REVUE FRANÇAISE D'AUTOMATIQUE, D'INFORMATIQUE ET DE RECHERCHE OPÉRATIONNELLE. RECHERCHE OPÉRATIONNELLE

J.-P. BORDAT

Calcul des idéaux d'un ordonné fini

Revue française d'automatique, d'informatique et de recherche opérationnelle. Recherche opérationnelle, tome 25, n° 3 (1991), p. 265-275.

<http://www.numdam.org/item?id=RO_1991__25_3_265_0>

Recherche opérationnelle/Operations Research
(vol. 25, n° 3, 1991, p. 265 à 275)

CALCUL DES IDÉAUX D'UN ORDONNÉ FINI (*)

par J.-P. BORDAT «([1])

Résumé. – *Les idéaux d'un ordonné fini P et leur structure de treillis sont utilisés dans plusieurs problèmes d'ordonnancement, de flots dans les réseaux et d'ensembles ordonnés. Nous présentons un algorithme engendrant ces idéaux dans une complexité $O(w(P))$ pour chaque idéal, $w(P)$ étant la largeur de P. Le treillis distributif peut être engendré dans la même complexité.*

Mots clés : Idéal; ordonnancement; treillis distributif; réseaux; extension linéaire.

Abstract. – *The ideals of a finite poset P and their lattice structure provide a useful tool for several problems in scheduling, network flows, and other poset problems. We present an algorithm which generates these ideals in $O(w(P))$ time per ideal, where $w(P)$ is the width of P. The distributive lattice may be generated in the same complexity.*

Keywords : Order ideal; scheduling; distributive lattice; networks; linear extension.

I. INTRODUCTION

Soit $P=(\{1,2,\ldots,n\}\leqq)$ un ensemble ordonné fini. Il existe une bijection entre l'ensemble des idéaux et l'ensemble des antichaînes de P. Le calcul de ces deux ensembles, dont on notera par $N(P)$ le cardinal, a des applications dans deux domaines particuliers :

– Optimisation Combinatoire et Recherche Opérationnelle :

Sont concernés essentiellement des problèmes d'emploi du temps avec contraintes et de recherche de coupes minimales dans les réseaux : en effet, la génération de toutes les coupes minimales d'un réseau peut s'effectuer par énumération des idéaux d'un ensemble ordonné auxiliaire [11] et s'applique au problème du stable maximal (vertex packing problem) dans un graphe non orienté muni de poids sur l'ensemble des sommets [10]. En outre, la

(*) Reçu janvier 1990.
([1]) C.R.I.M., 860, rue de St-Priest, 34090 Montpellier.

Recherche opérationnelle/Operations Research, 0399-0559/91/03 265 11/$ 3.10

génération de tous les idéaux de P (par des algorithmes de type programmation dynamique) est utilisée dans des problèmes d'ordonnancement avec contraintes de précédence : l'ensemble des *tâches* ou *activités* est partiellement ordonné par ces contraintes, et doit être ordonné totalement de façon à minimiser une fonction de coût donnée, différentes mesures de performance pouvant être définies (*voir* [7] pour une classification). La génération des antichaînes de P s'applique également à des problèmes d'ordonnancement avec contraintes de ressources [8]. Le lecteur trouvera dans [16] un excellent exposé sur ces sujets, ainsi qu'une bibliographie exhaustive.

– Algorithme des ordonnés finis :

$N(P)$ est relativement grand par rapport à n et son calcul est un problème $\#P$-complet [13]. Néanmoins la structure du treillis distributif $\mathscr{T}(P)$ des idéaux de P présente de l'intérêt. Elle jette une lumière particulière sur des problèmes NP-complets de calcul d'invariants dans les ordonnés finis [2]. Les rapports entre les deux domaines (problèmes d'ordonnancement et ordonnés finis) sont bien décrits dans deux ouvrages de synthèse [9 et 12].

La plupart des algorithmes de calcul des idéaux de P s'inspirent de techniques ensemblistes et ne prennent pas en compte la structure privilégiée de l'ensemble des idéaux qui est celle de treillis distributif. Plusieurs publications présentent ainsi des algorithmes de complexité $O(N(P).n^2)$ [6, 14], ou $O(N(P).n)$ [16]. L'algorithme proposé ici a pour complexité $O(N(P).w(P))$, où $w(P)$ est la *largeur* de P, encore égale, d'après le théorème de Dilworth, au cardinal minimal d'une décomposition en chaînes de cet ordre [4].

Cet algorithme s'inspire directement d'une étude des propriétés des arborescences recouvrantes d'un treillis distributif [1]. Nous proposons après son exposé un autre algorithme générant $\mathscr{T}(P)$. Le calcul d'invariants sera évoqué en précisant les résultats obtenus et en établissant le lien avec les travaux existant sur ce sujet.

II. RAPPELS ET NOTATIONS

Nous supposons connues les notions fondamentales de théorie des graphes, et de théorie des ordonnés finis.

– $I \subseteqq P = (\{1, 2, \ldots, n\}, \leqq)$ est un idéal de P si et seulement si

$$\forall i \in I, \quad \forall j \in P, \qquad j \leqq i \;\Rightarrow\; j \in I.$$

– $AC \subseteqq P$ est une *antichaîne* de P si deux éléments quelconques de AC sont incomparables dans P. Si I est un idéal, l'application max telle que

$\max(I) = \{ i \in I \mid \nexists j \in I \, i < j \}$ est une bijection entre l'ensemble des idéaux et l'ensemble des antichaînes de P.

– L'ensemble des idéaux de P ordonné par inclusion a une structure de *treillis distributif* [5], noté $\mathcal{T}(P)$. P et $\mathcal{T}(P)$ seront représentés par leurs graphes de *couverture* (ou graphes de Hasse) respectifs $G(P)$ et $H(P)$, ayant pour fonctions successeur respectives γ^+ et Γ^+.

– $H(P)$ possède une source unique notée X_0 et un puits unique noté X_P. Si l'on note $\mathcal{P}(X)$ la partie de $\{1, 2, \ldots, n\}$ associée au sommet X, $\mathcal{P}(X_0) = \varnothing$ et $\mathcal{P}(X_P) = \{1, 2, \ldots, n\}$.

– $\max \{ |\Gamma^+(X)| \mid X \text{ sommet de } H(P) \} = w(P)$, largeur de P (c'est-à-dire cardinal maximal d'une antichaîne de P) et $w(P) \leqq \log N(P)$.

– Chaque arc XY de $H(P)$ peut être *étiqueté* par $e(XY) = i \in P$, où $i = \mathcal{P}(Y) - \mathcal{P}(X)$. $\mathcal{P}(X)$ n'est autre que l'union des étiquettes d'un chemin quelconque d'origine X_0 et d'extrémité X. Par conséquent, l'ensemble des chemins de longueur maximale (n) dans $H(P)$ correspond bijectivement à l'ensemble des ordres totaux compatibles avec P.

Ces propriétés apparaissent clairement sur l'exemple de la figure 1.

III. GÉNÉRATION DES IDÉAUX DE P

Dans [1] sont étudiées les propriétés d'un parcours (en profondeur ou en largeur) dans le graphe de Hasse d'un treillis distributif quelconque. Un tel parcours induit une bipartition sur l'ensemble des arcs :

– arcs de l'*arborescence* recouvrante associée au parcours [en traits épais (*fig.* 1)].

– arcs *traversiers* (en traits fins).

Cette bipartition possède des propriétés remarquables relativement aux étiquettes des arcs : soient :

– ν la numérotation donnant l'ordre dans lequel les sommets sont atteints.

– $\mathrm{Pr}(X)$ le prédécesseur du sommet X dans l'arborescence recouvrante.

– $E(X)$ l'ensemble des étiquettes des arcs issus de X, union disjointe de $ET(X)$ (arcs traversiers) et $EA(X)$ (arcs de l'arborescence). ν induit un ordre total strict $\triangledown$ sur $ET(X)$, $EA(X)$, $E(X)$.
Alors on établit le résultat suivant [1] : $ET(X) = \{ e(\mathrm{Pr}(X) Z) \mid \nu(Z) < \nu(X) \}$, ces deux ensembles étant ordonnés par $\triangledown$ (voir *fig.* 1).

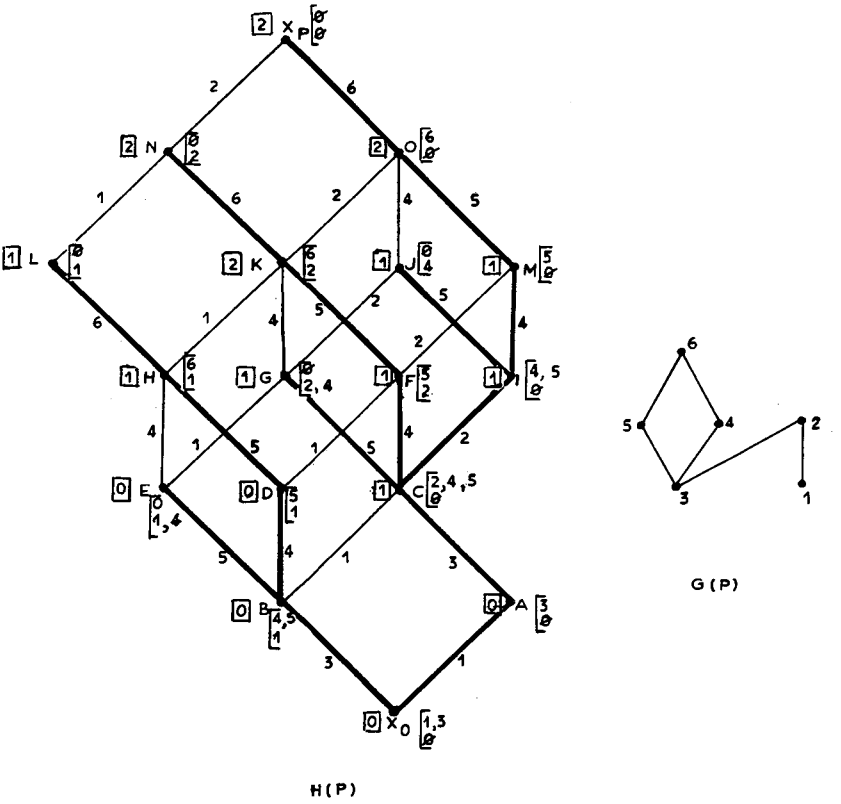

Figure 1.

En d'autres termes, l'ensemble d'étiquettes des arcs traversiers issus d'un sommet X peut être obtenu facilement à partir de l'ensemble d'étiquettes des arcs issus de son prédécesseur $\mathrm{Pr}(X)$ dans l'arborescence.

Conséquence

Le problème abordé ici est celui de la génération des idéaux de P. On obtiendra facilement ces idéaux, non pas en engendrant le graphe $H(P)$ en entier, mais en simulant un parcours constructif de ce graphe, c'est-à-dire en *engendrant uniquement une arborescence recouvrante étiquetée*, et ceci en profondeur d'abord pour des raisons de complexité. A chaque étape élémentaire de l'algorithme, $E(X)$ est constitué des éléments minimaux de l'ensemble

$P - \mathscr{P}(X)$ ordonné par l'ordre induit et donc aisément calculables à partir du graphe de Hasse de P.

Pour engendrer l'arborescence, on retiendra uniquement les étiquettes de l'ensemble $EA(X)$.

La figure 1 précise, pour chaque sommet X de $H(P)$, les ensembles $EA(X)$ et $ET(X)$, ordonnés par ∇. Afin de ne pas surcharger le dessin, les flèches sont omises, et les arcs considérés comme orientés de bas en haut, convention communément admise pour les graphes de Hasse.

L'algorithme se décrit agréablement de manière récursive :

```
                { Initialisation des appels récursifs }
{ Donnée : γ+ (resp. γ−) fonction successeur (resp. prédécesseur) de G(P) }
Pour i ∈ {1, 2, ..., n} faire d⁻(i) ← |γ⁻(i)|;
𝒫(X0) ← ET(X0) ← ∅; E(X0) ← EA(X0) ← {i | d⁻(i) = 0};
PROFIDEAL (X0);
  { ∇ est initialisé sur E(X0), et se construit lors des appels récursifs }
Procédure PROFIDEAL (X);
    Pour i ∈ EA(X) faire
        { création de l'idéal Y tel que 𝒫(Y) = 𝒫(X) ∪ {i} }
    𝒫(Y) ← 𝒫(X) ∪ {i};
    ET(Y) ← {j ∈ E(X) | j ∇ i}; EA(Y) ← {j ∈ E(X) | i ∇ j};
        { ET(Y) et EA(Y) sont ainsi ordonnés par △ }
    Pour j ∈ γ+(i) faire
        d⁻(j) ← d⁻(j) − 1;
        Si d⁻(j) = 0 alors EA(Y) ← EA(Y) ∪ {j};
    { △ est complété sur EA(Y) puis défini sur E(Y) : ET suivi de EA }
    E(Y) ← ET(Y) ∪ EA(Y);
    PROFIDEAL (Y);
        { Restauration du tableau d }
    Pour j ∈ γ+(i) faire d⁻(j) ← d⁻(j) + 1;
```

Complexité : Deux listes chaînées totalement ordonnées représenteront $EA(X)$ et $ET(X)$. $E(X)$ s'obtient par balayage de ces listes dans l'ordre ET puis EA. La complexité de la procédure est alors donnée par les deux boucles *Pour* $j \in \gamma^+(i)$. Chacune demande $O(w(P))$ opérations dans $G(P)$. PROFIDEAL a donc une complexité en $O(N(P).w(P))$. Notons qu'une place mémoire en $O(n)$ est affectée à chaque appel récursif. Le nombre maximal d'appels emboîtés à un instant donné étant majoré par n, la place-mémoire utilisée est $O(n^2)$.

IV. GÉNÉRATION DE $H(P)$

Dans la suite, si i est l'étiquette d'un arc XY, on notera sans ambiguité Y par $X.i$.

Si l'on veut maintenant engendrer le graphe $H(P)$ entier, il est nécessaire de considérer les arcs traversiers issus d'un sommet X. Soit XZ un arc traversier tel que $e(XZ)=i$. Si $e(\mathrm{Pr}(X)X)=j$, on remarque (*fig.* 2) que $Z=(\mathrm{pr}(X).i).j$.

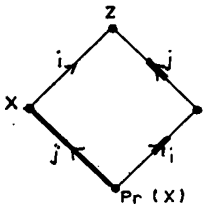

Figure 2.

On obtient la procédure HASSEIDEAL:

Procédure HASSEIDEAL (X);
$j \leftarrow e(\mathrm{Pr}(X)X)$; $\{X \neq X_0\}$
Pour $i \in EA(X)$ *faire*
$\mathrm{Pr}(Y) \leftarrow X$; SOMMET (Y); ARC(XY);
...
Pour $i \in ET(X)$ *faire*
$Z \leftarrow e(\mathrm{Pr}(X).i).j$; ARC$(XZ)$;

La mémorisation de $\mathrm{Pr}(X)$ et $e(\mathrm{Pr}(X)X)$ ne pose aucune difficulté. Les procédures SOMMET et ARC consistent à créer un nouveau sommet et un arc dans $G(P)$ et ont une complexité $O(1)$ si on utilise des listes d'adjacence. Par contre, pour tout sommet X on doit connaître la table des $X.i$, $i \in E(X)$ et les complexités en temps et en mémoire sont $O(N(P)w(P))$.

Il est possible de réduire cette occupation mémoire en remarquant que la table des $X.i$ doit être connue seulement pour deux niveaux consécutifs de $G(P)$. Une solution consiste à :

– générer en une première étape l'arborescence recouvrante;

– greffer les arcs traversiers par un parcours de l'arborescence en largeur en ne mémorisant les $X.i$ que sur deux niveaux consécutifs.

V. CALCUL D'INVARIANTS

DÉFINITION : Soit $t = x_1 x_2 \ldots x_n$ un ordre total des éléments de P. t est une *extension linéaire* de P si $x \leqq y$ implique x précède y dans t ($x \leqq_t y$). Deux éléments consécutifs x_i et x_{i+1} sont séparés par un *saut* si x_i et x_{i+1} sont non comparables dans P. Soit $s(t, P)$ le nombre de sauts de t.

Nous désignerons par $L(P)$ le nombre d'extensions linéaires de P, et soit $s(P)$ le min des $s(t, P)$ sur l'ensemble des extensions linéaires de P. Enfin, $\dim(P)$, *dimension* de P, est le nombre minimal d'extensions linéaires de P dont l'intersection restitue P.

$s(P)$, $L(P)$ et $\dim(P)$ sont des *invariants de comparabilité* de P, c'est-à-dire qu'ils ont la même valeur pour tous les ordres ayant des graphes de comparabilité isomorphes [$CG(P)$, graphe de comparabilité de P a pour ensemble de sommets $\{1, 2, \ldots, n\}$ et pour ensemble d'arêtes $\{(i,j) \mid i <_P j\}$].

[12] est une excellente introduction au rapport entre la Théorie des Ordonnés finis et les problèmes d'ordonnancement. Toutes ces notions y sont développées en vue des applications.

Le calcul de $s(P)$, $L(P)$ et $\dim(P)$ a donné lieu à de nombreux travaux (*voir* [2] pour un survey). $s(P)$ et $L(P)$ s'obtiennent facilement à partir de $H(P)$. En effet, définissons sur les sommets de $H(P)$ la fonction f:

- $f(X_0) = 1$;
- $f(X) = \Sigma f(Y)$, $Y \in \Gamma - (X)$.

Il est clair que $f(X)$ est le nombre d'extensions linéaires de $(X, \leqq)$. En conséquence, $L(P)$ est donné par la valeur de f au sommet X_P et est calculable à partir de $H(P)$ en une complexité $O(N(P)\, w(P))$.

De même, définissons algorithmiquement sur $H(P)$ la fonction g:

Pour tout sommet X *faire*
  Si $X \in \{X_0\} \cup \Gamma^+(X_0)$ *alors* $g(X) \leftarrow 0$ *sinon* $g(X) \leftarrow \infty$;
Pour tout sommet $X \neq X_0$ (pris par rang non décroissant) *faire*
Pour tout couple (Y, Z), $Y \in \Gamma^-(X)$, $Z \in \Gamma^+(X)$ *faire*
  $g(Z) \leftarrow \min(g(Z), g(X) + \alpha(e(YX), e(XZ))$;

Si $e(YX) = i$ et $e(XZ) = j$, $\alpha(i,j) = 0$ si $j \in \gamma^+(i)$, $\alpha(i,j) = 1$ sinon.

$g(Z)$ n'est autre que le nombre minimal de sauts d'une extension linéaire de $(\mathscr{P}(Z), \leqq)$. Cette propriété s'établit aisément par récurrence sur le rang (longueur d'un chemin joignant X_0 à Z) de Z dans $H(P)$. En effet, toutes les extensions linéaires de $(\mathscr{P}(Z), \leqq)$ s'obtiennent en considérant les extensions linéaires de $(\mathscr{P}(X), \leqq)$, $X \in \Gamma^-(Z)$, et le nombre minimal de sauts s'en déduit en utilisant la fonction α ci-dessus.

$s(P)$ est donc donné par la valeur de g au sommet X_P et est calculable en une complexité $O(N(P)w(P)^2)$ en utilisant un parcours en largeur de $H(P)$. Sur la figure 1, $g(Z)$ apparaît en encadré pour chaque sommet Z de $H(P)$.

Conséquence

La complexité de ces algorithmes est fortement reliée à la valeur de $w(P)$. En effet, s'il existe une constante k telle que $w(P) \leqq k$, comme d'après le théorème de Dilworth, P se décompose en k ordres totaux $C_1, C_2, \ldots, C_k$, $|C_i| \leqq n$, le nombre d'antichaînes de P est majoré par $(n+1)^k$. On obtient des algorithmes en $O(n^k)$. Si $w(P) \in O(\log n)$, $N(P) \in O(n^{\log n})$ et on obtient des algorithmes sous-exponentiels. Ces constatations nous conduisent à remarquer deux types d'ordre particuliers :

Ordres de largeur bornée (W_k)

k est une constante fixée, $k \geqq 2$, $P \in W_k$ si $w(P) \leqq k$.

Les résultats précédents produisent des algorithmes, vus sous l'angle du calcul d'une fonction en chaque sommet du graphe de Hasse d'un treillis distributif, de complexité $O(n^k)$ pour le calcul de $L(P)$ et $s(P)$. D'autres algorithmes, de style programmation dynamique, donnent une complexité $O(n^{k+1})$ pour le calcul de $L(P)$ [17] et $O(n^k)$ pour le calcul de $s(P)$, [3].

Ces résultats sont exploitables sur une classe plus générale :

Ordres décomposables en ordres de largeur bornée (SW_k)

Ces ordres sont très bien décrits dans l'ouvrage de G. Steiner [17], comme une classe contenant les ordres série-parallèles, ainsi que des ordres de largeur arbitraire.

Soient $Q, P_1, P_2, \ldots, P_k$ des ordonnés disjoints, et $a_1, \ldots, a_k \in Q$. On note par

$$S = Q[a_1, \ldots, a_k; P_1, \ldots, P_k], \qquad 1 < |P_i| < |S| \text{ pour au moins un } i$$

l'ensemble ordonné résultant de la *substitution* de chaque a_i par P_i. Plus formellement

$$a \leqq_S b \iff \exists i \quad \text{avec} \quad a, b \in P_i \quad \text{et} \quad a \leqq_{P_i} b$$

ou

$$\exists i \neq j \quad \text{avec} \quad a \in P_i, \quad b \in P_j \quad \text{et} \quad a_i <_Q a_j.$$

Exemple. – La figure 3 [9] montre un ensemble S généré par substitution à partir de quatre ensembles Q, P_1, P_2, P_3.

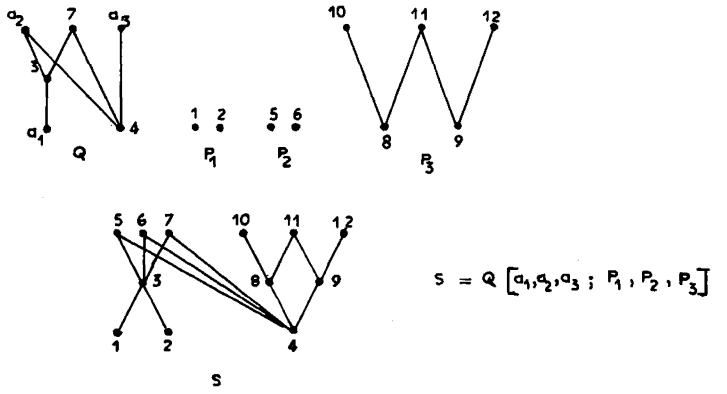

Figure 3.

Un ordre P est *décomposable* s'il peut être obtenu par une telle substitution. Si P est décomposable, un des trois cas suivants (mutuellement exclusifs) s'applique :

– Q est une antichaîne, et $L(P) = \prod_{1 \leq i \leq k} L(P_i)$ (décomposition en *Parallèle*);

– Q est une chaîne et $L(P) = \prod_{1 \leq i \leq k} L(P_i) \cdot \frac{(n_1 + \ldots + n_k)!}{n_1! \ldots n_k!}$ (décomposition en *Série*);

– Q est indécomposable et

$$L(P) = L(Q[a_1, \ldots, a_k; t_1, \ldots, t_k]) \cdot \prod_{1 \leq i \leq k} L(P_i)$$

où t_i est une extension linéaire arbitraire de P_i

On associe alors à P un *arbre canonique de décomposition* dont la racine représente P et les feuilles les éléments de P. Les fils d'un nœud intérieur correspondent aux ensembles P_i dans l'un des trois cas précédents.

La figure suivante [9] montre l'arbre de décomposition de l'ordre donné dans la figure 3. Les lettres P (parallèle), S (série), et N (non décomposable) sont associées aux trois types de décomposition. L'ordre n'est pas série-parallèle dans la mesure où apparaissent des nœuds de type N.

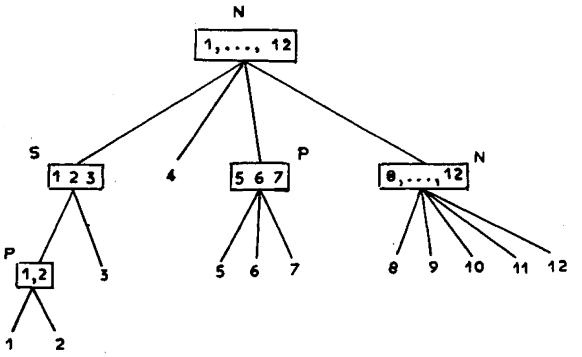

Figure 4.

$P \in SW_k$ si P peut êre construit par une succession de substitutions telles que *chaque ensemble Q appartient à W_k*.

En appliquant récursivement les formules donnant $L(P)$ dans l'arbre de décomposition de P, G. Steiner produit un algorithme de complexité $O(n^{k+2})$, utilisant le fait que chaque ensemble $Q[a_1, \ldots, a_k; t_1, \ldots, t_k]$ appartient à W_k. Les méthodes exposées ci-dessus permettent d'améliorer cette complexité en $O(n^{k+1})$.

Par analogie, on montre que $s(P)$ peut être calculé en $O(n^{2k+1})$ sur la classe SW_k [3]. Il est à noter que $\dim(P)$ est polynomialement calculable sur W_k si et seulement si la même propriété est vraie sur SW_k. Cette question reste ouverte.

BIBLIOGRAPHIE

1. J. P. Bordat, Efficient Polynomial Algorithms for Distributive Lattices, accepté pour publication par *Discrete Appl. Math.*
2. V. Bouchitte et M. Habib, The Calculation of Invariants for Ordered Sets, *Algorithms and Order*, I. Rival éd., Kluwer Acad. Publ., Dordrecht, 1989, p. 231-279.
3. C. J. Colbourn et W. R. Pulleyblank, Minimizing Setups in Ordered Sets with Fixed Width, *Order*, 1985, *1*, p. 225-229.
4. R. P. Dilworth, A Decomposition Theorem for Partially Ordered Sets, *Ann. of Math.*, 1950, *51*, p. 161-166.
5. G. Gratzer, General lattice Theory, Academic Press, 1978.

6. E. L. LAWLER, Efficient Implementation of Dynamic Programming Algorithms for Sequencing Problems, Rep. BW106/79, *Stichting Matematisch Centrum*, Amsterdam, 1979.
7. E. L. LAWLER, J. K. LENSTRA et A. H. G. RINNOOY KHAN, Recent Developments in Deterministic Sequencing and Scheduling: A Survey, M. A. H. DEMPSTER *et al.*, éd., *Deterministic and Stochastic Scheduling*, Reidel, Dordrecht, 1982, p. 35-73.
8. R. H. MOHRING, Scheduling Problems with a Singular Solution, *Discrete Appl. Math.*, 1982, *16*, p. 225-239.
9. R. H. MOHRING, Computationally Tractable Classes of Ordered Sets, *Algorithms and Order*, I. RIVAL éd., Kluwer Acad. Publ., Dordrecht, 1989, p. 105-113.
10. G. L. NEMHAUSER et L. E. TROTTER, Vertex Packings: Structural Properties and Algorithms, *Math. Progr.*, 1975, *8*, p. 232-248.
11. J. C. PICARD et M. QUEYRANNE, Structure of All Minimum Cuts in a Network and Applications, *Math. Progr. Study*, 1980, *13*, p. 8-16.
12. W. POGUNTKE, Order-Theoretic Aspects of Scheduling, *Combinatorics and Ordered sets* (Arcata, Calif.), 1985, p. 1-32, Contemp. Math., *57*, Amer. Math. Soc., Providence, R. I., 1986.
13. J. S. PROVAN et M. O. BALL, The Complexity of Counting Cuts and of Computing the Probability that a Graph is Connected, *SIAM J. Comput.*, 1983, *12*, p. 777-788.
14. L. SCHRAGE et K. R. BAKER, Dynamic Programming Solution for Sequencing Problems with Precedence Constraints. *Oper. Res.*, 1978, *26*, p. 444-449.
15. G. STEINER, Single Machine Scheduling with Precedence Constraints of Dimension 2, *Math. Oper. Res.*, 1984, *9*, p. 248-259.
16. G. STEINER, An Algorithm to Generate the Ideals of a Partial Order, *Oper. Res. Letters*, 1986, *5*, p. 317-320.
17. G. STEINER, On Computing the Information Theoretic Bound for Sorting: Counting the Linear Extensions of Posets; Res. Report n° 87459-OR, *McMaster University*, Hamilton, Ontario, Canada, 1987.