



Graph parameters for single machine scheduling with time windows

Maher Malle¹

¹Inria, CNRS, ENS de Lyon, Université Claude Bernard Lyon 1, LIP, UMR 5668,
69342, Lyon cedex 07, France

April 27, 2026

Outline

Introduction
Vertex cover number
Pathwidth
Proper level
Conclusion
References

- 1 Introduction
- 2 Vertex cover number
- 3 Pathwidth
- 4 Proper level
- 5 Conclusion
- 6 References

Outline

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References

1 Introduction

2 Vertex cover number

3 Pathwidth

4 Proper level

5 Conclusion

6 References

Introduction to scheduling

Introduction

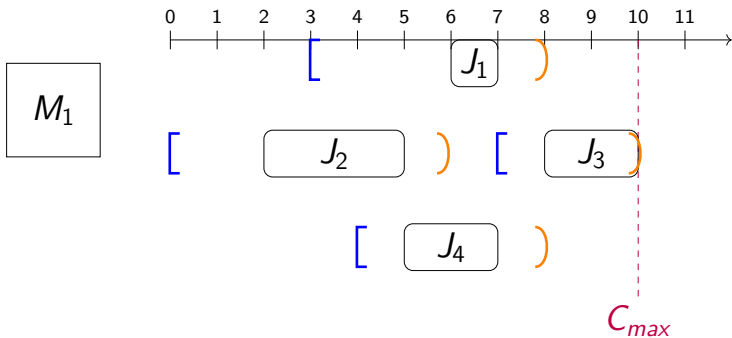
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Introduction to scheduling

Introduction

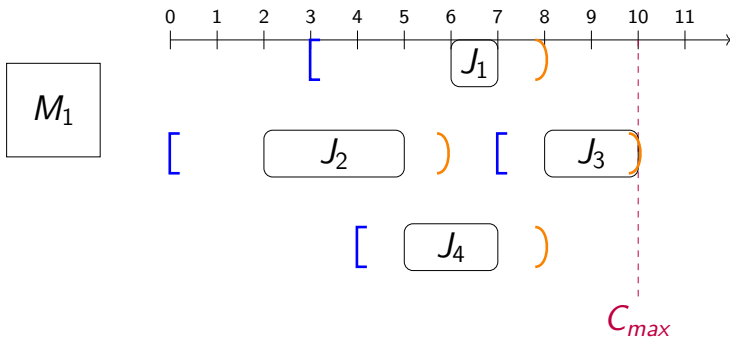
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

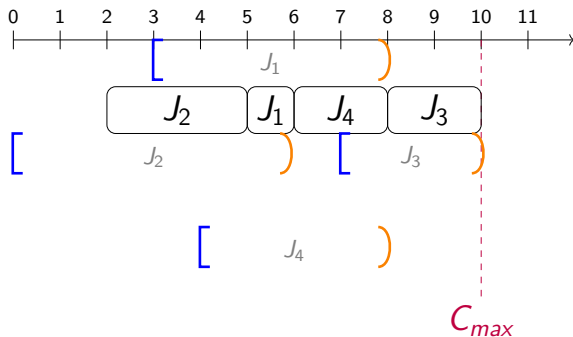
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

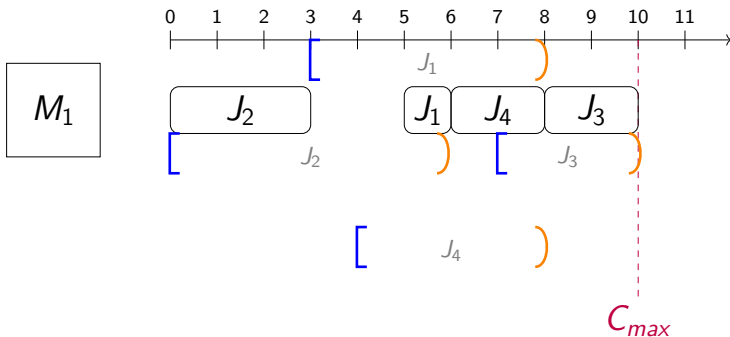
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

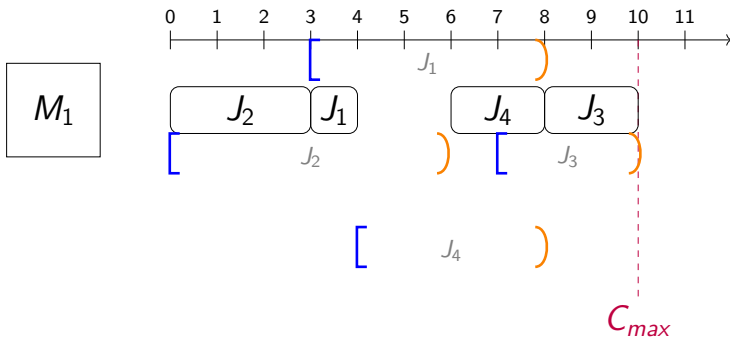
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

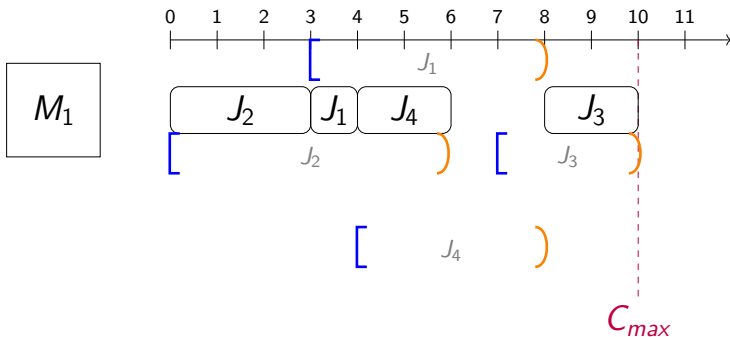
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

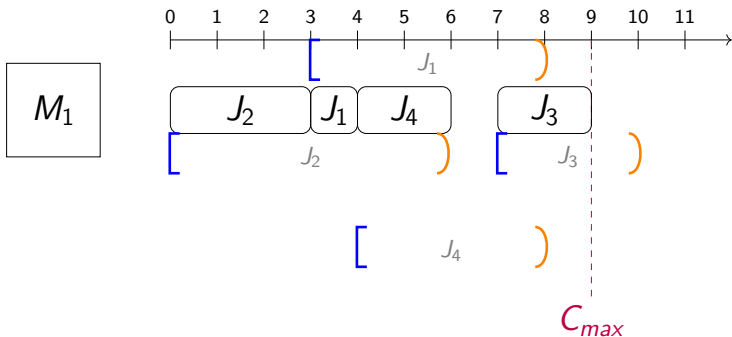
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

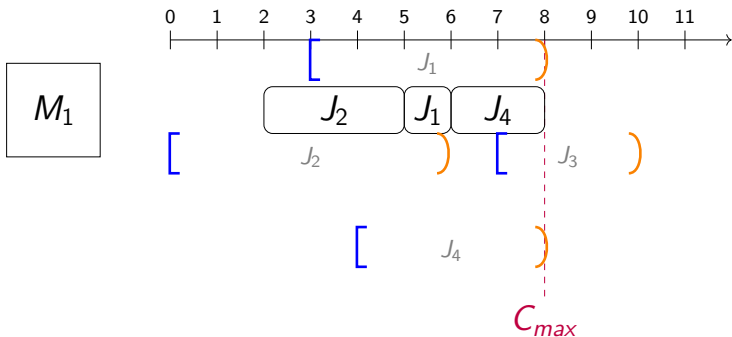
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

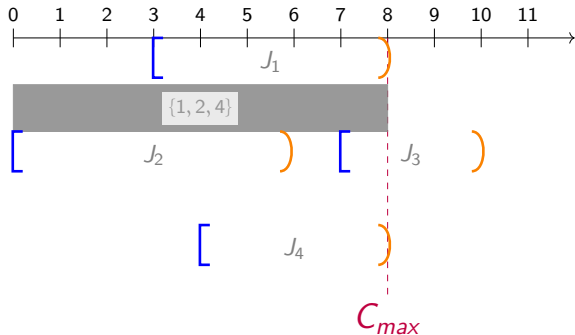
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

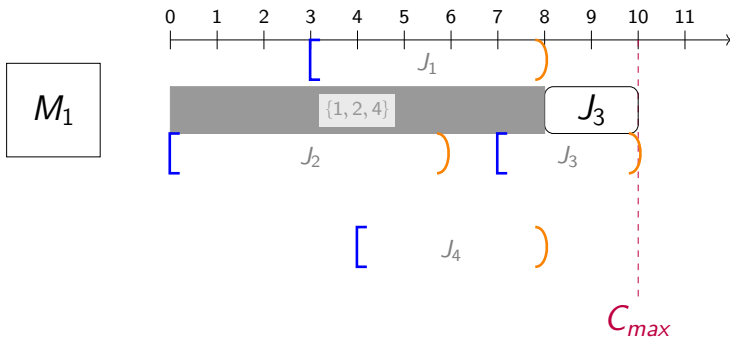
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

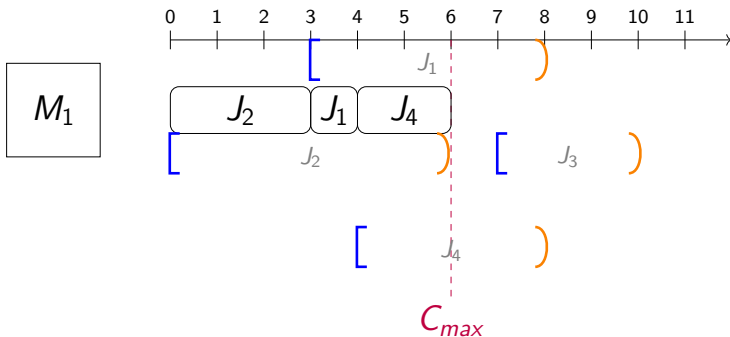
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

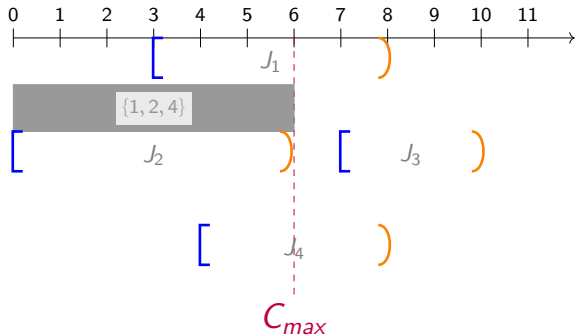
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

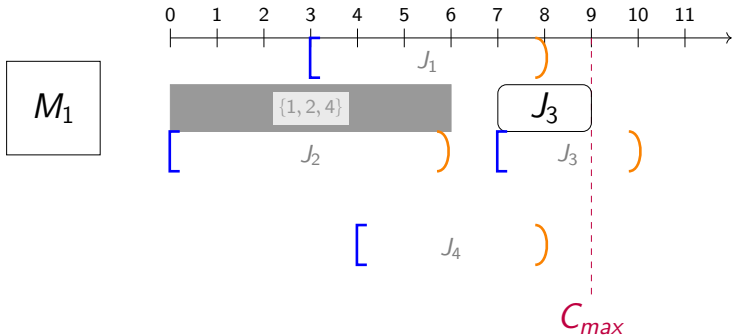
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Introduction to scheduling

Introduction

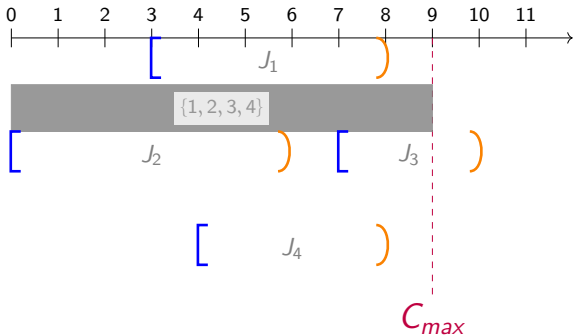
Vertex cover number

Pathwidth

Proper level

Conclusion

References



$$1|r_j, d_j|C_{max}$$

Strongly NP-complete [Lenstra et al., 1977].

Dynamic programming (baseline)

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References

Dynamic programming state (k, S_k)

- k = number of completed jobs.
- S_k = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.

Dynamic programming (baseline)

Introduction

Vertex cover number

Pathwidth

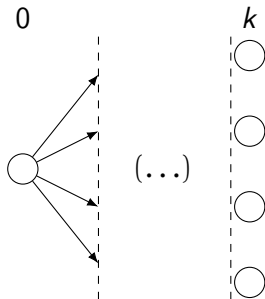
Proper level

Conclusion

References

Dynamic programming state (k, S_k)

- k = number of completed jobs.
- S_k = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline)

Introduction

Vertex cover number

Pathwidth

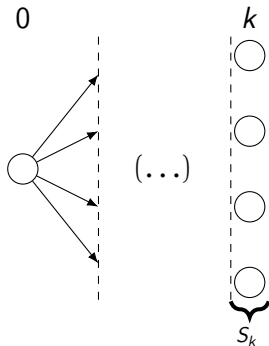
Proper level

Conclusion

References

Dynamic programming state (k, S_k)

- k = number of completed jobs.
- S_k = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline)

Introduction

Vertex cover number

Pathwidth

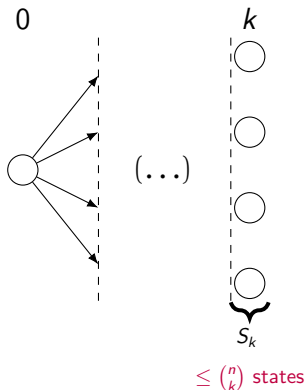
Proper level

Conclusion

References

Dynamic programming state (k, S_k)

- k = number of completed jobs.
- S_k = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline)

Introduction

Vertex cover number

Pathwidth

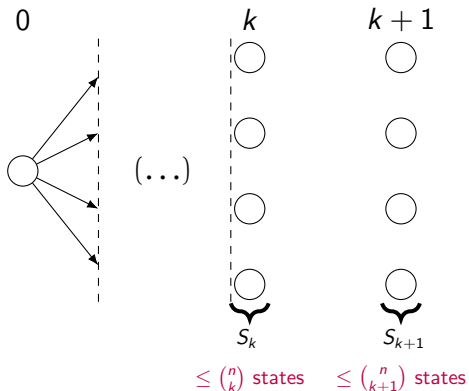
Proper level

Conclusion

References

Dynamic programming state (k, S_k)

- k = number of completed jobs.
- S_k = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline)

Introduction

Vertex cover number

Pathwidth

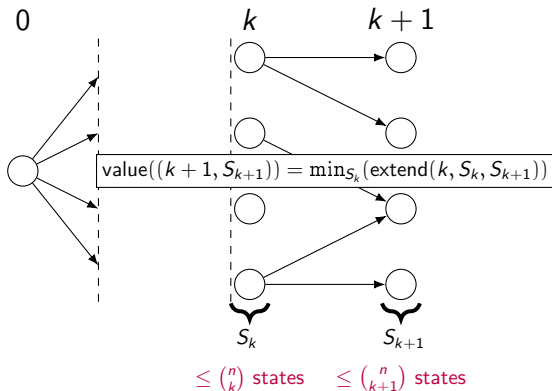
Proper level

Conclusion

References

Dynamic programming state (k, S_k)

- k = number of completed jobs.
- S_k = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline)

Introduction

Vertex cover number

Pathwidth

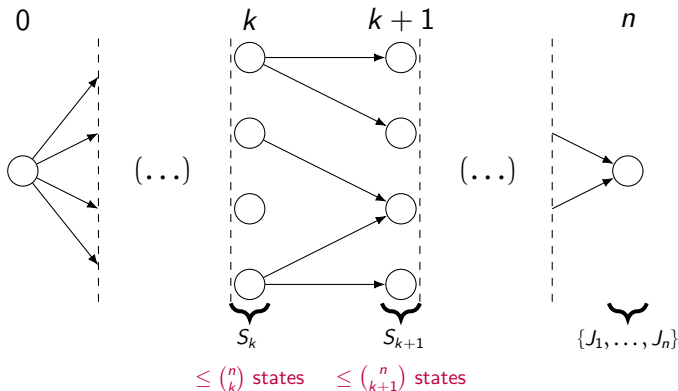
Proper level

Conclusion

References

Dynamic programming state (k, S_k)

- k = number of completed jobs.
- S_k = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline)

Introduction

Vertex cover number

Pathwidth

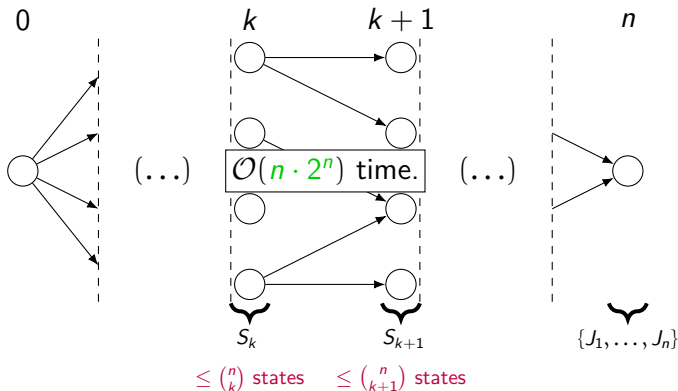
Proper level

Conclusion

References

Dynamic programming state (k, S_k)

- k = number of completed jobs.
- S_k = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline)

Introduction

Vertex cover number

Pathwidth

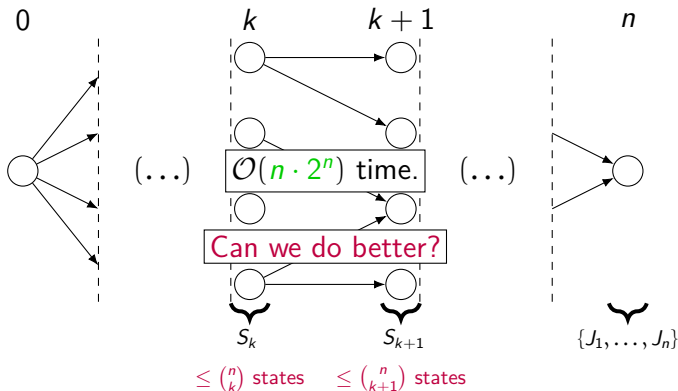
Proper level

Conclusion

References

Dynamic programming state (k, S_k)

- k = number of completed jobs.
- S_k = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline)

Introduction

Vertex cover number

Pathwidth

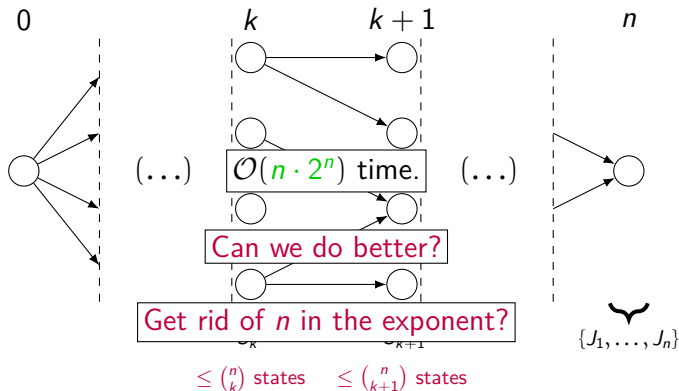
Proper level

Conclusion

References

Dynamic programming state (k, S_k)

- k = number of completed jobs.
- S_k = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References

- Problem $\mathcal{P}$, instance $\mathcal{I}$.
- P: deterministic $\text{poly}(|\mathcal{I}|)$ time.

- Problem $\mathcal{P}$, instance $\mathcal{I}$.
 - Parameter $\bar{k}$, instance $\mathcal{I}$ with parameter value $\leq k$.
- P: deterministic $\text{poly}(|\mathcal{I}|)$ time.
 - FPT: deterministic $f(k) \cdot \text{poly}(|\mathcal{I}|)$ time.

- Problem $\mathcal{P}$, instance $\mathcal{I}$.
 - Parameter $\bar{k}$, instance $\mathcal{I}$ with parameter value $\leq k$.
- P: deterministic $\text{poly}(|\mathcal{I}|)$ time.
 - **FPT: deterministic $f(k) \cdot \text{poly}(|\mathcal{I}|)$ time.**
- **Example:** $1|r_j, d_j|C_{\max}$ parameterized by n **is in FPT**.
 - (Note: input size in $\mathcal{O}(n \cdot \log(\max_j d_j))$.)

Outline

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References

1 Introduction

2 **Vertex cover number**

3 Pathwidth

4 Proper level

5 Conclusion

6 References

Vertex cover number vc

Introduction

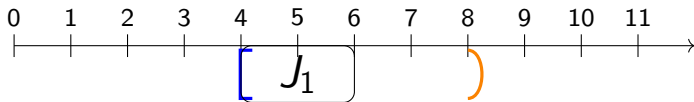
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Vertex cover number vc

Introduction

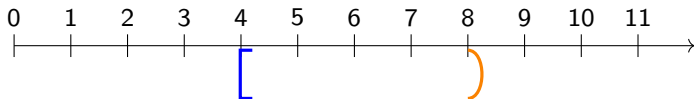
Vertex cover number

Pathwidth

Proper level

Conclusion

References



[

)

[

)

[

)

[

)

Vertex cover number vc

Introduction

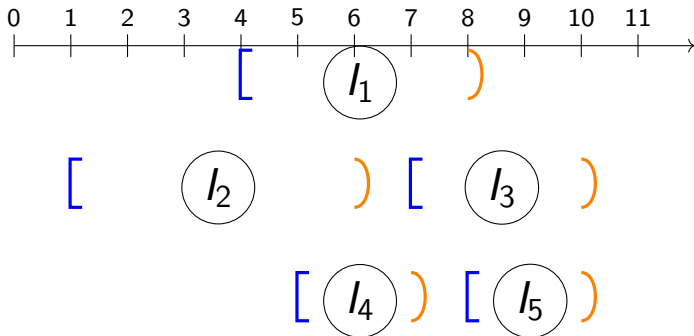
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Vertex cover number vc

Introduction

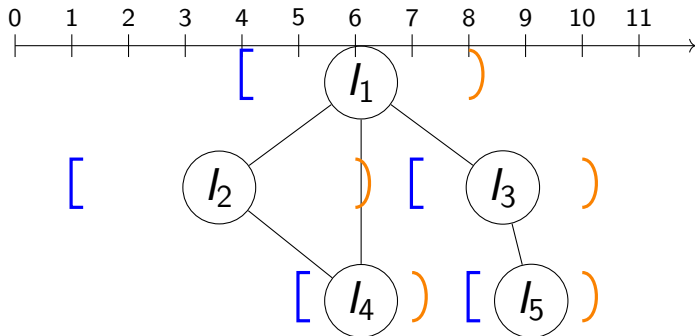
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Vertex cover number vc

Introduction

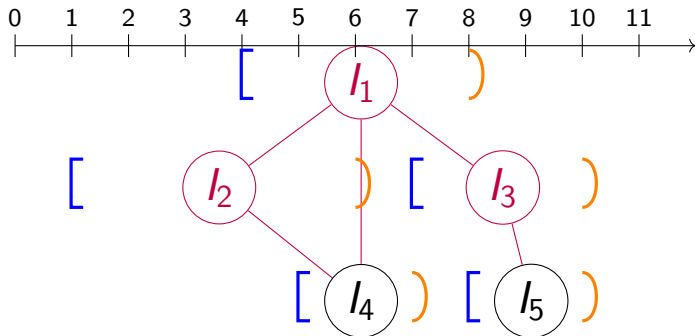
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Vertex cover number vc

Introduction

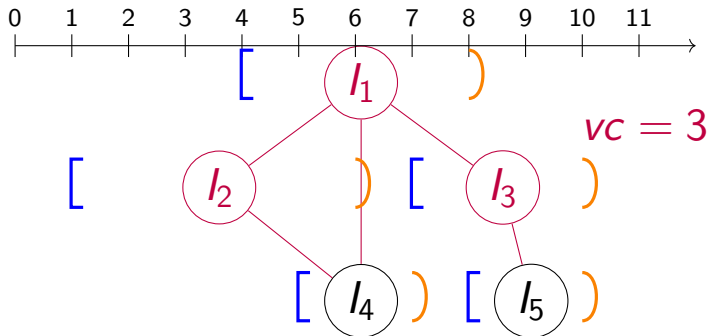
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Definition

$vc = \min_{\mathcal{VC} \text{ vertex cover}} |\mathcal{VC}|.$

Vertex cover over time windows

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References



Definition

$$vc = \min_{\mathcal{VC} \text{ vertex cover}} |\mathcal{VC}|.$$

Vertex cover over time windows

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References



Definition

$$vc = \min_{\mathcal{VC} \text{ vertex cover}} |\mathcal{VC}|.$$

Vertex cover over time windows

Introduction

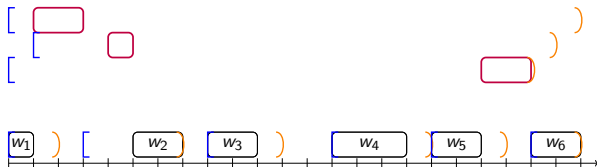
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Definition

$$vc = \min_{\mathcal{VC} \text{ vertex cover}} |\mathcal{VC}|.$$

Vertex cover over time windows

Introduction

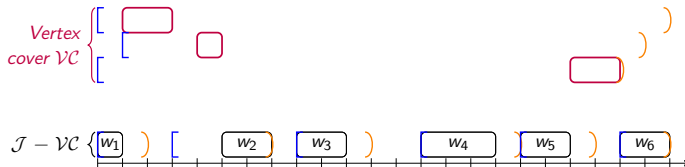
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Definition

$$vc = \min_{\mathcal{VC} \text{ vertex cover}} |\mathcal{VC}|.$$

Dynamic programming (baseline reinterpreted)

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References

Dynamic programming state (k, i_k, V_k)

- k = number of completed jobs.
- i_k = index of the latest completed job in $\mathcal{J} - \mathcal{VC}$.
- $\{w_1, \dots, w_{i_k}\} \cup V_k$ = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.

Dynamic programming (baseline reinterpreted)

Introduction

Vertex cover number

Pathwidth

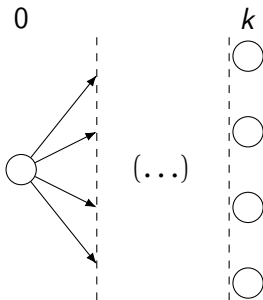
Proper level

Conclusion

References

Dynamic programming state (k, i_k, V_k)

- k = number of completed jobs.
- i_k = index of the latest completed job in $\mathcal{J} - \mathcal{VC}$.
- $\{w_1, \dots, w_{i_k}\} \cup V_k$ = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline reinterpreted)

Introduction

Vertex cover number

Pathwidth

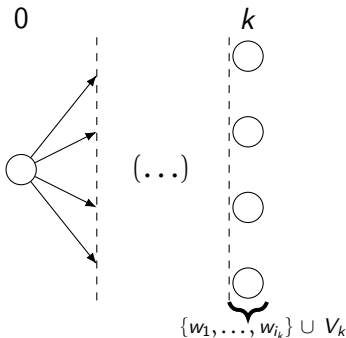
Proper level

Conclusion

References

Dynamic programming state (k, i_k, V_k)

- k = number of completed jobs.
- i_k = index of the latest completed job in $\mathcal{J} - \mathcal{VC}$.
- $\{w_1, \dots, w_{i_k}\} \cup V_k$ = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline reinterpreted)

Introduction

Vertex cover number

Pathwidth

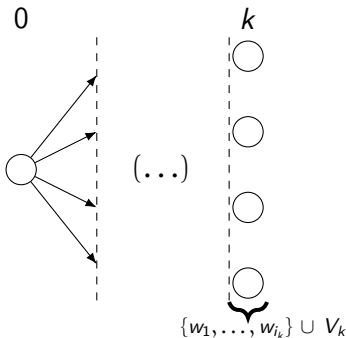
Proper level

Conclusion

References

Dynamic programming state (k, i_k, V_k)

- k = number of completed jobs.
- i_k = index of the latest completed job in $\mathcal{J} - \mathcal{VC}$.
- $\{w_1, \dots, w_{i_k}\} \cup V_k$ = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



$\leq n \cdot 2^{vc}$ states

Dynamic programming (baseline reinterpreted)

Introduction

Vertex cover number

Pathwidth

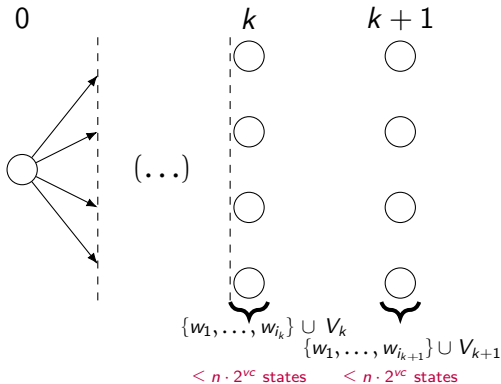
Proper level

Conclusion

References

Dynamic programming state (k, i_k, V_k)

- k = number of completed jobs.
- i_k = index of the latest completed job in $\mathcal{J} - \mathcal{VC}$.
- $\{w_1, \dots, w_{i_k}\} \cup V_k$ = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline reinterpreted)

Introduction

Vertex cover number

Pathwidth

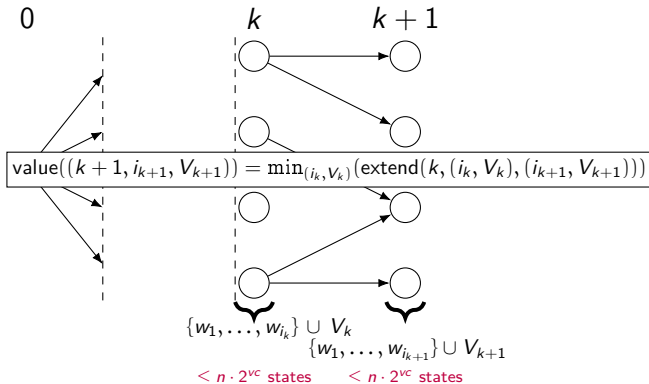
Proper level

Conclusion

References

Dynamic programming state (k, i_k, V_k)

- k = number of completed jobs.
- i_k = index of the latest completed job in $\mathcal{J} - \mathcal{VC}$.
- $\{w_1, \dots, w_{i_k}\} \cup V_k$ = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline reinterpreted)

Introduction

Vertex cover number

Pathwidth

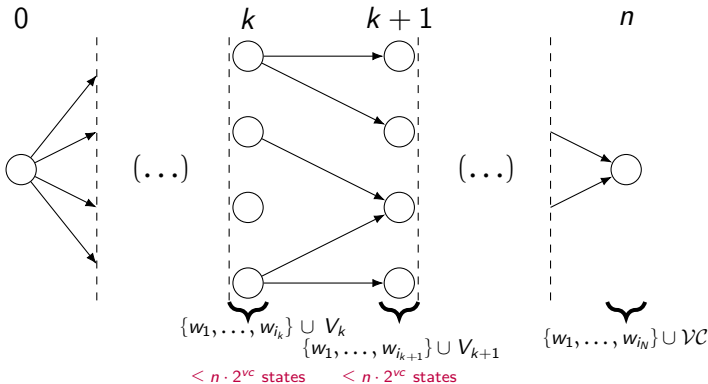
Proper level

Conclusion

References

Dynamic programming state (k, i_k, V_k)

- k = number of completed jobs.
- i_k = index of the latest completed job in $\mathcal{J} - \mathcal{VC}$.
- $\{w_1, \dots, w_{i_k}\} \cup V_k$ = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline reinterpreted)

Introduction

Vertex cover number

Pathwidth

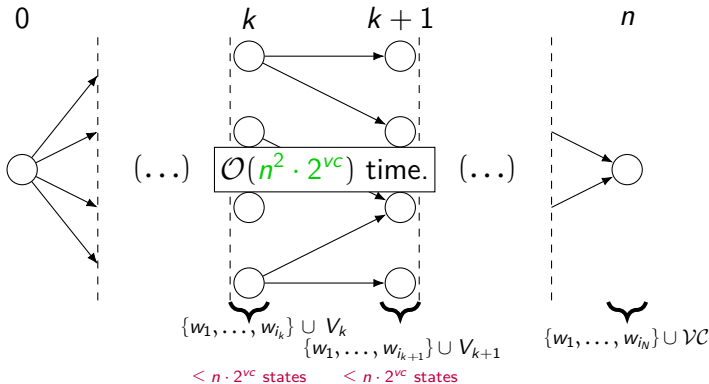
Proper level

Conclusion

References

Dynamic programming state (k, i_k, V_k)

- k = number of completed jobs.
- i_k = index of the latest completed job in $\mathcal{J} - \mathcal{VC}$.
- $\{w_1, \dots, w_{i_k}\} \cup V_k$ = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Dynamic programming (baseline reinterpreted)

Introduction

Vertex cover number

Pathwidth

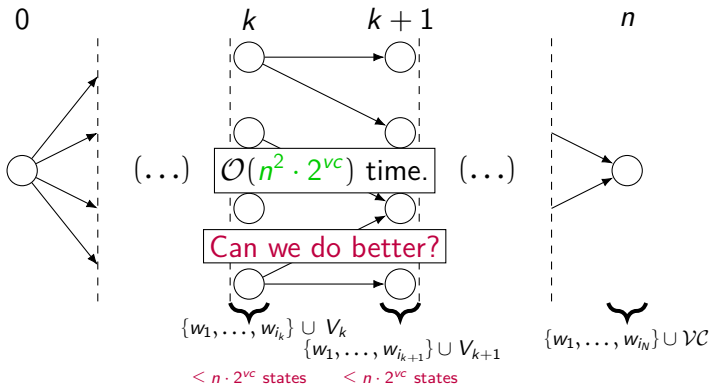
Proper level

Conclusion

References

Dynamic programming state (k, i_k, V_k)

- k = number of completed jobs.
- i_k = index of the latest completed job in $\mathcal{J} - \mathcal{VC}$.
- $\{w_1, \dots, w_{i_k}\} \cup V_k$ = subset of completed jobs.
- State value = minimum makespan among the associated partial schedules.



Outline

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References

1 Introduction

2 Vertex cover number

3 **Pathwidth**

4 Proper level

5 Conclusion

6 References

Pathwidth pw

Introduction

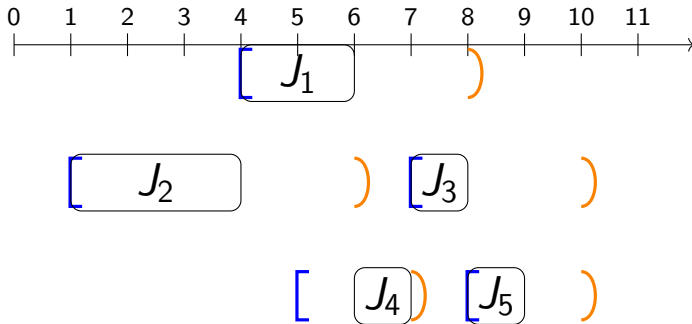
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Pathwidth pw

Introduction

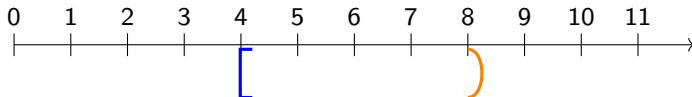
Vertex cover number

Pathwidth

Proper level

Conclusion

References



[

)

[

)

[

)

[

)

Pathwidth pw

Introduction

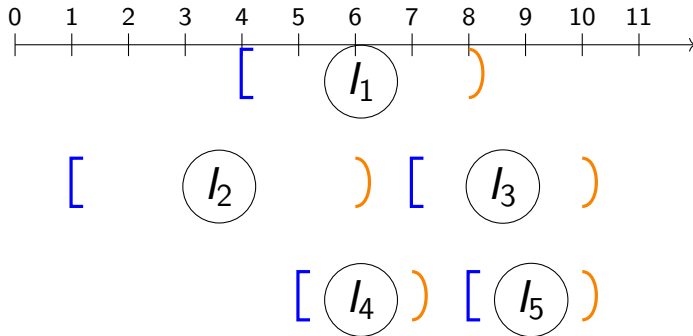
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Pathwidth pw

Introduction

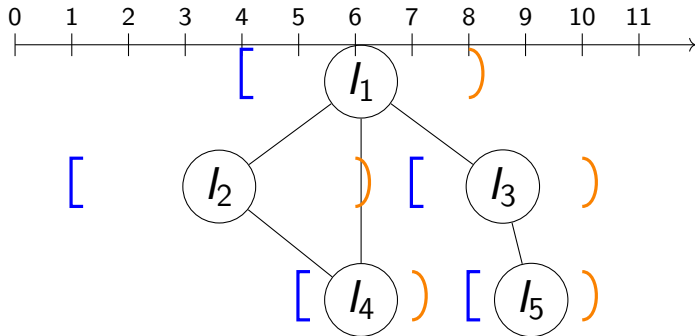
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Pathwidth pw

Introduction

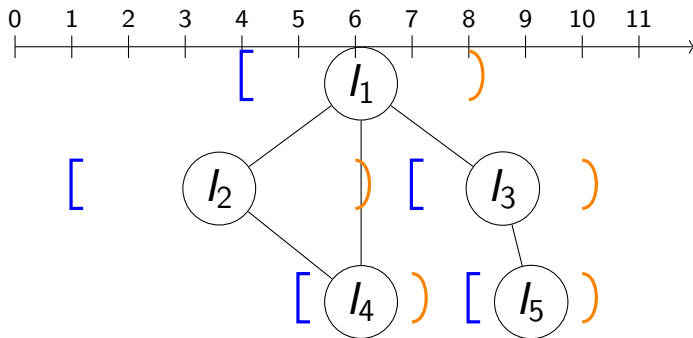
Vertex cover number

Pathwidth

Proper level

Conclusion

References



[Bodlaender, 1993] For any interval graph G :

$$\omega(G) = \chi(G) = tw(G) + 1 = pw(G) + 1.$$

Pathwidth pw

Introduction

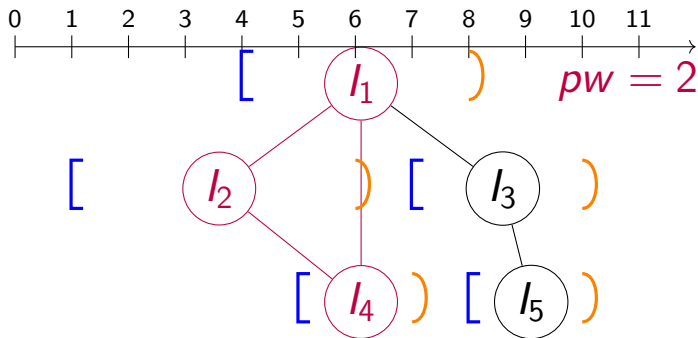
Vertex cover number

Pathwidth

Proper level

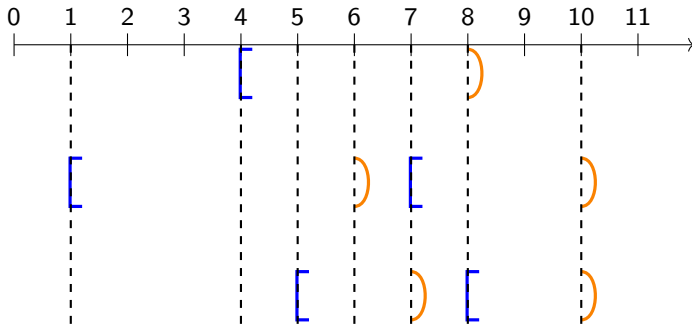
Conclusion

References



[Bodlaender, 1993] For any interval graph G :

$$\omega(G) = \chi(G) = tw(G) + 1 = pw(G) + 1.$$



[Bodlaender, 1993] For any interval graph G :

$$\omega(G) = \chi(G) = tw(G) + 1 = pw(G) + 1.$$

Pathwidth pw

Introduction

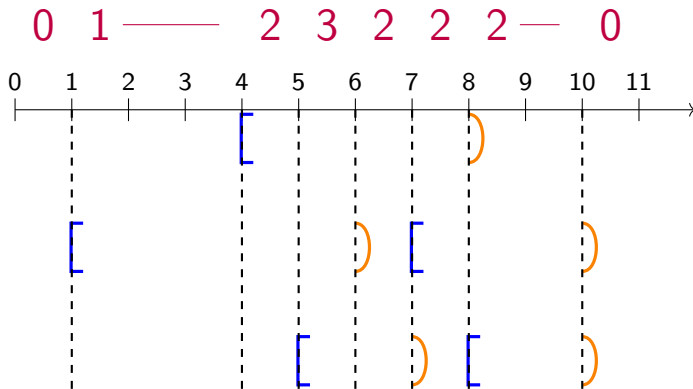
Vertex cover number

Pathwidth

Proper level

Conclusion

References



[Bodlaender, 1993] For any interval graph G :

$$\omega(G) = \chi(G) = tw(G) + 1 = pw(G) + 1.$$

Pathwidth pw

Introduction

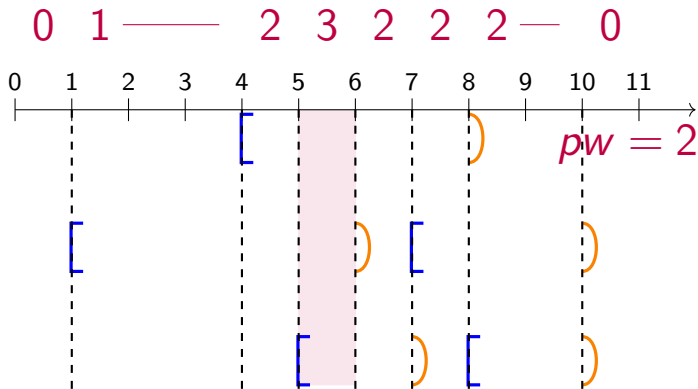
Vertex cover number

Pathwidth

Proper level

Conclusion

References



[Bodlaender, 1993] For any interval graph G :

$$\omega(G) = \chi(G) = tw(G) + 1 = pw(G) + 1.$$

Pathwidth pw

Introduction

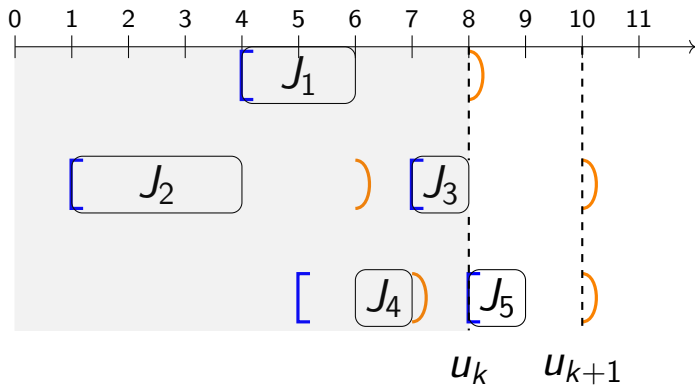
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Pathwidth pw

Introduction

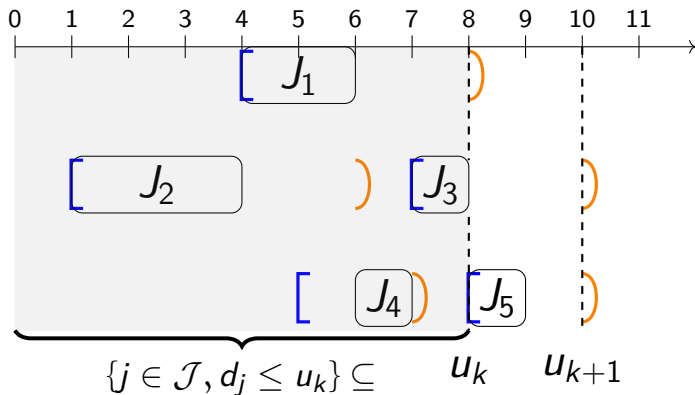
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Pathwidth pw

Introduction

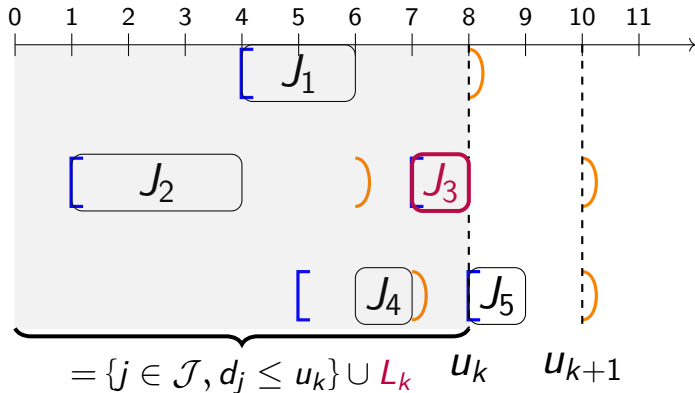
Vertex cover number

Pathwidth

Proper level

Conclusion

References



- $J_j \in L_k \implies [u_k, u_{k+1}) \subseteq [r_j, d_j)$: at most $pw + 1$ such jobs.

Pathwidth pw

Introduction

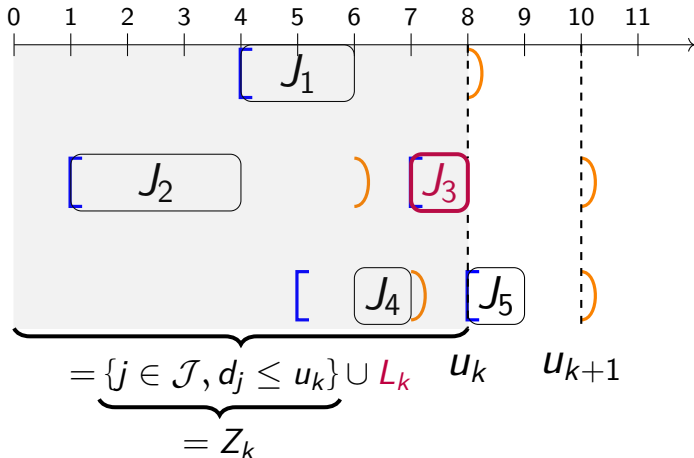
Vertex cover number

Pathwidth

Proper level

Conclusion

References



- $J_j \in L_k \implies [u_k, u_{k+1}) \subseteq [r_j, d_j)$: at most $pw + 1$ such jobs.

Dynamic programming with pathwidth pw

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = subset of jobs completed by time u_k .
- State value = minimum makespan among the associated partial schedules.

Dynamic programming with pathwidth pw

Introduction

Vertex cover number

Pathwidth

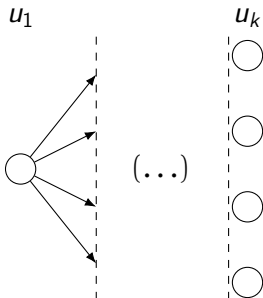
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = subset of jobs completed by time u_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth pw

Introduction

Vertex cover number

Pathwidth

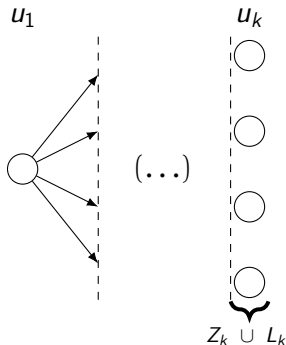
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = subset of jobs completed by time u_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth pw

Introduction

Vertex cover number

Pathwidth

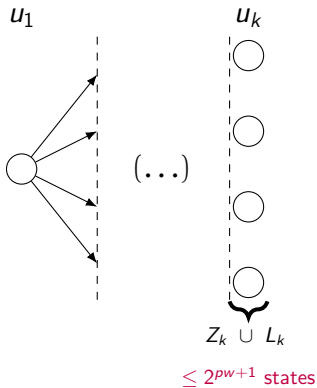
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = subset of jobs completed by time u_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth pw

Introduction

Vertex cover number

Pathwidth

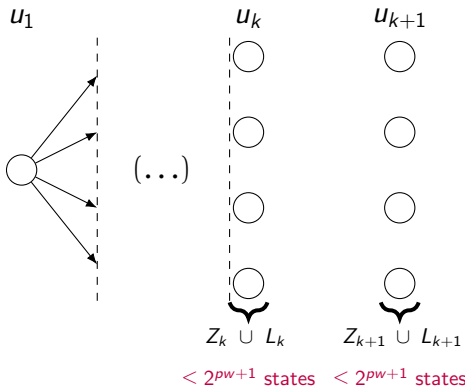
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = subset of jobs completed by time u_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth pw

Introduction

Vertex cover number

Pathwidth

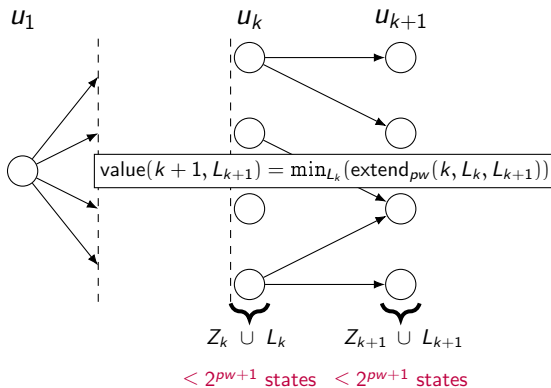
Proper level

Conclusion

References

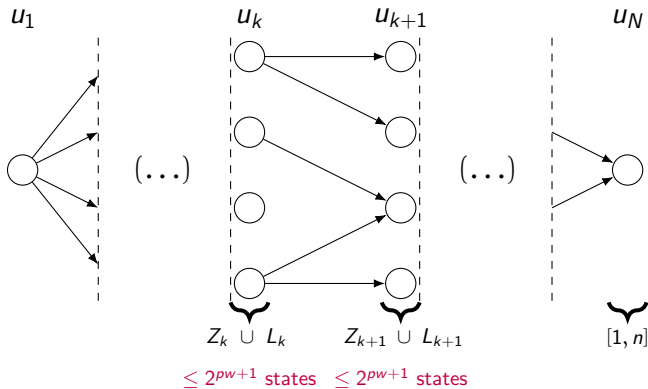
Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = subset of jobs completed by time u_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = subset of jobs completed by time u_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth pw

Introduction

Vertex cover number

Pathwidth

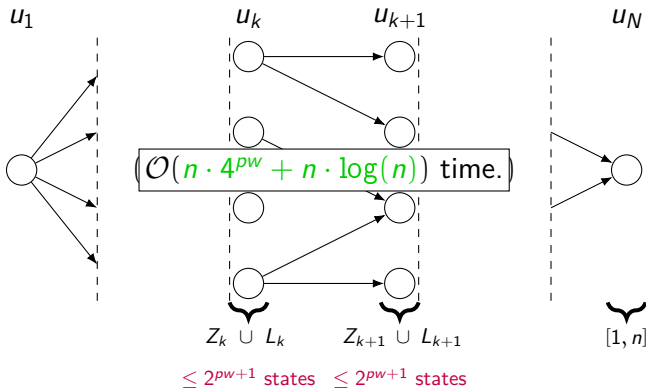
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = subset of jobs completed by time u_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with pathwidth pw

Introduction

Vertex cover number

Pathwidth

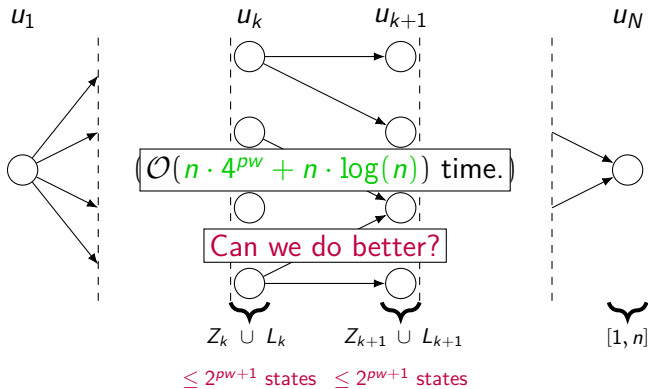
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- u_k = associated release date/deadline value.
- $Z_k \cup L_k$ = subset of jobs completed by time u_k .
- State value = minimum makespan among the associated partial schedules.



Outline

Introduction
Vertex cover number
Pathwidth
Proper level
Conclusion
References

- 1 Introduction
- 2 Vertex cover number
- 3 Pathwidth
- 4 Proper level**
- 5 Conclusion
- 6 References

Proper level

Introduction

Vertex cover number

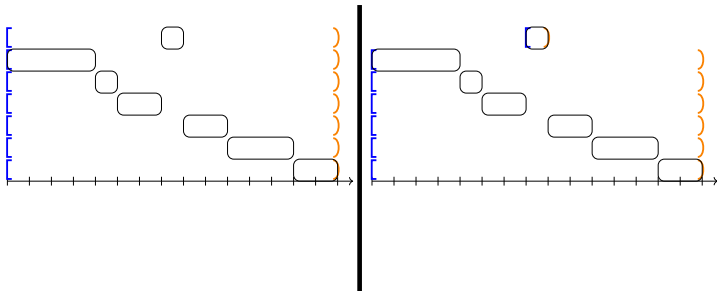
Pathwidth

Proper level

Conclusion

References

- Two single machine instances with time windows.



Proper level

Introduction

Vertex cover number

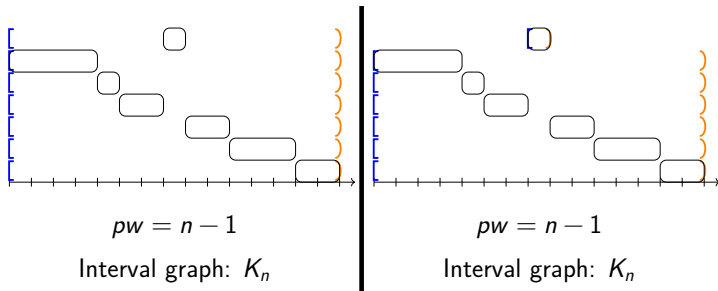
Pathwidth

Proper level

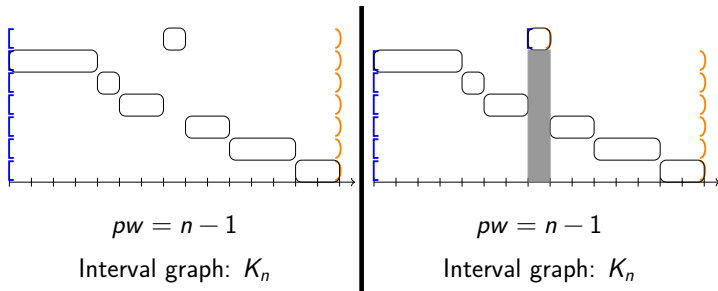
Conclusion

References

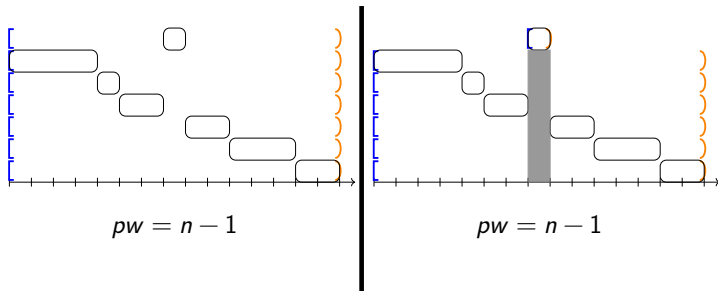
- Two single machine instances with time windows.



- Two single machine instances with time windows.



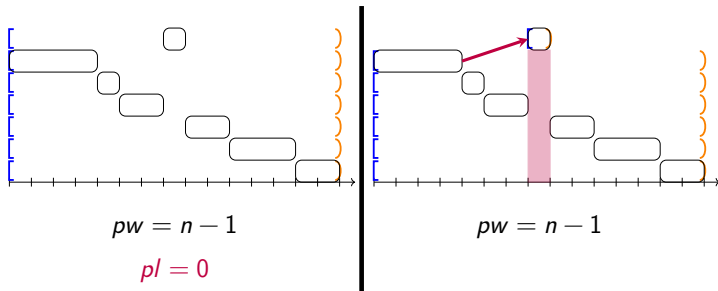
- Two single machine instances with time windows.



Definition: Proper sets and proper level

- $\forall J_i \in \mathcal{J}, \Pi_i = \{J_j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $pl = \max_{J_i \in \mathcal{J}} |\Pi_i|$

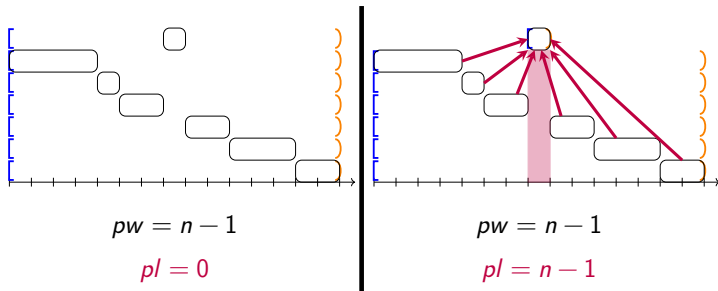
- Two single machine instances with time windows.



Definition: Proper sets and proper level

- $\forall J_i \in \mathcal{J}, \Pi_i = \{J_j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $pl = \max_{J_i \in \mathcal{J}} |\Pi_i|$

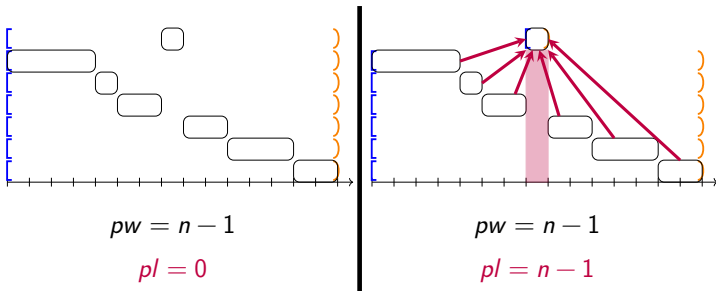
- Two single machine instances with time windows.



Definition: Proper sets and proper level

- $\forall J_i \in \mathcal{J}, \Pi_i = \{J_j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$
- $pl = \max_{J_i \in \mathcal{J}} |\Pi_i|$

- Two single machine instances with time windows.



Definition: Proper sets and proper level

- $\forall J_i \in \mathcal{J}, \Pi_i = \{J_j \in \mathcal{J}, r_j < r_i \wedge d_i < d_j\}$ [Erschler et al., 1985].
- $pl = \max_{J_i \in \mathcal{J}} |\Pi_i|$ [Proskurowski and Telle, 1999].

The “Earliest Deadline first” rule

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References

- “ $\prec$ ” = lexicographic order (nondecr. d_j , nondecr. r_j) on $\mathcal{J}$.
- Renaming the jobs of $\mathcal{J}$ in $[1, n]$ accordingly to $\prec$.

[Lawler and Moore, 1969] “Earliest Deadline first” rule (ED)

Schedule τ follows the (ED) rule if and only if:

$$\forall (i, j) \in \mathcal{J}^2, i \prec j \implies \tau(i) < \tau(j).$$

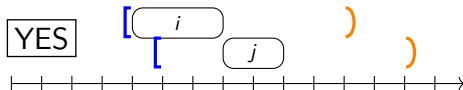
The “Earliest Deadline first” rule

- “ $\prec$ ” = lexicographic order (nondecr. d_j , nondecr. r_j) on $\mathcal{J}$.
- Renaming the jobs of $\mathcal{J}$ in $[1, n]$ accordingly to $\prec$.

[Lawler and Moore, 1969] “Earliest Deadline first” rule (ED)

Schedule τ follows the (ED) rule if and only if:

$$\forall (i, j) \in \mathcal{J}^2, i \prec j \implies \tau(i) < \tau(j).$$



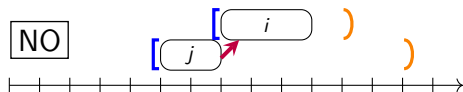
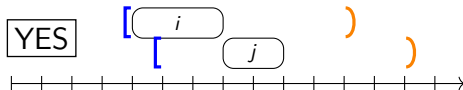
The “Earliest Deadline first” rule

- “ $\prec$ ” = lexicographic order (nondecr. d_j , nondecr. r_j) on $\mathcal{J}$.
- Renaming the jobs of $\mathcal{J}$ in $[1, n]$ accordingly to $\prec$.

[Lawler and Moore, 1969] “Earliest Deadline first” rule (ED)

Schedule τ follows the (ED) rule if and only if:

$$\forall (i, j) \in \mathcal{J}^2, i \prec j \implies \tau(i) < \tau(j).$$



Relaxing the “Earliest Deadline first” rule

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References

Definition: “weak Earliest Deadline first” rule (wED)

Schedule τ follows the (wED) rule if and only if:

$\forall (i, j) \in \mathcal{J}^2, i \prec j \implies$

(1) $\tau(i) < \tau(j)$, or

(2) $j \in \Pi_i$, or

(3) $\exists h \prec i, \tau(j) < \tau(h) < \tau(i)$.

Relaxing the “Earliest Deadline first” rule

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

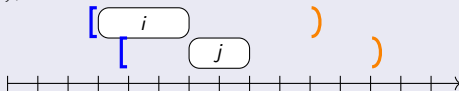
References

Definition: “weak Earliest Deadline first” rule (wED)

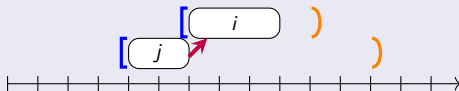
Schedule τ follows the (wED) rule if and only if:

$$\forall (i, j) \in \mathcal{J}^2, i \prec j \implies$$

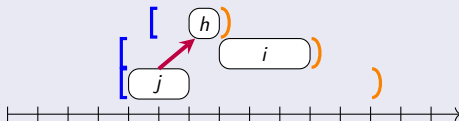
(1) $\tau(i) < \tau(j)$, or



(2) $j \in \Pi_i$, or



(3) $\exists h \prec i, \tau(j) < \tau(h) < \tau(i)$.



Relaxing the “Earliest Deadline first” rule

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

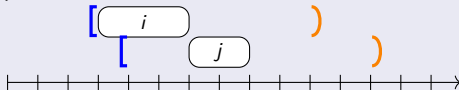
References

Definition: “weak Earliest Deadline first” rule (wED)

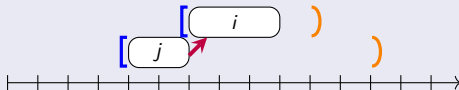
Schedule τ follows the (wED) rule if and only if:

$$\forall (i, j) \in \mathcal{J}^2, i \prec j \implies$$

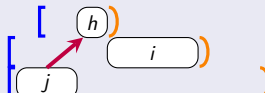
(1) $\tau(i) < \tau(j)$, or



(2) $j \in \Pi_i$, or



(3) $\exists h \prec i, \tau(j) < \tau(h) < \tau(i)$.



Proposition

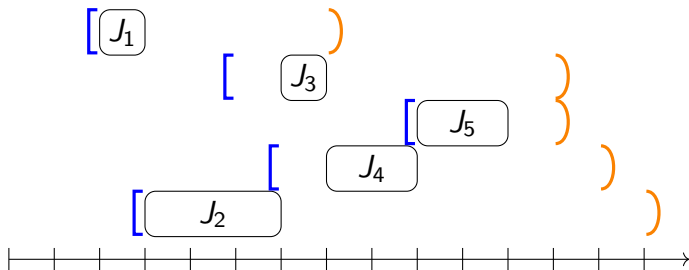
On $1|r_j, d_j|C_{max}$ the (wED)-following schedules are dominant.

Definition: Summit

A **summit** is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.

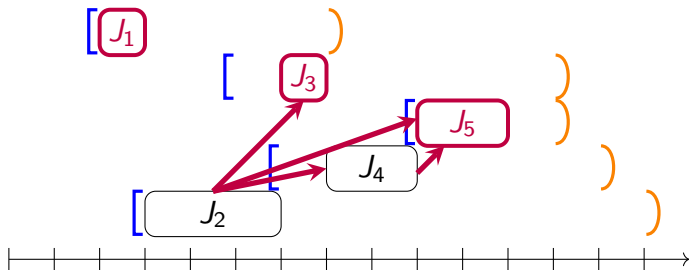
Definition: Summit

A **summit** is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.



Definition: Summit

A **summit** is a job i in $\mathcal{J}$ such that: $\forall j \in \mathcal{J}, i \notin \Pi_j$.



Lemma

Let $s_1 \prec \dots \prec s_N$ be the summits of the instance ($N \leq n$).

If a schedule τ follows the (wED) rule, then:

- (i) τ is of the form $T_1 s_1 \dots s_k T_{k+1} s_{k+1} \dots s_N T_{N+1}$,
- (ii) if $i \prec s_k$ then $\tau(i) < \tau(s_k)$,
- (iii) if $s_k \prec j$ and $\tau(j) < \tau(s_k)$ then $j \in \Pi_{s_k}$.

Summit-based schedule enumeration

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

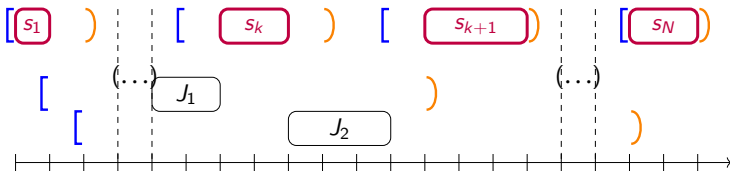
References

Lemma

Let $s_1 \prec \dots \prec s_N$ be the summits of the instance ($N \leq n$).

If a schedule τ follows the (wED) rule, then:

- (i) τ is of the form $T_1 s_1 \dots s_k T_{k+1} s_{k+1} \dots s_N T_{N+1}$,
- (ii) if $i \prec s_k$ then $\tau(i) < \tau(s_k)$,
- (iii) if $s_k \prec j$ and $\tau(j) < \tau(s_k)$ then $j \in \Pi_{s_k}$.



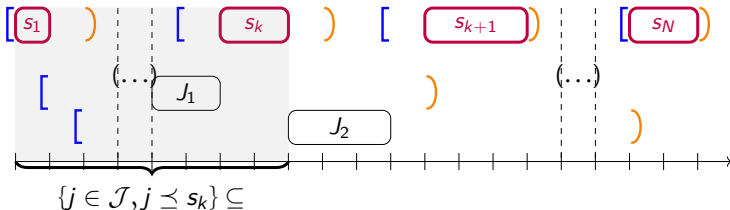
Summit-based schedule enumeration

Lemma

Let $s_1 \prec \dots \prec s_N$ be the summits of the instance ($N \leq n$).

If a schedule τ follows the (wED) rule, then:

- (i) τ is of the form $T_1 s_1 \dots s_k T_{k+1} s_{k+1} \dots s_N T_{N+1}$,
- (ii) if $i \prec s_k$ then $\tau(i) < \tau(s_k)$,
- (iii) if $s_k \prec j$ and $\tau(j) < \tau(s_k)$ then $j \in \Pi_{s_k}$.



Summit-based schedule enumeration

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

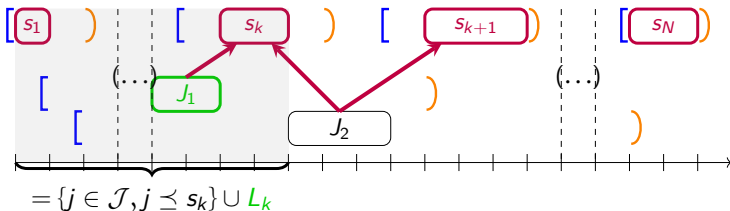
References

Lemma

Let $s_1 \prec \dots \prec s_N$ be the summits of the instance ($N \leq n$).

If a schedule τ follows the (wED) rule, then:

- (i) τ is of the form $T_1 s_1 \dots s_k T_{k+1} s_{k+1} \dots s_N T_{N+1}$,
- (ii) if $i \prec s_k$ then $\tau(i) < \tau(s_k)$,
- (iii) if $s_k \prec j$ and $\tau(j) < \tau(s_k)$ then $j \in \Pi_{s_k}$.



- $j \in L_k \implies j \in \Pi_{s_k}$: at most $p!$ such jobs j .

Dynamic programming with proper level pl

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- s_k = associated summit.
- $[1, s_k] \cup L_k$ = subset of jobs completed up to summit s_k .
- State value = minimum makespan among the associated partial schedules.

Dynamic programming with proper level pl

Introduction

Vertex cover number

Pathwidth

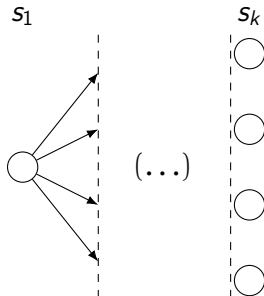
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- s_k = associated summit.
- $[1, s_k] \cup L_k$ = subset of jobs completed up to summit s_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with proper level p_l

Introduction

Vertex cover number

Pathwidth

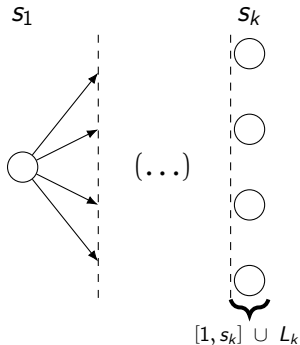
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- s_k = associated summit.
- $[1, s_k] \cup L_k$ = subset of jobs completed up to summit s_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with proper level p_l

Introduction

Vertex cover number

Pathwidth

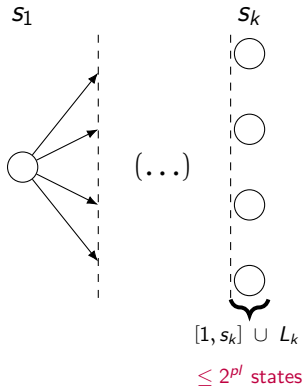
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- s_k = associated summit.
- $[1, s_k] \cup L_k$ = subset of jobs completed up to summit s_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with proper level pl

Introduction

Vertex cover number

Pathwidth

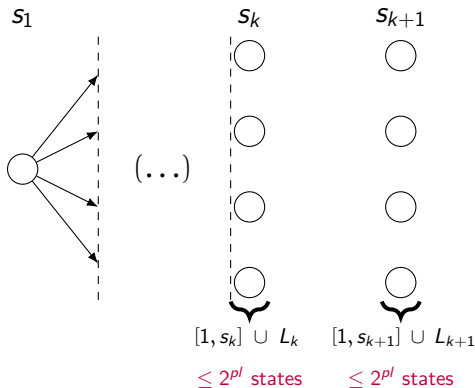
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- s_k = associated summit.
- $[1, s_k] \cup L_k$ = subset of jobs completed up to summit s_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with proper level pl

Introduction

Vertex cover number

Pathwidth

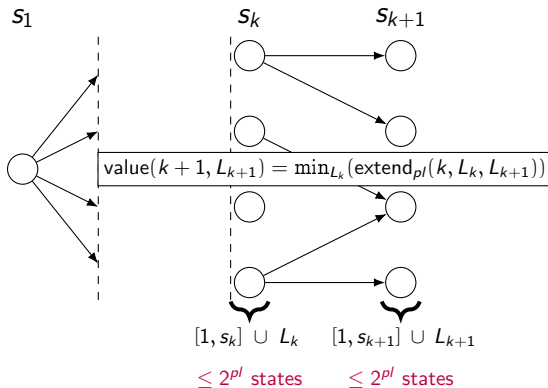
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- s_k = associated summit.
- $[1, s_k] \cup L_k$ = subset of jobs completed up to summit s_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with proper level pl

Introduction

Vertex cover number

Pathwidth

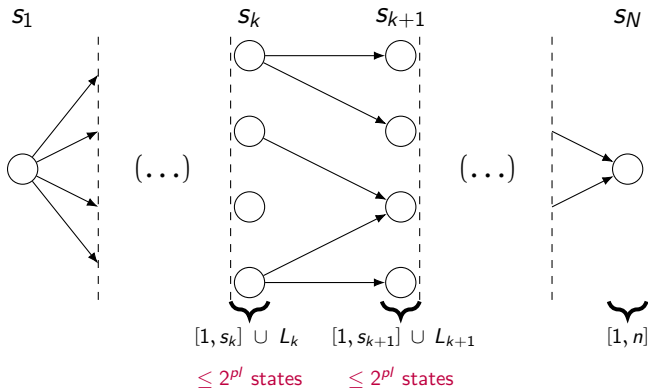
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- s_k = associated summit.
- $[1, s_k] \cup L_k$ = subset of jobs completed up to summit s_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with proper level pl

Introduction

Vertex cover number

Pathwidth

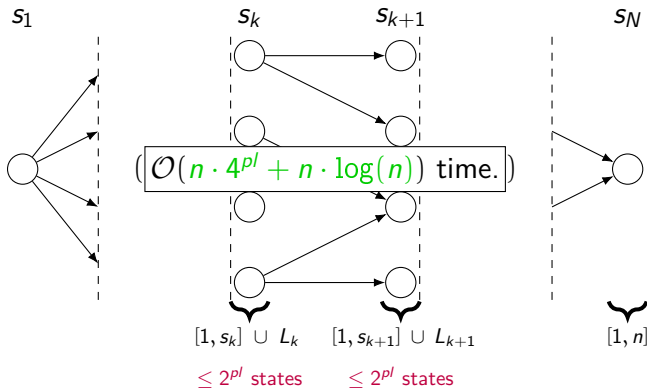
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- s_k = associated summit.
- $[1, s_k] \cup L_k$ = subset of jobs completed up to summit s_k .
- State value = minimum makespan among the associated partial schedules.



Dynamic programming with proper level p_l

Introduction

Vertex cover number

Pathwidth

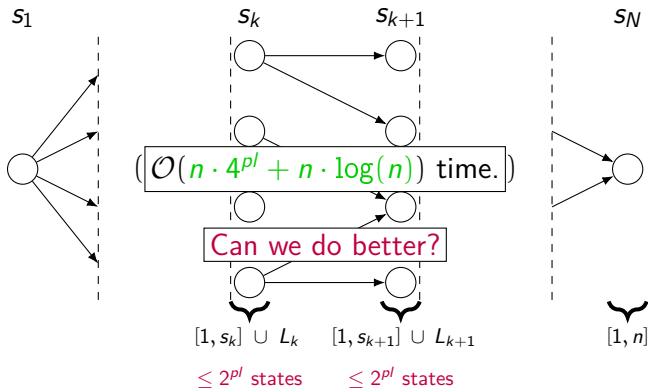
Proper level

Conclusion

References

Dynamic programming state (k, L_k)

- s_k = associated summit.
- $[1, s_k] \cup L_k$ = subset of jobs completed up to summit s_k .
- State value = minimum makespan among the associated partial schedules.



Outline

Introduction
Vertex cover number
Pathwidth
Proper level
Conclusion
References

- 1 Introduction
- 2 Vertex cover number
- 3 Pathwidth
- 4 Proper level
- 5 Conclusion**
- 6 References

Takeaway

- Better understanding of scheduling problems.
- More efficient exact algorithms.

Takeaway

- Better understanding of scheduling problems.
- More efficient exact algorithms.

Going further

- $4^{p_l} \rightarrow 2^{p_l}$.
- Generalizations: parallel machines, periodic time windows.
- Parameterized **approximation** algorithms.
- Parameterized hardness: **para-NP**, **W-hierarchy**, **XNLP**.

Takeaway

- Better understanding of scheduling problems.
- More efficient exact algorithms.

Going further

- $4^{p_l} \rightarrow 2^{p_l}$.
- Generalizations: parallel machines, periodic time windows.
- Parameterized **approximation** algorithms.
- Parameterized hardness: **para-NP**, **W-hierarchy**, **XNLP**.

Recent news

- IWOCA (with Bodlaender): scheduling with precedence delays.
- FUN!

Parameter hierarchy on $1|r_j, d_j| \star$

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References



Parameter hierarchy on $1|r_j, d_j| \star$

Introduction

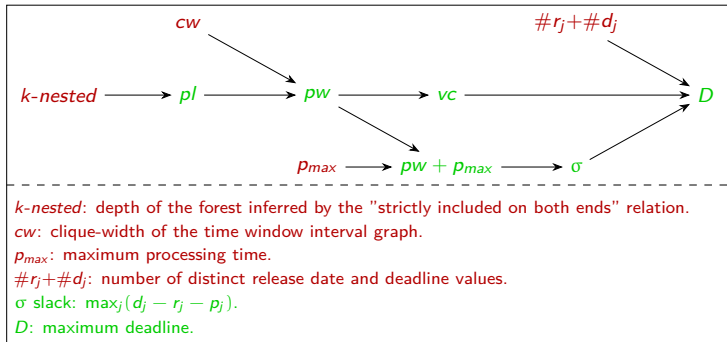
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Parameterized complexity classes

Introduction

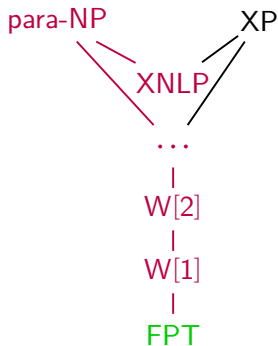
Vertex cover number

Pathwidth

Proper level

Conclusion

References



Parameterized complexity classes

Introduction

Vertex cover number

Pathwidth

Proper level

Conclusion

References

nondet. $f(k) \cdot \text{poly}(|\mathcal{I}|)$ time

det. $|\mathcal{I}|^{f(k)}$ time

para-NP

XP

XNLP

nondet. $f(k) \cdot \text{poly}(|\mathcal{I}|)$ time
& $g(k) \cdot \log(|\mathcal{I}|)$ space

...

W[2]

W[1]

FPT

det. $f(k) \cdot \text{poly}(|\mathcal{I}|)$ time



Erschler, J., Fontan, G., and Merce, C. (1985).

Un nouveau concept de dominance pour l'ordonnement de travaux sur une machine.

RAIRO-Operations Research, 19(1):15–26.



Flum, J. and Grohe, M. (1998).

Parameterized Complexity Theory.

Springer.



Lawler, E. L. and Moore, J. M. (1969).

A functional equation and its application to resource allocation and sequencing problems.

Management science, 16(1):77–84.



Lenstra, J., Rinnooy Kan, A., and Brucker, P. (1977).

Complexity of machine scheduling problems.

Ann. of Discrete Math., 1:343–362.



Proskurowski, A. and Telle, J. A. (1999).

Classes of graphs with restricted interval models.

Discrete Mathematics & Theoretical Computer Science, 3.