Hindawi
Journal of Mathematics
Volume 2023, Article ID 9920700, 11 pages
https://doi.org/10.1155/2023/9920700

Research Article

An Efficient Algorithm for Decomposition of Partially Ordered Sets

Elsayed Badr,[1] **Mohamed EL-Hakeem**,[2] **Enas E. El-Sharawy**,[3] **and Thowiba E. Ahmed**[3]

[1]*Scientific Computing Department, Faculty of Computers and Artificial Intelligence, Benha University, Banha, Egypt*
[2]*Artificial Intelligence Department, Faculty of Computers and Artificial Intelligence, Benha University, Banha, Egypt*
[3]*Computer Science Department, College of Science and Humanities, Imam Abdulrahman Bin Faisal University, P.O. Box 31961, Jubail, Saudi Arabia*

Correspondence should be addressed to Elsayed Badr; badrgraph@gmail.com

Received 23 January 2023; Revised 23 April 2023; Accepted 26 April 2023; Published 9 May 2023

Academic Editor: Mohammad W. Alomari

Efficient time complexities for partial ordered sets or posets are well-researched field. Hopcroft and Karp introduced an algorithm that solves the minimal chain decomposition in $O(n^{2.5})$ time. Felsner et al. proposed an algorithm that reduces the time complexity to $O(kn^2)$ such that n is the number of elements of the poset and k is its width. The main goal of this paper is proposing an efficient algorithm to compute the width of a given partially ordered set **P** according to Dilworth's theorem. It is an efficient and simple algorithm. The time complexity of this algorithm is $O(kn)$, such that n is the number of elements of the partially ordered set **P** and k is the width of **P**. The computational results show that the proposed algorithm outperforms other related algorithms.

1. Introduction

A partially-ordered set (poset) $P = (X, \prec)$ is a set of elements X, along with a binary relation, $\prec$, having the property that $\prec$ is transitive and antisymmetric for all elements in $\mathcal{P}$. The scheduling problem can be formulated as a partially ordered set. In the scheduling problem, there are some jobs that are parallel (can be executed simultaneously) and some that are nonparallel (cannot be executed simultaneously). Finding the largest number of nonparallel jobs is equivalent to finding the width of the partially ordered set. Accordingly, there are a very large number of applications in various fields based on the different applications of the scheduling problem. The width of a poset is the largest antichain cardinality.

Two elements a_i and a_j are *comparable* if $a_i \prec a_j$ otherwise are in comparable if $a_i \prec a_j$. A nonempty subset $C = \{a_1, a_2, \ldots, a_k\} \subseteq X$ is called a chain in **P** if $a_1 \prec a_2 \prec \ldots \prec a_k$. An antichain is a nonempty set in which there are no comparable pairs of elements. It is always feasible to divide the elements of X into disjoint chains since every single element in X is a chain in and of itself. Such a partition is referred to as a decomposition, and the minimum decomposition is one that has the fewest possible disjoint chains. The size of a minimum decomposition is equivalent to the size of a maximum antichain, according to Dilworth [1].

In 1973, Hopcroft and Karp [2] proposed the algorithm that takes $O(mn^2)$ time complexity to determine the width of a given poset such that n is the number of elements in **P** and m is the comparable pairs in **P**. They proved that the $O(mn^2)$ time complexity equals $O(n^{5/2})$. In 2003, Felsner et al. [3] demonstrated that determining whether an order has width k can be done in $O(kn^2)$ time, such that n is the number of components in **P** and k is the width of **P**. For special cases of the posets, In 1979, Papadimitriou and Yannakakis [4] introduced a linear time complexity $O(n)$ for determine the width of an interval partially ordered set with n elements. In 1980, Golumbic [5] proved that the width of a 2-dimensional poset can be computed in $O(n \log n)$ time such that n is the elements of **P**. In 1992, Garg [6] suggested a novel approach that requires $O(n^2m)$ comparisons, where n represents the number of chains and m is the maximum number of elements in any chain. They demonstrate that the temporal complexity of this approach for a chain poset is O

(m n log n). In 2022, Badr et al. [7] introduced the integer linear programming model (ILPM), which determines the width of a given poset. ILPM is distinguished from the previous mathematical models by its efficiency. For more details about the poset decomposition, the reader can refer to [1, 8–13]. The reader can refer to [14–17] for formulating the mathematical models that related to the poset decomposition.

The discipline of space-efficient data structures for partially ordered sets or posets has been intensively studied. A poset with n elements can be represented in $n^2/4 + o\ (n^2)$ bits [18] or in $(1+\epsilon)\ n \log n + 2nk + o(nk)$ bits [19], where k is the width of the poset. Yanagita et al. [20] made the latter data structure occupy $2n\ (k-1) + o\ (nk)$ bits by considering topological labeling on the elements of posets.

The best algorithm for this problem up to now need $O\ (kn^2)$ time, where n is the number of elements of a poset **P**. In this paper, we propose an efficient algorithm to compute the width of a given partially ordered set **P** according to Dilworth's theorem. It is an efficient and simple algorithm. The time complexity of this algorithm is $O\ (kn)$, such that n is the number of elements of the partially ordered set **P** and k is the width of **P**. The computational results show that the proposed algorithm outperforms other related algorithms.

The remaining of this paper is organized as follows: The proposed algorithm description is introduced in Section 2. In Section 3, the experimental results analysis of the proposed algorithm on different benchmark posets are provided. In Section 4, conclusions are made.

2. The Proposed Algorithm Description

In this section, we propose an efficient algorithm for determining the width of a poset P with n elements in $O\ (kn)$ time, such that n is the number of components in **P** and k is the width of **P**. On the other hand, the numerical example is introduced for explaining the mechanism of the proposed algorithm. Finally, we introduce a proposition that proves the proposed algorithm has the time complexity of $O\ (kn)$.

There are three basic parameters that affect the time complexity of finding the width of a given poset **P**. These parameters are n, k, and m the elements of **P**, the width of **P**, and the number of comparable pairs in **P,** respectively. Table 1 shows six different datasets of standard posets evaluate the proposed algorithm. On the other hand, it compares between the proposed algorithm (Algorithm 1) and Felsner et al.' algorithm [3].

Here, we describe the mechanism of the proposed algorithm. The algorithm works as follows: It reads the incidence matrix A of a poset P. It sums up all the rows and columns of A. It adds the sum of each row with the sum of the corresponding column. We choose the lowest sum (the lowest vertex connected to the others). We remove the selected vertex and the connected vertices with it from the matrix A. It repeats the above statements until the size of $A = 0$.

Example 1. Let the poset P be the crown poset C_3 as shown in Figure 1.

We put an initial value for the width of P: width = 0.

Iteration 1

Generate the incidence matrix A for the crown poset C_3:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{1}$$

It sums up all the rows and columns of A. It adds the sum of each row with the sum of the corresponding column. Sum_cr $[i]$ = column [i] + row [i]

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{2}$$
$$\text{Sum\_cr} = [2, 2, 2, 2, 2, 2].$$

We choose the lowest sum of sum_cr (min (sum_cr)) = 2 with index = 6 (break the tie by choosing the minimum value arbitrary).

Table 1: Six different datasets of standard posets evaluate the proposed algorithm.

No.	Poset	Width	Case	Felsner et al. [3], $O(mn^2) = O(kn^2)$	Algorithm 1, $O(kn)$
1	Chain poset	1	best case	$O(n^2)$	$O(n)$
2	K-tower	2	best case	$O(n^2)$	$O(n)$
3	Divisor poset	$\approx \log n$	Average case	$O(n^2 \log n)$	$O(n \log n)$
4	Crown poset	$n/2$	worst case	$O(n^3)$	$O(n^2)$
5	$2n$-cycle poset	$n/2$	worst case	$O(n^3)$	$O(n^2)$
6	Antichain poset	$n-1$	worst case	$O(n^3)$	$O(n^2)$

```
Input:
  n: the number of elements of a poset P
  A_{n×n}: the incidence matrix of a poset P
Output: the length of its largest antichain width (P)
Begin
width = 0
while length of matrix A! = 0 do
  sum_column = sum(A, axis = 0)
  sum_row = sum(A, axis = 1)
  sum_cr = $\sum_{i=0}^{n}$sum_ column$_i$ + sum_ row$_i$
  M = inf
  for j = 1 to n
    if (sum_cr [j] ≤ M):
      M = sum_cr [j]
      Index = j
    end
  end
  Index_row = []
  Index_column = []
  for j = 1 to n
    if (A[j, Index] == 1):
      Index_row.append(j)
    end
    if (A[Index, j] == 1):
      Index_column.append (j)
    end
  end
  Delete all columns and row when value = 1 in (rows = Index_row [:],
  columns = Index_column [:] and column = Index)
    width +=1
  end
  return the best population found (the width of its largest antichain)
End
```

Algorithm 1: Find the width of a poset.

We determine all the vertices that link to the current vertex x_6 which are x_1 and x_2.

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \quad (3)$$

We remove the current vertex x_6 and the vertices that linked with it.

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \text{width} = \text{width} + 1. \quad (4)$$

Iteration 2

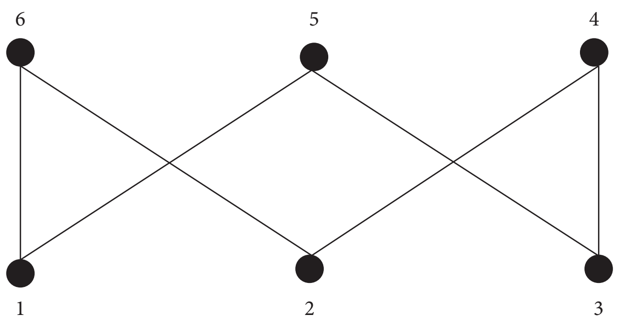

FIGURE 1: The crown poset C_3.

It sums up all the rows and columns of A. It adds the sum of each row with the sum of the corresponding column. Sum_cr $[i]$ = column [i] + row $[i]$

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \text{Sum\_cr} = [2, 1, 1]. \tag{5}$$

We choose the lowest sum of sum_cr (min (sum_cr)) = 1 with index = 3 (break the tie by choosing the minimum value arbitrary).

We determine all the vertices that link to the current vertex x_3 which are x_1.

We remove the current vertex x_3 and the vertices that linked with it x_1.

$$A = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \text{width} = \text{width} + 1. \tag{6}$$

Iteration 3

$$\text{Sum\_cr} = [0]. \tag{7}$$

We eliminate the final element (singleton): A = []

$$\text{width} = \text{width} + 1. \tag{8}$$

The algorithm must now terminate. The final width equals three iterations.

Proposition 1. *The algorithm 1 requires O (kn) comparisons at most.*

Proof. Suppose *comp* (k) denotes the number of comparisons required in the k^{th} iteration of the while loop. Suppose T denotes the total number of iterations of the *while loop*. Therefore, $\sum_{k=1}^{k=T} \text{comp}(k)$ represents the total number of comparisons. Let α denotes the total number of iterations of *for loop* to get the minimum index, β denotes the total number of iterations of *for loop* to select vertices to connected with selected vertex and γ denote the number of deleted columns and rows. Therefore, $\sum_{k=1}^{k=T} \text{comp}\,(k) = T * (\alpha + \beta + \gamma)$. We note that $T = k$ because every iteration the width increases by one until the size of the matrix = 0. It is clear that $\alpha = n$ because it searches for the minimum value in an array with size n. It is clear that $\beta = n$ because β *is related to* α (for every value of α there is a vertex in the matrix A). It is clear that, γ is less than or equal to n. Therefore,

$$\sum_{k=1}^{k=T} \text{comp}\,(\text{k}) = k * (\gamma + \beta + \gamma) = O(k * (n + n + n)) = O(3kn) = O(kn). \tag{9}$$

The time complexity of the proposed algorithm is $O\,(kn)$. □

3. Datasets

The time complexity of the proposed approach clearly depends on the width of the poset. We assess the proposed algorithm with a dataset of standard posets which have different width. Six different datasets of standard posets evaluate the proposed algorithm, as shown in Table 1.

The first dataset is the chain posets. A poset **P** is called chain if all elements of ***P*** are comparable. The second dataset is the *p-tower* posets. It is obtained by substituting p antichains (stables) of cardinality 2 in a total order on p vertices. The third dataset is the *divisor* posets. For any natural number n, the collection X of all of the natural numbers that divide n represents the divisor posets. For any two elements in X, *a*, *b*, *write* $a < b$ A = $[0]a|b$ *and* $a = b$, *that is if a divides b and* $a = b$. The fourth dataset is the crown poset. The poset with $2n$-elements $a_1, \ldots, a_n; b_1, \ldots, b_n$ is called crown poset

Table 2: A comparison between ILPM and the proposed algorithm on $2n$-cycle.

$N=2n$	Comparable pairs, N	Standard $w(P)$	ILPM		Proposed algorithm	
			$w(P)$	CPU time	$w(P)$	CPU time
50	50	25	25	0.089045	25	0.002301
100	100	50	50	0.106691	50	0.0043507
150	150	75	75	0.165378	75	0.0081102
200	200	100	100	0.239956	100	0.0108141
250	250	125	125	0.431634	125	0.016587
300	300	150	150	0.474245	150	0.0246753
350	350	175	175	0.622573	175	0.0349737
400	400	200	200	0.756483	200	0.0705701
450	450	225	225	0.948467	225	0.0999672
500	500	250	250	1.216359	250	0.1508073

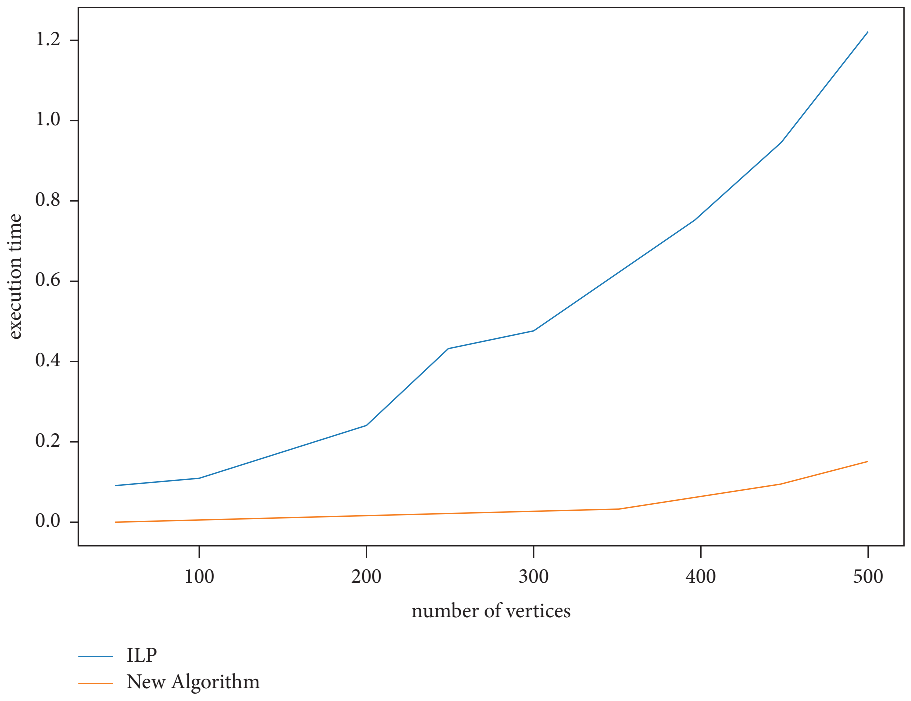

Figure 2: A comparison between ILPM and the proposed algorithm on $2n$-cycle.

Table 3: A comparison between ILPM and the proposed algorithm on crown posets.

$N=2n$	Comparable pairs, $N(N-1)$	Standard $w(P)$	ILPM [7]		Proposed algorithm	
			$w(P)$	CPU time	$w(P)$	CPU time
50	600	25	25	0.235665	25	0.0017501
100	2450	50	50	1.215363	50	0.0036623
150	5550	75	75	3.590369	75	0.0067241
200	9900	100	100	9.038238	100	0.0075052
250	15500	125	125	16.18856	125	0.010577
300	22350	150	150	27.90345	150	0.0135506
350	30450	175	175	43.35807	175	0.0184247
400	39800	200	200	69.64655	200	0.027318
450	50400	225	225	96.15276	225	0.0280215
500	62250	250	250	136.2815	250	0.0457993

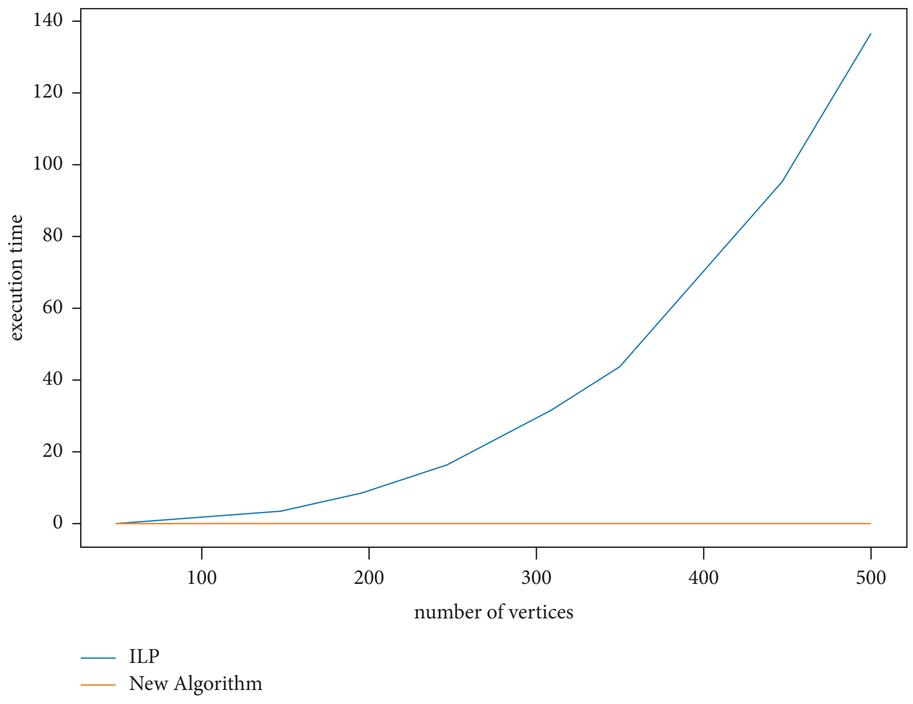

Figure 3: A comparison between ILPM and the proposed algorithm on crown posets.

Table 4: A comparison between ILPM and the proposed algorithm on divisor posets.

N	Comparable pairs $<N$	Standard w (P)	ILPM [7]		Proposed algorithm	
			w (P)	CPU time	w (P)	CPU time
50	12	2	2	0.043329	2	0.0002642
100	27	3	3	0.047882	3	0.0003259
150	42	4	4	0.054162	4	0.0003274
200	48	3	3	0.059799	3	0.0002829
250	22	2	2	0.055196	2	0.0002299
300	90	5	5	0.070541	5	0.0004843
350	42	4	4	0.054761	4	0.0006213
400	75	3	3	0.065461	3	0.0002782
450	90	5	5	0.070376	5	0.0004512
500	48	3	3	0.05527	3	0.0002953

if every element a_i is comparable with every element b_j excluding the case when $i=j$. The fifth dataset is the alternating $2n$-cycle poset. An ordered set $(x_1, y_1, x_2, y_2, \ldots, x_n, y_n)$ of size $2n$, $n \geq 2$, with these comparabilities, and no others, is called an *alternating* $2n$*-cycle*, or more briefly, a $2n$-cycle. The sixth dataset is the antichain poset. A poset **P** is called antichain if all elements of **P** are incomparable.

Table 1 shows the superiority of the proposed algorithm on the algorithm that was proposed in [3] for three cases.

4. Computational Study

Here, we evaluate the proposed algorithm by We evaluate the proposed algorithm with a dataset of standard partially ordered sets which have different width. Six different datasets of standard posets evaluate the proposed algorithm, as shown in Table 1. All of these are compatible with a *PC* having a Core i7 CPU@2.8 GHz, 8 GB of RAM, and a 64 bit operating system. They were all written in Python. The values of different algorithm-specific parameters are properly set to achieve their optimal performance, as shown in Table 1.

Table 2 and Figure 2 demonstrate that the proposed algorithm is superior to the mathematical model *ILPM* [7] in terms of the execution time required to determine the width of the $2n$-cycle poset. This superiority is due to the fact that the width is the primary parameter of the proposed algorithm, while the number of comparable pairs and the number of elements in the $2n$-cycle poset are secondary parameters. Alternatively, the mathematical model *ILPM* relies on the number of comparable pairs as its primary parameter, while the width and number of elements in the $2n$-cycle poset are secondary parameters. Figure 1 illustrates the execution time advantage of the proposed algorithm over the mathematical model for the $2n$*-cycle poset*. The second and fifth columns indicate that the running time of *ILPM*

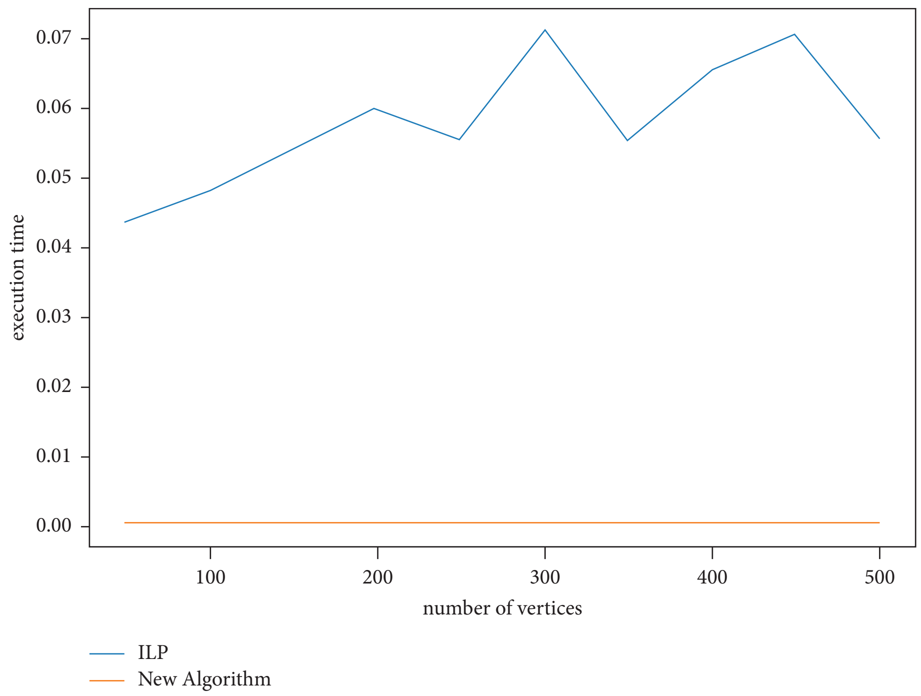

Figure 4: A comparison between ILPM and the proposed algorithm on divisor posets.

Table 5: A comparison between ILPM and the proposed algorithm on k-tower posets.

N	Comparable pairs $N(N-2)/2$	Standard $w(P)$	ILPM [7]		Proposed algorithm	
			$w(P)$	CPU time	$w(P)$	CPU time
50	1200	2	2	4.095658	2	0.000384
100	4900	2	2	30.97442	2	0.000446
150	11100	2	2	61.27756	2	0.000363
200	19800	2	2	150.0762	2	0.000415
250	31000	2	2	140.8486	2	0.000642
300	44700	2	2	283.7612	2	0.000654
350	60900	2	2	415.9408	2	0.000629
400	79600	2	2	399.5216	2	0.000634
450	100800	2	2	2568.489	2	0.000843
500	124500	2	2	2789.078	2	0.000850

increases with the number of comparable pairs, whereas the sixth and seventh columns indicate that the running time of the proposed algorithm increases with the width of the $2n$-cycle poset. We conclude from the preceding that the width of a poset **P** is an important and influential parameter for the proposed algorithm, whereas the number of comparable pairs is an important and influential parameter for the mathematical model *ILPM*.

Table 3 and Figure 3 demonstrate that the width is the most important parameter of the proposed algorithm, followed by the number of comparable pairs and the number of elements in the poset P. Alternatively, the primary parameter of the mathematical model ILPM is the number of comparable pairs, while the width and number of elements in the poset P are secondary parameters. Table 3's second and fifth columns demonstrate that as the number of comparable pairs increases, so does the execution time of the mathematical model ILPM. For instance, when $N = 200$, the execution time for the crown group is 9.038238 seconds, and when $N = 500$, it is 136.2815 seconds.

On the divisor poset, Table 4 and Figure 4 demonstrate the superiority of the proposed algorithm over the mathematical model ILPM [7]. Table 4's second column depicts the fluctuation of the number of comparable pairs between increases and decreases, but maintains that a longer execution time corresponds to a greater number of comparable pairs in the mathematical model ILPM.

Table 5 and Figure 5 demonstrate that the proposed algorithm is preferable to the ILPM [7] mathematical model of the k-tower poset. From the sixth and seventh columns of Table 5, it can be observed that the execution time of the proposed algorithm increases slightly as the number of comparable pairs and the number of elements of the k-tower poset increase, confirming that the number of comparable pairs and the number of elements are secondary parameters

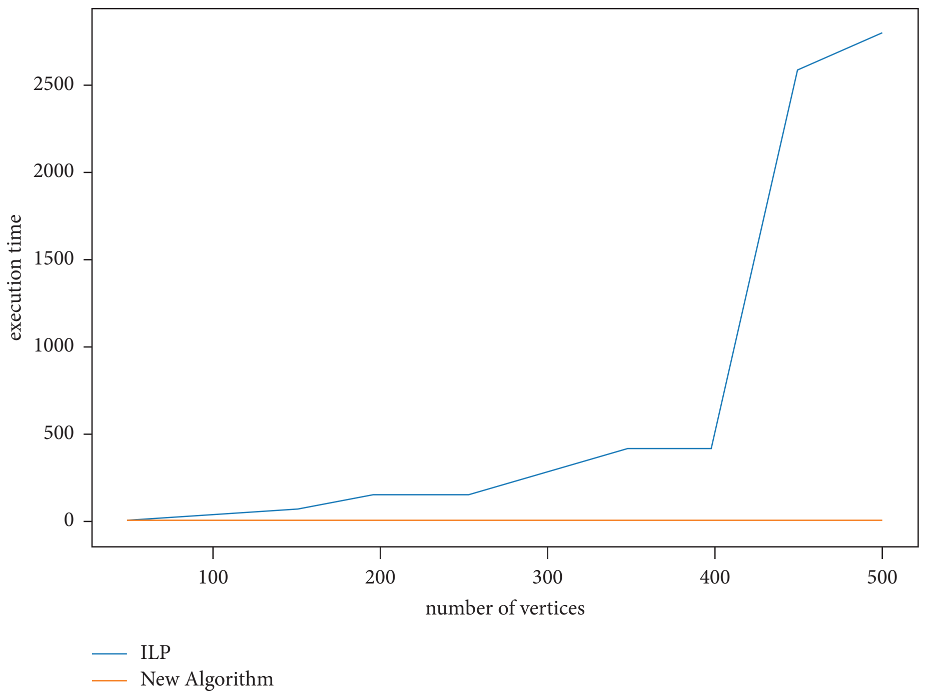

FIGURE 5: A comparison between ILPM and the proposed algorithm on k-tower posets.

TABLE 6: A comparison between ILPM and the proposed algorithm on chain posets.

N	Comparable pairs, $N(N-1)/2$	Standard $w(P)$	ILPM [7]		Proposed algorithm	
			$w(P)$	CPU time	$w(P)$	CPU time
50	1225	1	1	5.407565	1	0.000338
100	4950	1	1	17.2817	1	0.000384
150	11175	1	1	57.52828	1	0.000414
200	19900	1	1	58.09319	1	0.00046
250	31125	1	1	91.68824	1	0.000555
300	44850	1	1	146.893	1	0.000559
350	61075	1	1	209.748	1	0.000653
400	79800	1	1	301.1966	1	0.000845
450	101025	1	1	2956.88	1	0.005727
500	124750	1	1	3591.351	1	0.007079

of the proposed algorithm. In contrast, we find that the execution time of the mathematical model ILPM increases significantly as the number of comparable pairs of the k-tower poset increases, confirming that the number of comparable pairs is a fundamental and influential parameter for the mathematical model ILPM.

Table 6 and Figure 6 show that the proposed algorithm is superior to the ILPM [7] mathematical model of chain posets. It can be seen in the sixth and seventh columns of Table 6 that the execution time of the proposed algorithm increases slightly as the number of comparable pairs and the number of elements of the chain posets increase, confirming that the number of comparable pairs and the number of elements are secondary parameters of the proposed algorithm. In contrast, we discover that the execution time of the mathematical model ILPM increases considerably as the number of comparable pairs of the chain posets increases, confirming that the number of comparable pairs is an essential and influential parameter for the mathematical model ILPM.

Recall that the width of a poset **P** is dependent on three fundamental parameters: the number of elements in **P**, its

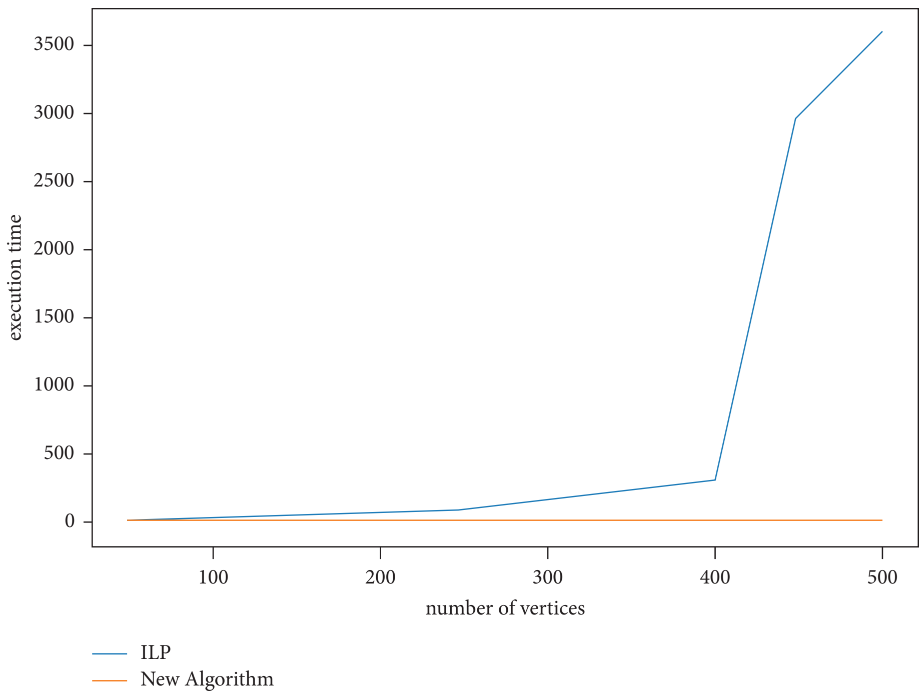

Figure 6: A comparison between ILPM and the proposed algorithm on chain posets.

Table 7: A comparison between ILPM and the proposed algorithm on antichain posets.

N	Comparable pairs	Standard w (P)	ILPM [7]		Proposed algorithm	
			w (P)	CPU time	w (P)	CPU time
50	0	50	50	0.050901	50	0.003775
100	0	100	100	0.053883	100	0.009561
150	0	150	150	0.071736	150	0.015508
200	0	200	200	0.062894	200	0.024025
250	0	250	250	0.071185	250	0.04499
300	0	300	300	0.074197	300	0.063899
350	0	350	350	0.082339	350	0.089227
400	0	400	400	0.085237	400	0.158079
450	0	450	450	0.090219	450	0.257679
500	0	500	500	0.162822	500	0.388347

width, and the number of comparable pairs in **P**. Table 7 and Figure 7 demonstrate that after 300, the proposed algorithm increases sharply. The weights of the aforementioned three parameters explain this significant increase. The execution time of the proposed algorithm increases relative to the execution time of the mathematical model as the poset size increases.

In terms of execution time for five posets (chain poset, k-tower, divisor poset, crown poset, and 2n-cycle poset) versus one poset (antichain poset) for ILPM [7], Tables 2–7 and

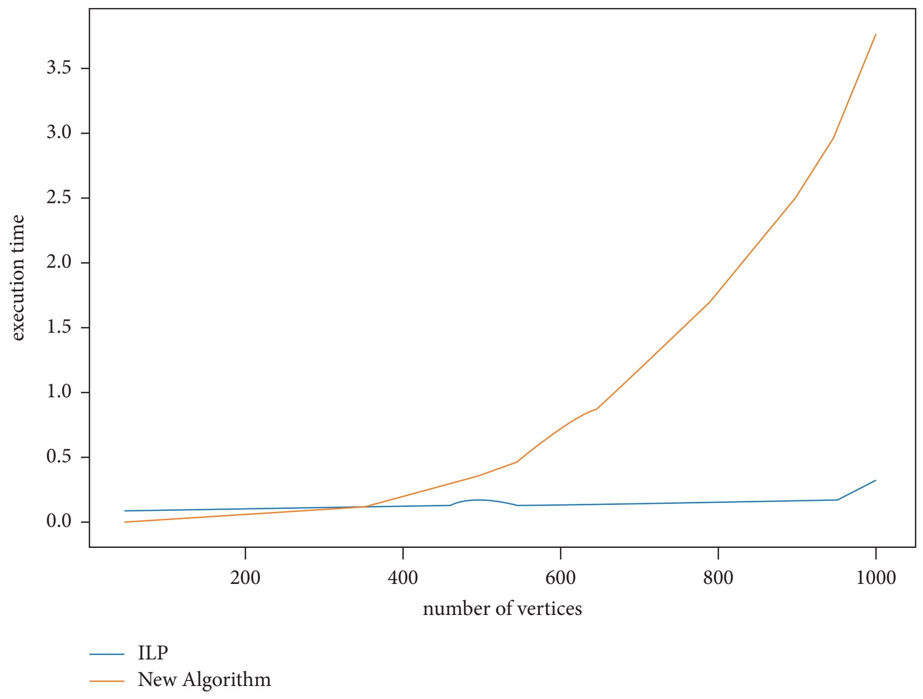

Figure 7: A comparison between ILPM and the proposed algorithm on antichain posets.

Figures 1–7 indicate that the proposed algorithm is superior to the mathematical model ILPM.

5. Conclusions

The main goal of this paper was proposing an efficient algorithm to compute the width of a given partially ordered set **P** according to Dilworth's theorem. It is an efficient and simple algorithm. The time complexity of this algorithm is $O(kn)$, such that n is the number of elements of the partially ordered set **P** and k is the width of **P**. The computational results show that the proposed algorithm outperforms other related algorithms. In future work, the mathematical relation among the size of poset **P**, its width, and the number of comparable pairs in **P** is introduced for special posets such as the chain, antichain, k-tower, divisor, crown, and $2n$-cycle posets.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] R. P. Dilworth, "A decomposition theorem for partially ordered sets," *Annals of Mathematics*, vol. 51, no. 1, pp. 161–166, 1950.

[2] J. E. Hopcroft and R. M. Karp, "An $n^{5/2} $ algorithm for maximum mbg," *SIAM Journal on Computing*, vol. 2, no. 4, pp. 225–231, 1973.

[3] S. Felsner, V. I. J. A. Y. Raghavan, and J. Spinrad, "Recognition algorithms for orders of small width and graphs of small Dilworth number," *Order*, vol. 20, no. 4, pp. 351–364, 2003.

[4] C. H. Papadimitriou and M. Yannakakis, "Scheduling interval ordered tasks," *SIAM Journal on Computing*, vol. 8, no. 3, pp. 405–409, 1979.

[5] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, NY, USA, 1980.

[6] V. K. Garg, "Some optimal algorithms for decomposed partially ordered sets," *Information Processing Letters*, vol. 44, no. 1, pp. 39–43, 1992.

[7] E. Badr, I. M. Selim, H. Mostafa, and H. Attiya, "An integer linear programming model for partially ordered sets," *Journal of Mathematics*, vol. 2022, Article ID 7660174, 9 pages, 2022.

[8] G. B. Dantzig and D. R. Fulkerson, "Minimizing the number of tankers to meet a fixed schedule," *Naval Research Logistics Quarterly*, vol. 1, no. 3, pp. 217–222, 1954.

[9] G. B. Dantzig and A. J. Hoffman, "Dilworth's theorem on partially ordered sets," *Linear Inequalities and Related Systems*, pp. 207–214, Princeton University Press, Princeton, NJ, USA, 1956.

[10] R. P. Dilworth, *Some Combinatorial Problems on Partially Ordered Sets Combinatorial Analysis," Book: The Dilworth Theorems: Selected Papers of Robert P*, Dil Worth, Springer Science + Business Media, Berlin, Germany, 1990.

[11] D. R. Fulkerson, "Note on Dilworth's decomposition theorem for partially ordered sets," *Proceedings of the American Mathematical Society*, vol. 7, no. 4, pp. 701-702, 1956.

[12] P. C. Fishburn, *Interval Graphs and Interval Orders*, Wiley, New York, NY, USA, 1985.

[13] E. Badr and M. Moussa, "On jump-critical ordered sets with jump number four," *Journal of Advances in Applied and Computational Mathematics*, vol. 1, pp. 8–13, 2014.

[14] E. M. Badr and H. elgendy, "A Hybrid water cycle - particle swarm optimization for solving the fuzzy underground water

confined steady flow," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 1, p. 492, 2020.

[15] E. M. Badr, Mustafa Abdul Salam, and H. Ahmed, *Optimizing Support Vector Machine Using Gray Wolf Optimizer Algorithm for Breast Cancer Detection*, in *Proceedings of the 1st international conference on information technology (IEEE/ ITMUSTCONF)*, Cairo, Egypt, April 2019.

[16] D. Salama AbdELminaam, A. M. Almansori, M. Taha, and E. Badr, "A deep facial recognition system using computational intelligent algorithms," *PLoS One*, vol. 15, no. 12, Article ID e0242269, 2020.

[17] E. Badr, M. Abdul Salam, S. Almotairi, and H. Ahmed, "From linear programming approach to metaheuristic approach: scaling techniques," *Complexity*, vol. 2021, Article ID 9384318, 10 pages, 2021.

[18] J. I. Munro and P. K. Nicholson, "Succinct posets," *Algorithmica*, vol. 76, no. 2, pp. 445–473, 2016.

[19] A. Farzan and J. Fischer, "Compact representation of posets," in *Proceedings of the 22nd International Conference On Algorithms And Computation (ISAAC)*, pp. 302–311, Springer-Verlag, Berlin, Heidelberg, January 2011.

[20] T. Yanagita, S. Chakraborty, K. Sadakane, and S. Rao Satti, "Space-Efficient Data Structure for Posets with Applications," in *Proceedings of the 18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, New York, NY, USA, August 2022.