Independent Domination on Tree Convex Bipartite Graphs

Yu Song[1], Tian Liu[1,⋆], and Ke Xu[2,⋆]

[1] Key Laboratory of High Confidence Software Technologies, Ministry of Education, Institute of Software, School of Electronic Engineering and Computer Science, Peking University, Beijing 100871, China
{songyufish,lt}@pku.edu.cn

[2] National Lab. of Software Development Environment, Beihang University, Beijing 100191, China
kexu@nlsde.buaa.edu.cn

Abstract. An independent dominating set in a graph is a subset of vertices, such that every vertex outside this subset has a neighbor in this subset (dominating), and the induced subgraph of this subset contains no edge (independent). It was known that finding the minimum independent dominating set (Independent Domination) is $\mathcal{NP}$-complete on bipartite graphs, but tractable on convex bipartite graphs. A bipartite graph is called tree convex, if there is a tree defined on one part of the vertices, such that for every vertex in another part, the neighborhood of this vertex is a connected subtree. A convex bipartite graph is just a tree convex one where the tree is a path. We find that the sum of larger-than-two degrees of the tree is a key quantity to classify the computational complexity of independent domination on tree convex bipartite graphs. That is, when the sum is bounded by a constant, the problem is tractable, but when the sum is unbounded, and even when the maximum degree of the tree is bounded, the problem is $\mathcal{NP}$-complete.

1 Introduction

A *dominating set* in a graph $G = (V, E)$ is a subset D of vertices, such that every vertex in $V \setminus D$ has a neighbor in D. An *independent dominating set* D is a special kind of dominating set which is also independent, that is, there is no edge whose both ends are in D. The problem of finding the minimum independent dominating set (IDS, in short) is $\mathcal{NP}$-complete on general graphs [3], chordal graphs [2], bipartite graphs [4], chordal bipartite graphs [7], etc.

A bipartite graph $G = (A, B; E)$ is called *tree convex*, if there is a tree $T = (A, F)$, such that for all vertex b in B, the neighborhood of b is a connected subtree in T [5,6]. When T is a path, G is called *convex*. It was known that IDS is $\mathcal{NP}$-complete on bipartite graphs [4], but becomes tractable on convex bipartite graphs [1]. A natural question is

⋆ Corresponding authors.

– *What is the boundary between intractability and tractability of IDS on bipartite graphs?*

In this paper, we answer this question. We first explore IDS on some simple cases, showing an intractability on star convex bipartite graphs, where T is a star, i.e. a bipartite complete graph $K_{1,|A|-1}$, and a tractability on triad convex bipartite graphs, where T is a triad, i.e. three paths with a common end. These results have already extended the known results of [4] and [1], respectively. Finally, we find the exact condition to differentiate $\mathcal{NP}$-completeness and $\mathcal{P}$: whether

$$t = \sum_{v_i : deg_T(v_i) > 2} deg_T(v_i)$$

is bounded by a constant or not, where $deg_T(v)$ is the degree of vertex v in tree T. The results of this paper are pictured in Figure 1.

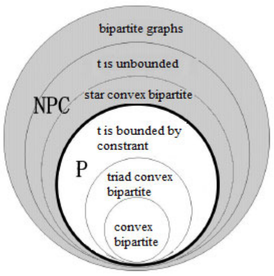

Fig. 1. The results of this paper

The remaining part of this paper is organized as follows. The $\mathcal{NP}$-completeness of IDS is shown on star convex bipartite graphs in Section 2, and then on more general graph classes in Section 3. Tractability of IDS is shown in Section 4. The conclusion and discussion are in Section 5.

2 Intractability of IDS on Star Convex Bipartite Graphs

IDS is $\mathcal{NP}$-complete in bipartite graphs [4]. We can refine this intractability into star convex bipartite graphs by a similar reduction.

Theorem 1. *IDS is $\mathcal{NP}$-complete on star convex bipartite graphs.*

Proof. We reduce from SAT to IDS on star convex bipartite graphs. Given an instance I of SAT, which has m variables $x_1, ..., x_m$ and n clauses $C_1, ..., C_n$, we construct a star convex bipartite graph $G = (A, B; E)$, such that I is satisfiable if and only if G has an IDS of size $2m$, as follows.

1. For each variable x_i in I $(1 \leq i \leq m)$, there is a small gadget involving six vertices $\{x_i, \bar{x}_i, u_i, v_i, y_i, z_i\}$ and six edges in G, as shown in Figure 2.

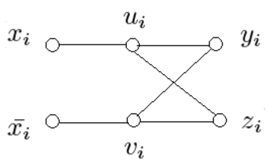

Fig. 2. The gadget for the literals $x_i, \bar{x}_i$

2. For each clause C_j in I $(1 \le i \le n)$, there is a vertex C_j in G.
3. For all literals $x_i, \bar{x}_i$ $(1 \le i \le m)$, and for all clauses C_j $(1 \le i \le n)$, we connect x_i and C_j if x_i is in C_j, connect $\bar{x}_i$ and C_j if $\bar{x}_i$ is in C_j.
4. Add a vertex v_0, which is connected to every u_i, v_i $(1 \le i \le m)$ and C_j $(1 < j < n)$.

Clearly, the construction is in polynomial time. An example of this construction is in Figure 3 for a SAT instance with two clauses $C_1 = x_1 \vee x_2$ and $C_2 = \bar{x}_1 \vee x_2$.

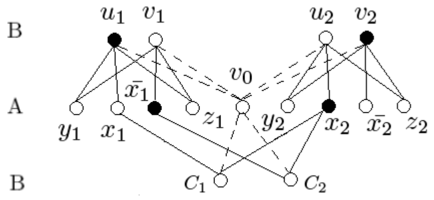

Fig. 3. An example of the construction of G

Lemma 1. *Graph G is star convex bipartite.*

Proof. The graph G is bipartite with respect to the following partition of vertices

$$A = \{x_i, \bar{x}_i, y_i, z_i | 1 \le i \le m\} \cup \{v_0\}, B = \{u_i, v_i | 1 \le i \le m\} \cup \{C_j | 1 \le j \le n\}.$$

Also G is star convex with the tree T on A be a star with central vertex v_0 and $4m$ leaves, since every vertex b in B is connected to v_0, the neighborhood of b is a subtree in T. □

Lemma 2. *If I is satisfiable, G has an IDS of size no more than $2m$.*

Proof. If there is a satisfying assignment to I, then the set

$$D = \bigcup_i \{x_i, v_i | x_i = true\} \cup \bigcup_i \{\bar{x}_i, u_i | x_i = false\}$$

is an independent set of size $2m$. The set D is also a dominating set, since every gadget is dominated by $\{x_i, v_i\}$ or $\{\bar{x}_i, u_i\}$, the vertex C_j is dominated by one of $\{x_i\}$ or $\{\bar{x}_i\}$ by the satisfying property, and v_0 is dominated by $\{u_i\}$ or $\{v_i\}$. In the example shown in Figure 3, a satisfying assignment of I is $x_1 = false$, $x_2 = true$, and $D = \{\bar{x}_1, u_1, x_2, v_2\}$. □

Lemma 3. *If G has an IDS of size no more than $2m$, I is satisfiable.*

Proof. Suppose there is an IDS of size no more than $2m$. Since there are m gadgets, and we can not use only one vertex to dominate all six vertices in one gadget, no matter whether we choose v_0 and C_j or not, we must choose exactly two vertices from each gadget. The limitation of the size $2m$ makes v_0 and C_j outside the IDS. For each i, the pair x_i and $\bar{x}_i$ can not be both in IDS, for otherwise, neither y_i nor z_i will be dominated. If we assign variable x_i to be true whenever IDS contains x_i, and false otherwise, we get an assignment that satisfies I, since each vertex C_j must be dominated, which implies that every clause C_j contains a true literal and thus is satisfied. □

The proof of Theorem 1 is finished. □

3 Intractability of IDS on Tree Convex Bipartite Graphs

We can transform the star T constructed in last section into a new tree whose maximum degree is bounded by a constant d_{max} as follows. We split the single central vertex v_0 with $4m$ leaves into a set of central vertices, whose cardinality is $\lceil 4m/(d_{max} - 2)\rceil$, to from a path in T. Every $(d_{max} - 2)$ of the original $4m$ leaves of v_0 form a group to connect to one of the central vertices. Figure 4 is an example of this transformation with $m = 5$ and $d_{max} = 6$. Amazingly, after this transformation, the whole reduction still works with some slight modification.

Theorem 2. *IDS is $\mathcal{NP}$-complete on tree convex bipartite graphs where the maximum degree of tree T is bounded by a constant.*

Proof. We still reduce from SAT. The modified whole reduction is as follows.

1. For each variable x_i in I $(1 \leq i \leq m)$, there is a small gadget in G involving six vertices $\{x_i, \bar{x}_i, u_i, v_i, y_i, z_i\}$ and six edges in G, as shown in Figure 2.
2. For each clause C_j in I $(1 \leq j \leq n)$, there is a vertex C_j in G.
3. If variable x_i is in clause C_j, we connect vertices x_i and C_j in G. If negated variable $\bar{x}_i$ is in C_j, we connect vertices $\bar{x}_i$ and C_j in G.
4. There are $p = \lceil 4m/(d_{max} - 2)\rceil$ central vertices v_{01}, v_{02}, ..., v_{0p} to form a path in T (not in G). The vertices y_i, x_i, $\bar{x}_i$, z_i $(1 \leq i \leq m)$ are ordered by y_1, x_1, $\bar{x}_1$, z_1, y_2, x_2, $\bar{x}_2$, z_2, and so on. Every $(d_{max} - 2)$ consecutive vertices in this order form a group, and one by one each group are leaves of one of the p central vertices in T (again not in G).

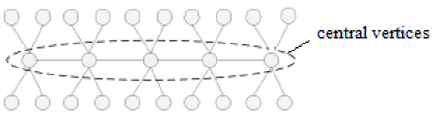

Fig. 4. Transforming a star into a tree of bounded maximum degree

5. For each C_j, find the minimum subtree T_j in T containing $\{x_i|x_i \in C_j\} \cup \{\bar{x}_i|\bar{x}_i \in C_j\}$, and connect C_j to the central vertices on T_j.
6. For x_i or $\bar{x}_i$ in C_j, connect u_i and v_i to the central vertices on above T_j.

Clearly, the construction is in polynomial time. Figure 5 shows an example of the construction with $d_{max} = 4$ and $p = 4$, for a SAT instance with two clauses $C_1 = x_1 \vee x_2$ and $C_2 = \bar{x}_1 \vee x_2$.

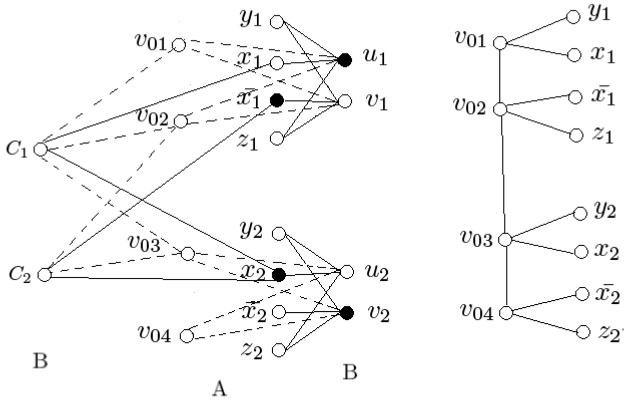

Fig. 5. An example of the constructions of G and T

Lemma 4. *Graph G is tree convex bipartite where the maximum degree of tree T is bounded by constant d_{max}.*

Proof. The graph G is bipartite with respect to the partition of vertices

$$A = \{x_i, \bar{x}_i, y_i, z_i|1 \leq i \leq m\} \cup \{v_{0k}|1 \leq k \leq p\},$$

$$B = \{u_i, v_i|1 \leq i \leq m\} \cup \{C_j|1 \leq j \leq n\}.$$

That the G is tree convex with maximum degree d_{max} in T is ensured by the construction, especially by the fifth and the sixth steps above. □

Lemma 5. *If I is satisfiable, G has an IDS of size no more than $2m$.*

Proof. If there is a satisfying assignment of I, the set

$$D = \bigcup_i \{x_i, v_i : x_i = true\} \cup \bigcup_i \{\bar{x}_i, u_i : x_i = false\}$$

is an independent set of size $2m$. The set D is also a dominating set, since every gadget is dominated by $\{x_i, v_i\}$ or $\{\bar{x}_i, u_i\}$, and C_j is dominated by x_i or $\bar{x}_i$ by the satisfying property, and the sixth step of construction ensures that each v_{0k} is connected to a u_i and a v_i. In the example in Figure 5, the satisfying assignment is $x_1 = false$ and $x_2 = true$, and the IDS is $D = \{\bar{x}_1, u_1, x_2, v_2\}$. □

Lemma 6. *If G has an IDS of size no more than $2m$, I is satisfiable.*

Proof. Suppose there is an IDS of size no more than $2m$. Since there are m gadgets, and we can not use a single vertex to dominate all six vertices in one gadget, no matter whether we choose v_{0k} and C_j ($1 \leq k \leq p$, $1 \leq j \leq m$) or not, so we must choose exactly two vertices from each gadget. The limitation of the size $2m$ kills both v_{0k} and C_j from the IDS. For each i, the pair x_i and $\bar{x}_i$ can not be both in IDS, for otherwise neither y_i nor z_i will be dominated. If we assign variable x_i to be true whenever IDS contains x_i, and false otherwise, we get an satisfying assignment for I, since each C_j must be dominated, which implies that every clause contains a true variable. □

This finishes the proof of Theorem 2. □

Note that in above two reductions, the sum of larger-than-two degrees in tree T is unbounded. Formally, let $t = \sum_{v_i:deg_T(v_i)>2} deg_T(v_i)$. Then $t = 4m$ for the T constructed in last section, and $t = 4m + p - 1$ for the T constructed in this section. In the former case, the number of larger-than-two degrees is bounded, but the maximum degree is unbounded, while in the later case, the number of larger-than-two degrees is unbounded, but the maximum degree is bounded, all in T. In both cases, the sum of larger-than-two degrees in tree T is unbounded. Thus we have the following intractability result.

Theorem 3. *IDS is $\mathcal{NP}$-complete on tree convex bipartite graphs whose t, the sum of larger-than-two degrees in tree T, is unbounded.*

4 Tractability of IDS on Tree Convex Bipartite Graphs

IDS is polynomial time on convex bipartite graphs [1]. We can extend this tractability onto more general tree convex bipartite graphs by a similar dynamic programming. We start with a simple situation as follows. Recall that a triad is just three paths with a common end.

Theorem 4. *IDS is in polynomial time on triad convex bipartite graphs.*

Proof. Our algorithm is an extension of the dynamic programming in [1]. Let D be a minimum subset of vertices with a desired property $\mathcal{Q}$. D is constructed as an increasing sequence $D_1 \subseteq D_2 \subseteq \cdots \subseteq D_m$ as follows.

- Step (a) (initialization): generate all possible versions of D_1.
- Step (b) (branching): extend all versions of present D_{i-1} to all possible versions of D_i with $D_{i-1} \subseteq D_i$. (Possible means that the versions satisfy $\mathcal{Q}$.)
- Step (c) (classification): classify the versions of D_i, such that if versions V and V' belong to the same class, then $V \subseteq D'$ and D' satisfies $\mathcal{Q}$ imply that $(D' \setminus V) \cup V'$ satisfies $\mathcal{Q}$.
- Step (d) (deletion): delete all versions except the one of minimum cardinality in each class.

The algorithm proceeds at steps (a), $(c)_1$, $(d)_1$, $(b)_2$, $(c)_2$, $(d)_2$, $(b)_3$, $(c)_3$, $\cdots$, $(b)_m$, $(c)_m$, $(d)_m$. We take some version of D_m with the minimum cardinality as the output minimum IDS.

Then we define some notations on triad convex bipartite graphs. Suppose $G = (A, B; E)$ and there is a triad T on A. The triad T consists of three paths $p = 1, 2, 3$ with a common vertex, as shown in the leftmost in Figure 6. For

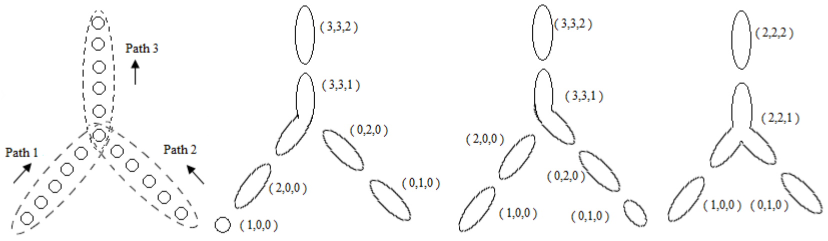

Fig. 6. Three paths in triad convex tree, and the label of A_i

each vertex x, $N(x)$ denotes the neighborhood of x in G. We partition A into nonempty sets A_1, A_2, $\cdots$, A_m, such that the following conditions hold:

- Each A_i consists of consecutive vertices in T.
- All vertices $x \in A_i$ have the same $N(x)$.
- Each A_i is maximal with respect to the above two conditions.
- Each A_i has a three-dimension label (x_i, y_i, z_i), where the p-th bit represents the order of A_i in path p of T in a bottom-up manner. Figure 6 shows some examples on how to label A_i.
- In the following text, we will also call A_i: $A_{(x_i,\ y_i,\ z_i)}$, and define $A_{(x_i,\ y_i,\ z_i)}$ $[1] := x_i$, $A_{(x_i,\ y_i,\ z_i)}[2] := y_i$, $A_{(x_i,\ y_i,\ z_i)}[3] := z_i$. Further defined three-dimension variable holds the same definition.

For each vertex y in B, we define $l(y)$ and $r(y)$ as a three-dimension variable, to record the range where y covers in each path. They are calculated as follows. If $N(y)$ include the common vertex of three paths, then

$$l(y) := \min\{A_{(x_i,y_i,z_i)}[p] | A_{(x_i,y_i,z_i)} \subseteq N(y) \cap \{v | v \text{ is on path } i \text{ in } T\}\},$$

$$r(y) := \max\{A_{(x_i,y_i,z_i)}[p] | A_{(x_i,y_i,z_i)} \subseteq N(y) \cap \{v | v \text{ is on path } i \text{ in } T\}\}.$$

Otherwise, $N(y)$ must intersect with only one path. If $N(y) \subseteq$ path 1, then $l(y)[2] = l(y)[3] = r(y)[2] = r(y)[3] = 0$. If $N(y) \subseteq$ path 2, then $l(y)[1] = l(y)[3]$ $= r(y)[1] = r(y)[3] = 0$. If $N(y) \subseteq$ path 3, then $l(y)[1] = r(y)[1] = A_{i'}[1]$, $l(y)[2] = r(y)[2] = A_{i'}[2]$, where $A_{i'}$ contains the common vertex of tree paths. Figure 7 shows an example of triad convex bipartite graph and the label of vertices of A and B.

There are some properties about A_i. Suppose that $N(A_i)$ are vertices connected to A_i in B, and D is the minimum IDS. If $N(A_i) \cap D \neq \emptyset$, from the

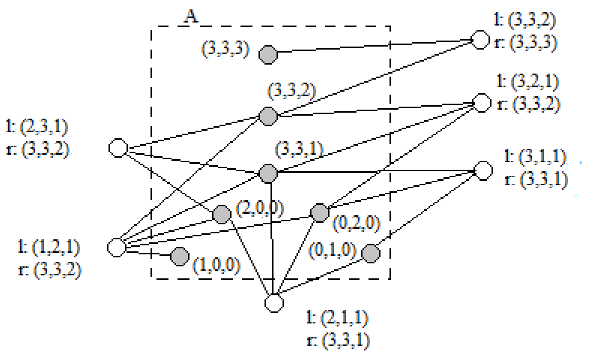

Fig. 7. The label of A_i, $l(y)$ and $r(y)$

independence of D it follows $A_i \cap D = \emptyset$. In the other case $N(A_i) \cap D = \emptyset$, to dominate A_i there must be $A_i \subseteq D$. For each A_i, we denote b_i with $b_i = 1$ iff $A_i \subseteq D$, and $b_i = 0$ iff $A_i \cap D = \emptyset$. Since there are only two possible b_i, we execute the dynamic programming process by dealing with one A_i per branching step, with a reverse order of breadth-first search, and extend D_{i-1} to D_i by enumerating the value of b_i.

Further, in each classification part of step i in the algorithm, we define s_i to be a three-dimension variable as follows. The p-th bit of s_i is

$$s_i[p] := max\{A_k[p] | A_k \cap \text{ path } p \neq \emptyset, A_k[p] < A_i[p], b_k = 1\}.$$

Actually, s_i represents the latest position where IDS has already dominated in each path of the convex tree. We define this variable because if we know s_i and whether A_i is chosen or not, i.e. b_i, we know how to choose vertices in B into IDS. So we can categorize versions with the same s_i into a class, and do the deletion procedure.

Recall the definition of D_i to be the minimum IDS found in step i. Then D_i has the following properties:

- $D_i = \bigcup\{A_k$ — $k \leq i$, $b_k = 1$ $\} \cup \{y \in B \mid$ r(y)[1] $\leq A_k[1]$, $r(y)[2] \leq A_k[2]$, $r(y)[3] \leq A_k[3]$, $b_k = 0$ for all $A_k \subseteq N(y)\}$;
- (opt) If $b_k = 0$, and every bit of $A_k \leq s_i$, then D_i must contain some vertex of $N(A_k)$.

Now we can represent our IDS algorithm as follows.

```
(a) version b1=0:
      D1:={y is vertex in B | r(y):=(D1[1],D1[2],D1[3])};
      s1:=(0,0,0).
    version b1=1:
      D1:=A1;
      s1:=(A1[1],A1[2],A1[3]).
(b) version bi=0:
```

```
         Di:=D(i-1) + {y is vertex in B|l(y)[1] > s(i-1)[1],
             l(y)[2] > s(i-1)[2], l(y)[3] > s(i-1)[3],
             r(y) = (Ai[1],Ai[2],Ai[3])};
        si:=s(i-1).
        If i=m, and Ai is not dominated, then delete the version.
       version bi=1:
         check (opt), if it fails, then delete the version;
         If (opt) hols, then Di:=D(i-1)+Ai;
         If Ai have common vertices with path p, si[p]:=Ai[p];
         else si[p]:=s(i-1)[p].
   (c) The version with equal sk belong to the same class.
   (d) Delete all versions except one of minimum cardinality in
       each class.
```

We can briefly analyze the running time of this algorithm as follows.

- Labeling each A_i and $l(y)$, $r(y)$ for each y in B cost $O(|A| + |A| \cdot |B|)$ time.
- Step (a) costs $O(|B|)$ time.
- Steps (b)(c)(d) will repeat $O(|A|)$ times. In each loop, there are at most $O(|A|^3)$ versions, we should calculate in $O(|A| + |B|)$ time for each version. Steps (c) and (d) cost $O(|A|^3)$ time. The total cost is $O(|A|^5 + |A|^4 \cdot |B|)$.

So the total running time is $O(|A|^5 + |A|^4 \cdot |B|)$, which is a polynomial. □

This algorithm is easily extended to more general situations. For tree convex bipartite graphs with the sum of larger-than-two degrees in the tree is bounded by t (this property will be call $\mathcal{Q}$), the tree is split into m paths, where $m \leq t$. Figure 8 briefly presents an example.

So, as long as we redefine the label of A_i, $l(y)$ and $r(y)$ ($y \in B$) to be m-dimension variables, in which each bit records the position and covering range of A_i and y in each path of the convex tree, the algorithm can operate as the same way:

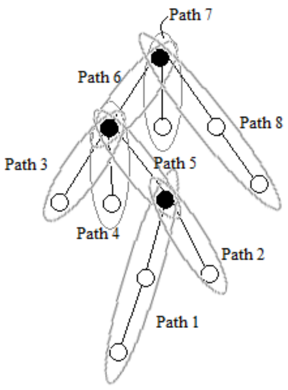

Fig. 8. An example of graphs satisfies $\mathcal{Q}$

- Initializing is as the same.
- Each A_i will be all or none chosen, which consist the two possibilities of branching in each step.
- We should record s_i, which is also a m-dimension variable, presenting the latest position where IDS has already dominated in each path. Versions with same s_i are classified into the same class.
- We only keep the version with minimum cardinality in each class.

With the same analysis, the above algorithm runs in time $O(|A|^{t+2} + |A|^{t+1} \cdot |B|)$, which proves the following theorem.

Theorem 5. *IDS is in polynomial time on tree convex bipartite graphs where the sum of larger-than-two degrees of the tree is bounded by a constant.*

5 Conclusion and Open Problems

We have shown a dichotomy of complexity of IDS on tree convex bipartite graphs: the problem is intractable when the sum of larger-than-two degrees in the tree is unbounded, and tractable when the sum is bounded by a constant. In our intractability reductions, the sum increases linearly. Can we make a reduction with an arbitrarily slow increasing of the sum? In our tractable algorithm, the running time is exponential in the sum. Can we get a better running time?

Acknowledgments. This research was partially supported by the National 973 Program of China (Grant No. 2010CB328103) and the National Natural Science Foundation of China (Grant No. 60973033).

References

1. Damaschke, P., Muller, H., Kratsch, D.: Domination in Convex and Chordal Bipartite Graphs Information Processing Letters 36, 231–236 (1990)
2. Farber, M.: Independent Domination in Chordal Graphs. Operations Research Letters 1, 134–138 (1982)
3. Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-Completeness (1979)
4. Irving, W.: On approximating the minimum independent dominating set. Information Processing Letters 37, 197–200 (1991)
5. Jiang, W., Liu, T., Ren, T.N., Xu, K.: Two Hardness Results on Feedback Vertex Sets. In: Atallah, M., Li, X.-Y., Zhu, B. (eds.) FAW-AAIM 2011. LNCS, vol. 6681, pp. 233–243. Springer, Heidelberg (2011)
6. Jiang, W., Liu, T., Xu, K.: Tractable Feedback Vertex Sets in Restricted Bipartite Graphs. In: Wang, W., Zhu, X., Du, D.-Z. (eds.) COCOA 2011. LNCS, vol. 6831, pp. 424–434. Springer, Heidelberg (2011)
7. Muller, H., Brandstadt, A.: The NP-completeness of Steiner Tree and Dominating Set for Chordal Bipartite Graphs. Theoretical Computer Science 53, 257–265 (1987)