Computing a Dominating Pair in an Asteroidal Triple-free Graph in Linear Time

Derek G. Corneil* Stephan Olariu† Lorna Stewart‡

January 31, 1998

Abstract

An independent set of three of vertices is called an *asteroidal triple* if between each pair in the triple there exists a path that avoids the neighborhood of the third. A graph is asteroidal triple-free (AT-free, for short) if it contains no asteroidal triple. The motivation for this work is provided, in part, by the fact that AT-free graphs offer a common generalization of interval, permutation, trapezoid, and cocomparability graphs. Previously, the authors have given an existential proof of the fact that every connected AT-free graph contains a dominating pair, that is, a pair of vertices such that every path joining them is a dominating set in the graph. The main contribution of this paper is a constructive proof of the existence of dominating pairs in connected AT-free graphs. The resulting simple algorithm can be implemented to run in time linear in the size of the input, whereas the best algorithm previously known for this problem has complexity $O(|V|^3)$ for input graph $G = (V, E)$.
Keywords: asteroidal triple-free graphs, dominating pairs, algorithms, interval graphs, permutation graphs, trapezoid graphs, cocomparability graphs

1 Introduction

Considerable attention has been paid to exploiting algorithmically different aspects of the linear structure exhibited by various families of graphs. Examples of such families include interval graphs [17], permutation graphs [13], trapezoid graphs [8, 7, 12], and cocomparability graphs [15].

Somewhat surprisingly, the linearity of these four classes is described in terms of ad-hoc properties of each of these classes of graphs. For example, in the case of interval graphs, the linearity property is traditionally expressed in terms of a linear

*Department of Computer Science, University of Toronto, Toronto, Ontario, Canada
†Department of Computer Science, Old Dominion University, Norfolk, Virginia, U.S.A.
‡Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada

order on the set of maximal cliques [5, 6]. For permutation graphs the linear behavior is explained in terms of the underlying partial order of dimension two [2], for cocomparability graphs the linear behavior is expressed in terms of the well-known linear structure of comparability graphs [16], and so on.

As it turns out, the classes mentioned above are all subfamilies of a class of graphs called the asteroidal triple-free graphs (AT-free graphs, for short). An independent triple of vertices is called an *asteroidal triple* if between any pair in the triple there exists a path that avoids the neighborhood of the third. AT-free graphs were introduced over three decades ago by Lekkerkerker and Boland [17] who showed that a graph is an interval graph if and only if it is chordal and AT-free. Thus, Lekkerkerker and Boland's result may be viewed as showing that the absence of asteroidal triples imposes the linear structure on chordal graphs that results in interval graphs. Recently, the authors [9, 11] have studied AT-free graphs with the stated goal of identifying the "agent" responsible for the linear behavior observed in the four subfamilies. Specifically, in [9, 11] we have presented evidence that the property of being asteroidal triple-free is what is enforcing the linear behavior of these classes.

One strong "certificate" of linearity is provided by the existence of a *dominating pair* – a pair of vertices with the property that every path connecting them is a dominating set. The authors have proved [9] that every connected AT-free graph contains a dominating pair. However, the proof in [9] is existential only and rather involved.

The main contribution of the present paper is to provide a constructive proof of the existence of dominating pairs in connected AT-free graphs. A remarkable feature of our approach is that the resulting algorithm is quite simple and can easily be implemented to run in time linear in the size of the input. For each of the four families mentioned above, vertices that occupy the extreme positions in the corresponding intersection model [14] constitute a dominating pair. It is interesting to note, however, that a linear time algorithm for this problem was not known even for cocomparability graphs, a strict subclass of AT-free graphs.

To put our result in perspective, we note that previously, the most efficient algorithm for finding a dominating pair in a graph $G = (V, E)$ was the straightforward $O(|V|^3)$ algorithm described in [3]. Our result shows that the task of finding a dominating pair can be solved optimally. An important spinoff of our algorithm is that it reveals a great deal of intriguing structural properties of asteroidal triple-free graphs. It is quite conceivable that this will lead to the discovery of linear-time algorithms for a number of computational problems for this class of graphs.

The remainder of this paper is organized as follows: Section 2 surveys relevant terminology and background material; Section 3 presents the details of our linear-time algorithm; Sections 4 and 5 argue about the correctness and establish the complexity of the algorithm. Finally, Section 6 offers concluding remarks and poses some open problems.

2 Background and Terminology

All the graphs in this paper are finite with neither loops nor multiple edges. In addition to standard graph theoretic terminology compatible with [4], we shall define some new terms. Given a graph $G = (V, E)$ and a set $A \subseteq V$, we let $G[A]$ denote the subgraph of G induced by A. For a vertex x, $N(x)$ denotes the set of vertices of G adjacent to x; $N'(x)$ denotes the set of vertices adjacent to x in the complement $\overline{G}$ of G.

All the paths in this work are assumed to be induced. We say that a vertex u *intercepts* a path π if u is adjacent to at least one vertex on π; otherwise, u is said to *miss* π. The *distance* between vertices u and v in a graph is the number of edges on a shortest path joining u and v.

For a connected AT-free graph with a pair of vertices x, y we let $D(x, y)$ denote the set of vertices that intercept all x, y-paths. Note that (x, y) is a dominating pair if and only if $D(x, y) = V$. In [9] the authors proved the following technical result that will be needed later in the paper.

Proposition 2.1. Let G be an asteroidal triple-free graph, let x be an arbitrary vertex of G and let C be a connected component of the subgraph of G induced by $N'(x)$. If u and v are vertices in C and $v \notin D(u, x)$, then $D(u, x) \subset D(v, x)$. □

In the remainder of this work we shall often reason about properties of the subgraph $H(x, C)$ of G induced by the set $\{x\} \cup N(x) \cup C$, where x is a distinguished vertex of G and C is a component of $N'(x)$.

We say that a vertex y of C is *special* with respect to x if $D(u, x) \subseteq D(y, x)$ for all u in C. In [9] we also showed that a vertex y of C is special with respect to x if and only if for every vertex v in C, v belongs to $D(y, x)$. In other words, (x, y) is a dominating pair in $H(x, C)$.

For a connected component C of $N'(x)$ we let S_C denote the set of vertices in $N(x)$ adjacent to at least one vertex in C. A vertex y of C is called *strong* if y is adjacent to all the vertices in S_C; otherwise y is called *weak*. A component consisting entirely of strong vertices is called *strong* and is *weak* if it contains at least one weak vertex.

Since our algorithm for finding a dominating pair in an AT-free graph relies heavily on Breadth First Search (BFS, for short), we shall now briefly review this well-known graph traversal discipline (see [1] for details). The essence of BFS is that the graph is searched as broadly as possible by visiting all the neighbors of the vertex currently being processed. BFS relies on a queue in which the vertices are inserted as they get "visited". When the vertices are dequeued, they become "processed". Initially, all vertices are both "unvisited" and "unprocessed". In our subsequent algorithms, we assume standard queue operations ENQUEUE(a), QUEUE_EMPTY, DEQUEUE. In addition, we shall find it convenient to use a few non-standard queue operations that we describe next:

- FLUSH: removes all the items in the queue;
- LOAD_QUEUE(A): involves a FLUSH operation followed by ENQUEUE(a) for all $a \in A$;
- EXTRACT(A, B, π): performs DEQUEUE repeatedly; each time an item is removed from the queue it is inserted in A or B depending on whether or not it satisfies predicate π.

From a slightly different perspective, BFS starting from an arbitrary vertex v of a graph G, partitions the vertices of G into *layers*, where all the vertices of a layer are at the same distance from v. In this paper we let $L^i(v)$ stand for the layer consisting of vertices at distance i from v. Let w be an arbitrary vertex in some layer $L^i(v)$. Every shortest w, v-path will be termed *direct*. Note that distinct vertices on a direct path belong to distinct layers. The *down-degree* of a vertex in layer $L^i(v)$ is the number of vertices in $L^{i-1}(v)$ to which it is adjacent. When performing a Breadth First layering of graph G, we let $L^*(v)$ and $L^{*-1}(v)$ denote the last and the second last layer, respectively. To keep notation simple, the reference to v in $L^i(v)$ will be dropped whenever no confusion can arise.

3 The Dominating Pair Algorithm

Our algorithm for finding a dominating pair in an AT-free graph is centered around a variant of BFS that we shall refer to as BFS*. Just as the classic BFS, BFS* searches the graph at hand, specified as $G[P \cup Q]$, until it reaches the last layer. At that moment, the (implicit) queue that governs BFS* is split into two sets P and Q. Q consists of the vertices in the queue having minimum down-degree (with respect to the current call), while P consists of all the other vertices in the queue. Next, the queue is flushed and a new BFS* begins with the queue loaded with P. The details of the procedure BFS* are spelled out as follows.

Procedure BFS* (P, Q);
  {Input: vertex-disjoint sets P and Q;
  Output: new sets P and Q}
    LOAD_QUEUE(P)
    {replace the contents of the queue with the vertices of P}
    mark all vertices in Q unvisited;
    current-layer $\leftarrow$ 0;
    $z \leftarrow$ DEQUEUE;
    while not QUEUE_EMPTY **and** not all vertices visited **do**
        while layer of z is current-layer **do**
            for each vertex w adjacent to z **do**

```
                if w in previous layer then
                    augment the down-degree of z
                else if w unvisited then
                    ENQUEUE(w)
            endfor
            z ← DEQUEUE
        endwhile
        current-layer ← current-layer +1
    endwhile
    if QUEUE_EMPTY then
        P ← ∅;
        Q ← unvisited vertices
    else
        {let π be the predicate "vertex has minimum down-degree"}
        EXTRACT(Q, P, π)
    endif
end BFS*;
```

Our dominating pair algorithm uses a function SPECIAL which, given a vertex v and graph $H(v, C)$, returns a vertex w, special with respect to v. SPECIAL calls BFS*, which performs a modified Breadth First Search on $H(v, C)$. The details follow.

```
Function SPECIAL(v, H(v, C));
 {Input: an AT-free graph H(v, C) with a distinguished vertex v;
  Output: vertex w that is special in H(v, C) with respect to v}
    P ← {v};
    Q ← H(v, C)\{v};
    while | P |> 0 and | Q |> 1 do
        BFS* (P, Q)
    endwhile;
    return an arbitrary vertex w of Q
end SPECIAL;
```

We are now in a position to specify our algorithm that finds a dominating pair in a given AT-free graph. Starting with an arbitrary vertex x, the algorithm involves two calls to SPECIAL. Each of these calls provides one of the vertices in the desired dominating pair. The details follow.

Algorithm DP(G);
{Input: connected AT-free graph G;
Output: (y, z) a dominating pair of G}
Step 1. Choose an arbitrary vertex x of G. If $N'(x) = \emptyset$ return (x, x);
Step 2. For every connected component C of $N'(x)$ compute $\mid S_C \mid$ and determine whether C is weak;
Step 3. Select a component C that minimizes $\mid S_C \mid$, breaking ties in favor of weak components;
Step 4. $y \leftarrow SPECIAL(x, H(x, C))$;
Step 5. Determine the connected components of $N'(y)$. Let X be the component that contains x;
Step 6. $z \leftarrow SPECIAL(y, H(y, X))$;
Step 7. return (y, z);
end DP;

Throughout the rest of the paper we use the following notation: L_i^j denotes the jth layer in the ith iteration; L_i^* and L_i^{*-1} denote the last and the second last layer, respectively, in the ith iteration; Q_i represents the vertices of L_i^* of minimum down-degree into L_i^{*-1} and P_i represents the rest of L_i^*. In general, $L_{i+1}^0 = P_i$ and initially $P_0 = \{x\}$, $Q_0 = N(x) \cup N'(x)$.

To help understand the function SPECIAL and the need for the iterative calls to the procedure BFS* refer to Figure 1 featuring an AT-free graph $G = (V, E)$ with $V = \{v, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o\}$. Here, $L_1^0 = P_0 = \{v\}$, $L_1^1 = \{a\}$, $L_1^2 = L_1^{*-1} = \{b, c\}$, $L_1^3 {=} L_1^* = V \setminus \{v, a, b, c\}$, $L_2^0 = P_1 = \{d\}$, $L_2^1 = \{e\}$, $L_2^2 = L_2^{*-1} = \{f, g\}$, $L_2^3 {=} L_2^* = V \setminus \{v, a, b, c, d, e, f, g\}$, $L_3^0 = P_2 = \{h\}$, $L_3^1 = \{i\}$, $L_3^2 = L_2^{*-1} = \{j, k\}$, $L_3^3 {=} L_3^* = \{l, m, n, o\}$, $L_4^0 = P_3 = \{l\}$, $L_4^1 = \{m\}$, $L_4^2 = L_4^{*-1} = \{n\}$, and $L_4^3 {=} L_4^* = \{o\}$.

4 Correctness of the Algorithm

Before proving the correctness of the DP algorithm, we show that SPECIAL does return a vertex w that is special with respect to the given vertex v. In addition to the notation introduced in the last section, we let subscript t denote the last iteration; thus L_t^* is the last layer in the last iteration.

Theorem 4.1. The vertex w returned by SPECIAL is a special vertex with respect to v.

Proof. Since $\mid P_{i+1} \cup Q_{i+1} \mid < \mid P_i \cup Q_i \mid$ for all $0 \leq i \leq t-1$, it is clear that SPECIAL terminates. We now establish a number of claims about paths in $H(v, C)$.

Claim 4.2. If a vertex p in P_i is non-adjacent to a vertex q in Q_i, then there exists

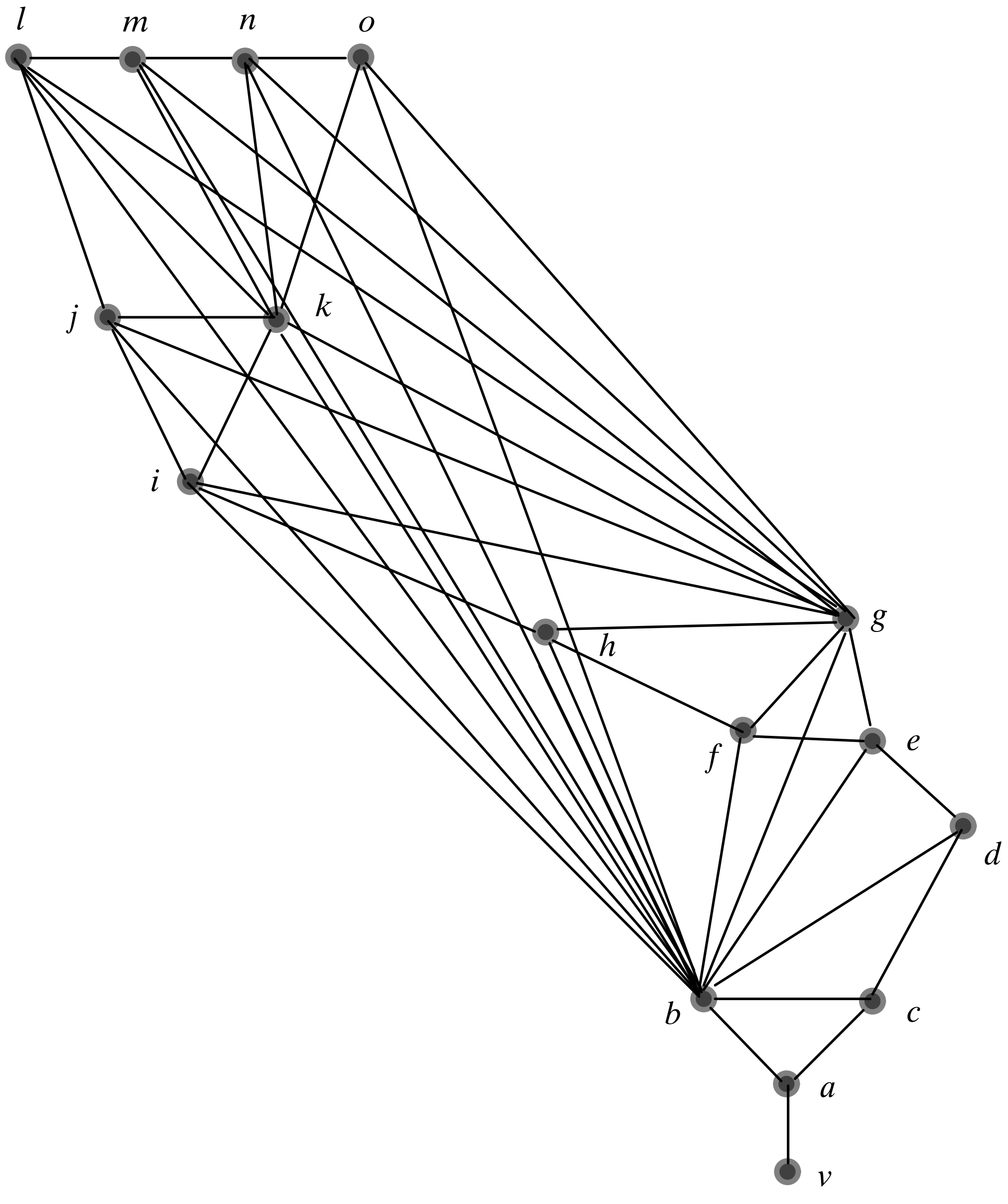

Figure 1: *An asteroidal triple-free graph G*

a path from p to a vertex in L_i^0 missed by q.

Proof. Since the down-degree of p is strictly larger than that of q, there exists vertex r in L_i^{*-1} adjacent to p but not to q. Now any shortest path in $P_{i-1} \cup Q_{i-1}$ from r to a vertex in L_i^0 is missed by q and the proof is complete. □

Claim 4.3. If a vertex p in P_i is non-adjacent to a vertex q in Q_i, then there exists a p,v-path missed by q.

Proof. By virtue of Claim 4.2, there exists a vertex p' in L_i^0 and a p,p'-path missed by q. Now the conclusion follows by an easy inductive argument, noting that p' belongs to P_{i-1} and that q belongs to Q_{i-1}. □

Claim 4.4. If a vertex r in $L_{i-1}^* \backslash L_i^*$ is non-adjacent to a vertex s in L_i^*, then there exists an r,v-path missed by s.

Proof. Straightforward application of Claim 4.3. □

Claim 4.5. Let r be an arbitrary vertex in L_i^*. No vertex in $H(v,C) \backslash L_i^*$ misses an r,v-path.

Proof. Suppose that there exists a vertex s in $H(v,C) \backslash L_i^*$ missing such a path. Clearly s does not belong to $N(v)$, and so s is in $N'(v)$. If s is in L_i^{*-1} then by Claim 4.4, there exists an s,v-path missed by r. Such a path also exists for s anywhere else in $C \backslash (L_i^* \cup L_i^{*-1})$. Thus, we have an asteroidal triple on $\{r,s,v\}$. □

We now return to the proof of Theorem 4.1. We note that SPECIAL returns a vertex in one of the following two situations:

Case 1. $P_t \neq \emptyset$ (thus $\mid Q_t \mid = 1$)

Let $Q_t = \{w\}$. Suppose there is a w,v-path missed by a vertex r. By Claim 4.5, r belongs to P_t, but now Claim 4.3 shows that $\{r,v,w\}$ is an asteroidal triple.

Case 2. $P_t = \emptyset$

Note that this may happen either because all vertices in L_t^* have the same down-degree or because BFS* did not visit all the vertices in $G[P_{t-1} \cup Q_{t-1}]$.

Subcase 2.1. $L_t^* = Q_t$

We claim that every vertex w in Q_t is special with respect to v. Suppose not, and let some vertex r miss a w,v-path π. By Claim 4.5, r belongs to Q_t. Let w' be the last vertex on π that belongs to Q_t. The next vertex on π, $\tilde{w}$ belongs to L_k^{*-1} for some $k \leq t$. But both r and w' are in Q_k and thus have the same down-degree into L_k^{*-1}. Since r and $\tilde{w}$ are not adjacent, there exists vertex $\tilde{r}$ in L_k^{*-1} adjacent to r and not adjacent to w'. Now by Claim 4.4, there exists an $\tilde{r},v$-path missed by $\tilde{w}$. Thus, $\{r,v,w'\}$ is an asteroidal triple, a contradiction.

Subcase 2.2. BFS* did not visit all vertices in $G[P_{t-1} \cup Q_{t-1}]$. Again, we claim that every vertex w in Q_t is special with respect to v. Assume to the contrary and let some vertex r miss a w, v-path π. By Claim 4.5, r does not belong to $H(v, C) \backslash L^*_{t-1}$. Furthermore, r does not belong to P_{t-1} nor may it be any vertex in L^*_{t-1} visited by BFS* starting with P_{t-1} since we may find an r, v-path missed by w. (This path would go directly, inside $L^*_{t-1} \backslash Q_t$, to a vertex in P_{t-1} and then follow a P_{t-1} to v path guaranteed by Claim 4.3.) Now, following a similar argument as in Subcase 2.1, we let w' be the last vertex on π that is in Q_t. The next vertex on π, $\tilde{w}$ belongs to L^{*-1}_k for some $k < t$. Again, both r and w' are in Q_k and the argument follows as in the previous subcase. □

We are now in a position to prove the correctness of the DP algorithm.

Theorem 4.6. The pair of vertices returned by the DP algorithm is a dominating pair in a connected AT-free graph G.

Proof. Clearly, this is true if (x, x) is returned in Step 1. We now examine the pair returned in Step 7 of the algorithm. Again, we proceed with some claims.

Claim 4.7. Let C be a component of $N'(x)$. If C is weak, then it contains a weak special vertex. Furthermore, the special vertex returned by SPECIAL is weak.

Proof. Let Y, W and S denote the set of special, weak and strong vertices in C respectively. If the first statement is false, then $Y \subseteq S$. Select a vertex w^* in W such that $D(w^*, x) \subset D(t, x)$ for no vertex t in W. If w^* does not belong to Y, then we find a vertex u in C that misses some w^*, x-path. By Proposition 2.1, $D(w^*, x) \subset D(u, x)$ thereby forcing u to belong to S. But now, we have a contradiction; u is strong, yet it misses some w^*, x-path. The fact that SPECIAL returns a weak special vertex in C follows from Theorem 4.1 and the fact that all vertices of maximum down-degree are initially placed in P and thus are eliminated from consideration as the vertex returned by SPECIAL. □

Claim 4.8. Let C be the component chosen in Step 3 and let B be any other component. Now $B \subset X$ unless both B and C are strong and $S_B = S_C$.

Proof. By the choice of C, every other component B of $N'(x)$ either has $S_B \setminus S_C \neq \emptyset$ or else $S_B = S_C$. In the former case, $B \subset X$. In the latter case, if C is strong and $S_B = S_C$, then by the choice of C, B is also strong. If C is weak, then by Claim 4.7, y, the vertex returned in Step 4, is weak and thus there exists a vertex y' in $S_C \cap N'(y)$ non-adjacent to y. Since y' belongs to S_B, B has a vertex adjacent to y' and thus $B \subset X$. □

We now finish the proof of Theorem 4.6 by showing that (y, z), the pair returned in Step 7, is a dominating pair of G. Suppose not, and let π be a y, z-path missed by some vertex w. By Theorem 4.1, w does not belong to $H(y, X)$. Thus, w either belongs to $C \backslash X$ or to some component B of $N'(x)$ where, by Claim 4.8, $S_B = S_C$ and both B and C are strong. In this case, since S_C is a cutset separating z and y, all z, y-paths must pass through S_C and thus dominate all vertices in B.

Therefore, w belongs to $C \backslash X$. Let z' be the last vertex of π in S_C and let π' be the subpath of π from z' to y. Clearly, w must miss π' and thus must also miss the x, y-path consisting of π' augmented with the edge xz', contradicting the fact that y is a special vertex with respect to x in $H(x, C)$. □

5 Complexity Considerations

We now sketch the proof of the claim that the DP algorithm may be implemented to run in linear time. With the exception of the two calls to SPECIAL, it is clear that the other steps may be implemented in linear time by using Depth First Search and standard techniques.

To show that SPECIAL can also have a linear time implementation, we show that every edge of G will be "processed" a constant number of times. By keeping appropriate counters, the last layer of the BFS layering of a connected graph can be recognized without examining the edges inside the last layer. The calculation of the down-degrees can be "charged" to the edges between the last two layers. For each call to BFS* from SPECIAL, the set P consists of vertices on the previous last layer; thus each vertex belongs to at most one P set. Finally, we note that when we are using BFS* to layer L_k^* we must remove all edges from vertices in L_k^* to previous layers. This can be accomplished through standard linking techniques. To summarize our findings we state the following result.

Theorem 5.1. Given an AT-free graph G with n vertices and m edges as input, the DP algorithm produces a dominating pair in G in $\mathrm{O}(n + m)$ time. □

6 Concluding Remarks

The class of asteroidal triple-free graphs (AT-free graphs) is a natural generalization of a number of classes of graphs including interval graphs, permutation graphs, trapezoid graphs, and cocomparability graphs. Recently the authors have argued [9] that the property of being asteroidal triple-free is responsible for the linearity featured by all these classes of graphs.

An interesting result in [9] asserts that every connected AT-free graph contains a dominating path, that is, a path with the property that every vertex outside the path is adjacent to some vertex on the path. More recently, the authors have shown [10] that a dominating path in a connected AT-free graph can be computed in linear time.

Yet another interesting concept put forth in [9] is the notion of a *dominating pair*. Specifically, a pair (x, y) of vertices of a connected AT-free graph is a dominating pair if every x, y-path is dominating. In [9] it has been shown that every connected AT-free graph contains a dominating pair. Recently, Balakrishnan *et al.* [3] have exhibited a straightforward algorithm that finds all dominating pairs in an arbitrary n-vertex graph in $O(n^3)$ time. In particular, their algorithm can be used to find a dominating pair in a connected n-vertex AT-free graph in $O(n^3)$ time. Unfortunately, for large values of n this is prohibitively expensive. Second, their algorithm does not exploit the structure of AT-free graphs.

The main contribution of this paper is to show that the task of computing a dominating pair in an AT-free graph with n vertices and m edges can be achieved in $O(n + m)$ time. An important side effect of our algorithm is to reveal intriguing structural properties of AT-free graphs. It is to be expected that these new insights will lead to efficient algorithms for a number of other computational problems for this class of graphs.

As a first step in this direction, the authors have recently shown that the algorithm presented in this paper can be extended to handle the task of computing *all* the dominating pairs in an AT-free graph of diameter at least three. We have shown that this task can be, indeed, solved in $O(n + m)$ time.

There are a number of problems that we cannot solve efficiently. Perhaps the most natural of them is the task of recognizing AT-free graphs. It is not hard to devise a recognition algorithm that, with a graph with n vertices as input, decides whether the graph is AT-free in $O(n^3)$ time – such an algorithm proceeds, essentially, by brute-force. Is it the case that the algorithm in this paper can be modified to yield an efficient recognition algorithm?

Acknowledgements

D. G. Corneil and L. Stewart wish to thank the Natural Sciences and Engineering Research Council of Canada for financial assistance. S. Olariu was supported, in part, by the National Science Foundation under grant CCR-9407180.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Reading, Massachusetts, 1983.

[2] K. A. Baker, P. C. Fishburn, and F. S. Roberts, Partial orders of dimension two, *Networks*, 2, (1971), 11–28.

[3] H. Balakrishnan, A. Rajaraman and C. Pandu Rangan, Connected domination and Steiner set on asteroidal triple-free graphs, *Proc. Workshop on Algorithms and Data Structures, WADS'93*, Montreal, Canada, August 1993, LNCS, Vol. 709, F. Dehne, J.-R. Sack, N. Santoro, S. Whitesides (Eds.), Springer-Verlag, Heidelberg, Berlin, 1993, 131–141.

[4] J. A. Bondy, U. S. R. Murty, *Graph Theory with Applications*, North-Holland, Amsterdam, 1976.

[5] K. S. Booth and G. S. Lueker, Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms, *Journal of Comput. Syst. Sci.*, 13 (1976), 335–379.

[6] K. S. Booth and G. S. Lueker, A linear time algorithm for deciding interval graph isomorphism, *Journal of the ACM*, 26 (1979), 183–195.

[7] F. Cheah, A recognition algorithm for II-graphs, Doctoral thesis, Department of Computer Science, University of Toronto, (available as technical report 246/90), 1990.

[8] D.G. Corneil and P.A. Kamula, Extensions of permutation and interval graphs, *Proceedings 18th Southeastern Conference on Combinatorics, Graph Theory and Computing* (1987), 267–276.

[9] D.G. Corneil, S. Olariu and L. Stewart, Asteroidal triple-free graphs, *Proc. 19th International Workshop on Graph Theoretic Concepts in Computer Science, WG'93*, Utrecht, The Netherlands, June 1993, LNCS, Vol. 790, J. van Leeuwen (Ed.), Springer-Verlag, Berlin, 1994, 211–224.

[10] D.G. Corneil, S. Olariu and L. Stewart, A linear time algorithm to compute a dominating path in an AT-free graph, *Information Processing Letters*, to appear.

[11] D.G. Corneil, S. Olariu and L. Stewart, Asteroidal triple-free graphs, Technical Report TR-94-31, Department of Computer Science, Old Dominion University, November, 1994.

[12] I. Degan, M.C. Golumbic and R.Y. Pinter, Trapezoid graphs and their coloring, *Discrete Applied Mathematics,* 21 (1988), 35–46.

[13] S. Even, A. Pnueli and A. Lempel, Permutation graphs and transitive graphs, *Journal of the ACM,* 19 (1972), 400-410.

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

[15] M.C. Golumbic, C.L. Monma and W.T. Trotter Jr., Tolerance graphs, *Discrete Applied Mathematics* 9 (1984), 157–170.

[16] D. Kratsch and L. Stewart, Domination on cocomparability graphs, *SIAM Journal on Discrete Mathematics*, 6 (1993), 400–417.

[17] C.G. Lekkerkerker and J.C. Boland, Representation of a finite graph by a set of intervals on the real line, *Fundamenta Mathematicae,* 51 (1962), 45-64.