algorithms

MDPI

Article

Approximating a Minimum Dominating Set by Purification

Ernesto Parra Inza [1], Nodari Vakhania [1,*], José María Sigarreta Almira [2] and José Alberto Hernández-Aguilar [3]

[1] Centro de Investigación en Ciencias, Universidad Autónoma del Estado de Morelos, Cuernavaca 62209, Morelos, Mexico; eparrainza@gmail.com
[2] Facultad de Matemáticas, Universidad Autónoma de Guerrero, Acapulco de Juárez 39650, Guerrero, Mexico; jsmathguerrero@gmail.com
[3] Facultad de Contaduría, Administración e Informática, Universidad Autónoma del Estado de Morelos, Cuernavaca 62209, Morelos, Mexico; jose_hernandez@uaem.mx
* Correspondence: nodari@uaem.mx

Abstract: A *dominating set* of a graph is a subset of vertices such that every vertex not in the subset has at least one neighbor within the subset. The corresponding optimization problem is known to be NP-hard. It is proved to be beneficial to separate the solution process in two stages. First, one can apply a fast greedy algorithm to obtain an initial dominating set and then use an iterative procedure to purify (reduce) the size of this dominating set. In this work, we develop the purification stage and propose new purification algorithms. The purification procedures that we present here outperform, in practice, the earlier known purification procedure. We have tested our algorithms for over 1300 benchmark problem instances. Compared to the estimations due to known upper bounds, the obtained solutions are about seven times better. Remarkably, for the 500 benchmark instances for which the optimum is known, the optimal solutions are obtained for 46.33% of the tested instances, whereas the average error for the remaining instances is about 1.01.

Keywords: graph theory; dominating set; approximation algorithm; time complexity

check for updates

Citation: Parra Inza, E.; Vakhania, N.; Sigarreta Almira, J.M.; Hernández-Aguilar, J.A. Approximating a Minimum Dominating Set by Purification. *Algorithms* **2024**, *17*, 258. https://doi.org/10.3390/a17060258

Academic Editor: Roberto Montemanni

Received: 14 May 2024
Revised: 31 May 2024
Accepted: 11 June 2024
Published: 12 June 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In a simple connected graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges, a subset of vertices $S \subseteq V$ is a *dominating set* in graph G if any vertex $v \in V$ is adjacent to some vertex from this subset unless vertex v itself belongs to set S. The general objective is to find an *optimal solution*, a dominating set with the minimum possible size $\gamma(G)$. The domination problem is known to be *NP-hard* [1], being among the hardest problems in the family. It is closely related to other well-known graph optimization problems such as set cover, graph coloring and independent set. The problem of domination in graphs was formalized in [2,3]. Currently, this topic has been detailed in the two well-known books by [4,5]. The theory of domination in graphs is an area of increasing interest in discrete mathematics and combinatorial computing, in particular, because of a number of important real-life applications,for example, in facility location problems in street networks [5–7], in monitoring electric power systems with the aim of minimizing stage measurement units [8], in electric power networks [9], and in the analysis of food web robustness [10]. Different versions of the graph domination problem have been utilized to model clustering in wireless ad hoc networks [11–14]. A number of other important graph-theoretic problems including set cover, maximum independent set and chromatic number are reducible to the dominance problem [5]. Dominating sets also find applications in the study of social networks [15–17].

Due to the complexity of the problem, major efforts have been made towards the development of heuristic algorithms. Among the earliest related works, we can mention that of Chvatal [18], who described an approximation algorithm with an approximation

ratio $\ln\left(\frac{n}{\gamma}\right) + 1 + \frac{1}{\gamma}$ for the related weighted set cover problem; here and later, γ is the optimal objective value. Iteratively, the algorithm selects a vertex v minimizing $\frac{w_v}{f(C\cup\{v\})-f(C)}$, where w_v is the weight of vertex v, and $f(C) = |\cup_{v\in C} N[v]|$ until $f(C) = |S|$. Adopting this algorithm, Parekh [19] solved the domination problem showing that the cardinality of the dominating set created by his algorithm is upper-bounded by $n + 1 - \sqrt{2m+1}$. Later, Eubank et al. [20] and Campan et al. [21] presented heuristic algorithms designed for special types of graphs estimating the performance of their algorithms solely with an experimental study. Recently, [22] described a heuristic algorithm with a better performance.

Little work has been carried out in the development of exact algorithms. To the best of our knowledge, the only exact algorithms which do better than a complete enumeration are described in [23–25]. In [23,24], the authors show that their algorithms run in times $O(1.4969^n)$ and $O(1.4689^n)$, respectively, without presenting experimental evidence of the practical performance of their algorithms. Although these bounds are notably better than 2^n, they remain too impractical. Quite recently, in [25], two exact algorithms, an implicit enumeration algorithm and an alternative integer linear programming (ILP) formulation were proposed. The authors also described an approximation algorithm. On the one hand, it gave solutions of notably better quality than the previously known approximation algorithms for the benchmark instances with up to 2100 vertices, some of them being solved in less than 1 min. The algorithm found an optimal solution for 61.54% of these instances. On the other hand, it failed to create solutions within a reasonable time limit for the graphs with 3000 and more vertices. In contrast, for large-scale practical problems, the heuristic from [22] gave solutions within the time limit of 50 s for graphs with up to 14,000 vertices.

Relying on the framework of the heuristic from [22], here, we propose new procedures that improve the quality of solutions for large-scaled instances. The heuristic works in two stages. In stage 1, a dominant set is generated, and in stage 2, it is *purified*, i.e., its size is reduced. The purification process is achieved through the analysis of the flowchart of stage 1, combined with a special kind of clustering of the dominating set of stage 1. A cluster exploits the special structural relationship between the vertices of a dominating set, which can beneficially be used during its purification process. Exploring the structural properties of a dominating set, we propose a new method for a more beneficial analysis of the structure of a dominating set. We present four different purification procedures, all of them outperforming, in practice, the earlier known purification procedure from [22]. For over 1300 benchmark problem instances [26], the improvement is about 10%, on average. Compared to the estimations due to the known upper bounds, the obtained solutions, on average, are about seven times better, resulting in a reduction of 85.71% of the value of the best of these upper bounds. For the 500 benchmark instances where the optimum is known (see [25,26]), at least one of the new purification procedures obtains an optimal solution for 46.33% of the instances, whereas the average error for the remained instances is 1.01.

The outline of the remaining part of the paper is as follows. In the following Section 2, we describe the necessary preliminaries. In Section 3, we give our clustering procedure. Section 4 details the four purification procedures. In Section 5, we present the approximation ratios, and in Section 6, we report our experimental results. We give some final remarks in Section 7.

2. Preliminaries

A formal description of our domination problem is as follows. Given a simple connected graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges, a subset of vertices $S \subseteq V$ is a *dominating set* in graph G if any vertex $v \in V$ is adjacent to some vertex x from this subset (i.e., there is an edge $(v, x) \in E$), unless vertex v itself belongs to set S. The objective is to find a feasible solution with the minimum possible size $\gamma(G)$.

Given a vertex $v \in V$, $N(v)$ is the set of neighbors or the open neighborhood of v in G, that is, $N(v) = \{u \in V : (u, v) \in E\}$, we denote by $\delta_G(v) = |N(v)|$ the *degree* of vertex v in G. We let $\delta_G = \min_{v \in V}\{\delta(v)\}$ and $\Delta_G = \max_{v \in V}\{\delta(v)\}$. The *private neighborhood* of vertex $v \in S \subseteq V$ is defined by $\{u \in V : N(u) \cap S = \{v\}\}$; a vertex in the private neighborhood of vertex v is said to be its *private neighbor* with respect to set S. For further details on the basic graph terminology, see [5].

In [22], a dominating set is formed in two stages. At the first stage, an initial dominating set $S = \{v_1, \ldots, v_k\}$ is created by a fast greedy algorithm, where v_i stands for the vertex included at iteration i of this greedy algorithm. At the second purification stage, a special iterative procedure is applied to reduce the initial dominating set. This iterative procedure will be referred to as the purification procedure 0, abbreviated P0. It is based on the analysis of the flowchart of the greedy algorithm, represented as a special kind of spanning forest T of the original graph G and consisting of the vertices of set S. This forest is the union of the so-called *clusters* that organize vertices from set S in special graphical structures. These structures are formed at the first stage of our algorithm, while the greedy algorithm generates set S. Clusters represent important dependencies in the dominant set that allows its purification in different ways. In particular, the order in which the vertices of each cluster are purified is important and affects the outcome of the purification process. A cluster can be seen as a specific type of spanning forest of the initial graph G. Different clustering methods yield a collection of different types of rooted trees. By a traversal of these trees, we will purify the initial dominating set at stage 2. Different combinations of a particular clustering and traversal/purification method result in different overall purification algorithms of different efficiency.

In the following sections, we describe different tree traversal methods starting with a formal definition of a cluster in the next section. Then, we describe how we carry out the purification process during a traversal of each cluster. We combine different clustering and traversal methods and obtain different purification procedures.

3. Stage 1: The Clustering

As briefly noted, the cluster C_i is a rooted tree, a sub-graph of graph G associated with some connected component of the graph induced by set S. Initially, a set of clusters, $C_1, \ldots, C_l$, form a partition of set S. We denote the union of these trees by T. In a cluster, the vertices of each connected component can be represented in different ways as trees. During the traversal of these trees, special conditions are verified, and some vertices are omitted (purified).

We generate the clusters during the construction of the initial dominating set S. This construction is based on the greedy procedure from [22], further referred to as *Greedy*. It works in a number of iterations, which is upper-bounded by n. At each iteration $h \geq 1$, one specially selected vertex, denoted by v_h, is added to the dominant set S_{h-1} of the previous iteration, i.e., $S_h := S_{h-1} \cup \{v_h\}$, where initially $S_0 = \emptyset$. S_{h-1} is the partial dominant set of iteration $h-1$. At each iteration h, the *active degree* of a vertex $v \in \overline{S_{h-1}}$ is $|N(v) \setminus [S_{h-1} \cup N(S_{h-1})]|$. At each iteration h, vertex v_h is a vertex of $\overline{S_{h-1}}$ with the maximum active degree. Note that v_1 is a vertex with the maximum degree in graph G. A vertex is said to be covered at iteration h if it has at least one neighbor in the set S_{h-1}. Greedy halts when S_h is a dominating set of G. At that iteration, all uncovered vertices with active degree 0 (if any) are included in the set S_h. A formal description is provided in Algorithm 1.

Throughout this section, v_h is used for the vertex with the maximum active degree selected by *Greedy* at iteration h; hence, $S = \{v_1, \ldots, v_k\}$ is the dominant set of Stage 1.

Algorithm 1 Algorithm Greedy

Input: A graph G.
Output: A dominating set S of G.
$h := 1$;
$i := 1$;
$S_0 := \emptyset$;
$T^0 := \emptyset$;
{ iterative step }
while S_h is not a dominating set of graph G **do**
$h := h + 1$;
$v_h :=$ any vertex of $\overline{S_{h-1}}$ with maximum active degree;
$S_h := S_{h-1} \cup \{v_h\}$;
$[T^h, i] :=$Cluster_Generation(v_h, T^{h-1}, h, i);
end while

Our aim is to *purify* this set, i.e., omit some vertices from that set so that the reduced set remains dominant. A combination of a particular clustering and traversal method results in a particular purification procedure. The four purification procedures that we propose here apply the same rules for the creation of the clusters. Unlike the purification procedure P0, which is oriented on the creation of clusters in depth, here, Stage 1 creates clusters in width. Iteratively, every next vertex v_h is included in the partial forest T^{h-1} at iteration h, $h = 1, \ldots, k$, as follows:

- Initially, we let $T^0 := \emptyset$ and $T^1 := C_1 := v_1$
- For $h > 1$,
  - If $N_{V(T^{h-1})}(v_h) = \emptyset$, then we create a new cluster $C_i := v_h$ and {update T^h} $T^h := T^{h-1} \cup C_i$
  - If $N_{V(T^{h-1})}(v_h) = \{x\}$, $x \in T$ (i.e., $|N_{V(T^{h-1})}(v_h)| = 1$), then T^h is T^{h-1} complemented by vertex v_h, a new child of vertex x
  - If $|N_{V(T^{h-1})}(v_h)| > 1$, we create a new cluster C_i with root v_h and with the immediate successors from $N_{V(T^{h-1})}(v_h)$. In this case, cluster C_i includes one or more clusters from T^{h-1} (these clusters disappear as individual clusters in forest T^h). This kind of a union may yield a cycle (see Figure 1). To avoid this, we eliminate a required number of edges from T^{h-1} incident with vertices in $N_{V(T^{h-1})}(v_h)$.

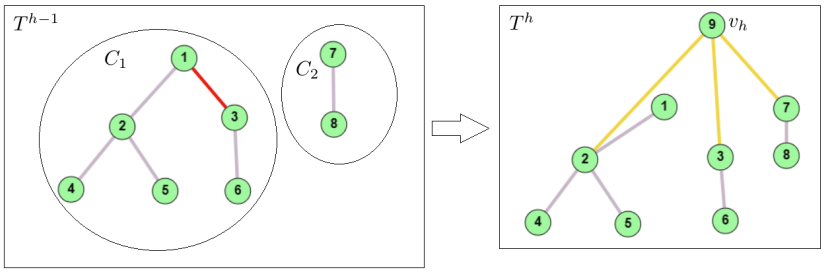


Figure 1. Forests T^{h-1} and T^h (the red edge in T^{h-1} is omitted in T^h and yellow edges are added in T^h.

Below, we give a formal description of the procedure (see Algorithm 2).

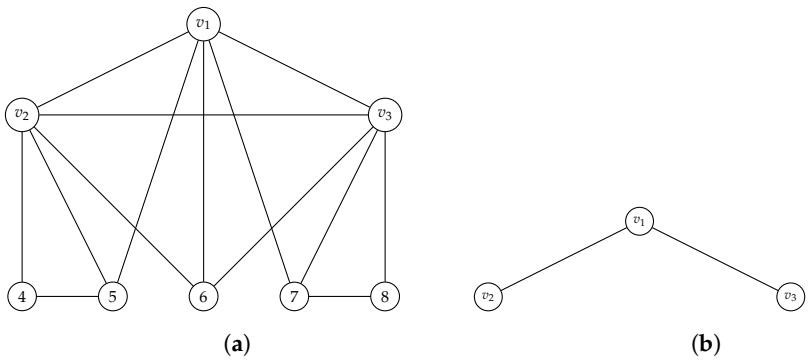
Algorithm 2 Cluster_Generation

Input: v_h, T^{h-1}, h, and number of clusters created i.
Output: Forest T^h and i .
if $h = 1$ **then**
  $C_1 := \{v_1\}$; {v_1 is root of C_1}
  $T^1 := C_1$;
else
  if $N_{V(T^{h-1})}(v_h) = \emptyset$ **then**
    $i := i + 1$;
    $C_i := \{v_h\}$; {v_h is root of C_i}
    $T^h := T^{h-1} \cup C_i$;
  else
    if $|N_{V(T^{h-1})}(v_h)| = 1$ **then**
      $x = N_{V(T^{h-1})}(v_h)$
      $T^h := T^{h-1} \cup \{x\}$ {insert v_h as a child of x};
    else
      $T^h := T^{h-1} \cup N_{V(T^{h-1})}(v_h)$ {insert $N_{V(T^{h-1})}(v_h)$ as a child of v_h and update T^h };
    end if
  end if
end if

Theorem 1. *Stage 1 runs in time* $O(n^3)$.

Proof. The number of trees C_i and the number of vertices in each tree are clearly bounded by $|V(T)| < n$. At an iteration h, $N_{V(T)}(v_h)$ can be determined in time $O(|V(T)|)$, and the same time is required to locate and update each vertex $x \in N_{V(T)}(v_h)$. Hence, at every iteration h, vertex v_h can be incorporated into the forest T^{h-1} in time $O(|V(T)| * |N_{V(T)}(v_h)|) = O(n^2)$. The theorem is followed as in Greedy, vertex v_h is determined and added in the dominating set S also in time $O(n^2)$ (see [22]). □

We use the graph in Figure 2a to illustrate our algorithms. For this graph, Greedy creates the dominating set $S = \{v_1, v_2, v_3\}$. At iteration $h = 0$, $S_0 = \emptyset$ and $T^0 = \emptyset$. At iteration $h = 1$, the vertices with the highest active degree (5) are 1, 2 and 3. So, vertex $v_1 = 1$ can be chosen. Consequently, $S_1 = \{v_1\}$ and $T^1 = \{v_1\}$. Note that the maximum active degree of the vertices in $\overline{S_{h-1}}$ is 1. Thus, in iteration $h = 2$, we can choose vertex $v_2 = 2$, resulting in $S_2 = \{v_1, v_2\}$ and $T^2 = \{v_1, v_2\}$. At iteration $h = 3$, the active degree of vertex 3 is 1, so $v_3 = 3$, $S_3 = \{v_1, v_2, v_3\}$, and $T^3 = \{v_1, v_2, v_3\}$, where T^3 is a tree with root vertex v_1, see Figure 2b. At iteration $h = 4$, S_3 is a dominating set, and the algorithm stops.



(**a**) (**b**)

Figure 2. Graph G and Cluster T. (**a**) Graph G with $\gamma(G) = 2$; (**b**) Cluster T.

4. Stage 2: Purification Procedures

Given a set of clusters formed during the execution of the algorithm of Stage 1, we wish to purify the forest T, i.e., to convert a possibly non-minimal dominating set to a

minimal one. Note that there is a difference between a minimal and a minimum dominating set. The latter one is optimal, whereas the former set is not necessarily optimal; however, it cannot be reduced by omitting any node from it. As we will see, our purification procedures return a minimal dominating set. In stage 2, Procedure P0 takes forest T as input, which, together with the initial dominating set $S = \{v_1, \ldots, v_k\}$, is constructed at stage 1. During the construction of set S, the vertex v_h is inserted into the current forest T^{h-1} as a child of one of the previously inserted vertices $v \in T^{h-1}$, such that $(v, v_h) \in E(G)$, where $v \in C$ ($C \in T^{h-1}$) is the vertex which is furthest away from the root in its cluster. If the root of another cluster in T^{h-1} is adjacent to vertex v_h in graph G, it becomes a child of vertex v_h. Note that the clusters with so-defined roots will be merged with the former cluster C into a new larger cluster.

We may observe that, in Procedure P0, the clusters are generated in depth (resulting in clusters with high depth). Dealing with clusters with high depth has some advantages but also disadvantages. In Procedure P0, each cluster is traversed in a bottom-up fashion from a leaf to the root. We will refer to a sequence of four or three consecutive vertices on a path in the currently purified tree $T^{h-1}(C_i)$ as a *quadruple* or *trio*, respectively. In Procedure P0, a quadruple (a, b, c, d) or a trio (a, b, c), forming sub-parts of the path, are identified. Given a quadruple (a, b, c, d), the vertices (b, c) are purified, unless these vertices possesses a *semi-private neighbor*, a vertex from set $V(G) \setminus T^h$, which is the only (remained) neighbor of the former vertex from set T^h. For a trio, a similar condition is verified only for vertex b.

As described above, two vertices can be purified at once. However, this might not be possible, either because there may exist no quadruple (the up going paths might be too short) or/and because a candidate vertex may possess a semi-private neighbor. It might be beneficial to set firm all vertices with a semi-private neighbor first (a firm vertex remains unpurified in the resultant dominant set). It may also be beneficial to leave a vertex possessing a considerable number of immediate descendants without semi-private neighbors unpurified in case all its immediate descendants are purified. Likewise, it might be good to leave a vertex unpurified if the number of its neighbors in $\overline{S}$ is "large enough". The above and similar considerations are taken into account in four new purification procedures.

The first three procedures P1–P3 set firm all vertices of forest T with private neighbors at iteration 0. If the so-formed set is dominating, all stop. Otherwise, three different disjoint subsets of set S forming a partition of S are distinguished, as defined below.

(1) The set of firm vertices F^h (as noted above, F^0 is the set of vertices from T with private neighbors; at iteration $h > 0$, F^h is the set of vertices from T with *semi-private* neighbors).
(2) The set of the pending vertices P^h formed by yet non-firm vertices which are neighbors of a firm vertex.
(3) Set U^h consisting of the remaining yet unconsidered vertices.

Iteratively, once F^h is determined, sets P^h and U^h are formed. The two purification parameters for each cluster vertex are defined below.

The *outer cover set* of vertex $v \in T^{h-1}$ at iteration h, $OCS(v, h)$, is the set of neighbors of v from $V \setminus S$, which are not adjacent with any firm vertex from set F^{h-1}; i.e.,

$$OCS(v, h) = N_{\overline{T^{h-1}}}(v) \setminus N_{\overline{S}}(F^{h-1}). \quad (1)$$

The *inner cover set* of vertex $v \in T^{h-1}$ at iteration h, $ICS(v, h)$, is the set of yet non-firm neighbors of vertex v in T^{h-1}, i.e.,

$$ICS(v, h) = N_{T^{h-1}}(v) \setminus F^{h-1}. \quad (2)$$

Roughly, vertices with high $|OCS(v,h)|$ and $|ICS(v,h)|$ tend to be non-purified; vice versa, direct descendants of such a vertex with low outer and inner cover indices tend to be purified. An overall purification balance of vertex v is

$$PB(v,h) = \alpha|OCS(v,h)| + \beta|ICS(v,h)|, \tag{3}$$

where $0 \leq \alpha, \beta \leq 1$.

The distinguished features of our procedures are defined below.

Procedure P1. At iteration $h > 0$, every cluster $C \in T$ is traversed in a bottom-up fashion. For every non-leaf vertex x at a given level of cluster C, if x is firm, its children without semi-private neighbors are purified. If x is not firm and has no firm children, it is set firm, and its children with no semi-private neighbors are purified, whereas the children with semi-private neighbors are set firm. If x has a firm child, the parent of x is set firm (unless it is already firm), and x is purified (see Algorithm 3):

Algorithm 3 Procedure Purify 1 (P1)

Input: Forest $T = \{C_1, C_2, \ldots, C_k\}$.
Output: S^*. { minimal dominating set }
{set vertices with private neighbors}
F^0 := {set of vertices from T with private neighbors};
if F^0 is dominating set **then**
  $S^* := F^0$;
else
  for any $C_i \in T$ **do** { in order }
    $C_i^0 := C_i$;
    $h := 0$;
    {Travel C_i by levels in ascending order}
    for any $x \notin$ leaf set in level j of C_i **do**
      if $x \in F^h$ **then**
        $C_i^{h+1} := C_i^h \setminus$ { children not a firm of x do not have semi-private neighbors};
      else
        if x has non firm children **then**
          $F^{h+1} := F^h \cup \{x\}$;
          $C_i^{h+1} := C_i^h \setminus$ {children of x do not have semi-private neighbors};
        else
          if parent of x is not in F^h **then**
            $F^{h+1} := F^h \cup$ { parent of x};
            $C_i^{h+1} := C_i^h \setminus \{x\}$;
          else
            $C_i^{h+1} := C_i^h \setminus \{x\}$;
          end if
        end if
      end if
      $h := h + 1$;
    end for
  end for
  $S^* := F^h$;
end if

Procedure P2. At any iteration $h > 1$, a non-firm vertex x with the maximum $PB(x,h)$ is set firm (for each non-firm vertex v, the sets $OCS(v,h)$ and $ICS(v,h)$ being updated, respectively). The procedure halts at iteration h if $U^h = \emptyset$ and outputs dominant set F^h (see Algorithm 4):

Algorithm 4 Procedure Purify 2 (P2)

Input: Forest $T = \{C_1, C_2, \ldots, C_k\}$.
Output: S^*. { minimal dominating set }
$F^0 :=$ {set of vertices from T with private neighbors};
$P^0 := \emptyset$;
if F^0 is dominating set **then**
  $S^* := F^0$;
else
  $h := 0$;
  for any $C_i \in T$ **do** { in order }
    while $V(C_i) \setminus (F^h \cup P^h) \neq \emptyset$ **do**
      $v :=$ any vertex with the maximum PB(v, h) in set $V(C_i) \setminus F^h$;
      $F^{h+1} := F^h \cup \{v\}$;
      $P^{h+1} := P^h \cup (N_{V(C_i)}(v) \setminus F^h)$;
      $h := h + 1$;
    end while
    if F^h is dominating set **then**
      $S^* := F^h$;
      break;
    end if
  end for
end if

Procedure P3. At any iteration $h > 1$, a vertex x with the minimum $PB(x, h)$ is purified, and the vertices from T^{h-1} with semi-private neighbors from $\overline{T}^{h-1}$ are set firm (for each non-firm vertex v, the sets OCS(v, h) and ICS(v, h) being updated, respectively). The procedure halts at iteration h if $U^h = \emptyset$ and outputs dominant set F^h (see Algorithm 5):

Algorithm 5 Procedure Purify 3 (P3)

Input: Forest $T = \{T(C_1), T(C_2), \ldots, T(C_k)\}$.
Output: S^*. { minimal dominating set }
$F^0 :=$ {set of vertices from T with private neighbors};
$P^0 := \emptyset$;
if F^0 is dominating set **then**
  $S^* := F^0$;
else
  $h := 0$;
  for any $T(C_i) \in T$ **do** { in order }
    while $T(C_i) \setminus (F^h \cup P^h) \neq \emptyset$ **do**
      $v :=$ any vertex with the minimum PB(v, h) in set $T(C_i) \setminus F^h$;
      $P^{h+1} := P^h \cup \{v\}$;
      $F^{h+1} := F^h \cup \{$all vetex in set $T \setminus (F^h \cup P^{h+1})$ that have semi-private neighbor $\}$;
      $h := h + 1$;
    end while
    if F^h is dominating set **then**
      $S^* := F^h$;
      break;
    end if
  end for
end if

Note that, for the example in Figure 2, $|S| = 3$ but $\gamma(G) = 2$; hence, set S can be purified. At iteration 0, Procedures P1–P3 set firm all vertices of forest T with private neighbors. The vertices v_2 and v_3 have as private neighbors vertices 4 and 8, respectively; so, $F^0 = \{v_2, v_3\}$. Since F^0 is a dominating set, $S^* = F^0$, and the algorithm stops.

Procedure P4. (This procedure was basically suggested by one of the anonymous referees of [22], which is highly acknowledged by the authors of this paper). The *private neighborhood* $pn(v, S)$ of vertex $v \in S \subseteq V(G)$ is

$$pn(v, S) = \{u \in V : N(u) \cap S = \{v\}\}.$$

Let a and b be adjacent vertices from set T_i, such that a is a leaf vertex added to set T_i after vertex b. Procedure P4 processes vertex a first and then vertex b, as follows (see Algorithm 6).

Algorithm 6 Procedure Purify 4 (P4)

Input: A graph $G(V, E)$ and dominating set S.
Output: S^*. { minimal dominating set }
$S^* = \{v_1, \ldots, v_p\}$; {vertices of S in the order they are added by *Greedy*.}
$h := p$;
{ iterative step }
while $h > 0$ **do**
  if $pn(v_h, S^*) = \emptyset$ **then**
    $S^* := S^* \setminus \{v_h\}$; {we purify vertex v_h}
  end if
  $h := h - 1$;
end while

We use the example of Figure 2 to demonstrate Procedure P4. Initially, $S^* = S = \{v_1, v_2, v_3\}$. In iterations $h = 3$ and $h = 2$, $pn(v_3, S^*) = 1$ and $pn(v_2, S^*) = 1$, respectively, so these vertices cannot be purified. In iteration $h = 1$, $pn(v_1, S^*) = 0$, so this vertex is purified, $S^* = \{v_2, v_3\}$ and the procedure stops. Note that all four purification procedures find an optimal solution for our example.

5. Approximation Ratios

In this section, we derive approximation ratios for our algorithms. By [22], if $|S| \leq 2$, then S is a minimum dominating set. Then, all the graphs G analyzed here will be such that $\gamma(G) > 2$. Note that G is a connected graph, and all $x \in S^*$ has at least one semi-private neighbor (S^* be the output of purification procedures). Then, $|S^*| \leq n/2$.

In the next, we derive an approximation ratio of purification procedures. By [5], $\gamma(G) \geq \frac{n}{\Delta+1}$. Then, $\frac{n}{\Delta+1} \leq |S^*| \leq \frac{n}{2}$. Since $|S^*| \geq \gamma(G)$, we obtain that an approximation ratio for purification is

$$\rho = \frac{|S^*|}{\gamma(G)} \leq \frac{\frac{n}{2}}{\frac{n}{\Delta+1}} = \frac{\Delta+1}{2}. \tag{4}$$

By [22], Algorithm Greedy achieves the approximation ratio $\rho \leq ln(\Delta + 1) + 1$. Using these approximation ratios, we obtain an overall approximation ratio for purification procedures.

$$\rho \leq \begin{cases} \frac{\Delta+1}{2}, & \text{if } 1 \leq \Delta \leq 4 \\ ln(\Delta+1)+1, & \text{otherwise.} \end{cases} \tag{5}$$

6. Experimental Results

In this section, we describe our computation experiments. We implemented our algorithms in C++ using Windows 10 operative system for 64 bits on a personal computer with Intel Core i7-9750H (2.6 GHz) and 16 GB of RAM DDR4. We generated different sets of problem instances using different pseudo-random methods for the generation of graphs. The order (the number of vertices) and the size (the number of edges) of each instance were determined randomly using the function *random*(). While creating a new edge, the

corresponding pair of currently non-adjacent vertices was selected randomly, until the corresponding amount of edges is attained. For a given order, the corresponding size was determined according to the desired density,

$$density(G) = \frac{2m}{n(n-1)}. \quad (6)$$

We employed instances with densities from 0.2 to 0.9.

We analyzed 1316 benchmark instances from [26]. A complete summary of the results for these instances can be found at [26]. Due to the space limitations, here, we summarize our results for 45 randomly selected sample instances in Tables 1 and 3, where S denotes the initial dominant set found by [22], and S^* is the purified dominant set returned by the corresponding purification procedures. The columns marked by % represent the percentage of the reduction in the number of vertices from the dominant set returned by the P0 in the dominant set returned by the purification procedures P1–P4.

Table 1 below presents the percentage of the improvement by purification procedures P1–P4 for the 45 sample instances with sizes between 800 and 15,000.

Table 1. Results for sample instances.

$\|V\|$	$\|E\|$	P0		Purification Procedures							
				P1		P2		P3		P4	
		$\|S\|$	$\|S^*\|$	$\|S^*\|$	%	$\|S^*\|$	%	$\|S^*\|$	%	$\|S^*\|$	%
840	310,396	7	5	4	20.00	4	20.00	5	0.00	4	20.00
1150	1185	500	470	431	8.30	430	8.51	430	8.51	428	8.94
1650	1711	717	660	614	6.97	609	7.73	611	7.42	607	8.03
1700	1740	725	681	634	6.90	632	7.20	633	7.05	632	7.20
1900	1977	802	754	702	6.90	698	7.43	703	6.76	697	7.56
2150	2287	909	850	787	7.41	781	8.12	784	7.76	781	8.12
2300	2407	966	907	842	7.17	835	7.94	841	7.28	835	7.94
2450	2588	1054	987	901	8.71	895	9.32	898	9.02	895	9.32
2650	2744	1147	1059	981	7.37	975	7.93	977	7.74	975	7.93
2800	2853	1225	1137	1048	7.83	1048	7.83	1044	8.18	1040	8.53
2950	3011	1277	1186	1091	8.01	1085	8.52	1089	8.18	1085	8.52
3300	3342	1440	1339	1232	7.99	1219	8.96	1221	8.81	1219	8.96
3400	3518	1480	1382	1273	7.89	1268	8.25	1269	8.18	1268	8.25
3500	3530	1537	1423	1305	8.29	1292	9.21	1296	8.92	1292	9.21
3650	3705	1589	1464	1367	6.63	1351	7.72	1356	7.38	1351	7.72
3950	4093	1688	1575	1464	7.05	1453	7.75	1455	7.62	1453	7.75
4200	4202	1832	1707	1584	7.21	1564	8.38	1565	8.32	1564	8.38
4550	4589	1963	1841	1701	7.60	1710	7.12	1701	7.60	1695	7.93
4850	4915	2111	1955	1809	7.47	1790	8.44	1796	8.13	1789	8.49
4950	5048	2164	2005	1865	6.98	1852	7.63	1853	7.58	1849	7.78
5150	5288	2204	2044	1899	7.09	1888	7.63	1890	7.53	1888	7.63
5300	5313	2293	2136	1999	6.41	1980	7.30	1982	7.21	1980	7.30
5450	5489	2369	2211	2067	6.51	2052	7.19	2055	7.06	2052	7.19
5550	5562	2424	2255	2084	7.58	2069	8.25	2074	8.03	2069	8.25
5700	5770	2463	2287	2130	6.86	2115	7.52	2119	7.35	2114	7.56
5950	6047	2614	2439	2256	7.50	2243	8.04	2246	7.91	2243	8.04
6000	6010	2614	2427	2241	7.66	2237	7.83	2232	8.03	2225	8.32
6200	6310	2694	2490	2304	7.47	2309	7.27	2289	8.07	2282	8.35
6400	6466	2769	2582	2400	7.05	2395	7.24	2388	7.51	2383	7.71
6500	6586	2804	2623	2440	6.98	2421	7.70	2424	7.59	2421	7.70
6750	6870	2952	2721	2522	7.31	2494	8.34	2499	8.16	2494	8.34
6850	6933	2969	2785	2579	7.40	2566	7.86	2566	7.86	2563	7.97
7000	7038	3043	2832	2620	7.49	2623	7.38	2611	7.80	2601	8.16
7100	7135	3067	2851	2646	7.19	2639	7.44	2633	7.65	2621	8.07
7250	7357	3080	2879	2671	7.22	2654	7.82	2656	7.75	2654	7.82

Table 1. *Cont.*

$\|V\|$	$\|E\|$	P0		Purification Procedures							
				P1		P2		P3		P4	
		$\|S\|$	$\|S^*\|$	$\|S^*\|$	%	$\|S^*\|$	%	$\|S^*\|$	%	$\|S^*\|$	%
7350	7474	3177	2944	2744	6.79	2717	7.71	2722	7.54	2717	7.71
7450	7497	3217	2980	2771	7.01	2772	6.98	2757	7.48	2751	7.68
7550	7557	3299	3093	2844	8.05	2827	8.60	2835	8.34	2821	8.79
7650	7696	3331	3095	2875	7.11	2849	7.95	2853	7.82	2846	8.05
7700	7716	3332	3096	2875	7.14	2861	7.59	2862	7.56	2855	7.78
7800	7804	3395	3176	2947	7.21	2933	7.65	2938	7.49	2924	7.93
7950	7982	3480	3260	3022	7.30	3000	7.98	3004	7.85	3000	7.98
8000	8126	3466	3215	2987	7.09	2985	7.15	2969	7.65	2957	8.02
9900	24,411,032	16	14	13	7.14	13	7.14	13	7.14	13	7.14
15,000	56,111,357	14	12	12	0	12	0	12	0	12	0

Table 2 below presents the percentage of the improvement accomplished by purification procedures P1–P4, on average, for all the tested benchmark instances.

Table 2. Average results over all instances.

% P1	% P2	% P3	% P4
9.17	9.48	8.88	9.63

On average over all the 1316 tested instances, a best solution among the four solutions generated by purification procedures P1–P4 is 9.68% smaller than the dominant set returned by P0. Purification procedures P1–P4 turn out to be more efficient for dense instances with about 40% of the improvement, though in absolute terms, the number of purified vertices for non-dense instances is higher.

Table 3 presents results for 45 sample benchmark instances from [26] for which the optima solutions are known. This table compares the quality of the solutions obtained by purification procedures P1–P4 vs. the corresponding optimal solutions and upper bounds. The complete comparative analysis for all the 500 benchmark instances with known optimal solutions can be found in [26] and is also reflected in Figure 3. Over all the tested instances, our solutions, on average, contained 1/7th part of the number of vertices established due to the best known upper bound U, an improvement of 85.71% (see Fact 1 at the end of this section). Remarkably, among the 500 instances with the known optimum, an optimal solution is generated for 46.33% of the instances, whereas the average error over all the created non-optimal solutions is 1.01 vertices.

Table 3. Comparison between results of purification procedures, $\gamma(G)$, and U.

$\|V\|$	$\|E\|$	$\gamma(G)$	P1	P2	P3	P4	U
			$\|S_1^*\|$	$\|S_2^*\|$	$\|S_3^*\|$	$\|S_4^*\|$	
100	2687	6	7	7	7	7	26
106	3744	4	4	4	4	4	18
126	4651	6	6	6	6	6	30
128	6331	4	4	4	4	4	13
134	5359	7	7	7	7	7	32
138	7470	4	5	5	5	5	13
140	6710	6	6	6	6	6	24
144	7267	5	5	5	5	5	25
148	8664	3	3	3	4	3	16
150	7950	5	5	5	5	5	27
154	8304	6	6	6	6	6	24

Table 3. *Cont.*

$\|V\|$	$\|E\|$	$\gamma(G)$	P1	P2	P3	P4	U
			$\|S_1^*\|$	$\|S_2^*\|$	$\|S_3^*\|$	$\|S_4^*\|$	
170	11,640	4	4	4	4	4	14
172	10,665	6	6	6	6	6	26
176	9978	7	7	7	7	7	39
182	12,100	6	6	6	6	6	25
194	13,675	5	5	5	5	5	34
196	14,229	6	6	6	6	6	26
198	12,936	7	7	7	7	7	42
200	16,389	4	4	4	4	4	19
204	15,169	5	5	5	5	5	35
222	20,370	4	4	4	4	4	18
224	16,923	7	7	7	7	7	43
234	22,709	4	4	4	4	4	21
238	20,902	5	5	5	5	5	39
264	27,175	5	5	5	6	5	30
326	37,878	7	7	7	7	7	60
328	45,703	4	4	4	4	4	24
330	43,714	6	6	6	6	6	35
332	41,661	5	5	5	5	5	56
342	47,142	7	7	7	7	7	36
344	50,364	4	4	4	4	4	26
370	58,531	4	4	4	4	4	28
400	65,545	6	6	6	6	6	37
458	77,296	7	7	7	7	7	79
506	95,091	6	6	6	7	6	91
538	125,734	5	5	5	5	5	47
612	140,918	7	7	7	8	7	101
714	198,539	5	5	5	5	5	117
724	199,141	7	7	7	7	7	124
790	243,725	5	5	5	5	5	130
794	240,592	7	7	7	7	7	138
840	310,396	4	4	4	5	4	67
844	306,681	7	7	7	7	7	72
870	296,197	5	7	7	7	7	145
1098	533,220	5	5	5	5	5	81

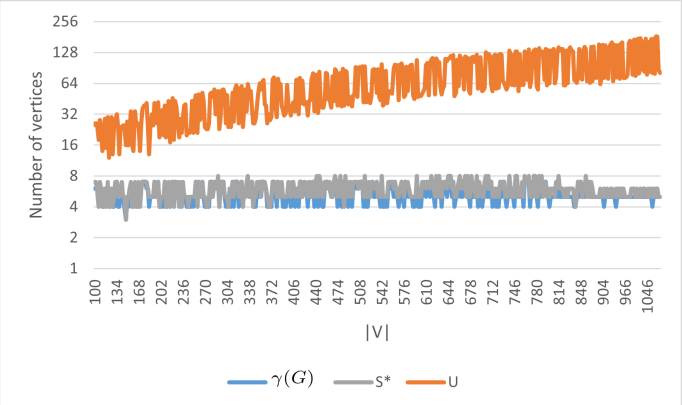
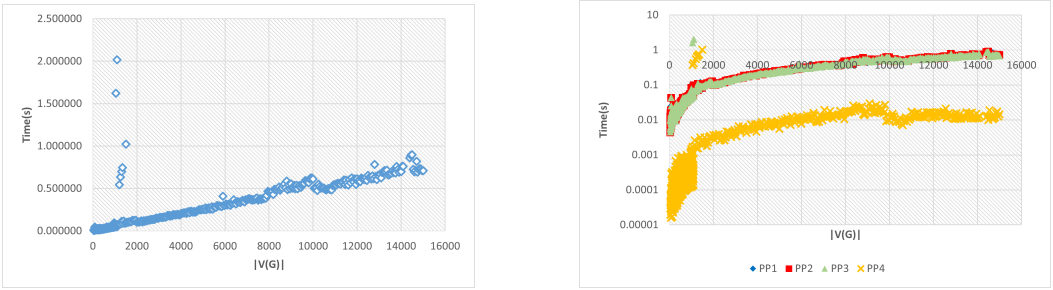


Figure 3. Results of purification procedures ($\min\{S_i^*\}$), $\gamma(G)$, and U.

As to the execution times, for the largest benchmark instances with 15,000 vertices, all four procedures finished in less than two seconds. Figure 4 represents the execution times for all the tested instances.



(**a**) (**b**)

Figure 4. Execution time of purification procedures. (**a**) Maximum execution times among P1–P4 for all instances. (**b**) Execution times for individual procedures.

The upper bound U is due to the following known results.

Fact 1 ([4]). *If $G(V, E)$ is a connected graph of order n, minimum degree $\delta(G)$ and maximum degree $\Delta(G)$, then $\gamma(G) \leq \frac{n}{2}$, $\gamma(G) \leq n - \Delta(G)$, and $\gamma(G) \leq \frac{n(\ln(\delta(G)+1))}{\delta(G)+1}$. Hence,*

$$U = \min\left\{\frac{n}{2}, n - \Delta(G), \frac{n(\ln(\delta(G)+1))}{\delta(G)+1}\right\} \tag{7}$$

is a valid upper bound on $\gamma(G)$.

7. Conclusions

We studied the purification process for the domination set problem in graphs and proposed four purification procedures that enhance solutions by 9.68% compared to the earlier state-of-the-art algorithm. Remarkably, among the 500 instances with a known optimum, optimal solutions were achieved for 46.33%, with an average error of 1.01 vertices for the remaining instances. We also derived approximation ratios for our purification procedures.

Although our purification procedures create minimal dominating sets, they are not necessarily optimal. As we can observe from our experimental study, there are problem instances, for which only one of the proposed purification procedures delivers the best obtained solution, whereas for some other instances, the best solution is delivered by another heuristic (see Tables 1 and 2). It is a challenging question to develop a single best possible purification procedure, one that, for any (non-minimal) dominating set, always omits the maximum possible number of (redundant) vertices. Another direction for future related work is to extend our procedures to more general domination and other related graph optimization problems.

Author Contributions: Investigation, E.P.I., J.M.S.A. and N.V.; writing—review and editing, E.P.I. and N.V.; Resources, UAEMor administrated by J.A.H.-A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was partial supported by funding from CCyTEM.

Data Availability Statement: https://data.mendeley.com/datasets/rr5bkj6dw5/5 (accessed on 3 April 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Garey, M.R.; Johnson, D.S. *Computers and Intractability*; Freeman: San Francisco, CA, USA, 1979; Volume 174.
2. Berge, C. *The Theory of Graphs and Its Applications*; Methuen & Co, Ltd.: London, UK, 1962.
3. Ore, O. *Theory of Graphs*; AMS Colloquium Publications: Providence, RI, USA, 1962; Volume 38.
4. Haynes, T.W.; Hedetniemi, S.T.; Slater, P.J. *Domination in Graphs*; Advanced Topics; Marcel Dekker Publications: New York, NY, USA, 1998.
5. Haynes, T.W. *Domination in Graphs*; Advanced Topics; Routledge: London, UK, 2017; Volume 2.

6. Corcoran, P.; Gagarin, A. Heuristics for k-domination models of facility location problems in street networks. *Comput. Oper. Res.* **2021**, *133*, 105368. [CrossRef]
7. Joshi, D.S.; Radhakrishnan, S.; Chandrasekharan, N. The k-neighbor, r-domination problems on interval graphs. *Eur. J. Oper. Res.* **1994**, *79*, 352–368. [CrossRef]
8. Liao, C.S.; Lee, D.T. Power domination problem in graphs. In Proceedings of the International Computing and Combinatorics Conference, Kunming, China, 16–19 August 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 818–828. [CrossRef]
9. Haynes, T.W.; Hedetniemi, S.M.; Hedetniemi, S.T.; Henning, M.A. Domination in graphs applied to electric power networks. *SIAM J. Discret. Math.* **2002**, *15*, 519–529. [CrossRef]
10. Parra Inza, E.; Sandoval Ramírez, A.; Hernández Gómez, J.; Cerdenares Ladrón de Guevara, G. Robustness Analysis of Trophic Networks using Outer k-independent Total Dominant Sets. In *Modelación Matemática IV Biomatemáticas, Epidemiología, Ingeniería*; Universidad Tecnológica de la Mixteca: Huajuapan de León, Mexico, 2021; Volume 4, pp. 3–18.
11. Balasundaram, B.; Butenko, S. Graph domination, coloring and cliques in telecommunications. In *Handbook of Optimization in Telecommunications*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 865–890. [CrossRef]
12. Wan, P.J.; Alzoubi, K.M.; Frieder, O. Distributed construction of connected dominating set in wireless ad hoc networks. *Mob. Netw. Appl.* **2004**, *9*, 141–149. [CrossRef]
13. Wu, J.; Wu, B.; Stojmenovic, I. Power-aware broadcasting and activity scheduling in ad hoc wireless networks using connected dominating sets. *Wirel. Commun. Mob. Comput.* **2003**, *3*, 425–438. [CrossRef]
14. Wu, J. Extended dominating-set-based routing in ad hoc wireless networks with unidirectional links. *IEEE Trans. Parallel Distrib. Syst.* **2002**, *13*, 866–881. [CrossRef]
15. Wang, F.; Camacho, E.; Xu, K. Positive influence dominating set in online social networks. In Proceedings of the Combinatorial Optimization and Applications: Third International Conference, COCOA 2009, Huangshan, China, 10–12 June 2009; Proceedings 3; Springer: Berlin/Heidelberg, Germany, 2009; pp. 313–321. [CrossRef]
16. Wang, F.; Du, H.; Camacho, E.; Xu, K.; Lee, W.; Shi, Y.; Shan, S. On positive influence dominating sets in social networks. *Theor. Comput. Sci.* **2011**, *412*, 265–269. [CrossRef]
17. Abu-Khzam, F.N.; Lamaa, K. Efficient heuristic algorithms for positive-influence dominating set in social networks. In Proceedings of the IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Honolulu, HI, USA, 15–19 April 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 610–615. [CrossRef]
18. Chvatal, V. A greedy heuristic for the set-covering problem. *Math. Oper. Res.* **1979**, *4*, 233–235. [CrossRef]
19. Parekh, A.K. Analysis of a greedy heuristic for finding small dominating sets in graphs. *Inf. Process. Lett.* **1991**, *39*, 237–240. [CrossRef]
20. Eubank, S.; Kumar, V.A.; Marathe, M.V.; Srinivasan, A.; Wang, N. Structural and algorithmic aspects of massive social networks. In Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, USA, 11–14 January 2004; pp. 718–727.
21. Campan, A.; Truta, T.M.; Beckerich, M. Fast Dominating Set Algorithms for Social Networks. In Proceedings of the MAICS, Greensboro, NC, USA, 25–26 April 2015; pp. 55–62.
22. Hernández Mira, F.; Parra Inza, E.; Sigarreta Almira, J.M.; Vakhania, N. A polynomial-time approximation to a minimum dominating set in a graph. *Theor. Comput. Sci.* **2022**, *930*, 142–156. [CrossRef]
23. Van Rooij, J.M.; Bodlaender, H.L. Exact algorithms for dominating set. *Discret. Appl. Math.* **2011**, *159*, 2147–2164. [CrossRef]
24. Iwata, Y. A faster algorithm for dominating set analyzed by the potential method. In Proceedings of the International Symposium on Parameterized and Exact Computation, Ljubljana, Slovenia, 12–14 September 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 41–54. [CrossRef]
25. Parra Inza, E.; Vakhania, N.; Sigarreta Almira, J.M.; Hernández Mira, F.A. Exact and heuristic algorithms for the domination problem. *Eur. J. Oper. Res.* **2024**, *313*, 926–936. [CrossRef]
26. Parra Inza, E. *Random Graph (1)*, version 5; Mendeley Data. 2024. Available online: https://data.mendeley.com/datasets/rr5bkj6dw5/5 (accessed on 3 April 2024).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.