Solving Group Interval Scheduling Efficiently

Arindam Biswas[1(✉)], Venkatesh Raman[1], and Saket Saurabh[1,2]

[1] The Institute of Mathematical Sciences, HBNI, Chennai, India
{barindam,vraman,saket}@imsc.res.in
[2] University of Bergen, Bergen, Norway

Abstract. The Group Interval Scheduling problem models the scenario where there is set $[\gamma] = \{1, \ldots, \gamma\}$ of jobs to be processed on a single machine, and each job i can only be scheduled for processing in exactly one time interval from a group G_i of allowed intervals. The objective is to determine if there is a set of $S \subseteq [\gamma]$ of k ($k \in \mathbb{N}$) jobs which can be scheduled in non-overlapping time intervals.

This work describes a deterministic algorithm for the problem that runs in time $O((5.18)^k n^d)$, where $n = |\bigcup_{i \in [\gamma]} G_i|$ and $d \in \mathbb{N}$ is a constant. For $k \geq d \log n$, this is significantly faster than the best previously-known deterministic algorithm, which runs in time $O((12.8)^k \gamma n)$. We obtain our speedup using efficient constructions of *representative families*, which can be used to solve the problem by a dynamic programming approach.

Keywords: Group · Job · Interval · Scheduling · Graph · Independent · Colourful · Representative · Hash · Fixed · Parameter · FPT · Multivariate

1 Introduction

A ubiquitous problem arising in industrial processes is when there are multiple jobs to be processed on a single machine, and some of the jobs have conflicting time constraints. In this scenario, the next best thing is for the machine to process as many jobs as possible without violating any time constraints. This can be modelled as follows.

> Group Interval Scheduling
>
> **Instance:** A pair (J, k), where $J = \{G_1, \ldots, G_\gamma\}$ such that G_i ($i \in [\gamma]$) is a set of disjoint intervals of $\mathbb{R}$, and $k \in \mathbb{N}$.
> **Question:** Is there a set of at least k disjoint intervals $S \subseteq \bigcup J$ such that $|S \cap G_i| \leq 1$ ($i \in [\gamma]$)?

The sets G_i represent time constraints: job i ($i \in [\gamma]$) can only be processed during a time interval from the set G_i. In this scheme, picking a set S of disjoint

C. J. Colbourn et al. (Eds.): IWOCA 2019, LNCS 11638, pp. 97–107, 2019.
https://doi.org/10.1007/978-3-030-25005-8_9

intervals such that $|S \cap G_i| \leq 1$ $(i \in [\gamma])$ is equivalent to scheduling the set $\{i \in [\gamma] \mid G_i \in S\}$ of jobs on the machine such that they occupy distinct time intervals.

Group Interval Scheduling is known to be NP-hard [6] while being polynomial-time solvable (via a reduction to 2-SAT; see [3]) when there are at most 2 intervals per job.

Consider a finite set X and a function $f : X \to [t]$. A subset $S \subseteq X$ is *colourful* with respect to f if for any $x, y \in S$, $x \neq y \implies f(x) \neq f(y)$. The following problem is an equivalent formulation of Job Interval Scheduling that models constraints among the jobs using a graph and a colouring function on its vertex set.

> Colourful Independent Set
>
> **Instance:** A triple (G, ϕ, k), where G is a graph, $\phi : V \to [\gamma]$ is a colouring and $k \in \mathbb{N}$.
> **Question:** Is there is an independent set $S \subseteq V$ in G of size k which is colourful with respect to ϕ?

Let (J, k) be an instance of Group Interval Scheduling. Define $V = \bigcup_{i \in [\gamma]} G_i$. Taking V as the vertex set, define $G = (V, E)$, where $E = \{uv \mid u \cap v \neq \emptyset\}$ and define $\phi : V \to [\gamma]$ by $\phi(v) = i$, where $i \in [\gamma]$ such that $v \in G_i$. This gives an equivalent instance (G, ϕ, k) of Colourful Independent Set on interval graphs: k jobs from J can be scheduled on the machine if and only if G has an independent set of size k that is colourful with respect to ϕ.

This alternative formulation of Group Interval Scheduling is used throughout the remainder of this paper.

Fixed-Parameter Tractability. The results are presented here in the framework of Parameterized Complexity. Consider a computational problem P and let x be a instance of P. Suppose that there is a number $k_x \in \mathbb{N}$ that describes a property of x, e.g. the optimal solution value for x. Such a scheme is called a *parameterization* of P, and k_x is called the *parameter* of x. Attaching the parameter to the problem instance gives us a *parameterized* problem: $\{\langle x, k_x \rangle \mid x \in P\}$.

Given any parameterized problem Q with parameterization k, if there is an algorithm that solves it in time $\mathrm{O}(f(k)n^c)$, where $f : \mathbb{N} \to \mathbb{N}$ is a computable function and $c \in \mathbb{N}$ is a constant, then Q is said to be *fixed-parameter tractable*.

Our Results and Previous Work. We consider two parameterizations of Colourful Independent Set: k, the size of the solution sought, and γ, the number of colours used by the colouring ϕ. The question of fixed-parameter tractability was studied by Halldórsson and Karlsson [2] and later by van Bevern et al. [8], which led to the following results.

Proposition 1 (Halldórsson and Karlsson [2], Theorem 4). *Instances* (G, ϕ, k) *of* Colourful Independent Set *on interval graphs can be solved deterministically in time* $\mathrm{O}(2^{\gamma}n)$*, where* γ *is the number of colours.*

Proposition 2 (van Bevern et al. [8], Theorem 4). *Instances* (G, ϕ, k) *of* Colourful Independent Set *on interval graphs can be solved with error probability* ϵ *in time* $\mathrm{O}\Big(|\log \epsilon|(5.5)^k n\Big)$*. The algorithm can be derandomized to solve the problem deterministically in time* $\mathrm{O}\Big((12.8)^k \gamma n\Big)$*, where* γ *is the number of colours.*

Proposition 1 establishes fixed-parameter tractability with respect to γ while Proposition 2 shows that there is a (randomized) fixed-parameter algorithm with respect to k. This work makes the following improvements.

- We show that the running time of the deterministic algorithm of Proposition 2 can actually be improved to $\Big(e^{k+\mathrm{O}\left((\log k)^2\right)} \log \gamma\Big) 2^k n = \mathrm{O}\Big((5.44)^k (\log \gamma) n\Big)$ using smaller families of perfect hash functions (Theorem 1).
- Using efficiently-constructible *representative families*, we obtain an algorithm (Theorem 2) for Colourful Independent Set that runs in time $\mathrm{O}\Big((5.18)^k n^d\Big)$ ($d \in \mathbb{N}$, a constant).

2 Preliminaries

In this section, we introduce notation used in the rest of the paper and review some basic concepts concerning matroids, representative families and perfect hash families.

2.1 Basics

- $\mathbb{N}$ denotes the set of natural numbers and $\mathbb{R}$ denotes the set of real numbers.
- For $t \in \{1, 2, \ldots\}$, $[t]$ denotes the set $\{1, \ldots, t\}$ and for $a, b \in \mathbb{R}$ with $a \leq b$, $[a, b]$ denotes the set $\{x \mid x \in \mathbb{R}, a \leq x \leq b\}$.
- Let X be a set and $\mathcal{F}$ be a family of subsets of X. For $x \in X$, define $x + \mathcal{F} = \{\{x\} \cup S \mid S \in \mathcal{F}\}$.
- Let $G = (V, E)$ be a graph and $\phi : V \to [\gamma]$ be a colouring of its vertices.
  - $\mathrm{V}(G)$ denotes the vertex set V and $\mathrm{E}(G)$ denotes the edge set E.
  - For each $i \in [\gamma]$, define $V_i = \{v \in V \mid \phi(v) = i\}$. A set of vertices $V' \subseteq V$ is called *colourful* if $|V_i \cap V'| \leq 1$ for all $i \in [\gamma]$.
- For a function $g : A \to B$, dom g denotes the set A and rng g denotes the set $B' = \{y \in B \mid \exists x \in B : g(x) = y\}$.

A *matroid* is a pair $(E, \mathcal{I})$ consisting of *ground* set E and a family $\mathcal{I}$ of subsets called *independent* sets that has the following properties.

1. $\emptyset \in \mathcal{I}$.

2. If $X \in \mathcal{I}$ and $Y \subseteq X$, then $Y \in \mathcal{I}$.
3. For any two sets $X, Y \in \mathcal{I}$ with $|X| < |Y|$, there is an element $e \in Y \setminus X$ such that $X \cup \{e\} \in \mathcal{I}$.

Because of Property 3 (also known as the *exchange property*) above, all maximal independent sets in a matroid have the same size. This number is called the *rank* of the matroid.

Given a matroid $\mathcal{M} = (E, \mathcal{I})$ and an integer $k \in \mathbb{N}$, it is easy to see that the pair $\mathcal{M}' = (E, \mathcal{I}')$ with $\mathcal{I}' = \{S \in \mathcal{I} \mid |S| \leq k\}$ is also a matroid. It is called the k-truncation of $\mathcal{M}$. Since the independent sets $S \in \mathcal{I}'$ all satisfy $|S| \leq k$, the rank of $\mathcal{M}'$ is at most k.

Let $A_{\mathcal{M}}$ be a matrix over some field F whose columns are $A_1, \ldots A_t$. Suppose that there is a injective function $\rho : E \to \{A_1, \ldots, A_t\}$ such that for any $S \subseteq E$, S is independent in $\mathcal{M}$ if and only if the set of columns $\rho(S)$ is linearly independent. In this case, the matrix $A_{\mathcal{M}}$ is called a representation for $\mathcal{M}$, and $\mathcal{M}$ is said to be representable over the field F.

Definition 1 (Linear Matroid). *A matroid $\mathcal{M}$ is called a* linear *matroid if it has a representation $A_{\mathcal{M}}$ over some field F.*

2.2 Matroids of Colourful Sets

Let $G = (V, E)$ be a graph, $\phi : V \to [\gamma]$ be a colouring of its vertices and $k \in \mathbb{N}$. Define $\mathcal{I} = \{S \subseteq V \mid S \text{ is colourful and } |S| \leq k\}$ and let $\mathcal{K} = (V, \mathcal{I})$. In the following, we show that $\mathcal{K}$ is a linear matroid with a representation that can be computed efficiently.

Consider the partition $V = V_1 \cup \cdots \cup V_\gamma$, where V_i $(i \in [\gamma])$ comprises vertices of colour i. Define $\mathcal{P} = (V, \mathcal{I}')$ where $\mathcal{I}'$ comprises all sets $S \subseteq V$ such that $|S \cap V_i| \leq 1$ $(i \in [\gamma])$.

Lemma 1. *$\mathcal{P}$ is a linear matroid. A representation $A_{\mathcal{P}}$ (over $\mathbb{F}_2$) for $\mathcal{P}$ of size $\gamma \times n$ can be computed in time $\mathrm{O}(\gamma n)$.*

Proof (Sketch). It is easy to verify that $\mathcal{P}$ is a *partition* matroid. Consider the $\gamma \times n$ matrix $A_{\mathcal{K}}$ defined by

$$A_{\mathcal{P}} = \left({e_1}^{|V_1|}, \ldots, {e_\gamma}^{|V_\gamma|} \right),$$

where e_i $(i \in [t])$ denotes the column vector with a 1 at the i^{th} coordinate and 0's everywhere else. $A_{\mathcal{P}}$ represents $\mathcal{P}$ and because the column vectors can be computed in time $\mathrm{O}(\gamma)$, the entire matrix can be constructed in time $\mathrm{O}(\gamma n)$. □

Lemma 2. *$\mathcal{K}$ is the k-truncation of $\mathcal{P}$.*

Proof. Let $S \in \mathcal{I}$. Because S is colourful, we have $|S \cap V_i| \leq 1$ $(i \in [\gamma])$, i.e. $S \in \mathcal{I}'$. Conversely, any $S \in \mathcal{I}'$ with $|S| \leq k$ is a colourful set, so $S \in \mathcal{I}$. Thus, $\mathcal{K}$ is the k-truncation of $\mathcal{P}$. □

The next proposition provides a method for computing a truncation of a matroid from its representation. It will be used later to construct a representation for $\mathcal{K}$.

Proposition 3 (Lokshtanov et al. [5], Theorem 1.1). *Let A be an $m \times n$ matrix of rank m over a field F that represents the matroid $\mathcal{M}$. For any natural number $k \leq m$, the k-truncation of $\mathcal{M}$ admits a representation A_k over $F(X)$, the field of fractions of the polynomial ring $F[X]$. This representation can be computed in $\mathrm{O}(mnk)$ operations over F, and given any set of columns of A_k, it can be determined whether they are linearly independent in $\mathrm{O}\left(m^2k^3\right)$ operations over F.*

Lemma 3. *$\mathcal{K}$ is a linear matroid of rank k that admits a representation $A_{\mathcal{K}}$ over $\mathbb{F}_2(X)$ which can be computed in time $\mathrm{O}(k\gamma n)$.*

Proof. We show that $\mathcal{K}$ is a linear matroid by computing a representation for it. Note that the ground set of $\mathcal{P}$ has $n = |V|$ elements.

Using the procedure of Lemma 1, obtain representation $A_{\mathcal{P}}$ for $\mathcal{P}$. This takes time $\mathrm{O}(\gamma n)$ and the representation is a 0-1 matrix of size $\gamma \times n$. Now use the procedure of Proposition 3 to obtain the k-truncation $A_{\mathcal{K}}$ of $\mathcal{P}$. This can be done in $\mathrm{O}(k\gamma n)$ operations over $\mathbb{F}_2$, each of which takes time $\mathrm{O}(1)$. Thus, the overall running time of the algorithm is $\mathrm{O}(k\gamma n)$. □

2.3 Representative Families

Definition 2 (q-Representative Family). *Let $p, q \in \mathbb{N}, M = (E, \mathcal{I})$ be a matroid, and $\mathcal{F} \subseteq \mathcal{I}$ be a family of independent sets of size p. A subfamily $\mathcal{F}' \subseteq \mathcal{F}$ is q-representative of F if the following statement holds. For any $X \subseteq E$ with $|X| \leq q$, if there is a set $Y \in \mathcal{F}$ such that $X \cap Y = \emptyset$ and $X \cup Y \in \mathcal{I}$, then there is a set $Y' \in \mathcal{F}'$ such that $X \cap Y' = \emptyset$ and $X \cup Y' \in \mathcal{I}$.*

Proposition 4 (Fomin et al. [1], Theorem 1.1). *Let $\mathcal{M} = (E, \mathcal{I})$ be a linear matroid of rank $p+q = k$, and $A_{\mathcal{M}}$ be a matrix over some field F that represents it. For any family $\mathcal{R} = \{S_1, \ldots, S_t\}$ of independent sets of size p in $\mathcal{M}$, there is a family $\hat{\mathcal{R}} \subseteq \mathcal{R}$ with at most $\binom{k}{p}$ sets which is q-representative of $\mathcal{R}$. The family $\hat{\mathcal{R}}$ can be found in $\mathrm{O}\left(t\left(p^\omega \binom{k}{p} + \binom{k}{p}^{\omega-1}\right)\right)$ operations over F, where $\omega < 2.373$ is the matrix multiplication exponent.*

2.4 Perfect Hash Families

Definition 3 (Perfect Hash Family). *Let $n, k \in \mathbb{N}$ with $n \geq k$. A family of functions $\mathcal{H}_{n,k} \subseteq [k]^{[n]}$ is called an (n,k)-perfect hash family if for any set $S \subseteq [n]$ with $|S| \leq k$, there is a function $f \in \mathcal{H}$ such that f is injective on S.*

Proposition 5 (Naor et al. [7], Theorem 3). *For any $n, k \in \mathbb{N}$ with $n \geq k$, there is an (n,k)-perfect hash family $\mathcal{H}_{n,k}$ of cardinality $e^{k+\mathrm{O}\left((\log k)^2\right)} \log n$ that can be computed in time $e^{k+\mathrm{O}\left((\log k)^2\right)} n \log n$.*

3 Colourful Independent Set on Interval Graphs

Here, we give two algorithms for Colourful Independent Set on interval graphs. The first uses small families of perfect hash functions to make an improvement over the algorithm of Proposition 2. The second algorithm employs a dynamic programming approach using representative families.

Definition 4 (Compact Representation). *Let G be an interval graph and $\mathcal{R} = \{L_v \mid v \in \mathrm{V}(G)\}$ be an interval representation for G. $\mathcal{R}$ is called c-compact ($c \in \mathbb{N}$) if the endpoints of every interval in R are in $\{0, \ldots, c\}$. If G has such a representation, it is called c-compact.*

Proposition 6 (van Bevern et al. [8], Observation 2). *Interval graphs of order n are n-compact.*

Proposition 7 (van Bevern et al. [8], Observation 4). *Given an adjacency list representation for an interval graph G with n vertices and m edges, a c-compact representation $\mathcal{R}$ for G that minimizes c can be computed in time $\mathrm{O}(n + m)$.*

We begin by observing that the deterministic algorithm of Proposition 2 can be improved on by using slightly more efficient constructions of hash families.

3.1 Using Hash Families

By using the hash families of Theorem 5 with the algorithm of van Bevern et al. [2], we make the following improvement on the derandomization claim of Proposition 2.

Theorem 1. *Instances (G, ϕ, k) of* Colourful Independent Set *on interval graphs can be solved in time $\left(e^{k+\mathrm{O}((\log k)^2)} \log \gamma\right) 2^k n = \mathrm{O}\left((5.44)^k (\log \gamma) n\right)$.*

Lemma 4. *Let (G, ϕ, k) be an instance of* Colourful Independent Set *on interval graphs with $\gamma = |\mathrm{rng}\ \phi|$ colours. There is a family of colouring functions $\mathcal{C}_{\phi,k} \subseteq [n] \to [k]$ of size $e^{k+\mathrm{O}((\log k)^2)} \log \gamma$ such that (G, ϕ, k) is a* YES *instance if and only if there is a function $\phi' \in \mathcal{C}_{\phi,k}$ such that (G, ϕ', k) is a* YES *instance. The family $\mathcal{C}_{\phi,k}$ can be constructed in time $e^{k+\mathrm{O}((\log k)^2)} \gamma \log \gamma$.*

Proof. Let $\mathcal{H}_{\gamma,k}$ be the perfect hash family obtained from Proposition 5 by substituting $n = \gamma$. This family is of size $e^{k+\mathrm{O}((\log k)^2)} \log \gamma$ and can be computed in time $e^{k+\mathrm{O}((\log k)^2)} \gamma \log \gamma$. Define $\mathcal{C}_{\phi,k} = \{\rho \circ \phi \mid \rho \in \mathcal{H}_{\gamma,k}\}$. Clearly, $\mathcal{C}_{\phi,k} \subseteq [n] \to [k]$. Note that $\mathcal{C}_{\phi,k}$ can be obtained by chaining ϕ to each function in $\mathcal{H}_{\gamma,k}$, and this takes time $\mathrm{O}(1)$ per function. Thus, $\mathcal{C}_{\phi,k}$ can be computed in time $e^{k+\mathrm{O}((\log k)^2)} \gamma \log \gamma$.

Suppose that (G, ϕ, k) is a YES instance and let S be a colourful independent set in G, i.e. ϕ is injective on S. Consider $R = \phi(S)$. Since $\mathcal{H}_{\gamma,k}$ is (γ, k)-perfect,

there is a function $\rho \in \mathcal{H}_{\gamma,k}$ such that ρ is injective on R. Because of this, $\rho \circ \phi \in \mathcal{C}_{\gamma,k}$ is injective on S, i.e. S is a colourful independent set with respect to $\phi' = \rho \circ \phi$. Thus, (G, ϕ', k) is a YES instance.

Conversely, if there is a function $\phi' \in \mathcal{C}_{\gamma,k}$ such that there is a colourful independent S with respect to ϕ' and $|S| \geq k$, then S is also colourful with respect to ϕ. □

The proof of Theorem 1 is now quite straightforward.

Proof (Theorem 1). Let (G, ϕ, k) be an instance of Colourful Independent Set on interval graphs. Using the construction of Lemma 4, we obtain a family of colourings $\mathcal{C}_{n,k}$ of size $e^{k+\mathrm{O}((\log k)^2)} \log \gamma$. Consider the following algorithm.

For each colouring $\phi' \in \mathcal{C}_{n,k}$, run the procedure of Proposition 1 on the instance (G, ϕ', k). If the procedure returns YES on any of the instances, then return YES. Otherwise, return NO. The correctness of the algorithm follows from Lemma 4.

Note that for each $\phi' \in \mathcal{C}_{n,k}$, $|\mathrm{rng}\ \phi'| = k$, so the instance (G, ϕ', k) has k colours. Thus, each invocation of the algorithm of Proposition 1 takes time $\mathrm{O}(2^k n)$. The overall running time of the algorithm is therefore $(e^{k+\mathrm{O}((\log k)^2)} \log \gamma) 2^k n + e^{k+\mathrm{O}((\log k)^2)} \gamma \log \gamma = \left(e^{k+\mathrm{O}((\log k)^2)} \log \gamma\right) 2^k n = \mathrm{O}\left((5.44)^k (\log \gamma) n\right)$. □

3.2 Using Representative Families

In this subsection, we employ a dynamic programming approach using representative families to obtain the following result.

Theorem 2. *Instances (G, ϕ, k) of* Colourful Independent Set *on interval graphs can be solved deterministically in time $\mathrm{O}\left((5.18)^k n^d\right)$, where ω is the matrix multiplication exponent and $d \in \mathbb{N}$ is a constant.*

Consider an instance (G, ϕ, k) of Colourful Independent Set. By Proposition 6, G has a c-compact representation with $c \leq n$. Let $\mathcal{D}$ be such a representation. Define L_v $(v \in V)$ to be the interval corresponding to v in $\mathcal{D}$ and let $l(v)$ denote the length of L_v. We say that v lies in the interval $[i, j]$ $(0 \leq i < j \leq n)$ if $L_v \subseteq [i, j]$. A set $S \subset V$ lies in $[i, j]$ if all its elements lie in $[i, j]$.

Families of Colourful Independent Sets. For $i \in [c]$ and $j \in [k]$, define $\mathcal{R}^i_j$ to be the family of all colourful independent sets in G of size exactly j in the interval $[0, i]$. Consider the matroid $\mathcal{K} = (V, \mathcal{I})$ of sets of colourful vertices defined in Subsect. 2.2. Since the sets in $\mathcal{R}^i_j$ are colourful, they are independent in $\mathcal{K}$. In what follows, we show how to efficiently compute a $(k-j)$-representative family for $\mathcal{R}^i_j$ with respect to $\mathcal{K}$.

For each $(i, j) \in (\{0\} \times [k]) \cup ([c] \times \{0\})$, the family $\mathcal{R}^i_j$ is empty, and is trivially represented by $\hat{\mathcal{R}}^i_j = \emptyset$.

Lemma 5. *Let $i \in [c]$ and $j \in [k]$. For each $r < i$ and $s \leq j$, let $\hat{\mathcal{R}}_s^r$ be a $(k-s)$-representative family for $\mathcal{R}_s^r$ with respect to $\mathcal{K}$. Define*

$$\bar{\mathcal{R}}_j^i = \hat{\mathcal{R}}_j^{i-1} \cup \left(\bigcup_{L_v \text{ ends at } i} \left[v + \hat{\mathcal{R}}_{j-1}^{i-(l(v)+1)} \right] \right), \text{ where} \tag{1}$$

$$[\mathcal{R}] = \{S \in \mathcal{R} \mid S \text{ is a colourful independent set in } G\}.$$

The family $\bar{\mathcal{R}}_j^i$ $(k-j)$-represents $\mathcal{R}_j^i$.

Proof. Let $X \subseteq V$ with $|X| \leq k-j$ such that there is a set $S \in \mathcal{R}_j^i$ with $S \cap X = \emptyset$ and $S \cup X \in \mathcal{I}$, i.e. S is a colourful independent set in G. We have the following cases. □

Case 1. S contains a vertex v such that L_v ends at i.

In this case, $S' = S \setminus \{v\}$ is a colourful independent set (in G) appearing in $S_j^{i-(l(v)+1)}$ such that $S' \cap (X \cup \{v\}) = \emptyset$ and $S' \cup (X \cup \{v\}) \in \mathcal{I}$. Let $X' = X \cup \{v\}$. Because $\hat{\mathcal{R}}_{j-1}^{i-(l(v)+1)}$ is a $(k-j+1)$-representative family for $\mathcal{R}_{j-1}^{i-(l(v)+1)}$ and $|X'| \leq k-j+1$, there is a colourful independent set $\tilde{S} \in \mathcal{R}_{j-1}^{i-(l(v)+1)}$ such that $\tilde{S} \cap X' = \emptyset$ and $\tilde{S} \cup X' \in \mathcal{I}$.

Note that $\tilde{S} \cup \{v\} \subseteq \tilde{S} \cup X' \in \mathcal{I}$ is colourful. Since $\tilde{S} \in \mathcal{R}_{j-1}^{i-(l(v)+1)}$ only contains vertices that lie in the interval $[0, i-(l(v)+1)]$, v has no neighbours in $\tilde{S}$. Because $S^* = \tilde{S} \cup \{v\}$ is a colourful independent set in G, the $[\cdot]$ operator in Eq. 1 preserves it. Thus, there is a set S^* in $\bar{\mathcal{R}}_j^i$ such that $S^* \cap X = \emptyset$ and $S^* \cup X \in \mathcal{I}$.

Case 2. S contains no vertex v such that L_v ends at i.

Observe that S lies entirely in the interval $[0, i-1]$, so it appears in $\mathcal{R}_j^{i-1}$. Since $\hat{\mathcal{R}}_j^{i-1}$ is a $(k-j)$-representative family for $\mathcal{R}_j^{i-1}$, there is a set $\tilde{S} \in \hat{\mathcal{R}}_j^{i-1}$ such that $\tilde{S} \cap X = \emptyset$ and $\tilde{S} \cup X \in \mathcal{I}$.

The procedure ComputeTable constructs a table $S[0\,..\,c][0\,..\,k]$ using the predefined procedure `ComputeRepresentativesFLPS` that computes representative families according to Proposition 4.

Lemma 6. *The procedure ComputeTable computes $S[0\,..\,c][0\,..\,k]$ such that each entry $S[i][j]$ is $(k-j)$-representative of $\mathcal{R}_j^i$. The table is computed in time $\mathrm{O}\big(c\chi k^\omega 2^{\omega k} n\big)$, where χ is the time required to perform field operations over $\mathbb{F}_2(X)$.*

Proof. In Line 1, the initialization step ensures that for each $(i,j) \in (\{0\} \times [k]) \cup ([c] \times \{0\})$, $\mathcal{R}_j^i$ is $(k-j)$-represented by $S[i][j]$. The family constructed in Line 8 is $\bar{\mathcal{R}}_j^i$, which has $t = |S[i-1][j]| + \sum_{L_v \text{ ends at } i} |S[i-(l(v)+1)][j-1]|$ sets.

Each entry $S[i'][j']$ referenced in this step was computed in a previous iteration, and (by Proposition 4) Line 9 ensures that $|S[i'][j']| \leq \binom{k}{j'}$. Thus, we

Procedure ComputeTable: compute a table of representative families

Input: $G, \mathcal{D}, A_{\mathcal{K}}, c, k$, where G is a graph, $\mathcal{D}$ is a c-compact representation and $k \in \mathbb{N}$

Output: $S[0\,..\,c][0\,..\,k]$, where each entry $S[i][j]$ $(k-j)$-represents $\mathcal{R}^i_j$

1 initialize $S[0\,..\,c][0\,..\,k]$ with $\emptyset$;
2 **for** $i \in [1\,..\,c]$ **do**
3 **for** $j \in [1\,..\,k]$ **do**
4 $S[i][j] \leftarrow S[i-1][j]$;
5 **for** $v \in \mathrm{V}(G)$ *such that* L_v *ends at* i **do**
6 $T_v \leftarrow v + S[i-(l(v)+1)][j-1]$;
7 $T_v \leftarrow \{A \in T_v \mid A \text{ is a colourful independent set}\}$;
8 $S[i][j] \leftarrow S[i][j] \cup T_v$;
9 $S[i][j] \leftarrow$ `ComputeRepresentativesFLPS`$(A_{\mathcal{K}}, S[i][j], k-j)$;
10 **return** $S[0\,..\,c][0\,..\,k]$;

have $t \leq \binom{k}{j} + n\binom{k}{j-1}$. Note that (again because of Proposition 4) $S[i][j]$ is $(k-j)$-representative of $\bar{\mathcal{R}}^i_j$, so it also $(k-j)$-represents $\mathcal{R}^i_j$.

The computation can be carried out using $\mathrm{O}\left(t\left(j^\omega\binom{k}{j} + \binom{k}{j}^{\omega-1}\right)\right)$ operations over $\mathbb{F}_2(X)$. This takes time $\mathrm{O}\left(t\left(j^\omega\binom{k}{j} + \binom{k}{j}^{\omega-1}\right)\chi\right)$, where χ is the time required to perform field operations over $\mathbb{F}_2(X)$. The expression further simplifies to $\mathrm{O}\left(\binom{k}{j}^\omega(j^\omega+1)\chi n\right) = \mathrm{O}\left(\binom{k}{j}^\omega j^\omega \chi n\right)$.

The construction of $S[i][j]$ in Lines 4–8 takes time $\mathrm{O}(nt) = \mathrm{O}\left(\binom{k}{j}^\omega j^\omega n\right)$, since it only involves copying and adding (single) elements to $\mathrm{O}(t)$ sets. Thus, the running time of the double loops is

$$\mathrm{O}\left(\chi n \sum_{i=1}^{c}\sum_{j=1}^{k}\binom{k}{j}^\omega j^\omega\right) = \mathrm{O}\left(\chi k^\omega n \sum_{i=1}^{c}\sum_{j=1}^{k}\binom{k}{j}^\omega\right).$$

By straightforward arguments, it can be shown that this expression is $\mathrm{O}\left(c\chi k^\omega 2^{\omega k} n\right)$. The other steps of the procedure take time $\mathrm{O}(n)$, so the overall running time is $\mathrm{O}\left(c\chi k^\omega 2^{\omega k} n\right)$. □

We are now ready to prove Theorem 2.

Proof (Theorem 2). Using SolveIntervalCIS, we solve (G, ϕ, k). Its correctness follows directly from Lemmas 5 and 6. A c-compact representation $\mathcal{D}$ for G can be computed using the procedure of Proposition 7 in time $\mathrm{O}\left(n^2\right)$. Then using the procedure of Lemma 3, a representation $A_{\mathcal{K}}$ for $\mathcal{K}$ can be computed in time $\mathrm{O}(k\gamma n)$ (Lemma 3). Finally, ComputeTable$(G, \mathcal{D}, A_{\mathcal{K}}, c, k)$ takes time $\mathrm{O}\left(c\chi k^\omega 2^{\omega k} n\right)$ (Lemma 5). All other operations take time $\mathrm{O}\left(n^2\right)$.

Since the entries of $A_{\mathcal{K}}$ are computed in $\mathrm{O}(k\gamma n)$ field operations over $\mathbb{F}_2$, they at most n^a bits in size for some constant $a \in \mathbb{N}$. A fact we use without proof is

that these entries can be interpreted as elements of a field $\mathbb{F}_q$ for some $q \sim 2^{n^a}$ while still maintaining the condition that $A_{\mathcal{K}}$ represents $\mathcal{K}$. Field operations over $\mathbb{F}_q$ take time $O(n^b)$ for some constant $b \in \mathbb{N}$, i.e. $\chi = O(n^b)$.

Thus, ComputeTable$(G, \mathcal{R}, A_{\mathcal{K}}, c, k)$ takes time $O(c\chi k^\omega 2^{\omega k} n) = O(ck^\omega 2^{\omega k} n^{b+1})$. The matrix multiplication algorithm of Gall [4] has $\omega \leq 2.3728639$, and because of Proposition 6, $c \leq n$. Therefore, the overall running time of SolveIntervalCIS is $O\left((5.18)^k n^d\right)$ for some constant $d \in \mathbb{N}$. □

Algorithm SolveIntervalCIS. determine if G has a colourful independent set of size k under ϕ

Input: G, ϕ, k, where G is a graph, $\phi : V(G) \to [\gamma]$ is a colouring and $k \in \mathbb{N}$

Output: `YES` if G has a colourful independent set of size k under ϕ and `NO` otherwise

```
1 D ← ComputeCompactRepresentation(G);
2 let c ∈ ℕ such that D is c-compact;
3 compute a representation A_K for the matroid of colourful sets of size at
  most k;
4 S ← ComputeTable(G, D, A_K, c, k);
5 if S[c][k] is non-empty then
6 |   return YES;
7 else
8 |   return NO;
```

4 Conclusion

We have designed improved algorithms for Colourful Independent Set via two distinct approaches:

- using improved constructions of hash families, and
- using representative families.

The algorithm of Theorem 1 is an improvement over earlier algorithms with regard to the parameter k, i.e. the number of jobs to be scheduled, as well as n, the total number of jobs. On the other hand, SolveIntervalCIS (Theorem 2), which runs in time $O\left((5.18)^k n^d\right)$ and outperforms previous algorithms in the case $k \geq d \log n$.

Using a variant of Proposition 3 (see [5], Theorem 3.15), we were able to obtain the bound $d \leq 4$. An interesting question is to see if the dependence on n in the running time of SolveIntervalCIS could be made quadratic or even linear.

References

1. Fomin, F.V., Lokshtanov, D., Panolan, F., Saurabh, S.: Efficient computation of representative families with applications in parameterized and exact algorithms. J. ACM **63**(4), 29:1–29:60 (2016)
2. Halldórsson, M.M., Karlsson, R.K.: Strip graphs: recognition and scheduling. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 137–146. Springer, Heidelberg (2006). https://doi.org/10.1007/11917496_13
3. Keil, J.M.: On the complexity of scheduling tasks with discrete starting times. Oper. Res. Lett. **12**(5), 293–295 (1992)
4. Le Gall, F.: Powers of tensors and fast matrix multiplication. In: Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation, pp. 296–303. ACM Press, Kobe (2014)
5. Lokshtanov, D., Misra, P., Panolan, F., Saurabh, S.: Deterministic truncation of linear matroids. ACM Trans. Algorithms **14**(2), 14:1–14:20 (2018)
6. Nakajima, K., Hakimi, S.L.: Complexity results for scheduling tasks with discrete starting times. J. Algorithms **3**(4), 344–361 (1982)
7. Naor, M., Schulman, L.J., Srinivasan, A.: Splitters and near-optimal derandomization. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science, pp. 182–191. IEEE Computer Society Press, Milwaukee, October 1995
8. van Bevern, R., Mnich, M., Niedermeier, R., Weller, M.: Interval scheduling and colorful independent sets. J. Sched. **18**(5), 449–469 (2015)