Sort and Search: Exact algorithms for generalized domination*

Fedor V. Fomin† Petr A. Golovach† Jan Kratochvíl‡ Dieter Kratsch§
Mathieu Liedloff§

October 23, 2008

Abstract

Let σ and ϱ be two sets of non negative integers. A vertex subset $S \subseteq V$ of an undirected graph $G = (V, E)$ is called a (σ, ϱ)-dominating set of G if $|N(v) \cap S| \in \sigma$ for all $v \in S$ and $|N(v) \cap S| \in \varrho$ for all $v \in V \setminus S$. This notion introduced by Telle generalizes many domination-type graph invariants. We present algorithms to check the existence, to find maximum and minimum $(\{p\}, \{q\})$-dominating sets and to count the $(\{p\}, \{q\})$-dominating sets of a graph in time $O^*(2^{n/2})$. These results are extended to infinite $\sigma = \{p + m \cdot \ell \colon \ell \in \mathbb{N}_0\}$ and $\varrho = \{q + m \cdot \ell \colon \ell \in \mathbb{N}_0\}$. Also 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.

1 Introduction

We consider finite undirected graphs without loops or multiple edges. Thus a graph is a pair $G = (V, E)$ where V is the (finite) set of vertices and E the set of edges. The size of G is the number of vertices, and throughout the paper we reserve $n = |V|$ to denote this quantity. We call two vertices u, v *adjacent* if they form an edge, i.e., if $uv \in E$. The *open neighborhood* of a vertex $u \in V$ is the set of the vertices adjacent to it, denoted by $N(u) = \{x : xu \in E\}$. A set of vertices $S \subseteq V$ 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 belongs to the most important optimization problems on graphs. In some sense the problem is harder than typical graph invariants such as clique or independent set – the problem is NP-hard even for chordal graphs (cf. [7]), 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. [7]).

In [11], Telle introduced the following framework of domination-type graph invariants (see also [6, 8]). Let σ and ϱ be two non empty sets of non negative integers. A vertex

*Preliminary version of these results appeared in proceedings of WADS'07[4]

†Department of Informatics, University of Bergen, 5020 Bergen, Norway. E-mail: fomin|petrg@ii.uib.no

‡Department of Applied Mathematics, and Institute for Theoretical Computer Science, Charles University, Malostranské nám. 25, 118 00 Praha 1, Czech Republic. E-mail: honza@kam.ms.mff.cuni.cz

§Laboratoire d'Informatique Théorique et Appliquée, Université Paul Verlaine - Metz, 57045 Metz Cedex 01, France. E-mail: (kratsch|liedloff)@univ-metz.fr

subset $S \subseteq V$ of an undirected graph $G = (V, E)$ is called a (σ, ϱ)-*dominating set* of G if $|N(v) \cap S| \in \sigma$ for all $v \in S$ and $|N(v) \cap S| \in \varrho$ for all $v \in V \setminus S$. The following table shows a sample of previously defined and studied graph invariants which can be expressed in this framework.

σ	ϱ	(σ, ϱ)-dominating set
$\mathbb{N}_0$	$\mathbb{N}$	dominating set
$\{0\}$	$\mathbb{N}_0$	independent set
$\mathbb{N}_0$	$\{1\}$	efficient dominating set
$\{0\}$	$\{1\}$	1-perfect code
$\{0\}$	$\{0, 1\}$	strong stable set
$\{0\}$	$\mathbb{N}$	independent dominating set
$\{1\}$	$\{1\}$	total perfect dominating set
$\mathbb{N}$	$\mathbb{N}$	total dominating set
$\{1\}$	$\mathbb{N}_0$	induced matching
$\{r\}$	$\mathbb{N}_0$	r-regular induced subgraph

Table 1: Examples of (σ, ϱ)-dominating sets, $\mathbb{N}$ is the set of positive integers, $\mathbb{N}_0$ is the set of non negative integers.

We are interested in the computational complexity of decision, search and counting problems related to (σ, ϱ)-domination. Explicitly, we consider the following problems for some special sets σ and ϱ.

$\exists(\sigma, \varrho)$-DS: Does an input graph G contain a (σ, ϱ)-dominating set?

$\#$-(σ, ϱ)-DS: Given a graph G, determine the number of (σ, ϱ)-dominating sets of G.

Max-(σ, ϱ)-DS: Given a graph G, find a (σ, ϱ)-dominating set of maximum size.

Min-(σ, ϱ)-DS: Given a graph G, find a (σ, ϱ)-dominating set of minimum size.

It is interesting to note that already the existence problem is NP-complete for many parameter pairs σ and ϱ, including some of those listed in Table 1 (1-perfect code and total perfect dominating set). In fact, Telle [11] proves that $\exists(\sigma, \varrho)$-DS is NP-complete for every two finite non empty sets σ, ϱ such that $0 \notin \varrho$.

There are several ways how to deal with NP-hard problems. One may look for heuristics or approximation algorithms, or aim at speeding up the (exponential) running time of exact algorithms. The latter approach is also the goal of the present paper. We use a classical technique for the design of exact algorithms (see e.g. [10, 12]). In his seminal survey paper [12] on exact algorithms Woeginger describes how to design algorithms using this paradigm to solve the subset sum problem and the binary knapsack problem in time $O^*(2^{n/2})$ [1] (instead of time $O^*(2^n)$ by exhaustive search). The basic idea is a clever use of sorting and searching, and thus we call it Sort & 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 $S \subseteq V_i$ ($i \in \{1, 2\}$) a vector of

[1] As has recently become standard, we write $f(n) = O^*(g(n))$ if $f(n) \leq p(n) \cdot g(n)$ for some polynomial $p(n)$.

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 $\vec{a}$ of the first subproblem and a vector $\vec{b}$ of the second one such that the sum of the two vectors is a fixed goal vector $\vec{c}$, i.e. $\vec{a}+\vec{b}=\vec{c}$. All such pairs $(\vec{a},\vec{b})$ of satisfying vectors can be found by searching for each first vector $\vec{a} \in T_1$ the vector $\vec{c}-\vec{a}$ in T_2. When the vectors of the second problem 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 Volume 3 of "The Art of Computer Programming" by Knuth [9, p. 409 ff.].

We establish $O^*(2^{n/2})$ time algorithms for the $\exists(\sigma,\varrho)$-DS, Min-(σ,ϱ)-DS, Max-(σ,ϱ)-DS and the $\#$-(σ,ϱ)-DS problem in the case that σ and ϱ are singletons. These results are extended to infinite $\sigma=\{p+m\cdot\ell\colon \ell\in\mathbb{N}_0\}$ and $\varrho=\{q+m\cdot\ell\colon \ell\in\mathbb{N}_0\}$, 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.

2 A Sort & Search for the case $|\sigma|=|\varrho|=1$

We start with the use of Sort & Search for constructing algorithms for $\exists(\sigma,\varrho)$-DS. The task is to decide whether the given graph has a (σ,ϱ)-dominating set. Those problems are polynomial-time solvable or NP-complete depending on (σ,ϱ). For more details, we refer to [11].

The NP-complete $\exists(\{0\},\{1\})$-DS problem is called Perfect Code and it can be solved in time $O(1.1730^n)$ [1]. The algorithm is based on solving the exact satisfiability problem (called XSAT) and the approach has been generalized to the so-called X$_i$SAT problem. Those algorithms use Sort & Search and their running time is $O^*(2^{n/2})$. Our use of Sort & Search was inspired by the aforementioned algorithms.

Theorem 1. *The $\exists(\{p\},\{q\})$-DS problem can be solved 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 $\vec{s_1}=(x_1,\ldots,x_k,x_{k+1},\ldots,x_n)$ where

$$x_i=\begin{cases} p-|N(v_i)\cap S_1| & \text{if } 1\leq i\leq k \text{ and } v_i\in S_1\\ q-|N(v_i)\cap S_1| & \text{if } 1\leq i\leq k \text{ and } v_i\notin S_1\\ |N(v_i)\cap S_1| & \text{if } k+1\leq i\leq n,\end{cases}$$

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

$$x_i=\begin{cases} |N(v_i)\cap S_2| & \text{if } 1\leq i\leq k\\ p-|N(v_i)\cap S_2| & \text{if } k+1\leq i\leq n \text{ and } v_i\in S_2\\ q-|N(v_i)\cap S_2| & \text{if } k+1\leq i\leq n \text{ and } v_i\notin S_2.\end{cases}$$

After the computation of these at most 2^{k+1} vectors, the algorithm sorts those corresponding to V_2 in lexicographic order. Then, for each vector $\vec{s_1}$ (corresponding to an

$S_1 \subseteq V_1$) using binary search it tests whether there exists a vector $\vec{s_2}$ (corresponding to an $S_2 \subseteq V_2$), such that $\vec{s_2} = \vec{s_1}$. Note that the choice of the vectors guarantees that $\vec{s_2} = \vec{s_1}$ iff $S_1 \cup S_2$ is a $(\{p\},\{q\})$-dominating set. Such fixed vector $\vec{s_1}$ can be found in time $n \log 2^{n/2}$ in the lexicographic order of the vectors of V_2. Thus the overall running time is $O^*(2^{n/2})$. □

Corollary 2. *The problems* #-$(\{p\},\{q\})$-DS, MAX-$(\{p\},\{q\})$-DS *and* MIN-$(\{p\},\{q\})$-DS *can also be solved in time* $O^*(2^{n/2})$.

Proof. We consider at first #-$(\{p\},\{q\})$-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 $\#_1(\vec{s_1})$ be the number of subsets of V_1 which correspond to $\vec{s_1} \in X_1$, and let $\#_2(\vec{s_2})$ be the number of subsets of V_2 corresponding to $\vec{s_2}$. As for $\exists(\{p\},\{q\})$-DS, for every $\vec{s} \in X_1$, we check whether $\vec{s}$ is included to X_2 as well. Clearly, the number of different (σ,ϱ)-dominating sets is $\sum_{\vec{s} \in X_1 \cap X_2} \#_1(\vec{s}) \cdot \#_2(\vec{s})$ if $X_1 \cap X_2 \neq \emptyset$, and this number is 0 otherwise.

Furthermore for MAX-$(\{p\},\{q\})$-DS, with each vector $\vec{s} \in X_i$ we store $S_i(\vec{s}) \subseteq V_i$ of maximum cardinality that generates this vector. It can be easily seen that a (σ,ϱ)-dominating set of maximum size (if it exists) is the set $S = S_1(\vec{s^*}) \cup S_2(\vec{s^*})$ such that $|S_1(\vec{s^*})| + |S_2(\vec{s^*})| = \max_{\vec{s} \in X_1 \cap X_2} |S_1(\vec{s})| + |S_2(\vec{s})|$. Clearly, MIN-$(\{p\},\{q\})$-DS can be solved in the same way by replacing *maximum* by *minimum*. □

Now we extend our approach to certain infinite σ and ϱ. 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 $\{k + m \cdot \ell : \ \ell \in \mathbb{N}_0\}$.

Theorem 3. *Let* $m \geq 2$ *and* $p,q \in \mathbb{N}_0$. *The problems* $\exists(p+m\mathbb{N}_0, q+m\mathbb{N}_0)$-DS, #-$(p+m\mathbb{N}_0, q+m\mathbb{N}_0)$-DS, MAX-$(p+m\mathbb{N}_0, q+m\mathbb{N}_0)$-DS *and* MIN-$(p+m\mathbb{N}_0, q+m\mathbb{N}_0)$-DS *and can be solved in time* $O^*(2^{n/2})$.

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

$$x_i = \begin{cases} (p - |N(v_i) \cap S_1|) \mod m & \text{if } 1 \leq i \leq k \text{ and } v_i \in S_1 \\ (q - |N(v_i) \cap S_1|) \mod m & \text{if } 1 \leq i \leq k \text{ and } v_i \notin S_1 \\ |N(v_i) \cap S_1| \mod m & \text{if } k+1 \leq i \leq n, \end{cases}$$

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

$$x_i = \begin{cases} |N(v_i) \cap S_2| \mod m & \text{if } 1 \leq i \leq k \\ (p - |N(v_i) \cap S_2|) \mod m & \text{if } k+1 \leq i \leq n \text{ and } v_i \in S_2 \\ (q - |N(v_i) \cap S_2|) \mod m & \text{if } k+1 \leq i \leq n \text{ and } v_i \notin S_2. \end{cases}$$

As before, after the computation of these at most 2^{k+1} vectors, the algorithm sorts those corresponding to V_2 in lexicographic order. Then, for each vector $\vec{s_1}$ (corresponding to an $S_1 \subseteq V_1$) using binary search it tests whether there exists a vector $\vec{s_2}$ (corresponding to an $S_2 \subseteq V_2$), such that $\vec{s_2} = \vec{s_1}$.

For #-$(p+m\mathbb{N}_0, q+m\mathbb{N}_0)$-DS, MAX-$(p+m\mathbb{N}_0, q+m\mathbb{N}_0)$-DS and MIN-$(p+m\mathbb{N}_0, q+m\mathbb{N}_0)$-DS, this algorithm is modified in the same way as in Corollary 2. □

It can be mentioned here that results of Theorem 3 can be used for the case when σ and ϱ 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 positive integers. It was shown in [5] that $\exists(\text{EVEN}, \text{EVEN})$-DS, $\exists(\text{EVEN}, \text{ODD})$-DS, $\exists(\text{ODD}, \text{EVEN})$-DS and $\exists(\text{ODD}, \text{ODD})$-DS can be solved in polynomial time while maximization and minimization problems are NP-hard. The next claim follows immediately from Theorem 3.

Corollary 4. *For $\sigma, \varrho \in \{\text{EVEN}, \text{ODD}\}$, the problems $\#$-(σ, ϱ)-DS, MAX-(σ, ϱ)-DS and MIN-(σ, ϱ)-DS and can be solved in time $O^*(2^{n/2})$.*

Variants of these problems for red/blue bipartite graphs were considered in [3]. Suppose that G is a bipartite graph and R, B is a bipartition of the set of vertices. 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 following theorem shows that sometimes it convenient to combine the Sort & Search approach with other techniques.

Theorem 5. *Let $G = (R, B, E)$ be a bipartite red/blue graph. All even or odd sets can be counted, and maximum or minimum even or odd sets can be find in time $O^*(2^{\min\{|R|/2, |B|\}})$ (i.e. in time $O^*(2^{n/3})$).*

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

If $k/2 \leq r$ then we apply the Sort & Search algorithm. Let $s = \lfloor k/2 \rfloor$. The algorithm partitions the set of vertices R into $R_1 = \{u_1, \ldots u_s\}$ and $R_2 = \{u_{s+1}, \ldots, u_k\}$. Then for each subset $S_1 \subseteq R_1$, it computes the vector $\vec{s_1} = (x_1, \ldots, x_r)$ where

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

and similarly for each subset $S_2 \subseteq R_2$, it computes the corresponding vector $\vec{s_2} = (x_1, \ldots, x_r)$ where

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

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 $\#_1(\vec{s_1})$ be the number of subsets of R_1 which correspond to $\vec{s_1} \in X_1$, and let $\#_2(\vec{s_2})$ be the number of subsets of R_2 corresponding to $\vec{s_2}$. After the computation of these vectors, the algorithm sorts vectors of X_2 in lexicographic order. Then, for each vector $\vec{s_1} \in X_1$ using binary search it looks for a vector $\vec{s_2} \in X_2$, such that $\vec{s_2} = \vec{s_1}$. It can be easily seen that the total number of non empty even sets is $\sum_{\vec{s} \in X_1 \cap X_2} \#_1(s) \cdot \#_2(s) - 1$. Clearly, the running time of this procedure is $O^*(2^{|R|/2})$.

For the case $k/2 > r$, we use another classical technique for the design of exact algorithms (see [12]) known as the dynamic programming across the subsets. For any $S \subseteq R$, let $\vec{s}(S) = (x_1, \ldots, x_r)$ 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 any $i \in \{1,\ldots,k\}$ and any vector $\vec{s} \in \mathbb{Z}_2^r$, let $\#(i,\vec{s}) = |\{S \subseteq \{u_1,\ldots,u_i\} \colon \vec{s}(S) = \vec{s}\}|$, and assume that $\#(0,\vec{s}) = 0$ for all non zero vectors $\vec{s}$, and $\#(0,\vec{0}) = 1$ for the zero vector $\vec{0}$. Denote by $\vec{z_i}$ vector $(y_1,\ldots,y_r)$ where

$$y_j = \begin{cases} 1 & \text{if } v_j \in N(u_i) \\ 0 & \text{if } v_j \notin N(u_i) \end{cases}$$

for $i \in \{1,\ldots,k\}$. Since $\#(i,\vec{s}) = \#(i-1,\vec{s}) + \#(i-1,\vec{s}+\vec{z_i})$, all values $\#(i,\vec{s})$ can be computed in time $O^*(2^{|B|})$ by a dynamic programming. It remains to note, that the number of non empty even sets is $\#(k,\vec{0})$-1. □

3 A Sort & Search for the case $|\sigma| + |\varrho| = 3$

It is possible to extend our results for single-element sets for the case when one set contains two elements and another set is a singleton.

Theorem 6. *The problems* $\exists(\sigma,\varrho)$-DS, $\#$-(σ,ϱ)-DS, MAX-(σ,ϱ)-DS *and* MIN-(σ,ϱ)-DS *and can be solved in time* $O^*(3^{n/2})$ *if* $|\sigma| + |\varrho| = 3$.

Proof. We prove theorem for $\exists(\sigma,\varrho)$-DS and $\sigma = \{p_1,p_2\}$, $\varrho = \{q\}$. Let $k = \lfloor n/2 \rfloor$. As before, the algorithm partitions the set of vertices into $V_1 = \{v_1,v_2,\ldots v_k\}$ and $V_2 = \{v_{k+1},\ldots,v_n\}$. Now for every partition of V_1 into three sets $\{S_1^{(1)}, S_2^{(1)}, \overline{S}^{(1)}\}$ (some sets can be empty), it computes the vector $\vec{s_1} = (x_1,\ldots,x_k,x_{k+1},\ldots,x_n)$ where

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

and symmetrically, for each partition of V_2 into three sets $\{S_1^{(2)}, S_2^{(2)}, \overline{S}^{(2)}\}$, it computes the corresponding vector $\vec{s_2} = (x_1,\ldots,x_k,\, x_{k+1},\ldots,x_n)$ where

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

After the computation of at most 3^{k+1} these vectors, the algorithm sorts those corresponding to V_2 in lexicographic order. Then, for each vector $\vec{s_1}$ (corresponding to a partition of V_1) using binary search it tests whether there exists a vector $\vec{s_2}$ (corresponding to a partition of V_2), such that $\vec{s_2} = \vec{s_1}$. Note that the choice of the vectors guarantees that $\vec{s_2} = \vec{s_1}$ iff $(S_1^{(1)} \cup S_2^{(1)}) \cup (S_1^{(2)} \cup S_2^{(2)})$ is a $(\{\sigma\},\{\varrho\})$-dominating set. Such vector $\vec{s_1}$ can be found in time $n \log 3^{n/2}$ in the lexicographic order of the vectors of V_2. Thus the overall running time is $O^*(3^{n/2})$.

The problem $\exists(\sigma,\varrho)$-DS for $\sigma = \{p\}$, $\varrho = \{q_1,q_2\}$ is solved similarly, and the results for existence problems are extended for counting, maximization and minimization problems in the same way as it was done in Corollary 2 for single-element sets. □

The corresponding algorithms in Theorem 6 can be modified for some infinite sets as it was done in Theorem 3 (all components of vectors are taken modulo m and the addition and/or subtraction of vector components is taken modulo m).

Corollary 7. *Let $m \geq 2$ and $p_1, p_2, q_1, q_2 \in \mathbb{N}_0$. The problems $\exists(\sigma,\varrho)$-DS, $\#$-(σ,ϱ)-DS, MAX-(σ,ϱ)-DS and MIN-(σ,ϱ)-DS and can be solved in time $O^*(3^{n/2})$ for pairs of sets $\sigma = (p_1+m\mathbb{N}_0)\cup(p_2+m\mathbb{N}_0)$, $\varrho = q_1+m\mathbb{N}_0$ and $\sigma = p_1+m\mathbb{N}_0$, $\varrho = (q_1+m\mathbb{N}_0)\cup(q_2+m\mathbb{N}_0)$.*

4 Conclusion

We considered exact algorithm for (σ,ϱ)-dominating set domination problems for some special sets σ and ϱ, which are same for all vertices. But it is possible to define more general notion. 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 (σ,ϱ)-*dominating set* of G if $|N(v) \cap S| \in \sigma(v)$ for all $v \in S$ and $|N(v) \cap S| \in \varrho(v)$ for all $v \in V \setminus S$. It can be noted that all our results can be reformulated for this definition.

References

[1] V.DAHLÖF, P. JONSSON AND R. BEIGEL. *Algorithms for four variants of the exact satisfiability problem.* Theoretical Computer Science 320 (2004), 373–394.

[2] R. G. DOWNEY AND M. R. FELLOWS. *Parameterized complexity.* Springer-Verlag, New York, 1999.

[3] R. G. DOWNEY, M. R. FELLOWS, A. VARDI AND G. WHITTE. *The parameterized complexity of some fundamental problems in coding theory.* SIAM J. Comput. 29 (1999), 545–570.

[4] F. V. Fomin, P. Golovach, D. Kratsch, J. Kratochvil and M. Liedloff *Branch and Recharge: Exact algorithms for generalized domination* Proceedings of WADS 2007, Springer, 2007, LNCS 4619, pp. 508–519.

[5] M. M. HALLDORSSON, J. KRATOCHVIL AND J. A. TELLE. *Mod-2 independence and domination in graphs.* Internat. J. Found. Comput. Sci. 11 (2000), 355–363.

[6] M. M. HALLDORSSON, J. KRATOCHVIL AND J. A. TELLE. *Independent sets with domination constraints.* Discrete Applied Mathematics 99 (2000), 39–54.

[7] T. W. HAYNES, S. T. HEDETNIEMI, P. J. SLATER. *Fundamentals of Domination in Graphs.* Marcel Dekker, New York, 1998.

[8] P. HEGGERNES AND J. A. TELLE. *Partitioning graphs into generalized dominating sets.* Nordic Journal of Computing 5 (1998), 128–142.

[9] D. E. KNUTH. *The Art of Computer Programming, Volume 3: Sorting and Searching.* Addison-Wesley, 1997 (second edition).

[10] R. SCHROEPPEL AND A. SHAMIR. *A $T = O(2^{n/2})$, $S = O(2^{n/d})$ algorithm for certain NP-complete problems.* SIAM Journal on Computing 3 (1981), 456–464.

[11] J. A. TELLE. *Complexity of domination-type problems in graphs.* Nordic Journal of Computing 1 (1994), 157–171.

[12] G. J. Woeginger. *Exact algorithms for NP-hard problems: A survey.* Combinatorial Optimization - Eureka, You Shrink!, Springer-Verlag, 2003, Berlin, LNCS 2570, pp. 185–207.