This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier’s archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/copyright
Sort and Search: Exact algorithms for generalized domination

Fedor V. Fomin a,*, Petr A. Golovach a, Jan Kratochvíl b, Dieter Kratsch c, Mathieu Liedloff d

A R T I C L E  I N F O

Article history:
Received 19 December 2008
Received in revised form 7 February 2009
Accepted 25 March 2009
Available online 27 March 2009
Communicated by K. Iwama

Keywords:
Exact algorithms
Domination
Graph algorithms

A B S T R A C T

In 1994, Telle introduced the following notion of domination, which generalizes many domination-type graph invariants. Let σ and ϱ be two sets of non-negative integers. A vertex subset S ⊆ V of an undirected graph G = (V, E) is called a (σ, ϱ)-dominating set of G if |N(v) ∩ S| ∈ σ for all v ∈ S and |N(v) ∩ S| ∈ ϱ for all v ∈ V \ S. In this paper, we prove that decision, optimization, and counting variants of (|p|, |q|)-domination are solvable in time 2|V|/2 ⋅ |V|0(1). We also show how to extend these results for infinite sets of non-negative integers. For the case |σ| + |ϱ| = 3, these problems can be solved in time 3|V|/2 ⋅ |V|0(1), and similarly to the case |σ| = |ϱ| = 1 it is possible to extend the algorithm for some infinite sets.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

Let G = (V, E) be a finite undirected graph without loops or multiple edges. Here V is the set of vertices and E the set of edges. Throughout the paper we reserve n = |V|. We call two vertices u, v adjacent if they form an edge, i.e., if uv ∈ E. The open neighborhood of a vertex u ∈ V is the set of the vertices adjacent to it, denoted by N(u) = \{x : xu ∈ E\}. A set of vertices S ⊆ V is dominating if every vertex of G is either in S or adjacent to a vertex in S. Finding a dominating set of the smallest possible size is one of the basic optimization problems on graphs. This problem is also known to be notoriously hard. The problem is NP-hard even for chordal graphs (cf. [6]), and the parameterized version is W[2]-complete [2].

Many generalizations have been studied, such as independent dominating set, connected dominating set, efficient dominating set, etc. (cf. [6]). In [10], Telle introduced the following framework of domination-type graph invariants. Let σ and ϱ be two non-empty sets of non-negative integers. A vertex subset S ⊆ V of an undirected graph G = (V, E) is called a (σ, ϱ)-dominating set of G if |N(v) ∩ S| ∈ σ for all v ∈ S and |N(v) ∩ S| ∈ ϱ for all v ∈ V \ S. In this paper, we prove that decision, optimization, and counting variants of (|p|, |q|)-domination are solvable in time 2|V|/2 ⋅ |V|0(1). We also show how to extend these results for infinite sets of non-negative integers. For the case |σ| + |ϱ| = 3, these problems can be solved in time 3|V|/2 ⋅ |V|0(1), and similarly to the case |σ| = |ϱ| = 1 it is possible to extend the algorithm for some infinite sets.

© 2009 Elsevier B.V. All rights reserved.
It is interesting to note that already the existence problem is NP-complete for many parameter pairs \( \sigma \) and \( \varrho \), including some of those listed in Table 1 (1-perfect code and total perfect dominating set). In fact, Telle [10] proves that \( \exists (\sigma, \varrho)\)-DS is NP-complete for every two finite non-empty sets \( \sigma, \varrho \) such that \( 0 \notin \varrho \).

In this paper we show that for a sufficiently large set of decision, optimization, and even counting \((\sigma, \varrho)\)-dominating problems there are exact algorithms of running time \( O^*(2^n/2) \). Our approach is based on a classical technique of Horowitz and Sahni [7], and Schroeppel and Shamir [9] (see also the survey of Woeginger [11]). The basic idea is a clever use of sorting and searching, and thus we call it Sort and Search.

Let us briefly recall the main ideas of this paradigm.

The original problem of size \( n \), say an input graph \( G \) on \( n \) vertices, is divided into two subproblems, say two disjoint vertex subsets \( V_1 \) and \( V_2 \) of size \( n/2 \). For each subset \( S \subseteq V_1 \) (\( i \in \{1, 2\} \)) a vector of length \( n \) is assigned and stored in a table \( T_i \). The definition of the vectors is of course problem dependent. Now \( T_1 \) and \( T_2 \) contain each at most \( 2^n/2 \) different vectors. Then each solution of the problem corresponds to a vector \( \bar{a} \) of the first subproblem and a vector \( \bar{b} \) of the second one such that the sum of the two vectors is a fixed goal vector \( \bar{c} \). All such pairs \((\bar{a}, \bar{b})\) of satisfying vectors can be found by searching for each first vector \( \bar{a} \in T_1 \) the vector \( \bar{c} - \bar{a} \in T_2 \). When the vectors of the second table are sorted in lexicographic order in a preprocessing, then searching a vector can be done in \( O(n) \) times the length of the vectors, and thus the overall running time of the algorithm is \( O^*(2^n/2) \). For more details on searching in a lexicographically ordered table, we refer to vol. 3 of “The Art of Computer Programming” by Knuth [8, p. 409 ff.].

We establish \( O^*(2^n/2) \) time algorithms for the \( \exists (\sigma, \varrho)\)-DS, \( \exists (\sigma, \varrho)\)-DS, \( \exists (\sigma, \varrho)\)-DS, \( \exists (\sigma, \varrho)\)-DS, and the \( #(\sigma, \varrho)\)-DS problem in the case that \( \sigma \) and \( \varrho \) are singletons. These results are extended to infinite \( \sigma = [p + m \cdot \ell : \ell \in \mathbb{N}_0] \) and \( \varrho = [q + m \cdot \ell : \ell \in \mathbb{N}_0], \) for \( m \geq 2 \) and \( p, q \in \{0, 1, \ldots, m - 1\} \). Finally, we show that for the case \(|\sigma| + |\varrho| = 3\), these problems can be solved in time \( O^*(3^n/2) \), and similarly to the case \(|\sigma| = |\varrho| = 1\) it is possible to generalize the algorithm for some infinite sets.

Table 1 Examples of \((\sigma, \varrho)\)-dominating sets, \( N \) is the set of positive integers, \( N_0 \) is the set of non-negative integers.

<table>
<thead>
<tr>
<th>( \sigma )</th>
<th>( \varrho )</th>
<th>((\sigma, \varrho))-dominating set</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbb{N}_0 )</td>
<td>( \mathbb{N} )</td>
<td>dominating set</td>
</tr>
<tr>
<td>( (0) )</td>
<td>( N_0 )</td>
<td>independent set</td>
</tr>
<tr>
<td>( (0) )</td>
<td>( (1) )</td>
<td>efficient dominating set</td>
</tr>
<tr>
<td>( (0) )</td>
<td>( (0, 1) )</td>
<td>1-perfect code</td>
</tr>
<tr>
<td>( (0) )</td>
<td>( N )</td>
<td>strong stable set</td>
</tr>
<tr>
<td>( (1) )</td>
<td>( N_0 )</td>
<td>independent dominating set</td>
</tr>
<tr>
<td>( (1) )</td>
<td>( N )</td>
<td>total perfect dominating set</td>
</tr>
<tr>
<td>( (1) )</td>
<td>( N_0 )</td>
<td>induced matching</td>
</tr>
<tr>
<td>( f )</td>
<td>( N_0 )</td>
<td>( r )-regular induced subgraph</td>
</tr>
</tbody>
</table>

2. Sort and Search algorithms for the case \(|\sigma| = |\varrho| = 1\)

Even very special case of \( \exists (\sigma, \varrho)\)-DS, namely Perfect Code \( \exists (\{0\}, \{1\})\)-DS, is NP-complete. It is known that Perfect Code can be solved in time \( O(1.1730^n) \) by reduction to the exact satisfiability problem (called XSAT) [11]. Our use of Sort and Search is inspired by the aforementioned algorithms.

**Theorem 1.** \( \exists ([p], [q])\)-DS, \( #([p], [q])\)-DS, \( Max-([p], [q])\)-DS, and \( Min-([p], [q])\)-DS are solvable in time \( O^*(2^{n/2}) \).

**Proof.** Let \( p, q \in \mathbb{N}_0 \). Let \( G = (V, E) \) be the input graph and let \( k = \lfloor n/2 \rfloor \). As explained in the introduction, the algorithm partitions the set of vertices into \( V_1 = \{v_1, v_2, \ldots, v_k\} \) and \( V_2 = \{v_{k+1}, \ldots, v_n\} \). Then for each subset \( S_1 \subseteq V_1 \), it computes the vector \( s_1 = (x_1, x_2, \ldots, x_k, x_{k+1}, \ldots, x_n) \) where

\[
\left\{ \begin{array}{ll}
p - |N(v_i) \cap S_1| & \text{if } 1 \leq i \leq k \text{ and } v_i \in S_1, \\
q - |N(v_i) \cap S_1| & \text{if } 1 \leq i \leq k \text{ and } v_i \notin S_1, \\
|N(v_i) \cap S_1| & \text{if } k + 1 \leq i \leq n,
\end{array} \right.
\]

and for each subset \( S_2 \subseteq V_2 \), it computes the corresponding vector \( s_2 = (x_1, x_2, \ldots, x_k, x_{k+1}, \ldots, x_n) \) where

\[
\left\{ \begin{array}{ll}
|N(v_i) \cap S_2| & \text{if } 1 \leq i \leq k, \\
q - |N(v_i) \cap S_2| & \text{if } k + 1 \leq i \leq n \text{ and } v_i \in S_2, \\
p - |N(v_i) \cap S_2| & \text{if } k + 1 \leq i \leq n \text{ and } v_i \notin S_2.
\end{array} \right.
\]

After computing all these vectors (the total number of vectors is at most \( 2k^{k+1} \)), we sort vectors corresponding to \( V_2 \) lexicographically. Then for each vector \( s_1 \) representing \( S_1 \subseteq V_1 \), we use binary search to tests whether there exists a vector \( s_2 \) representing \( S_2 \subseteq V_2 \), such that \( s_1 = s_2 \). Note that the choice of the vectors guarantees that \( s_2 = s_1 \) if and only if \( S_1 \cup S_2 \) is a \( ([p], [q]) \)-dominating set. Such a vector \( s_1 \) can be found in time \( n \log^2(2/n) \) among the lexicographically ordered vectors of \( V_2 \). Thus \( \exists ([p], [q])\)-DS is solvable in \( O^*(2^{n/2}) \), the overall running time is \( O^*(2^{n/2}) \).

Now we consider \( #([p], [q])\)-DS. The algorithm of the previous theorem only needs to be modified as follows: Instead of storing all vectors corresponding to \( V_1 \) and \( V_2 \) multiple copies are removed and each vector is stored with an entry indicating its number of occurrences. Denote by \( X_1 \) the set of all different vectors corresponding to subsets of \( V_1 \), and by \( X_2 \) the set of vectors corresponding to subsets of \( V_2 \). Let \( \#_1(s_1) \) be the number of subsets of \( V_1 \) which correspond to \( s_1 \in X_1 \), and let \( \#_2(s_2) \) be the number of subsets of \( V_2 \) corresponding to \( s_2 \). As for \( \exists ([p], [q])\)-DS, for every \( s \in X_1 \), we check whether \( s \) is included to \( X_2 \) as well. Then the number of different \((\sigma, \varrho)\)-dominating sets is

\[
\sum_{s \in X_1 \times X_2} \#_1(s) \cdot \#_2(s)
\]

if \( X_1 \cap X_2 = \emptyset \), and this number is 0 otherwise.

Furthermore, for \( Max-([p], [q])\)-DS, with each vector \( s \in X_1 \) we store the subset \( S_1(s) \subseteq V_1 \) of maximum cardinality that generates this vector. It can be easily seen that a \((\sigma, \varrho)\)-dominating set of maximum size (if it exists) is the
set $S = S_1(s^2) \cup S_2(s^3)$ such that $s^2$ is a vector of $X_1 \cap X_2$ with $|S_1(s^2)| + |S_2(s^3)| = \max_{x \in x_1 \cap X_2} |S_1(s^2) + |S_2(s^3)|$. It is not hard to see that $\text{Min}-(p, |q|)-\text{DS}$ can be solved in the same way by replacing maximum by minimum. □

Now we extend our approach to contain infinite $\sigma$ and $\varrho$. Let $m \geq 2$ be a fixed integer and $k \in \{0, 1, \ldots, m-1\}$. We denote by $k + m \mathbb{N}_0$ the set $\{k + m \ell : \ell \in \mathbb{N}_0\}$.

**Theorem 2.** Let $m \geq 2$ and $p, q \in \mathbb{N}_0$. The problems $\exists(p + m \mathbb{N}_0, q + m \mathbb{N}_0)-\text{DS}$, $\exists(p + m \mathbb{N}_0, q + m \mathbb{N}_0)-\text{DS}$, $\exists(p + m \mathbb{N}_0, q + m \mathbb{N}_0)-\text{DS}$, $\exists(p + m \mathbb{N}_0, q + m \mathbb{N}_0)-\text{DS}$, and $\exists(p + m \mathbb{N}_0, q + m \mathbb{N}_0)-\text{DS}$ are solvable in time $O^*(2^{m/2})$.

**Proof.** For $\exists-(p + m \mathbb{N}_0, q + m \mathbb{N}_0)-\text{DS}$, the algorithm in Theorem 1 is modified such that for each subset $S_1 \subseteq V_1$, we compute the vector $s_1 = (x_1, x_2, \ldots, x_n)$ where

$$x_i = \begin{cases} 0 & \text{if } N(v_i) \cap S_1 \notin \text{EVEN}, \\ 1 & \text{if } N(v_i) \cap S_1 \notin \text{ODD} \end{cases}$$

and for each subset $S_2 \subseteq V_2$, the algorithm computes the corresponding vector $s_2 = (x_1, x_2, x_3, x_4, \ldots, x_n)$, where

$$x_i = \begin{cases} 0 & \text{if } v_i \notin S_2, \\ 1 & \text{if } v_i \in S_2 \end{cases}$$

Again, after computing at most $2^k \cdot 2^k$ vectors, the algorithm sorts the vectors representing $V_2$ lexicographically. By making use of binary search, for each vector $s_1$ representing $S_1 \subseteq V_1$, we search for a vector $s_2$ representing some $S_2 \subseteq V_2$, and such that $s_2 = s_1$. For $\exists-(p + m \mathbb{N}_0, q + m \mathbb{N}_0)-\text{DS}$, $\exists(p + m \mathbb{N}_0, q + m \mathbb{N}_0)-\text{DS}$, and $\exists(p + m \mathbb{N}_0, q + m \mathbb{N}_0)-\text{DS}$, the modification of the algorithm is similar to the one from Theorem 1, and we omit it here. □

The results of Theorem 2 can be used for the case when $\sigma$ and $\varrho$ are the sets of even or odd integers [3,5]. These problems are of importance in the coding theory. Let EVEN be the set of all even non-negative integers and ODD be the set of odd positive integers. It was shown in [5] that $\exists(\text{EVEN}, \text{EVEN})-\text{DS}$, $\exists(\text{EVEN}, \text{ODD})-\text{DS}$, $\exists(\text{ODD}, \text{EVEN})-\text{DS}$ and $\exists(\text{ODD}, \text{ODD})-\text{DS}$ can be solved in polynomial time while maximization and minimization problems are NP-hard. The next claim follows immediately from Theorem 2.

**Corollary 3.** For $\sigma, \varrho \in \{\text{EVEN}, \text{ODD}\}$, the problems $\exists-(\sigma, \varrho)-\text{DS}$, $\exists-(\sigma, \varrho)-\text{DS}$, and $\exists-(\sigma, \varrho)-\text{DS}$ are solvable in time $O^*(2^{k/2})$.

Variants of these problems for red/blue bipartite graphs were considered in [3]. Suppose that $G = (R, B, E)$ is a bipartite graph with $R, B$ a partition of the vertex set. Vertices of $R$ are called red and vertices of $B$ are blue. Let $S \subseteq R$ be a non-empty set of red vertices. It is said that $S$ is an even set if for every vertex $v \in B$, $|N(v)| \in \text{EVEN}$, and $S$ is an odd set if for every vertex $v \in B$, $|N(v)| \in \text{ODD}$. The proof of the following theorem is based on combining the Sort and Search approach with dynamic programming.

**Theorem 4.** Let $G = (R, B, E)$ be a red/blue bipartite graph. All even or odd sets can be counted, and maximum or minimum even or odd sets can be found in time $O^*(2^{\min(|R|/2, |B|)}) = O^*(2^{k/3})$.

**Proof.** We prove this claim for the counting problem for even sets. All other problems can be solved similarly. Let $R = \{u_1, \ldots, u_k\}$ and $B = \{v_1, \ldots, v_r\}$.

If $k/2 \leq r$, then we apply the following Sort and Search algorithm. Let $s = [k/2]$. We partition the set of vertices $R$ into $R_1 = \{u_1, \ldots, u_s\}$ and $R_2 = \{u_{s+1}, \ldots, u_k\}$. For each subset $S_1 \subseteq R_1$, we compute its corresponding vector $s_1 = (x_1, \ldots, x_n)$, where

$$x_i = \begin{cases} 0 & \text{if } |N(v_i) \cap R_1| \in \text{EVEN}, \\ 1 & \text{if } |N(v_i) \cap R_1| \in \text{ODD} \end{cases}$$

Similarly for each subset $S_2 \subseteq R_2$, we compute the corresponding vector $s_2 = (x_1, \ldots, x_n)$, such that

$$x_i = \begin{cases} 0 & \text{if } |N(v_i) \cap R_2| \in \text{EVEN}, \\ 1 & \text{if } |N(v_i) \cap R_2| \in \text{ODD} \end{cases}$$

Denote by $X_1$ the set of all different vectors corresponding to subsets of $V_1$, and by $X_2$ the set of vectors corresponding to subsets of $V_2$. Let $\#_1(s_1)$ be the number of subsets of $R_1$ corresponding to $s_1 \in X_1$, and let $\#_2(s_2)$ be the number of subsets of $R_2$ corresponding to $s_2$. After vectors are computed, we sort the vectors of $X_2$ lexicographically. For each vector $s_1 \in X_1$ we search for a vector $s_2 \in X_2$ such that $s_2 = s_1$. The total number of non-empty even sets is

$$\sum_{s_1 \in X_1 \cap X_2} \#_1(s_1) \cdot \#_2(s_2) - 1,$$

and then the running time of this procedure is $O^*(2^{|R|/2})$.

For the case $k/2 > r$, we use dynamic programming approach across the subsets. For every subset $S \subseteq R$, let $\tilde{s}(S) = (x_1, \ldots, x_r)$, where

$$x_i = \begin{cases} 0 & \text{if } |N(v_i) \cap S| \in \text{EVEN}, \\ 1 & \text{if } |N(v_i) \cap S| \in \text{ODD} \end{cases}$$

For every $i \in \{1, \ldots, k\}$, and every vector $\tilde{s} \in \mathbb{Z}_2^r$, we put

$$\#(i, \tilde{s}) = \left| \left\{ S \subseteq \{u_1, \ldots, u_k\} : \tilde{s}(S) = \tilde{s} \right\} \right|.$$ We also put $\#(0, \tilde{s}) = 0$ for all non-zero vectors $\tilde{s}$, and $\#(0, \tilde{s}) = 1$. For $i \in \{1, \ldots, k\}$, we denote by $\tilde{y}_i$ the vector $(y_1, \ldots, y_r)$, where

$$\tilde{y}_j = \begin{cases} 1 & \text{if } v_j \notin N(u_i), \\ 0 & \text{if } v_j \notin N(u_i) \end{cases}$$

Since $\#(i, \tilde{s}) = \#(i - 1, \tilde{s}) + \#(i - 1, \tilde{s} + \tilde{z}_i)$, we have that all values $\#(i, \tilde{s})$ can be computed in time $O^*(2^{|(B|})$ by a dynamic programming approach considering the values $i$ by increasing order. It remains to note that the number of non-empty even sets is $\#(k, \tilde{0}) - 1$. □
3. Extending the Sort and Search approach

It is possible to extend (albeit with a worse running time) our results for single-element sets for the case when one set contains two elements and the other set is a singleton.

Theorem 5. The problems $\exists(\sigma, q)$-DS, $\neg(\sigma, q)$-DS, $\Delta(\sigma, q)$-DS, and $\text{Min-}(\sigma, q)$-DS are solvable in time $O^{*}(3^{n/2})$ if $|\sigma| + |q| = 3$.

Proof. We prove the theorem for $\exists(\sigma, q)$-DS and $\sigma = \{p_1, p_2\}$, $q = \{q\}$. Let $G = (V, E)$ be a graph and $k = \lfloor n/2 \rfloor$. As in all algorithms above, we partition the set of vertices $V$ into $V_1 = \{v_1, v_2, \ldots, v_k\}$ and $V_2 = \{v_{k+1}, \ldots, v_n\}$. Now for every partition of $V_1$ into three sets $\{s_1, s_2, s_3\}$ (some of these sets can be empty), we compute the vector $X_1 = (x_1, x_2, \ldots, x_k, x_{k+1}, \ldots, x_n)$ where

$$ x_i = \begin{cases} p_1 - |N(v_i) \cap S_1^{(1)}| & \text{if } 1 \leq i \leq k \text{ and } v_i \in S_1^{(1)}, \\ p_2 - |N(v_i) \cap S_2^{(1)}| & \text{if } 1 \leq i \leq k \text{ and } v_i \in S_2^{(1)}, \\ q - |N(v_i) \cap S_3^{(1)}| & \text{if } 1 \leq i \leq k \text{ and } v_i \in S_3^{(1)}, \\ |N(v_i) \cap (S_1^{(1)} \cup S_2^{(1)})| & \text{if } k+1 \leq i \leq n. \end{cases} $$

Symmetrically, for each partition of $V_2$ into three sets $\{s_2^{(1)}, s_2^{(2)}, s_2^{(3)}\}$, we compute the corresponding vector $X_2 = (x_1, x_k, x_{k+1}, \ldots, x_n)$, where

$$ x_i = \begin{cases} |N(v_i) \cap (S_1^{(2)} \cup S_2^{(2)})| & \text{if } 1 \leq i \leq k, \\ p_1 - |N(v_i) \cap S_1^{(2)}| & \text{if } k+1 \leq i \leq n \text{ and } v_i \in S_1^{(2)}, \\ p_2 - |N(v_i) \cap S_2^{(2)}| & \text{if } k+1 \leq i \leq n \text{ and } v_i \in S_2^{(2)}, \\ q - |N(v_i) \cap S_3^{(2)}| & \text{if } k+1 \leq i \leq n \text{ and } v_i \in S_3^{(2)}.
\end{cases} $$

After computing these $3^{k+1}$ vectors, the algorithm sorts vectors of $V_2$ lexicographically, and for each vector $s_1$ (corresponding to a partition of $V_1$), search for a vector $s_2$ from $V_2$, such that $s_2 = s_1$. Note that $s_2 = s_1$ if and only if $(s_1^{(1)} \cup S_1^{(2)}) \cup (S_1^{(2)} \cup s_2^{(2)})$ is a ($\sigma$, $q$)-dominating set. Since the search of $s_2$ can be done in time $n \log 3^{n/2}$, we have that the overall running time of the algorithm is $O^{*}(3^{n/2})$.

The problems $\exists(\sigma, q)$-DS with $\sigma = \{p\}$ and $q = \{q_1, q_2\}$ are solved similarly. Moreover the algorithm can easily be extended to solve the counting, maximization and minimization version of the problem as it was done in Theorem 1 for single-element sets. □

The algorithms of Theorem 5 can be modified to handle some infinite sets as it was done in Theorem 2. In that case, all components of vectors are taken modulo $m$ and the addition and/or subtraction of vector components is taken modulo $m$.

Corollary 6. Let $m \geq 2$ and $p_1, p_2, q_1, q_2 \in N_0$. The problems $\exists(\sigma, q)$-DS, $\neg(\sigma, q)$-DS, $\Delta(\sigma, q)$-DS and $\text{Min-}(\sigma, q)$-DS are solvable in time $O^{*}(3^{n/2})$ for pairs of sets $\sigma = (p_1 + mN_0, p_2 + mN_0)$, $q = (q_1 + mN_0)$, $\sigma = (p_1 + mN_0, q)$.

4. Conclusion

We considered exact algorithms for $(\sigma, q)$-dominating set problems for some special sets $\sigma$ and $q$, assuming they are the same for all vertices. However it is possible to define a more general problem. Let $G$ be a graph such that for any vertex $v \in V$, two non-empty sets of non-negative integers $\sigma(v)$ and $\rho(v)$ are given. A vertex subset $S \subseteq V$ of the graph $G$ is called now a $(\sigma, q)$-dominating set of $G$ if $|N(v) \cap S| \in \sigma(v)$ for all $v \in S$ and $|N(v) \cap S| \in \rho(v)$ for all $v \in V \setminus S$. It should be noted that all of our algorithms can be adopted to solve these problems too.

A natural open question is whether $(\sigma, q)$-dominating set problem can be solved in time $(2 - \varepsilon)^n$ for some $\varepsilon > 0$ for any choice of sets $\sigma$ and $q$. It does not seem that Sort and Search can be used to settle this question. In [4], we suggested a different approach for obtaining $(2 - \varepsilon)^n$ algorithms for various choices of $\sigma$ and $q$, but we are still far from the complete answer.

References