A New Parameter for Scheduling Dependent Tasks

Alix Munier-Kordon[1], Claire Hanen[1,2]

[1]Sorbonne Université, CNRS, LIP6, F-75005 Paris, France,
[2]Université Paris Nanterre, F-92000 Nanterre, France,

March 19, 2026

Outline

Problems with directed acyclic graphs

There are many NP-hard problems dealing with directed acyclic graphs.

- Scheduling problems;
- Assembly line balancing problems;
- Ordering problems: find a topological order that optimizes some criteria (ex: Directed Sum Cut).

Parameterized analysis of problems with DAG G

FPT algorithms

Fixed Parameter algorithm with respect to a parameter k

- Time complexity $\mathcal{O}(f(k)n^{\mathcal{O}(1)})$,
- n size of the instance, f computable function.

Several parameters have been studied

- The width $w(G)$ of the graph (max # of disjoint chains)$\rightarrow$ Negative results even for simple settings : no hope for FPT algorithms.
  - $P2|prec, p_j \in \{1,2\}|C_{\max}$ is W[2]-hard parameterized by $w(G)$
  - $P|prec, p_j = 1|C_{\max}$ is XNLP-complete parameterized by $m + w(G)$
- With time windows, two parameters $\rightarrow$ FPT results for scheduling:
  - Slack (maximum # of starting times of a job within its time window)
  - Pathwidth (maximum # of overlapping time windows)

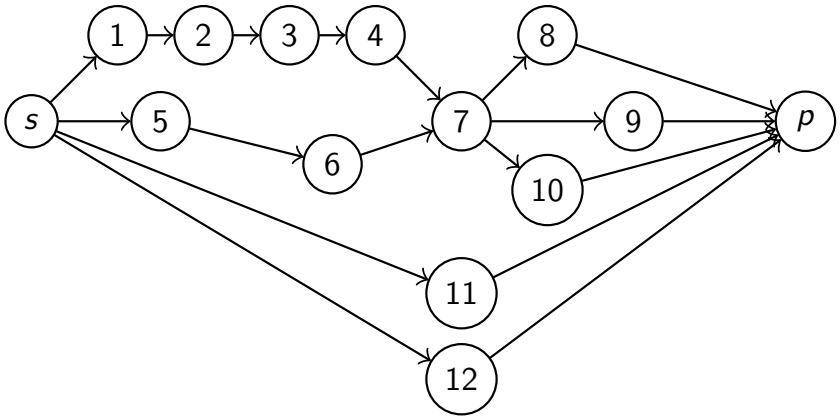
Motivation

Questions as starting point:

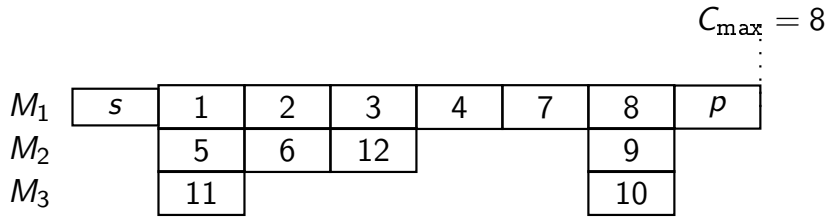
- Are there structural parameters of the DAG that would yield FPT algorithms for:
  - scheduling problems?
  - other related problems with DAG?
- Is dynamic programming suitable for such algorithms?

Parallel machine scheduling with unit processing times

We consider $P|prec, p_j = 1|C_{max}$, NP-Hard in the strong sense.



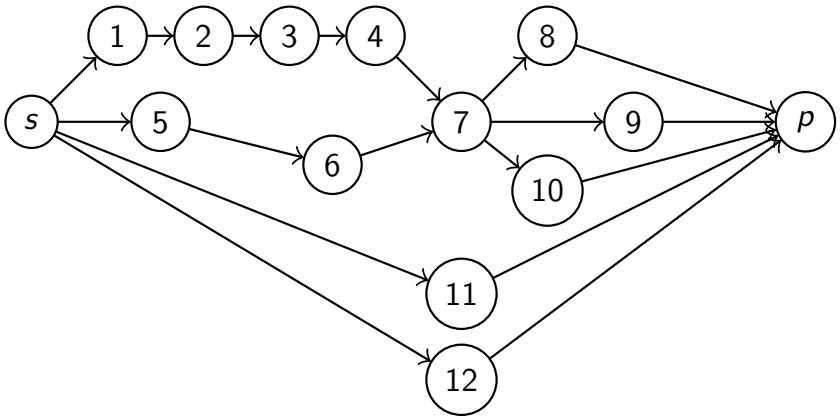
A precedence graph (DAG) with a source and sink.

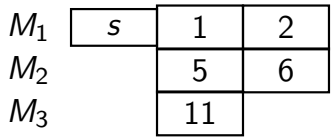


A schedule on 3 machines.

Dynamic programming

Modeling a schedule as a sequential decision process:



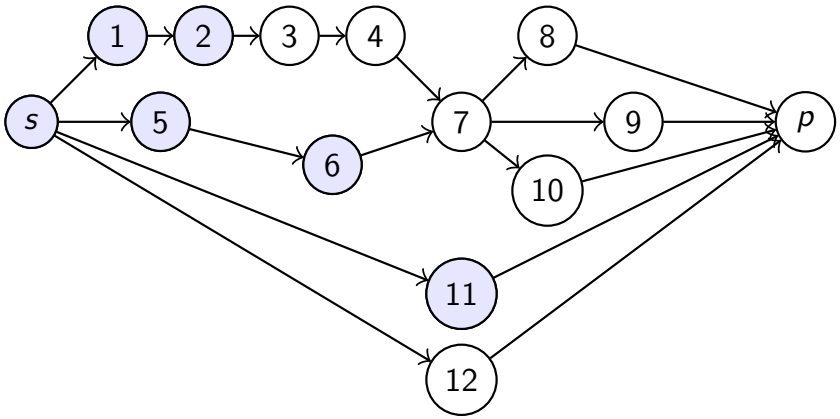


Partial schedule

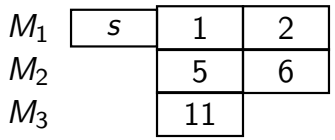
A step of the decision process is to extend current partial schedule to the next time unit.

Dynamic programming

What must be known to make the decision:



The set of scheduled jobs

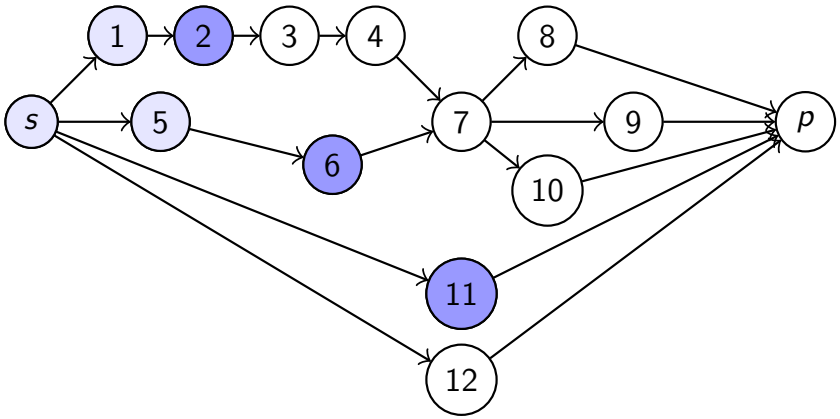


Partial schedule

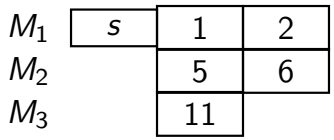
A step of the decision process is to extend current partial schedule to the next time unit.

Dynamic programming

A compact representation:



The set of scheduled jobs
is represented by scheduled nodes
without scheduled successors



Partial schedule

A step of the decision process is to extend current partial schedule to the next time unit.

Borders

Cut set

A cut set of G is a set of nodes S such that if $i \in S$ all its precedessors are in S.

Border

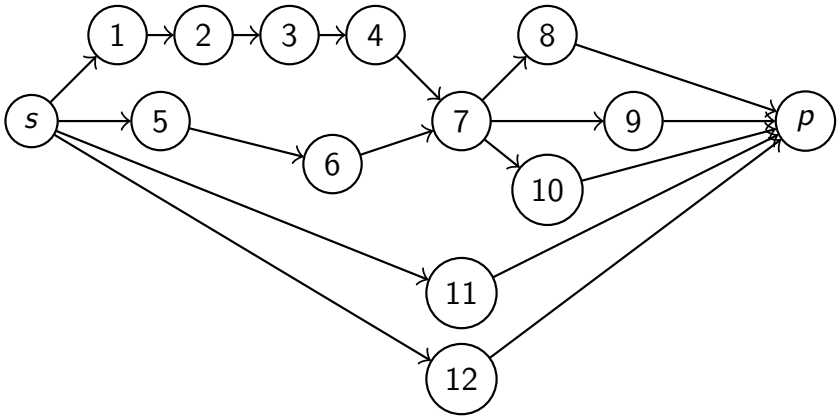
The border $B(S)$ of a cut set S is the set of nodes in S without successors in S

Property

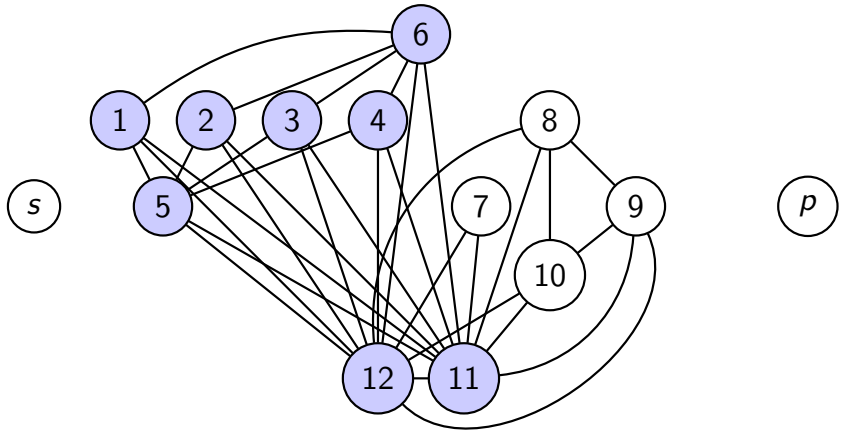
If B is a set of jobs such that there are no precedence relations between any two jobs of B, then $N_G^{-*}(B)$, the precedessors of B define a cut set $C(B)$.

Co-comparability graph and degeneracy

Edge $\{i, j\}$ in $H_G \iff$ no path $i \to j$ nor $j \to i$ in G



a DAG G



Its co-comparability graph H_G.

Degeneracy of the co-comparability graph δ

Minimum k such that in each subset S, there is a node with degree at most k in $H_G(S)$

Exemple: in the subgraph induced by $S = \{1, 2, 3, 4, 5, 6, 11, 12\}$ node 1 has the minimum degree 4. For this example $\delta = 4$.

Properties

Property (Computation of the degeneracy)

The degeneracy of a graph can be easily computed in polynomial time. It defines an order of the nodes such that if S_j is the set of nodes $\geq j$ in this order, j has degree at most δ in $H_G(S_j)$.

$$\delta(H_G) \geq w(G)$$

Property (Borders and cliques)

B border of $G \iff H_G(B)$ is a clique.

Theorem (Number of Borders/cliques)

For each border B, $|B| \leq \delta$. The number of borders is $\leq n2^{\delta}$

NB: Eppstein et al[2010] provide an efficient algorithm in $\mathcal{O}(n\delta 3^{\frac{\delta}{3}})$ for enumerating Maximal cliques, but not all the cliques.

Building a state graph

- States = Borders
- There is an arc between two states B and B' if any schedule of $C(B)$ can be extended by one time unit to get a schedule of $C(B')$:
  - $C(B) \subset C(B')$
  - $C(B') \setminus C(B) = B' \setminus B$ is a clique of H_G
  - $|C(B') \setminus C(B)| \leq m$

Theorem

A path from the initial state $B_0 = \emptyset$ to state B with t arcs $\iff$ a schedule of $C(B)$ of makespan t.

Corollary (Problem reformulation)

Finding the shortest path from B_0 to the last border $\{p\}$.

Building a state graph

- States = Borders
- There is an arc between two states B and B' if any schedule of $C(B)$ can be extended by one time unit to get a schedule of $C(B')$:
  - $C(B) \subset C(B')$ $\mathcal{O}(\delta^2)$
  - $C(B') \setminus C(B) = B' \setminus B$ is a clique of H_G $\mathcal{O}(\delta^2)$
  - $|C(B') \setminus C(B)| \leq m$ $\mathcal{O}(\delta)$

Theorem

A path from the initial state $B_0 = \emptyset$ to state B with t arcs $\iff$ a schedule of $C(B)$ of makespan t.

Corollary (Problem reformulation)

Finding the shortest path from B_0 to the last border $\{p\}$.

Complexity analysis of a naive DP algorithm:

1. Compute the transitive closure $\bar{G}$ of G
2. Compute H_G
3. Compute the ordering of nodes w.r.t. degeneracy $\delta(H_G)$
4. Enumerate all the borders
5. For each tuple (B, B') check the existence of an arc.
6. Perform a breadth first search to compute the shortest path (in number of arcs) from B_0 to $\{p\}$

Complexity analysis of a naive DP algorithm:

1. Compute the transitive closure $\bar{G}$ of G $\quad \mathcal{O}(n^3)$
2. Compute H_G
3. Compute the ordering of nodes w.r.t. degeneracy $\delta(H_G)$
4. Enumerate all the borders
5. For each tuple (B, B') check the existence of an arc.
6. Perform a breadth first search to compute the shortest path (in number of arcs) from B_0 to $\{p\}$

Complexity analysis of a naive DP algorithm:

1. Compute the transitive closure $\bar{G}$ of G $\quad \mathcal{O}(n^3)$
2. Compute H_G $\quad \mathcal{O}(n^2)$
3. Compute the ordering of nodes w.r.t. degeneracy $\delta(H_G)$
4. Enumerate all the borders
5. For each tuple (B, B') check the existence of an arc.
6. Perform a breadth first search to compute the shortest path (in number of arcs) from B_0 to $\{p\}$

Complexity analysis of a naive DP algorithm:

1. Compute the transitive closure $\bar{G}$ of G $\quad \mathcal{O}(n^3)$
2. Compute H_G $\quad \mathcal{O}(n^2)$
3. Compute the ordering of nodes w.r.t. degeneracy $\delta(H_G)$ $\quad \mathcal{O}(n^2)$
4. Enumerate all the borders
5. For each tuple (B, B') check the existence of an arc.
6. Perform a breadth first search to compute the shortest path (in number of arcs) from B_0 to $\{p\}$

Complexity analysis of a naive DP algorithm:

1. Compute the transitive closure $\bar{G}$ of G $\quad \mathcal{O}(n^3)$
2. Compute H_G $\quad \mathcal{O}(n^2)$
3. Compute the ordering of nodes w.r.t. degeneracy $\delta(H_G)$ $\quad \mathcal{O}(n^2)$
4. Enumerate all the borders $\quad \mathcal{O}(\delta^2 n 2^\delta)$
5. For each tuple (B, B') check the existence of an arc.
6. Perform a breadth first search to compute the shortest path (in number of arcs) from B_0 to $\{p\}$

Complexity analysis of a naive DP algorithm:

1. Compute the transitive closure $\bar{G}$ of G $\quad \mathcal{O}(n^3)$
2. Compute H_G $\quad \mathcal{O}(n^2)$
3. Compute the ordering of nodes w.r.t. degeneracy $\delta(H_G)$ $\quad \mathcal{O}(n^2)$
4. Enumerate all the borders $\quad \mathcal{O}(\delta^2 n 2^\delta)$
5. For each tuple (B, B') check the existence of an arc. $\quad \mathcal{O}(\delta^2 n^2 4^\delta)$
6. Perform a breadth first search to compute the shortest path (in number of arcs) from B_0 to $\{p\}$

Complexity analysis of a naive DP algorithm:

1. Compute the transitive closure $\bar{G}$ of G $\quad \mathcal{O}(n^3)$
2. Compute H_G $\quad \mathcal{O}(n^2)$
3. Compute the ordering of nodes w.r.t. degeneracy $\delta(H_G)$ $\quad \mathcal{O}(n^2)$
4. Enumerate all the borders $\quad \mathcal{O}(\delta^2 n 2^\delta)$
5. For each tuple (B, B') check the existence of an arc. $\quad \mathcal{O}(\delta^2 n^2 4^\delta)$
6. Perform a breadth first search to compute the shortest path (in number of arcs) from B_0 to $\{p\}$ $\quad \mathcal{O}(n^2 4^\delta)$

Theorem

The algorithm is FPT with time complexity $\mathcal{O}(n^3 + \delta^2 n^2 4^\delta)$.

Extensions to other problems

This approach can be applied to any problem with a DAG such that

1. Cut sets summarizes the past decisions of decision process;
2. Feasibility of a decision should be easily computable;
3. Suboptimal property of the objective function: any complete optimal solution should have optimal partial solutions.

Directed SUM CUT (DSC)

Let $\mathcal{G} = (V, A)$ be a DAG.

- A topological ordering of the vertices is a bijection $\varphi : V \to \{1, \ldots, n\}$ and such that if $(i, j) \in A$, $\varphi(i) < \varphi(j)$;
- For any value $\alpha \in \{1, \ldots, n\}$, $\Delta(\alpha, \varphi) = |\{i \in V, (\varphi(i) \leq \alpha) \wedge (\exists j \in V, \varphi(j) > \alpha \wedge (i, j) \in A)\}|$;
- Find a topological ordering φ that minimizes the objective function $DSC(\varphi) = \sum_{\alpha=1}^{n+1} \Delta(\alpha, \varphi)$.

Theorem

DSC *parametrized by the degeneracy k of the co-comparability graph is* FPT *with a time complexity in* $\mathcal{O}(n^3 k 2^k + k^2 n^2 2^{2k})$.

Simple Basic Assembly Line Problem (SALBP-1)

- The workstations are indexed from 1 to m.
- Each task $i \in V$ has a given processing time $p_i \in \mathbb{N}$. They are dependent and to be executed by a minimum number of workstations m with a throughput bounded by C.
- A feasible mapping is a function $\pi : V \to \{1, \ldots, m\}$ such that:
  1. for any arc $(i, j) \in A$, $\pi(i) \leq \pi(j)$;
  2. for any workstation $k \in \{1, \ldots, m\}$, $\sum_{\pi(i)=k} p_i \leq C$.
- The SALBP-1 problem is to find a feasible mapping minimizing m.

Theorem

SALBP-1 parametrized by the degeneracy k of the co-comparability is FPT *with a time complexity in $\mathcal{O}(n^3 2^{2k})$.*

Conclusion

- Degeneracy of the co-comparability graph is a structural parameter that yield FPT algorithms for scheduling and graph problems on DAGs
- Implicit representation of solutions as path of a state graph, $\Longrightarrow$ application to several objective functions with the suboptimality property (sum, weighted sum, etc).
- Future work:
  - implementation of clever ways to build useful part of the state graph;
  - Implementation of bounds;
  - Approximation.