



# Linear-Time Algorithms for Maximum-Weight Induced Matchings and Minimum Chain Covers in Convex Bipartite Graphs

Boris Klemz<sup>1</sup> · Günter Rote<sup>2</sup>

Received: 14 August 2019 / Accepted: 23 November 2021 / Published online: 6 January 2022  
© The Author(s) 2022

## Abstract

A bipartite graph  $G = (U, V, E)$  is *convex* if the vertices in  $V$  can be linearly ordered such that for each vertex  $u \in U$ , the neighbors of  $u$  are consecutive in the ordering of  $V$ . An *induced matching*  $H$  of  $G$  is a matching for which no edge of  $E$  connects endpoints of two different edges of  $H$ . We show that in a convex bipartite graph with  $n$  vertices and  $m$  *weighted* edges, an induced matching of maximum total weight can be computed in  $O(n + m)$  time. An *unweighted* convex bipartite graph has a representation of size  $O(n)$  that records for each vertex  $u \in U$  the first and last neighbor in the ordering of  $V$ . Given such a *compact representation*, we compute an induced matching of maximum cardinality in  $O(n)$  time. In convex bipartite graphs, maximum-cardinality induced matchings are dual to minimum *chain covers*. A chain cover is a covering of the edge set by *chain subgraphs*, that is, subgraphs that do not contain induced matchings of more than one edge. Given a compact representation, we compute a representation of a minimum chain cover in  $O(n)$  time. If no compact representation is given, the cover can be computed in  $O(n + m)$  time. All of our algorithms achieve optimal linear running time for the respective problem and model, and they improve and generalize the previous results in several ways: The best algorithms for the *unweighted* problem versions had a running time of  $O(n^2)$  (Brandstädt et al. in Theor. Comput. Sci. 381(1–3):260–265, 2007. <https://doi.org/10.1016/j.tcs.2007.04.006>). The *weighted* case has not been considered before.

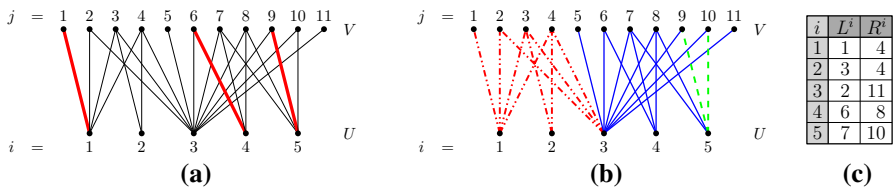
**Keywords** Graph algorithm · Induced matching · Chain cover · Convex bipartite graph · Certifying algorithm · Dynamic programming

---

✉ Boris Klemz  
boris.klemz@uni-wuerzburg.de  
Günter Rote  
rote@inf.fu-berlin.de

<sup>1</sup> Institut für Informatik, Universität Würzburg, Am Hubland, 97074 Würzburg, Germany

<sup>2</sup> Institut für Informatik, Freie Universität Berlin, Takustr. 9, 14195 Berlin, Germany



**Fig. 1** **a** A convex bipartite graph  $G = (U, V, E)$  containing an induced matching  $H$  of size 3, highlighted in red. Since we use natural numbers as elements of  $U$  and  $V$ , we will explicitly indicate whether we regard a number  $x$  as a vertex of  $U$  or of  $V$ . There is no induced matching with more than 3 edges: vertex  $3 \in U$  is adjacent to all vertices of  $V$  except  $1 \in V$ . Thus, if we match  $3 \in U$ , this can only lead to induced matchings of size at most 2. Furthermore, we cannot simultaneously match  $1 \in U$  and  $2 \in U$  since every neighbor of  $2 \in U$  is also adjacent to  $1 \in U$ . **b** A minimum chain cover of  $G$  with 3 chain subgraphs  $Z_1, Z_2, Z_3$  (in different colors and dash styles), providing an independent proof that  $H$  is optimal. Here,  $Z_1, Z_2, Z_3$  have disjoint edge sets, which is not necessarily the case in general. **c** The compact representation of  $G$

### 1 Introduction

A bipartite graph  $G = (U, V, E)$  is *convex* if the vertices in  $V$  can be numbered as  $1, 2, \dots, n_V$  so that the neighbors of every vertex  $i \in U$  form an *interval*  $\{L^i, L^i + 1, L^i + 2, \dots, R^i\} \subseteq \{1, 2, \dots, n_V\}$ , which we denote by  $[L^i, R^i]$ , see Fig. 1a. For such graphs, we consider the problem of computing an induced matching (a) of maximum cardinality or (b) of maximum total weight, for graphs with edge weights.

An *induced matching*  $H \subseteq E$  is a matching such that the subgraph of  $G$  induced by the matched vertices has  $H$  as its edge set. This amounts to requiring that no edge of  $E$  connects endpoints of two different edges of  $H$ , see Fig. 1a. More formally, a set  $H \subseteq E$  is an induced matching in  $G$  if for any two distinct edges  $(a, b), (a', b') \in H$ , the four vertices  $a, b, a', b'$  are pairwise distinct, and none of the edges  $aa', ab', ba', bb'$  is present in  $E$ .

In terms of the line graph, an induced matching is an independent set in the *square* of the line graph. The square of a graph connects every pair of nodes whose distance is one or two. Accordingly, we call two edges of  $E$  *independent* if they can appear together in an induced matching, or in other words, if their endpoints induce a  $2K_2$  (a disjoint union of two edges) in  $G$ . Otherwise, they are called *dependent*.

In convex bipartite graphs, maximum-cardinality induced matchings are dual to minimum chain covers. A *chain graph*  $Z$  is a bipartite graph that contains no induced matching of more than one edge, i. e., it contains no pair of independent edges. (Chain graphs are also called *difference graphs* [13] or *non-separable* bipartite graphs [8].) A *chain cover* of a graph  $G$  with edge set  $E$  is a set of (not necessarily induced) chain subgraphs  $Z_1, Z_2, \dots, Z_k$  of  $G$  such that the union of the edge sets of  $Z_1, Z_2, \dots, Z_k$  is  $E$ , see Fig. 1b. A chain cover with  $k$  chain subgraphs provides an obvious certificate that the graph cannot contain an induced matching with more than  $k$  edges. We will elaborate on this aspect of a chain cover as a certificate of optimality in Sect. 5. A *minimum* chain cover of  $G$  is a chain cover with the smallest possible number  $k$  of chain subgraphs. In a convex bipartite graph, we have the following strong duality statement, which is due to Yu et al. [27]: the maximum size of an induced matching is equal to the minimum number of chain subgraphs of a chain cover, see Theorem 3.

We denote the number of vertices by  $n_U = |U|$ ,  $n_V = |V|$ ,  $n = n_U + n_V$ , and the number of edges by  $m = |E|$ . If a convex graph is given as an ordinary bipartite graph without the proper numbering of  $V$ , it can be transformed into this form in linear time  $O(n + m)$  [2]. (In terms of the bipartite adjacency matrix, convexity is the well-known consecutive-ones property.) Unweighted convex bipartite graphs have a natural implicit representation [24] of size  $O(n)$ , which is often called a *compact representation* [14,23]: every interval  $\{L^i, L^i + 1, \dots, R^i\}$  is specified by its endpoints  $L^i$  and  $R^i$ , see Fig. 1c. Since the numbering of  $V$  can be computed in  $O(n + m)$  time, it is easy to obtain a compact representation in total time  $O(n + m)$  [23,25]. The chain covers that we construct will consist of convex bipartite subgraphs with the same ordering of  $V$  as the original graph. Thus, we will be able to use the same representation for the chain graphs of a chain cover.

*Related Work and Motivation* The problem of finding an induced matching of maximum size was first considered by Stockmeyer and Vazirani [26] as the “risk-free marriage problem” with applications in interference-free network communication. The decision version of the problem is known to be  $NP$ -complete in many restricted graph classes [5,16,17], in particular bipartite graphs [5,17] that are  $C_4$ -free [17] or have maximum degree 3 [17]. On the other hand, it can be solved in linear time in chordal graphs [4], and in polynomial time in weakly chordal graphs [6], trapezoid graphs,  $k$ -interval-dimension graphs and co-comparability graphs [12], amongst others. For a more exhaustive survey we refer to [9].

The class of convex bipartite graphs was introduced by Fred Glover [11], who motivates the computation of matchings in these graphs with industrial manufacturing applications. Items that can be matched when some quantity fits up to a certain tolerance naturally lead to convex bipartite graphs. The computation of matchings in convex bipartite graphs also corresponds to a scheduling problem of tasks of discrete length on a single disjunctive resource [15]. The problem of finding a (classic, not induced) matching of maximum cardinality in convex bipartite graphs has been studied extensively [10,11,25] culminating in an  $O(n)$  algorithm when a compact representation of the graph is given [25]. Several other combinatorial problems have been studied in convex bipartite graphs. While some problems have been shown to be  $NP$ -complete even if restricted to this graph class [1], many problems that are  $NP$ -hard in general can be solved efficiently in convex bipartite graphs. For example, a maximum independent set can be found in  $O(n)$  time (assuming a compact representation) [23] and the existence of Hamiltonian cycles can be decided in  $O(n^2)$  time [19]. For a comprehensive summary we refer to [14].

One of the applications given by Stockmeyer and Vazirani [26] for the induced matching problem can be stated as follows. We want to test (or use) a maximum number of connections between receiver-sender pairs in a network. However, testing a particular connection produces noise so that no other node in reach may be tested simultaneously. We remark that this type of motivation extends very naturally to convex bipartite graphs when we consider *wireless* networks in which nodes broadcast or receive messages in specific frequency ranges. Further, *weighted* edges can model the importance of connections.

Recently, Panda et al. [20] have built on our results to obtain polynomial algorithms for finding maximum-weight induced matchings in circular-convex and triad-convex bipartite graphs. These graph classes generalize convex bipartite graphs. Their algorithms use Theorem 1 of our paper as a subroutine.

*Previous Work* Yu et al. [27] describe an algorithm that finds both a maximum-cardinality induced matching and a minimum chain cover in a convex bipartite graph in runtime  $O(m^2)$ . Their procedure is improved by Brandstädt et al. [3], resulting in a runtime of  $O(n^2)$ . Chang [7] computes maximum-cardinality induced matchings and minimum chain covers in  $O(n + m)$  time in bipartite permutation graphs, which form a proper subclass of convex bipartite graphs.

*Our Contribution* We improve and generalize the previous results in several ways.

In Sect. 2 we give an algorithm for finding a maximum-weight induced matching in a convex bipartite graph in  $O(n + m)$  time. The previous best algorithm [3] had a runtime of  $O(n^2)$  and was restricted to the *unweighted* case.

In Sect. 3 we show that for the *unweighted* case, a further speed-up is possible if a compact representation of the graph is given: we specialize our algorithm from Sect. 2 to find induced matchings of maximum *cardinality* in  $O(n)$  runtime.

In Sect. 4 we extend the approach from Sect. 3 to obtain in  $O(n)$  time a compact representation of a minimum chain cover. If the input graph is not given in compact form, our algorithm can be adapted to produce a minimum chain cover (in standard representation) in  $O(n + m)$  time. This improves the previous best algorithm [3], which had a runtime of  $O(n^2)$ .

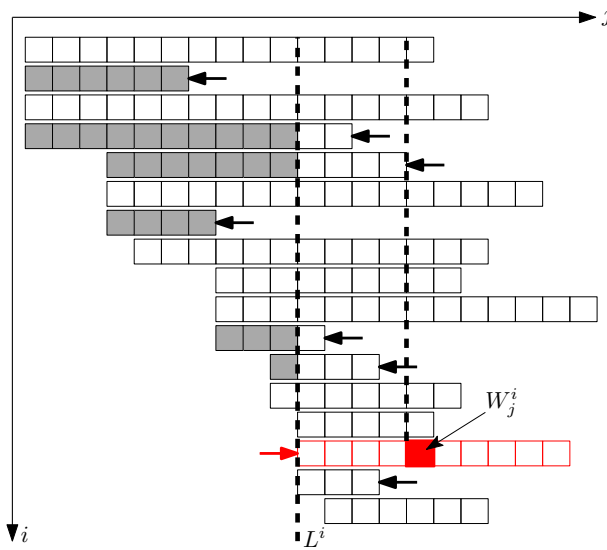
All of our algorithms achieve optimal running time for the respective problem and model. Our results for finding a maximum-cardinality induced matching also improve the running times of the algorithms of Pandey et al. [21] for the circular-convex and triad-convex case, as they use the convex case as a building block.

An induced matching together with a chain cover of the same cardinality constitute a certificate of optimality, of linear size. In Sect. 5, we show how to check this certificate for validity with very simple linear time algorithms. Thus, our algorithms for the unweighted case are *certifying algorithms*, see [18] for a survey about this concept.

Strong duality suggests that there should be a weighted chain cover with the same weight as the maximum weight of an induced matching in a convex bipartite graph. We discuss this aspect in Sect. 6 and leave it as an open problem to extend our maximum-weight matching algorithm to an efficient algorithm that also finds a minimum-weight chain cover.

## 2 Maximum-Weight Induced Matchings

In this section, we compute a maximum-weight induced matching of a given edge-weighted convex bipartite graph  $G = (U, V, E)$  in time  $O(n + m)$ . We generally write indices  $i \in U$  as superscripts and indices  $j \in V$  as subscripts. We consider  $E$  as a subset of  $U \times V$ . We assume that  $V = \{1, \dots, n_V\}$  is numbered as described in Sect. 1 and the interval  $\{L^i, L^i + 1, \dots, R^i\} \subseteq V$  of each vertex  $i \in U$  is given by the left endpoint  $L^i$  and right endpoint  $R^i$ . Each edge  $(i, j) \in E$  has a weight  $C_{ij}^i$ .



**Fig. 2** The table entries that enter in the computation of  $W_j^i$  are shaded: They lie in rows that end to the left of  $W_j^i$  (marked by arrows), and only the entries to the left of  $L^i$  are considered

Our dynamic-programming approach considers the following subproblems: For an edge  $(i, j) \in E$ , we define  $W_j^i$  as the cost of the maximum-weight induced matching that uses the edge  $(i, j)$  and contains only edges in  $U \times \{1, \dots, j\}$ . As we will see, the following dynamic-programming recursion computes  $W_j^i$ :

$$W_j^i = C_j^i + \max \left( \{ W_{j'}^{i'} \mid R^{i'} < j, j' < L^i \} \cup \{0\} \right) \tag{1}$$

The range over which the maximum is taken is illustrated in Fig. 2. In this recursion, we build the induced matching  $H$  of weight  $W_j^i$  by adding the edge  $(i, j)$  to some induced matching  $H'$  of weight  $W_{j'}^{i'}$ . We want  $H$  to be an induced matching: By construction, the edge  $(i', j')$  is independent of  $(i, j)$ , but it remains to show that the other edges of  $H'$  are also independent of  $(i, j)$ . In order to prove this (Lemma 2), we use some sort of transitivity of the independence relation for edge pairs (Lemma 1). First we state an observation:

**Proposition 1** *Two edges  $(i, j)$  and  $(i', j')$  are independent if and only if  $j' \notin [L^i, R^i]$  and  $j \notin [L^{i'}, R^{i'}]$ .  $\square$*

We emphasize that the independence of  $(i, j)$  and  $(i', j')$  does not require that the intervals  $[L^i, R^i]$  and  $[L^{i'}, R^{i'}]$  are disjoint, see, e.g., edges (4,6) and (5,9) in Fig. 1a.

**Lemma 1** *Let  $(i'', j''), (i', j'), (i, j) \in E$  with  $j'' < j' < j$ . Assume that  $(i'', j'')$  and  $(i', j')$  are independent, and  $(i', j')$  and  $(i, j)$  are independent. Then  $(i'', j'')$  and  $(i, j)$  are independent.*

**Proof** By Proposition 1, we have  $j'' \leq R^{i''} < j' \leq R^{i'} < j$  and  $j'' < L^{i'} \leq j' < L^i \leq j$ . Thus,  $j \notin [L^{i''}, R^{i''}]$  and  $j'' \notin [L^i, R^i]$ .  $\square$

**Lemma 2** *The recursion (1) is correct.*

**Proof** By Proposition 1, any edge  $(i', j')$  with  $j' < j$  that is independent of  $(i, j)$  satisfies  $R^{i'} < j$  and  $j' < L^i$ . By Lemma 1, all other edges  $(i'', j'')$  used to obtain the matching value  $W_{j'}^{i'}$  are also independent of  $(i, j)$ .  $\square$

We create a table in which we record the entries  $W_j^i$ . We assume that the intervals are sorted in nondecreasing order by  $L^i$ , that is,  $L^i \leq L^h$  for  $i < h$ . The values  $W_{L^i}^i, \dots, W_{R^i}^i$  form the  $i$ th row of the table. We fill the table row by row proceeding from  $i = 1$  to  $i = n_U$ . Each row  $i$  is processed from left to right. This ensures that the values on the right side of the recursion (1) have already been computed when they are needed, see Fig. 2. The only challenge in evaluating (1) is the maximum-expression, for which we introduce the following notation.

$$M_j^i = \max \left( \{ W_{j'}^{i'} \mid R^{i'} < j, j' < L^i \} \cup \{0\} \right) \tag{2}$$

Each row starts with the computation of the leftmost entry  $W_{L^i}^i$ , which we discuss later. When we proceed from  $W_j^i$  to  $W_{j+1}^i$  we want to go incrementally from  $M_j^i$  to  $M_{j+1}^i$ . Direct comparison of the respective defining sets leads to

$$M_{j+1}^i = \max \left( \{ M_j^i \} \cup \{ W_{j'}^{i'} \mid R^{i'} = j, j' < L^i \} \right) \tag{3}$$

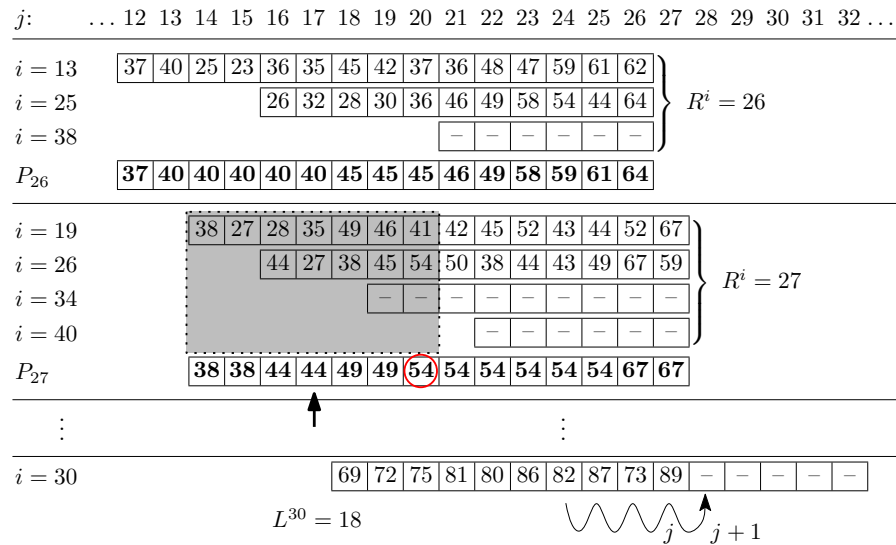
In order to evaluate the maximum of the second set in (3) efficiently, we group intervals  $i'$  with a common right endpoint  $R^{i'} = j$  together. Let  $S_j$  be the earliest startpoint of an interval with endpoint  $j$ . If there are no intervals with endpoint  $j$ , we set  $S_j := j$ . (It would be more logical to set  $S_j := j + 1$  in this case, but this choice makes the algorithm simpler.) We maintain an array  $P_j[\ell]$  for  $S_j \leq \ell \leq j$  that is defined as follows:

$$P_j[\ell] := \max \left( \{ W_{j'}^{i'} \mid R^{i'} = j; \text{row } i' \text{ has already been processed; } j' \leq \ell \} \cup \{0\} \right) \tag{4}$$

In a sense,  $P_j[\ell]$  is a provisional version of the expression  $\max\{ W_{j'}^{i'} \mid R^{i'} = j, j' \leq \ell \}$ , which takes into account only the already processed rows. For (3), we need the entry  $P_j[L^i - 1]$ , and we will show that all relevant entries have already been computed whenever we access this entry:

**Lemma 3** *When entry  $W_{j+1}^i$  is processed,  $M_{j+1}^i$  can be computed by the formula*

$$M_{j+1}^i = \begin{cases} \max\{M_j^i, P_j[L^i - 1]\}, & \text{if } L^i - 1 \geq S_j \\ M_j^i, & \text{otherwise} \end{cases} \tag{5}$$



**Fig. 3** Example. We are in the process of filling row 30 from left to right. All rows with smaller index  $i$  have been processed and are filled with the entries  $W_j^i$ . Unprocessed entries are marked as “-”. The figure does not show the rows in the order in which they are processed, but intervals with the same right endpoint  $R^i = r$  are grouped together. The bold entries collect the provisional maxima  $P_r$  in each group. By way of example, the encircled entry  $P_{27}[20] = 54$  is the maximum among the shaded entries of the intervals that end at  $R^i = 27$ , ignoring the yet unprocessed entries. As we proceed from  $j = 27$  to  $j = 28$  in row 30, the intervals with  $R^i = 27$  become relevant. The maximum usable entry from these intervals is found in position 17 of this array, because  $17 = L^{30} - 1$ . The entry  $P_{27}[17] = 44$  is marked by an arrow. The next entry  $W_{28}^{30}$  is equal to  $C_{28}^{30} + \max\{P_{27}[17], P_{26}[17], \dots, P_{17}[17]\}$  according to (2), by the interpretation of the entries  $P_j[\ell]$ . (Some of these entries might not exist.) The maximum in this expression is  $M_{28}^{30}$ , and in the algorithm it is computed incrementally by formula (5) from  $P_{27}[17]$  and the term  $M_{27}^{30} = \max\{P_{26}[17], \dots, P_{17}[17]\}$ , which has been used for calculating  $W_{27}^{30}$ . We can confirm that the minimum over which  $P_{27}[17]$  is defined involves no unprocessed entries at this time (Lemma 3). Later, when the next row  $i = 34$  in the group with  $R^i = 27$  is filled, the array  $P_{27}$  will be updated

Figure 3 illustrates the role of the arrays  $P_j$  when processing a row.

Before proving that (5) defines indeed the same quantity as (3), we first discuss that the expression (5) does not access the array  $P_j$  beyond its boundaries: The condition  $L^i - 1 \geq S_j$  ensures that the array index  $L^i - 1$  does not exceed the left boundary of the array  $P_j$ . Also, the index  $L^i - 1$  never exceeds the right boundary  $j$  of the array  $P_j$ , since  $L^i < j + 1 \leq R^i$ , and therefore  $L^i - 1 \leq j$ . Thus,  $P_j[L^i - 1]$  is always defined when it is accessed.

**Proof** (of Lemma 3) We distinguish three cases.

*Case 1* No interval ends at  $j$ , and accordingly,  $S_j = j$ .

In this case  $M_{j+1}^i = M_j^i$  in (3) since its rightmost set is empty. Since  $L^i < j + 1 \leq R^i$ , we have  $L^i - 1 < S_j = j$  and, thus, the right side of (5) evaluates also to  $M_j^i$ .

*Case 2* There exists an interval ending at  $j$ , and  $L^i - 1 < S_j$ . The right side of (5) evaluates to  $M_j^i$ . In (3), intervals  $i'$  that end at  $R^{i'} = j$  have  $L^{i'} \geq S_j > L^i - 1$ . Thus,

**Algorithm 1:** Weighted Maximum Induced Matching

```

▷ Preprocessing:
for  $r := 1$  to  $n_V$  do
    Find startpoint  $S_r$  of the longest interval  $[S_r, r]$  with endpoint  $r$ 
    Create an array  $P_r[S_r \dots r]$  and initialize it to 0.
    (If there is no such interval with endpoint  $r$ , set  $S_r := r$  and create an array with a single dummy
    entry  $P_r[r]$  that will remain at 0.)
▷ Main program:
 $F := 0$  ▷ maximum entry in finished intervals
for  $\ell := 1$  to  $n_V$  do
    ▷  $F = \max\{P_1[1], P_2[2], \dots, P_{\ell-1}[\ell-1]\}$ 
    for all rows  $i \in U$  with  $L^i = \ell$  do ▷ Process each interval  $i$  that starts at  $\ell$ 
         $r := R^i$ 
        ▷ Process the  $i$ th interval  $[L^i, R^i] = [\ell, r]$  and fill row  $i$  of the table:
         $M := M_\ell^i := F$  ▷  $M$  will be the current value of  $M_j^i$ 
         $W_\ell^i := C_\ell^i + M$  ▷ leftmost entry
        for  $j := \ell + 1$  to  $r$  do ▷ compute successive entries
            if  $S_j \leq \ell - 1$  then
                 $M := \max\{M, P_{j-1}[\ell-1]\}$  ▷  $M_j^i := \max\{M_{j-1}^i, P_{j-1}[\ell-1]\}$ 
                 $W_j^i := C_j^i + M$ 
            ▷ Go through the computed entries again to update the array  $P_r$ :
             $q := 0$  ▷ the row maximum so far
            for  $j := \ell$  to  $r$  do
                 $q := \max\{q, W_j^i\}$  ▷  $q = \max\{0, W_\ell^i, W_{\ell+1}^i, \dots, W_j^i\}$ 
                 $P_r[j] := \max\{P_r[j], q\}$ 
         $F := \max\{F, P_\ell[\ell]\}$  ▷ update  $F$  as  $\ell$  is incremented
    return  $F$  ▷ the maximum weight of an induced matching
    
```

an edge  $(i', j')$  with  $j' < L^i$  and  $R^{i'} = j$  does not exist, and the second set in (3) is empty. Therefore, (3) evaluates to  $M_{j+1}^i = M_j^i$ .

Case 3 There exists an interval ending at  $j$ , and  $L^i - 1 \geq S_j$ . In this case,  $P_j[L^i - 1]$  is defined:

$$P_j[L^i - 1] = \max\{W_{j'}^{i'} \mid R^{i'} = j, j' \leq L^i - 1, \text{row } i' \text{ already processed}\} \quad (6)$$

For each entry  $W_{j'}^{i'}$  with  $j' < L^i$ , we conclude that  $L^{i'} \leq j' < L^i$  and, thus, row  $i'$  has already been processed. This means that the condition that row  $i'$  has been processed is redundant, and (6) coincides with maximum of the right set in (3). □

After processing row  $i$  with startpoint  $\ell = L^i$  and endpoint  $r = R^i$ , we have to update the values in  $P_r[j]$ . This is straightforward.

It remains to discuss the computation of the first value  $W_\ell^i$  of the row. An edge  $(i', j')$ ,  $j' < \ell$  and edge  $(i, \ell)$  are independent if and only if the interval  $i'$  ends before  $\ell$ , that is  $R^{i'} < \ell$ . Since we process the intervals in nondecreasing order by their startpoints, it suffices to maintain a value  $F$  with the maximum  $W_{j'}^{i'}$



in all *finished* intervals: those intervals  $i'$  that end before  $\ell$ . In other words  $F = \max\{P_1[1], P_2[2], \dots, P_{\ell-1}[\ell - 1]\}$ . This value is easily maintained by updating  $F$  as  $\ell$  increases. The full details are stated as Algorithm 1.

The update of the array  $P_r[j]$  in the second loop can be integrated with the computation of  $W_j^i$  in the first loop. When this is done, the values  $W_j^i$  need not be stored at all because they are not used.

As stated earlier, when no interval ends at a point  $r \in V$ , we set  $S_r = r$ . The array  $P_r$  consists of a single dummy entry  $P_r[r] = 0$ . In this way, this case needs no special treatment in the algorithm.

We have described the computation of the *value* of the optimal matching. It is straightforward to augment the program so that the optimal matching itself can be recovered by backtracking how the optimal value was obtained, but this would clutter the program.

**Theorem 1** *A maximum-weight induced matching of an edge-weighted convex bipartite graph can be computed in  $O(n + m)$  time.*  $\square$

### 3 Maximum-Cardinality Induced Matchings

For the unweighted version of the problem, we assume a compact representation of a convex bipartite graph  $G = (U, V, E)$ , that is, for each  $i \in U$  we are given the startpoint  $L^i$  and endpoint  $R^i$  of its interval  $\{L^i, L^i + 1, \dots, R^i\}$ . This makes it possible to obtain a linear runtime of  $O(n)$ .

The recursion (1) can be specialized to the unweighted case by setting  $C_j^i \equiv 1$ .

$$W_j^i = 1 + \max \left( \{W_{j'}^{i'} \mid R^{i'} < j, j' < L^i\} \cup \{0\} \right) \quad (7)$$

This recursion is related to the previous algorithms [3,27] for computing maximum-cardinality induced matchings (and minimum chain covers) in convex bipartite graphs: Yu et al. describe a greedy procedure that colors the edges of the bipartite adjacency matrix with integer values  $\hat{W}_j^i$  [27, Greedy Decomposition Algorithm]. This procedure is later also used by Brandstädt et al. [3, Procedure: Greedy Coloring]. It assigns to each edge  $(i, j)$  the *smallest* possible color  $\hat{W}_j^i$  such that independent edges receive different colors. Figure 4 below shows an example of such a coloring. Formally, we can express this approach as a recursion:

$$\hat{W}_j^i = \min \left( \{1, 2, \dots, m\} \setminus \{\hat{W}_{j'}^{i'} \mid R^{i'} < j, j' < L^i\} \right) \quad (8)$$

Our assignment (7), by contrast, picks the next color after the *largest* color of an independent edge. By Lemma 1 it can be derived that the choices (7) and (8) agree, i.e.  $W_j^i = \hat{W}_j^i$  for every edge  $(i, j)$ . Thus, our explicit dynamic programming approach produces the same coloring as the algorithm of Yu et al. and it gives a new interpretation and a different explanation of the assigned colors.

The original implementation given in [27] runs in time  $O(m^2)$ . Brandstädt et al. [3] give an improved implementation which obtains the colors  $\hat{W}_j^i$  in  $O(n^2)$  time. Our



**Algorithm 2:** Unweighted Maximum Induced Matching, initial version

---

```

Set  $Q_1 := Q_2 := \dots := Q_{n_U} := 0$ 
 $F := 0$ 
for  $\ell := 1$  to  $n_V$  do
  for all rows  $i \in U$  with  $L^i = \ell$  do  $\triangleright$  Process each interval  $i$  that starts at  $\ell$ 
     $w := F + 1$   $\triangleright$  leftmost entry
     $t_w :=$  leftmost endpoint  $R^{i'}$  of a row  $i'$  that contains an entry  $W_j^{i'} = w$  with  $j < L^i \equiv \ell$ 
    if  $t_w < R^i$  then  $\triangleright$  There are two values  $w$  and  $w + 1$  in this row:
       $\triangleright W_j^i = w$  for  $j = L^i, \dots, t_w$ 
       $\triangleright W_j^i = w + 1$  for  $j = t_w + 1, \dots, R^i$ 
       $Q_{R^i} := \max\{Q_{R^i}, w + 1\}$   $\triangleright$  The largest entry is  $w + 1$ .
    else  $\triangleright$  The same entry  $w$  is used for the whole row.
       $Q_{R^i} := \max\{Q_{R^i}, w\}$   $\triangleright$  The largest entry is  $w$ .
     $F := \max\{F, Q_\ell\}$   $\triangleright$  update  $F$  as  $\ell$  advances
  return  $F$ 

```

---

**Algorithm 3:** Unweighted Maximum Induced Matching, second version

---

```

 $\Delta$  Set  $t_1 := t_2 := \dots := t_{n_U} := n_V + 1$   $\triangleright$  The value  $n_V + 1$  acts like  $\infty$ .
Set  $Q_1 := Q_2 := \dots := Q_{n_U} := 0$ 
 $F := 0$ 
for  $\ell := 1$  to  $n_V$  do
  for all rows  $i \in U$  with  $L^i = \ell$  do  $\triangleright$  Process each interval  $i$  that starts at  $\ell$ 
     $w := F + 1$   $\triangleright$  leftmost entry
     $\Delta$   $\triangleright t_w$  is no longer computed from scratch
    if  $t_w < R^i$  then  $\triangleright$  There are two values  $w$  and  $w + 1$  in this row:
       $\triangleright W_j^i = w$  for  $j = L^i, \dots, t_w$ ,
       $\triangleright W_j^i = w + 1$  for  $j = t_w + 1, \dots, R^i$ .
       $Q_{R^i} := \max\{Q_{R^i}, w + 1\}$   $\triangleright$  The largest entry is  $w + 1$ .
    else  $\triangleright$  The same entry  $w$  is used for the whole row.
       $Q_{R^i} := \max\{Q_{R^i}, w\}$   $\triangleright$  The largest entry is  $w$ .
     $F := \max\{F, Q_\ell\}$   $\triangleright$  update  $F$  as  $\ell$  is incremented
   $\Delta$  for all entries  $W_\ell^{i'}$  in column  $\ell$  do
   $\Delta$   $w := W_\ell^{i'}$ 
   $\Delta$   $t_w := \min\{t_w, R^{i'}\}$ ;
  return  $F$ 

```

---

$W_j^i$  must be  $w + 1$  as soon as  $j > R^{i'}$ . The algorithm determines the *threshold position*  $t_w$  as the smallest right endpoint  $R^{i'}$  under these constraints. Then the entries  $w + 1$  in row  $i$  start at  $j = t_w + 1$  if these entries are still part of the row.

– (IV) We do not maintain the whole array  $P_r$  for each  $r$ , but only its last entry  $P_r[r]$ ; this is sufficient for updating  $F$  and thus for computing the leftmost entries in the rows. We call this value  $Q_r$ .

This leads to Algorithm 2.

**Algorithm 4:** Unweighted Maximum Induced Matching, final version

```

Δ Initialize lists  $\mathcal{T}_1, \dots, \mathcal{T}_{n_V}$  to empty lists
Set  $t_1 := t_2 := \dots := t_{n_U} := n_V + 1$ 
Set  $Q_1 := Q_2 := \dots := Q_{n_U} := 0$ 
 $F := 0$ 
for  $\ell := 1$  to  $n_V$  do
  for all rows  $i \in U$  with  $L^i = \ell$  do ▷ Process each interval  $i$  that starts at  $\ell$ 
     $w := F + 1$  ▷ leftmost entry
    if  $t_w < R^i$  then ▷ There are two values  $w$  and  $w + 1$  in this row:
      ▷  $W_j^i = w$  for  $j = L^i, \dots, t_w$ ,
      ▷  $W_j^i = w + 1$  for  $j = t_w + 1, \dots, R^i$ .
    Δ add  $(w + 1, R^i)$  to the list  $\mathcal{T}_{t_w + 1}$  ▷ don't forget to update  $t_{w + 1}$  when  $\ell$  reaches  $t_w + 1$ 
    Δ add  $(w, R^i)$  to the list  $\mathcal{T}_\ell$  ▷ don't forget to update  $t_w$  when  $\ell$  advances
       $Q_{R^i} := \max\{Q_{R^i}, w + 1\}$ 
    else ▷ The same entry  $w$  is used for the whole row.
    Δ add  $(w, R^i)$  to the list  $\mathcal{T}_\ell$ 
       $Q_{R^i} := \max\{Q_{R^i}, w\}$ 
     $F := \max\{F, Q_\ell\}$  ▷ update  $F$  as  $\ell$  advances
    Δ for all  $(w, r) \in \mathcal{T}_\ell$  do  $t_w := \min\{t_w, r\}$  ▷ perform the necessary updates
  return  $F$ 

```

We will improve Algorithm 2 by maintaining the values  $t_w$  instead of computing them from scratch. We use the fact that the smallest value  $w$  in the row is known, and hence we can associate  $t_w$  with the value  $w$  instead of the row index  $i$ , as is already apparent from our chosen notation. We update  $t_w$  whenever  $\ell$  increases. The details are shown in Algorithm 3. The differences to Algorithm 2 are marked by  $\Delta$ .

This still does not achieve  $O(n)$  running time. The final improvement comes from realizing that it is sufficient to update  $t_w$  when  $W_1^{i'}$  is the leftmost entry  $w$  in row  $i'$ . The time when such an update occurs can be predicted when a row is generated. To this end, we maintain a list  $\mathcal{T}_j$  for  $j = 1, \dots, n_V$  that records the updates that are due when  $\ell$  becomes  $j$ . This final version is Algorithm 4.

The runtime of Algorithm 4 is  $O(n_U + n_V)$ : Processing each interval  $i$  takes constant time and adds at most two pairs to the lists  $\mathcal{T}$ . Thus, processing the lists  $\mathcal{T}$  for updating the  $t_w$  array takes also only  $O(n_U)$  time.

Some simplifications are possible: The addition of  $(w, R^i)$  to the list  $\mathcal{T}_\ell$  in the case of two values can actually be omitted, as it leads to no decrease in  $t_w$ :  $t_w$  is already  $< R^i$ . The algorithm could be further streamlined by observing that at most two consecutive values of  $t_w$  need to be remembered at any time. Again, it is easy to modify the algorithm to return a maximum induced matching in addition to its size.

**Theorem 2** *Given a compact representation, a maximum-cardinality induced matching of a convex bipartite graph can be computed in  $O(n)$  time.* □

## 4 Minimum Chain Covers

For convex bipartite graphs, we have the following important duality result of Yu et al. [27], see also [3]:

**Theorem 3** *In a convex bipartite graph, the size of a maximum-cardinality induced matching equals the number of chain subgraphs of a minimum chain cover.*

Along the lines of this duality relation, we are going to extend Algorithm 4 to obtain a minimum chain cover of a convex bipartite graph  $G = (U, V, E)$ .

Let  $W^*$  be the cardinality of a maximum induced matching of  $G$ . Accordingly, the values  $W_j^i$  cover the range  $\{1, \dots, W^*\}$ . We create  $W^*$  chain subgraphs  $Z_1, \dots, Z_{W^*}$  of  $G$ . The edges  $(i, j)$  with  $W_j^i = w$  will be part of the chain subgraph  $Z_w$ . As already observed in [27], the edges with a fixed value of  $W_j^i$  may contain independent edges and, thus, do not necessarily constitute a chain graph. Yu et al. [27] describe a strategy to extend the edge set for each value of  $W_j^i = w$  to a chain graph  $Z_w$ . Their original implementation runs in time  $O(m^2)$ . Brandstädt et al. [3] give an improved implementation with runtime  $O(n^2)$ . We improve on these previous results, by implementing the extension strategy in  $O(n + m)$  time; and even in  $O(n)$  time if a compact representation of the input graph is given.

The correctness of this approach was already shown in [27], thus establishing Theorem 3. We give a new independent proof.

The following characterization is often used as an alternative definition of chain graphs:

**Lemma 6** *A bipartite graph  $(\bar{U}, \bar{V}, \bar{E})$  is a chain graph if and only if the sets of neighbors  $\bar{V}(i) := \{j \in \bar{V} \mid (i, j) \in \bar{E}\}$  of the vertices  $i \in \bar{U}$  form a chain in the inclusion order. (Equal sets are allowed.) In other words, among any two sets  $\bar{V}(i)$  and  $\bar{V}(i')$ , one must be contained in the other.*

**Proof** This is a direct consequence of the fact that edges  $(i, j)$  and  $(i', j')$  are independent if and only if  $j' \notin \bar{V}(i)$  and  $j \notin \bar{V}(i')$ .  $\square$

The condition that the neighborhoods must form a chain is apparently the reason for calling these graphs *chain graphs*, however, we did not find a reference for this.

We use  $U_w$  to denote the set of rows that contain entries  $W_j^i = w$ . For every row  $i \in U_w$ , we determine the beginning and ending points  $B_w^i, E_w^i$  with this color, that is,  $W_j^i = w \iff B_w^i \leq j \leq E_w^i$ . We extend every such interval  $[B_w^i, E_w^i]$  to the left by choosing a new starting point  $\hat{B}_w^i$  according to the formula

$$\hat{B}_w^i := \min \left( \{B_w^i\} \cup \{B_w^{i'} \mid i' \in U_w, E_w^{i'} < E_w^i\} \right) \quad (9)$$

$$= \min \left( \{B_w^i\} \cup \{\hat{B}_w^{i'} \mid i' \in U_w, E_w^{i'} < E_w^i\} \right) \quad (10)$$

The second expression uses the new values  $\hat{B}$  on the right-hand side. It is easy to see that the two expressions are equivalent: Using (9) for the definition of  $\hat{B}_w^{i'}$ , the

**Algorithm 5:** Creating a chain graph  $\{(i, j) \mid i \in U_w, \hat{B}_w^i \leq j \leq E_w^i\}$  for a given  $w$  in the range  $1 \leq w \leq W^*$

---

```

▷  $U_w := \{i \in U \mid \text{row } i \text{ contains an entry } w\}$ 
▷ Let  $B_w^i$  and  $E_w^i$  such that in row  $i$ , the entries with  $W_j^i = w$  are those with  $B_w^i \leq j \leq E_w^i$ 
Set  $G_1 := G_2 := \dots := G_{W^*} := n_V + 1$  ▷ The value  $n_V + 1$  acts like  $\infty$ 
for  $r := 1$  to  $n_V$  do
    ▷ We maintain the quantities  $G_w \equiv \min\{B_w^i \mid E_w^i < r\}$  for  $w = 1, \dots, W^*$ .
    for all  $(B_w^i, E_w^i, w)$  with  $E_w^i = r$  do
        |  $\hat{B}_w^i := \min\{B_w^i, G_w\}$ 
    for all  $(B_w^i, E_w^i, w)$  with  $E_w^i = r$  do ▷ update  $G_w$  for the increment of  $r$ 
        |  $G_w := \min\{B_w^i, G_w\}$ 

```

---

expression (10) becomes

$$\min \left( \{B_w^i\} \cup \{B_w^{i'} \mid i' \in U_w, E_w^{i'} < E_w^i\} \cup \{B_w^{i''} \mid i'' \in U_w, E_w^{i''} < E_w^i < E_w^{i'}, i' \in U_w\} \right). \tag{11}$$

The third set is contained in the second set, and thus, (11) is equal to  $\hat{B}_w^i$  according to (9).

We construct the chain graph  $Z_w$  as the graph with the extended intervals  $[\hat{B}_w^i, E_w^i]$ . Figure 4 shows an example. It is obvious by construction that these intervals satisfy the conditions of a chain graph: By Lemma 6, we have to show that there are no two intervals  $[\hat{B}_w^i, E_w^i], [\hat{B}_w^{i'}, E_w^{i'}]$  with  $\hat{B}_w^{i'} < \hat{B}_w^i$  and  $E_w^{i'} < E_w^i$ . But if the last condition holds, (10) ensures that  $\hat{B}_w^i \leq \hat{B}_w^{i'}$ . The only thing that could go wrong is that  $\hat{B}_w^i$  becomes too small so that the chain graph is not a subgraph of  $G$ . The following lemma shows that this is not the case.

**Lemma 7**  $\hat{B}_w^i \geq L^i$  for every  $i \in U_w$ .

**Proof** For the sake of contradiction, assume  $\hat{B}_w^i < L^i$ . By (9), there is a row  $i' \in U_w$  such that  $B_w^{i'} < L^i$  and  $E_w^{i'} < E_w^i$ . Setting  $j = E_w^i$  and  $j' = B_w^{i'}$  in the recursion (7), we conclude that  $E_w^i \leq R^{i'}$ , because otherwise, (7) would imply  $w = W_{E_w^i}^i \geq 1 + W_{B_w^{i'}}^{i'} = 1 + w$ . Thus,  $(i', E_w^i)$  is an edge of  $G$ . By Lemma 5,  $W_{E_w^i}^{i'} = w + 1$ . By (7), there is an edge  $(i'', j'')$  with  $W_{j''}^{i''} = w, R^{i''} < E_w^i$  and  $j'' < L^{i'} < L^i$ . Again by (7), such an edge  $(i'', j'')$  would imply that  $W_{E_w^i}^{i''} \geq w + 1$ , a contradiction.  $\square$

Algorithm 5 carries out the computation of (9). It processes the triplets  $(B_w^i, E_w^i, w)$  in increasing order of the endpoints  $E_w^i = r$ . This can be done in linear time, by first sorting the  $O(n_U)$  triples  $(B_w^i, E_w^i, w)$  into  $n_V$  buckets according to the value of  $E_w^i$ . Thus, Algorithm 5 takes linear time  $O(n)$ . By Lemma 6, the result is a chain cover, which by duality is minimum. Each row belongs to at most two chain subgraphs, and thus the chain cover consists of at most  $2n_U$  such row intervals in total. It is

straightforward to extend Algorithm 4 to compute the sets  $U_w$  and the quantities  $B_w^i, E_w^i$ , and thus the cover can be constructed in  $O(n)$  time in compressed form.

**Theorem 4** *Given a compact representation of a convex bipartite graph, a compact representation of a minimum chain cover can be computed in  $O(n)$  time.*  $\square$

Given a compact representation of a minimum chain cover, we can list all the edges of its chain subgraphs in  $O(n + m)$  time since every edge is contained in at most two chain subgraphs. A compact representation of a convex bipartite graph can be computed in  $O(n + m)$  time [2,23,25]. Thus, Algorithm 4 and Algorithm 5 can also be used to obtain:

**Theorem 5** *A minimum chain cover of a convex bipartite graph can be computed in  $O(n + m)$  time.*  $\square$

## 5 Certification of Optimality

An induced matching  $H$  together with a chain cover of the same cardinality provides a *certificate of optimality*, of size  $O(n)$ . As we will establish in the following discussion, it is easy to *check* this certificate for validity in linear time. This is easier than *constructing* the largest induced matching with our algorithm. Thus, it is possible to establish correctness of the result beyond doubt, for each particular instance of the problem, without having to trust the correctness of our algorithms and their implementations, see [18] for a survey about this concept.

It is trivial to check whether the matching  $H$  is contained in the graph. To test whether it forms an induced matching, we sort the edges  $(i, j)$  by  $j$ . This takes  $O(n)$  time with bucket-sort. Then, by Lemma 1, it is sufficient to test consecutive edges for independence, and each such test takes only constant time according to Proposition 1.

To establish the validity of a chain cover  $\{Z_1, \dots, Z_{W^*}\}$ , we need to check that the edges of  $G$  are covered and each  $Z_w$  is a chain subgraph. The chain subgraphs  $Z_w = \{(i, j) \mid i \in U_w, \hat{B}_w^i \leq j \leq E_w^i\}$ , for  $1 \leq w \leq W^*$  are compactly represented by a set of at most  $2n_U$  quadruples  $(w, i, \hat{B}_w^i, E_w^i)$ . The following checking procedure works in linear time for any chain cover as long as it consists of convex bipartite subgraphs. It does not use any special properties of the cover produced by our algorithm.

We sort the quadruples  $(w, \hat{B}_w^i, -E_w^i, i)$  lexicographically. Then it is easy to check the chain graph property using the characterization of Lemma 6: The intervals  $[\hat{B}_w^i, E_w^i]$  that belong to a fixed chain graph  $Z_w$  (these are consecutive in the list) ought to be nested. Since the starting points  $\hat{B}_w^i$  are weakly increasing, this amounts to checking that the endpoints  $E_w^i$  decrease weakly.

To check that the chain graphs are contained in  $G$  and they collectively cover  $G$ , we sort the quadruples  $(i, \hat{B}_w^i, E_w^i, w)$ . The union of the intervals  $[\hat{B}_w^i, E_w^i]$  that are the neighbors of a fixed vertex  $i \in U$  (these are consecutive in the list) can be incrementally formed, and the resulting interval is compared against  $[L^i, R^i]$ . As soon as a gap would form in this union, we can abort the test, since the intervals are sorted by left endpoint and it is then impossible to form a connected interval  $[L^i, R^i]$ .

The required lexicographic sorting operations can be carried out in  $O(n)$  time by repeated bucket-sort (radix-sort).

## 6 Outlook: Duality

The existence of a maximum induced matching and a smallest chain cover with the same size is a manifestation of strong duality between independent sets and clique covers in perfect graphs. We mentioned in the introduction that our maximum induced matching problem is an instance of a maximum independent set problem in the square of a line graph, and the chain cover is a covering by cliques. Yu et al. [27] established that the square of the line graph of a convex bipartite graph is a co-comparability graph. Therefore, it is also a perfect graph. It follows that the linear program for maximizing the size of an induced matching in a convex bipartite graph is totally dual integral, see [22, Corollary 65.2f]. As a corollary of this fact, we recover our strong duality result: the existence of a *primal* optimal solution (maximum induced matching) and a *dual* optimal solution (smallest chain cover) with matching objective function values, see [22, Corollary 65.2d].

This duality relation for perfect graphs extends to the weighted version. Thus, there should also be a weighted chain cover with the same weight as the maximum weight of an induced matching. A weighted chain cover of a weighted graph consists of chain subgraphs together with a positive real weight for each chain subgraph, such that for every edge  $(i, j)$ , the total weight of all chain subgraphs covering the edge  $(i, j)$  is at least the weight  $C_j^i$  of this edge. The weight of a weighted chain cover is the sum of all weights. It is an open problem to extend our primal Algorithm 1 in weighted graphs to a fast combinatorial algorithm for finding minimum-weight chain covers, as Algorithm 5 does for the unweighted version.

**Funding** Open Access funding enabled and organized by Projekt DEAL.

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest whatsoever.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Asdre, K., Nikolopoulos, S.D.: NP-completeness results for some problems on subclasses of bipartite and chordal graphs. *Theor. Comput. Sci.* **381**(1), 248–259 (2007). <https://doi.org/10.1016/j.tcs.2007.05.012>
2. Booth, K.S., Lueker, G.S.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.* **13**(3), 335–379 (1976). [https://doi.org/10.1016/S0022-0000\(76\)80045-1](https://doi.org/10.1016/S0022-0000(76)80045-1)
3. Brandstädt, A., Eschen, E.M., Sritharan, R.: The induced matching and chain subgraph cover problems for convex bipartite graphs. *Theor. Comput. Sci.* **381**(1–3), 260–265 (2007). <https://doi.org/10.1016/j.tcs.2007.04.006>



4. Brandstädt, A., Hoàng, C.T.: Maximum induced matchings for chordal graphs in linear time. *Algorithmica* **52**(4), 440–447 (2008). <https://doi.org/10.1007/s00453-007-9045-2>
5. Cameron, K.: Induced matchings. *Discrete Appl. Math.* **24**(1–3), 97–102 (1989). [https://doi.org/10.1016/0166-218X\(92\)90275-F](https://doi.org/10.1016/0166-218X(92)90275-F)
6. Cameron, K., Sritharan, R., Tang, Y.: Finding a maximum induced matching in weakly chordal graphs. *Discrete Math.* **266**(1–3), 133–142 (2003). [https://doi.org/10.1016/S0012-365X\(02\)00803-8](https://doi.org/10.1016/S0012-365X(02)00803-8)
7. Chang, J.M.: Induced matchings in asteroidal triple-free graphs. *Discrete Appl. Math.* **132**(1–3), 67–78 (2003). [https://doi.org/10.1016/S0166-218X\(03\)00390-1](https://doi.org/10.1016/S0166-218X(03)00390-1)
8. Ding, G.: Covering the edges with consecutive sets. *J. Graph Theory* **15**(5), 559–562 (1991). <https://doi.org/10.1002/jgt.3190150508>
9. Duckworth, W., Manlove, D., Zito, M.: On the approximability of the maximum induced matching problem. *J. Discrete Algorithms* **3**(1), 79–91 (2005). <https://doi.org/10.1016/j.jda.2004.05.001>
10. Gallo, G.: An  $O(n \log n)$  algorithm for the convex bipartite matching problem. *Oper. Res. Lett.* **3**(1), 31–34 (1984). [https://doi.org/10.1016/0167-6377\(84\)90068-3](https://doi.org/10.1016/0167-6377(84)90068-3)
11. Glover, F.: Maximum matching in a convex bipartite graph. *Nav. Res. Logist. Q.* **14**(3), 313–316 (1967). <https://doi.org/10.1002/nav.3800140304>
12. Golombic, M.C., Lewenstein, M.: New results on induced matchings. *Discrete Appl. Math.* **101**(1–3), 157–165 (2000). [https://doi.org/10.1016/S0166-218X\(99\)00194-8](https://doi.org/10.1016/S0166-218X(99)00194-8)
13. Hammer, P.L., Peled, U.N., Sun, X.: Difference graphs. *Discrete Appl. Math.* **28**(1), 35–44 (1990). [https://doi.org/10.1016/0166-218X\(90\)90092-Q](https://doi.org/10.1016/0166-218X(90)90092-Q)
14. Hung, R.: Linear-time algorithm for the paired-domination problem in convex bipartite graphs. *Theory Comput. Syst.* **50**(4), 721–738 (2012). <https://doi.org/10.1007/s00224-011-9378-8>
15. Katriel, I.: Matchings in node-weighted convex bipartite graphs. *INFORMS J. Comput.* **20**(2), 205–211 (2008). <https://doi.org/10.1287/ijoc.1070.0232>
16. Kobler, D., Rotics, U.: Finding maximum induced matchings in subclasses of claw-free and  $P_5$ -free graphs, and in graphs with matching and induced matching of equal maximum size. *Algorithmica* **37**(4), 327–346 (2003). <https://doi.org/10.1007/s00453-003-1035-4>
17. Lozin, V.V.: On maximum induced matchings in bipartite graphs. *Inf. Process. Lett.* **81**(1), 7–11 (2002). [https://doi.org/10.1016/S0020-0190\(01\)00185-5](https://doi.org/10.1016/S0020-0190(01)00185-5)
18. McConnell, R.M., Mehlhorn, K., Näher, S., Schweitzer, P.: Certifying algorithms. *Comput. Sci. Rev.* **5**(2), 119–161 (2011). <https://doi.org/10.1016/j.cosrev.2010.09.009>
19. Müller, H.: Hamiltonian circuits in chordal bipartite graphs. *Discrete Math.* **156**(1–3), 291–298 (1996). [https://doi.org/10.1016/0012-365X\(95\)00057-4](https://doi.org/10.1016/0012-365X(95)00057-4)
20. Panda, B.S., Pandey, A., Chaudhary, J., Dane, P., Kashyap, M.: Maximum weight induced matching in some subclasses of bipartite graphs. *J. Combin. Optim.* **40**, 713–732 (2020). <https://doi.org/10.1007/s10878-020-00611-2>
21. Pandey, A., Panda, B.S., Dane, P., Kashyap, M.: Induced matching in some subclasses of bipartite graphs. In: Gaur, D.R., Narayanaswamy, N.S. (eds.) *Algorithms and Discrete Applied Mathematics—Third International Conference, CALDAM 2017, Sancoale, Goa, India, February 16–18, 2017, Proceedings, Lecture Notes in Computer Science*, vol. 10156, pp. 308–319. Springer (2017). [https://doi.org/10.1007/978-3-319-53007-9\\_27](https://doi.org/10.1007/978-3-319-53007-9_27)
22. Schrijver, A.: *Combinatorial Optimization—Polyhedra and Efficiency*, Vol. B, Algorithms and Combinatorics, vol. 24. Springer (2003)
23. Soares, J., Stefanos, M.A.: Algorithms for maximum independent set in convex bipartite graphs. *Algorithmica* **53**(1), 35–49 (2009). <https://doi.org/10.1007/s00453-007-9006-9>
24. Spinrad, J.P.: *Efficient Graph Representations*. American Mathematical Society, Providence (2003)
25. Steiner, G., Yeomans, J.: A linear time algorithm for maximum matchings in convex, bipartite graphs. *Comput. Math. Appl.* **31**(12), 91–96 (1996). [https://doi.org/10.1016/0898-1221\(96\)00079-X](https://doi.org/10.1016/0898-1221(96)00079-X)
26. Stockmeyer, L.J., Vazirani, V.V.: NP-completeness of some generalizations of the maximum matching problem. *Inf. Process. Lett.* **15**(1), 14–19 (1982). [https://doi.org/10.1016/0020-0190\(82\)90077-1](https://doi.org/10.1016/0020-0190(82)90077-1)
27. Yu, C.W., Chen, G.H., Ma, T.H.: On the complexity of the  $k$ -chain subgraph cover problem. *Theor. Comput. Sci.* **205**(1–2), 85–98 (1998). [https://doi.org/10.1016/S0304-3975\(97\)00036-4](https://doi.org/10.1016/S0304-3975(97)00036-4)