Minimum Weight Independent Dominating Sets In Interval And Circular-Arc Graphs

Zachary Frenette
University of Waterloo

April 1, 2014

1 Introduction

Given an undirected graph $G = (V, E)$ with n vertices and m edges, we will define $N(v)$ to be the set of neighbours of a vertex v and $N[v] = N(v) \cup \{v\}$. Moreover, we will define analogous pieces of notation for a set of vertices $S \subseteq V$. In particular, let $N(S) = \bigcup_{v \in S} N(v)$ and let $N[S] = N(S) \cup S$. In these notes, the problem of interest will be the weighted independent dominating set problem. In particular, given a weight function $\omega : V \to \mathbb{R}^+$, we are interested in finding a set of vertices $S \subseteq V$ of minimum weight such that S is an independent set and $V \subseteq N[S]$. In other words, we want to find an independent set S such that for every vertex $v \in V$, if $v \notin S$ then v has a neighbour $u \in S$. Unfortunately for us, this problem is NP-complete for general graphs [3], and remains NP-complete even when we restrict the input to chordal graphs [2]. However, if we restrict ourselves to interval and circular-arc graphs, we can obtain linear time solutions to the problem [1].

Given a collection of intervals $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_n\}$, the intersection graph $G(\mathcal{I})$ is called an interval graph if there exists a bijection between the intervals of $\mathcal{I}$ and the vertices of G such that any two vertices are adjacent, if and only if their corresponding intervals have a nonempty intersection. Similarly, if we have a collection of arcs $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ around a circle, then the intersection graph $G(\mathcal{A})$ is instead called a circular-arc graph. We say that $\mathcal{I}$ is an interval model of $G(\mathcal{I})$ and $\mathcal{A}$ a circular-arc model of $G(\mathcal{A})$. Furthermore, we will make the following simplifying assumption, namely that every endpoint in an interval model will be an integer between 1 and $2n$. If this is not the case, we can simply sort them in nondecreasing order and then use the indices of the endpoints as a new interval model. In particular, a set of intervals satisfying this property will be called a set of sorted intervals, which will be the type of input provided to the algorithms we describe.

The rest of these notes are organized as follows. Section 2 will discuss a linear time algorithm for finding a minimum weight independent dominating set in an interval graph and section 3 will extend this result to circular-arc graphs. Lastly, we will finish with some brief concluding remarks.

2 Minimum Weight Independent Dominating Sets In Interval Graphs

In this section, we will present the details of an algorithm for finding a minimum weight independent dominating set in interval graphs, which requires $\mathcal{O}(n)$ time and $\mathcal{O}(n)$ space [1]. We will begin

by assuming that the input graph is given as an interval model $\mathcal{I}$, in which the n intervals $[a_i, b_i]$ are sorted in increasing order of their right endpoint. That is, $\mathcal{I}$ will be given in such a way that $i < j$ implies that $b_i \leq b_j$.

In order to facilitate the presentation of the required results, we will define the following pieces of notation. For a set $\mathcal{S} \subseteq \mathcal{I}$ of intervals, the largest left endpoint of $\mathcal{S}$ will be denoted by $\max_a(\mathcal{S})$ and the largest right endpoint of $\mathcal{S}$ will be denoted by $\max_b(\mathcal{S})$. Similarly, the smallest left endpoint and the smallest right endpoint of $\mathcal{S}$ will be denoted by $\min_a(\mathcal{S})$ and $\min_b(\mathcal{S})$ respectively. Furthermore, the interval with the largest and smallest right endpoint will be denoted by $\text{last}(\mathcal{S})$ and $\text{first}(\mathcal{S})$. We will also need some notation to denote certain sets of intervals. In particular, for any endpoint e, we define two sets of intervals: $\Delta_f(e) = \{[a_i, b_i] \in \mathcal{I} : b_i < e\}$ and $\Delta_s(e) = \{[a_i, b_i] \in \mathcal{I} : a_i < e\}$. In other words, $\Delta_f(e)$ is the set of all intervals that finish before e and $\Delta_s(e)$ is the set of all intervals that start before e. Immediately out of these definitions, we can make the following observation.

Lemma 1. *For any two endpoints e_1 and e_2 in $\mathcal{I}$ with $e_1 < e_2$, there does not exist an interval $[a_j, b_j] \in \mathcal{I}$ such that $e_1 < a_j < b_j < e_2$ if and only if $e_1 \geq \max_a(\Delta_f(e_2))$.*

Proof. If $e_1 \geq \max_a(\Delta_f(e_2))$, then no interval $[a_j, b_j]$ can lie in between e_1 and e_2. If we assume otherwise, then we have that $e_1 < a_j < b_j < e_2$. This would imply that $e_1 < \max_a(\Delta_f(e_2))$ since $\max_a(\Delta_f(e_2))$ would produce an endpoint that's at least as large as a_j. To prove the other direction, we can make a similar type of argument. If there exists no interval $[a_j, b_j]$ such that $e_1 < a_j < b_j < e_2$, then $e_1 \geq \max_a(\Delta_f(e_2))$. If we assume otherwise, then we have that $e_1 < \max_a(\Delta_f(e_2))$. However, this implies that $\max_a(\Delta_f(e_2))$ produced some endpoint a_k that is not part of the same interval as e_1. Hence there exists an interval $[a_k, b_k]$ contained in between e_1 and e_2, which contradicts our initial assumption. □

Using this Lemma, we can prove a second result that will help characterize dominating sets in interval graphs.

Lemma 2. *For any two intervals $[a_k, b_k], [a_i, b_i] \in \mathcal{I}$ with $k < i$, we have that the interval $[a_i, b_i]$ dominates $\Delta_s(b_i) - \Delta_s(b_k)$ if and only if $b_k > max_a(\Delta_f(a_i))$.*

Proof. Consider the partition of $\Delta_s(b_i) - \Delta_s(b_k)$ in which $\mathcal{S}_1 \subseteq N[i]$ and $\mathcal{S}_2 = \Delta_s(b_i) - \Delta_s(b_k) - \mathcal{S}_1$. In particular, observe that if some interval $[a_j, b_j] \in \Delta_s(b_i) - \Delta_s(b_k)$ does not intersect with $[a_i, b_i]$, then it must lie in between the endpoints b_k and a_i, as it would otherwise be part of $\Delta_s(b_k)$. Hence $\mathcal{S}_2 = \{[a_j, b_j] \in \mathcal{I} : b_k < a_j < b_j < a_i\}$. Observe that if $\mathcal{S}_2 = \emptyset$, then $[a_i, b_i]$ dominates $\Delta_s(b_i) - \Delta_s(b_k)$. However, by Lemma 1, $\mathcal{S}_2 = \emptyset$ if and only if $b_k > max_a(\Delta_f(a_i))$, as required. □

We are now ready to develop the framework that we will use to find a minimum weight independent dominating set in interval graphs. For later convenience, we will define $\mathcal{I}'$ to be the set of intervals obtained by augmenting $\mathcal{I}$ with intervals $\mathcal{I}_0 = [a_0, b_0] = [-1, 0]$ and $\mathcal{I}_{n+1} = [a_{n+1}, b_{n+1}] = [2n + 1, 2n + 2]$ such that $\omega(\mathcal{I}_0) = \omega(\mathcal{I}_{n+1}) = 0$. Clearly, $\mathcal{S} \subseteq \mathcal{I}$ is an independent dominating set of $G(\mathcal{I})$ if and only if $\mathcal{S} \cup \mathcal{I}_0 \cup \mathcal{I}_{n+1}$ is an independent dominating set of $G(\mathcal{I}')$. Hence, to solve the desired problem, it suffices to find a minimum weight independent dominating set in $G(\mathcal{I}')$.

The main idea behind the algorithm is to incrementally build a minimum weight partial independent dominating set until we find an optimal solution over the entire graph. In particular, given

a set $\mathcal{S} \subseteq \mathcal{I}$ of intervals, we say that $\mathcal{S}$ is a partial dominating set if it dominates all intervals starting before the right endpoint of $\text{last}(\mathcal{S})$. Moreover, $\mathcal{S}$ is called a partial independent dominating set if it is also an independent set. In particular, observe that all independent partial dominating sets must contain $\mathcal{I}_0$. Before specifying the main Lemma behind the algorithm, we will need a few more pieces of notation. Let Ψ_k denote the collection of all independent partial dominating sets whose last interval is $\mathcal{I}_k$, and let $\min_\omega(\Psi_k)$ denote the minimum weight independent partial dominating set in Ψ_k. Using this notation, we will provide a recursive set of formulas for calculating a minimum weight independent dominating set in an interval graph, which we characterize by the following Lemma.

Lemma 3. *The following three statements are true:*

- $\min_\omega(\Psi_0) = \{\mathcal{I}_0\}$.
- $\min_\omega(\Psi_k) = \{\mathcal{I}_k\} \cup \min_\omega(\{\min_\omega(\Psi_j) : \max_a(\Delta_f(a_k)) < b_j < a_k\})$, *for* $1 \leq k \leq n+1$.
- $\min_\omega(\Psi_{n+1})$ *is a minimum weight independent dominating set of* $G(\mathcal{I}')$.

Proof. The correctness of statement one and three follow directly by definition and so it suffices to prove the correctness of the second statement. To prove this statement, we will first show that if $\mathcal{S} \in \Psi_j$ such that $\max_a(\Delta_f(a_k)) < b_j < a_k$, then $\mathcal{S} \cup \{\mathcal{I}_k\} \in \Psi_k$. First, observe that since $\mathcal{S} \in \Psi_j$, it follows by definition that $\text{last}(\mathcal{S}) = \mathcal{I}_j$. Therefore, since $b_j < a_k$, we have that $\mathcal{I}_j \cap \mathcal{I}_k = \emptyset$ and so $\mathcal{S} \cup \{\mathcal{I}_k\}$ is an independent set. Moreover, since $\max_a(\Delta_f(a_k)) < b_j$, Lemma 2 tells us that $\mathcal{I}_k$ dominates $\Delta_s(b_k) - \Delta_s(b_j)$, and since $\mathcal{S}$ dominates $\Delta_s(b_j)$, we have that $\mathcal{S} \cup \{\mathcal{I}_k\}$ is a partial dominating set whose last interval is $\mathcal{I}_k$. Hence $\mathcal{S} \cup \{\mathcal{I}_k\} \in \Psi_k$, as required.

Now we need to prove that the reverse direction also holds true, which will complete the proof of the Lemma. In particular, we will show that if $\mathcal{S} \in \Psi_k$ and $\mathcal{I}_j = \text{last}(\mathcal{S} - \{\mathcal{I}_k\})$, then $\mathcal{S} - \{\mathcal{I}_k\} \in \Psi_j$ with the property that $\max_a(\Delta_f(a_k)) < b_j < a_k$. First, observe that since $\mathcal{S}$ is an independent set, no two intervals in $\mathcal{S}$ intersect. In particular, this means that $\mathcal{I}_j$ and $\mathcal{I}_k$ do not intersect, and so we have that $b_j < a_k$. This also implies that $\mathcal{S} - \{\mathcal{I}_k\}$ must be an independent set. Moreover, since $\mathcal{S}$ dominates $\Delta_s(b_k)$ (by definition) and since $b_j < a_k$, it follows that $\mathcal{S} - \{\mathcal{I}_k\}$ dominates $\Delta_s(b_j)$ and $\mathcal{I}_k$ dominates $\Delta_s(b_k) - \Delta_s(b_j)$. Therefore, $\mathcal{S} - \{\mathcal{I}_k\}$ is an independent partial dominating set whose last interval is $\mathcal{I}_j$, and by Lemma 2, it follows that $\max_a(\Delta_f(a_k)) < b_j$, as required. □

Using this Lemma, we can develop the desired algorithm. In particular, the algorithm will maintain a list $\mathcal{L}$ of intervals that are sorted by their right endpoint. Furthermore, it will ensure that $\mathcal{L}$ satisfies the following two properties:

- $\mathcal{L}$ is always a subset of the intervals satisfying the inequality requirement of the second statement in Lemma 3. That is, $\mathcal{L} \subseteq \{\mathcal{I}_k : \max_a(\Delta_f(e)) < b_k < e\}$, for some appropriate endpoint e that is chosen during the execution of the algorithm.
- For any two intervals $\mathcal{I}_j, \mathcal{I}_k \in \mathcal{L}$, if $j < k$ then $\omega(\min_\omega(\Psi_j)) \leq \omega(\min_\omega(\Psi_k))$. Note the abuse of notation, in which the weight of a set of intervals is simply the sum of the weight of each interval in that set.

With these properties in mind, the idea is to perform a linear scan through the set of intervals $\mathcal{I}$. Using Lemma 3, whenever a left endpoint a_k is encountered, we will calculate the minimum weight

independent partial dominating set whose last interval is $\mathcal{I}_k$. Intuitively, this will be accomplished using the head of $\mathcal{L}$, since by the properties above, it will contain the interval whose independent partial dominating set has minimum weight and satisfies the inequality requirements of Lemma 3. When a right endpoint b_k is encountered, we will remove some of the intervals from $\mathcal{L}$ so that the properties above can be maintained.

In the following algorithm, we will use δ and τ to denote the head and tail of the list $\mathcal{L}$ respectively. Moreover, we will use ω_k to denote the weight of $\min_\omega(\Psi_k)$.

Input: An augmented sorted set of intervals $\mathcal{I}'$ and a weight function $\omega : V \to \mathbb{R}^+$
Output: A minimum weight independent dominating set of $G(\mathcal{I}')$

```
1  L ← {I_0}, min_ω(Ψ_0) ← {I_0}, ω_0 ← 0;
2  for e ← 1 to 2n + 1 do
3      if e is the left endpoint of some interval I_k then
4          Set I_x ← δ, min_ω(Ψ_k) ← {I_k} ∪ min_ω(Ψ_x);
5          ω_k ← ω(I_k) + ω_x;
6      else e is the right endpoint of some interval I_k
7          Set I_x ← τ and delete all intervals from L whose right endpoints are less than a_k;
8          while L ≠ ∅ and ω_x > ω_k do
9              Delete τ from L;
10         end
11         Appened I_k to the end of L;
12     end
13 end
14 return min_ω(Ψ_{n+1})
```

To prove the correctness of the algorithm, we will first prove the claims we made regarding the properties of $\mathcal{L}$ during the execution of the algorithm.

Lemma 4. *At any point during the execution of the algorithm, the intervals in $\mathcal{L}$ are stored in increasing order of their right endpoints.*

Proof. We can see that this statement is true, as the right endpoints of each interval are visited in increasing order. Furthermore, an interval is only added to the tail τ of $\mathcal{L}$ when the right endpoint of that interval is encountered, and so $\mathcal{L}$ stores these intervals in increasing order of their right endpoint, as required. □

Lemma 5. *For any two intervals $\mathcal{I}_j$ and $\mathcal{I}_k$ in $\mathcal{L}$ such that $j < k$, we have that the weight of $\min_\omega(\Psi_j)$ is less than or equal to the weight of $\min_\omega(\Psi_k)$.*

Proof. Whenever an interval $\mathcal{I}_k$ is added at the end of $\mathcal{L}$, Lemma 4 tells us that every other interval $\mathcal{I}_j$ in $\mathcal{L}$ has the property that $j < k$. Moreover, we can observe that right before $\mathcal{I}_k$ is added to the tail of $\mathcal{L}$, lines 8 and 9 of the algorithm remove every interval from $\mathcal{L}$ whose minimum independent partial dominating set has a larger weight than the one associated with $\mathcal{I}_k$. Hence, for every other interval $\mathcal{I}_j \in \mathcal{L}$, we have that the weight of $\min_\omega(\Psi_j)$ is less than or equal to the weight of $\min_\omega(\Psi_k)$, and so, this property is maintained throughout the execution of the entire algorithm. □

Lemma 6. *Right before the endpoint e is encountered during the execution of the algorithm, we have that $\mathcal{L} \subseteq \{\mathcal{I}_j : \max_a(\Delta_f(e)) < b_j < e\}$.*

Proof. We begin by showing that, before endpoint e is encountered, $\mathcal{L}$ consists only of intervals $\mathcal{I}_j$ that satisfy $b_j < e$. This is indeed the case because an interval $\mathcal{I}_j$ is only added to $\mathcal{L}$ when its right endpoint is encountered, and by Lemma 4, the intervals in $\mathcal{L}$ are stored in increasing order of their right endpoints. Hence $\mathcal{L}$ will only contain intervals $\mathcal{I}_j$ that satisfy $b_j < e$. Now we will show that, before e is encountered, $\mathcal{L}$ consists only of intervals satisfying $\max_a(\Delta_f(e)) < b_j$, which will complete the proof. If we assume otherwise, then there exists some interval $\mathcal{I}_j \in \mathcal{L}$ satisfying $\max_a(\Delta_f(e)) > b_j$. In other words, there exists some interval $\mathcal{I}_i$ that finishes before e but whose left endpoint is larger than b_j, which implies that $\mathcal{I}_i$ is strictly between $\mathcal{I}_j$ and e. However, this causes a contradiction since $\mathcal{I}_j$ would have been removed from $\mathcal{L}$ by the algorithm on line 7 when it processed the right endpoint of $\mathcal{I}_i$. □

By combining the results of Lemma 4 and 5, we have that the minimum weight independent partial dominating set of the interval associated with δ is the smallest amongst all other intervals in $\mathcal{L}$. Furthermore, we know that by Lemma 6, whenever the left endpoint of an interval $\mathcal{I}_k$ is encountered, $\mathcal{L}$ has the form $\{\mathcal{I}_j : \max_a(\Delta_f(a_k)) < b_j < a_k\}$. Combining these two observations gives that, for $\mathcal{I}_x = \delta$, $\min_\omega(\Psi_x) = \min_\omega(\{\min_\omega(\Psi_j) : \max_a(\Delta_f(a_k)) < b_j < a_k\})$. This matches the second statement of Lemma 3, which in turn proves the correctness of our algorithm. In terms of running time, if we implement $\mathcal{L}$ and $\min_\omega(\Psi_k)$ as a doubly linked list, the union and delete operations can all be performed in constant time. Moreover, since an interval is appended to the tail exactly one time throughout the execution of the algorithm, the total number of deletions performed is at most n. Hence our algorithm runs in a linear amount of time and uses a linear amount of space. This result can be summarized by the following Theorem.

Theorem 1. *Given a set $\mathcal{I}$ of intervals sorted with respect to their right endpoints, a minimum weight independent dominating set of $G(\mathcal{I})$ can be found in $\mathcal{O}(n)$ time and $\mathcal{O}(n)$ space.*

3 Extensions In Circular-Arc Graphs

In this section, we will extend the results of the previous section to circuar-arc graphs. More specifically, the goal will be to find a minimum weight independent dominating set in $G(\mathcal{A})$. To begin with, we will arbitrarily pick an arc in $\mathcal{A}$, then starting from the head of the arc, label the endpoints of each arc from 1 to $2n$ in clockwise order. The arcs themselves will be labelled from 1 to n in increasing order of their tail. This labelling is analogous to what we did for interval graphs. Once again, we will assume that the input graph is given as a circular-arc model $\mathcal{A}$ that is labelled and sorted as described above.

The main idea behind the algorithm will be to relate finding a minimum weight independent dominating set in $\mathcal{G}(A)$ to finding a corresponding minimum weight independent dominating set in several interval graphs. Observe that for any arc $\mathcal{A}_j \in \mathcal{A}$, the graph $G(\mathcal{A} - N[\mathcal{A}_j])$ is an interval graph. The intuition behind this idea is that by removing the arc and all of its neighbours, we are creating a gap around the inner circle. Hence the arc model of $G(\mathcal{A} - N[\mathcal{A}_j])$ can simply be viewed as an interval model that has been bent around the inner circle. Moreover, this observation implies that $G(\mathcal{A} - N(\mathcal{A}_j))$ forms an interval graph with an additional isolated vertex for $\mathcal{A}_j$.

By definition, it follows that any dominating set must contain at least one of the arcs in $N[\mathcal{A}_j]$. Hence the procedure for finding a minimum weight independent dominating set in $G(\mathcal{A})$ will be as follows. Pick the arc in $G(\mathcal{A})$ that has minimum degree, which we will denote by $\mathcal{A}'$. Furthermore, let $N(\mathcal{A}') = \{\mathcal{A}'_1, \ldots, \mathcal{A}'_d\}$, where d is the degree of $\mathcal{A}'$. Since we know that the minimum weight independent dominating set of $G(\mathcal{A})$ will contain at least one of the arcs in $N[\mathcal{A}']$, we will simply run the algorithm for interval graphs once for each $\mathcal{A}'_k \in N[\mathcal{A}']$ on $G(\mathcal{A} - N(\mathcal{A}'_k))$. Then choosing the set with minimum weight amongst each iteration will clearly give us the minimum weight independent dominating set for $G(\mathcal{A})$. Summarizing this idea gives the following Theorem.

Theorem 2. *Given a set $\mathcal{A}$ of circular-arcs sorted with respect to their tails, a minimum weight independent dominating set of $G(\mathcal{A})$ can be found in $\mathcal{O}(n+m)$ time and $\mathcal{O}(n)$ space.*

Proof. We have already argued the correctness of the algorithm above, so it suffices to prove the runtime. Since $\mathcal{A} - N(\mathcal{A}_k)$ can be computed in $\mathcal{O}(n)$ time, it follows that a minimum weight independent dominating set on $G(\mathcal{A} - N(\mathcal{A}_k))$ can be computed in $\mathcal{O}(n)$ time by Theorem 1. Then repeating this procedure $d+1$ times gives a runtime of $\mathcal{O}((d+1)n)$. However, using the fact that $dn \leq \sum_{v \in G(\mathcal{A})} deg(v) \leq 2m$, it follows that the algorithm has total running time $\mathcal{O}(n+m)$ and uses $\mathcal{O}(n)$ extra space. □

4 Concluding Remarks

In these notes, we have discussed how to find a minimum weight independent dominating set in both interval and circular-arc graphs in $\mathcal{O}(n)$ and $\mathcal{O}(n+m)$ time respectively. It is worth noting that there are other kinds of dominating sets that can be found efficiently in interval and circular-arc graphs. For example, a connected dominating set of G is a dominating set $\mathcal{S}$ such that the graph induced on $\mathcal{S}$ is connected. Similarly, we can define a total dominating set of G to be a dominating set $\mathcal{S}$ such that the graph induced on $\mathcal{S}$ has no isolated vertices. In both cases, such a set can be found in polynomial time for both interval and circular-arc graphs. We refer the interested reader to [1] for more information on these algorithms.

5 Acknowledgements

I would like to thank Hisham El-Zein for his feedback and suggestions on an earlier version of these notes. His comments are greatly appreciated.

References

[1] Maw-Shang Chang. Efficient algorithms for the domination problems on interval and circular-arc graphs. *SIAM J. Comput*, 27:1671–1694, 1998.

[2] Martin Farber. Independent domination in chordal graphs. *Oper. Res. Lett.*, 1(4):134–138, September 1982.

[3] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.