Journal of Discrete Algorithms 9 (2011) 252–262

Contents lists available at ScienceDirect

Journal of Discrete Algorithms

www.elsevier.com/locate/jda

An exact algorithm for connected red–blue dominating set ☆

Faisal N. Abu-Khzam [a,*], Amer E. Mouawad [a], Mathieu Liedloff [b]

[a] Department of Computer Science and Mathematics, Lebanese American University, Beirut, Lebanon
[b] Laboratoire d'Informatique Fondamentale d'Orléans, Université d'Orléans, 45067 Orléans Cedex 2, France

ARTICLE INFO

Article history:
Available online 23 March 2011

Keywords:
Exact algorithms
Dominating set
Weighted Steiner tree

ABSTRACT

In the Connected Red–Blue Dominating Set problem we are given a graph G whose vertex set is partitioned into two parts R and B (red and blue vertices), and we are asked to find a connected subgraph induced by a subset S of B such that each red vertex of G is adjacent to some vertex in S. The problem can be solved in $\mathcal{O}^*(2^{n-|B|})$ time by reduction to the Weighted Steiner Tree problem. Combining exhaustive enumeration when $|B|$ is small with the Weighted Steiner Tree approach when $|B|$ is large, solves the problem in $\mathcal{O}^*(1.4143^n)$. In this paper we present a first non-trivial exact algorithm whose running time is in $\mathcal{O}^*(1.3645^n)$. We use our algorithm to solve the Connected Dominating Set problem in $\mathcal{O}^*(1.8619^n)$. This improves the current best known algorithm, which used sophisticated run-time analysis via the measure and conquer technique to solve the problem in $\mathcal{O}^*(1.8966^n)$.

1. Introduction

Given a graph $G = (V, E)$, the Dominating Set problem asks for a minimum subset of V whose neighborhood is its complement in V. Several variations of this problem have recently become of significant theoretical interest because of their numerous applications. Problems like directed domination, multiple domination, distance domination, independent domination and connected domination are now common in the literature, and illustrate the growing interest and the increasing work on different variants of the domination problem.

The Connected Dominating Set problem (CDS) requires that the dominating set induces a connected subgraph. CDS received considerable attention lately, due to its applications in wireless ad hoc networks, in which a connected dominating set is used as an underlying structure for performing various functions. Examples include protocols for location-based routing, multicast/broadcast and energy conservation (see [3,11] for surveys on some of the algorithms and techniques for computing connected dominating sets in wireless ad hoc networks).

CDS is $\mathcal{NP}$-complete [10], remains so even when restricted to planar graphs [4], and is not fixed parameter tractable in arbitrary graphs [5]. Most techniques discussed in [3] use approximation algorithms, which is not surprising when the best known exact algorithm has running time in $\mathcal{O}^*(1.8966^n)$[1] [7] on graphs of order n. In fact, the algorithm proposed in [7] solves the Maximum-Leaf Spanning Tree problem, which is equivalent to CDS. In [9], Fomin et al. presented an $\mathcal{O}^*(1.94^n)$ algorithm for a direct solution to the problem, being the first exact algorithm breaking the 2^n barrier.

☆ This research has been supported in part by the research council of the Lebanese American University.

* Corresponding author.
E-mail addresses: faisal.abukhzam@lau.edu.lb (F.N. Abu-Khzam), amer.mouawad@lau.edu.lb (A.E. Mouawad), mathieu.liedloff@univ-orleans.fr (M. Liedloff).
URLs: http://www.csm.lau.edu.lb/fabukhzam (F.N. Abu-Khzam), http://www.univ-orleans.fr/lifo/Members/Mathieu.Liedlof (M. Liedloff).

[1] Throughout this paper we use the modified big-Oh notation that suppresses all polynomially bounded factors. For functions f and g we say $f(n) \in \mathcal{O}^*(g(n))$ if $f(n) \in \mathcal{O}(g(n)poly(n))$, where $poly(n)$ is a polynomial.

1570-8667/$ – see front matter
doi:10.1016/j.jda.2011.03.006

In this paper we consider the Connected Dominating Set problem applied to red–blue graphs. A graph $G = (V, E)$ is a red–blue graph whenever the vertex set V is partitioned into two sets R and B. The elements of R and B are called red and blue vertices respectively. A connected red–blue dominating set of G is a subset S of the blue vertices such that S dominates all red vertices of G and the subgraph of G induced by S is connected. The corresponding problem, denoted by CRBDS, asks to find a connected red–blue dominating set of smallest possible cardinality.

CRBDS can effectively replace the more general CDS in a large number of network protocols. Naturally, "powerful" computers in a network (i.e. servers) could be represented by blue nodes, while destination hosts (i.e. laptops) could be represented by red nodes thus making dominating set based routing more efficient. In power management, CRBDS would be more appropriate than CDS since nodes in sleep mode can be mapped to red nodes while blue nodes would preserve the ability of the network to forward messages. Despite the many natural applications of CRBDS, the problem seems to have been neglected prior to this work. The connectivity property required in both CDS and CRBDS makes them part of the family of non-local problems which are usually harder to solve exactly. Nevertheless, we shall first present an exact algorithm that solves the CRBDS problem in $\mathcal{O}^*(1.3645^n)$ time and polynomial space and then prove that CDS is solvable in $\mathcal{O}^*(1.8619^n)$ by reduction to CRBDS.

2. Preliminaries

Throughout this paper we denote by $G = (R \cup B, E)$ a red–blue graph whose vertex set $V = R \cup B$. For a vertex $v \in B$, we denote by $N_R(v)$ the set of red neighbors of v. The set $N_B(v)$ is defined analogously when $v \in R$. The Red–Blue Dominating Set problem, henceforth RBDS, is defined formally as follows:

Given: a red–blue graph $G = (R \cup B, E)$ and a positive integer k.
Question: does B have a subset S such that $|S| \leqslant k$ and $N_R(S) = R$?

RBDS is $\mathcal{NP}$-complete [6], even if the input is restricted to planar graphs [1]. We note here that the technique described in [13] for Dominating Set can also be adopted for RBDS and yields a worst-case running time in $\mathcal{O}^*(1.2302^n)$. In fact, the mentioned algorithm reduces the Dominating Set problem to Minimum Set Cover and has to keep track of an instance of size $2n$ that contains both the vertices and their neighborhoods. In the case of RBDS, the run-time is in $\mathcal{O}^*(1.2302^n)$ because the number of vertices and neighborhoods is bounded by n only. A slightly better asymptotic bound was obtained recently by van Rooij et al. in [14].

In this paper, we consider a variant of both CDS and RBDS, namely CRBDS, in which the required dominating set induces a connected subgraph. CRBDS is $\mathcal{NP}$-complete since it is equivalent to RBDS when B, the set of blue vertices, forms a fully connected subset. We shall see that CRBDS can be reduced to the Weighted Steiner Tree problem WST, which can be solved in $\mathcal{O}^*(2^k)$ and polynomial space [12] where k is the number of terminal nodes (the red nodes in a CRBDS instances). Our proposed algorithm uses the following "either-or" approach.

- While favorable branching rules exist, the algorithm proceeds by applying the branch and reduce paradigm. Favorable branching rules imply that the $\mathcal{O}^*(1.3645^n)$ upper bound will never be exceeded.
- When the algorithm reaches a search state where favorable branching rules are no longer possible, an instance of the Weighted Steiner Tree problem is generated to solve the remaining part of the problem.

This either-or approach can be applied, potentially, to other problems whenever the absence of favorable branching conditions leads to a reduction of the input instance to some manageable size. To simplify our algorithm and its analysis, in the sequel, we assume that:

- R is an independent set.
- B is a dominating set of G (otherwise, we have a no instance).
- B induces a connected component of G.

If G contains distinct blue connected components, the algorithm is applied to each connected component that dominates all red vertices.

The following lemma is needed in the analysis of our algorithm. It does provide us with a breakpoint, as to when to stop branching and apply the WST algorithm.

Lemma 1. *Let $G = (R \cup B, E)$ be a red–blue bipartite graph such that*:

1. *Every blue vertex has degree four or less and every red vertex has degree two or more.*
2. *For $d \in \{2, 3, 4\}$, every red vertex of degree d has at most one blue neighbor of degree d while every other blue neighbor has a smaller degree.*
3. *No two degree-two vertices in R can have a common neighbor in B.*

4. *When a degree-three vertex and a degree-two vertex in R have a common neighbor in B, the degree-three red vertex can have no degree-three blue neighbor.*
5. *Every degree-three vertex in R has at most two common neighbors with degree-two red vertices.*

Whenever all the stated properties are satisfied then $|B| \geqslant \frac{5}{4}|R|$.

Proof. Let R_1, R_2, R_3, R_4 and R_5 be subsets of R such that:

- $R_1 = \{u \in R\colon\ |N_B(u)| = 2\}$ (the set of degree-two vertices of R).
- $R_2 = \{u \in R\colon\ |N_B(u)| = 3$ and $|N_B(u) \cap N_B(R_1)| \geqslant 1\}$. In other words, R_2 is the set of degree-three vertices of R having at least one common neighbor with some vertex in R_1.
- $R_3 = \{u \in R\colon\ |N_B(u)| = 3$ and $|N_B(u) \cap N_B(R_1)| = 0\}$. R_3 consists of degree-three vertices that are not in R_2.
- $R_4 = \{u \in R\colon\ |N_B(u)| = 4\}$.
- $R_5 = \{u \in R\colon\ |N_B(u)| \geqslant 5\}$.

Let $B_i = \{u \in B$ such that degree$(u) = i\}$, and let $r_i = |R_i|$ and $b_i = |B_i|$. We partition the B_i sets further, as follows:

- B_1^i: The set of degree-one elements of B_1 whose neighbors are in R_i.
- B_2^{ij}: The set of degree-two elements of B_2 having one neighbor in R_i and the other in R_j.
- B_3^{ijk}: The set of degree-three elements of B_3 having one neighbor in R_i, one in R_j and the other in R_k.
- B_4^{ijkp}: The set of degree-four elements of B_4 having one neighbor in R_i, one in R_j, one in R_k and the other in R_p.[2]

Let $b_1^i = |B_1^i|$, $b_2^{ij} = |B_2^{ij}|$, $b_3^{ijk} = |B_3^{ijk}|$ and $b_4^{ijkp} = |B_4^{ijkp}|$. Then, by counting the number of edges in the graph, and given the constraints on vertex-degrees:

$$
\begin{aligned}
2r_1 &\leqslant b_1^1 + b_2^{12} + b_2^{14} + b_2^{15}\\
3r_2 &\leqslant b_1^2 + b_2^{12} + 2b_2^{22} + b_2^{23} + b_2^{24} + b_2^{25}\\
3r_3 &\leqslant b_1^3 + b_2^{23} + 2b_2^{33} + b_2^{34} + b_2^{35} + 3b_3^{333} + 2b_3^{334} + b_3^{344} + 2b_3^{335} + b_3^{355} + b_3^{345}\\
4r_4 &\leqslant b_1^4 + b_2^{14} + b_2^{24} + b_2^{34} + 2b_2^{44} + b_2^{45} + b_3^{334} + 2b_3^{344} + 3b_3^{444} + 2b_3^{445} + b_3^{455} + b_3^{345} + 4b_4^{4444} + 3b_4^{4445}\\
&\quad + 2b_4^{4455} + b_4^{4555}\\
5r_5 &\leqslant b_1^5 + b_2^{15} + b_2^{25} + b_2^{35} + b_2^{45} + 2b_2^{55} + b_3^{335} + 2b_3^{355} + b_3^{445} + 2b_3^{455} + 3b_3^{555} + b_3^{345} + 4b_4^{5555} + 3b_4^{5554}\\
&\quad + 2b_4^{5544} + b_4^{5444}
\end{aligned}
$$

Since every vertex in R_2 can have at most two common neighbors with vertices from R_1, we have: $b_2^{12} \leqslant 2r_2$ and $\frac{b_2^{12}}{4} \leqslant \frac{r_2}{2}$. Consequently, we get:

$$
\begin{aligned}
3r_2 + 0.5r_2 &\leqslant b_1^2 + b_2^{12} + 2b_2^{22} + b_2^{23} + b_2^{24} + b_2^{25} + 0.5r_2\\
3r_2 + 0.25b_2^{12} &\leqslant b_1^2 + b_2^{12} + 2b_2^{22} + b_2^{23} + b_2^{24} + b_2^{25} + 0.5r_2\\
3r_2 - 0.5r_2 &\leqslant b_1^2 + b_2^{12} - 0.25b_2^{12} + 2b_2^{22} + b_2^{23} + b_2^{24} + b_2^{25}\\
2.5r_2 &\leqslant b_1^2 + 0.75b_2^{12} + 2b_2^{22} + b_2^{23} + b_2^{24} + b_2^{25}
\end{aligned}
$$

In addition, we know that any vertex in R_3 can have at most one degree-three neighbor in B_3. Thus:

$$N_{R_3}\left(B_3^{333}\right) \cap N_{R_3}\left(B_3^{334}\right) \cap N_{R_3}\left(B_3^{344}\right) \cap N_{R_3}\left(B_3^{335}\right) \cap N_{R_3}\left(B_3^{355}\right) \cap N_{R_3}\left(B_3^{345}\right) = \emptyset$$

And:

$$3b_3^{333} + 2b_3^{334} + b_3^{344} + 2b_3^{335} + b_3^{355} + b_3^{345} \leqslant r_3$$

Replacing in the counting equation for r_3 we get:

$$
\begin{aligned}
3r_3 + 0.5r_3 &\leqslant b_1^3 + b_2^{23} + 2b_2^{33} + b_2^{34} + b_2^{35} + 3b_3^{333} + 2b_3^{334} + b_3^{344} + 2b_3^{335} + b_3^{355} + b_3^{345} + 0.5r_3\\
2.5r_3 &\leqslant b_1^3 + b_2^{23} + 2b_2^{33} + b_2^{34} + b_2^{35} + 1.5b_3^{333} + b_3^{334} + 0.5b_3^{344} + b_3^{335} + 0.5b_3^{355} + 0.5b_3^{345}
\end{aligned}
$$

[2] Note that i, j, k and p need not all be different.

Similarly, any vertex in R_4 can have at most one degree-four neighbor in B_4. Thus:

$$N_{R_4}\left(B_4^{4444}\right) \cap N_{R_4}\left(B_4^{4445}\right) \cap N_{R_4}\left(B_4^{4455}\right) \cap N_{R_4}\left(B_4^{4555}\right) = \emptyset$$

And:

$$4b_4^{4444} + 3b_4^{4445} + 2b_4^{4455} + b_4^{4555} \leqslant r_4$$

Replacing in the counting equation for r_4 we get:

$$\begin{aligned} 4r_4 + 0.25r_4 &\leqslant b_1^4 + b_2^{14} + b_2^{24} + b_2^{34} + 2b_2^{44} + b_2^{45} + b_3^{334} + 2b_3^{344} + 3b_3^{444} + 2b_3^{445} + b_3^{455} + b_3^{345} + 4b_4^{4444} \\ &\quad + 3b_4^{4445} + 2b_4^{4455} + b_4^{4555} + 0.25r_4 \\ 3.75r_4 &\leqslant b_1^4 + b_2^{14} + b_2^{24} + b_2^{34} + 2b_2^{44} + b_2^{45} + b_3^{334} + 2b_3^{344} + 3b_3^{444} + 2b_3^{445} + b_3^{455} + b_3^{345} + 3b_4^{4444} \\ &\quad + 2.25b_4^{4445} + 1.5b_4^{4455} + 0.75b_4^{4555} \end{aligned}$$

Furthermore, we know that $\sum_i |B_1^i| = |B_1|$, $\sum_{i,j} |B_2^{i,j}| = |B_2|$, $\sum_{i,j,k} |B_3^{i,j,k}| = |B_3|$, $\sum_{i,j,k,p} |B_4^{i,j,k,p}| = |B_4|$ and $B_1 \cup B_2 \cup B_3 \cup B_4 = B$.

Putting it all together, we finally obtain:

$$60|R| \leqslant 30|B_1| + 48|B_2| + 48|B_3| + 48|B_4|$$

Hence: $60|R| \leqslant 48(|B_1| + |B_2| + |B_3| + |B_4|) = 48|B|$.

This completes the proof. □

3. Reduction to weighted Steiner tree

Given a weighted undirected graph $G = (V, E)$, with a weight function $w : E \to \mathbb{N}$ and a partition $\{X, Y\}$ of V. The Weighted Steiner Tree problem asks for a minimum-weight subtree of G that connects all the vertices of Y. Elements of Y are termed terminals, while X contains the Steiner nodes.

Lemma 2. *An arbitrary instance of* CRBDS *is reducible to a* WST *instance where the red vertices are terminal nodes and the blue ones are the Steiner nodes.*

Proof. Let $G = (R \cup B, E)$ be an undirected simple red–blue graph, with R the independent set of red vertices and B the set of blue vertices. We assign weights to edges of G as follows: each edge $\{u, v\}$ such that both u and v are in B is assigned a weight $w(u, v) = 1$, and we set the weight of any other edge to n.

Let R be the set of terminals when G is considered as an instance of WST and let T be a minimum Steiner tree of G. We claim that the vertex set of T ($V(T)$) is a minimum connected red–blue dominating set. To prove this claim we observe the following:

- Since R is an independent set and R is the set of terminals, each red vertex is adjacent to a blue vertex in T. Thus $V(T) \cap B$ is a red–blue dominating set of G.
- Every red vertex is a leaf node in T. Otherwise, some red vertex u would have (at least) two blue neighbors, v and w, in T. Removing one of the two edges, say uv, and connecting v to w via a path of blue vertices (since B is connected), gives a Steiner tree of smaller weight (since the weight of any path connecting blue vertices is bounded above by $n-1$).

Since every red vertex is a leaf node, $V(T) \cap B$ is connected. And since every edge that connects internal nodes of T is of weight 1, the number of internal nodes in T is minimum. □

The current fastest known algorithm for WST has a running time in $\mathcal{O}^*(2^k)$ where k is the number of terminal nodes [2]. The algorithm proposed in [12] uses the Möbius inversion approach, an extension of the so-called Inclusion–Exclusion principle, to solve the problem in polynomial space and guarantees the same running time.

Now let $0 < \alpha < 1$ be such that $\max(2^\alpha, 2^{1-\alpha})$ is minimized. Then α is 0.5. Accordingly, we distinguish two possible cases:

(i) When $|B| < \alpha n$, then the CRBDS problem is solvable by exhaustive enumeration of all subsets of B in $\mathcal{O}^*(2^{\alpha n})$.
(ii) When $|B| \geqslant \alpha n$, then the CRBDS problem is solvable via the WST algorithm in $\mathcal{O}^*(2^{(1-\alpha)n})$.

In both cases, the run-time is in $\mathcal{O}^*(1.4143^n)$. In the following section, we show how to obtain a better worst-case run-time.

4. An improved exact algorithm

We shall assume the given instance has the following three subsets of vertices: (i) the set D of blue vertices that must belong to an optimum solution, (ii) the set B of remaining blue vertices, and (iii) the set R of red vertices that are yet "dominating vertices." Once a blue vertex is added to D, its red neighbors are deleted. Moreover, we shall contract an edge if it joins two elements of D.

Our algorithm adopts a branch and reduce strategy as long as the following reduction rules and/or favorable branching conditions are detected in a problem instance. In the absence of such conditions, the algorithm uses a Weighted Steiner Tree algorithm as a subroutine denoted by $\text{WST}(S, T)$ where $S = B$ and $T = R \cup D$. Solving the WST instance results in connecting all remaining isolated vertices in $R \cup D$ with the minimum number of nodes in B. The algorithm terminates returning the cardinality of the minimum connected red–blue dominating set. The algorithm can be modified easily to return an optimum solution.

4.1. Reduction rules

We describe reduction rules that are used both as a preprocessing stage and for pruning purposes during our algorithm.

1. If $\exists u, v \in D$ such that $\{u, v\} \in E$, then contract the edge $\{u, v\}$ and decrement the problem size by one.
   The soundness of this rule follows from the fact that any subsequent solution must connect any of the two vertices with the rest of the connected dominating set.
2. If $\exists u$ such that $u \in R$ and $|N_B(u)| = 1$, then delete u and add its unique blue neighbor to D. Decrement the problem size by one. The soundness of this rule is obvious.
3. If $\exists u$ such that $u \in D$, $N_B(u) = \{v\}$ and D is not a connected dominating set or $|D| > 1$, then add v to D, contract edge uv and delete $N_R(v)$. (Since u cannot be isolated in a solution, and it has only one neighbor in B, then its unique blue neighbor must be added to connect u to the rest of the target dominating set.)
4. If $\exists\{u, v\}$ such that $\{u, v\} \subseteq R$ and $N_B(u) \subseteq N_B(v)$ then delete v. This "domination" rule is sound because covering u implies covering v.
5. If $|D| > 1$ and $\exists\{u, v\}$ such that $u \in D$, $v \in R$ and $N_B(u) \subseteq N_B(v)$, then delete v. This rule is sound because at least one neighbor of u must be in D, and because the red vertex v is not needed to connect the elements of D in the target solution.

4.2. Branching rules

According to the above reduction rules, elements of $R \cup D$ form an independent set. To make sure the set R is dominated by a connected subset of the blue vertices, our algorithm proceeds until all red vertices are deleted. As for the set D, and due to Reduction Rule 2, we should not have more than one element in D when the algorithm terminates. Whether R is empty or not, elements of the set $R \cup D$ must be connected by adding a minimum number of blue vertices, which may lead to the use of WST.

While contracting edges between elements of D, we must keep track of the number of such elements, being the objective value that we seek to minimize. Therefore, we shall refer to this number by *minDS* in the pseudo-code of our algorithm.

When it comes to branching at a vertex $v \in B$, we consider the neighbors of v that are in $R \cup D$. When v is added to an optimum solution, it becomes an element of D and we delete all its red neighbors. Thus, by Reduction Rule 2, we contract the edges between v and its dominating neighbors, which is equivalent to deleting such "independent" neighbors and connecting v to their neighbors in B. In some sense, R and D are indistinguishable when branching at an element v of B, because the problem size is reduced by the number of both red and dominating neighbors of v. We denote by $N_{R\cup D}(v)$ the set of neighbors of v belonging to $R \cup D$.

When none of the reduction rules can be applied, the algorithm "tries" to branch according to seven different cases such that the $\mathcal{O}^*(1.3645^n)$ bound on the run-time is not exceeded. Looking at vertices in B, we notice that the regular branching strategy of including or excluding a node from an optimum solution can only be applied to vertices $v \in B$ having $|N_{R\cup D}(v)| \geqslant 5$.

After branching on all such vertices, we are left with a set of vertices in B having bounded degree four with respect to $R \cup D$. Since our goal is to minimize the cardinality of $R \cup D$ (i.e. when branching alone does not solve the instance), which is the parameter k for the WST algorithm, we turn our attention to vertices belonging to $R \cup D$ and their respective neighborhoods in B. Any vertex $u \in R \cup D$ with $|N_B(u)| \geqslant 5$ can be ignored because all of its blue neighbors have smaller degree and this structure respects our breaking point criteria.

The problematic cases arise when a vertex $u \in R \cup D$ has degree four, three or two since u could have one or more neighbors with equal or greater degree. We distinguish those three cases and provide a general branching strategy for each. For every vertex $u \in R \cup D$ having degree $d \in \{2, 3, 4\}$ and at least one neighbor of equal or greater degree, we branch when one of the following rules applies:

- $|N_B(u)| = 2$ and $|N_{R\cup D}(N_B(u))| \geqslant 4$.

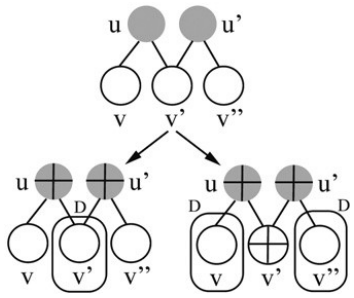

Fig. 1. Branching rule (2): Either v' is added to D and $\{u, u'\}$ is deleted or v and v'' are added to D and $\{u, u', v'\}$ is deleted.

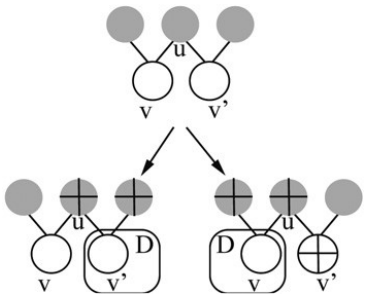

Fig. 2. Branching rule (3): Either v' is added to D and $N_{R\cup D}(v')$ is deleted or v is added to D and $N_{R\cup D}(v)$ is deleted in addition to v'.

- $|N_B(u)| = 3$ and $|N_{R\cup D}(N_B(u))| \geqslant 8$.
- $|N_B(u)| = 4$ and $|N_{R\cup D}(N_B(u))| \geqslant 12$.

The order in which the branching occurs is important for attaining the desired run-time. The algorithm ranks vertices in $N_B(u)$ according to their degree (i.e. in decreasing order) and branches accordingly. After every branch, the previously considered vertices can be discarded, which further reduces the size of the instance. Additional favorable branching rules are derived when considering degree-two vertices in $R \cup D$. Whenever we exclude one of the neighbors of such a vertex the other must automatically be added to an optimum solution to ensure that the vertex is dominated.

Following the presented rules, we succeed in obtaining a structure where every vertex in $R \cup D$ of degree d has at most one blue neighbor of degree d while every other blue neighbor has a smaller degree. A detailed description of all seven branching rules is provided below. The soundness of these rules is straightforward.

1. If $\exists v$ such that $v \in B$ and $|N_{R\cup D}(v)| \geqslant 5$, then we branch with two sub-problems:
   - $SUB_{B11} = (\{B \setminus v\}, R, D)$ deleting v from B.
   - $SUB_{B12} = (\{B \setminus v\}, \{R \setminus N_R(v)\}, \{D \cup v\})$ deleting five or more neighbors of v from $R \cup D$ and adding v to D (and, of course, joining v to the blue neighbors of its dominating neighbors).
2. If $\exists\{u, u'\}$ such that $\{u, u'\} \subseteq R \cup D$, $N_B(u) = \{v, v'\}$, $N_B(u') = \{v', v''\}$, and $|N_{R\cup D}(v)| = |N_{R\cup D}(v'')| = 1$, then we branch with two sub-problems (Fig. 1):
   - $SUB_{B21} = (\{B \setminus v'\}, \{R \setminus N_R(v')\}, \{D \cup v'\})$. In other words, v' is added to D and $N_{R\cup D}(v')$ is deleted.
   - $SUB_{B22} = (\{B \setminus \{v, v', v''\}\}, \{R \setminus N_R(v) \cup N_R(v'')\}, \{D \cup \{v, v''\}\})$. ($v'$ is deleted, which forces the two other blue vertices to be added to D.)
3. If $\exists u$ such that $u \in R \cup D$, $N_B(u) = \{v, v'\}$, and $|N_{R\cup D}(v)| + |N_{R\cup D}(v')| \geqslant 4$, then, assuming (w.l.o.g.) $|N_{R\cup D}(v)| \leqslant |N_{R\cup D}(v')| \leqslant 4$, we branch with two sub-problems (Fig. 2):
   - $SUB_{B31} = (\{B \setminus v'\}, \{R \setminus N_R(v')\}, \{D \cup v'\})$: v' is added to D and $N_{R\cup D}(v')$ is deleted.
   - $SUB_{B32} = (\{B \setminus \{v, v'\}\}, \{R \setminus N_R(v)\}, \{D \cup v\})$: v is added to D and both v and v' are deleted from B.
4. If $\exists u$ such that $u \in R \cup D$, $N_B(u) = \{v, v', v''\}$ and $|N_{R\cup D}(v)| + |N_{R\cup D}(v')| + |N_{R\cup D}(v'')| \geqslant 8$, then, assuming $|N_{R\cup D}(v')| \leqslant |N_{R\cup D}(v)| \leqslant |N_{R\cup D}(v'')| \leqslant 4$, we branch with three sub-problems (Fig. 3):
   - $SUB_{B41} = (\{B \setminus v''\}, \{R \setminus N_R(v'')\}, \{D \cup v''\})$: delete v'' from B after adding it to D and deleting $N_{R\cup D}(v'')$. Note, again, that the neighbors of v'' in D are deleted due to edge-contraction.
   - $SUB_{B42} = (\{B \setminus \{v, v''\}\}, \{R \setminus N_R(v)\}, \{D \cup v\})$: delete v and v'' from B after adding v to D and the neighbors of v are deleted ($N_{R\cup D}(v)$) from $R \cup D$.
   - $SUB_{B43} = (\{B \setminus \{v, v', v''\}\}, \{R \setminus N_R(v')\}, \{D \cup v'\})$: delete v, v' and v'' from B. v' is added to D and its neighbors in $R \cup D$ are deleted.
5. If $\exists u$ such that: $u \in R \cup D$, $N_B(u) = \{v, v', v'', v'''\}$, $|N_{R\cup D}(v''')| \leqslant |N_{R\cup D}(v'')| \leqslant |N_{R\cup D}(v')| \leqslant |N_{R\cup D}(v)| = 4$, and $|N_{R\cup D}(v)| + |N_{R\cup D}(v')| + |N_{R\cup D}(v'')| + |N_{R\cup D}(v''')| \geqslant 12$, then we branch with four sub-problems:
   - $SUB_{B51} = (\{B \setminus v\}, \{R \setminus N_R(v)\}, \{D \cup v\})$.

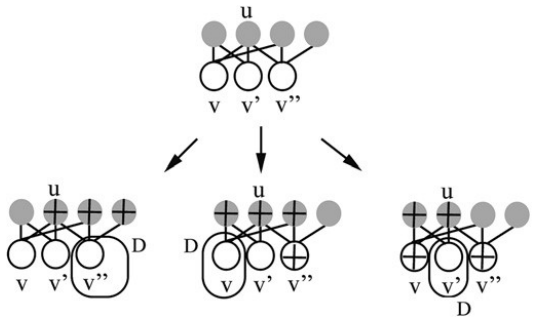

Fig. 3. Branching rule (4): Either v'' is added to D and $N_{R\cup D}(v'')$ is deleted or v is added to D and $N_{R\cup D}(v)$ and v'' are deleted or v' is added to D and $N_{R\cup D}(v')$, v and v'' are deleted.

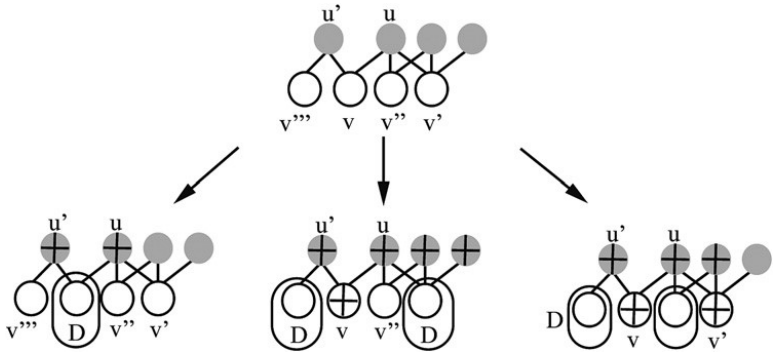

Fig. 4. Branching rule (6): Either v is added to D and $N_{R\cup D}(v)$ is deleted or $\{v', v'''\}$ is added to D and $N_{R\cup D}(\{v', v'''\})$ and v are deleted or $\{v'', v'''\}$ is added to D and $N_{R\cup D}(\{v'', v'''\})$, v and v' are deleted.

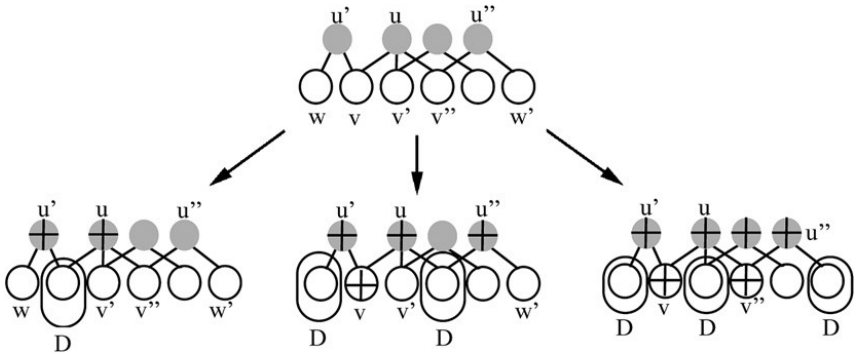

Fig. 5. Branching rule (7): Either v is added to D and $N_{R\cup D}(v)$ is deleted or $\{v'', w\}$ is added to D and $N_{R\cup D}(\{v'', w\})$ and v are deleted or $\{v', w, w'\}$ is added to D and $N_{R\cup D}(\{v', w, w'\})$, v and v' are deleted.

- $SUB_{B52} = (\{B \setminus \{v, v'\}\}, \{R \setminus N_R(v')\}, \{D \cup v'\})$.
- $SUB_{B53} = (\{B \setminus \{v, v', v''\}\}, \{R \setminus N_R(v'')\}, \{D \cup v''\})$.
- $SUB_{B54} = (\{B \setminus \{v, v', v'', v'''\}\}, \{R \setminus N_R(v''')\}, \{D \cup v'''\})$.

6. If $\exists\{u, u'\}$ such that: $\{u, u'\} \subseteq R \cup D$, $N_B(u) = \{v, v', v''\}$, $N_B(u') = \{v, v'''\}$ (u and u' have one common neighbor), $|N_{R\cup D}(v')| = 3$ and $|N_{R\cup D}(v'')| = 2$, then we branch with three sub-problems (Fig. 4):
   - $SUB_{B61} = (\{B \setminus v\}, \{R \setminus N_R(v)\}, \{D \cup v\})$.
   - $SUB_{B62} = (\{B \setminus \{v, v', v'''\}\}, \{R \setminus N_R(v') \cup N_R(v''')\}, \{D \cup \{v', v'''\}\})$.
   - $SUB_{B63} = (\{B \setminus \{v, v', v'', v'''\}\}, \{R \setminus N_R(v'') \cup N_R(v''')\}, \{D \cup \{v'', v'''\}\})$ are deleted from $R \cup D$.
7. If $\exists\{u, u', u''\}$ such that: $\{u, u', u''\} \subseteq R \cup D$, $N_B(u) = \{v, v', v''\}$, $N_B(u') = \{v, w\}$, $N_B(u'') = \{v'', w'\}$, $w \neq w'$ and $|N_{R\cup D}(v')| \geqslant 2$, then we branch with three sub-problems (Fig. 5):
   - $SUB_{B71} = (\{B \setminus v\}, \{R \setminus N_R(v)\}, \{D \cup v\})$.
   - $SUB_{B72} = (\{B \setminus \{v, v'', w\}\}, \{R \setminus N_R(v'') \cup N_R(w)\}, \{D \cup \{v'', w\}\})$.
   - $SUB_{B73} = (\{B \setminus \{v, v', v'', w, w'\}\}, \{R \setminus N_R(v') \cup N_R(w) \cup N_R(w')\}, \{D \cup \{v', w, w'\}\})$.

4.3. The algorithm

As described above, we could reach a situation where none of the previous branching rules applies and no more reductions are possible. In this case, the algorithm generates and solves a WEIGHTED STEINER TREE instance WST(S, T). We denote by *Generate_*WST*_Instance* the routine that takes a red–blue graph G and generates an instance of WST, as described in Lemma 2. Upon its termination, the call of WST(S, T) accordingly updates sets B, R and D so the termination condition (base case) of the algorithm is respected.

Algorithm 1 An algorithm for CONNECTED RED–BLUE DOMINATING SET: CRBDS

Input: A red–blue graph $G = (\{R \cup B\}, E)$ and a set D initially empty.
Output: Size of a solution (if any), denoted by *minDS*, initially 0.

```
if (|R| = 0 ∧ |D| = 1) then
    return minDS;
Apply all reduction rules until none of them can be applied;
Take v ∈ B where |N(v) ∩ {R ∪ D}| is maximum;
if |N(v) ∩ {R ∪ D}| > 4 then
    return min{crbds(SUB_B11), crbds(SUB_B12)};
if BR2 applies then
    return min{crbds(SUB_B21), crbds(SUB_B22)};
if BR3 applies then
    return min{crbds(SUB_B31), crbds(SUB_B32)};
if BR4 applies then
    return min{crbds(SUB_B41), crbds(SUB_B42), crbds(SUB_B43)};
if BR5 applies then
    return min{crbds(SUB_B51), crbds(SUB_B52), crbds(SUB_B53), crbds(SUB_B54)} ;
if BR6 applies then
    return min{crbds(SUB_B61), crbds(SUB_B62), crbds(SUB_B63)};
if BR7 applies then
    return min{crbds(SUB_B71), crbds(SUB_B72), crbds(SUB_B73)};
(S, T) ← Generate_WST_Instance(G)
return WST(S, T);
```

The correctness of the algorithm follows easily from the description and soundness of the reduction and branching rules. It is guaranteed to return an optimal solution when one does exist in G. The next section will present a run-time analysis for the branch and reduce part of the algorithm, then we derive the claimed total running time.

5. Analysis

We first note that an instance of CRBDS is solved by either branching only, branching and then solving WST(S, T) or by solving WST(S, T) only. The WST instance that results satisfies the following lemma.

Lemma 3. *Let $G = (R \cup D \cup B, E)$ be the graph that results after applying all the favorable branching rules. Then $|R \cup D| \leqslant 4|B|/5$.*

Proof. Let $G = (R \cup D \cup B, E)$ be the graph that results after applying all the favorable branching rules. G satisfies the following properties:

- No vertex in $R \cup D$ can have less than two blue neighbors in B otherwise the algorithm would have reduced the size of the instance. Therefore, $\forall u \in R \cup D$ we have $|N_B(u)| \geqslant 2$.
- By the first branching rule (BR1), every vertex of $N_B(R \cup D)$ has degree four or less.
- By BR2, we know that no two vertices of degree two in $R \cup D$ can have a common neighbor in B.
- By BR3, every degree-two vertex of $R \cup D$ has a pendant (degree-one) blue neighbor and another neighbor of degree two at most.
- By BR4, every degree-three vertex of $R \cup D$ having a degree-four neighbor in B must also have a pendant neighbor and a degree-two (at most) neighbor in B. BR4 also guarantees that any degree-three vertex in $R \cup D$ can have at most two degree-three neighbors in B and its third neighbor is pendant.
- By BR5, degree-four vertices in $R \cup D$ can have at most two degree-four neighbors in B. When a degree-four vertex in $R \cup D$ has two degree-four neighbors in B, one of the remaining neighbors is pendant and the other has degree two or less. When a degree-four vertex in $R \cup D$ has one degree-four neighbor in B, at least one of the remaining neighbors must have degree two or less.
- By BR6, we know that when a degree-three and a degree-two vertices in $R \cup D$ have a common neighbor in B and the degree-three vertex also has a degree-three neighbor in B then its third neighbor must be of degree one.
- By BR7, every degree-three vertex in $R \cup D$ can only have two of its neighbors (from B) common with degree-two vertices in $R \cup D$.

In order to apply Lemma 1, we need to make some modifications to the graph. Let $Q \subseteq R \cup D$ be the set of all vertices in $R \cup D$ identified by one of the following cases:

(a) If a degree-three vertex $u \in R \cup D$ has a degree-four neighbor in B, then $u \in Q$.
(b) If a degree-three vertex $u \in R \cup D$ has two degree-three neighbors in B, then $u \in Q$.
(c) If a degree-three vertex $u \in R \cup D$ and a degree-two vertex $v \in R \cup D$ have a common neighbor in B and u has a degree-three neighbor in B, then $u \in Q$.
(d) If a degree-four vertex $u \in R \cup D$ has two degree-four neighbors in B, then $u \in Q$.

While Q is not empty, repeat the following steps for each vertex $u \in Q$:

1. Let $v \in B$ denote the neighbor of u of highest degree.
2. Pick from $R \cup D$ a neighbor of v of highest degree. Denote it by w.
3. Connect the neighbor of u (from B) with lowest degree to w.
4. Delete the edge $\{v, w\}$.
5. If (a), (b), (c) and (d) are false, delete u from Q.

The above procedure does not change the number of edges and vertices in the graph since for every deleted edge a new edge is added and the cardinality of $R \cup D$ and B is not modified. In the four cases ((a), (b), (c) and (d)) described above, the neighbor of u with lowest degree will have degree one and the neighbor of v of highest degree must be of degree three or more (from BR3). Thus, the "modified" degree of a vertex in $v \in B$ cannot exceed two and no vertex in $N_R(v)$ can be of degree less than three. After applying the described procedure, every degree-three vertex in R can have no degree-four and at most one degree-three neighbor in B. Similarly, every degree-four vertex in R can have at most one degree-four neighbor in B. A degree-three vertex in R that has a common neighbor (in B) with a degree-two vertex from R can have no degree-three neighbor in B. Thus $R \cup D$ and $N_B(R \cup D)$ induce a bipartite graph satisfying Lemma 1 and $|R \cup D| \leqslant 4|B|/5$. The proof is now complete. □

We choose the total number of vertices as a measure for the size of a CRBDS instance (R, D, B): $n = |R| + |D| + |B|$. Let $T(n)$ be the number of sub-problems generated to solve an instance of size n. When we apply any of the reduction rules the size of the problem decreases by at least 1.

Consider the case when the algorithm branches. Following the description of algorithm CRBDS we distinguish seven different sub-cases whose quantitative effect on the run-time can be described as follows: when branching according to BR1 the size of the problem is reduced by either one or at least five ($T(n) \leqslant T(n-1) + T(n-5)$). Similarly for BR2 and BR3, the problem size is reduced by two or three, and so on.

The reader can easily verify that, based on the reduction procedures and all seven branching rules, the worst case running time of CRBDS when only branching is used satisfies:

$$T(n) \leqslant \max \begin{cases} T(n-1) + T(n-5) \\ T(n-2) + T(n-3) \\ T(n-3) + T(n-4) + T(n-4) \\ T(n-4) + T(n-4) + T(n-5) + T(n-5) \\ T(n-2) + T(n-5) + T(n-5) \\ T(n-2) + T(n-4) + T(n-6) \end{cases} \tag{1}$$

We conclude that $T(n) \leqslant \alpha^n$ where $\alpha = 1.36443\ldots \leqslant 1.3645$ is the greatest positive root of the six polynomials induced from the previous equation. Thus the worst case run-time is $\mathcal{O}^*(1.3645^n)$ whenever a solution is found without generating a WEIGHTED STEINER TREE instance.

The next step would be to consider the run-time when either partial branching occurs and is followed by WST(S, T) or when no branching or reductions occur and WST(S, T) returns a solution. In the latter case, D is empty. So in both cases $R \cup D$ and $N_B(R \cup D)$ induce a bipartite graph that satisfies Lemma 3. It follows that $|N_B(R \cup D)| \geqslant \frac{5}{4}|R \cup D|$. Hence, whenever WST$(S, T)$ is called, $|T| \leqslant \frac{4}{5}|S|$ (at least) and $|T| \leqslant \frac{4}{9}n$ since $|S| \leqslant n - |T|$. By the result obtained in [12], WST(S, T) runs in $\mathcal{O}^*(2^{|T|})$, which is $\mathcal{O}^*(2^{\frac{4}{9}n}) \equiv \mathcal{O}^*(1.3608^n)$.

Finally, note that in the branching part of the algorithm the reduction in problem size is due only to vertex deletions. It follows that, when both branching and WST are applied, the run-time is in $\mathcal{O}^*(1.3645^{n_1} 1.3608^{n_2})$, where n_1 is the number of vertices deleted during the branch and reduce phase and $n_2 = n - n_1$. We have now proved our first result:

Theorem 4. CRBDS *can be solved exactly by an algorithm whose running time is in* $\mathcal{O}^*(1.3645^n)$ *and uses polynomial space.*

6. An improved algorithm for connected dominating set

Next, we show that CDS can be solved using our described algorithm for CRBDS at the cost of doubling the input size. The current best-known algorithm was presented recently in [7] where the authors solve CDS by reduction to the MAXIMUM-LEAF SPANNING TREE problem, in $\mathcal{O}^*(1.8966^n)$. The following reduction to CRBDS yields a better worst-case run-time.

An instance $G = (V, E)$ of CDS is transformed into a CRBDS instance $G' = (R \cup B, E')$ as follows:

- R and B are sets of $|V|$ vertices that are in one-to-one correspondence, each, with V. In other words, there exist $r : R \to V$ and $b : B \to V$ such that both r and b are bijective.
- R is an independent set.
- For each pair $\{u, v\} \in B$, if $\{b(u), b(v)\}$ is an edge in G, then add $\{u, v\}$ to E'.
- For each pair $\{v, w\} \in R \times B$, if $\{r(v), b(w)\} \in E$, then add the edge $\{v, w\}$ to E'.
- For each pair $\{v, w\} \in R \times B$, if $r(v) = b(w)$ then add the edge $\{v, w\}$ to E'.

In addition, let r' and b' denote the inverses of r and b, respectively. In other words, $r'(u)$ and $b'(u)$ are the red and blue images (in G') of a vertex $u \in V$.

Claim. *D is a minimum connected red–blue dominating set of G' iff $b(D)$ is a minimum connected dominating set of G.*

Proof. Let D be a connected red–blue dominating set of G'. Clearly $b(D)$ is a connected dominating set of G since $G'[D] = G[b(D)]$ is connected and $r(R) = V$. Moreover, $|b(D)| = |D|$. Conversely, let D' be a connected dominating set of G. If we replace D by $b'(D')$ in the modified graph G', then $G'[b'(D')]$ is connected (by construction) and dominates R since D' dominates V and $r'(V) = R$. Moreover, $|b'(D')| = |D'|$. □

Theorem 5. CONNECTED DOMINATING SET *can be solved exactly by an algorithm whose running time is in $\mathcal{O}^*(1.8619^n)$ and uses polynomial space.*

Proof. Any CDS instance $G = (V, E)$ can be reduced to a CRBDS instance $G' = (R \cup B, E')$ in polynomial time using the above procedure. Since $|R \cup B| = 2|V|$, the running time of our CRBDS algorithm is bounded above by $\mathcal{O}^*(1.3645^{2n})$. □

7. Conclusion

We showed that the CONNECTED RED–BLUE DOMINATING SET problem is solvable exactly in $\mathcal{O}^*(1.3645^n)$ time and polynomial space. Our algorithm starts by applying branch and reduce strategies. Then, when the branching conditions cannot guarantee a desired running time, we transform the instance of CRBDS into an equivalent instance of the WEIGHTED STEINER TREE problem. In addition, we proved that CDS can be solved in $\mathcal{O}^*(1.8619^n)$ time and polynomial space by a reduction to CRBDS, improving the previous result of [7].

Additional work is needed to determine whether the reduction to WST can be replaced by direct work on the CRBDS instance, and to make use of the fact that the terminal nodes (or red vertices) form an independent set in the WST instance. Finally, we note that the use of measure and conquer [8] in the branching phase might not lead to a better run-time since there may be worst-case instances for which favorable branching conditions do not hold.

References

[1] J. Alber, H. Bodlaender, H. Fernau, R. Niedermeier, Fixed parameter algorithms for planar dominating set and related problems, in: Proceedings of the 7th Scandinavian Workshop on Algorithm Theory SWAT, Springer, 2000, pp. 97–110.
[2] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto, Fourier meets Möbius: Fast subset convolution, in: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, 2007, pp. 67–74.
[3] J. Blum, M. Ding, A. Thaeler, X. Cheng, Connected dominating set in sensor networks and MANETs, in: Handbook of Combinatorial Optimization, vol. B (Suppl.), Springer, New York, 2005, pp. 329–369.
[4] F. Dorn, E. Penninkx, H. Bodlaender, F. Fomin, Efficient exact algorithms on planar graphs: Exploiting sphere cut branch decompositions, in: Proceedings of the 13th Annual European Symposium on Algorithms ESA, Springer, 2005, pp. 95–106.
[5] R. Downey, M. Fellows, U. Stege, Parameterized complexity: A framework for systematically confronting computational intractability, in: DIMACS Ser. Discrete Math. Theoret. Comput. Sci., 1997, pp. 49–99.
[6] R. Downey, M. Fellows, A. Vardy, G. Whittle, The parametrized complexity of some fundamental problems in coding theory, SIAM J. Comput. 29 (2) (1999) 545–570.
[7] H. Fernau, J. Kneis, D. Kratsch, A. Langer, M. Liedloff, D. Raible, P. Rossmanith, An exact algorithm for the maximum leaf spanning tree problem, in: Proceedings of the 4th International Workshop on Parameterized and Exact Computation IWPEC, 2009.
[8] F. Fomin, F. Grandoni, D. Kratsch, Measure and conquer: Domination – A case study, in: Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, vol. 3580, 2004, pp. 191–203.
[9] F. Fomin, F. Grandoni, D. Kratsch, Solving connected dominating set faster than 2^n, Algorithmica 52 (2) (2008) 153–166.
[10] M. Garey, D. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., New York, USA, 1990.
[11] I. Navid, Data reduction for connected dominating set, Master thesis, Simon Fraser University, 2005.

[12] J. Nederlof, Fast polynomial-space algorithms using Möbius inversion: Improving on Steiner tree and related problems, in: Proceedings of the 36th International Colloquium on Automata, Languages and Programming ICALP, Springer-Verlag, Berlin, 2009, pp. 713–725.
[13] J. van Rooij, H. Bodlaender, Design by measure and conquer, a faster exact algorithm for dominating set, in: Proceedings of the 25th International Symposium on Theoretical Aspects of Computer Science STACS, in: Leibniz Internat. Proc. Inform., vol. 1, Dagstuhl, Germany, 2008, pp. 657–668.
[14] J. van Rooij, J. Nederlof, T. van Dijk, Inclusion/exclusion meets measure and conquer: Exact algorithms for counting dominating sets, in: Proceedings of the Seventeenth European Symposium on Algorithms ESA, Springer, Berlin, 2009, pp. 554–565.