Theory Comput Syst (2012) 50:721–738
DOI 10.1007/s00224-011-9378-8

Linear-Time Algorithm for the Paired-Domination Problem in Convex Bipartite Graphs

Ruo-Wei Hung

Published online: 3 December 2011

Abstract A bipartite graph $G = (U, W, E)$ with vertex set $V = U \cup W$ is *convex* if there exists an ordering of the vertices of W such that for each $u \in U$, the neighbors of u are consecutive in W. A *compact representation* of a convex bipartite graph for specifying such an ordering can be computed in $O(|V| + |E|)$ time. The paired-domination problem on bipartite graphs has been shown to be NP-complete. The complexity of the paired-domination problem on convex bipartite graphs has remained unknown. In this paper, we present an $O(|V|)$ time algorithm to solve the paired-domination problem on convex bipartite graphs given a compact representation. As a byproduct, we show that our algorithm can be directly applied to solve the total domination problem on convex bipartite graphs in the same time bound.

Keywords Graph algorithms · Linear-time algorithms · Paired-domination · Total domination · Convex bipartite graphs

1 Introduction

All graphs considered in this paper are finite and undirected, without loops or multiple edges. For any two sets X and Y, let $X - Y$ denote the set of elements of X that are not in Y. Let $G = (V, E)$ be a graph with vertex set V and edge set E. Throughout this paper, let n and m denote the numbers of vertices and edges of graph G, respectively. Let v be a vertex of V and let D be a subset of V. The *open neighborhood* $N(v)$

A preliminary version of this paper has appeared in: Proceedings of the International MultiConference of Engineers and Computer Scientists 2010 (IMECS'2010), Hong Kong, vol. I, pp. 365–369 (2010) [24].

R.-W. Hung (✉)
Department of Computer Science and Information Engineering, Chaoyang University of Technology, Wufeng, Taichung 41349, Taiwan, ROC
e-mail: rwhung@cyut.edu.tw

of the vertex v consists of the set of vertices adjacent to v, that is, $N(v) = \{w \in V | (v, w) \in E\}$, and the *closed neighborhood* of v is $N[v] = N(v) \cup \{v\}$. The *open neighborhood* $N(D)$ is defined to be $\cup_{v \in D} N(v)$, and the *closed neighborhood* of D is $N[D] = N(D) \cup D$. The set D *dominates* vertex v if either $v \in D$ or $N(v) \cap D \neq \emptyset$. If D dominates all vertices in a subset S of V, then we say that D *dominates* S. The set D is called a *dominating set* of G if and only if D dominates V, that is, every vertex in $V - D$ is adjacent to a vertex in D. The *domination number* $\gamma(G)$ equals the minimum cardinality of a dominating set of G. A dominating set D is called *total* if $N(v) \cap D \neq \emptyset$ for all $v \in D$, that is, the subgraph of G induced by D contains no isolated vertex. The minimum cardinality of a total dominating set of G is called the *total domination number* $\gamma_t(G)$. A *minimum dominating set* of G is a dominating set with cardinality $\gamma(G)$. A *minimum total dominating set* of G is a total dominating set with cardinality $\gamma_t(G)$. The *domination problem* is to find a minimum dominating set of G. The *total domination problem* is defined analogously. These two problems are known to be NP-complete for general graphs.

Variations of the domination problem seek to find a minimum dominating set with some additional properties, e.g., to be independent or to induce a acyclic graph. These problems arise in a number of distributed network applications, where the problem is to locate the smallest number of centers in networks such that every vertex is nearby at least one center. Domination and its variations in graphs have been thoroughly studied, and the literature on this subject has been surveyed and detailed in two books [21, 22]. In this paper, we will study a variant of the domination problem, namely the *paired-domination problem*.

A set $S \subseteq V$ is a *paired-dominating set* of G if S dominates V and the subgraph $G[S]$ induced by S contains a perfect matching. That is, a paired-dominating set S with matching M is a dominating set $S = \{v_1, v_2, \ldots, v_{2k-1}, v_{2k}\}$ with independent edge set $M = \{e_1, e_2, \ldots, e_k\}$ where each edge e_i is incident to two vertices of S, that is, M is a perfect matching in $G[S]$. Two vertices joined by an edge of M are said to be *paired*. A vertex v is called a paired-vertex of a matching if there is an edge in it incident to v. Every graph without isolated vertices has a paired-dominating set, since the paired-vertices of any maximal matching form such a set [20]. The *paired-domination number* $\gamma_p(G)$ of a graph G equals the minimum cardinality of a paired-dominating set of G. A *minimum paired-dominating set* of G is a paired-dominating set with cardinality $\gamma_p(G)$. The *paired-domination problem* is to find a minimum paired-dominating set of G. For example, for the three-dimensional hypercube graph Q_3 shown in Fig. 1, $S = \{v_1, v_5, v_4, v_8\}$ is a paired-dominating set of Q_3 since S is a dominating set and the subgraph induced by S contains a perfect matching $M = \{(v_1, v_5), (v_4, v_8)\}$ where v_1 and v_5 (v_4 and v_8) form a pair, and $\gamma_p(Q_3) = 4$.

Paired-domination was introduced by Haynes and Slater [20] with the following application in mind. If, in a graph G, we consider each vertex as the possible location for a guard capable of protecting every vertex adjacent to it, then "domination" requires every vertex to be protected. In paired-domination, each guard is assigned another adjacent guard, and they are designed to provide a backup for each other. The problem of determining the paired-domination number $\gamma_p(G)$ of an arbitrary graph G has been known to be NP-complete [20]. The paired-domination problem is still NP-complete in some special classes of graphs such as bipartite graphs, chordal

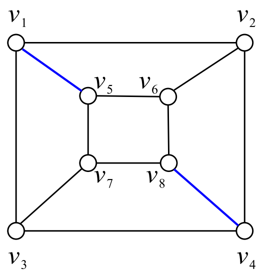


Fig. 1 The three-dimensional hypercube graph Q_3

graphs, and split graphs [12]. However, the problem admits polynomial-time algorithms when the input is restricted to some special classes of graphs, including trees [33], circular-arc graphs [13], permutation graphs [14, 27], strongly chordal graphs [11], block graphs and interval graphs [12].

Let $G = (U, W, E)$ represent an undirected, bipartite graph with vertex set $U \cup W$, where U and W is a partition of the vertices and E is the edge set in which each edge (u, w) is such that $u \in U$ and $w \in W$. The bipartite graph G is called *convex* if the vertices in W can be ordered in a way that for each $u \in U$, the neighbors of u are consecutive in W. This ordering can be obtained in a preprocessing step by an $O(n + m)$ time algorithm [4]. A *compact representation* for specifying such an ordering can be computed in $O(n + m)$ time [34, 36]. Convex bipartite graphs form a subclass of bipartite graphs and are a superclass of bipartite permutation graphs and biconvex graphs [7]. They were introduced by Glover [19], motivated by some industrial applications. Since then several problems have been studied in this kind of graphs. In the following, we give a brief survey on convex bipartite graphs. Asdre and Nikolopoulos proved that the harmonious coloring problem, the pair-complete coloring problem, and the k-path partition problem on convex bipartite graphs are NP-complete [2]. However, several problems on them have been shown to be polynomial time solvable. The problem of finding a maximum independent set in convex bipartite graphs given a compact representation can be solved in $O(n)$ time [29, 34]. The Hamiltonian cycle problem on them is $O(n^2)$ time solvable [30]. Liang and Chang solved the feedback vertex set problem on weighted convex bipartite graphs in $O(n^2m)$ time [28]. Yang showed that the link-orientation problem on weighted convex bipartite graphs can be solved in polynomial time [37]. Yu et al. solved the k-chain subgraph cover problem on convex bipartite graphs in $O(m^2)$ time [39]. Brandstädt et al. improved the result in [39] to obtain an $O(n^2)$ time algorithm for the k-chain subgraph cover problem [8]. Katriel considered the problem of finding a U-perfect matching of maximum weight in a weighted convex bipartite graph, and solved it in $O(m + n \log n)$ time [25]. Brodal et al. considered the problem of maintaining a maximum matching in a convex bipartite graph under a set of update operations which includes insertion and deletions of vertices and edges [6]. They developed a data structure to implement a set of update operations in $O(\log^2 n)$ amortized time per operation. Park et al. designed a boolean circuit to find a maximum matching of a convex bipartite graph [32]. Recently, Nussbaum et al. computed a maximum edge biclique of a convex bipartite graph in $O(n \log^3 n \log\log n)$ time [31]. The maximum matching problem is the most frequently studied problem in the literature for convex bipartite graphs.

Matchings in convex bipartite graphs correspond to the problem of scheduling unit length tasks on a single disjunctive resource, given a release time and a deadline for every task [25]. Glover first considered the maximum matching problem in 1967 and provided an $O(m)$ time algorithm for determining the maximum matching of a convex bipartite graph [19]. This algorithm progresses iteratively through each vertex of W in nondecreasing order and, of all the unmatched vertices of U incident to the vertex consideration, matches it the one which is selected by a greedy strategy. Subsequently, Gallo used a special data structure based on a binary heap to design an $O(n \log n)$ time algorithm for finding a maximum matching [18]. Now, finding a maximum matching of a convex bipartite graph can be done in $O(n)$ time [36]. For parallel algorithms on convex bipartite graphs, we refer readers to [5, 9, 17].

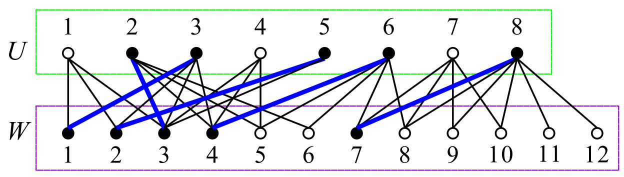
The paired-domination problem on bipartite graphs has been shown to be NP-complete [12]. The complexity of the paired-domination problem in convex bipartite graphs is still unknown. In this paper, we present the first $O(n)$ time algorithm to solve the paired-domination problem on convex bipartite graphs given a compact representation. Further, we show that our algorithm can be applied to solve the total domination problem on convex bipartite graphs in the same time bound. The paired-domination problem in convex bipartite graphs has the following application. Consider a system represented by a convex bipartite graph. The problem of placing monitoring devices in the system such that every site in it (including the monitoring devices themselves) is adjacent to a monitor and every monitor is paired with a backup monitor, can be modelled by paired-domination in convex bipartite graphs.

Previous related works are summarized below. Damaschke et al. solved the domination problem for convex bipartite graphs in $O(n^3)$ time [16]. To the best of our knowledge, the best algorithms for the domination and independent domination problems on convex bipartite graphs run in $O(n^2)$ time [3]. Yen solved the bottleneck independent domination problem on convex bipartite graphs in $O(m)$ time [38]. Srinivasan et al. solved the edge domination problem on bipartite permutation graphs in $O(n^2 + mn^{1.5})$ time [35]. To the best of our knowledge, the complexity status of the edge domination problem on convex bipartite graphs is still unknown. Recently, Korpelainen proved that the efficient edge domination problem on convex bipartite graphs is polynomial solvable [26].

The rest of this paper is organized as follows. In Sect. 2, we review some properties of convex bipartite graphs and define some basic notations used in this paper. Section 3 introduces a clustering procedure used in our algorithm. In Sect. 4, we present an $O(n)$ time algorithm for solving the paired-domination problem given a compact representation of a convex bipartite graph. Section 5 shows that our algorithm can be directly applied to solve the total domination problem on convex bipartite graphs given a compact representation in $O(n)$ time. Finally, some concluding remarks and future work are given in Sect. 6.

2 Preliminaries

Let $G = (U, W, E)$ be a bipartite graph, where U and W define the bipartition of the vertices, and E is the edge set in the form (u, w) such that $u \in U$ and $w \in W$.



(a)

U	*begin*	*end*
1	1	3
2	3	6
3	1	4
4	3	5
5	2	3
6	4	8
7	7	10
8	7	12

(b)

Fig. 2 (**a**) A convex bipartite graph and a paired-dominating set on it (*filled circles together with bold edges*), and (**b**) a compact representation of (**a**)

The graph G is called *convex* if the vertices in W can be ordered in such a way that for each $u \in U$, the neighbors of u are consecutive in W. For convenience, we consider that $U = \{1, 2, \dots, |U|\}$ and $W = \{1, 2, \dots, |W|\}$, and that the vertices in W are given according to the ordering mentioned above. This ordering can be obtained in a preprocessing step by an $O(n + m)$ time algorithm [4]. We say that a vertex $w' \in W$ is smaller (larger) than a vertex $w'' \in W$ if the integer representing w' is smaller (larger) than the integer representing w''. That is, we will represent vertices of W by integers from 1 to $|W|$. A convex bipartite graph has a *compact representation* by a set of $|U|$ triples of the form $(i, \mathit{begin}(i), \mathit{end}(i))$, where i is a vertex in U, $\mathit{begin}(i)$ and $\mathit{end}(i)$ are the smallest and largest vertices, respectively, in the interval of vertices (i.e., a sequence of consecutively numbered vertices) of W connected to i. For example, Fig. 2(a) shows a convex bipartite graph and Fig. 2(b) depicts its compact representation.

Theorem 1 [4, 34, 36] *Given a convex bipartite graph, its compact representation can be computed in $O(n + m)$ time.*

Throughout the remainder of the paper, we assume that the input of the algorithm is a compact representation of a convex bipartite graph. A paired-dominating set of a convex bipartite graph $G = (U, W, E)$ contains a perfect matching M such that for $(u, w) \in M$, $u \in U$ and $w \in W$. For instance, Fig. 2(a) shows a paired-dominating set of a convex bipartite graph, where filled circles indicate the vertices in the paired-dominating set and bold edges form the perfect matching.

Hereafter, let $G = (U, W, E)$ be a convex bipartite graph with a compact representation. We denote by $[i, j]$ the set of consecutive integers $\{i, i + 1, \dots, j\}$. The vertices of U (resp. W) are represented by integers in $[1, |U|]$ (resp. $[1, |W|]$). Thus, $U = [1, |U|]$ and $W = [1, |W|]$. We call $[i, j]$ an (integer) interval starting from i and ending at j. Further, we call i and j the *left* endpoint and *right* endpoint of interval $I = [i, j]$, denoted by $l(I)$ and $r(I)$, respectively. For $p < i \leqslant \ell \leqslant j$, we say that ℓ *dominates* interval $[i, j]$, interval $[i, j]$ *covers* ℓ, and interval $[i, j]$ is to the right of p. Note that p and ℓ may be the integers representing vertices. Two intervals are called *disjoint* if the intersection of them is empty. In addition, we also let U denote an array representing G in a compact representation. Each element of the array $U[1..|U|]$ has the fields *begin* and *end*. The triple $(i, \mathit{begin}(i), \mathit{end}(i))$ of the compact representation of G is represented here by $(i, U[i].\mathit{begin}, U[i].\mathit{end})$. We may assume that the input

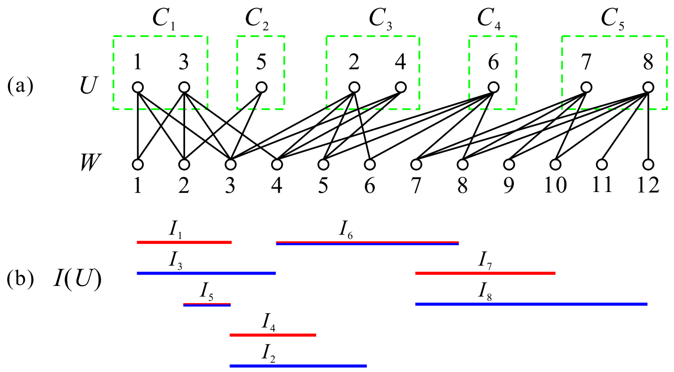


Fig. 3 (**a**) Sorted clusters of U, and (**b**) interval representation $I(U)$ of neighbors of vertices in U for the convex bipartite graph shown in Fig. 2(a)

convex bipartite graph has no isolated vertices since isolated vertices can be easily determined. By the definition of convex bipartite graphs, the neighbors of a vertex u in U can be represented as an interval $I_u = [u.begin, u.end]$, called the *neighbor interval* of u. Then, the neighbors of vertices of U can be represented by a set $I(U)$ of intervals. We call $I(U)$ the *interval representation* of neighbors of vertices in U. For example, Fig. 3(b) shows the interval representation $I(U)$ for the convex bipartite graph shown in Fig. 2(a).

3 The Clustering Process

We first observe one property of a convex bipartite graph $G = (U, W, E)$. For two vertices x, y in U, either $x.begin = y.begin$ or not. If $x.begin = y.begin$, then the smallest neighbors of x and y are the same vertex in W. Thus, we define clusters of U as follows.

Definition 1 Two vertices x and y of U are said to be in the same *cluster* if $x.begin = y.begin$, i.e., their smallest neighbors are the same vertex of W.

To obtain an $O(n)$ time algorithm, we will sort the clusters of U in increasing manner according to their smallest neighbors. We then define the *sorted clusters* of U as follows.

Definition 2 The sequential clusters $C_1, C_2, \dots, C_k$ of U are called *sorted clusters* if $a.begin < b.begin$ for $a \in C_i$, $b \in C_j$, and $i < j$. Cluster C_i is said to be smaller than cluster C_{i+1} for $1 \leqslant i \leqslant k-1$. Moreover, for two vertices u_i and u_j in two distinct clusters, we say that $u_i < u_j$ if $u_i.begin < u_j.begin$.

For example, Fig. 3(a) shows the sorted clusters for the convex bipartite graph shown in Fig. 2(a).

In the following, we will partition U into k disjoint sorted clusters $C_1, C_2, \dots, C_k$. We call it the *clustering process*. It is easy to see that the clustering process runs in $O(n \log n)$ time if it is implemented by a general purpose sorting algorithm. Since it is too time-consuming, we do not use a general purpose sorting algorithm to implement the clustering process. Given an array $U[1..|U|]$ representing G in a compact representation, each element of U forms a triple $(i, U[i].begin, U[i].end)$ and $U[i].begin$ is an integer in the range from 1 to $|W|$. Thus, we can use the *Counting Sort* algorithm in [15] to implement the clustering process in $O(|U| + |W|)$ time.

For readers' convenience, we introduce how to apply Counting Sort algorithm to compute the sorted clusters as follows. We require three other arrays: The clustered array U^c holds the sorted clusters, where each element of U^c consists of a triple $(u, u.begin, u.end)$ for $u \in U$, and two auxiliary arrays, namely *beginCounts*$[1..\omega]$ and *startingLoc*$[1..\omega]$, provide temporary working storage, where ω is the largest integer in $\bigcup_{1 \leqslant i \leqslant |U|} U[i].begin$ and is bounded by $|W|$. The clustering procedure is sketched as follows. It first determines the number of elements in each *begin* field of the array U and stores them in the auxiliary array *beginCounts*. For each integer ℓ, $1 \leqslant \ell \leqslant \omega$, the item *beginCounts*$[\ell]$ stores the number of elements whose $U[i].begin$ equals ℓ. Then, it uses the information in array *beginCounts* to determine the starting location of each cluster in U^c and stores them in array *startingLoc*. Initially, the item *startingLoc*$[\ell]$ stores the starting location of cluster C_i in the clustered array U^c, where the *begin* field of each element in C_i equals ℓ, i.e., $U[j].begin = \ell$ for $j \in C_i$. Let ℓ' be the largest integer in $[1, \ell - 1]$ such that *beginCounts*$[\ell'] \neq 0$. Then, the starting location, *startingLoc*$[\ell]$, is *beginCounts*$[\ell']$ + *startingLoc*$[\ell']$. By using these two auxiliary arrays, the procedure can move the elements in the original array U one by one into their correct location in the clustered array U^c. During the moving process, the contents of array *startingLoc* will be modified. Suppose that it is about to move element i of U with triple $(i, U[i].begin, U[i].end)$. It first finds the location *startingLoc*$[U[i].begin]$ of triple $(i, U[i].begin, U[i].end)$ from auxiliary array *startingLoc*. Then, it moves this triple to the *startingLoc*$[U[i].begin]$th location of U^c. Finally, it lets *startingLoc*$[U[i].begin]$ = *startingLoc*$[U[i].begin] + 1$. After completing the above moving process, it outputs U^c as the clustered array for storing the sorting clusters.

For example, given the compact representation U of a convex bipartite graph shown in Fig. 2(b), Fig. 4(a) shows the initial auxiliary arrays, *beginCounts* and *startingLoc*, Fig. 4(b) reveals the contents of the final auxiliary arrays after moving all elements of U into U^c, and Fig. 4(c) shows the clustered array U^c. Figure 4(c) also depicts the sorted clusters of the graph. Since every element of U is processed once and the length ω of each auxiliary array is at most $|W|$, the above clustering process runs in $O(|U| + |W|)$ time. Thus, we have the following lemma.

Lemma 1 *The sorted clusters of a convex bipartite graph $G = (U, W, E)$ can be computed in $O(|U| + |W|)$ time.*

From now on, it is assumed that the clustering process has been done, i.e., sorted clusters of U and clustered array U^c are given. The number of clusters gives us an upper bound of $\gamma_p(G)$ as follows.

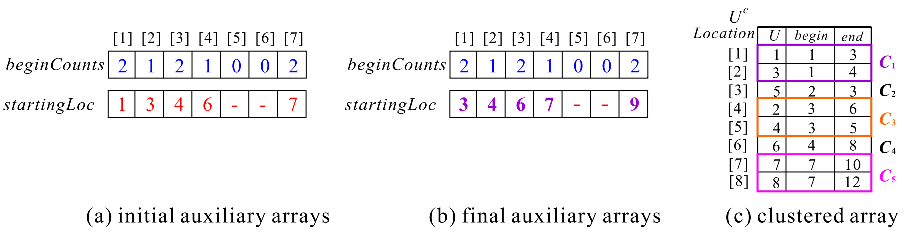


Fig. 4 (**a**) The initial auxiliary arrays, *beginCounts* and *startingLoc*, (**b**) the final auxiliary arrays after moving every element of U into U^c, and (**c**) the clustered array U^c, where the input compact representation U is shown in Fig. 2

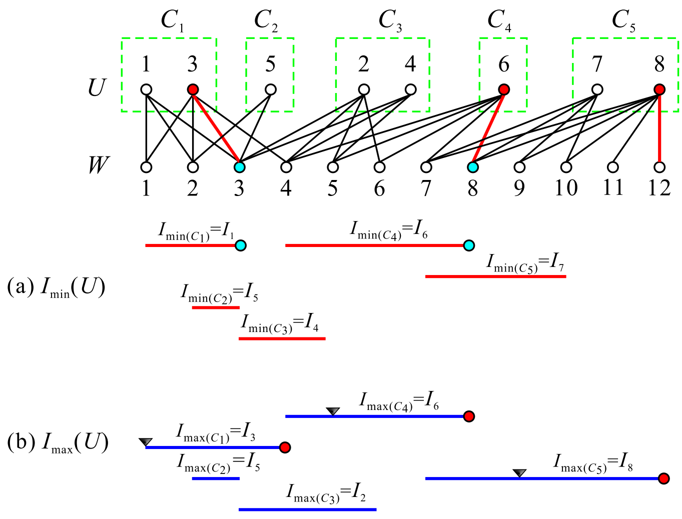


Fig. 5 (**a**) $I_{\min}(U)$, and (**b**) $I_{\max}(U)$ for the convex bipartite graph shown in Fig. 3

Lemma 2 *Let $G = (U, W, E)$ be a convex bipartite graph without isolated vertices, and let U be partitioned into k sorted clusters $C_1, C_2, \cdots, C_k$. Then, $2k \geqslant \gamma_p(G)$.*

Proof Let u_i be a vertex in C_i, $1 \leqslant i \leqslant k$, such that $u'.end \leqslant u_i.end$ for all $u' \in C_i$, and let $w_i \in W$ such that $w_i = u_i.begin$. By pairing u_i with w_i for $1 \leqslant i \leqslant k$, we obtain a paired-dominating set of size $2k$. Thus, $2k \geqslant \gamma_p(G)$. □

Let C_i be a cluster of U. Define $\min(C_i)$ and $\max(C_i)$ to be two vertices in C_i such that $\min(C_i).end \leqslant u'.end \leqslant \max(C_i).end$ for all $u' \in C_i$. Further, $I_{\min(C_i)} = [\min(C_i).begin, \min(C_i).end]$ and $I_{\max(C_i)} = [\max(C_i).begin, \max(C_i).end]$. We can see that for $u' \in C_i$, $I_{\min(C_i)} \subseteq I_{u'} \subseteq I_{\max(C_i)}$. In addition, every vertex of C_i is adjacent to all vertices of $I_{\min(C_i)}$ in W. The vertex of $I_{\min(C_i)}$ in W is called a *common neighbor* of vertices of C_i. For example, let C_1 be a cluster of U shown in Fig. 3. Then, $\min(C_1) = 1$, $I_{\min(C_1)} = [1, 3]$, $\max(C_1) = 3$, and $I_{\max(C_1)} = [1, 4]$. Let $C_1, C_2, \ldots, C_k$ be the disjoint clusters of U. We define $I_{\min}(U) = \{I_{\min(C_i)} | 1 \leqslant i \leqslant k\}$ and $I_{\max}(U) = \{I_{\max(C_i)} | 1 \leqslant i \leqslant k\}$. For example, Fig. 5 shows $I_{\min}(U)$ and

$A_{\min}$	U	*begin*	*end*	
[1]	1	1	3	$I_{\min(C_1)} = I_1$
[2]	5	2	3	$I_{\min(C_2)} = I_5$
[3]	4	3	5	$I_{\min(C_3)} = I_4$
[4]	6	4	8	$I_{\min(C_4)} = I_6$
[5]	7	7	10	$I_{\min(C_5)} = I_7$

(a)

$A_{\max}$	U	*begin*	*end*	
[1]	3	1	4	$I_{\max(C_1)} = I_3$
[2]	5	2	3	$I_{\max(C_2)} = I_5$
[3]	2	3	6	$I_{\max(C_3)} = I_2$
[4]	6	4	8	$I_{\max(C_4)} = I_6$
[5]	8	7	12	$I_{\max(C_5)} = I_8$

(b)

Fig. 6 (**a**) The representing sorted array $A_{\min}$ of $I_{\min}(U)$, and (**b**) the representing sorted array $A_{\max}$ of $I_{\max}(U)$, where they are constructed from Fig. 4(c), and $I_{\min}(U)$ and $I_{\max}(U)$ are shown in Fig. 5

$I_{\max}(U)$ for the convex bipartite graph shown in Fig. 3. By visiting every vertex of a cluster C_i once, $I_{\min(C_i)}$ and $I_{\max(C_i)}$ can be easily found. In addition, by visiting every element of clustered array U^c once, we can easily construct the representing sorted arrays $A_{\min}$ and $A_{\max}$ of $I_{\min}(U)$ and $I_{\max}(U)$, respectively, in $O(|U|)$ time. For instance, Fig. 6 shows the representing sorted arrays of $I_{\min}(U)$ and $I_{\max}(U)$ shown in Fig. 5. Thus, we have the following lemma.

Lemma 3 *Let $G = (U, W, E)$ be a convex bipartite graph without isolated vertices, and let U be partitioned into k sorted clusters $C_1, C_2, \ldots, C_k$. Then, $I_{\min}(U)$ and $I_{\max}(U)$ together with their representing sorted arrays can be constructed in $O(|U|)$ time.*

4 The Paired-Domination Problem in Convex Bipartite Graphs

In this section, we will present an $O(n)$ time algorithm to solve the paired-domination problem on a convex bipartite graph $G = (U, W, E)$ given a compact representation. Let $\hat{W}$ and $\hat{U}$ be subsets of W and U, respectively. Recall that the set $\hat{W}$ (resp. $\hat{U}$) dominates U (resp. W) if every vertex of U (resp. W) is adjacent to at least one vertex of $\hat{W}$ (resp. $\hat{U}$). Let $C_1, C_2, \ldots, C_k$ be the disjoint sorted clusters of U. By Lemma 2, $2k \geqslant \gamma_p(G)$. In the following, we will obtain a lower bound of $\gamma_p(G)$. Our basic idea is described as follows. We first find a minimum cardinality subset $\tilde{W}$ of W and a minimum cardinality subset $\tilde{U}$ of U such that U and W are dominated by $\tilde{W}$ and $\tilde{U}$, respectively. By the definition of a convex bipartite graph, it is not hard to verify the following lemma.

Lemma 4 *Let $\tilde{W}$ and $\tilde{U}$ be the minimum cardinality subsets of W and U, respectively, such that $\tilde{W}$ dominates U and $\tilde{U}$ dominates W. Then, $\gamma_p(G) \geqslant 2 \times \max\{|\tilde{W}|, |\tilde{U}|\}$.*

After computing such two sets $\tilde{W}$ and $\tilde{U}$, we construct a paired-dominating set of G with cardinality not less than $2 \cdot \max\{|\tilde{W}|, |\tilde{U}|\}$. By Lemma 4, the constructed paired-dominating set is a minimum paired-dominating set of G if its size equals $2 \cdot \max\{|\tilde{W}|, |\tilde{U}|\}$. We then show that the constructed paired-dominating set is a minimum paired-dominating set of G if its size is larger than $2 \cdot \max\{|\tilde{W}|, |\tilde{U}|\}$. In the following, we show how to construct two such sets $\tilde{W}$ and $\tilde{U}$.

We first construct a minimum cardinality subset $\tilde{W}$ of W such that $\tilde{W}$ dominates U. Observe that every vertex not in $I_{\min}(U)$ is not in $\tilde{W}$. We give the following lemma to verify this observation.

Lemma 5 *Let $\tilde{W}$ be a minimum cardinality subset of W such that $\tilde{W}$ dominates U. Then, $\tilde{W} \subseteq I_{\min}(U)$.*

Proof Assume by contradiction that $\tilde{W} - I_{\min}(U) \neq \emptyset$. Let $w \in \tilde{W} - I_{\min}(U)$ such that w is adjacent to one vertex of cluster C_i in U. Then, $w \in I_{\max(C_i)} - I_{\min(C_i)}$, i.e., $\min(C_i).end < w \leqslant \max(C_i).end$. Consider that $I_{\min(C_i)} \cap \tilde{W} = \emptyset$. Then, $\min(C_i) \in U$ is not adjacent to any vertex of $\tilde{W}$, and, hence, $\tilde{W}$ does not dominate U, a contradiction. On the other hand, consider that $I_{\min(C_i)} \cap \tilde{W} \neq \emptyset$. Then, every vertex of C_i is adjacent to one vertex of $I_{\min(C_i)} \cap \tilde{W}$. Thus, $\tilde{W} - \{w\}$ also dominates U. It contradicts that $\tilde{W}$ is a minimum cardinality subset of W dominating U. In any case, a contradiction occurs. Thus, $\tilde{W} - I_{\min}(U) = \emptyset$, i.e., $\tilde{W} \subseteq I_{\min}(U)$. □

By the above lemma, we can only consider the vertices in $I_{\min(C_i)}$, $1 \leqslant i \leqslant k$, for computing $\tilde{W}$. Then, we are given by $I_{\min}(U)$. Note that the vertices of W and U are represented by integers. In the following, we will use integers and vertices interchangeably. Then, the problem of finding a minimum cardinality subset of W dominating U is equivalent to seek a minimum set of integers such that they together dominate intervals of $I_{\min}(U)$. We introduce Procedure **GD-W** to compute such a set $\tilde{W}$ of integers of W. This procedure is inspired by the greedy algorithms on interval graphs [1, 10, 23], but the basic approach is different from theirs. Given a set $I_{\min}(U)$ of intervals, Procedure **GD-W** uses a greedy principle to obtain a subset $\tilde{W}$ of W as follows. Initially, let $\tilde{W} = \emptyset$, $I_{\min} = I_{\min}(U)$, and let $s(I_{\min})$ denote the interval in $I_{\min}$ with the smallest right endpoint. Let w be the right endpoint of $s(I_{\min})$ and let I^w be the set of intervals dominated by w in $I_{\min}$. Let $\tilde{W} = \tilde{W} \cup \{w\}$ and let $I_{\min} = I_{\min} - I^w$. Repeat the above process until $I_{\min} = \emptyset$. Then it outputs $\tilde{W}$. For example, given a set $I_{\min}(U)$ of intervals shown in Fig. 5(a), Procedure **GD-W** outputs $\tilde{W} = \{3, 8\}$.

Before we show how to verify the optimality of Procedure **GD-W**, let us observe the behavior of Procedure **GD-W** and the integers obtained by it from $I_{\min}$. It is easy to see that every interval in $I_{\min} - \{s(I_{\min})\}$ is either dominated by w which is the right endpoint of $s(I_{\min})$, or to the right of w. We see that Procedure **GD-W** maintains the following invariant while growing set $\tilde{W} = \{w_1, w_2, \ldots, w_p\}$ with $w_1 < w_2 < \cdots < w_p$ to be computed:

Every interval $x \in I_{\min}$ dominated by none of $\tilde{W}$ is to the right of w_p, and the intervals with right endpoints w_i, $1 \leqslant i \leqslant p$, are disjoint.

Initially, $\tilde{W} = \{w_1\}$, w_1 is the right endpoint of interval $s(I_{\min}(U))$, and the invariant holds trivially. Assume now $\tilde{W} = \{w_1, w_2, \ldots, w_h\}$, where $w_1 < w_2 < \cdots < w_h$ and $h \geqslant 1$, and the invariant holds. In this time, every interval of $I_{\min}$ is dominated by none of $\tilde{W}$. Let w be the right endpoint of $s(I_{\min})$ and let I^w be the set of intervals dominated by w in $I_{\min}$. By the variant, $s(I_{\min})$ is to the right of w_h. In addition, every interval $x \in I_{\min} - I^w$ dominated by none of $\tilde{W} \cup \{w\}$ is to the right of w. By induction then, the invariant maintained by Procedure **GD-W** holds true.

Since every interval needs to be dominated by at least one integer, the following proposition can be easily verified by the pigeonhole principle.

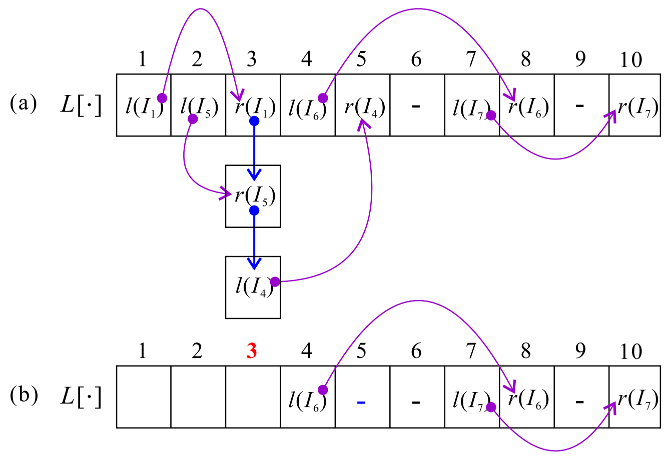
Proposition 1 *Let I be a set of disjoint intervals, and let D be a set of integers such that every interval of I is dominated by at least one integer of D. Then, $|D| \geqslant |I|$.*

Let $\tilde{W}$ be the output by Procedure **GD-W**. By the invariant maintained by Procedure **GD-W**, every interval of $I_{\min}(U)$ is dominated by at least one integer of $\tilde{W}$, and intervals with right endpoints in $\tilde{W}$ are disjoint. By Proposition 1, the minimum number of integers that dominate intervals of $I_{\min}(U)$ is at least $|\tilde{W}|$. Thus, $\tilde{W}$ is a minimum cardinality subset of W such that $\tilde{W}$ dominates U.

We have proved the correctness of Procedure **GD-W**. Now, we show how to implement it in $O(|U| + |W|)$ time. By Lemma 3, the representing sorted array $A_{\min}$ of $I_{\min}(U)$ can be constructed in $O(|U|)$ time. We will use an array L together with linked pointers to store the endpoints of intervals of $I_{\min}(U)$, where $|L| = |W|$ and the index of L corresponds to the integer (vertex) of W. In addition, we need a stack S to perform operations of this procedure. Initially, let L and S be empty. We first visits the elements of $A_{\min}$ one by one. Suppose that it is about to process element $A_{\min}[i]$ with a triple $(u, u.begin, u.end)$, where $u \in U$. Then, we insert the left endpoint $l(I_u)$ of the neighbor interval I_u of u into $L[u.begin]$, insert the right endpoint $r(I_u)$ of I_u into $L[u.end]$, and link $l(I_u)$ to $r(I_u)$. If $L[u.begin]$ (resp. $L[u.end]$) is not empty, then we append $l(I_u)$ (resp. $r(I_u)$) to the tail of $L[u.begin]$ (resp. $L[u.end]$). After completing the above process, we construct an array L with linked points to maintain the sorted endpoints of intervals of $I_{\min}(U)$. For example, Fig. 7(a) shows the linked array L for $I_{\min}(U)$ shown in Fig. 5(a). It is easy to see that the above process runs in $O(|U|)$ time. We then progress iteratively through each element of the constructed linked array L in nondecreasing order. When a left endpoint $l(I_i)$ is visited, it is pushed into stack S. When a right endpoint $r(I_i)$ is visited, $s(I_{\min})$ is found, where $L[w]$ stores $r(I_i)$ which is the right endpoint of $s(I_{\min})$. In this time, let $\tilde{W} = \tilde{W} \cup \{w\}$. And, it repeats popping $l(I_i)$ from stack S and removing its linked right endpoint $r(I_i)$ from L until S is empty, and it removes $r(I_j)$ from L for all $l(I_j)$ stored in $L[w]$. For instance, Fig. 7(b) depicts the remnants of linked array L for Fig. 7(a) after finding the first vertex 3 of $\tilde{W}$. After processing each element of L once, $\tilde{W}$ is computed. Since $|L| = |W|$ and the number of endpoints of intervals of $I_{\min}(U)$ is $2|U|$, the above implementation of Procedure **GD-W** can be done in $O(|U| + |W|)$ time. Thus, we have the following lemma.

Lemma 6 *Procedure **GD-W** computes a minimum cardinality subset $\tilde{W}$ of W that dominates U, and it runs in $O(|U| + |W|)$ time.*

By similar strategy in computing $\tilde{W}$, we can find a minimum cardinality subset $\tilde{U}$ of U such that $\tilde{U}$ dominates W. Notice that the vertices of U and W are always represented by integers, and that an interval covers an integer p if p is in the interval. Observe that if there exists a vertex j in $\tilde{U}$ such that its neighbor interval $I_j \notin I_{\max}(U)$,



(a) $L[\cdot]$
1 2 3 4 5 6 7 8 9 10
$l(I_1)$ $l(I_5)$ $r(I_1)$ $l(I_6)$ $r(I_4)$ - $l(I_7)$ $r(I_6)$ - $r(I_7)$
$r(I_5)$
$l(I_4)$
(b) $L[\cdot]$
1 2 3 4 5 6 7 8 9 10
$l(I_6)$ - - $l(I_7)$ $r(I_6)$ - $r(I_7)$

Fig. 7 (**a**) The linked array L maintaining sorted endpoints of $I_{\min}(U)$ shown in Fig. 5(a), and (**b**) the remnants of linked array L after computing the first vertex 3 of $\tilde{W}$

then j can be replaced by one vertex i such that $I_i \in I_{\max}(U)$ and $I_j \subseteq I_i$. Thus, we can only consider the vertices whose neighbor intervals are in $I_{\max}(C_i)$, $1 \leqslant i \leqslant k$. Then, we are given by $I_{\max}(U)$. The problem of finding a minimum cardinality subset of U dominating W is then equivalent to compute a minimum set of intervals in $I_{\max}(U)$ such that they together cover all integers of W. Before showing how to compute such a set of intervals in $I_{\max}(U)$, we observe one property below. Let $\hat{W}$ be a subset of W such that every interval of $I_{\max}(U)$ covers at most one integer of $\hat{W}$. Since every integer needs to be covered by at least one interval, the number of intervals covering all integers of $\hat{W}$ is at least $|\hat{W}|$. For example, given a set $I_{\max}(U)$ of intervals shown in Fig. 5(b), let $\hat{W} = \{1, 5, 9\}$. Then, no interval of $I_{\max}(U)$ covers more than one integer of $\hat{W}$. By the pigeonhole principle, the minimum number of intervals covering all integers of $\hat{W}$ is at least three. Thus we have the following proposition.

Proposition 2 *Let $\hat{W}$ be a subset of W such that no two integers of $\hat{W}$ are covered by one interval of $I_{\max}(U)$. Then, the minimum number of intervals of $I_{\max}(U)$ covering all integers of W is at least $|\hat{W}|$.*

By the above proposition, we can find a subset $\tilde{U}$ of U and a subset $\hat{W}$ of W such that $\tilde{U}$ dominates W, no interval of $I_{\max}(U)$ covers more than one integer of $\hat{W}$, and $|\tilde{U}| = |\hat{W}|$. Then, $\tilde{U}$ is the required set. We now introduce Procedure **GD-U** to construct such a minimum cardinality subset $\tilde{U}$ of U that dominates W. Given a set $I_{\max}(U)$ of intervals, Procedure **GD-U** uses a greedy principle to obtain a subset $\tilde{U}$ of U as follows. Initially, let $\tilde{U} = \emptyset$, $\hat{W} = \emptyset$, and let $I_{\max} = I_{\max}(U)$. Let w be the smallest integer of intervals in $I_{\max}$, let I^w be the set of intervals in $I_{\max}$ covering w, and let $s(I_{\max})$ denote the interval in I^w with the largest right endpoint z. Let u be a vertex of U such that its neighbor interval I_u is the interval with right endpoint z. Let $\tilde{U} = \tilde{U} \cup \{u\}$ and let $\hat{W} = \hat{W} \cup \{w\}$. Remove from $I_{\max}$ all integers which are not larger than z. Repeat the above process until $I_{\max} = \emptyset$. Then it outputs $\tilde{U}$. For example, given a set $I_{\max}(U)$ of intervals shown in Fig. 5(b), Procedure **GD-U** outputs $\tilde{U} = \{3, 6, 8\}$. Note that $\hat{W} = \{1, 5, 9\}$ is used to verify the optimality of Procedure **GD-U**.

We can see that Procedure **GD-U** maintains the following invariant while growing sets $\tilde{U} = \{u_1, u_2, \ldots, u_p\}$ and $\hat{W} = \{w_1, w_2, \ldots, w_p\}$ with $u_1 < u_2 < \cdots < u_p$ and $w_1 < w_2 < \cdots < w_p$ to be computed:

Every integer of W covered by none of neighbor intervals of $\tilde{U}$ is larger than z_p, where z_p is the right endpoint of neighbor interval of u_p, and no interval of $I_{\max}$ covers more than one integer of $\hat{W}$.

The above invariant can be easily verified from the greedy principle of Procedure **GD-U** by induction. Let $\tilde{U}$ be the output by Procedure **GD-U**. By the invariant maintained by Procedure **GD-U**, $\tilde{U}$ dominates W and no interval of $I_{\max}$ covers more than one integer of $\hat{W}$. Note that $|\tilde{U}| = |\hat{W}|$. By Proposition 2, the minimum number of intervals of $I_{\max}(U)$ covering all integers of W is at least $|\hat{W}|$. This proves the optimality of Procedure **GD-U**. By similar arguments in analyzing the complexity of Procedure **GD-W**, it is not hard to verify that Procedure **GD-U** can be implemented in $O(|U| + |W|)$ time. Thus, we have the following lemma.

Lemma 7 *Procedure **GD-U** computes a minimum cardinality subset $\tilde{U}$ of U that dominates W, and it runs in $O(|U| + |W|)$ time.*

Based upon Lemmas 1, 3, 4, 6, and 7, our algorithm for the paired-domination problem is given by a compact representation of a convex bipartite graph $G = (U, V, E)$ and is sketched as follows.

Stage 1 Partition U into k disjoint sorted clusters $C_1, C_2, \ldots, C_k$;
Stage 2 Compute the interval representation $I(U)$ of U, and construct $I_{\min}(U)$ and $I_{\max}(U)$ from $I(U)$;
Stage 3 Call Procedure **GD-W** given $I_{\min}(U)$ to compute a minimum cardinality subset $\tilde{W}$ of W such that $\tilde{W}$ dominates U, and call Procedure **GD-U** given $I_{\max}(U)$ to compute a minimum cardinality subset $\tilde{U}$ of U such that $\tilde{U}$ dominates W;
Stage 4 Construct a minimum paired-dominating set D_{mpd} of G with size not less than $2 \cdot \max\{|\tilde{W}|, |\tilde{U}|\}$, and output D_{mpd}.

In Stage 4, we need to construct a minimum paired-dominating set D_{mpd} of G with cardinality not less than $2 \cdot \max\{|\tilde{W}|, |\tilde{U}|\}$. This construction is shown as follows. Let $\tilde{U} = \{u_1, u_2, \cdots, u_{|\tilde{U}|}\}$ such that $u_i < u_{i+1}$ for $1 \leqslant i \leqslant |\tilde{U}| - 1$. Note that $u_i < u_{i+1}$ indicates that $u_i.begin < u_{i+1}.begin$, i.e., the cluster containing u_i is smaller than the cluster containing u_{i+1} in the sorted clusters. For convenience, we denote by C_i the cluster containing u_i for $1 \leqslant i \leqslant |\tilde{U}|$. Let $\tilde{W} = \{w_1, w_2, \ldots, w_{|\tilde{W}|}\}$ such that $w_i < w_{i+1}$ for $1 \leqslant i \leqslant |\tilde{W}| - 1$. We first consider that $|\tilde{U}| = \max\{|\tilde{W}|, |\tilde{U}|\}$. Initially, let $D_{\text{mpd}} = \emptyset$. We then traverse the vertices of $\tilde{U}$ in a nondecreasing order, i.e., u_i is visited before u_{i+1} for $1 \leqslant i \leqslant |\tilde{U}| - 1$. Suppose that it is about to process u_i. Let w_h be the smallest unpaired vertex of $\tilde{W} \cap N(u_i)$, i.e., $w_h \leqslant w_j$ for

$w_j \in (\tilde{W} \cap N(u_i)) - D_{\text{mpd}}$. If such a vertex w_h does not exist, then let $w_h = u_i.end$. By the construction of Procedure **GD-U**, w_h does exist. By pairing u_i with w_h, we obtain a new set $D_{\text{mpd}} = D_{\text{mpd}} \cup \{u_i, w_h\}$. After processing each vertex of $\tilde{U}$, let $\hat{W} = \tilde{W} - D_{\text{mpd}}$. If $\hat{W} = \emptyset$, then D_{mpd} is a paired-dominating set of G with cardinality $2 \cdot \max\{|\tilde{W}|, |\tilde{U}|\}$, and, hence, D_{mpd} is a minimum paired-dominating set of G by Lemma 4. Suppose that $\hat{W} \neq \emptyset$. Let $\hat{w}$ be the smallest vertex of $\hat{W}$ and let $\hat{u}$ be the vertex in cluster $\hat{C}$ such that $\hat{u}.end = \hat{w}$. Then, D_{mpd} does not dominate $\hat{C}$ and hence it is not a dominating set of G. In this time, we pair $\hat{u}$ with $\hat{w}$, i.e., let $D_{\text{mpd}} = D_{\text{mpd}} \cup \{\hat{u}, \hat{w}\}$, and let $\hat{W} = \hat{W} - \{\hat{w}\}$. Repeat the above process until $\hat{W} = \emptyset$. Then, D_{mpd} is the constructed paired-dominating set of G.

The optimality of the above process under that $\hat{W} \neq \emptyset$ is shown as follows. Let D_{mpd} be the constructed set before processing the vertices of $\hat{W}$, i.e., $|D_{\text{mpd}}| = 2 \cdot |\tilde{U}|$, and let $\hat{W} = \tilde{W} - D_{\text{mpd}} \neq \emptyset$. We will prove the optimality of the above process by induction on $|\hat{W}|$. Let $\hat{W} = \{\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_{|\hat{W}|}\}$ such that $\hat{w}_j < \hat{w}_{j+1}$ for $1 \leqslant j \leqslant |\hat{W}| - 1$, and let $\hat{u}_i$ be the vertex in cluster $\hat{C}_i$ such that $\hat{u}_i.end = \hat{w}_i$ for $1 \leqslant i \leqslant |\hat{W}|$. Let $\hat{U}_i$ be the set of clusters that are less than cluster $\hat{C}_i$, and let $\hat{W}_i = N(\hat{U}_i)$. Let $G_i = (\hat{U}_i, \hat{W}_i, E_i)$ be a subgraph of G induced by $\hat{U}_i \cup \hat{W}_i$, and let $\hat{G}_i = (\hat{U}_i \cup \hat{C}_i, \hat{W}_i, \hat{E}_i)$ be a subgraph of G induced by $\hat{U}_i \cup \hat{C}_i \cup \hat{W}_i$, where $1 \leqslant i \leqslant |\hat{W}|$. Then, G_i and $\hat{G}_i$ are convex bipartite graphs, $\hat{u}_i \notin G_i$, $\hat{u}_i \in \hat{G}_i$, $\hat{w}_i \in G_i$, and $\hat{w}_i \in \hat{G}_i$. Let $D_{\text{mpd}|G'} = D_{\text{mpd}} \cap G'$, where G' is either G_i or $\hat{G}_i$. Then, $D_{\text{mpd}|G_i} = D_{\text{mpd}|\hat{G}_i}$. Let $D_{\text{p}} = \{\hat{u}_1, \hat{w}_1, \ldots, \hat{u}_{i-1}, \hat{w}_{i-1}\}$ if $i > 1$; otherwise, let $D_{\text{p}} = \emptyset$. We will prove the following invariant:

$D_{\text{mpd}|G_i} \cup D_{\text{p}}$ is a minimum paired-dominating set of G_i, there exists no minimum paired-dominating set of G_i such that it contains $\hat{w}_i$, and $D_{\text{mpd}|G_i} \cup D_{\text{p}} \cup \{\hat{u}_i, \hat{w}_i\}$ is a minimum paired-dominating set of $\hat{G}_i$.

Initially, consider the smallest vertex $\hat{w}_1$ of $\hat{W}$. Let $\tilde{U}_1$ and $\tilde{W}_1$ be the outputs of Procedure **GD-U** and Procedure **GD-W** given G_1, respectively. By the construction of D_{mpd}, $|\tilde{U}_1| = |\tilde{W}_1| \geqslant 1$ and $\hat{w}_1$ is dominated by $\tilde{U}_1$. Then, $D_{\text{mpd}|G_1} = \tilde{U}_1 \cup \tilde{W}_1$ is a minimum paired-dominating set of G_1 by Lemma 4. If there exists a set of paired vertices of G_1 with size $|\tilde{U}_1| + |\tilde{W}_1|$ such that it contains $\hat{w}_1$, then it does not dominate at least one cluster of $\hat{U}_1$ since the clusters of $\hat{U}_1$ must be dominated by a subset of $\tilde{W}_1$ whose cardinality is at least $|\tilde{W}_1|$, and, hence, it is not a dominating set. Thus, there exists no minimum paired-dominating set of G_1 such that it contains $\hat{w}_1$. On the other hand, let $\tilde{U}'_1$ and $\tilde{W}'_1$ be the outputs of Procedure **GD-U** and Procedure **GD-W** given $\hat{G}_1$, respectively. Then, $\tilde{U}'_1 = \tilde{U}_1$ and $\tilde{W}'_1 = \tilde{W}_1 \cup \{\hat{w}_1\}$. Since $|\tilde{U}_1| = |\tilde{W}_1|$, $|\tilde{W}'_1| = |\tilde{U}_1| + 1$. By Lemma 4, $\gamma_p(\hat{G}_1) \geqslant 2 \cdot \max\{|\tilde{W}'_1|, |\tilde{U}'_1|\} = 2 \cdot |\tilde{W}'_1| = 2 \cdot (|\tilde{U}'_1| + 1)$. Thus, $D_{\text{mpd}|G_1} \cup \{\hat{u}_1, \hat{w}_1\}$ is a minimum paired-dominating set of $\hat{G}_1$. Assume now the invariant holds true when $i = h \geqslant 1$. Let $D_{\text{p}} = \{\hat{u}_1, \hat{w}_1, \ldots, \hat{u}_{h-1}, \hat{w}_{h-1}, \hat{u}_h, \hat{w}_h\}$. Consider that G_{h+1} and $\hat{G}_{h+1}$. Let $\tilde{U}_{h+1}$ and $\tilde{W}_{h+1}$ be the outputs of Procedure **GD-U** and Procedure **GD-W** given G_{h+1}, respectively. By the construction of D_{mpd}, $|\tilde{U}_{h+1} - \hat{G}_h| = |\tilde{W}_{h+1} - \hat{G}_h| \geqslant 1$ and $\hat{w}_{h+1}$ is dominated by $\tilde{U}_{h+1}$ since $\tilde{U}$ dominates $\tilde{w}_{h+1}$ but $\tilde{U} - \tilde{U}_{h+1}$ does not dominate $\tilde{w}_{h+1}$. By the induction hypothesis, $D_{\text{mpd}|G_h} \cup D_{\text{p}}$ is a minimum paired-dominating set of $\hat{G}_h$. Thus, $D_{\mathrm{mpd}|G_{h+1}} \cup D_{\mathrm{p}}$ is a minimum paired-dominating set of G_{h+1}. If there exists a set of paired vertices of G_{h+1} with size $|D_{\mathrm{mpd}|G_{h+1}} \cup D_{\mathrm{p}}|$ such that it contains $\hat{w}_{h+1}$, then it does not dominate at least one cluster of $\hat{U}_{h+1}$ which is less than cluster $\hat{C}_{h+1}$, and, hence it is not a dominating set. Thus, there exists no minimum paired-dominating set of G_{h+1} such that it contains $\hat{w}_{h+1}$. Since $D_{\mathrm{mpd}|G_{h+1}} \cup D_{\mathrm{p}}$ is a minimum paired-dominating set of G_{h+1}, there exists no minimum paired-dominating set of G_{h+1} such that it contains $\hat{w}_{h+1}$, and $D_{\mathrm{mpd}|G_{h+1}} \cup D_{\mathrm{p}}$ is not a dominating set of $\hat{G}_{h+1}$, we get that $D_{\mathrm{mpd}|G_{h+1}} \cup D_{\mathrm{p}} \cup \{\hat{u}_{h+1}, \hat{w}_{h+1}\}$ is a minimum paired-dominating set of $\hat{G}_{h+1}$. Thus, the invariant holds true when $i = h + 1$. By induction then, the invariant holds true. This proves the optimality of our construction.

Next, we consider that $|\tilde{W}| = \max\{|\tilde{W}|, |\tilde{U}|\}$. By the similar construction for the case of $|\tilde{U}| = \max\{|\tilde{W}|, |\tilde{U}|\}$, we can construct a minimum paired-dominating set of G as follows. Initially, let $D_{\mathrm{mpd}} = \emptyset$. We first traverse the vertices of $\tilde{W}$ in an increasing manner, i.e., w_i is visited before w_{i+1} for $1 \leqslant i \leqslant |\tilde{W}| - 1$. Suppose that it is about to process w_i. Let u_h be the smallest vertex of $N(w_i)$ such that it is in $\tilde{U}$ but is not in D_{mpd}. If such a vertex u_h does not exist, then let u_h be the vertex of cluster C_h such that $w_i = u_h.end$ and $C_h \cap \tilde{U} = \emptyset$. By the construction of Procedure **GD-W**, u_h does exist. By pairing w_i with u_h, we obtain a new set $D_{\mathrm{mpd}} = D_{\mathrm{mpd}} \cup \{w_i, u_h\}$. After processing each vertex of $\tilde{W}$, let $\hat{U} = \tilde{U} - D_{\mathrm{mpd}}$. If $\hat{U} = \emptyset$, then D_{mpd} is a paired-dominating set of G with cardinality $2 \cdot |\tilde{W}|$, and, hence, D_{mpd} is a minimum paired-dominating set of G by Lemma 4. Suppose that $\hat{U} \neq \emptyset$. Let $\hat{u}$ be the smallest vertex of $\hat{U}$ and let $\hat{w}$ be the vertex in W such that $\hat{w} = \hat{u}.end$. Then, D_{mpd} does not dominate $\hat{w}$ and hence it is not a dominating set of G. In this time, we pair $\hat{w}$ with $\hat{u}$, i.e., let $D_{\mathrm{mpd}} = D_{\mathrm{mpd}} \cup \{\hat{w}, \hat{u}\}$, and let $\hat{U} = \hat{U} - \{\hat{u}\}$. Repeat the above process until $\hat{U} = \emptyset$. Then, D_{mpd} is the constructed paired-dominating set of G. The optimality of the above process can be verified by similar arguments in proving the case of $|\tilde{U}| = \max\{|\tilde{W}|, |\tilde{U}|\}$.

For instance, given $I_{\min}(U)$ and $I_{\max}(U)$ shown in Fig. 5, Procedure **GD-W** outputs $\tilde{W} = \{3, 8\}$ and Procedure **GD-U** outputs $\tilde{U} = \{3, 6, 8\}$. Then, $|\tilde{U}| = \max\{|\tilde{W}|, |\tilde{U}|\} = 3$. By the above construction, we obtain a set of pairs $(3, 3)$, $(6, 8)$, $(8, 12)$, where pair (u, w) satisfies that $u \in U$ and $w \in W$, bold lines in Fig. 5 depicts such three pairs, and a minimum paired-dominating set D_{mpd} of size 6. Let k be the number of disjoint clusters of U. By Lemmas 2 and 4, $2k \geqslant \gamma_p(G) \geqslant 2 \cdot \max\{|\tilde{W}|, |\tilde{U}|\}$. Then, $|U| \geqslant k \geqslant \max\{|\tilde{W}|, |\tilde{U}|\}$. Thus, the above process for constructing a minimum paired-dominating set of G runs in $O(|U|)$ time, and, hence, Stage 4 can be done in $O(|U|)$ time.

By Lemma 1, Stage 1 can be done in $O(|U| + |W|)$ time. By Lemma 3, Stage 2 runs in $O(|U|)$ time. By Lemmas 6 and 7, Stage 3 can be done in $O(|U| + |W|)$ time. Thus, the algorithm runs in $O(|U| + |W|)$ time and we conclude the following theorem.

Theorem 2 *The paired-domination problem on an n-vertex convex bipartite graph given a compact representation can be solved in $O(n)$ time.*

5 The Total Domination Problem in Convex Bipartite Graphs

In this section, we will extend our algorithm to solve the total domination problem on convex bipartite graphs. Let $G = (U, W, E)$ be a convex bipartite graph. Let $\tilde{U}$ and $\tilde{W}$ be the outputs of Procedure **GD-U** and Procedure **GD-W** given $I_{\max}(U)$ and $I_{\min}(U)$ of G, respectively, and let $D_{\mathrm{mtd}} = \tilde{U} \cup \tilde{W}$. We will show that D_{mtd} is a minimum total dominating set of G, i.e., $\gamma_t(G) = |\tilde{U}| + |\tilde{W}|$. Note that $\tilde{U}$ and $\tilde{W}$ are minimum cardinality subsets of U and W, respectively, such that $\tilde{U}$ dominates W and $\tilde{W}$ dominates U, and that $C_1, C_2, \ldots, C_k$ are sorted clusters of U.

We first partition U into three disjoint subsets $U_{\min}$, $U_{\max}$, and $\overline{U}$ such that $U_{\min} = \bigcup_{1 \leqslant i \leqslant k} \min(C_i)$, $U_{\max} = \bigcup_{1 \leqslant i \leqslant k} \max(C_i)$, and $\overline{U} = U - (U_{\min} \cup U_{\max})$. Then, $N(\min(C_i)) \subseteq N(u) \subseteq N(\max(C_i))$ for $u \in C_i$. We first claim that there exists a minimum total dominating set $\mathcal{D}_{\mathrm{td}}$ of G such that $(\mathcal{D}_{\mathrm{td}} \cap U) \subseteq U_{\max}$. We will prove the above claim as follows. Assume that $\mathcal{D}_{\min}$ is a minimum total dominating set of G. Let $\mathcal{D}_{U_{\min}} = \mathcal{D}_{\min} \cap U_{\min}$, $\mathcal{D}_{U_{\max}} = \mathcal{D}_{\min} \cap U_{\max}$, $\mathcal{D}_{\overline{U}} = \mathcal{D}_{\min} \cap \overline{U}$, and let $\mathcal{D}_W = D_{\min} \cap W$. Consider that $\mathcal{D}_{U_{\min}} \neq \emptyset$. Let $u_{\min} \in D_{U_{\min}}$ and $u_{\max} \in U_{\max}$ such that $u_{\min}$ and $u_{\max}$ are in the same cluster. Then, $N(u_{\min}) \subseteq N(u_{\max})$. Since $\mathcal{D}_{\min}$ is a total dominating set of G, $u_{\min}$ is adjacent to at least one vertex of $\mathcal{D}_W$. Thus, $\mathcal{D}_W$ dominates $u_{\min}$. If $u_{\max} \in \mathcal{D}_{\min}$, then $\mathcal{D}_{\min} - \{u_{\min}\}$ is still a total dominating set of G since $N(u_{\min}) \subseteq N(u_{\max})$ and $\mathcal{D}_W$ dominates $u_{\min}$, and, hence, it contradicts that $\mathcal{D}_{\min}$ is a minimum total dominating set of G. Thus, $u_{\max} \notin \mathcal{D}_{\min}$. Let $\mathcal{D}_{\mathrm{td}} = \mathcal{D}_{\min} - \{u_{\min}\} \cup \{u_{\max}\}$. Then, $\mathcal{D}_{\mathrm{td}}$ is a total dominating set of G with size $|\mathcal{D}_{\min}|$. Thus, there exists a minimum total dominating set $\mathcal{D}_{\mathrm{td}}$ of G such that $\mathcal{D}_{\mathrm{td}} \cap U_{\min} = \emptyset$. By the same arguments, we can show that there exists a minimum total dominating set $\mathcal{D}_{\mathrm{td}}$ of G such that $\mathcal{D}_{\mathrm{td}} \cap \overline{U} = \emptyset$. It follows from the above arguments that the claim holds true.

By the above claim, there exists a minimum total dominating set $\mathcal{D}_{\mathrm{td}}$ of G such that $(\mathcal{D}_{\mathrm{td}} \cap U) \subseteq U_{\max}$. Then, $|\mathcal{D}_{\mathrm{td}}| = |\mathcal{D}_{\mathrm{td}} \cap W| + |\mathcal{D}_{\mathrm{td}} \cap U_{\max}|$. Since $\mathcal{D}_{\mathrm{td}}$ is a total dominating set of G, every vertex of $\mathcal{D}_{\mathrm{td}} \cap W$ (resp. $\mathcal{D}_{\mathrm{td}} \cap U_{\max}$) is adjacent to at least one vertex of $\mathcal{D}_{\mathrm{td}} \cap U_{\max}$ (resp. $\mathcal{D}_{\mathrm{td}} \cap W$). Thus, $\mathcal{D}_{\mathrm{td}} \cap U_{\max}$ dominates W and $\mathcal{D}_{\mathrm{td}} \cap W$ dominates U. By the constructions of $\tilde{U}$ and $\tilde{W}$, $|\mathcal{D}_{\mathrm{td}} \cap U_{\max}| \geqslant |\tilde{U}|$ and $|\mathcal{D}_{\mathrm{td}} \cap W| \geqslant |\tilde{W}|$. Thus, $|\mathcal{D}_{\mathrm{td}}| \geqslant |\tilde{U}| + |\tilde{W}| = |D_{\mathrm{mtd}}|$. That is, $D_{\mathrm{mtd}} = \tilde{U} \cup \tilde{W}$ is a minimum total dominating set of G. Therefore, we conclude the following theorem.

Theorem 3 *The total domination problem on an n-vertex convex bipartite graph given a compact representation can be solved in $O(n)$ time.*

6 Concluding Remarks

The paired-domination problem on bipartite graphs has been shown to be NP-complete. The complexity of the paired-domination problem on convex bipartite graphs was unknown prior to our work. In this paper, we give an innovative approach to solve the paired-domination problem on convex bipartite graphs given a compact representation in $O(n)$ time. We also show that the proposed approach can be directly applied to solve the total domination problem on convex bipartite graphs in the same time bound. To the best of our knowledge, the best algorithms for the domination and independent domination problems on convex bipartite graphs run in $O(n^2)$ time. It is interesting to see if the proposed procedures can be used to solve the domination and independent domination problems on convex bipartite graphs given a compact representation in $O(n)$ time. We would like to post it as an open problem to interested readers.

Acknowledgements The author gratefully acknowledges the helpful comments and suggestions of the reviewers, which have improved the presentation. We also deeply appreciate the anonymous reviewers for giving the comment that the proposed approach can be applied to the total domination problem in convex bipartite graphs.

References

1. Arikati, S.R., Pandu Rangan, C.: Linear algorithm for optimal path cover problem on interval graphs. Inf. Process. Lett. **35**, 149–153 (1990)
2. Asdre, K., Nikolopoulos, S.D.: NP-completeness results for some problems on subclasses of bipartite and chordal graphs. Theor. Comput. Sci. **381**, 248–259 (2007)
3. Bang-Jensen, J., Huang, J., MacGillivray, G., Yeo, A.: Domination in convex bipartite and convex-round graphs. Tech. Rep. PP-1999-08, University of Southern Denmark (1999)
4. Booth, K., Lueker, G.: Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. J. Comput. Syst. Sci. **13**, 335–379 (1976)
5. Bose, P., Chan, A., Dehne, F., Latzel, M.: Coarse grained parallel maximum matching in convex bipartite graphs. In: 13th International Parallel Processing Symposium (IPPS'99), pp. 125–129 (1999)
6. Brodal, G.S., Georgiadis, L., Hansen, K.A., Katriel, I.: Dynamic matchings in convex bipartite graphs. Lect. Notes Comput. Sci. **4708**, 406–417 (2007)
7. Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph Classes: A Survey. Society for Industrial and Applied Mathematics, Philadelphia (1999)
8. Brandstädt, A., Eschen, E.M., Sritharan, R.: The induced matching and chain subgraph cover problems for convex bipartite graphs. Theor. Comput. Sci. **381**, 260–265 (2007)
9. Chan, A., Dehne, F., Bose, P., Latzel, M.: Coarse grained parallel algorithms for graph matching. Parallel Comput. **34**, 47–62 (2008)
10. Chang, M.S., Peng, S.L., Liaw, J.L.: Deferred-query: an efficient approach for some problems on interval graphs. Networks **34**, 1–10 (1999)
11. Chen, L., Lu, C., Zeng, Z.: A linear-time algorithm for paired-domination problem in strongly chordal graphs. Inf. Process. Lett. **110**, 20–23 (2009)
12. Chen, L., Lu, C., Zeng, Z.: Labelling algorithms for paired-domination problems in block and interval graphs. J. Comb. Optim. **19**, 457–470 (2010)
13. Cheng, T.C.E., Kang, L., Ng, C.T.: Paired domination on interval and circular-arc graphs. Discrete Appl. Math. **155**, 2077–2086 (2007)
14. Cheng, T.C.E., Kang, L., Shan, E.: A polynomial-time algorithm for the paired-domination problem on permutation graphs. Discrete Appl. Math. **157**, 262–271 (2009)
15. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd edn. MIT Press, Cambridge (2009)
16. Damaschke, P., Müller, H., Kratsch, D.: Domination in convex and chordal bipartite graphs. Inf. Process. Lett. **36**, 231–236 (1990)
17. Dekel, E., Sahni, S.: A parallel matching for convex bipartite graphs and applications to scheduling. J. Parallel Distrib. Comput. **1**, 185–205 (1984)
18. Gallo, G.: An $O(n \log n)$ algorithm for the convex bipartite matching problem. Oper. Res. Lett. **3**, 31–34 (1984)
19. Glover, F.: Maximum matching in a convex bipartite graph. Nav. Res. Logist. Q. **14**, 313–316 (1967)
20. Haynes, T.W., Slater, P.J.: Paired-domination in graphs. Networks **32**, 199–206 (1998)
21. Haynes, T.W., Hedetniemi, S.T., Slater, P.J.: Fundamentals of Domination in Graphs. Dekker, New York (1998)

22. Haynes, T.W., Hedetniemi, S.T., Slater, P.J.: Domination in Graphs: Advanced Topics. Dekker, New York (1998)
23. Hung, R.W., Chang, M.S.: Linear-time certifying algorithms for the path cover and Hamiltonian cycle problems on interval graphs. Appl. Math. Lett. **24**, 648–652 (2011)
24. Hung, R.W., Laio, C.H., Wang, C.K.: Efficient algorithm for the paired-domination problem in convex bipartite graphs. In: Proceedings of the International MultiConference of Engineers and Computer Scientists 2010 (IMECS'2010), vol. I, pp. 365–369 (2010)
25. Katriel, I.: Matchings in node-weighted convex bipartite graphs. INFORMS J. Comput. **20**, 205–211 (2008)
26. Korpelainen, N.: A polynomial-time algorithm for the dominating induced matching problem in the class of convex graphs. Electron. Notes Discrete Math. **32**, 133–140 (2009)
27. Lappas, E., Nikolopoulos, S.D., Palios, L.: An $O(n)$-time algorithm for the paired-domination problem on permutation graphs. Lect. Notes Comput. Sci. **5874**, 368–379 (2009)
28. Liang, Y.D., Chang, M.S.: Minimum feedback vertex sets in cocomparability graphs and convex bipartite graphs. Acta Inform. **34**, 337–346 (1997)
29. Lipski, W., Preparata, F.: Efficient algorithms for finding maximum matchings in convex bipartite graphs and related problems. Acta Inform. **15**, 329–346 (1981)
30. Müller, H.: Hamiltonian circuits in chordal bipartite graphs. Discrete Math. **156**, 291–298 (1996)
31. Nussbaum, D., Pu, S., Sack, J.R., Uno, T., Zarrabi-Zadeh, H.: Finding maximum edge bicliques in convex bipartite graphs. Lect. Notes Comput. Sci. **6196**, 140–149 (2010)
32. Park, E., Park, K.: An improved Boolean circuit for maximum matching in a convex bipartite graph. Fundam. Inform. **84**, 81–107 (2008)
33. Qiao, H., Kang, L., Cardei, M., Du, D.Z.: Paired-domination of trees. J. Glob. Optim. **25**, 43–54 (2003)
34. Soares, J., Stefanes, M.A.: Algorithms for maximum independent set in convex bipartite graphs. Algorithmica **53**, 35–49 (2009)
35. Srinivasan, A., Madhukar, K., Nagavamsi, P., Pandu Rangan, C., Chang, M.S.: Edge domination on bipartite permutation graphs and cotriangulated graphs. Inf. Process. Lett. **56**, 165–171 (1995)
36. Steiner, G., Yeoman, J.: A linear time algorithm for maximum matchings in convex, bipartite graphs. Comput. Math. Appl. **31**(12), 91–96 (1996)
37. Yang, S.J.: Efficient algorithms to solve the link-orientation problem for multi-square, convex-bipartite, and convex-split networks. Inf. Sci. **171**, 475–493 (2005)
38. Yen, W.C.K.: The bottleneck independent domination on the classes of bipartite graphs and block graphs. Inf. Sci. **157**, 199–215 (2003)
39. Yu, C.W., Chen, G.H., Ma, T.H.: On the complexity of the k-chain subgraph cover problem. Theor. Comput. Sci. **205**, 85–98 (1998)