Le nombre de posets à isomorphie près ayant 12 éléments

Claude Chaunier et Nik Lygeros

Département de Mathématiques, Université Lyon-I, 43 boulevard du 11 novembre, 69622 Villeurbanne Cedex, France

A Jean-Pierre Serre, notre premier mentor

Abstract

Chaunier C. and N. Lygeros, Le nombre de posets à isomorphie près ayant 12 éléments. Theoretical Computer Science, 123 (1994) 89-94.
We describe what algorithm has enabled to compute the number of nonisomorphic posets having 12 elements P_{12}=1104981746 [1], which currently constitute the greatest known value of this enumeration.

Dans cet article qui fait suite aux travaux exposés dans [7], nous ne considérons que des posets non isomorphes 2 à 2. On note P_n le nombre de posets à n éléments. Le calcul exact de P_n est un problème très difficile et donc actuellement ouvert (voir [14]), Ainsi l'utilisation des ordinateurs est rendue nécessaire. Les recherches dans ce domaine ont commencé il y a une vingtaine d'années par l'obtention à la "main" de la première valeur non triviale P_7 par Wright [16]. Le premier calcul sur ordinateur, effectué par Das [5], donne P_8. Pour calculer P_9, Möhring [10] se base sur l'identification des graphes de comparabilité en exploitant les données de Read et Wormald [13] sur les graphes. Enfin pour parvenir à P_{10} et P_{11}, Culberson et Rawlins [4] élaborent le premier algorithme spécifique aux posets en s'appuyant sur la structure du poset des posets à n éléments. Quant à notre algorithme [2], dont les étapes principales ont été décrites dans [2], pour obtenir P_{12} il est (entre autres) fondé sur une idée qui tire son origine heuristique de la théorie des fractals [9]. On a donc :

Tableau 1

$0 \leqslant n \leqslant 6$, $P_n = 1; 1; 2; 5; 16; 63; 318;$	
$P_7 = 2\,045$	Wright (1972)
$P_8 = 16\,999$	Das (1977)
$P_9 = 183\,231$	Möhring (1984)
$P_{10} = 2\,567\,284$	Culberson et Rawlins (1990)
$P_{11} = 46\,749\,427$	Culberson et Rawlins (1990)
$P_{12} = 1\,104\,891\,746$	(1991)

Structure de l'algorithme. On représente un poset $P = (\{x_1, \ldots, x_n\}, <)$ par sa matrice d'incidence $(a_{ij}) \in \{0,1\}^{n^2}$ si et seulement si $x_i < x_j$.

Ce choix de représentation est fait pour l'avantage informatique suivant : la manipulation des tableaux de données est prévue électroniquement dans les ordinateurs actuels, c'est la structure interne même de la mémoire adoptée par les concepteurs. Une représentation arborescente serait-elle préférable sur des machines parallèles ?

1. Annonce envoyée aux Abstracts of the A.M.S le 3 octobre 1991.
2. Le second auteur a été financé par le PRC Math, et Info, et le CNRS

Mais une matrice a le désavantage mathématique suivant : alors que la notation ensembliste, et dans une moindre mesure la représentation graphique, ne se préoccupent pas d'ordonner linéairement les sommets du poset, l'indiçage dans une matrice force la considération d'un tel ordre.

Tout l'algorithme est alors fondé sur une restriction des permutations parmi toutes les permutations possibles des sommets afin de retrouver l'unicité de représentation. L'algorithme privilégie un premier ordre qui réduit considérablement l'ensemble des permutations de sommets, autrement dit des permutations simultanées $(a_{\sigma(i)\sigma(j)})$ des lignes et colonnes de (a_{ij}) en vérifiant la

Condition 1. La suite $(d_i)_{1\leq i\leq n}$ est décroissante, où $d_i = \sum_{j=1}^{n} a_{ij}$

(d_i est aussi le demi-degré extérieur de x_i). Cette condition et la transitivité du poset impliquent une proposition qu'il faut sans doute rapprocher du lemme d'anticircularité de Birkhoff [1].

Proposition 1. (a_{ij}) est plus que triangulaire supérieure, i.e. $\forall k, l/d_k = d_{k+l-1}$ les $l(l+1)/2$ coefficients $a_{ij}, k \leq i \leq j < k+l$ sur et au-dessus de la diagonale, sont nuls.

Nous avons donc un ensemble de permutations autorisées intermédiaire entre les $n!$ éléments du groupe symétrique S_n, qui perturbent la matrice, et les éléments du groupe d'automorphismes du poset, qui se caractérisent par l'invariance de la matrice sous leur action. Peut-on estimer le gain apporté, en déterminant le comportement asymptotique du nombre de ces permutations intermédiaires sur les posets à n éléments ?

La condition suivante permet d'identifier les permutations aux seuls automorphismes, suivant une méthode introduite par Read [12].

Condition 2. $(a_{1,n}, a_{1,n-1}, \ldots, a_{1,1}, a_{2,n}, \ldots, a_{n,1})$ est maximal pour l'ordre lexico-graphique de $\{0,1\}^{n^2}$ en attribuant le plus grand poids à $a_{1,n}$.

La représentation qui réalise le maximum est dite canonique. Sans la Condition 1, cette condition redonne aussi au poset sa représentation unique, mais aussitôt l'avantage mathématique retrouvé, la partie informatique se complique. En effet, dans ce travail préliminaire, il n'est pas demandé à l'ordinateur de manipuler des représentations canoniques, mais avant tout de les générer et de les reconnaître : il lui faut alors prendre en compte toutes les permutations autorisées avant la Condition 2 pour savoir si le maximum est réalisé. Le problème n'existerait pas si l'on savait réaliser et tester la Condition 2 rapidement. Or ce test est évidemment lié au test d'isomorphie de graphes, dont on ignore toujours la complexité minimale, même en se restreignant aux posets [8]. A l'autre extrémité la Condition 1 est insuffisante mais facile à mettre en oeuvre, elle est d'ailleurs l'origine classique de recherches d'invariants de plus en plus efficaces pour approcher le test d'isomorphie tout en restant programmables en temps polynomial. On voit donc l'intérêt premier de la restriction préalablement imposée par la Condition I : elle serait utilisée dans une partie du programme, avant un éventuel recours à la Condition 2, pour tester rapidement si les posets générés sous forme de matrice par une autre partie sont canoniques et doivent être comptabilisés.

Elle a des conséquences plus intéressantes : la partie du programme qui génère les posets peut en faire un usage constant (voir la simplicité de la Proposition 1), c'est-à-dire que la condition peut être entièrement déplacée de la partie test du programme à la partie génération, évitant des générations superflues. Surtout, elle donne à la Condition 2 des caractéristiques internes dont certaines peuvent encore infiltrer la partie génération.

Cela révèle un lien dont il faudra tenir compte si l'on désire améliorer encore l'efficacité de l'algorithme, notion - efficacement définie dans [15] comme un équilibre judicieux entre la simplicité et la puissance : chercher à remplacer la première condition, ou bien à introduire une troisième condition entre la première, simple mais insuffisante, et la deuxième, complexe mais puissante, se traduit par l'étude d'un équilibre entre le caractère effectif de leur programmation et la richesse de leurs implications mathématiques.

Comme caractéristique plus riche que la Proposition 1, les deux conditions entraînent la proposition qui doit sa génèse à une notion appartenant à la théorie des fractals, l'autosimilarité :

Proposition 2. Pour tout i, soit A_i la partition des indices $\{1, 2, \ldots, n\}$ réunissant j et j' dans une même classe si et seulement si $d_j = d_{j'}$ et $\forall k < i, a_{kj} = a_{kj'}$. Alors $\forall j < j'$ dans une même classe de A_i, on a $a_{ij} \leq a_{ij'}$.

L'on a aussi :

Proposition 3. Pour tester la Condition 2, on peut se limiter à ne permuter l'indice i qu'avec les indices de sa classe dans $A_i, \forall i$.

dont l'intérêt est dû au fait que souvent les classes considérées ne contiennent qu'un sommet (dans quelle proportion asymptotique ?), à rapprocher de la rigidité de presque tous les posets [11]. Pour réduire les autres, à commencer par l'antichaîne, on utilise :

Proposition 3 bis. Il est inutile de permuter l'indice i avec les indices k de sa classe dans A_i qui ont des lignes identiques $(a_{ij})_{1 \leq j \leq n} = (a_{kj})_{1 \leq j \leq n}$.

Une autre proposition élémentaire permettrait-elle d'éviter également de permuter entre eux les n premiers sommets de la couronne à $2n$ sommets ?

Les deux conditions impliquent aussi une proposition énoncée à notre connaissance pour la première fois par Colbourn [3], qui infiltre de la même façon que les Propositions 1 et 2 la génération des posets :

Proposition 4. $\forall i < k$ dans la même classe de A_i, leur ligne vérifie pour l'ordre lexicographique :

$$(a_{i,n}, a_{i,n-1}, \ldots, a_{i,1}) \geq (a_{k,n}, a_{k,n-1}, \ldots, a_{k,1})$$

La partie du programme qui génère les posets consiste donc à faire varier la suite (d_i), et pour chaque suite, à générer de manière exhaustive les matrices transitives ayant cette suite pour demi-degrés et vérifiant la Condition 1 et les Propositions 1.2 et 4. Cela est fait ligne par ligne, en commençant par la première ligne de la matrice, en descendant lorsque les conditions sont satisfaites, ou en remontant par chaînage arrière lorsque toutes les possibilités ont été tentées pour une ligne. A chaque ligne les calculs qui concernent le test de canonicité mais ne dépendent pas des lignes suivantes sont effectués, pour éviter les calculs redondants. Chaque fois que la dernière ligne est atteinte, la Condition 2 est testée sur la matrice candidate, et l'on compte un poset de plus lorsque celle-ci s'avère sous forme canonique.

Le test de canonicité, à commencer par $i = 1$, considère les permutations σ_i agissant sur la classe de i de la partition A_i et ne transgressant pas les propositions précédentes. Pour chacune d'entre elles :

-Il essaie de permuter tous les indices des classes suivantes de telle sorte que la Proposition 2 soit encore vérifiée après l'application de σ_i, : ceci nous fournit une permutation des indices σ'_i des indices $j \geq i$. -Il compare après application de σ'_i la matrice avec la matrice d'origine :
• si le vecteur intervenant dans la Condition 2 dépasse le vecteur de la matrice d'origine, c'est que cette dernière notait pas canonique, et elle n'est à compter que parmi le surplus.
• si le vecteur est strictement inférieur au vecteur d'origine, il est inutile de prendre en compte la permutation σ_i, car elle ne permettra jamais de dépasser le vecteur d'origine. On passe donc à une autre σ_i.
• si par contre le vecteur est inchangé, alors la matrice est inchangée σ'_i, est la restriction d'un automorphisme du poset, et l'on peut passer au i suivant, indice d'un sommet de la classe qui suit dans A_i celle de i.
Chaque fois que l'on arrive à $i > n$ la composition $\sigma'_1 \circ \sigma'_1 \circ \ldots$ des permutations partielles définit un automorphisme, on peut donc compter ainsi les automorphismes d'un poset canonique. Le programme a utilisé simultanément 12 machines différentes en tâche de fond avec une priorité minimale (8 stations de travail Apollo,

dont une 400, une 3500, et six 2500, et 4 compatibles IBM PC. dont un 486 à 25 Mhz et les autres 386 ST à 16 Mhz), ce qui rend difficile l'estimation exacte du temps de calcul. On peut donner cependant l'approximation de 14 jours sur un 486 à 25 Mhz.

Il nécessite asymptotiquement n^3 bits ($O(n^2)$ si l'on fait une concession importante à la vitesse). Résultats. On obtient le Tableau 2, énumérant les posets à 12 éléments suivant le nombre r de relations.

Tableau 2

r	P^r_{12}	r	P^r_{12}	r	P^r_{12}
0	1				
1	1	23	25 468 042	45	6 370 240
2	3	24	33 157 695	46	4 491 015
3	7	25	41 495 336	47	3 100 063
4	19	26	50 008 606	48	2 094 942
5	47	27	58 130 096	49	1 386 092
6	133	28	65 270 723	50	897 535
7	352	29	70 888 253	51	568 627
8	997	30	74 562 234	52	352 196
9	2 753	31	76 042 383	53	213 115
10	7 558	32	75 275 671	54	125 818
11	19 801	33	72 402 491	55	72 382
12	49 795	34	67 726 046	56	40 515
13	117 875	35	61 666 534	57	21 985
14	263 019	36	54 699 028	58	11 545
15	550 013	37	47 302 979	59	5 808
16	1 080 422	38	39 908 316	60	2 779
17	1 993 865	39	32 869 931	61	1 249
18	3 469 819	40	26 443 898	62	509
19	5 707 944	41	20 792 175	63	184
20	8 909 624	42	15 984 309	64	55
21	13 234 277	43	12 020 498	65	11
22	18 766 663	44	8 844 848	66	1

Remarques. Les valeurs pour $0 \leq r \leq 7$ sont en accord avec celles obtenues par Culberson et Rawlins.

(ii) Les valeurs pour $55 \leq r \leq 66$ sont confirmées par les formules de Erné [6].

Références

[1] G Birkhoff, Lattice theory. *A.M.S. colloque publ.* vol. 25. 3ème édition (1967).
[2] C. Chaunier et N. Lygeros, Progrès dans I énumération des posets. *C. R. Acad. Sci. Paris, série I* 314 (1992) 691-694.
[3] C J. Colbourn. Graph generation. Research Report CS-77-37. University of Waterloo. 1977. p. 100.
[4] J.C Culberson et J.E. Rawlins. New results from an algorithm for counting posets, *Order 7* (1991) 361-374.
[5] S.K. Das. A machine representation of finite T_0 topologies. *J. Assoc. Comput. Mach.* 24 (1977) 676-692.
[6] M. Erné, The number of partially ordered sets with more points that incomparable pairs, preprint pour *Discrete Math.*
[7] R. Fraïssé et N. Lygeros, Petits posets : dénombrement, représentabilité par cercles et 'compenseurs'. *C. R. Acad. Sci. Paris, série 1* 313 (1991) 417-420.
[8] D.S. Johnson. The NP-completeness column : an ongoing guide, *J. Algorithms* 6 (1985) 434-451.
[9] B. Mandelbrot. Les objets fractals, *Flammarion*, 1ère édn. (1975).
[10] R.H. Möhring. Algorithmic aspects of comparability graphs and interval graphs. in : *Graph and Order*, I. Rival, éd. (Reidel, Dordrecht, 1985) 41-101.
[11] H.J. Prömel, Counting unlabeled structures. *J. Combin. Theory.* Series A 44 (1987) 83 93.
[12] R.C. Read. Every one a winner, *Ann. Discrete Math. 2* (1978) 107-120.

[13] R.C. Read et N.C. Wormald, Catalogues of graphs and digraphs (announcement). *Discrete Math.* 31 (1980) 224.
[14] R.P., Stanley. *Enumerative Combinatorics, Vol. I* (Wadsworth and Brooks, California, 1986).
[15] J.G. Wolff, Simplicity and Power - Some unifying ideas in computing. *Comput.* J. 33 (1990) 518-534.
[16] J. Wright, Cycle indicators of certain classes of types of quasi-orders or topologies. PhD. Dissertation. University of Rochester. 1972.