22. Scheduling and Jump Number

Goal: Study the parameterized complexity of a scheduling-related problem.

PRECEDENCE-CONSTRAINED h-PROCESSOR SCHEDULING:

Input: A set T of unit length jobs
and a partial order $\leq \subseteq T \times T$ on T.
A deadline $D \in \mathbb{N}$.
A number of processors $h \in \mathbb{N}$.

Parameter: The number of processors h.

Question: Is there a mapping $f: T \to \{1, \dots, D\}$ so that
for all $t < t'$ we have $f(t) < f(t')$ // Respect ordering and deadline
and for all $1 \leq i \leq D$: $|f^{-1}(i)| \leq h$ // Respect number of processors.

- The problem is NP-complete
and known to be in P for 2 processors.
- For $h \geq 3$, the behavior is a long-standing open problem
(Garey & Johnson: Computers and Intractability, 1979, Problem 8).
- Def: There is a parameterized complexity analysis.

Theorem:
PCPS is W[2]-hard.

We will study the complexity class later.
It is akin to NP-completeness and it is not believed
that W[2]-hard problems are FPT.

Scheduling for a single processor can be understood as finding an extension of a partial order to a total (also called linear) order.

Theorem:

If $(T, \leq)$ is a partial order,
then there is a total order $(T, \sqsubseteq)$
so that $a \leq b$ implies $a \sqsubseteq b$ // Make incomparable elements comparable, but do not invalidate $\leq$.

Proof: For finite T by induction on the number of elements (homework).

Intuitively, a jump occurs in $(T, \sqsubseteq)$,
whenever we have two successive elements $a \sqsubset b$
that were incomparable in $\leq$.

Example:

shoes
socks ! hat

socks . hat . shoes $\Rightarrow$ 2 jumps
socks . shoes . hat $\Rightarrow$ 1 jump.

Intuitively, a jump corresponds to working on a new task (like a context switch).

Definition:

Let $(X, \leq_X)$ and $(Y, \leq_Y)$ be two partial orders, $X \cap Y = \emptyset$.
The linear sum is $(X, \leq_X) \oplus (Y, \leq_Y) := (X \cup Y, \leq)$
with $a \leq b$, if
- $a, b \in X$ and $a \leq_X b$
- $a, b \in Y$ and $a \leq_Y b$
- $a \in X$ and $b \in Y$.

Definition:

- A linear extension $L = (P, \leq)$ of a finite poset order $(P, \leq)$ (also called total order extending $(P, \leq)$) is a linear sum

$$L = C_1 \oplus \dots \oplus C_m$$

of disjoint chains in P so that $x \in C_i$ and $y \in C_j$ with $x \leq y$ (in $(P, \leq)$) implies $i \leq j$.

- If the maximal element in C_i is incomparable in $(P, \leq)$ with the minimum in C_{i+1}, we call the pair

$(\max C_i, \min C_{i+1})$ a jump in L.

The number of jumps in L is denoted by $s_L(P)$.

The jump number of P is

$$s(P) := \min \{ s_L(P) \mid L \text{ a linear extension of } P \}.$$

(Sperner, 1905-1980)

- The width of P, denoted by $w(P)$, is the size of the maximal antichain in P.
The antichain is a set of incomparable elements.

Theorem (Dilworth '50):
Consider $(P, \leq)$. The minimum number of chains $C_1, \dots, C_m$ that form a partition of P equals $w(P)$.

Corollary:
$s(P) \geq w(P) - 1$.

Theorem (Pulleyblank '81, Bouchitté & Habib '87):

Computing $s(P)$ is NP-complete.

We consider a parametrized version of the problem.

JUMPNUMBER

Given: A finite partial order $(P, \leq)$ and $k \in \mathbb{N}$.

Parameter: The k.

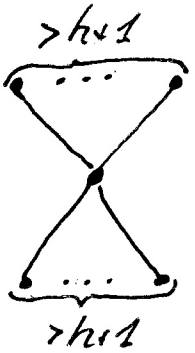
Question: Is $s(P) \leq k$? If so, output $L(P)$ with $s_L(P) \leq k$.

Here, $L(P)$ is a linearization of P.

Theorem (McCartin '01):

JUMPNUMBER can be solved in $O(k^2 k! \cdot |P|)$.

There is a story about the algorithm that yields the upper bound.
When it was first implemented, it gave wrong results.
Indeed, there was a mistake in the first proof.



Proof:

- If $s(P) \leq k$ then no element of P covers or is covered by more than $k+1$ elements.

This would imply $w(P) > k+1$

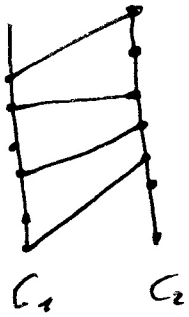
and hence $s(P) \geq w(P) - 1$
$> k+1-1 = k$.

Thus, if we represent P as a Hasse diagram no node has indegree or outdegree $> k+1$.

- We now analyse the structure of ranked orders with restricted jump number.

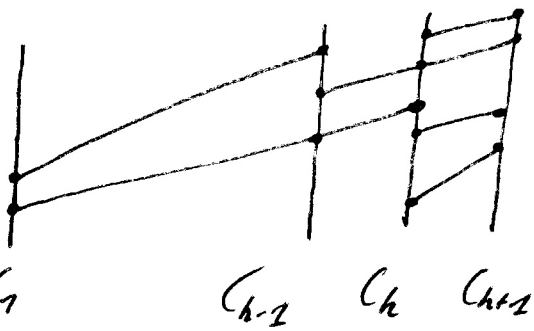
Assume $k=1$:

If $s(P) \leq 1$, the structure of P is very restricted.
P can be decomposed into at most two chains C_1 and C_2.
Moreover, any element $a \in C_1$ comparable to $b \in C_2$ must satisfy $a < b$.
Hence, P forms a <u>skewed ladder</u>:



Fix k, let $w(P)=k+1$, $s(P) \leq k$:

With the corollary, we have $s(P)=k$.
Then P must be decomposable into a <u>width-$k+1$-skewed ladder</u>:



There are $k+1$ disjoint chains that can be ordered so that:

(1) Any element $a \in C_i$ comparable to $b \in C_j$ with $i<j$ must satisfy $a < b$.

(2) $w(P_i) = i$ with $P_i = C_1 \oplus \dots \oplus C_i$.

(Algorithmically, the skewed ladder is detected starting from the maximal chain.
This property is needed for the recursion.)

Definition:

- For any $a \in P$, let $a\downarrow := \{x \in P \mid x \leq a\}$ be the downward closure of a.
- We call a accessible, if $a\downarrow$ is a chain. Element a is maximally accessible if $a < b$ implies b is not accessible.

Observation:

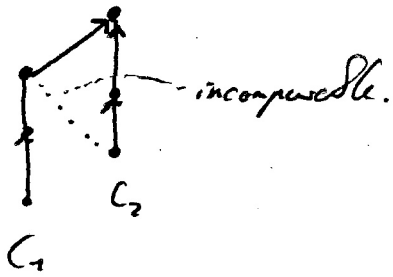
- In any linear extension $L = C_1 \oplus \dots \oplus C_n$ of P, each $a \in C_i$ is accessible in $C_i \oplus \dots \oplus C_n$. In particular, C_1 only contains elements that are accessible in P.
- If C_1 does not contain a maximally accessible element, we can transform L to
$$L' = C_1' \oplus \dots \oplus C_n'$$
where · C_1' does contain a maximally accessible element
· and the number of jumps does not increase.

(This claim can be shown as an exercise.)

Lemma:
$s(P) \leq h$ iff $s(P \setminus a\downarrow) \leq h-1$ for some maximally accessible element $a \in P$.

Proof:
$\Rightarrow$: If $s(P) \leq h$, then there is $L = C_1 \oplus \dots \oplus C_n$ a linearization of P so that · $s_L(P) \leq h$
· C_1 contains a maximally accessible element, say a.

Now $L' = C_2 \oplus \dots \oplus C_m$ is a linearization of $P \setminus a\downarrow$,
where
$$s_{L'}(P \setminus a\downarrow) \le h-1.$$
This follows from maximal accessibility:



$\Leftarrow$ Suppose $s(P \setminus a\downarrow) \le h-1$ for some maximally accessible element $a \in P$.
Then there is $L(P \setminus a\downarrow) = C_1 \oplus \dots \oplus C_m$
with $s_L(P \setminus a\downarrow) \le h-1$.
Now $L' = a\downarrow \oplus C_1 \oplus \dots \oplus C_m$ is a linear extension of P
with $s_{L'}(P) \le h$. □

Algorithm (McCartin '01):

We build a search tree of height $\le h$:

- Label the root by P.
- Find all maximally accessible elements in P.
  If there are more than $h+1$,
  then $w(P) \ge h+1$ and the answer is no.
  So assume there are $\le h+1$ maximally accessible elements.
- If there are exactly $h+1$ maximally accessible elements,
  then P has width $\ge h+1$.
  So to match $s(P) \le h$, we have to check
  whether P has a width-$h+1$-skewed ladder decomposition.
  ↳ Yes, output L defined later
  ↳ No, output no.

- If there are $< h+1$ maximally accessible elements
  then for each of the elements $a_1, \dots, a_m$
  produce a child node
  labelled by $(a_i \downarrow, P \mid a_i \downarrow)$.

  Ask: Does $P \mid a_i \downarrow$ have jump number $\leq h-1$?
  If so, output $L = a_i \downarrow \oplus L_i$.

Analysis:

- ↳ The height of the search tree is $\leq h$,
  since at each level we reduce the parameter value by 1.
  - ↳ The branching is bounded by the current parameter value,
    since we only branch if we have less than
    parameter + 1
    maximally accessible elements to choose from.
  - ↳ Thus, the number of paths is bounded by
    $h!$
    Moreover, each path contains at most h steps.

- ↳ At each step, we must find all maximally accessible elements
  of the current P.
  - ↳ Then we have to produce a child node
    for each maximally accessible element
  - ↳ or, if the number matches the current width,
    we have to output a skewed ladder decomposition.

Lemma:

(1) Finding all maximally accessible elements
can be done in $O((h+1)n)$.

(2) Producing a child node labelled $(a;L, P \setminus a;L)$
can be done in $O((h+1)n)$.

(3) Recognize and output a skewed ladder decomposition
of width n can be done in $O((n(n+1)/2)n)$.

Assuming this lemma (the proof of which is long but not too difficult)
we can estimate the running time of the overall algorithm.

On each path:

↳ At some point we have to recognize a skewed ladder.

Suppose this occurs after q iterations of the
"find all maximally accessible elements"-phase,

where the parameter decreases by 1 for each iteration.
The skewed ladder to be recognized has width

$$(h+1) - (q-1)$$

↳ The first q steps require
$O((h+1)n), O(((h+1)-1)n), \ldots, O(((h+1)-(q-1))n)$.

Recognizing the skewed ladder takes
$O((n(n+1)/2) \cdot n)$, with $n = (h+1)-(q-1)$.

↳ Altogether, no path requires more than
$O((h+1)n + ((h+1)(h+2)/2) \cdot n)$.

↳ The running time is $O(h^2 \cdot h^2 n)$.

□