Complexity of Splits Reconstruction for Low-Degree Trees

Serge Gaspers [1] Mathieu Liedloff [2]
Maya Stein [3] Karol Suchan [4,5]

[1]Institute of Information Systems, Vienna University of Technology
Vienna, Austria

[2]Laboratoire d'Informatique Fondamentale d'Orléans Université d'Orléans,
Orléans, France

[3]CMM, Universidad de Chile
Santiago, Chile

[4]FIC, Universidad Adolfo Ibáñez
Santiago, Chile

[5]WMS, AGH - University of Science and Technology
Krakow, Poland

WG 2011

Outline

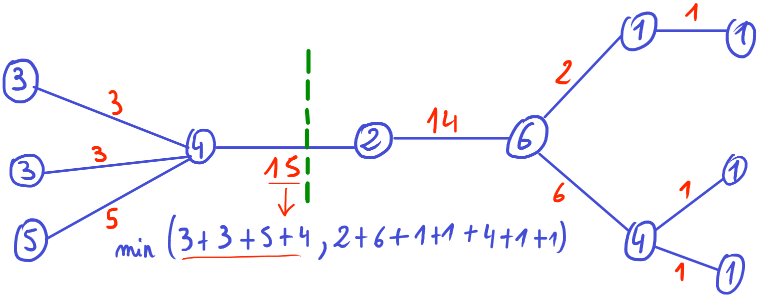
Introduction

The splits reconstruction problem

Definition

Let $T = (V, E)$ be a tree and $\omega = V \to \mathbb{N}$ be a weight function. The split of an edge e is the minimum of $\Omega(T_1)$ and $\Omega(T_2)$ where

- T_1 and T_2 are the two trees obtained by deleting e from T
- $\Omega(T_i) = \sum_{v \in T_i} \omega(v)$



$$\mathcal{S}(T) = \{3, 3, 5, 15, 14, 2, 1, 6, 1, 1\}$$

→ We denote the multiset of splits of T by $\mathcal{S}(T)$.

The splits reconstruction problem

The problem :

WEIGHTED SPLITS RECONSTRUCTION (WSR)

Input : A set V of n vertices, a weight function ω, and a multiset $\mathcal{S}$ of integers.

Question : Is there a tree T whose multiset of splits is $\mathcal{S}$?

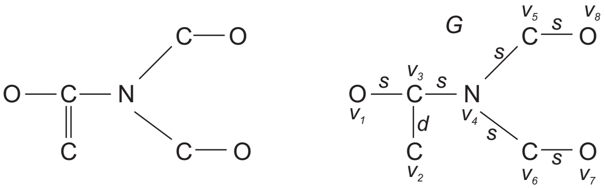
WSR_k : Same problem, but T is of maximum degree at most k.

$\rightarrow$ The problem is to construct a tree being consistent with both weights and splits.

Applications

Applications in chemistry :

- Molecules are modeled by graphs in order to study physical properties.
- Chemical graphs : Vertices represent atoms and edges the chemical bonds.



A chemical structure and its corresponding labeled graph version.

M. Dehmer, N. Barbarini, K. Varmuza, A. Grabe
Novel topological descriptors for analyzing biological networks
BMC Structural Biology 2010

Applications

Applications in chemistry :

- Within the area of *quantitative structure-activity relationship*, several structural measures of chemical graphs were identified that quantitatively correlate with some defined process (like biological activity or chemical reactivity).
- Widely known example of such measure is the *Wiener index* : the sum of the distances between each pair of vertices.
- Other measures were introduced and investigated.

Known results

In 2000, Goldman et al. (SODA 2000) introduced the SPLITS RECONSTRUCTION problem and recall that the Wiener index of a tree T on n vertices with unit weights is $\sum_{s \in \mathcal{S}(T)} s \cdot (n - s)$.

As it is not reasonable to construct chemical trees with arbitrary high vertex degrees, Li and Zhang (2004) studied the restriction to maximum degree at most 4 (SR_4) and show its NP-completeness. They provided an exponential-time algorithm which creates weighted vertices in intermediate steps.

Our results

Since it was proved that SR_4 is NP-complete,
and SR_2 is trivially polynomial,
it is of interest to know the computational complexity of SR_3.

$\rightarrow$ We close this gap by showing its NP-completeness.

(The problem is also NP-complete for caterpillars with unbounded hairs.)

Main result : WSR_2 is strongly NP-complete.

We also provide a polynomial-time algorithm solving WSR_2, assuming that the number of distinct vertex weights is constant-bounded.

Strongly NP-completeness of WSR_2

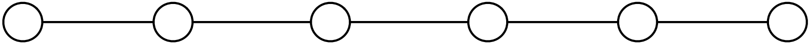
The weighted splits reconstruction problem on paths

We first restrict our focus to WSR_2 :

WEIGHTED SPLITS RECONSTRUCTION for paths.

Splits : 1, 5, 6, 10, 11

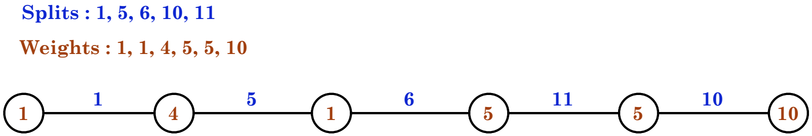
Weights : 1, 1, 4, 5, 5, 10



The weighted splits reconstruction problem on paths

We first restrict our focus to WSR_2 :

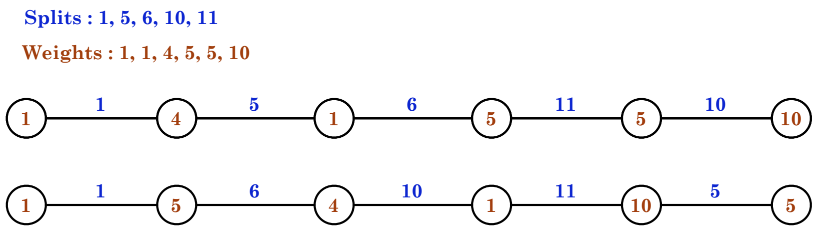
Weighted Splits Reconstruction for paths.



The weighted splits reconstruction problem on paths

We first restrict our focus to WSR_2 :

WEIGHTED SPLITS RECONSTRUCTION for paths.



Strongly NP-completeness of WSR_2

To show the NP-completeness of WEIGHTED SPLITS RECONSTRUCTION for paths, we make a reduction from :

SCHEDULING WITH COMMON DEADLINES (SCD)

Input : A set of n jobs with integer lengths and n deadlines.
Question : Can the jobs be scheduled on two processors such that at each deadline a processor finishes a job, and processors are never idle between the execution of two jobs ?

Intuition : Simulate the two processors by considering the sub-path starting from the left endpoint and the sub-path starting from the right endpoint.

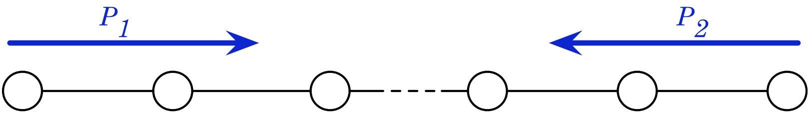
Strongly NP-completeness of WSR_2

To show the NP-completeness of WEIGHTED SPLITS RECONSTRUCTION for paths, we make a reduction from :

SCHEDULING WITH COMMON DEADLINES (SCD)

Input : A set of n jobs with integer lengths and n deadlines.
Question : Can the jobs be scheduled on two processors such that at each deadline a processor finishes a job, and processors are never idle between the execution of two jobs ?

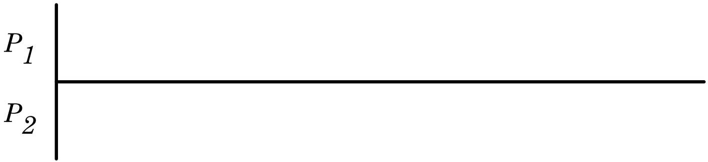
Intuition : Simulate the two processors by considering the sub-path starting from the left endpoint and the sub-path starting from the right endpoint.



Strongly NP-completeness of WSR_2

Scheduling With Common Deadlines (SCD)

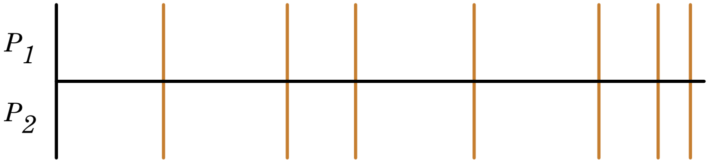
Input : A set of n jobs with integer lengths and n deadlines.
Question : Can the jobs be scheduled on two processors such that at each deadline a processor finishes a job, and processors are never idle between the execution of two jobs ?



Strongly NP-completeness of WSR_2

Scheduling With Common Deadlines (SCD)

Input : A set of n jobs with integer lengths and n deadlines.
Question : Can the jobs be scheduled on two processors such that at each deadline a processor finishes a job, and processors are never idle between the execution of two jobs ?

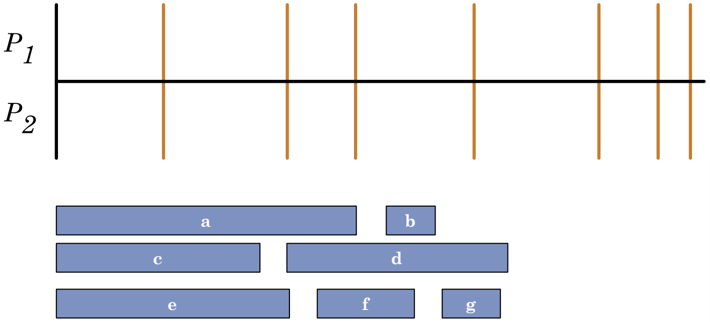


One may imagine that we want to satisfy delivery deadlines and avoid using any warehouse space to store a product between its fabrication and the delivery date.

Strongly NP-completeness of WSR_2

Scheduling With Common Deadlines (SCD)

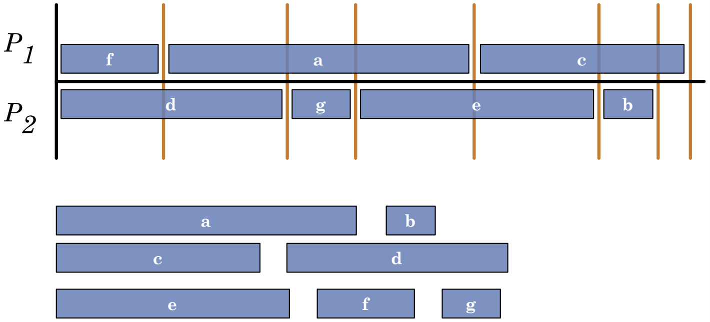
Input : A set of n jobs with integer lengths and n deadlines.
Question : Can the jobs be scheduled on two processors such that at each deadline a processor finishes a job, and processors are never idle between the execution of two jobs ?



Strongly NP-completeness of WSR_2

Scheduling With Common Deadlines (SCD)

Input : A set of n jobs with integer lengths and n deadlines.
Question : Can the jobs be scheduled on two processors such that at each deadline a processor finishes a job, and processors are never idle between the execution of two jobs ?



Strongly NP-completeness of WSR_2

1. $\mathrm{SCD} \leq_p \mathrm{WSR}_2$

(Remark : Clearly all these problems belongs to NP.)

Strongly NP-completeness of WSR_2

1. $\mathrm{SCD} \leq_p \mathrm{WSR}_2$

2. SCD is NP-complete

(Remark : Clearly all these problems belongs to NP.)

Strongly NP-completeness of WSR_2

1. $\mathrm{SCD} \leq_p \mathrm{WSR}_2$

"easy"

2. SCD is NP-complete

"much harder"

(Remark : Clearly all these problems belongs to NP.)

1. $\mathrm{SCD} \leq_p \mathrm{WSR}_2$

Given an instance $(j_1, \ldots, j_n; d_1 \leq \cdots \leq d_n)$ for SCD (j_i's represent the job lengths ; d_i's represent the deadlines), we construct an instance for WSR_2 as follows :

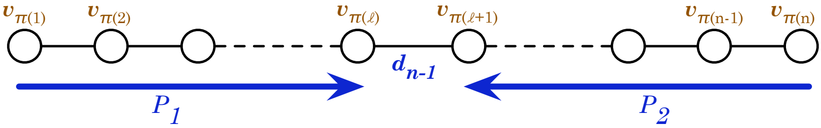
- For each job j_i, $1 \leq i \leq n$,
  create a vertex v_i with weight $\omega(v_i) = j_i$.
- For each deadline d_i, $1 \leq i \leq n-1$, create a split d_i.

W.l.o.g. we assume that $\sum_{i=1}^{n} j_i = d_{n-1} + d_n$

1. $\mathrm{SCD} \leq_p \mathrm{WSR}_2$

"$\Leftarrow$"

Suppose the path $P = (v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(n)})$ is a solution to WSR_2. Say $\{v_{\pi(\ell)}, v_{\pi(\ell+1)}\}$ is the edge associated to the split d_{n-1}.



We construct a solution for SCD by assigning the jobs $j_{\pi(1)}, j_{\pi(2)}, \ldots, j_{\pi(\ell)}$ to processor P_1, and the jobs $j_{\pi(n)}, j_{\pi(n-1)}, \ldots, j_{\pi(\ell+2)}, j_{\pi(\ell+1)}$ to processor P_2, in this order.

Note that then, one of the jobs $j_{\pi(\ell)}$, $j_{\pi(\ell+1)}$ ends at d_{n-1}, and the other at $-d_{n-1} + \sum_{i=1}^{n} j_i = d_n$, which is as desired.

1. $\mathrm{SCD} \leq_p \mathrm{WSR}_2$

"$\Rightarrow$"

On the other hand, if SCD has a solution, then WSR_2 has a solution as well, because the previous construction is easily inverted.

Visually, the list of jobs of P_2 is reversed and appended to the list of jobs of P_1. Job lengths correspond to vertex weights and deadlines correspond to splits.

(The last deadline where a job from P_1 finishes is *merged* with the last deadline where a job from P_2 finishes.)

Thus,

Theorem

SCD is polynomial-time-reducible to WSR_2.

Strongly NP-completeness of WSR_2

1. $\mathrm{SCD} \leq_p \mathrm{WSR}_2$

2. SCD is NP-complete

2. SCD is NP-complete

To show that SCD is NP-complete, we give a polynomial-time reduction from dNMTS :

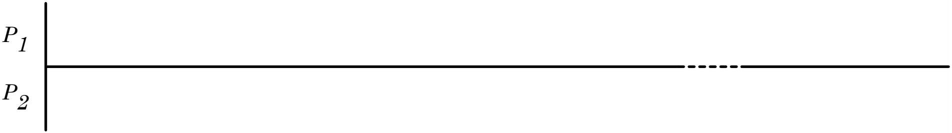
NUMERICAL MATCHING WITH TARGET SUMS (NMTS)

Input : 3 multisets A, B, and $S = \{s_1, \ldots, s_m\}$ of size m from $\mathbb{N}$.
Question : Can $A \cup B$ be partitioned into m disjoint sets $C_1, C_2, \ldots, C_m$, each containing exactly one element from each of A and B, such that $\sum_{c \in C_i} c = s_i$, $1 \le i \le m$?

- NMTS : [SP17] in Garey-Johnson
- dNMTS : all integers in $A \cup B \cup S$ are pairwise distinct
- dNMTS : strongly NP-hard [Hulett, Will, Woeginger, 2008]

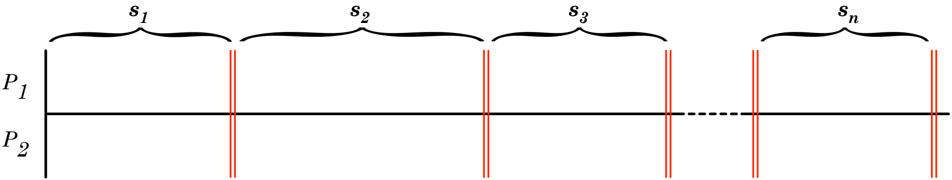
2. SCD is NP-complete

The whole (but incomplete) picture :



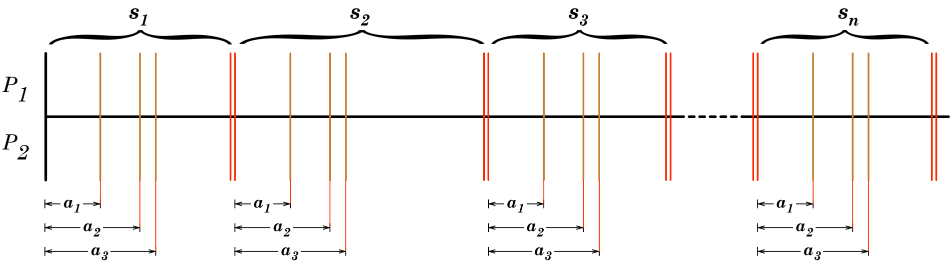
2. SCD is NP-complete

The whole (but incomplete) picture :



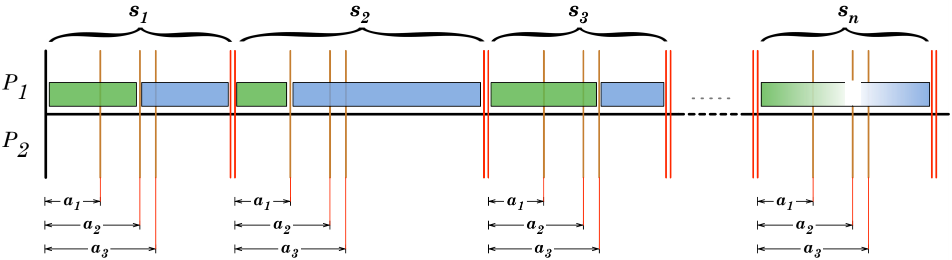
2. SCD is NP-complete

The whole (but incomplete) picture :



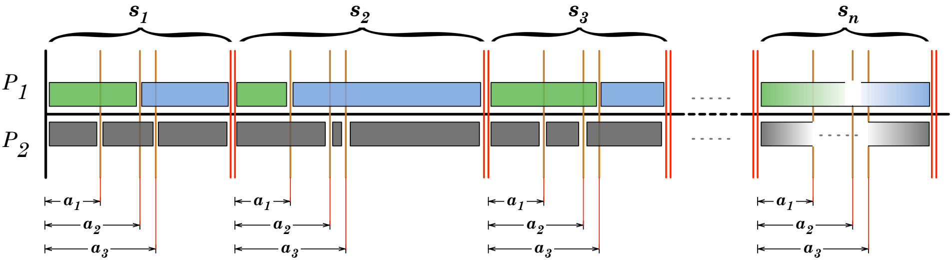
2. SCD is NP-complete

The whole (but incomplete) picture :



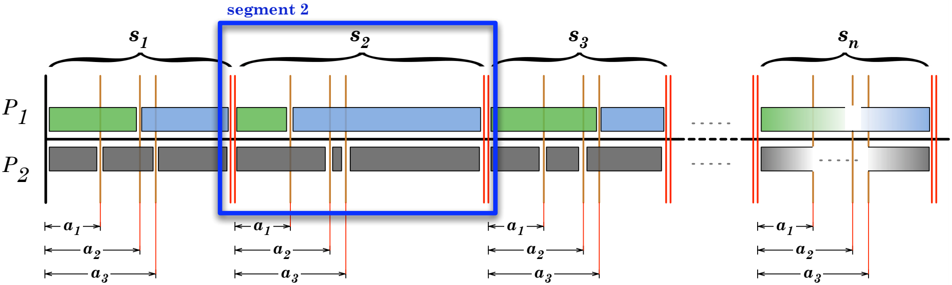
2. SCD is NP-complete

The whole (but incomplete) picture :



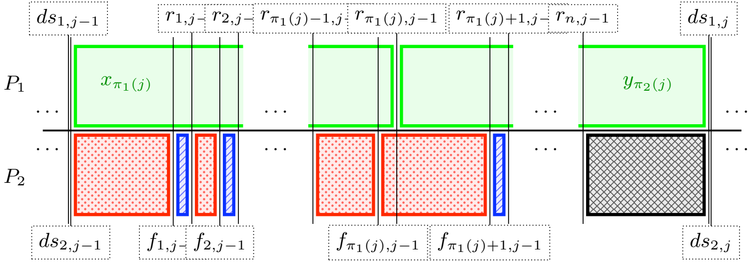
2. SCD is NP-complete

The whole (but incomplete) picture :



2. SCD is NP-complete

The full details of a segment :



- More deadlines ...
- ... and thus more jobs.

2. SCD is NP-complete

The reduction from dNMTS needs to scale the numbers of the given instance to ensure some properties :

for $i \in \{1, \ldots, n-1\}$,

$x_i := 2 \cdot (a_i + (b_m + 2))$,

$y_i := 2 \cdot (b_i + 3 \cdot (b_m + 2))$,

$z_i := 2 \cdot (s_i + 4 \cdot (b_m + 2))$, and

$x_n := 2 \cdot (a_m + 1 + (b_m + 2))$,

$y_n := 2 \cdot (b_m + 1 + 3 \cdot (b_m + 2))$,

$z_n := 2 \cdot (a_m + b_m + 2 + 4 \cdot (b_m + 2))$.

2. SCD is NP-complete

Property

Each element of $X \cup Y \cup Z$ is an even positive integer.

Property

For every $i \in \{1, \ldots, n-1\}$, we have that $x_i < x_{i+1}$, that $y_i < y_{i+1}$, and that $z_i < z_{i+1}$.

Property

For every $i \in \{1, \ldots, n\}$, we have

$$2 \cdot b_m + 4 \leq x_i \leq 4 \cdot b_m + 4,$$
$$6 \cdot b_m + 12 \leq y_i \leq 8 \cdot b_m + 14, \text{ and}$$
$$8 \cdot b_m + 16 \leq z_i \leq 12 \cdot b_m + 18.$$

The last property implies that $y_1 > x_n$, that $z_1 > y_n$, and that $2 \cdot y_1 > z_n$.

2. SCD is NP-complete

Property

If k and ℓ are integers such that $x_k + y_\ell = z_n$, then $k = \ell = n$.

Property

Let $p, q \in X \cup Y$, $p \leq q$, and $z \in Z$.
If $p + q = z$, then $p \in X$ and $q \in Y$.

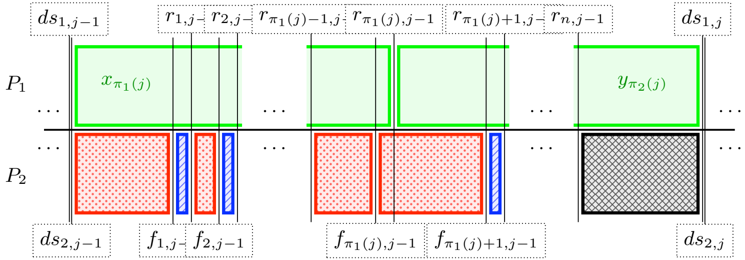
By previous properties :

- the sum of any two X-elements is smaller than any element of Z
- the sum of any two Y-elements is larger than any element of Z

2. SCD is NP-complete

Then we create the following deadlines :

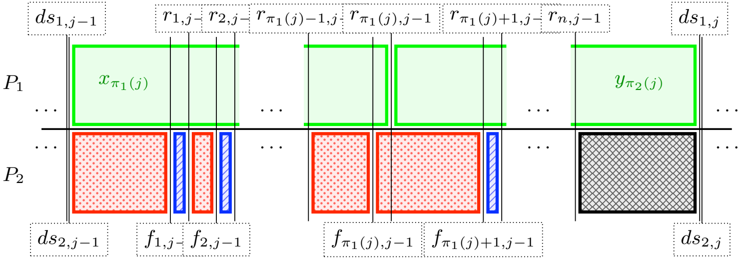
- real deadlines : $r_{i,j} := x_i + \sum_{k=1}^{j} z_k$, for each $j \in \{0, \ldots, n-1\}$ and each $i \in \{1, \ldots, n\}$,
- fake deadlines : $f_{i,j} := r_{i,j} - 1$, for each $j \in \{0, \ldots, n-1\}$ and each $i \in \{1, \ldots, n\}$, and
- sum deadlines : two deadlines $ds_{1,j} := ds_{2,j} := \sum_{k=1}^{j} z_k$, for each $j \in \{1, \ldots, n\}$.



2. SCD is NP-complete

And we create the jobs with the following lengths :

- green x-jobs : x_i, for each $i \in \{1, \ldots, n\}$,
- green y-jobs : y_i, for each $i \in \{1, \ldots, n\}$,
- blue jobs : $n \cdot (n-1)$ times a job of length 1,
- red fill jobs : $n-1$ times a job of length $x_i - 1 - x_{i-1}$, for each $i \in \{1, \ldots, n\}$,
- red overlap jobs : $x_i - x_{i-1}$, for each $i \in \{1, \ldots, n\}$,
- black fill jobs : $z_i - x_n$ for $i \in \{1, \ldots, n-1\}$, and
- a black overlap job : $z_n - x_n + 1$.



2. SCD is NP-complete

Afterwards we are able to prove a collection of claims which together show the NP-completeness of SCD.

Theorem

$$\mathrm{dNMTS} \leq_p \mathrm{SCD} \leq_p \mathrm{WSR}_2$$

The problem WSR_2 is strongly NP-complete.

An algorithm for WSR_2 with few distinct vertex weights

An algorithm for WSR_2 with few distinct vertex weights

We just showed that WSR_2 is strongly NP-complete.

Assume that we face an instance with, say k, distinct vertex weights.

Is it possible to design a polynomial-time algorithm, assuming k is a constant ?

Main idea : Dynamic Programming

An algorithm for WSR_2 with few distinct vertex weights

We just showed that WSR_2 is strongly NP-complete.

Assume that we face an instance with, say k, distinct vertex weights.

Is it possible to design a polynomial-time algorithm, assuming k is a constant ?

Main idea : Dynamic Programming

An algorithm for WSR_2 with few distinct vertex weights

Let $k = |\{\omega(v) \ : \ v \in V\}|$ be the number of distinct vertex weights.

Let $w_1 < w_2 < \cdots < w_k$ denote the distinct vertex weights and $m_1, m_2, \ldots, m_k$ denote their respective multiplicities, i.e. :

$$m_i = |\{v \in V \ : \ \omega(v) = w_i\}|.$$

Let $\mathcal{S} = \{s_1, s_2, \ldots, s_{n-1}\}$ be the multiset of splits, with $s_1 \leq s_2 \leq \cdots \leq s_{n-1}$.

An algorithm for WSR_2 with few distinct vertex weights

Boolean table :

$$\mathrm{T}[p, W_L, W_R, v_1, v_2, \ldots, v_k]$$

being defined for each :

- integer p , $1 \leq p \leq n-1$
- split $W_L \in \mathcal{S}$
- split $W_R \in \mathcal{S}$
- $v_1 \in \{0, 1, \ldots, m_1\}$
- ...
- $v_k \in \{0, 1, \ldots, m_k\}$

set to true iff there is an assignement of the splits $s_1, s_2, \ldots, s_p$ to the ℓ leftmost edges and the r rightmost edges of the path, s.t. :

- $p = \ell + r$
- v_1 weights w_1, v_2 weights w_2, ..., v_k weights w_k are assigned to the ℓ leftmost and the r rightmost vertices s.t. each split assigned to the left (resp. to the right) part of the path corresponds to the sum of the vertex weights assigned to vertices to the left (resp. to the right) of this split
- W_L is equal to the value of the ℓ^{th} split from the left and W_R is equal to the r^{th} split from the right

An algorithm for WSR_2 with few distinct vertex weights

Boolean table :

$$\mathrm{T}[p, W_L, W_R, v_1, v_2, \ldots, v_k]$$

being defined for each :

- integer p , $1 \leq p \leq n-1$
- split $W_L \in \mathcal{S}$
- split $W_R \in \mathcal{S}$
- $v_1 \in \{0, 1, \ldots, m_1\}$
- ...
- $v_k \in \{0, 1, \ldots, m_k\}$

set to `true` iff there is an assignement of the splits $s_1, s_2, \ldots, s_p$ to the ℓ leftmost edges and the r rightmost edges of the path, s.t. :

- $p = \ell + r$
- v_1 weights w_1, v_2 weights w_2, ..., v_k weights w_k are assigned to the ℓ leftmost and the r rightmost vertices s.t. each split assigned to the left (resp. to the right) part of the path corresponds to the sum of the vertex weights assigned to vertices to the left (resp. to the right) of this split
- W_L is equal to the value of the ℓ^{th} split from the left and W_R is equal to the r^{th} split from the right

An algorithm for WSR_2 with few distinct vertex weights

Boolean table :

$$\mathrm{T}[p, W_L, W_R, v_1, v_2, \ldots, v_k]$$

being defined for each :

- integer p , $1 \leq p \leq n-1$
- split $W_L \in \mathcal{S}$
- split $W_R \in \mathcal{S}$
- $v_1 \in \{0, 1, \ldots, m_1\}$
- ...
- $v_k \in \{0, 1, \ldots, m_k\}$

set to `true` iff there is an assignement of the splits $s_1, s_2, \ldots, s_p$ to the ℓ leftmost edges and the r rightmost edges of the path, s.t. :

- $p = \ell + r$
- v_1 weights w_1, v_2 weights w_2, ..., v_k weights w_k are assigned to the ℓ leftmost and the r rightmost vertices s.t. each split assigned to the left (resp. to the right) part of the path corresponds to the sum of the vertex weights assigned to vertices to the left (resp. to the right) of this split
- W_L is equal to the value of the ℓ^{th} split from the left and W_R is equal to the r^{th} split from the right

An algorithm for WSR_2 with few distinct vertex weights

Boolean table :

$$\mathrm{T}[p, W_L, W_R, v_1, v_2, \ldots, v_k]$$

being defined for each :

- integer p , $1 \leq p \leq n-1$
- split $W_L \in \mathcal{S}$
- split $W_R \in \mathcal{S}$
- $v_1 \in \{0, 1, \ldots, m_1\}$
- ...
- $v_k \in \{0, 1, \ldots, m_k\}$

set to `true` iff there is an assignement of the splits $s_1, s_2, \ldots, s_p$ to the ℓ leftmost edges and the r rightmost edges of the path, s.t. :

- $p = \ell + r$
- v_1 weights w_1, v_2 weights w_2, ..., v_k weights w_k are assigned to the ℓ leftmost and the r rightmost vertices s.t. each split assigned to the left (resp. to the right) part of the path corresponds to the sum of the vertex weights assigned to vertices to the left (resp. to the right) of this split
- W_L is equal to the value of the ℓ^{th} split from the left and W_R is equal to the r^{th} split from the right

An algorithm for WSR_2 with few distinct vertex weights

Boolean table :

$$\mathrm{T}[p, W_L, W_R, v_1, v_2, \ldots, v_k]$$

being defined for each :

- integer p , $1 \leq p \leq n-1$
- split $W_L \in \mathcal{S}$
- split $W_R \in \mathcal{S}$
- $v_1 \in \{0, 1, \ldots, m_1\}$
- ...
- $v_k \in \{0, 1, \ldots, m_k\}$

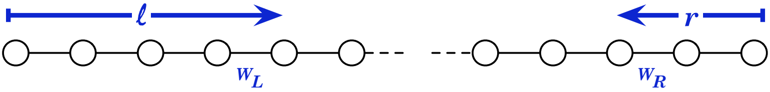
set to `true` iff there is an assignement of the splits $s_1, s_2, \ldots, s_p$ to the ℓ leftmost edges and the r rightmost edges of the path, s.t. :

- $p = \ell + r$
- v_1 weights w_1, v_2 weights w_2, ..., v_k weights w_k are assigned to the ℓ leftmost and the r rightmost vertices s.t. each split assigned to the left (resp. to the right) part of the path corresponds to the sum of the vertex weights assigned to vertices to the left (resp. to the right) of this split
- W_L is equal to the value of the ℓ^{th} split from the left and W_R is equal to the r^{th} split from the right

An algorithm for WSR_2 with few distinct vertex weights

Intuitively, the algorithm assigns splits and weights by starting from both endpoints of the path and trying to meet these two sub-solutions.

$$\mathrm{T}[p, W_L, W_R, v_1, v_2, \ldots, v_k]$$



Base case. $\mathrm{T}[0, W_L, W_R, v_1, v_2, \ldots, v_k]$ is `true` if $W_L = W_R = v_1 = v_2 = \ldots = v_k = 0$ and `false` otherwise.

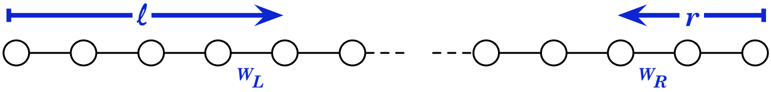
Remaining entries are computed by increasing values of p using the recurrence :

$$\mathrm{T}[p, W_L, W_R, v_1, v_2, \ldots, v_k] = \bigvee_{i=1}^{k} \begin{cases} \mathrm{T}[p-1, W_L - w_i, W_R, v_1, v_2, \ldots, v_{i-1}, \\ \quad v_i - 1, v_{i+1}, v_{i+2}, \ldots, v_k] \quad \vee \\ \mathrm{T}[p-1, W_L, W_R - w_i, v_1, v_2, \ldots, v_{i-1}, \\ \quad v_i - 1, v_{i+1}, v_{i+2}, \ldots, v_k] \end{cases}$$

An algorithm for WSR_2 with few distinct vertex weights

Intuitively, the algorithm assigns splits and weights by starting from both endpoints of the path and trying to meet these two sub-solutions.

$$\mathrm{T}[p, W_L, W_R, v_1, v_2, \ldots, v_k]$$



Base case. $\mathrm{T}[0, W_L, W_R, v_1, v_2, \ldots, v_k]$ is `true` if $W_L = W_R = v_1 = v_2 = \ldots = v_k = 0$ and `false` otherwise.

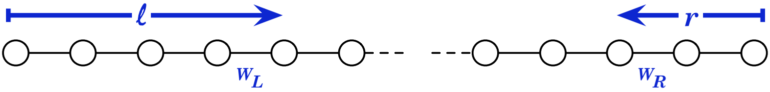
Remaining entries are computed by increasing values of p using the recurrence :

$$\mathrm{T}[p, W_L, W_R, v_1, v_2, \ldots, v_k] = \bigvee_{i=1}^{k} \begin{cases} \mathrm{T}[p-1, W_L - w_i, W_R, v_1, v_2, \ldots, v_{i-1}, \\ \quad v_i - 1, v_{i+1}, v_{i+2}, \ldots, v_k] \quad \vee \\ \mathrm{T}[p-1, W_L, W_R - w_i, v_1, v_2, \ldots, v_{i-1}, \\ \quad v_i - 1, v_{i+1}, v_{i+2}, \ldots, v_k] \end{cases}$$

An algorithm for WSR_2 with few distinct vertex weights

Intuitively, the algorithm assigns splits and weights by starting from both endpoints of the path and trying to meet these two sub-solutions.

$$\mathtt{T}[p, W_L, W_R, v_1, v_2, \ldots, v_k]$$



Base case. $\mathtt{T}[0, W_L, W_R, v_1, v_2, \ldots, v_k]$ is `true` if $W_L = W_R = v_1 = v_2 = \ldots = v_k = 0$ and `false` otherwise.

Remaining entries are computed by increasing values of p using the recurrence :

$$\mathtt{T}[p, W_L, W_R, v_1, v_2, \ldots, v_k] = \bigvee_{i=1}^{k} \begin{cases} \mathtt{T}[p-1, W_L - w_i, W_R, v_1, v_2, \ldots, v_{i-1}, \\ \qquad v_i - 1, v_{i+1}, v_{i+2}, \ldots, v_k] \quad \vee \\ \mathtt{T}[p-1, W_L, W_R - w_i, v_1, v_2, \ldots, v_{i-1}, \\ \qquad v_i - 1, v_{i+1}, v_{i+2}, \ldots, v_k] \end{cases}$$

An algorithm for WSR_2 with few distinct vertex weights

The final result is computed by evaluating :

$$\bigvee_{\substack{W_L, W_R \in \mathcal{S} \\ i \in \{1,2,\ldots,k\} \\ (W_L \le w_i + W_R) \,\wedge\, (W_R \le w_i + W_L)}} \mathtt{T}[|\mathcal{S}|, W_L, W_R, m_1, m_2, \ldots, m_{i-1}, m_i - 1, m_{i+1}, m_{i+2}, \ldots, m_k]$$

Theorem

WSR_2 can be solved in time $O(n^{k+3} \cdot k)$ where k is the number of distinct vertex weights of any input instance $(V, \omega, \mathcal{S})$ and n is the number of vertices.

An algorithm for WSR_2 with few distinct vertex weights

The final result is computed by evaluating :

$$\bigvee_{\substack{W_L, W_R \in \mathcal{S} \\ i \in \{1,2,\ldots,k\} \\ (W_L \le w_i + W_R) \,\wedge\, (W_R \le w_i + W_L)}} \mathtt{T}[|\mathcal{S}|, W_L, W_R, m_1, m_2, \ldots, m_{i-1}, m_i - 1, m_{i+1}, m_{i+2}, \ldots, m_k]$$

Theorem

WSR_2 can be solved in time $O(n^{k+3} \cdot k)$ where k is the number of distinct vertex weights of any input instance $(V, \omega, \mathcal{S})$ and n is the number of vertices.

NP-completeness of SR_3

NP-completeness of SR_3

Here we show that SPLITS RECONSTRUCTION with unit weights is NP-complete for trees with maximum degree 3.

Again, we do a reduction from :

NUMERICAL MATCHING WITH TARGET SUMS (NMTS)

Input : 3 multisets A, B, and $S = \{s_1, \dots, s_m\}$ of size m from $\mathbb{N}$.
Question : Can $A \cup B$ be partitioned into m disjoint sets $C_1, C_2, \dots, C_m$, each containing exactly one element from each of A and B, such that $\sum_{c \in C_i} c = s_i$, $1 \leq i \leq m$?

Problem NMTS remains NP-complete even if each integer of the instance is at most $p(m)$, where p is a polynomial and m is the length of the description of the instance.

NP-completeness of SR_3

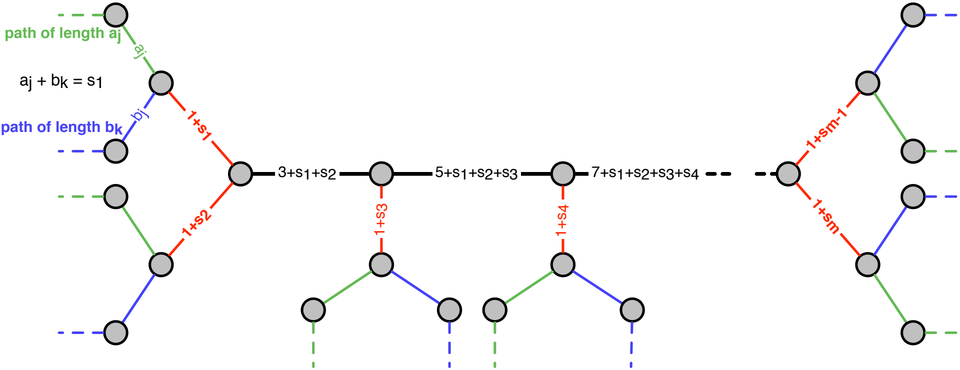
Given an instance $(\tilde{A}, \tilde{B}, \tilde{S})$, we start by scaling the integers :

Let $C = \max\{x \ : \ x \in \tilde{A} \cup \tilde{B}\}$.

$$a_i := \tilde{a}_i + 2 + 3C, \quad 1 \leq i \leq m,$$
$$b_i := \tilde{b}_i + 3 + 5C, \quad 1 \leq i \leq m,$$
$$s_i := \tilde{s}_i + 5 + 8C, \quad 1 \leq i \leq m.$$

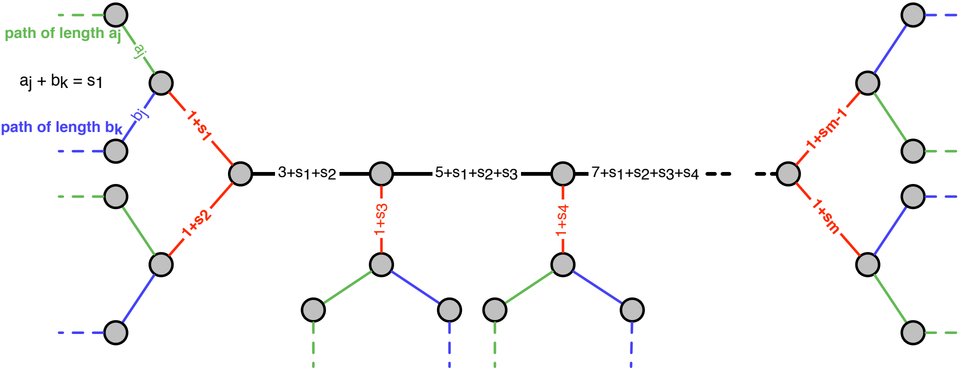
It remains to construct an instance $(V, \mathcal{S})$ of SR_3 being a YES-instance iff (A, B, S) is a YES-instance of NMTS.

NP-completeness of SR_3



Let $n = 2m - 2 + \sum_{i=1}^{m} a_i + \sum_{i=1}^{m} b_i$ be the number of vertices with unit weights. The multiset $\mathcal{S}$ of splits is defined as follows :

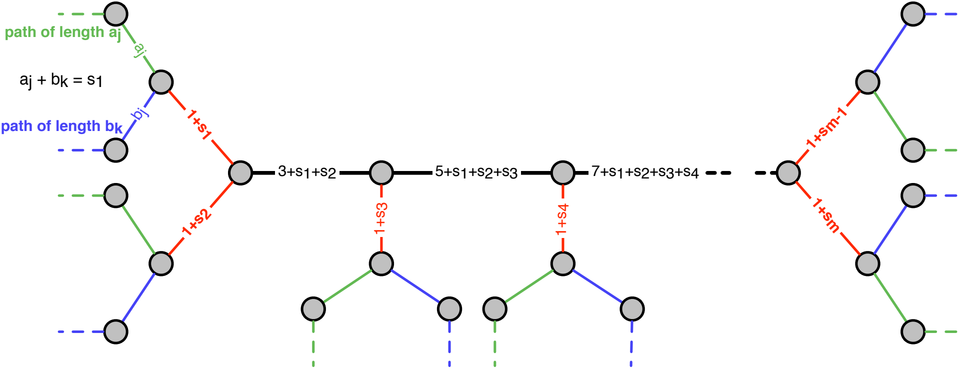
NP-completeness of SR_3



Let $n = 2m - 2 + \sum_{i=1}^{m} a_i + \sum_{i=1}^{m} b_i$ be the number of vertices with unit weights. The multiset $\mathcal{S}$ of splits is defined as follows :

- For each value s_i, $1 \leq i \leq m$, the value $1 + s_i$ is added to $\mathcal{S}$ and we refer to these splits as **red** splits.

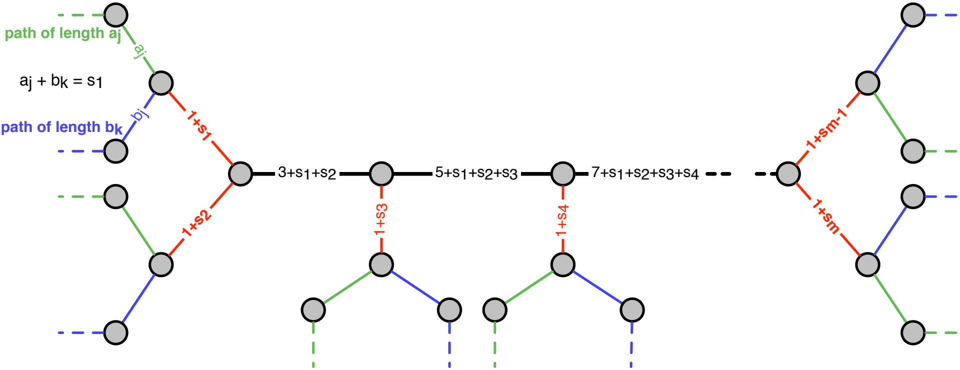
NP-completeness of SR_3



Let $n = 2m - 2 + \sum_{i=1}^{m} a_i + \sum_{i=1}^{m} b_i$ be the number of vertices with unit weights. The multiset $\mathcal{S}$ of splits is defined as follows :

- For each value s_i, $2 \leq i \leq m - 2$, the value $(i-1) + \sum_{j=1}^{i}(1 + s_j)$ is added to $\mathcal{S}$ and we refer to these splits as **black** splits.

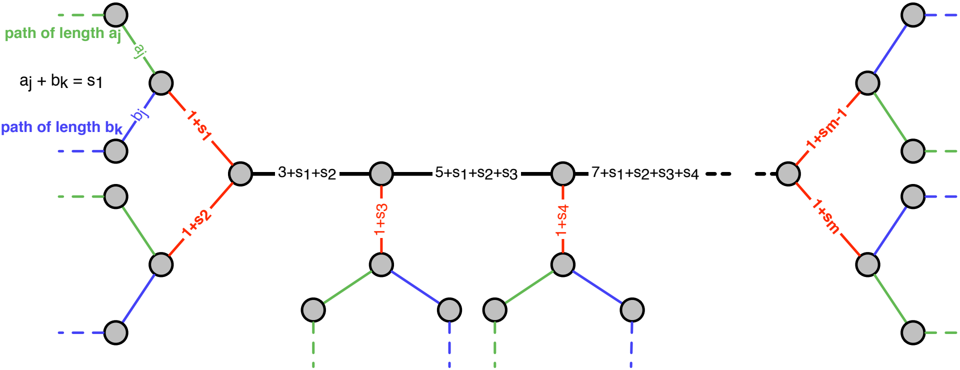
NP-completeness of SR_3



Let $n = 2m - 2 + \sum_{i=1}^{m} a_i + \sum_{i=1}^{m} b_i$ be the number of vertices with unit weights. The multiset $\mathcal{S}$ of splits is defined as follows :

- For each value a_i, $1 \leq i \leq m$, the values $\{1, 2, \ldots, a_i\}$ are added to $\mathcal{S}$ and we refer to these splits as **green** splits.

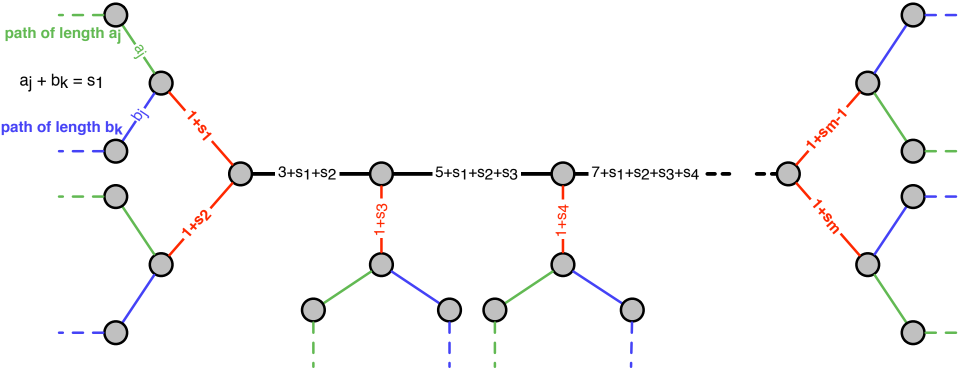
NP-completeness of SR_3



Let $n = 2m - 2 + \sum_{i=1}^{m} a_i + \sum_{i=1}^{m} b_i$ be the number of vertices with unit weights. The multiset $\mathcal{S}$ of splits is defined as follows :

- For each value b_i, $1 \leq i \leq m$, the values $\{1, 2, \ldots, b_i\}$ are added to $\mathcal{S}$ and we refer to these splits as **blue** splits.

NP-completeness of SR_3



Let $n = 2m - 2 + \sum_{i=1}^{m} a_i + \sum_{i=1}^{m} b_i$ be the number of vertices with unit weights. The multiset $\mathcal{S}$ of splits is defined as follows :

- For each value b_i, $1 \leq i \leq m$, the values $\{1, 2, \ldots, b_i\}$ are added to $\mathcal{S}$ and we refer to these splits as **blue** splits.

Finally each value x of $\mathcal{S}$ is replaced by $\min(x, n - x)$.

NP-completeness of SR_3

Scaling the input ensures that for any $i, j, k \in \{1, 2, \ldots, m\}$:

- $a_i + s_j > s_k$
- $a_i + a_j < s_k$
- $b_i + b_j > s_k$
- $a_i + a_j > b_k$

Claim. For every $i \in \{1, 2, \ldots, m\}$, there is a path on a_i edges, called the a_i-path, using the splits $1, 2, \ldots, a_i$ and there is a path on b_i edges, called the b_i-path, using the splits $1, 2, \ldots, b_i$.
All these a-paths and b-paths are edge-disjoint.

Claim. For every $i \in \{1, 2, \ldots, m\}$, the red split of value $1 + s_i$ is assigned to an edge e_i of T whose vertex u_i is the common extremity of an a-path and a b-path,
where u_i is in the subtree of $T - e_i$ that has $s_i + 1$ vertices.

NP-completeness of SR_3

Scaling the input ensures that for any $i, j, k \in \{1, 2, \ldots, m\}$:

- $a_i + s_j > s_k$
- $a_i + a_j < s_k$
- $b_i + b_j > s_k$
- $a_i + a_j > b_k$

Claim. For every $i \in \{1, 2, \ldots, m\}$, there is a path on a_i edges, called the a_i-path, using the splits $1, 2, \ldots, a_i$ and there is a path on b_i edges, called the b_i-path, using the splits $1, 2, \ldots, b_i$. All these a-paths and b-paths are edge-disjoint.

Claim. For every $i \in \{1, 2, \ldots, m\}$, the red split of value $1 + s_i$ is assigned to an edge e_i of T whose vertex u_i is the common extremity of an a-path and a b-path,
where u_i is in the subtree of $T - e_i$ that has $s_i + 1$ vertices.

NP-completeness of SR_3

Scaling the input ensures that for any $i, j, k \in \{1, 2, \ldots, m\}$:

- $a_i + s_j > s_k$
- $a_i + a_j < s_k$
- $b_i + b_j > s_k$
- $a_i + a_j > b_k$

Claim. For every $i \in \{1, 2, \ldots, m\}$, there is a path on a_i edges, called the a_i-path, using the splits $1, 2, \ldots, a_i$ and there is a path on b_i edges, called the b_i-path, using the splits $1, 2, \ldots, b_i$. All these a-paths and b-paths are edge-disjoint.

Claim. For every $i \in \{1, 2, \ldots, m\}$, the red split of value $1 + s_i$ is assigned to an edge e_i of T whose vertex u_i is the common extremity of an a-path and a b-path,
where u_i is in the subtree of $T - e_i$ that has $s_i + 1$ vertices.

NP-completeness of SR_3

Scaling the input ensures that for any $i, j, k \in \{1, 2, \ldots, m\}$:

- $a_i + s_j > s_k$
- $a_i + a_j < s_k$
- $b_i + b_j > s_k$
- $a_i + a_j > b_k$

Claim. For every $i \in \{1, 2, \ldots, m\}$, there is a path on a_i edges, called the a_i-path, using the splits $1, 2, \ldots, a_i$ and there is a path on b_i edges, called the b_i-path, using the splits $1, 2, \ldots, b_i$. All these a-paths and b-paths are edge-disjoint.

Claim. For every $i \in \{1, 2, \ldots, m\}$, the red split of value $1 + s_i$ is assigned to an edge e_i of T whose vertex u_i is the common extremity of an a-path and a b-path,
where u_i is in the subtree of $T - e_i$ that has $s_i + 1$ vertices.

Theorem

The problem SR_3 is NP-complete.

Conclusion

Conclusion

In this talk, we have shown the following :

- Scheduling With Common Deadlines is NP-complete
- WSR_2 is strongly NP-complete
- WSR_2 is polynomial-time solvable, assuming that the number of distinct vertex weights is constant-bounded
- SR_3 is NP-complete, which closes the gap (SR_2 poly-time solvable ; SR_4 NP-c)

In the paper we also show :

- Splits Reconstruction for caterpillars of unbounded hair-length and maximum degree 3 is NP-complete
- Given a multiset $\mathcal{S}$ of splits, the problem asking whether there exists a tree $T = (V, E)$ and a weight function $\omega : V \to \mathbb{N}$ s.t. $\mathcal{S}$ is the multiset of splits of T, always admits a solution that can be built in polynomial-time.

Conclusion

Interesting questions :

- We have shown that WSR_2 is in XP (parameterized by the number of distinct vertex weights).

  Is the problem FPT ?

  A generalization is known to be W[1]-hard [Fellows, Gaspers, Rosamond]

- For which restrictions on the multiset of vertex weights does the problem become polynomial-time solvable, or FPT with respect to some interesting parameterizations.

Merci !