Path Color Switching

Supervisor: Jean-Charles Régin

Fissore Davide

MASTER INFORMATIQUE

UNIVERSITÉ CÔTE D'AZUR

Mars 30, 2023

Problem Description

> We want to generate sequences of musical "chords" with some known constraints as well as control on the complexity of the sequence.
>
> — Spotify

Problem Description

> We want to generate sequences of musical "chords" with some known constraints as well as control on the complexity of the sequence.
>
> — Spotify

Input An oriented graph whose arcs are colored with a set of colors, two nodes of the graphs s and t and a length k.

Output Set single colors to edges to find a path of length k from s to t minimizing the number of color switch.

Definitions & notations

Color switch (CS): given two adjacent arcs a_1 and a_2 colored respectively with c_1 and c_2, we have a color CS if $c_1 \neq c_2$.

Definitions & notations

Color switch (CS): given two adjacent arcs a_1 and a_2 colored respectively with c_1 and c_2, we have a color CS if $c_1 \neq c_2$.

$\mathcal{G} = (V, A)$: A directed graph where V is the set of its nodes and A is the set of its arcs.

$\mathcal{C}$: A finite set of colors.

$\mathcal{F}$: The coloring function defined as $\mathcal{F} : A \to 2^{\mathcal{C}}$.

$\mathcal{P} = (v_1, \ldots, v_k)$: A path going from v_1 to v_k.

Definitions & notations

Color switch (CS): given two adjacent arcs a_1 and a_2 colored respectively with c_1 and c_2, we have a color CS if $c_1 \neq c_2$.

$\mathcal{G} = (V, A)$: A directed graph where V is the set of its nodes and A is the set of its arcs.

$\mathcal{C}$: A finite set of colors.

$\mathcal{F}$: The coloring function defined as $\mathcal{F} : A \to 2^{\mathcal{C}}$.

$\mathcal{P} = (v_1, \ldots, v_k)$: A path going from v_1 to v_k.

$w(\mathcal{P})$: The cost of the path $\mathcal{P}$ which is given by the sum of its CS.

Problem decomposition

The problem can decomposed in small parts:

- Minimize CS on paths;
- Minimize CS on graphs.

Minimize CS on Paths

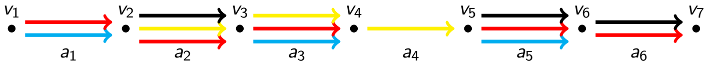


Figure: A path $\mathcal{P}$

What is the color assignation minimizing $w(\mathcal{P})$?

Algorithm

Let $\mathcal{P} = (a_1, \ldots, a_k)$ a path
Let $\mathcal{T} : A \to 2^{\mathcal{C}}$ a function such that:

- $\mathcal{T}(a_1) = \mathcal{F}(a_1)$
- $\mathcal{T}(a_i) = \mathcal{F}(a_i) \cap \mathcal{T}(a_{i-1})$ `if not empty else` $\mathcal{F}(a_i)$

Algorithm

Let $\mathcal{P} = (a_1, \ldots, a_k)$ a path
Let $\mathcal{T} : A \to 2^{\mathcal{C}}$ a function such that:

- $\mathcal{T}(a_1) = \mathcal{F}(a_1)$
- $\mathcal{T}(a_i) = \mathcal{F}(a_i) \cap \mathcal{T}(a_{i-1})$ `if not empty else` $\mathcal{F}(a_i)$

$\mathcal{H} : A \to \mathcal{C}$ the function minimizing $w(\mathcal{P})$ such that:

- $\mathcal{H}(a_k) =$ `a rnd elt from` $\mathcal{T}(a_k)$
- $\mathcal{H}(a_i) = \mathcal{H}(a_{i+1})$ `if it is in` $\mathcal{T}(a_i)$ `else` $\mathcal{T}(a_i)$`.peek()`

Example run

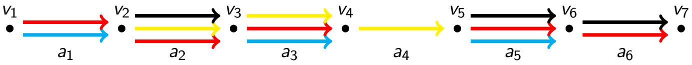


Figure: A path $\mathcal{P}$

Start to compute $\mathcal{T}(\mathcal{P})$

Example run

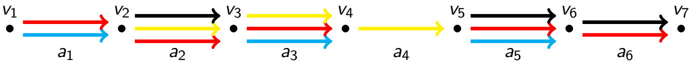


Figure: Computing $\mathcal{T}(\mathcal{P})$

$$\mathcal{T}(a_1) = \mathcal{F}(a_1)$$

Example run

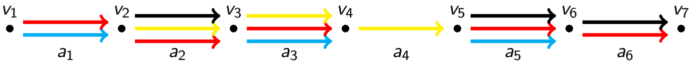


Figure: Computing $\mathcal{T}(\mathcal{P})$

$\mathcal{T}(a_2) = \mathcal{F}(a_2) \cap \mathcal{T}(a_1)$ `since not empty`

Example run

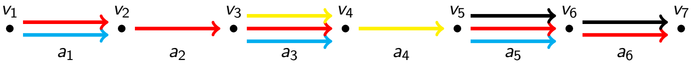


Figure: Computing $\mathcal{T}(\mathcal{P})$

$\mathcal{T}(a_2) = \mathcal{F}(a_2) \cap \mathcal{T}(a_1)$ `since not empty`

Example run

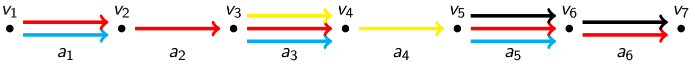


Figure: Computing $\mathcal{T}(\mathcal{P})$

$\mathcal{T}(a_3) = \mathcal{F}(a_3) \cap \mathcal{T}(a_2)$ `since not empty`

Example run

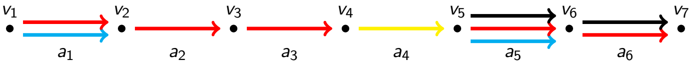


Figure: Computing $\mathcal{T}(\mathcal{P})$

$\mathcal{T}(a_3) = \mathcal{F}(a_3) \cap \mathcal{T}(a_2)$ `since not empty`

Example run

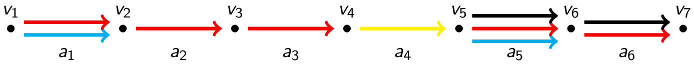


Figure: Computing $\mathcal{T}(\mathcal{P})$

$$\mathcal{T}(a_4) = \mathcal{F}(a_4) \ \texttt{since} \ \mathcal{F}(a_4) \cap \mathcal{T}(a_3) = \varnothing$$

Example run

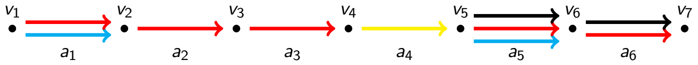


Figure: Computing $\mathcal{T}(\mathcal{P})$

$$\mathcal{T}(a_5) = \mathcal{F}(a_5) \ \texttt{since} \ \mathcal{F}(a_5) \cap \mathcal{T}(a_4) = \varnothing$$

Example run

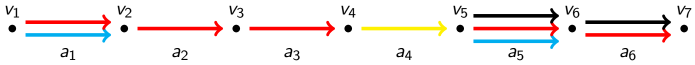


Figure: Computing $\mathcal{T}(\mathcal{P})$

$\mathcal{T}(a_6) = \mathcal{F}(a_6) \cap \mathcal{T}(a_5)$ `since not empty`

Example run

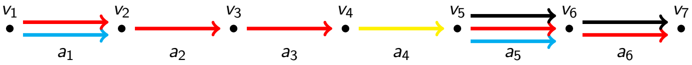


Figure: Computing $\mathcal{T}(\mathcal{P})$

Start to compute $\mathcal{H}(\mathcal{P})$

Example run

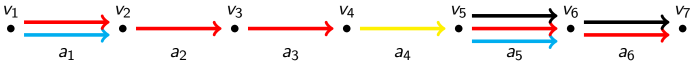


Figure: Computing $\mathcal{H}(\mathcal{P})$

$$\mathcal{H}(a_6) = black$$

Example run

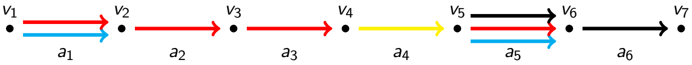


Figure: Computing $\mathcal{H}(\mathcal{P})$

$\mathcal{H}(a_6) = black$

Example run

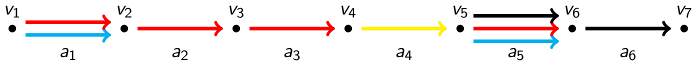


Figure: Computing $\mathcal{H}(\mathcal{P})$

$$\mathcal{H}(a_5) = black \ \ \texttt{since} \ \ black \in \mathcal{T}(a_5)$$

Example run

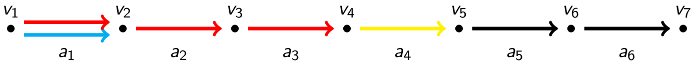


Figure: Computing $\mathcal{H}(\mathcal{P})$

$$\mathcal{H}(a_5) = black \ \ \texttt{since} \ \ black \in \mathcal{T}(a_5)$$

Example run

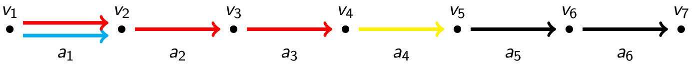


Figure: Computing $\mathcal{H}(\mathcal{P})$

Nothing to do for a_4, a_3 and a_2 since they only have 1 color

Example run

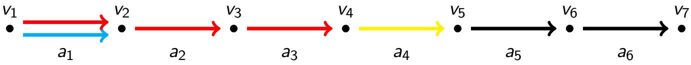


Figure: Computing $\mathcal{H}(\mathcal{P})$

$$\mathcal{H}(a_1) = \mathcal{H}(a_2) \texttt{ since } red \in \mathcal{T}(a_1)$$

Example run

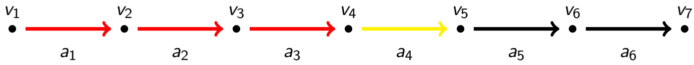


Figure: Minimum cost assignation

Example run

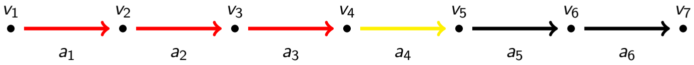


Figure: Minimum cost assignation

$$w(\mathcal{P}) = 2$$

Proof sketch

Algo Part 1.

Induction proof on the length k of $\mathcal{P}$.

$\square$

Proof sketch

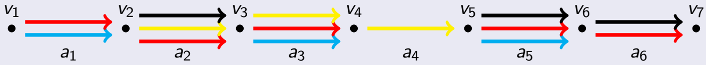
Algo Part 1.

Induction proof on the length k of $\mathcal{P}$.
If $k = 1$ then $w(\mathcal{P}) = 0$ which is optimal. $\square$

Proof sketch

Algo Part 1.

Induction proof on the length k of $\mathcal{P}$.
We suppose the algo to be true for an arbitrary length k.
If $\mathcal{F}(a_k) \cap \mathcal{F}(a_{k+1}) = \varnothing$

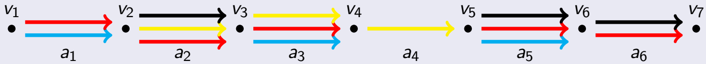


$\square$

Proof sketch

Algo Part 1.

Induction proof on the length k of $\mathcal{P}$.
We suppose the algo to be true for an arbitrary length k.
If $\mathcal{T}(a_k) \cap \mathcal{F}(a_{k+1}) \neq \varnothing$

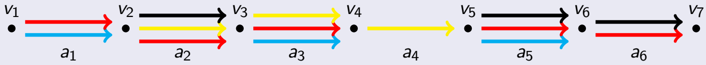


$\square$

Proof sketch

Algo Part 1.

Induction proof on the length k of $\mathcal{P}$.
We suppose the algo to be true for an arbitrary length k.
If $\mathcal{T}(a_k) \cap \mathcal{F}(a_{k+1}) = \varnothing$ and $\mathcal{F}(a_k) \cap \mathcal{F}(a_{k+1}) \neq \varnothing$



Proof sketch

Algo Part 1.

Induction proof on the length k of $\mathcal{P}$.
Done $\square$

Algo Part 2.

The number of CS inside $\mathcal{T}$ is the same as the number of CS inside $\mathcal{H}$. $\square$

Time Complexity

The algo is made by two sub-procedures:

Recall the first part:

- $\mathcal{T}(a_1) = \mathcal{F}(a_1)$
- $\mathcal{T}(a_i) = \mathcal{F}(a_i) \cap \mathcal{T}(a_{i-1})$ `if not empty else` $\mathcal{F}(a_i)$

Complexity:

- First part : $\mathcal{O}(k * |\mathcal{C}|)$

Time Complexity

The algo is made by two sub-procedures:

Recall the second part:

- $\mathcal{H}(a_k) = \texttt{a rnd elt from } \mathcal{T}(a_k)$
- $\mathcal{H}(a_i) = \mathcal{H}(a_{i+1}) \texttt{ if it is in } \mathcal{T}(a_i) \texttt{ else } \mathcal{T}(a_i)\texttt{.peek()}$

Complexity:

- First part : $\mathcal{O}(k * |\mathcal{C}|)$
- Second part : $\mathcal{O}(k * \log |\mathcal{C}|)$

Time Complexity

Complexity:

- First part : $\mathcal{O}(k * |\mathcal{C}|)$
- Second part : $\mathcal{O}(k * \log |\mathcal{C}|)$

Global complexity: $\mathcal{O}(k * |\mathcal{C}|)$.
This complexity is optimal wrt the entry of the problem.

Minimize CS in Graph

Strategy: Use the *MDD* data structure

A state of a *MDD* is:

$$\{\text{name: String, cost: Int, colors: Set of Colors}\}$$

Algo:

- The root $= \{\text{name: } s, \text{ cost: } 0, \text{ colors: } \mathcal{C}\}$
- Construction of the layer $\mathcal{L}_{i+1}$: $\forall \texttt{state} \in \mathcal{L}_i, v \in \texttt{succ(state.name)}, \mathcal{L}_{i+1}.add(\{\text{name: } v, \text{ cost: } state.cost + (\mathcal{F}(\texttt{state.name}, v) \cap \texttt{state.colors}) = \varnothing ? 1 : 0, \text{ colors: } \mathcal{F}(\texttt{state.name}, v) \cap \texttt{state.colors} = \varnothing\} ? \mathcal{F}(\texttt{state.name}, v) : \mathcal{F}(\texttt{state.name}, v) \cap \texttt{state.colors})$

MDD reduction

Proof and Complexity

The *allDiff* variant

My Implementation

Another representation of the problem

Benchmark

- Sample of Spotify

Conclusion

- Perspective