JOURNAL OF THE EGYPTIAN MATHEMATICAL SOCIETY
(J. Egypt. Math. Soc.)Vol.14(1)(2006)pp5-13

Recognition of prime posets and one of its applications

S. M. Khamis

The paper contains a recognition algorithmic method for deciding whether a given poset, P, is prime. The algorithm is designed to determine whether there exists a proper P-autonomous set including specified two distinct elements of P. The steps are repeated for all ordered pairs of distinct elements of P, if it is prime, in a polynomial time.

As an application of this test, the author counted the number of unlabeled prime posets regarding heights up to 13 elements. The height counting algorithm depends on: (1) Create the colexecographic list of all strictly upper triangular matrices satisfying the bucket from; (2) Apply the given test to choose those corresponding to prime posets; and (3) Enumerate the required number regarding height by summing the weights of accepted matrices, that are also computed in the algorithm. The algorithm added to height counting field, the original numbers of unlabeled 13-element prime posets of height $k, k \leq 13$, and verified the previously known numbers of unlabeled n-element prime posets of height $k, 1 \leq k \leq n \leq 12$, that have been introduced in [11].

1 Introduction

In [6], one can find some important information about enumeration problems of different classes of posets and their comparability graphs. The enumeration of various classes of posets is an interesting combinatorial problem for which several techniques have been developed. All previous studies in the field of counting different types of posets treated the problem for getting exact or estimate numbers only according to the number of elements within the poset. Numerous results ranging from exact and algorithmic counting to asymptotic estimates appear in the literature. Examples for the exact counting regardless of height include: labeled and unlabeled series-parallel posets in [18], labeled and unlabeled interval graphs [8], interval orders [6]. The author et.al. gave an exact count for unlabeled 2-dimensional posets and permutation graphs, respectively, in [2] and [10], labeled graded posets [13], and unlabeled graded posets [14].

However, for most classes of posets, an exact counting cannot be successfully found. To treat such classes, researchers can use either asymptotic or algorithmic enumeration. Some example of obtaining asymptotic estimates or upper and lower bounds for required numbers are: In [15], an asymptotic estimate for the number of labeled n-element posets has been obtained. Also, lower and upper bounds for labeled and unlabeled n-element N-free posets are given in [1].

In the algorithmic counting area, there exist several algorithms, which have been constructed to generate all posets (of a small number of elements) and count them. For counting unlabeled posets see: Mohring in [17] recorded results for $n \leq 10$. The algorithm that was developed by Culberson and Rawlins [5] gave the numbers of all unlabeled posets with $n \leq 11$ elements, that agree with Mohring's results, [17], except for $n = 10$. In [4], Claunier and Lygerõs counted the number of unlabeled posets for

[0]Department of Mathematics, Faculty of Science , Ain Shams University, Cairo, Egypt

$n = 12, 13$. In the case $n = 13$, they used nine Apollo wokstations for six months each workstation generate 250 posets per second on average. In [9], Heitzig and Reinhold gave a new orderly algorithm to count the number of unlabeled posets on up to 14 elements. This algorithm generates about 3000 objects per second (on a DEC Alpha, 450 Mhz), a single processor would still run for more than 17 months for $n = 14$. So they worked on 50 processors simultaneously to record the number.

A related interesting problem is to count classes of posets according to the height of a poset. Some studies in the height counting field began at year 2000. The first paper, [7], introduced a general technique for height counting of any class of posets closed with respect to series and parallel compositions provided that the height counting of irreducible posets in the considered class is known. According to this technique, the height counting problem of series-parallel posets is exactly solved in terms of generating functions, [7]. Then the numbers of prime and uniquely partially orderable posets of n elements and height k are algorithmically counted for $1 \leq k \leq n \leq 12$, [11]. Also, the height counting problems of general posets has been treated using the height counting technique depending on the number of $(+, \oplus)$-irreducible posets according to height are determined in [11] by applying an algorithmic method. In year 2004, [12], the hight counting of interval orders, and interval N-free posets are enumerated in terms of generating functions. In the same paper, an algorithmic method for height counting of connected N-free posets together with the principle of height counting technique have been employed to enumerate the numbers of N-free posets on n elements and height $k, 1 \leq k \leq n \leq 14$.

In this paper, the author describes a new polynomial time algorithm to decide whether a given poset is prime. This test is employed to count the exact number, P_{nk}, of unlabeled prime posets on n elements and k heights. Section 2 introduces the basic concepts and the underlying definitions. Section 3 contains the description of the recognition method for testing prime. The technique of counting P_{nk} is given in Section 4. Finally the appendix contains the values of P_{nk} for $1 \leq k \leq n \leq 13$.

2 Basic definitions

The terminology, notation and concepts will follow that in [6] and [17].

A *partially ordered set* (poset) P is defined by $(V, <_p)$, where V is a nonempty set of elements and $<_p$, or $<$ if P is understood, is a partial order relation on V, that is, an asymmetric and transitive relation. Two elements $u, v \in V$ are said to *comparable* $(u \sim v)$ in P if $u < v$ or $v < u$, otherwise they are *incomparable* $(u \| v)$. We will say v covers $u(u < v)$ if $u < v$ and there is no $w \in V$ such that $u < w < v$. A set of pairwise *incomparable* (resp. *comparable*) elements is called an *antichain* (resp. a *chain*). The *height* of an element $u \in P$, denoted by $h(u)$, is the maximum cardinality of a chain in P having u as its maximum element. The *height* of P is defined as $h(P) = \max\{h(u) : u \in P\}$.

Let $P = (V, <)$ be a poset and $U \subseteq V$. Then U is valled P-*autonomous* if $\forall\, u, v \in U$ and $\forall\, w \in V \backslash U$, we have: $v < w \Leftrightarrow u < w$, or $w < c \Leftrightarrow w < u$. U is proper iff $1 < |U| < |V|$. The proper autonomous sets of any poset arise through using the following operation which is known as substitution decomposition, modular decomposition, or lexicographic decomposition.

Let $P' = (V', <')$ be a poset with element set $V' = \{v_1, \ldots, v_m\}$ and let $P_1 = (V_1, <_1), \ldots, P_m = (V_m, <_m)$ be posets with disjoint sets of element. The *composition* $P'[P_1, \ldots, P_m]$ is the poset $P = (V, <)$ where $V = V_1 \cup \ldots \cup V_m$ and $u < v$ iff either one of the following is satisfied:

(A) For some $i; u, v \in V_i$ and $u <_i v$, or

(B) $u \in V_i$ and $v \in V_j$ and $v_i <' v_j$.
Thus P is obtained by substituting each element v_i of P' by P_i and making each element of P_i comparable to each element of P_j whenever v_i is comparable to v_j in P'.

The posets $P_1, \ldots, P_m$ are called the *inner posets* and their sets of elements are P-autonomous sets while P' is the outer poset or the quotient poset. The poset P is called *decomposable* if $P = P'[P_1, \ldots, P_m]$ where $m > 1$ and at least one of the $P_i's$ is nontrivial, i.e. has more than one element. An *indecomposable* poset is called *prime.* This means that a poset P is prime if and only if P has no proper autonomous set. This property of prime posets is employed to recognize such posets.

The operation of substitution plays an essential role in solving many combinatorial optimization problems on posets. The main idea of such divide-and-conquer techniques is to solve the problem for each of $P', P_1, \ldots, P_m$ and combine these solutions to get a solution for $P = P'[P_1, \ldots, P_m]$. To name a few of such optimization problems we mention: minimum covering by chains / antichains, determining the dimension and the Möbious function and counting partial orders (see [17]).

The imortance of the substitution and the inverse operations: decomposition of posets into prime ones mandates the search for algorithmic methods to recognize and count prime posets. Algorithmic methods for recognizing prime posets and for counting the numbere of prime posets are two main problems, which we tackle here.

3 A recognition of prime posets

This section contains the description of a polynomial time algorithm, for testing whether or not the given poset P is prime. Recall that a poset is prime iff it has no proper autonomous set. Thus the basic task of the suggested method is: for any two distinct elements $l, k \in V$, the algorithm finds a proper autonomous set that contains l, k. If a such set exists, then the given poset is not prime. Otherwise, if each pair of elements of P does not belong to a proper autonomous set, then P is prime. This method is described in the next procedure.

Data structure needed:

(i) $A_1, \ldots, A_n$ are n lists, where A_i consists of all elements comparable to element i in P.

(ii) L is a list of integers.

(ii) Include is a 1-dimensional array of integers whose entries have value one if corresponding elements belong to a possible current autonomous set and zero otherwise.

(iv) Counter is an integer variable used to count the number of elements that belong to a possible autonomous set corresponding to the current pair of elements of P.

(v) Autonomous is a boolean variable having the value true if a collection of elements w.r.t. the given pair of elements of P is an autonomous set, false otherwise,

(vi) i, j, k, l are integer variables for the elements of P.

(vii) TestSet is a set of integers.

Procedure Exist Autonomous Set (l, k**; Autonomous**)**;**

```
Begin
1)    For $i \leftarrow 1$ to $n$ Do Include [$i$] $\leftarrow$ 0; od;
2)    Autonomous $\leftarrow$ False;   Include [$l$] $\leftarrow$ 1;   Include [$k$] $\leftarrow$ 1;
3)    $L \leftarrow \{k\}$;      Counter $\leftarrow$ 2;
4)    While $L \neq \emptyset$ Do
5)       $j \leftarrow$ Min [$l$];      // Get a minimum element of $L$.
6)       DELETEMIN ($L$);   // Delete the chosen element from $L$.
```

```
7)          TestSet $\leftarrow (A_l - A_j) \cup (A_j - A_l)$;
8)          If there existsd $i \in$ TestSet  s.t. $1 \le i < l \vee l < i < k$ Then return fi;
9)          For $i \leftarrow k+1$ to $n$ Do
10)             If $i \in$ TestSet Then
11)                 If Include $[i] = 0$ Then
12)                     Include $[i] \leftarrow 1$;
13)                     Counter $\leftarrow$ Counter $+1$;
14)                     Insert $i$ in the first position of $L$;
15)                 fi;
16)             fi;
17)           od;
18)           If Counter $= n - k + 2$ Then return fi;
19)       od;
20)       Autonomous $\leftarrow$ True;
    End;
```

The primality test checks whether a given poset P is prime or not via repeating the procedure ExistAutonomousSet for pairs of distinct elements of P. All required paris (l, k) where $1 < k \le n$ must satisfy either $l \| k$ or $l < k$ and the sets of predecessor elements of l and k whose labeled are less than l, are the same. This process is halted when a proper autonomous set including the given pair is found. If there exists no such set, the test decides that the current poset is prime.

Obviously, the complexity status of the primality test depends on the status of the procedure ExistAutonomousSet multiplying by the number of used pairs. In ExistAutonomousSet, the While cycle from step (4) to step (19) does not repeat more than $n-k$ times for any k, where $1 \le l < k < n$. During each of these iterations, the comparison in step (5) can be done in at most $n-k$ steps. Also the For-loop (step (9)- step (17)) takes in the worst case $O(n-k)$. Therefore the complexity of the procedure is at most $O((n-k)^2)$. Hence the total running time of the primality test in the worst case is given by $\sum_{l=1}^{n-1} \sum_{k=l+1}^{n} O(n-k)^2 \approx O(n^4)$. We remark that a code of the procedure runs much faster than is expected. This is because the number of used pairs is less than $\frac{1}{2}n(n-1)$ and also we don't need all of them if a poset is not prime. Moreover, if a poset is prime, the while-De loop (steps (4)-(19)) is not completed. So, this leads to count the numbers of unlabeled n-element prime posets on $n \le 13$ as shown in the following.

4 An application in the counting field

In this Section, the height counting problem of unlabeled prime posets will be treated. Unfortunately, this type of posets is difficult to be treated mathematically. Thus, it will be manipulated using an algorithmic approach. The given algorithm depends on the new primality test (see 3). It used to count the numbers, P_{nk}, of unlabeled n-element prime posets of height $k, k \le n$. The algorithm is based on the technique that is introduced in [3]. That technique counted unlabeled posets through constructing all strictly upper triangular matrics with some special properties in a certain order. These are defined as follows:

Suppose P is a poset on the set $\{v_1, v_2, \ldots, v_n\}$ which can be represented by an adjacency matrix $A = [a_{ij}]$, where $a_{ij} = 1$ iff $v_i < v_j$ in P and 0 otherwise. P is restricted to be a natural order i.e., if $v_i < v_j$ in P then $i < j$ in N. A natural labeled poset P on n elements can be represented by a square matrix of order n whose elements on the main diagonal and lower triangular part are zeroes.

Apart from isomorphism, it is clear that there exists a one-to-many relation between unlabeled posets and the above defined matrices. To improve this relation, only the matrices whose rows appear in the bucket sort according to rows sums are considered, (see Fig.1).

This class of matrices is sufficient to count the numbers of specified posets. To construct all these matrices, we follow a *colexecographic* (colex.) order w.r.t. the rows of matrices. This ordering is defined as follows:

Let A and B be two distinct matrices of order n. Then A is before B in colex. order w.r.t. the rows of matrices, if for some k the last $n-k$ rows of A and B are the same and the kth row of A precedes the kth row of B in *lexicographic* (lex.) order. Thus, the colex list of matrices begins with the zero matrix which represents the antichain of n elements. While it ends with the matrix whose elements in the upper traingular are ones. Clearly, this last matrix corresponds to a chain of n elements.

Now, the description of the suggested algorithm for computing the required numbers P_{nk} is given in the following steps, (see [3] and [11] for details):

(1) Create a strictly upper triangular matrix, A, satisfying the bucket form. This can be done by determining the first row of A having at least one zero entry at the right of its main diagonal. If there is no such row the algorithm will stop. Otherwise, the algorithm gets the successor of this row in the lex. order and hence adjust the entries of this row to satisfy the transitivity constraint. Then the rows of A above the current one are updated without destroying the bucket sort w.r.t. rows sums.

(2) Check whether for the created matrix A, the sum of elements of the ith column is greater than the sum of elements of $(i+1)st$ whenever the number of ones of the corresponding rows are equal, for any $i \in \{1, \ldots, n-1\}$. If the condition is satisfied, A will be ignored and the algorithm creates the next matrix. Otherwise A is accepted to perform the next test. Note that this constraint is necessary to discard some redundant matrices before applying any test.

(3) Check whether the current matrix, A, represents a prime poset. The main function of the test is given in §3. If P is a prime then the matrix is accepted and the algorithm will execute the next step. Otherwise, A is discarded and go to step (1). The three steps, together, construct a strictly upper triangular matrix in the bucket form which represents a natural labeled prime poset. While, our purpose is to enumerate the numbers of unlabeled prime posets. So, the number of all matrices corresponding to the same poset that is represented by the current matrix must be computed to get the weight of this matrix regarding to height i.e. its contribution in the required counting. Then carry out the following steps to compute the height and weight for each accepted matrix:

(4) Determine the height of the poset that is represented by the accepted matrix. This can be done via counting the maximum number of elements in a chain of the poset. This is achieved by determining for each $v \in \mathrm{Max}(P)$, the set of all maximal elements of P, the maximum cardinality, $h(v)$, of a chain in P having V as its maximum element. Then get $h(P) = \max\{h(v) : v \in \max(P)\}$.

(5) Determine the weight of the accepted matrix of enumeration. This can be done by, first, partitioning the elements of P into blocks such that v_i and v_j belong to the same block if and only if they have the same number of predecessors and successors. If Γ denotes the set of all permutations σ of $\{1, 2, \ldots, n\}$ such that σ permutes only those elements of the same block of P. For $\sigma \in \Gamma$ denote by $\sigma(A)$ the matrix obtain from A by applying simultaneously σ to its rows and columns.

All the matrices $\sigma(A), \sigma \in |\Gamma|$ represent the same poset. To find the weight of A, the algorithm determines the cardinality, $k(A)$, of the set $\{\sigma(A) : \sigma \in \Gamma\}$. First of all, the algorithm calculates $|\Gamma|$ which is $r_1!r_2! \dots r_k!$ where $r_1, r_2, \dots, r_k$ are the sizes of the blocks of P. Then $m(A) = |\{\sigma \in \Gamma : \sigma(A) = A\}|$ is determined. This is done via generating and counting all possible $\sigma \in \Gamma$ such that $\sigma(A) = A$. Then $k(A) = |\Gamma|/m(A)$. Thus the matrix A is assigned the weight $1/k(A)$. For example, Fig.2 illustrates a 10-element prime poset P, its matrix-representation $A(P)$ and its blocks. Clearly $|\Gamma| = r_5! = 2$, i.e. there exist only two permutations that can be applied on $A(P)$ to obtaion isomorphic copies of P. Consequently, $m(A) = 1, k(A) = 2$ and so the weight of $A(P) = \frac{1}{2}$. Therefore, this method is preferable rather than to generate and count all possible permutations ($\leq 10!$ in the case of the example) that can be applied on P to obtain isomorphic copies of it.

The advantage of the calculation of the weight avoids storing too many matrices. Since we do not need any comparison for isomorphism. So, we need only one matrix in the memory at any time, i.e. $O(n^2)$ space. Clearly, the running time of the weight step is given by $O(n^2 \prod_{i=1}^{m} (r_i!))$, where $m \leq n$ is the number of blocks of poset. Therefore, the technique of calculating weight depends on the largest value of r_i. If the values of r_i's are quit small (with respect to n), then the technique will be considered a bounded time algorithm. Unfortunately, the whole counting algorithm depends on the number of constructed matrices which is exponential so the problem of counting unlabeled prime is still NP-hard as in the general case.

5 Appendix

According to the well-known fact that "almost all posets are prime", [16]. Likewise total posets, the numbers of prime posets $P_{nk}, k \leq n$ will be rapidly increasing as n increases. Therefore, it is expected that the running time increases more rapidly like the case of counting unlabeled posets on 14 elements, [9]. For this reason we excuted the suggested algorithm up to 13 elements only. In the case of calculating $P_{13} = \sum_{k=1}^{13} P_{13,k}$ the author used a single processor (2.8 GMHZ) for about 2 months to generate in averge about 2850 unlabeled prime posets per second. Table (1) contains the values of P_{nk} for $k \leq n \leq 13$. The results of the paper agree with those for $n \leq 12$ introduced in [3] and [11]. Moreover, it strengthens the case that P_{10} in [17] is incorrect (In [17], P_{10} is less than the actual number with 35 posets)

References

[1] B.I.Bayoumi, M.H.El-Zahar and S.M.Khamis, Asymptotic enumeration of N-free partial orders, Order 6 (1989), 219-232.

[2] B.I.Bayoumi, M.H.El-Zahar and S.M.Khamis, Counting 2-dimensional posets, Disc Math., 131 (1994), 29-37.

[3] B.I.Bayoumi, M.H.El-Zahar and S.M.Khamis, Algorithmic counting of types of UPO graphs and posets, Congressus Numerantium 127 (1997), 117-122.

[4] C.Chaunier and N.Lygerõs, The number of orders with thirteen element, Order 9 (1992), 203-204.

[5] C.Culberson and G.J.E.Rawlins, New results from an algorithm for counting posets, Order 7 (1991), 361-374.

[6] M.H.El-Zahar, Enumeration of ordered sets, in Algorithm and Order (ed. I. Rival), NATO Adv. Sci. Inst. Ser. C: Math. Phys. Sci. (1989), 327-352, Klluwer Academic Publishers.

[7] Mohamed H. El-Zahar and Soheir M. Khamis, Enumeration of series-parallel posets according to heights, J. of the Egyptian Mathematical Society, 8 (2000), 1-7.

[8] P.Hanlon, Counting interval graphs, Trans. Amer. Math. Soc., 272 (1982), 383-426.

[9] J.Heitzig and J.Reinhold, The number of unlabeled orders on fourteen elements, Order 17 (2000), 333-341.

[10] S.M.Khamis, On Enumerative and algorithmic aspects of partially ordered sets and graphs, Thesis of Ph.D., Dept. of Math. Faculty of Science, Ain Shams University, Cairo, Egypt (unpublished) (1991).

[11] S.M.Khamis, On numerical counting of prime, UPO, and the general type of posets according to heights, Congressus Numerantium 146 (2000), 157-171.

[12] S.M.Khamis, Heigh counting of unlabeled interval and N-Free posets, Discrete Math. 275 (2004), 165-175.

[13] D.A.Klarner, The number of graded partially ordered sets, J. Combin. Theory 6 (1969), 12-19.

[14] D.A.Klarner, The number of classes of isomorphic graded partially ordered sets, J.Combin. Theory 9 (1970), 412-419.

[15] D.J.Keitman and B.L.Rothschild, Asymptotic enumeration of partial orders, Trans. Am. Math. Soc. 20 (1975), 205-220.

[16] R.H.Mohring, Almost all comparability graphs are UPO, Discrete Math. 50 (1984). 63-70.

[17] R.H.mohring, Algorithmic aspects of comparability graphs and interval graphs, in Graphs and Order: The Role of Graphs in the Theory of Ordered Sets and Applications (ed.I. Rival), NATO Adv. Study Inst. Ser. C: Math. Phys. Sci., 147 (1985), 41-102.

[18] R.P.Stanley, Enumeration of posets generated by disjoint unions and ordinal sums" Proc. Am. Math. Soc. 45 (1974), 295-299.

Received 11/5/2004
Revised 10/10/2004