figure 5 The first part of subgraph $D_2^{1,2}$ computed by the reduction from Multicolored Clique. We assume that $w_{1,1}$ is adjacent to $w_{2,1}$, $w_{2,j}$ and $w_{2,|W_2|}$.

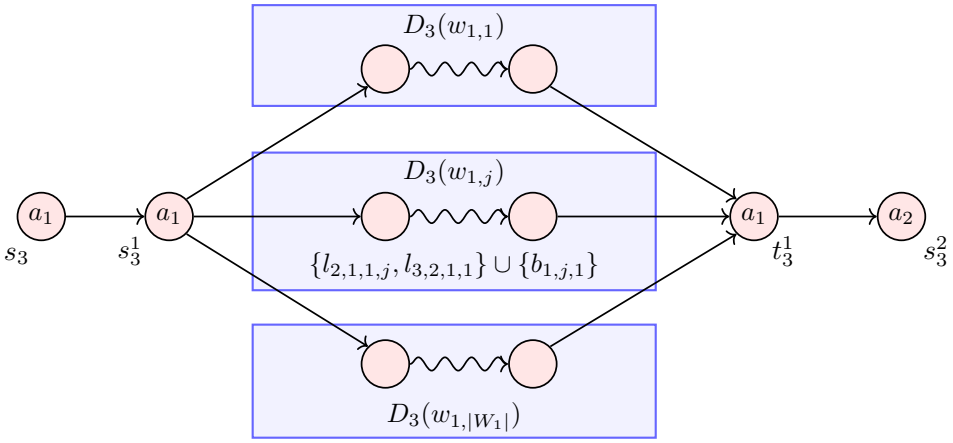

Figure 6 The first subgraph of the DAG D_3 computed by the reduction from Multicolored Clique, associated with the nodes in W_1. We assume that $w_{1,j}$ is adjacent to $w_{2,1}$, $w_{3,2}$, thus $D_3(w_{1,j})$ is a path whose nodes have labels $l_{2,1,1,j}, l_{3,2,1,j}$ (these labels are represented in the box of $D_3(w_{1,j})$). We include also s_3, the source of D_3, and node s_3^2.

For each edge $\{w_{i,h}, w_{j,q}\} \in E$, $h \in [|W_i|]$ and $q \in [|W_j|]$, $D_2^{i,j}$ contains one path, denoted by $D_2(w_{i,h}, w_{j,q})$ whose nodes are labeled by one symbol associated with $w_{i,h}$, and set $L_{i,j}$ (encoding edges between nodes of W_i and of W_j) except for the symbol associated with edge $\{w_{i,h}, w_{j,q}\}$.

The DAG D_3 (see Fig. 6) consists of k concatenated DAGs D_3^1, ..., D_3^k, each one associated with a color class W_i, $i \in [k]$. Each D_3^i, $i \in [k]$, has a source node s_3^i and a target node t_3^i. Node s_3^i has arcs to $|W_i|$ paths, one path $D_3(w_{i,h})$ for each node $w_{i,h} \in W_i$, $h \in [|W_i|]$. The nodes in $D_3(w_{i,h})$ have labels that encode $w_{i,h}$ and the edges incident in $w_{i,h}$. The idea is that the symbols not covered by a path in D_1 and D_2 can be covered by a path in D_3 only if the path in D_1 and D_2 contains all the symbols except those encoding edges of a multicolored clique in G and one symbol for each node in the multicolored clique.

Now, we present the details of the reduction. We start by defining the alphabet Σ and some subsets of Σ.

$$\Sigma = \{a_i : W_i \subseteq W, i \in [k]\} \cup \{b_{i,h,q} : w_{i,h} \in W_i, i \in [k], h \in [|W_i|], q \in [k]\} \cup \{l_{i,h,j,q} : i, j \in [k], i \neq j, h \in [|W_i|], q \in [|W_j|] \wedge w_{i,h} \in W_i \wedge w_{j,q} \in W_j \wedge \{w_{i,h}, w_{j,q}\} \in E\}.$$

We define sets $B(w_{i,h})$, $w_{i,h} \in W_i$, and B_i, $i \in [k]$, of symbols:

$$B(w_{i,h}) = \bigcup_{q \in [k]} b_{i,h,q}, \quad B_i = \bigcup_{h \in [|W_i|]} B(w_{i,h}).$$

Given $i, j \in [k]$, with $i \neq j$, we denote the subset $L_{i,j}$ of symbols as follows:

$$L_{i,j} = \{l_{i,h,j,q} : h \in [|W_i|], q \in [|W_j|] \wedge w_{i,h} \in W_i \wedge w_{j,q} \in W_j \wedge \{w_{i,h}, w_{j,q}\} \in E\}.$$

The set L_i, $i \in [k]$, is defined as follows:

$$L_i = \bigcup_{j \in [k] \wedge j \neq i} L_{i,j}.$$

Note that $L_{i,j}$ and $L_{j,i}$ are different subsets, in particular that $l_{i,h,j,q} \neq l_{j,q,i,h}$.

Now, we define the DAG D_1. D_1 has a source node $s_1 = s$ and a target node t_1, both labeled by a_1. D_1 is obtained by concatenating DAGs D_1^i, $i \in [k]$, each one associated with a color class. Each D_1^i has a source node s_1^i and a target node t_1^i. For each $i \in [k-1]$, there exists an arc from t_1^i to s_1^{i+1} (this defines the concatenation of subgraphs D_1^i). Now, we define each subgraph D_1^i, $i \in [k]$:

- Each subgraph D_1^i has a source node s_1^i and a target node t_1^i, labeled by a_i.
- Node s_1^i is connected to $|W_i|$ disjoint paths $D_1(w_{i,h})$, each one associated with a node $w_{i,h} \in W_i$.
- Each path $D_1(w_{i,h})$ is labeled by symbols

$$B_i \setminus \bigcup_{q \in [k]} \{b_{i,h,q}\}$$

Finally, there is an arc from node s_1 to s_1^1 and an arc from node t_1^k to node t_1.

Next, we define the DAG D_2. D_2 has a source node $s_2 = t_1$ and a target node t_2, both labeled by symbol a_2. D_2 is obtained by concatenating DAGs $D_2^{i,j}$, with $i, j \in [k]$ and $i \neq j$, each one associated with W_i and W_j. Each $D_2^{i,j}$ has a source node $s_2^{i,j}$ and a target node $t_2^{i,j}$. We assume that DAGs $D_2^{i,j}$ are concatenated as follows: for $i, j \in [k]$ with $i \neq j$, if $j < k$ then there is an arc from the target $t_2^{i,j}$ of $D_2^{i,j}$ to the source $s_2^{i,j+1}$ of $D_2^{i,j+1}$, and if $j = k$ (and $i < k$) there is an arc from the source $s_2^{i,j}$ of $D_2^{i,j}$ to the target $t_2^{i+1,1}$ of $D_2^{i+1,1}$.

Now, we define each subgraph $D_2^{i,j}$, $i, j \in [k]$ and $i \neq j$:

- Each subgraph $D_2^{i,j}$ has a source node $s_2^{i,j}$ and a target node $t_2^{i,j}$, labeled by a_i.
- Node $s_2^{i,j}$ is connected to paths $D_2(w_{i,h}, w_{j,q})$, each one associated with a node $w_{i,h} \in W_i$ and a node $w_{j,q} \in W_j$, such that $\{w_{i,h}, w_{j,q}\} \in E$.
- Each path $D_2(w_{i,h}, w_{j,q})$ is labeled by the set of symbols (recall that $i \neq j$, hence symbol $b_{i,h,i}$ does not label any node on each path $D_2(w_{i,h}, w_{j,q})$):

$$(L_{i,j} \cup \{b_{i,h,j}\}) \setminus \{l_{i,h,j,q}\}$$

Finally, there is an arc from node s_2 to $s_2^{1,2}$ and an arc from node $t_2^{k,k-1}$ to node t_2.

Now, we define the DAG D_3. D_3 has source s_3 and target t_3, both labeled by a_3. D_3 is obtained by concatenating DAGs D_3^i, $i \in [k]$, each one associated with a color class. Each subgraph D_3^i, $i \in [k]$, is defined as follows:

- Each subgraph D_3^i has a source node s_3^i and a target node t_3^i, labeled by a_i.
- Between nodes s_3^i and t_3^i there are $|W_i|$ disjoint paths, $D_3(w_{i,h})$, $h \in [|W_i|]$, each one associated with a node $w_{i,h} \in W_i$.
- The nodes in each path $D_3(w_{i,h})$ are labeled by the following set of symbols

$$\bigcup_{\{w_{i,h},w_{j,q}\}\in E} \{l_{j,q,i,h}\} \cup \{b_{i,h,i}\}.$$

Note that the path $D_3(w_{i,h})$ contains nodes having labels $\{l_{j,q,i,h}\}$ that encode edges of E incident in $w_{i,h}$. Note also that these nodes, for each edge $\{w_{i,h}, w_{j,q}\} \in E$, have labels $\{l_{j,q,i,h}\}$ not $\{l_{i,h,j,q}\}$.

Finally, there is an arc from t_3^i to s_3^{i+1}, with $i \in [k-1]$, an arc from s_3 (the source of D_3) to s_3^1 and an arc from t_3^k to t_3.

Having defined D and its labeling, we start to prove some properties of graph D_1 and D_2.

▶ **Lemma 6.** *Consider an instance G of Multicolored Clique and a corresponding instance (D, λ) of Σ-Representing Path. Given a path p from s_1 to t_1 in D_1, then p covers the following set of symbols:*

1. *Each symbol a_i, $i \in [k]$*
2. *A set B' defined as follows:*

$$B' = \bigcup_{i\in[k]} B_i \setminus \{b_{i,h,q} : \text{such that } p \text{ traverses path } D_1(w_{i,h}), q \in [k]\}.$$

Proof. Let p be a path from s_1 to t_1 in D_1. Nodes s_1^i and t_1^i, $i \in [k]$, must be traversed by any path in D_1, hence also by p, and they are labeled by a_i. Hence point 1 holds.

We prove now point 2. Consider in particular a DAG D_1^i, $i \in [k]$, and the subpath p_i of p in D_1^i. By construction, p_i traverses exactly one of $D_1(w_{i,h})$, with $h \in [|W_i|]$, between s_1^i and t_1^i. Also note that by construction a subgraph of $D_1(w_{i,h})$ contains the symbols B_i, except for the symbols $b_{i,h,q}$, $q \in [k]$, that by construction do not label any node of $D_1(w_{i,h})$, thus point 2 holds. ◀

▶ **Lemma 7.** *Consider an instance G of Multicolored Clique and a corresponding instance (D, λ) of Σ-Representing Path. A path p in D that covers all the symbols in Σ contains a path p_2 from s_2 to t_2 in D_2 such that:*

1. *For each $D_1(w_{i,h})$ traversed by p in D_1, p_2 traverses a subgraph $D_2(w_{i,h}, w_{j,q})$, for some $j \in [k]$, $q \in [|W_j|]$ and covers a set $B'' = \bigcup_{i,q\in[k],i\neq q,h\in[|W_i|]}\{b_{i,h,q} : \text{such that } p \text{ traverses path } D_1(w_{i,h})\}$*
2. *For each $i \in [k]$, p_2 covers a set $L_i' \subseteq L_i$ of symbols, where $L_i' = L_i \setminus N_i$ and N_i is defined as follows:*

$$N_i = \bigcup_{h,j,q \text{ such that } p_2 \text{ traverses subgraph } D_2(w_{i,h}, w_{j,q})} \{l_{i,h,j,q}\}.$$

Proof. Let p be a path from s to t that covers all the symbols in Σ. First, consider point 1. By Lemma 6, the path p in D_1 does not cover the set of symbols $b_{i,h,q}$, $h \in [|W_i|]$, $q \in [k]$, such that $D_1(w_{i,h})$ is traversed by p. Each symbol $b_{i,h,q}$, with $i, q \in [k]$, $i \neq q$ and $h \in [|W_i|]$, labels only nodes of D_1^i and D_2, thus if it is not covered by p in D_1^i it must be covered in D_2. It follows that for each $D_1(w_{i,h})$ traversed by p in D_1, p_2 traverses a subgraph $D_2(w_{i,h}, w_{j,q})$. This implies that p_2 covers

$$B'' = \bigcup_{i,h,q,i,q\in[k],i\neq q,h\in[|W_i|]} \{b_{i,h,q} : \text{ such that } p \text{ traverses path } D_1(w_{i,h})\}$$

and point 1 is proven.

Now, we consider point 2. In each $D_2^{i,j}$,$i, j \in [k]$ and $i \neq j$, path p traverses exactly one subpath $D_2(w_{i,h}, w_{j,q})$, for some $h \in [|W_i|]$ and $q \in [|W_j|]$, whose nodes have labels $L_{i,j} \setminus \{l_{i,h,j,q}\}$. Then p in D_2^i covers a set $L'_i \subseteq L_i$ of symbols, where $L'_i = L_i \setminus N_i$ and

$$N_i = \bigcup_{h,q \text{ with } D_2(w_{i,h}, w_{j,q}) \text{ traversed by } p \text{ in } D_2^{i,j}} \{l_{i,h,j,q}\}$$

hence point 2 is proven. ◀

Based on Lemma 6 and Lemma 7, we can prove the main result of this section.

▶ **Lemma 8** (*). *Consider an instance G of* Multicolored Clique *and a corresponding instance (D, λ) of* Σ-Representing Path. *Then, G contains a multicolored clique if and only if there exists a path in D that covers all the symbols in Σ.*

D has distance from a set of disjoint paths bounded by $2k(k-1)+4k$, since by removing the source and target node of each D_1^i, with $i \in [k]$, of each $D_2^{i,j}$, with $i, j \in [k]$ and $i \neq j$, and of each D_3^i, with $i \in [k]$, we obtain a set of disjoint paths. Since Multicolored Clique is W[1]-hard when parameterized by k [10], we can prove the following theorem.

▶ **Theorem 9** (*). Σ-*Representing Path is W[1]-hard when parameterized by distance to disjoint paths.*

We extend the result of Theorem 9 to the Maximum Representing Path problem.

▶ **Corollary 10** (*). *Maximum Representing Path is W[1]-hard when parameterized by distance to disjoint paths.*

6 An Approximation Algorithm

In this section we present a polynomial time approximation algorithm for the Maximum Representing Path problem that achieves an approximation factor of $\sqrt{OPT}$, where OPT is the number of distinct symbols in an optimal solution. Notice that, since $OPT \leq |\Sigma|$, our algorithm is also a $\sqrt{|\Sigma|}$-approximation. Informally, the algorithm is as follows. First, we create a compatibility DAG $D' = (V', A')$, that is essentially the transitive closure of the input DAG D (see Section 2 for the definition of transitive closure). Then, we consider a total order on Σ, e.g., standard alphabetic order, such that we have $\lambda(u) < \lambda(v)$, for two nodes u and v if the label of u precedes the label of v based on this order. We create two subgraphs of D' (that are implicitly DAGs), D^1 and D^2 as follows:

1. $D^1 = (V, A^1)$ where $(v_1, v_2) \in A^1$ if and only if $(v_1, v_2) \in A'$ and $\lambda(v_1) < \lambda(v_2)$
2. $D^2 = (V, A^2)$ where $(v_1, v_2) \in A^2$ if and only if $(v_1, v_2) \in A'$ and $\lambda(v_1) > \lambda(v_2)$.

Observe that $A^1, A^2 \subseteq A$ and that the set of nodes of D^1 and D^2 is V. We then compute p_1, a longest path between s and t in D^1, and p_2, a longest path between s and t in D^2. As pointed out in Section 2, a longest path in a DAG can be computed in linear time. The algorithm (formally presented in Algorithm 2) outputs the path p_i, $i \in \{1, 2\}$, that has a largest number of distinct symbols. Theorem 11 proves that it is indeed a $\sqrt{OPT}$-approximation for Maximum Representing Path.

▶ **Theorem 11.** *Algorithm 2 is a $\sqrt{OPT}$-approximation for the* Maximum Representing Path *problem and requires $O(|V||A|)$ time.*

Algorithm 2 A $\sqrt{OPT}$-approximation algorithm for Maximum Representing Path.

Input: A labeled DAG $D = (V, A)$, a start node s and a target node t.

1. Construct $D' = (V, A')$ such that $A' = \{(v_1, v_2) \mid \exists$ a path between v_1 and v_2 in $D\}$.
2. Construct $D^1 = (V, A^1)$ such that $A^1 = \{(v_1, v_2) \mid (v_1, v_2) \in A'$ and $\lambda(v_1) < \lambda(v_2)\}$.
3. Construct $D^2 = (V, A^2)$ such that $A^2 = \{(v_1, v_2) \mid (v_1, v_2) \in A'$ and $\lambda(v_1) > \lambda(v_2)\}$.
4. Compute p_1 a longest path in D^1.
5. Compute p_2 a longest path in D^2.
6. Let $p = p_1$ if $|\Sigma(p_1)| > |\Sigma(p_2)|$, otherwise, let $p = p_2$.
7. Output p', the path in D in which we replace each arc $(v_i, v_j) \in p$ with a corresponding path from v_i to v_j in D, and we add s and a path from s to the first node of p (if s is not in p), and t and a path from the last node of p to t (if t is not in p).

Proof. First, we prove that Algorithm 2 computes a feasible solution of Maximum Representing Path, that is, the path p' returned by the Algorithm 2 is an $s - t$-path in D. Consider the DAG $D' = (V, A')$. Since D' is the transitive closure of D, it follows that for any path in D' there exists a corresponding path in D, since for any arc $(u, v) \in A'$, there is a path $p(u, v)$ from u to v in D. Thus, given a path p^* in D', we can compute a corresponding path p in D by concatenating the paths $p(u, v)$ associated with arcs (u, v) in p^*. Since D^1 and D^2 are subgraphs of D', in particular they have the same set of nodes and a subset of the arcs of D', any path in D^1 or D^2 corresponds to a path in D', hence also in D. Moreover, we add s (t, respectively) to the returned path if s (or t, respectively) is not part of the path p, and possibly a path from s to the first node of p (a path from the last node of p to t, respectively), hence the algorithm returns an $s - t$ path in D.

We show now the approximation factor of Algorithm 2. Let p^o be an optimal solution of the Maximum Representing Path problem, that is, a path p^o such that $|\Sigma(p^o)| = OPT$ is maximized. Let $V^o = \{v_1, v_2 \ldots, v_{OPT}\}$ be a subset of the nodes that appear on the path p^o in this order and have pairwise distinct labels, that is $\lambda(v_i) \neq \lambda(v_j), \forall 1 \leq i < j \leq OPT$.

According to the Erdös-Szekeres theorem [9], every sequence of $z^2 + 1$ distinct integers contains a monotonic (increasing or decreasing) sequence of length $z + 1$. Thus, the string associated with p^o contains a a monotonic (increasing or decreasing) sequence of length at least $\sqrt{OPT}$, hence the path p^o contains $k \geq \sqrt{OPT}$ nodes $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$, such that v_{i_x} appears before v_{i_y} in p, for $x < y$ and $x, y \in [k]$, and either $\lambda(v_{i_1}) < \lambda(v_{i_2}) < \cdots < \lambda(v_{i_k})$ or $\lambda(v_{i_1}) > \lambda(v_{i_2}) > \cdots > \lambda(v_{i_k})$. Notice that since p^o is a path in D, for any two nodes $v_i, v_j \in p^o$ such that $i < j$, we have $(v_i, v_j) \in A'$. Thus, the path $v_{i_1}, v_{i_2}, \ldots, v_{i_k}$ is either a path in D^1 or in D^2, thus it has length not larger than that of p_1 or of p_2, respectively. Since the path p' returned by Algorithm 2 is obtained by taking the nodes in one of $\{p_1, p_2\}$ that covers the maximum number of symbols (and possibly adding other nodes), and each p_i, $i \in \{1, 2\}$, contains nodes with distinct labels, then $|\Sigma(p')| \geq k \geq \sqrt{OPT}$. Thus, Algorithm 2 is a $\sqrt{OPT}$-approximation algorithm for Maximum Representing Path.

We consider now the time complexity of Algorithm 2. Step 1 can be computed in $O(|V||A|)$ time [23] and $|A'|$ contains $O(|V|^2)$ arcs. Step 2 and Step 3, can be computed in $O(|V| + |A'|)$ time by traversing D'. Step 4 and 5 can be computed in $O(|V| + |A^1|)$ and $O(|V| + |A^2|)$ time [26], respectively, where $O(|V| + |A^1|)$ and $O(|V| + |A^2|)$ are bounded by $O(|V| + |A'|)$. Thus the overall time complexity is $O(|V||A| + |V|^2)$ and, since we can assume that no node is isolated, then $|A| \geq |V| - 1$, thus the overall time complexity is $O(|V||A|)$. ◀

7 Conclusion

In this contribution we have introduced two combinatorial problems (Σ-Representing Path and Maximum Representing Path) that ask to identify a path in a node labeled DAG that contains all (or a subset of maximum size) of the alphabet symbols. We have proved results on the computational complexity and parameterized complexity of the two problems, and we have studied the approximation of Maximum Representing Path.

One of the most interesting future directions is to further investigate the approximate complexity of the Maximum Representing Path problem: does it admit constant factor approximation algorithms? It is interesting to design approximation algorithms for some restrictions on the DAG structure, for example when the degree is bounded. It is also interesting to study other structural properties of the DAG that may lead to polynomial-time algorithms for both problems.

References

1 Tatsuya Akutsu. A linear time pattern matching algorithm between a string and a tree. In *Combinatorial Pattern Matching, 4th Annual Symposium, CPM 93, Padova, Italy, June 2-4, 1993, Proceedings*, pages 1–10, 1993. doi:10.1007/BFb0029792.

2 Amihood Amir, Moshe Lewenstein, and Noa Lewenstein. Pattern matching in hypertext. *Journal of Algorithms*, 35(1):82–99, 2000. doi:10.1006/jagm.1999.1063.

3 Yuichi Asahiro, Hiroshi Eto, Mingyang Gong, Jesper Jansson, Guohui Lin, Eiji Miyano, Hirotaka Ono, and Shunichi Tanaka. Approximation algorithms for the longest run subsequence problem. In Laurent Bulteau and Zsuzsanna Lipták, editors, *34th Annual Symposium on Combinatorial Pattern Matching, CPM 2023, June 26-28, 2023, Marne-la-Vallée, France*, volume 259 of *LIPIcs*, pages 2:1–2:12. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPICS.CPM.2023.2.

4 Bengt Aspvall, Michael F. Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Inf. Process. Lett.*, 8(3):121–123, 1979. doi:10.1016/0020-0190(79)90002-4.

5 Jasmijn A. Baaijens, Paola Bonizzoni, Christina Boucher, Gianluca Della Vedova, Yuri Pirola, Raffaella Rizzi, and Jouni Sirén. Computational graph pangenomics: a tutorial on data structures and their applications. *Nat. Comput.*, 21(1):81–108, 2022. doi:10.1007/S11047-022-09882-6.

6 Riccardo Dondi, Giancarlo Mauri, and Italo Zoppis. On the complexity of approximately matching a string to a directed graph. *Inf. Comput.*, 288:104748, 2022. doi:10.1016/J.IC.2021.104748.

7 Riccardo Dondi and Florian Sikora. The longest run subsequence problem: Further complexity results. In Pawel Gawrychowski and Tatiana Starikovskaya, editors, *32nd Annual Symposium on Combinatorial Pattern Matching, CPM 2021, July 5-7, 2021, Wrocław, Poland*, volume 191 of *LIPIcs*, pages 14:1–14:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. doi:10.4230/LIPICS.CPM.2021.14.

8 Massimo Equi, Veli Mäkinen, Alexandru I. Tomescu, and Roberto Grossi. On the complexity of string matching for graphs. *ACM Trans. Algorithms*, 19(3):21:1–21:25, 2023. doi:10.1145/3588334.

9 Paul Erdös and George Szekeres. A combinatorial problem in geometry. *Compositio mathematica*, 2:463–470, 1935.

10 Michael R. Fellows, Danny Hermelin, Frances A. Rosamond, and Stéphane Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009. doi:10.1016/J.TCS.2008.09.065.

11 Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

12 Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.

13 Chirag Jain, Haowen Zhang, Yu Gao, and Srinivas Aluru. On the complexity of sequence to graph alignment. In Lenore J. Cowen, editor, *Research in Computational Molecular Biology - 23rd Annual International Conference, RECOMB 2019, Washington, DC, USA, May 5-8, 2019, Proceedings*, volume 11467 of *Lecture Notes in Computer Science*, pages 85–100. Springer, 2019. doi:10.1007/978-3-030-17083-7_6.

14 David R. Karger, Rajeev Motwani, and G. D. S. Ramkumar. On approximating the longest path in a graph. *Algorithmica*, 18(1):82–98, 1997. doi:10.1007/BF02523689.

15 Manuel Lafond, Wenfeng Lai, Adiesha Liyanage, and Binhai Zhu. The longest subsequence-repeated subsequence problem. In Weili Wu and Jianxiong Guo, editors, *Combinatorial Optimization and Applications - 17th International Conference, COCOA 2023, Hawaii, HI, USA, December 15-17, 2023, Proceedings, Part I*, volume 14461 of *Lecture Notes in Computer Science*, pages 446–458. Springer, 2023. doi:10.1007/978-3-031-49611-0_32.

16 Wenfeng Lai, Adiesha Liyanage, Binhai Zhu, and Peng Zou. The longest letter-duplicated subsequence and related problems. *Acta Informatica*, 61(3):315–329, 2024. doi:10.1007/S00236-024-00459-7.

17 Udi Manber and Sun Wu. Approximate string matching with arbitrary cost for text and hypertext. In *Advances in Structural and Syntactic Pattern Recognition*, pages 22–33, 1992. doi:10.1142/9789812797919_0002.

18 Gonzalo Navarro. Improved approximate pattern matching on hypertext. *Theoretical Computer Science*, 237(1-2):455–463, 2000. doi:10.1016/S0304-3975(99)00333-3.

19 Ngan Nguyen, Glenn Hickey, Daniel R. Zerbino, Brian J. Raney, Dent Earl, Joel Armstrong, W. James Kent, David Haussler, and Benedict Paten. Building a pan-genome reference for a population. *Journal of Computational Biology*, 22(5):387–401, 2015. doi:10.1089/cmb.2014.0146.

20 Kunsoo Park and Dong Kyue Kim. String matching in hypertext. In Zvi Galil and Esko Ukkonen, editors, *Combinatorial Pattern Matching, 6th Annual Symposium, CPM 95, Espoo, Finland, July 5-7, 1995, Proceedings*, volume 937 of *Lecture Notes in Computer Science*, pages 318–329. Springer, 1995. doi:10.1007/3-540-60044-2_51.

21 Sven Schrinner, Manish Goel, Michael Wulfert, Philipp Spohr, Korbinian Schneeberger, and Gunnar W. Klau. Using the longest run subsequence problem within homology-based scaffolding. *Algorithms Mol. Biol.*, 16(1):11, 2021. doi:10.1186/S13015-021-00191-8.

22 Robert Sedgewick and Kevin Wayne. *Algorithms (Fourth edition deluxe)*. Addison-Wesley, 2016.

23 Steven Skiena. *The Algorithm Design Manual, Third Edition*. Texts in Computer Science. Springer, 2020. doi:10.1007/978-3-030-54256-6.

24 Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972. doi:10.1137/0201010.

25 The Computational Pan-Genomics Consortium. Computational pan-genomics: status, promises and challenges. *Briefings in Bioinformatics*, 19(1):118–135, 2018. doi:10.1093/bib/bbw089.

26 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. URL: http://www.cambridge.org/de/knowledge/isbn/item5759340/?site_locale=de_DE.