Algorithmen & mathematische Modellierung

Forschungsschwerpunkt

Algorithmen und mathematische Modellierung

TU Graz
Graz University of Technology

A Parity Domination Problem in Graphs with Bounded Treewidth and Distance-hereditary Graphs

Elisabeth Gassner and Johannes Hatzl

Project Area(s):
Effizient lösbare kombinatorische Optimierungsprobleme

Institut für Optimierung und Diskrete Mathematik (Math B)

Report 2006-17, September 2006

A Parity Domination Problem in Graphs with Bounded Treewidth and Distance-hereditary Graphs

Elisabeth Gassner Johannes Hatzl *

Abstract

This paper concerns a domination problem in graphs with parity constraints. The task is to find a subset of the vertices with minimum cost such that for every vertex the number of chosen vertices in its neighbourhood has a prespecified parity. This problem is known to be $\mathcal{NP}$-hard for general graphs. A linear time algorithm was developed for series-parallel graphs and trees with unit cost and restricted to closed neighbourhoods. We present a linear time algorithm for the parity domination problem with open and closed neighbourhoods and arbitrary cost functions on graphs with bounded treewidth and distance-hereditary graphs.

Keywords: Neighbourhood domination, parity constraints, bounded treewidth, distance-hereditary graphs.

1 Introduction

The dominating set problem is an extensively studied problem in graph theory and combinatorial optimization and can be proved to be $\mathcal{NP}$-complete for general graphs by a reduction from the vertex cover problem. The concept of domination is used in many applications like location problems and network design. Numerous variants of this classical problem have been investigated. For a survey on this topic the reader is referred to [17]. In this paper, we discuss a modification of the dominating set problem by considering parity constraints for the vertices. In order to state this problem in a rigorous way some notation is introduced.

Throughout the paper it is assumed that all graphs are finite, simple, undirected and connected. The vertex set of a graph G is referred to as $V(G)$, its edge set is denoted by $E(G)$. If no ambiguity is possible, we will write V and E instead of $V(G)$ and $E(G)$. For a given graph $G = (V, E)$ and a subset of the vertices $U \subseteq V$ we denote by $G[U]$ the induced subgraph of U, i.e., the vertex set of $G[U]$ is U and the edges are the edges of G where

*{gassner,hatzl}@opt.math.tu-graz.ac.at. Department of Optimization and Discrete Mathematics, Graz University of Technology, Steyrergasse 30, A-8010 Graz, Austria.

both vertices are in U. Furthermore, the open neighbourhood of a vertex v in a graph G is defined by

$$N_G(v) := \{w \in V(G) \mid \exists e = (v, w) \in E(G)\}$$

and the closed neighbourhood is

$$N_G[v] := N_G(v) \cup \{v\}.$$

Again, we will write $N(v)$ or $N[v]$ if no confusion is possible. Using these definitions a dominating set in a graph $G = (V, E)$ is a subset $D \subseteq V$ of the vertices such that $N_G[v] \cap D \neq \emptyset$ for all $v \in V$. An open dominating set can be defined in an analogue way by requiring that $N_G(v) \cap D \neq \emptyset$ for all $v \in V$. This variant of dominating set was also considered intensively (see for example [17]).

This paper considers both kind of neighbourhoods and additionally takes parity constraints into account. Let $\pi : V \to \{0, 1\}$ be a given function and $V = V^o \uplus V^c$ a partition of the vertex set, then a set $D \subseteq V$ is called a π-parity dominating set if

$$|N_G(v) \cap D| \equiv \pi(v) \mod 2 \quad \forall v \in V^o \tag{1}$$

and

$$|N_G[v] \cap D| \equiv \pi(v) \mod 2 \quad \forall v \in V^c. \tag{2}$$

If $V = V^o$ ($V = V^c$) the set D is called a π-parity open (closed) dominating set. The constraints (1) and (2) are referred to as parity constraints. The parity domination problem discussed in this paper can now be stated as follows:

Given a graph $G = (V, E)$, a function $\pi : V \to \{0, 1\}$, a partition of the vertex set $V = V^o \uplus V^c$ and a cost function $c : V \to \mathbb{R}$, find a π-parity dominating set D such that

$$c(D) := \sum_{v \in D} c(v)$$

is minimized. Note that in many instances the problem does not even have a feasible solution. In particular, if G is the complete graph with n vertices and $D \subseteq V(G)$ then $|N_G[v] \cap D|$ is either even or odd for all vertices. There is one famous result due to Sutner [24], which states that if $\pi(v) = 1$ for all $v \in V$ and $V = V^c$ then every graph G has a π-parity dominating set. He could prove this by showing that the system of linear equations $(A + I)x = \mathbf{1}$ always has a solution over the binary field $GF(2)$, where A is the adjacency matrix of a the graph G with n vertices, I is the $n \times n$ identity matrix and $\mathbf{1}$ is the vector of length n with all elements equal to 1. It is easy to see that each solution of the system of linear equations corresponds to a feasible solution to the parity domination problem. A simpler proof was given by Caro [8]. Caro and Jacobson [9] could even show that every tree contains a subset of the vertices D such that $|N_G[v] \cap D| \neq 0 \mod k$ holds for all k. They also conjectured that this result holds for general graphs, but this problem is still open.

However, Sutner [24] also observed that finding the minimum cardinality of an π-parity closed dominating set is $\mathcal{NP}$-hard. Thus, special graph classes for which this problem can be solved in polynomial time were considered. The first steps in this direction are due to Dawes [13]. He proposed a polynomial time algorithm for the special case where G is a tree, $V = V^c$ and $\pi(v) = 1$ for all $v \in V$. This result was generalized by Amin and Slater [1] who considered the class of series parallel graphs and allowed arbitrary functions π. They proposed a linear time algorithm for the case $V = V^c$ which is based on a dynamic programming approach. Our contribution to this topic is a further generalization of the model used by Amin and Slater applied to graphs with bounded treewidth (which include trees and series parallel graphs) and to distance-hereditary graphs.

This paper is organized as follows: In Section 2 we introduce the class of graphs with bounded treewidth and give an overview of some basic properties of tree decompositions. Afterwards, in Section 3 a linear time algorithm for the parity domination problem on graphs with bounded treewidth is discussed. Section 4 contains an introduction to distance-hereditary graphs while in Section 5 we propose a linear time algorithm for the considered domination problem on distance-hereditary graphs.

2 Tree decomposition and treewidth

In this section, we introduce the concept of treewidth and the underlying tree decomposition. These terms were originally introduced by Robertson and Seymour [23] and play an important role in algorithmic graph theory. Many well-known $\mathcal{NP}$-complete problems like independent set, Hamiltonian circuit, vertex cover and Steiner trees are solvable in polynomial time for graphs with bounded treewidth (see [3, 22]). Extensive surveys on treewidth can be found in Bodlaender [3] and Kloks [21]. Bodlaender [4] published a linear time algorithm which determines for a given graph $G = (V, E)$ whether the treewidth of G is at most k, for a fixed k, and if so, outputs a tree-decomposition of G with treewidth at most k. Furthermore, there exist polynomial time algorithms to compute the treewidth for cographs and distance-hereditary graphs. Distance-hereditary graphs are treated in Section 4 of this paper. In contrast, the computation of the treewidth is $\mathcal{NP}$-hard for general graphs (see [6] for more details). Examples for graphs that have bounded treewidth are trees (treewidth 1), series-parallel graphs (treewidth 2), outerplanar graphs (treewidth 2), cactus graphs (treewidth 2) and Halin graphs (treewidth 3).

Definition 2.1. A tree decomposition of a graph $G = (V(G), E(G))$ is a pair $(T, \mathcal{X})$ consisting of a tree $T = (V(T), E(T))$ and a family of vertex sets, i.e., $\mathcal{X} = \{X_t : X_t \subseteq V(G), t \in V(T)\}$ such that

(T1) $V(G) = \bigcup_{t \in V(t)} X_t$;

(T2) for every edge $(u, v) \in E(G)$ there exists a vertex $t \in V(T)$ such that $u \in X_t$ and $v \in X_t$;

(T3) if $t_2 \in V(T)$ is on the path between t_1 and t_3 in T then $X_{t_1} \cap X_{t_3} \subseteq X_{t_2}$.

Note that (T1) and (T2) together imply that the given graph G is the union of the subgraphs $G[X_t]$. Property (T3) is equivalent to the requirement that for each vertex $v \in V$ the sets X_t containing v induce a subtree of T. An example of a tree decomposition is given in Figure 1(a) and 1(b).

Observe that the cardinalities of the sets X_t depend on the given graph G. It is easy to see that if G contains a clique of size k there is a set X_t in the tree decomposition of G with $|X_t| \geq k$, because there has to exist a set containing all vertices of the clique. On the other hand, if G is a tree, there exists a tree decomposition in such a way that all sets in $\mathcal{X}$ have cardinality two. These observations lead to the following definition of treewidth, which in some sense measures how tree-like a given graph is.

Definition 2.2. The width of a tree decomposition $(T, \mathcal{X})$ of a graph $G = (V(G), E(G))$ is the number

$$\max\{|X_t| - 1 : t \in V(T)\},$$

and the treewidth $\mathrm{tw}(G)$ of G is the least width of any tree decomposition of G.

Using the comments above, it follows that the treewidth of any tree is 1 and that $\mathrm{tw}(K_n) = n - 1$, where K_n is the complete graph with n vertices.
Another useful property of tree decomposition which will be essential for our algorithm is the so called separator property. It follows directly from the definition and a formal proof can for example be found in Diestel [14].

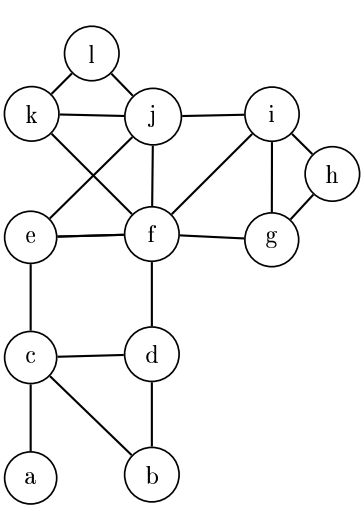
Lemma 2.3. *Let $e = (t_1, t_2)$ be an edge in T. Then $X_{t_1} \cap X_{t_2}$ is a separator of G, i.e., the graph $G[V \setminus (X_{t_1} \cap X_{t_2})]$ is not connected.*

For the description of the algorithms proposed in this paper it is helpful to use a tree decomposition which has a particular simple structure. In the following a *nice tree decomposition* is a tree decomposition $(T, \mathcal{X})$ with the following properties:

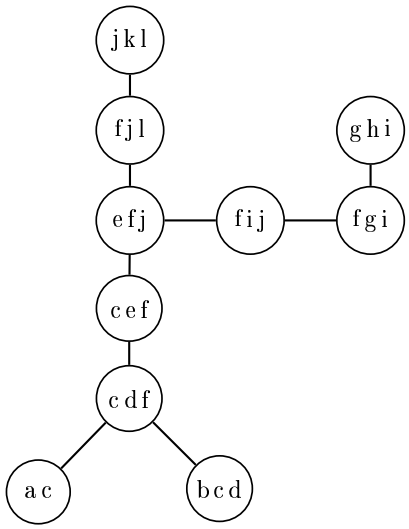
- T is a rooted binary tree with root vertex r.
- The vertices of T belong to one of the following four types:
  1. *Leaf vertex* i is a leaf of T and $|X_i| = 1$
  2. *Introduce vertex* i has one child j and $X_i = X_j \cup \{v\}$ for some vertex $v \in V(G)$
  3. *Forget vertex* i has one child j and $X_i = X_j \setminus \{v\}$ for some vertex $v \in V(G)$
  4. *Join vertex* i has two children j and l with $X_i = X_j = X_l$

See Figure 1(c) for a nice tree decomposition. Using Lemma 2.3 one can conclude that given a nice tree decomposition, for every non-leaf vertex $t \in V(T)$ the set X_t is a separator of the graph G. However, it should be pointed out that the nice tree decomposition is

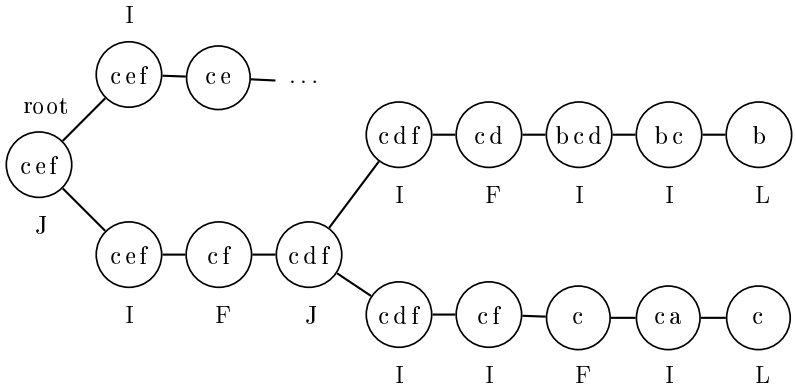
not necessary for the proposed algorithm, but enables us to simplify notations and the description of the algorithm. In [21] it is shown that if a graph G has treewidth k then there exists a tree decomposition $(T, \mathcal{X})$ with $|V(T)| = \mathcal{O}(n)$. Moreover, given such a tree decomposition $(T, \mathcal{X})$ a nice tree decomposition of G that has also $\mathcal{O}(n)$ vertices and the same width k can be found in $\mathcal{O}(n)$ time. This result ensures that it suffices to develop a linear time algorithm for a given nice tree decomposition.



(a) A graph G with treewidth 2.



(b) A treedecomposition of G with width 2.



(c) A part of a nice tree decomposition of G with the different vertex types.

Figure 1: An example for a graph, its tree decomposition and a nice tree decomposition.

3 A linear time algorithm for graphs with bounded treewidth

A lot of optimization problems on graphs with bounded treewidth can be solved with dynamic programming. It turns out that the parity domination problem can also be solved by this technique. Before we discuss the linear time algorithm more notation is needed. Suppose we are given a graph G and a nice tree decomposition $(T, \mathcal{X})$ with treewidth k which is rooted at r. Let us denote by $T(t)$ the subtree of T rooted at vertex t. Furthermore, the subgraph

$$G_t = (V_t, E_t) := G\left[\bigcup_{j \in T(t)} X_j\right]$$

is assigned to $t \in V(T)$. It is easy to see that $G_r = G$ holds. In the dynamic programming approach we will solve the parity domination problem for each graph G_t in an efficient way. At the end a solution for G_r corresponds to a solution of the original problem.

Note that the separator property of a tree decomposition implies that given a subset $W \subseteq V_t$ for some $t \in V(T)$ which should be extended to a feasible solution in G then all vertices $v \in V_t \setminus X_t$ have to satisfy the parity constraints with respect to W. However, a vertex $v \in X_t$ for some $t \in V(T)$ may violate its parity constraint with respect to W because the extension of W may include vertices of $V \setminus V_t$ that are adjacent to v. Thus, the definition of the following function seems to be appropriate for a dynamic programming approach.

For $t \in V(T)$, $Z \subseteq X_t$ and a function $s : X_t \to \{0, 1\}$ define $\mathrm{opt}_t(Z, s)$ to be minimum cost of a vertex set W in G_t with $W \cap X_t = Z$ and

$$|N_{G_t}[v] \cap W| \equiv \begin{cases} s(v) \mod 2 & \forall v \in X_t \cap V^c \\ \pi(v) \mod 2 & \forall v \in (V_t \setminus X_t) \cap V^c, \end{cases} \tag{3}$$

$$|N_{G_t}(v) \cap W| \equiv \begin{cases} s(v) \mod 2 & \forall v \in X_t \cap V^o \\ \pi(v) \mod 2 & \forall v \in (V_t \setminus X_t) \cap V^o. \end{cases} \tag{4}$$

Formally, we set $\mathrm{opt}_t(Z, s) = \infty$ if no such set W exists.
The idea of the algorithm is to compute for each vertex $t \in T$ the value $\mathrm{opt}_t(Z, s)$ for all subsets $Z \subseteq X_t$ and all functions $s : X_t \to \{0, 1\}$. In fact, we compute for each vertex t of the tree exactly $2^{|X_t|}2^{|X_t|}$ numbers. This is done in a bottom-up manner from the leaves to the root, i.e., the table opt_t is computed if all the tables of the children of t are known. The entries in the root table opt_r for which $s(v) = \pi(v)$ for all $v \in X_r$ represents an optimal solution to the problem in the original graph G.

In the following, it is assumed that we are given a nice tree decomposition of a graph G. Based on the vertex type it is explained how the tables can be obtained. In order to

describe the computation as clearly as possible some more notation is introduced. Suppose that $X_i = X_j \cup \{v\}$ for some sets $X_i, X_j \in \mathcal{X}$ and some vertex $v \in V(G) \setminus X_j$. If $s : X_i \to \{0,1\}$, then the function $s|_{X_j} \to \{0,1\}$, i.e., the restriction of s to the domain X_j, is denoted by $\tilde{s}$. Finally, if $s : X_j \to \{0,1\}$, then we define for a subset $Y \subseteq X_j$ the function $\bar{s}_Y : X_j \to \{0,1\}$ where $\bar{s}_Y(x) = 1$ if and only if $s(x) = 0$ and $x \in Y$ or $s(x) = 1$ and $x \notin Y$.

Leaf vertices: Let t be a leaf of the tree and assume that $X_t = \{u\}$. Then we immediately obtain the following table for opt_t:

	$u \in V^c$		$u \in V^o$	
	$s(u) = 0$	$s(u) = 1$	$s(u) = 0$	$s(u) = 1$
$\emptyset$	0	∞	0	∞
X_t	∞	$c(u)$	$c(u)$	∞

Introduce vertices: Let i be an introduce vertex with child j and assume $X_i = X_j \cup \{v\}$ for some vertex $v \in V(G)$. Due to the separator property there are no edges from v to a vertex $u \in V_i \setminus X_i$. In order to compute the table opt_i, several cases have to be distinguished:

1. $v \notin Z$
   In this case the parity of the vertices in X_i does not change and we only have to look at the parity of the new vertex v. Thus, we get

$$\mathrm{opt}_i(Z, s) = \begin{cases} \mathrm{opt}_j(Z, \tilde{s}) & \text{if } |N_{G[X_i]} \cap Z| \equiv s(v) \mod 2 \\ \infty & \text{otherwise.} \end{cases}$$

2. $v \in Z \cap V^o$
   If v is added to the set Z, the current parity of the vertices in $N(v) \cap X_i$ changes. Furthermore, the parity of v has to be checked. These requirements can be expressed by

$$\mathrm{opt}_i(Z, s) = \begin{cases} \mathrm{opt}_j(Z \setminus v, \bar{s}_{N(v) \cap X_i}) + c(v) & \text{if } |N_{G[X_i]} \cap Z| \equiv s(v) \mod 2 \\ \infty & \text{otherwise.} \end{cases}$$

3. $v \in Z \cap V^c$
   This case is very similar to the second one. We only have to consider that $v \in V^c$. Thus,

$$\mathrm{opt}_i(Z, s) = \begin{cases} \mathrm{opt}_j(Z \setminus v, \bar{s}_{N(v) \cap X_i}) + c(v) & \text{if } |N_{G[X_i]} \cap Z| \equiv s(v) \mod 2 \\ \infty & \text{otherwise} \end{cases}$$

   follows.

Forget vertices: We assume that the forget vertex i has one child j and $X_i = X_j \setminus \{v\}$ for some vertex $v \in V(G)$. The definition of a tree decomposition implies that $v \notin V(G) \setminus V_j$. Consequently, we have to claim that the parity of v equals $\pi(v)$ in order to guarantee feasibility. This can be obtained by

$$\mathrm{opt}_i(Z, \tilde{s}) = \mathrm{opt}_j(Z, s) \text{ with } s(v) = \pi(v).$$

Join vertices: This case is the most difficult one. Suppose that i is a join vertex that has two children j and l with $X_i = X_j = X_l$.
Let $s : X_i \to \{0, 1\}$ and $Z \subseteq X_i$ be given. In order to compute $\mathrm{opt}_i(Z, s)$ we are looking at all sets W with $W \cap X_i = Z$ that fulfill (3) and (4). The cheapest cost of these sets will be $\mathrm{opt}_i(Z, s)$. Due to the fact that $V_j \cap V_l = X_i$ it follows using the inclusion-exclusion principle

$$|N_{G_i}(v) \cap W| = |N_{G_j}(v) \cap W| + |N_{G_l}(v) \cap W| - |N_{G[X_i]}(v) \cap W| \tag{5}$$

for all $v \in X_i \cap V^o$ and $W \subseteq V_i$. Similarly, we have

$$|N_{G_i}[v] \cap W| = |N_{G_j}[v] \cap W| + |N_{G_l}[v] \cap W| - |N_{G[X_i]}[v] \cap W| \tag{6}$$

for all $v \in X_i \cap V^c$ and $W \subseteq V_i$. Let us define

$$p_Z^s(v) := \begin{cases} 1 & \text{if } v \in V^o \text{ and } s(v) + |N_{G[X_i]}(v) \cap Z| \equiv 1 \mod 2 \\ 1 & \text{if } v \in V^c \text{ and } s(v) + |N_{G[X_i]}[v] \cap Z| \equiv 1 \mod 2 \\ 0 & \text{if } v \in V^o \text{ and } s(v) + |N_{G[X_i]}(v) \cap Z| \equiv 0 \mod 2 \\ 0 & \text{if } v \in V^c \text{ and } s(v) + |N_{G[X_i]}[v] \cap Z| \equiv 0 \mod 2 \end{cases}$$

for all $v \in X_i$. Observe that $p_Z^s(v)$ indicates the parity of neighbours not in X_i that have to be chosen in a feasible solution W with $W \cap X_i = Z$ in order to achieve $s(v)$. Moreover, $N_{G[X_i]}(v) \cap Z = N_{G[X_i]}(v) \cap W$ and $N_{G[X_i]}[v] \cap Z = N_{G[X_i]}[v] \cap W$ holds for all $W \subseteq V_i$ because $W \cap X_i = Z$. Using equations (5) and (6), (3) and (4) can be rewritten as

$$|N_{G_j}[v] \cap W| + |N_{G_l}[v] \cap W| \equiv \begin{cases} p_Z^s(v) \mod 2 & \forall v \in X_i \cap V^c \\ \pi(v) \mod 2 & \forall v \in (V_j \setminus X_i) \cap V^c \\ \pi(v) \mod 2 & \forall v \in (V_l \setminus X_i) \cap V^c, \end{cases}$$

and

$$|N_{G_j}(v) \cap W| + |N_{G_l}(v) \cap W| \equiv \begin{cases} p_Z^s(v) \mod 2 & \forall v \in X_i \cap V^o \\ \pi(x) \mod 2 & \forall v \in (V_j \setminus X_i) \cap V^o \\ \pi(x) \mod 2 & \forall v \in (V_l \setminus X_i) \cap V^o. \end{cases}$$

However, this means that there exist functions $s' : V_j \to \{0, 1\}$ and $s'' : V_l \to \{0, 1\}$ with $s'(v) = |N_{G_j}[v] \cap W|$ ($|N_{G_j}(v) \cap W|$) and $s''(v) = |N_{G_l}[v] \cap W|$ ($|N_{G_l}(v) \cap W|$) with $s'(v) + s''(v) \equiv p_Z^s(v) \mod 2$ for all $v \in X_i$ and herewith

$$\mathrm{opt}_i(Z,s) = \min_{s'(v)+s''(v)\equiv p_Z^s(v)} \big(\mathrm{opt}_j(Z,s') + \mathrm{opt}_l(Z,s'')\big) - \sum_{v\in Z} c(v).$$

Observe that a vertex $v \in Z$ is contained in $X_j \cap Z$ and in $X_l \cap Z$. Therefore the cost of $v \in Z$ has to be subtracted. In the above formula for join vertices we have to compare at most 2^k values for a given s and Z. Each entry in opt_i where i is a leaf, introduce and forget vertex can be determined in constant time. Hence, the previous discussion immediately implies the following main theorem:

Theorem 3.1. *The parity domination problem on graphs with treewidth k can be solved in $\mathcal{O}(2^{3k}n)$ time which is linear for graphs with bounded treewidth.*

4 Distance-hereditary graphs

A connected graph $G = (V, E)$ is called distance-hereditary if every pair of vertices $i, j \in V$ has the same shortest distance (measured by the number of edges) in every connected subgraph containing i and j. Distance-hereditary graphs were introduced by Howorka [18]. Since then several characterizations by means of metric properties and forbidden subgraphs have been given (e. g., see [2, 16]). From the algorithmic point of view the so-called one-vertex extension property is one of the most important characterizations of distance-hereditary graphs.

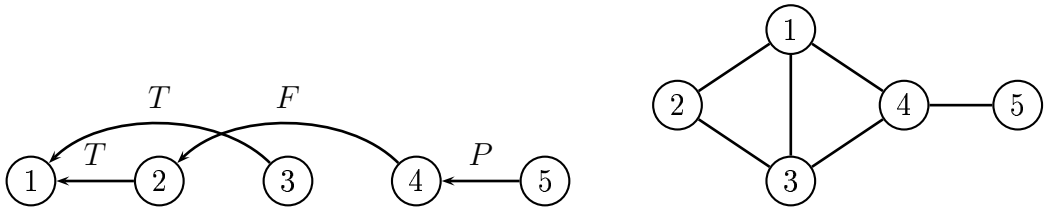
Lemma 4.1 (Bandelt and Mulder [2]). *A connected graph $G = (V, E)$ is distance-hereditary if and only if it can be obtained starting from a single vertex by a sequence of extensions of one of the following forms:*

- *True twin: Choose a vertex $v \in V$ and add a new vertex u such that $N[v] = N[u]$.*
- *False twin: Choose a vertex $v \in V$ and add a new vertex u such that $N(u) = N(v)$ (edge (u, v) is not added).*
- *Pendant: Choose a vertex $v \in V$ and add a new vertex u with $N(u) = \{v\}$, i. e., u is only adjacent to v.*

An ordering $(v_1, \ldots, v_n)$ of the vertices is called pruning sequence if v_i is a pendant or a twin of some vertex v_j in $G[\{v_1, \ldots, v_i\}]$ for $i = 2, \ldots, n$. See Figure 2 for a distance-hereditary graph and its pruning sequence.

Let $(v_1, \ldots, v_n)$ be a pruning sequence of $G = (V, E)$ such that v_n and some vertex v_j are twins. Then the graph $G' = (V', E')$ represented by the pruning sequence $(v'_1, \ldots, v'_n) = (v_1, \ldots, v_{j-1}, v_n, v_{j+1}, \ldots, v_{n-1}, v_j)$ is isomorph to G with the bijection $\sigma : V \to V'$ where $\sigma(v_i) = v'_i$ for all $v_i \in V$.

The recognition of a distance-hereditary graph and its pruning sequence can be done in $\mathcal{O}(m)$ time. Originally Hammer and Maffray [16] published a recognition algorithm which



(a) Pruning sequence of G (b) Graph G

Figure 2: A distance-hereditary graph G with its pruning sequence.

fails in some cases. However, Damiand et al. [15] recently proposed a new linear time recognition algorithm.

Many efficient algorithms for problems on distance-hereditary graphs are based on the one-vertex extension (e. g., Hamiltonian cycle [19], [20], minimum fill-in and treewidth [7], Steiner trees [12], [5], maximum stable set [11], center and median [27] and especially several variants of dominating set problems [5, 10, 12, 25, 26]).

Restrictions on the operations within the one-vertex extension procedure lead to the following subclasses of the class of distance-hereditary graphs: Trees are produced by a sequence of pendant-operations, false and true twins lead to cographs, true twins and pendants are the ingredients of Ptolemaic graphs and bipartite distance-hereditary graphs consist of false twins and pendants.

5 A linear time algorithm for distance-hereditary graphs

In this section, we develop an algorithm for the parity domination problem on distance-hereditary graphs which runs in $\mathcal{O}(m)$ time where m is the number of edges of the graph. The main idea of the proposed algorithm is to reduce the size of the graph successively by deleting some vertices while different constraints have to be taken into account in the remaining graph in order to guarantee feasibility for the original problem. This is done by a new weight function $w : V \to \{0, 1\}$ that models the contribution of v to the parity constraints of its neighbours. Recall that a subset $D \subseteq V$ is feasible if the parity constraints (1) and (2) are satisfied. At the beginning we set $w(v) = 1$ for all $v \in V$. Furthermore, we define for a set $X \subseteq V$

$$w(X) \equiv \sum_{i \in X} w(i) \mod 2$$

and the indicator function

$$\delta_X(v) = \begin{cases} 1 & \text{if } v \in X \\ 0 & \text{otherwise.} \end{cases}$$

Using the additional weight function the parity constraints can now be rewritten as

$$w(N_G(v) \cap D) \equiv \pi(v) \mod 2 \quad \forall v \in V^o \tag{7}$$

and

$$w(N_G(v) \cap D) + \delta_D(v) \equiv \pi(v) \mod 2 \quad \forall v \in V^c. \tag{8}$$

Equations (7) and (8) are called feasibility constraints. Obviously, they are equivalent to the parity constraints (1) and (2) if $w(v) = 1$ for all $v \in V$. Furthermore, a vertex v is called feasible with respect to a set D, if the corresponding feasibility constraint is satisfied. A set $D \subseteq V$ is called feasible if every vertex in D is feasible.

From now on, an instance of the parity domination problem is a quintuple $I = (G, V^o \uplus V^c, c, \pi, w)$ where G is a distance-hereditary graph with a pruning sequence $(v_1, \ldots, v_n)$, $V^o \uplus V^c$ is a partition of the vertex set, c is a cost function, π describes the parity constraints and w denotes the additional weight function. The set of feasible solutions of this instance, i.e., the sets $D \subseteq V$ satisfying (7) and (8), is denoted by $\mathcal{S}_I$.

Definition 5.1. Let $I = (G, V^o \uplus V^c, c, \pi, w)$ and $I' = (G', V'^o \uplus V'^c, c', \pi', w')$ be two instances of the parity domination problem and let $D^* \in \mathcal{S}_I$ be an optimal solution of instance I. Then we say that I' is a reduction of I if the following properties are satisfied:

1. There exists a subset $Y \subseteq \mathcal{S}_I$ with $D^* \in Y$ and a bijection $f : Y \to \mathcal{S}_{I'}$ with

   $$c(D) = c'(f(D)) + K.$$

   for some constant K and all sets $D \in Y$.

2. Let $(v_1, \ldots, v_n)$ be the pruning sequence of G. Then the pruning sequence of G' is given by $(v_1, \ldots, v_{n-1})$.

If I' is a reduction of I, we will write $I' \prec I$.

This definition means that if $I' \prec I$ then it suffices to find an optimal solution D'^* of the instance I', because $f^{-1}(D'^*)$ is an optimal solution of I. The proposed algorithm iteratively computes reduced instances and the corresponding bijections until the remaining graph is trivial.

Let us start the description of the algorithm with a lemma, which is very useful for the development of the proposed approach.

Lemma 5.2. *Let I be a given instance and $v_n = j$ and $v_k = i$ for some $1 \leq k \leq n-1$ (true or false) twins. Furthermore, let*

$$Y = \{D \in \mathcal{S}_I \mid j \in D \ (j \notin D)\}$$

such that Y contains an optimal solution of I. Assume that if $j \in X$ ($j \notin X$) for some vertex set X then i is feasible if and only j is feasible with respect to X. Then there exists an instance I' with $I' \prec I$.

Proof. The proof of this lemma is divided into two parts:

1. Let $j \in D$ for all $S \in Y$. Then a new instance I' with pruning sequence $(v_1, \ldots, v_{n-1})$ can be defined as follows: delete the vertex v_n from the corresponding set V^o or V^c of the vertex partition in instance I in order to get the new vertex partition $V(G') = V'^o \uplus V'^c$. Moreover, set

$$\pi'(v) \equiv \begin{cases} \pi(v) + w(j) \mod 2 & \forall v \in N_G(j) \\ \pi(v) & \text{otherwise.} \end{cases}$$

   Furthermore, define $c'(v) = c(v)$ and $w'(v) = w(v)$ for all $v \in V(G) \setminus \{j\}$. Finally, let $f : Y \to \mathcal{S}_{I'}$ be a function with $f(D) = D \setminus \{j\}$ for all $D \in Y$. Due to the fact that $N_G(v) = N_{G'}(v) \cup \{j\}$ holds for all $v \in N_G(j)$ and $N_G(v) = N_{G'}(v)$ holds for all $v \in V \setminus N_G(j)$ it can be concluded that f maps indeed a set in Y to a set in $\mathcal{S}_{I'}$. Furthermore, f is a bijection with inverse function $f^{-1}(D') = D' \cup \{j\}$ because the feasibility of every vertex $v \in V \setminus \{j\}$ with respect to D' in I' implies feasibility with respect to $f^{-1}(D')$ in I and j is feasible since i is feasible in I. Moreover, $c(D) = c'(f(D)) + c(j)$ holds for all $D \in Y$. Thus, the constructed instance I' is a reduction of I.

2. For the case where $j \notin D$ for all $D \in Y$ a reduced instance I' is constructed in a similar way. The vertex j is deleted from the vertex partition and the functions c', π' and w' are the restrictions of the corresponding functions in I to the domain $V(G) \setminus \{j\}$. Furthermore, we have $f(D) = D$ for all $D \in Y$ and $c(D) = c'(f(D)) = c'(D)$. It is again an easy task to prove that f is a bijection and $I' \prec I$.

□

Observe that Lemma 5.2 can also be applied for the case where the roles of i and j are interchanged. However, we may assume without loss of generality that $j \in D$ or $j \notin D$ for all $D \in Y$ because otherwise i and j are interchanged in the pruning sequence.

5.1 False twins

In this subsection, we describe how to deal with a pruning sequence of an instance I where the last vertex is a false twin. In order to describe the construction of I' several cases have to be distinguished:

- **Case 1** $i, j \in V^c$:
  Observe that for this special situation the following equation holds for any feasible set $D \in \mathcal{S}_I$:

$$\begin{aligned}\pi(i) &\equiv w(N(i) \cap D) + \delta_D(i) = w(N(j) \cap D) + \delta_D(j) + \delta_D(i) - \delta_D(j) \\ &\equiv \pi(j) + \delta_D(i) - \delta_D(j) \mod 2.\end{aligned}$$

  From this equation it can be concluded that if $\pi(i) \neq \pi(j)$ then exactly one vertex is in any feasible set D. On the other hand, if $\pi(i) = \pi(j)$ either both vertices are in a feasible solution D or none of them. The situation is recapitulated in the following table:

if	then
Case 1a: $\pi(i) \neq \pi(j)$	either $i \in D$ or $j \in D$ (cf. Lemma 5.3)
Case 1b: $\pi(i) = \pi(j)$	either $i, j \in D$ or $i, j \notin D$ (cf. Lemma 5.3)

- **Case 2** $i, j \in V^o$:
  Here we have

$$\pi(i) \equiv w(N(i) \cap S) = w(N(j) \cap S) \equiv \pi(j) \mod 2$$

  for all $D \in \mathcal{S}_I$ and it can be seen immediately that if $\pi(i) \neq \pi(j)$ the problem instance I is infeasible, i.e., $\mathcal{S}_I = \emptyset$. On the other hand, if $\pi(i) = \pi(j)$ then i is feasible with respect to D if and only if j is feasible with respect to D. These results are summarized in the following table:

if	then
Case 2a: $\pi(i) \neq \pi(j)$	infeasible (cf. Lemma 5.3)
Case 2b: $\pi(i) = \pi(j)$	no further requirements on i and j (cf. Lemma 5.3)

- **Case 3** $i \in V^c$ and $j \in V^o$:
  In this case

$$\pi(i) \equiv w(N(i) \cap D) + \delta_D(i) = w(N(j) \cap D) + \delta_D(i) \equiv \pi(j) + \delta_D(i) \mod 2$$

  holds for all sets $D \in \mathcal{S}_I$ and the following table gives again some information about $D \in \mathcal{S}_I$ depending on $\pi(i)$ and $\pi(j)$:

if	then
Case 3a: $\pi(i) = \pi(j)$	$i \notin D$ (cf. Lemma 5.2)
Case 3b: $\pi(i) \neq \pi(j)$	$i \in D$ (cf. Lemma 5.2)

  Note that the case where $i \in V^o$ and $j \in V^c$ can be handled in exactly the same way.

The above tables describe properties of a feasible solution D. Moreover, we immediately get the following result: Let $D \subseteq V$ such that one twin is feasible with respect to D and in addition the required property of the corresponding table holds then the other twin is also feasible with respect to D. Using the three different cases mentioned above the following lemma can be proved.

Lemma 5.3. *Let I be an instance of the parity domination problem and $(v_1, \ldots, v_n)$ the corresponding pruning sequence. Suppose $v_n = j$ and $v_k = i$ are false twins for some $1 \leq k \leq n-1$. Then there exists an instance I' with $I' \prec I$, which can be computed in $\mathcal{O}(\deg(j))$ time.*

Proof. Let us start with **Case 1a** where exactly one vertex of i and j is in every feasible set $D \in \mathcal{S}_I$ and $i, j \in V^c$. There are three different subcases that have to be distinguished:

1. $w(i) = w(j) = 1$: In this case the new instance I' is defined as follows: The new vertex partition is given by $V'^o = V^o$ and $V'^c = V^c \setminus \{j\}$, the function π' is defined by
$$\pi'(v) \equiv \begin{cases} \pi(v) - 1 \mod 2 & \text{if } v \in N_G(j) \\ \pi(v) & \text{otherwise,} \end{cases}$$
w' is set to
$$w'(v) = \begin{cases} w(v) & \text{if } v \neq i \\ 0 & \text{if } v = i, \end{cases}$$
and c' is defined by
$$c'(v) = \begin{cases} c(v) - c(j) & \text{if } v = i \\ c(v) & \text{otherwise.} \end{cases}$$
Furthermore, let us define the function $f : \mathcal{S}_I \to \mathcal{S}_{I'}$ with
$$f(D) = \begin{cases} D & \text{if } i \in D \quad (\text{and } j \notin D) \\ D \setminus \{j\} & \text{if } i \notin D \quad (\text{and } j \in D) \end{cases}$$
In the following it is proved that $I' \prec I$, f is the corresponding bijection and $c(D) = c'(f(D)) + c(j)$ for all $D \in \mathcal{S}_I$.
First of all it is shown that $D' := f(D)$ is indeed a feasible solution of instance I', i.e., the feasibility constraints (7) and (8) hold for D'. Note that the vertices in $V(G') \setminus N_{G'}(j)$ are feasible with respect to D' because
$$w'(N_{G'}(v) \cap D') = w(N_G(v) \cap D)$$

and $\pi'(v) = \pi(v)$ for all $v \in V(G') \setminus N_{G'}(j)$. Furthermore,

$$\begin{aligned}
w'(N_{G'}(v) \cap D') &\equiv w'(N_{G'}(v) \setminus \{i\} \cap D') + w'(\{i\} \cap D') && (w'(i) = 0) \\
&\equiv w'(N_G(v) \setminus \{i,j\} \cap D') && (j \notin D') \\
&\equiv w(N_G(v) \setminus \{i,j\} \cap D) && (w(v) = w'(v), v \neq i,j) \\
&\equiv w(N_G(v) \cap D) - w(i)\delta_D(i) - w(j)\delta_D(j) \\
&\equiv w(N_G(v) \cap D) - \delta_D(i) - \delta_D(j) && (w(i) = w(j) = 1) \\
&\equiv w(N_G(v) \cap D) - 1 && (i \in D \text{ or } j \in D) \mod 2
\end{aligned}$$

holds for all $v \in N_G(j)$. Due to the fact that $\pi'(v) \equiv \pi(v) - 1 \mod 2$ it can be concluded that all vertices in the open neighbourhood of j are feasible with respect to D'. The feasibility of vertex i in instance I' with respect to D' follows easily from

$$w'(N_{G'}(i) \cap D') = w(N_{G'}(i) \cap D') = w(N_G(i) \cap D)$$

and the fact that $\pi'(i) = \pi(i)$. Thus, it can be concluded that $f(D) \in \mathcal{S}_{I'}$.
In the next step it is shown that $f^{-1}(D') \in \mathcal{S}_I$ for all $D' \in \mathcal{S}_{I'}$. Let D' be a feasible solution of instance I'. Because of the arguments above it follows that all vertices $v \neq j \in V(G)$ are feasible with respect $f^{-1}(D')$. It is only necessary to show that the deleted vertex j is feasible. However, this follows immediately because we know that vertex i is feasible with respect to a set X that contains either i or j if and only if vertex j is feasible with respect to X. Moreover, it is obvious that f and f^{-1} are injective. Hence, f is bijective. Finally, it follows from the definition of f and c' that $c(D) = c'(f(D)) + c(j)$ for all $D \in \mathcal{S}_I$.
The remaining cases of this proof are not treated in such an explicit way, because they are similar to this case. The construction of the reduced instance I' and the corresponding function f are given, but a formal proof that f is indeed bijective is omitted, but can be done in an analogue way.

2. $w(i) = w(j) = 0$: Here, the new instance I' is defined by setting $\pi'(v) = \pi(v)$ and $w'(v) = w(v)$ for all $v \in V(G')$. The new vertex partition, the cost function and the bijection $f : \mathcal{S}_I \to \mathcal{S}_{I'}$ are the same as in the previous case. Using these definitions it can be shown that $I' \prec I$ and $c(D) = c'(f(D)) + c(j)$ for all $D \in \mathcal{S}_I$.

3. $w(i) \neq w(j)$: Without loss of generality, we may assume that $w(j) = 0$ and $w(i) = 1$. Then everything can be defined as in the case where $w(i) = w(j) = 0$ and all arguments can be used almost literally.

We know that in **Case 1b** either both or none of the false twins i and j are in any feasible solution D. Here the instance I' is given by $V'^o = V^o$, $V'^c = V^c \setminus \{j\}$, $\pi'(v) = \pi(v)$ for all $v \in V(G) \setminus \{j\}$,

$$c'(v) = \begin{cases} c(i) + c(j) & \text{if } v = i \\ c(v) & \text{otherwise} \end{cases}$$

and

$$w'(v) \equiv \begin{cases} w(i) + w(j) \mod 2 & \text{if } v = i \\ w(v) & \text{otherwise.} \end{cases}$$

To see that I' is a reduction of I, one has to consider the function $f : \mathcal{S}_I \to \mathcal{S}_{I'}$ with

$$f(D) = \begin{cases} D & \text{if } i \notin D \quad (\text{and } j \notin D) \\ D \setminus \{j\} & \text{if } i \in D \quad (\text{and } j \in D). \end{cases}$$

This is indeed a bijection and $c(D) = c'(f(D))$ for all $D \in \mathcal{S}_I$.

In case **Case 2** we can assume that $\pi(i) = \pi(j)$, because otherwise $\mathcal{S}_I = \emptyset$ and there is nothing to show. As before subcases have to considered:

1. $w(i) = w(j)$: Note that if $i, j \notin D$ then $D \in \mathcal{S}_I$ if and only if $\tilde{D} := D \cup \{i, j\}$ is feasible, i.e., $\tilde{D} \in \mathcal{S}_I$, because $i, j \in V^o$ and i and j contribute 0 to the feasibility constraints of their neighbours. Furthermore, if $i \notin D$ and $j \in D$ for a set D then $D \in \mathcal{S}_I$ if and only if $\tilde{D} \in \mathcal{S}_I$, where $\tilde{D} = D \setminus \{j\} \cup \{i\}$. These statements hold because $w(N_G(v) \cap D) = w(N_G(v) \cap \tilde{D})$ for all $v \in V(G)$. This implies that there is an optimal solution $D^* \in \mathcal{S}_I$ fulfilling one of the following properties:

if	then
$0 \leq c(i) + c(j)$ and $c(i) \leq c(j)$	either $i, j \notin D^*$ or $j \notin D^* \iff j \notin D^*$
$0 \leq c(i) + c(j)$ and $c(i) > c(j)$	either $i, j \notin D^*$ or $i \notin D^* \iff i \notin D^*$
$0 > c(i) + c(j)$ and $c(i) \leq c(j)$	either $i, j \in D^*$ or $i \in D^* \iff i \in D^*$
$0 > c(i) + c(j)$ and $c(i) > c(j)$	either $i, j \in D^*$ or $j \in D^* \iff j \in D^*$

Table 1: Two-vertex inclusion/exclusion property of an optimal solution D^*.

Depending on the values $c(i)$ and $c(j)$ we can restrict the set of possible optimal solutions and decide whether $j \in D^*$ or $i \in D^*$ for some optimal solution D^*. Hence, we can apply Lemma 5.2.

2. $w(i) \neq w(j)$: Let us assume without loss of generality that $w(j) = 0$. Here it is again true that if $j \notin D$ for a set $D \in \mathcal{S}_I$, then $\tilde{D} = D \cup \{j\}$ is also in $\mathcal{S}_I$ and vice versa. However, this gives the information of the following table and Lemma 5.2 can be applied again.

Observe that **Case 3a** and **Case 3b** have already been treated in Lemma 5.2.

In order to proof the complexity, note that the time complexity is either constant or $\mathcal{O}(\deg(j))$ if it is necessary to update the function values of π for all $v \in N_G(j)$. □

if	then
$c(j) \leq 0$	$j \in D^*$
$c(j) > 0$	$j \notin D^*$

Table 2: One-vertex inclusion/exclusion property for vertex j in an optimal solution D^*.

5.2 True twins

Now we consider true twins, i. e., we are given an instance I with $G = (V, E)$ where the final vertex $v_n = j$ of the pruning sequence is a true twin of i. It is shown how to construct a reduced instance I'. Observe that if $i, j \in V$ are true twins then

$$N[i] = N[j] \text{ and } (i, j) \in E(G)$$

holds. Again we distinguish whether i and j are in V^o or V^c. Let D be a feasible solution.

- **Case 1** $i, j \in V^c$: Observe that

$$\begin{aligned}
\pi(i) &\equiv w(N(i) \cap D) + \delta_D(i) \\
&\equiv w(N[i] \cap D) + \delta_D(i)(1 - w(i)) \\
&\equiv w(N[j] \cap D) + \delta_D(j) - \delta_D(j) + \delta_D(i)(1 - w(i)) \\
&\equiv w(N(j) \cap D) + \delta_D(j) + \delta_D(j)(w(j) - 1) + \delta_D(i)(1 - w(i)) \\
&\equiv \pi(j) + \delta_D(j)(w(j) - 1) + \delta_D(i)(1 - w(i)) \mod 2
\end{aligned}$$

  holds.

  - **Case 1a** If $\pi(i) = \pi(j)$ then every feasible solution $D \in \mathcal{S}_I$ fulfills

$$\delta_D(j)(w(j) - 1) + \delta_D(i)(1 - w(i)) \equiv 0 \mod 2.$$

    Therefore, we get the following subcases:

if	then
$w(i) = 1$, $w(j) = 0$	$j \notin D$ (cf. Lemma 5.2)
$w(i) = 0$, $w(j) = 1$	$i \notin D$ (cf. Lemma 5.2)
$w(i) = 0$, $w(j) = 0$	either $i, j \in D$ or $i, j \notin D$ (cf. Lemma 5.5)
$w(i) = 1$, $w(j) = 1$	no restrictions on i and j (cf. Lemma 5.6)

    If no restrictions are made on i and j then i is feasible with respect to a set D if and only if j is feasible with respect to D.

  - **Case 1b** If $\pi(i) \neq \pi(j)$ then we have

$$\delta_D(j)(w(j) - 1) + \delta_D(i)(1 - w(i)) \equiv 1 \mod 2$$

    and we get

if	then
$w(i) = 1$, $w(j) = 1$	infeasible
$w(i) = 1$, $w(j) = 0$	$j \in D$ (cf. Lemma 5.2)
$w(i) = 0$, $w(j) = 1$	$i \in D$ (cf. Lemma 5.2)
$w(i) = 0$, $w(j) = 0$	either $i \in D$ or $j \in D$ (cf. Lemma 5.4)

- **Case 2** $i, j \in V^o$: Observe that

$$\begin{aligned}\pi(i) &\equiv w(N(i) \cap D) \\ &\equiv w(N[i] \cap D) - \delta_D(i)w(i) \\ &\equiv w(N[j] \cap D) - \delta_D(i)w(i) \\ &\equiv w(N(j) \cap D) + \delta_D(j)w(j) - \delta_D(i)w(i) \\ &\equiv \pi(j) + \delta_D(j)w(j) - \delta_D(i)w(i) \mod 2\end{aligned}$$

  holds.

  - **Case 2a** $\pi(i) = \pi(j)$ then every $D \in \mathcal{S}_I$ fulfills

$$\delta_D(j)w(j) - \delta_D(i)w(i) \equiv 0 \mod 2.$$

    We distinguish all possible cases for the weight functions:

if	then
$w(i) = 1$, $w(j) = 1$	either $i, j \in D$ or $i, j \notin D$ (cf. Lemma 5.5)
$w(i) = 1$, $w(j) = 0$	$i \notin D$ (cf. Lemma 5.2)
$w(i) = 0$, $w(j) = 1$	$j \notin D$ (cf. Lemma 5.2)
$w(i) = 0$, $w(j) = 0$	no restrictions on i and j (cf. Lemma 5.6)

  - **Case 2b** $\pi(i) \neq \pi(j)$: Then $D \in \mathcal{S}_I$ satisfies

$$\delta_D(j)w(j) - \delta_D(i)w(i) \equiv 1 \mod 2.$$

if	then
$w(i) = 1$, $w(j) = 1$	either $i \in D$ or $j \in D$ (cf. Lemma 5.4)
$w(i) = 1$, $w(j) = 0$	$i \in D$ (cf. Lemma 5.2)
$w(i) = 0$, $w(j) = 1$	$j \in D$ (cf. Lemma 5.2)
$w(i) = 0$, $w(j) = 0$	infeasible

- **Case 3** $i \in V^o$ and $j \in V^c$:

$$\begin{aligned}\pi(i) &\equiv w(N(i) \cap D) \\ &\equiv w(N[i] \cap D) - \delta_D(i)w(i) \\ &\equiv w(N[j] \cap D) - \delta_D(i)w(i) \\ &\equiv w(N(j) \cap D) + \delta_D(j) + \delta_D(j)(w(j) - 1) - \delta_D(i)w(i) \\ &\equiv \pi(j) + \delta_D(j)(w(j) - 1) - \delta_D(i)w(i) \mod 2\end{aligned}$$

– **Case 3a** $\pi(i) = \pi(j)$: Then for $D \in \mathcal{S}_I$

$$\delta_D(j)(w(j) - 1) - \delta_D(i)w(i) \equiv 0 \mod 2$$

holds. We distinguish all possible cases for the weight functions:

if	then
$w(i) = 1$, $w(j) = 1$	$i \notin D$ (cf. Lemma 5.2)
$w(i) = 1$, $w(j) = 0$	either $i, j \in D$ or $i, j \notin D$ (cf. Lemma 5.5)
$w(i) = 0$, $w(j) = 1$	no restrictions on i and j (cf. Lemma 5.6)
$w(i) = 0$, $w(j) = 0$	$j \notin D$ (cf. Lemma 5.2)

– **Case 3b** $\pi(i) \neq \pi(j)$: For $D \in \mathcal{S}_I$ we have

$$\delta_D(j)(w(j) - 1) - \delta_D(i)w(i) \equiv 1 \mod 2.$$

if	then
$w(i) = 1$, $w(j) = 1$	$i \in D$ (cf. Lemma 5.2)
$w(i) = 1$, $w(j) = 0$	either $i \in D$ or $j \in D$ (cf. Lemma 5.4)
$w(i) = 0$, $w(j) = 1$	infeasible
$w(i) = 0$, $w(j) = 0$	$j \in D$ (cf. Lemma 5.2)

If $i \in V^c$ and $j \in V^o$ then i and j can be interchanged in the pruning sequence and Case 3 can be applied.

The above discussion describes the properties of a feasible solution $D \in \mathcal{S}_I$ depending on the neighbourhood, parity and weight of the true twins $i, j \in V$. The goal is to delete one vertex and to update the data in order to get a reduced instance I'.

The first lemma within this discussion deals with the case where either $i \in D$ or $j \in D$ for all $D \in \mathcal{S}_I$.

Lemma 5.4. *Let $G = (V, E)$ be distance-hereditary with pruning sequence $(v_1, \ldots, v_n)$, $v_n = j$ and true twins $i, j \in V$. If either $i \in D$ or $j \in D$ for all $D \in \mathcal{S}_I$ then there exists a reduction I' of I which can be computed in $\mathcal{O}(\deg(j))$ time.*

Proof. Let $i, j \in V$ be true twins. Then the property that either $i \in D$ or $j \in D$ holds for all $D \in \mathcal{S}_I$ is fulfilled if we have one of the following cases:

1. $i, j \in V^c$, $\pi(i) \neq \pi(j)$ and $w(i) = w(j) = 0$ (**Case 1b**) or
2. $i \in V^o$, $j \in V^c$, $\pi(i) \neq \pi(j)$ and $w(i) = 1$, $w(j) = 0$ (**Case 3b**) or
3. $i, j \in V^o$, $\pi(i) \neq \pi(j)$ and $w(i) = w(j) = 1$ (**Case 2b**).

For the first two cases the reduction I' is defined in the same way since its correctness bases on the fact that $w(j) = 0$. The cost function of I' is defined as follows:

$$c'(v) = \begin{cases} c(v) - c(j) & \text{if } v = i \\ c(v) & \text{otherwise.} \end{cases}$$

All remaining data are the same as in I. The bijection is defined by $f : \mathcal{S}_I \to \mathcal{S}_{I'}$ with

$$f(D) = \begin{cases} D & \text{if } i \in D \quad (\text{and } j \notin D) \\ D \setminus \{j\} & \text{if } i \notin D \quad (\text{and } j \in D). \end{cases}$$

For the remaining third case the neighbours $v \in N(j) \setminus \{i\}$ do not distinguish between $i \in D$ and $j \in D$ since $w(i) = w(j) = 1$. In both cases i and j contribute 1 to the feasibility constraint of v. Moreover, if $j \in D$ then i and j contribute 1 to the feasibility constraint of i while if $i \in D$ then i and j contribute 0 since $i \in V^o$. Hence, I' is defined as follows: $V'^o = V^o \setminus \{i, j\}$, $V'^c = V^c \cup \{i\}$ and

$$c'(v) = \begin{cases} c(i) - c(j) & \text{if } v = i \\ c(v) & \text{otherwise} \end{cases}$$

$$\pi'(v) \equiv \begin{cases} \pi(v) - 1 \mod 2 & \text{if } v \in N(j) \\ \pi(v) & \text{otherwise} \end{cases}$$

and finally

$$w'(v) = \begin{cases} 0 & \text{if } v = i \\ w(v) & \text{otherwise.} \end{cases}$$

The function $f : \mathcal{S}_I \to \mathcal{S}_{I'}$ is defined as in the previous case.

Observe that in both cases f is bijective and $c(D) = c(f(D)) + c(j)$ holds for all $D \in \mathcal{S}_I$. □

In the next lemma we deal with the case where we have to choose either both twins or none of them.

Lemma 5.5. *Let $G = (V, E)$ be distance-hereditary with pruning sequence $(v_1, \ldots, v_n)$, $v_n = j$ and true twins $i, j \in V$. If either $i, j \in D$ or $i, j \notin D$ for all $D \in \mathcal{S}_I$ then there exists an reduction I' of I that can be computed in $\mathcal{O}(\deg(j))$ time.*

Proof. Let $i, j \in V$ be true twins. Then the property that either $i, j \in D$ or $i, j \notin D$ holds for all $D \in \mathcal{S}_I$ is fulfilled if we have one of the following cases:

1. $i, j \in V^c$, $\pi(i) = \pi(j)$ and $w(i) = w(j) = 0$ (**Case 1a**) or
2. $i \in V^o$, $j \in V^c$, $\pi(i) = \pi(j)$ and $w(i) = 1$, $w(j) = 0$ (**Case 3a**) or

3. $i, j \in V^o$, $\pi(i) = \pi(j)$ and $w(i) = w(j) = 1$ (**Case 2a**).

Again the first two cases are considered together since $w(j) = 0$. I' is defined in the following way: G' is obtained from G by deleting j and setting $c'(i) = c(i) + c(j)$ and $c'(v) = c(v)$ for $v \in V' \setminus \{i\}$. All remaining data of I' stay unchanged. Then we have

$$f(D) = \begin{cases} D \setminus \{j\} & \text{if } i \in D \quad (\text{and } j \in D) \\ D & \text{if } i \notin D \quad (\text{and } j \notin D) \end{cases}$$

for $D \in \mathcal{S}_I$ and $Y = \mathcal{S}_I$. Observe that $c(D) = c(f(D))$ holds.

The third case where $w(i) = w(j) = 1$ is considered next. The vertices i and j always contribute 0 to the feasibility constraint for all $v \in N(j) \setminus \{i\}$. However, if $i, j \in D$ then 1 is contributed to the feasibility constraint of i and 0 otherwise ($i, j \notin D$). Hence, i changes from V^o to V'^c, i. e., $V'^o = V^o \setminus \{i, j\}$, $V'^c = V^c \cup \{i\}$ and

$$c'(v) = \begin{cases} c(i) + c(j) & \text{if } v = i \\ c(v) & \text{otherwise,} \end{cases}$$

$$w'(v) = \begin{cases} 0 & \text{if } v = i \\ w(v) & \text{otherwise,} \end{cases}$$

$$\pi'(v) \equiv \begin{cases} \pi(v) - 1 \mod 2 & \text{if } v \in N(j) \\ \pi(v) & \text{otherwise.} \end{cases}$$

The function f is defined as above and $c(D) = c(f(D))$ for $D \in \mathcal{S}_I$. □

Finally, we investigate the case where no additional requirement is given for i and j.

Lemma 5.6. *Let $G = (V, E)$ be distance-hereditary with pruning sequence $(v_1, \ldots, v_n)$, $v_n = j$ and true twins $i, j \in V$ and let $D \subseteq V$. If j is feasible with respect to D if and only if i is feasible with respect to D (without any restrictions on i and j) then there exists a reduced instance $I' \prec I$ which can be determined in $\mathcal{O}(\deg(j))$ time.*

Proof. Observe that this property holds for the following three cases:

1. $i, j \in V^c$, $\pi(i) = \pi(j)$ and $w(i) = w(j) = 1$ (**Case 1a**) or
2. $i, j \in V^o$, $\pi(i) = \pi(j)$ and $w(i) = w(j) = 0$ (**Case 2a**) or
3. $i \in V^o$, $j \in V^c$, $\pi(i) = \pi(j)$ and $w(i) = 0$, $w(j) = 1$ (**Case 3a**).

Let us start with the first case where $w(i) = w(j) = 1$ and $i, j \in V^c$: Let $\tilde{D} = D \cup \{i, j\}$ with $i, j \notin D$. We show that $D \in \mathcal{S}_I$ if and only if $\tilde{D} \in S_I$: Let $v \in N(j) \setminus \{i\}$ then

$$w(N(v) \cap \tilde{D}) \equiv w(N(v) \cap D) + w(i) + w(j) \equiv w(N(v) \cap D) \mod 2$$

holds because $w(i) + w(j)$ is even. Hence, v is feasible with respect to D if and only if it is feasible with respect to $\tilde{D}$. And for vertex i we have

$$\begin{aligned} w(N(i) \cap D) + \delta_D(i) &\equiv w(N(i) \cap \tilde{D}) + w(j) + \delta_D(i) \\ &\equiv w(N(i) \cap \tilde{D}) + \delta_{\tilde{D}}(i) \mod 2. \end{aligned}$$

The first equation is due to the definition of D and $\tilde{D}$ and the second one holds because $\delta_D(i) = 0$ and $\delta_{\tilde{D}}(i) = 1 = w(j)$. Since $i \in V^c$, we conclude that i is feasible with respect to D if and only if it is feasible with respect to $\tilde{D}$. Since j is always feasible if i is feasible we conclude the above statement. Observe that in both alternatives i and j contribute 0 to the feasibility constraint of the neighbours of j.

On the other hand, let $\tilde{D} = D \setminus \{i\} \cup \{j\}$ with $i \in D$. Again we can show that D is feasible if and only if $\tilde{D}$ is feasible and i and j contribute 1 to the feasibility constraint of the neighbours of j.

The above discussion implies that there is an optimal solution D^* satisfying the two-vertex inclusion/exclusion property listed in Table 1. Since these properties describe either excluding or including a vertex in D^* we can apply Lemma 5.2.

The remaining two cases are simpler because $w(i) = 0$. Note that the one-vertex property of Table 2 holds for vertex i. Therefore, we can apply Lemma 5.2 once more. □

5.3 Pendant

In this final subsection, we consider pendants. If j is a pendant of i then $N(j) = \{i\}$.

- **Case 1** $j \in V^c$: Then

  $$\pi(j) \equiv w(N(j) \cap D) + \delta_D(j) \equiv w(i)\delta_D(i) + \delta_D(j) \mod 2$$

  holds. We distinguish the following cases:

  - **Case 1a** $w(i) = 0$: Then j is feasible with respect to D if and only if $\delta_D(j) = \pi(j)$.
  - **Case 1b** $w(i) = 1$:

if	then
Case 1b(1): $\pi(j) = 0$	either $i, j \in D$ or $i, j \notin D$
Case 1b(2): $\pi(j) = 1$	either $i \in D$ or $j \in D$

- **Case 2** $j \in V^o$: Then

  $$\pi(j) \equiv w(N(j) \cap D) \equiv w(i)\delta_D(i) \mod 2$$

  holds.

- **Case 2a** $w(i) = 0$: If $\pi(j) = 1$ then the problem is infeasible. Otherwise j is feasible with respect to every subset D.
- **Case 2b** $w(i) = 1$: Then j is feasible with respect to D if and only if $\delta_D(j) = \pi(j)$.

All cases described above are considered in the following lemma:

Lemma 5.7. *Let $G = (V, E)$ be distance-hereditary with pruning sequence $(v_1, \ldots, v_n)$, $v_n = j$ and j is a pendant of i. Then there exists a reduction I' of I which can be determined in constant time.*

Proof. Throughout this proof we define a reduction I' such that j is removed and all data except the cost and parity function value of i and the vertex partition remain unchanged.

We start with **Case 1a** and **Case 2b** since in both cases j is feasible with respect to D if and only if $\delta_D(j) = \pi(j)$. Observe that this property means that $j \in D$ if $\pi(j) = 1$ and $j \notin D$ if $\pi(j) = 0$ for all $D \in \mathcal{S}_I$. Here we define $c'(i) = c(i)$, $\pi'(i) \equiv \pi(i) - \pi(j)w(j)$ and $V'^o = V^o$, $V'^c = V^c \setminus \{j\}$. Furthermore, $f := \mathcal{S}_I \to \mathcal{S}_{I'}$ with

$$f(D) = \begin{cases} D & \text{if } \pi(j) = 0 \\ D \setminus \{j\} & \text{if } \pi(j) = 1. \end{cases}$$

Note that f is bijective and $c(D) = c(f(D)) + \pi(j)c(j)$.

Now consider **Case 1b(1)** with $\pi(j) = 0$, i. e., either $i, j \in D$ or $i, j \notin D$: If $w(j) = 0$ then $c'(i) = c(i) + c(j)$ and all other data remain unchanged. However, if $w(j) = 1$ then it makes a difference whether $i \in V^o$ or $i \in V^c$. If $i \in V^o$ then i and j contribute 0 to the feasibility constraint of i if $i, j \notin D$ and 1 otherwise ($i, j \in D$). Hence, we set $c'(i) = c(i) + c(j)$, $\pi'(i) = \pi(i)$ and $V'^o = V^o \setminus \{i\}$, $V'^c = V^c \cup \{i\} \setminus \{j\}$.

Otherwise if $i \in V^c$ then i and j contribute 0 to i independent of $i, j \in D$ or $i, j \notin D$. Thus, we define $c'(i) = c(i) + c(j)$, $\pi'(i) = \pi(i)$ and $V'^o = V^o \cup \{i\}$, $V'^c = V^c \setminus \{i, j\}$.

In all cases of Case 1b(1) we have $f : \mathcal{S}_I \to \mathcal{S}_{I'}$ with

$$f(D) = \begin{cases} D & \text{if } i \notin D \quad (\text{and } j \notin D) \\ D \setminus \{j\} & \text{if } i \in D \quad (\text{and } j \in D). \end{cases}$$

Moreover, we have $c(D) = c(f(D))$.

Then we investigate **Case 1b(2)** with $\pi(j) = 1$, i. e., either $i \in D$ or $j \in D$. If $w(j) = 0$ then we set $c'(i) = c(i) - c(j)$ and all other data remain unchanged. The interesting case occurs if $w(j) = 1$. If $i \in V^c$ then i and j contribute 1 to the feasibility constraint of i independent of $i \in D$ or $j \in D$. Hence, we set $c'(i) = c(i) - c(j)$, $\pi'(i) \equiv \pi(i) - 1 \mod 2$ and $V'^o = V^o \cup \{i\}$, $V'^c = V^c \setminus \{i, j\}$. For both cases of Case 1b(2) we set $f : \mathcal{S}_I \to \mathcal{S}_{I'}$ with

$$f(D) = \begin{cases} D & \text{if } i \in D \quad (\text{and } j \notin D) \\ D \setminus \{j\} & \text{if } i \notin D \quad (\text{and } j \in D) \end{cases}$$

Observe that $c(D) = c(f(D)) + c(j)$. However, if $i \in V^o$ then i and j contribute 0 to the feasibility constraint of i if $i \in D$ and 1 if $j \in D$. Here we define $c'(i) = c(j) - c(i)$, $\pi'(i) = \pi(i)$ and $V'^o = V^o \setminus \{i\}$, $V'^c = V^c \cup \{i\} \setminus \{j\}$. In this case we have $f : \mathcal{S}_I \to \mathcal{S}_{I'}$ with

$$f(D) = \begin{cases} D \setminus \{i\} & \text{if } i \in D \quad (\text{and } j \notin D) \\ D \cup \{i\} \setminus \{j\} & \text{if } i \notin D \quad (\text{and } j \in D) \end{cases}$$

Moreover, we have $c(D) = c(f(D)) + c(i)$.

Finally, we consider **Case 2a** where j is feasible with respect to every subset D and $w(i) = 0$.

We start with the rather simple case where $w(j) = 0$: Since j contributes 0 to the feasibility constraint of i, we conclude that every optimal solution D^* includes j if $c(j) < 0$ and excludes j if $c(j) > 0$. Thus, the one-vertex inclusion/exclusion property of Table 2 holds for j. Therefore, I' can defined as in Case 1a.

Now let $w(j) = 1$ and $i \in V^o$. Let $\tilde{D} = D \setminus \{i\} \subseteq V$. Then D is feasible if and only if $\tilde{D}$ is feasible since $i \in V^o$ and $w(i) = 0$. Therefore, the one-vertex inclusion/exclusion property of Table 2 holds for i. We define $c'(i) = c(j)$, $\pi'(i) = \pi(i)$, $V'^o = V^o \setminus \{i, j\}$, $V'^c = V^c \cup \{i\}$ and $f : Y \to \mathcal{S}_{I'}$ with

$$f(D) = \begin{cases} D \cup \{i\} \setminus \{j\} & \text{if } j \in D \\ D \setminus \{i\} & \text{if } j \notin D. \end{cases}$$

Here we have $c(D) = c(f(D))$ if $c(i) > 0$ and $c(D) = c(f(D)) + c(i)$ if $c(i) \leq 0$.

Finally, let $w(j) = 1$ and $i \in V^c$. Let $\tilde{D} = D \cup \{i, j\}$ with $i, j \notin D$. Then D is feasible if and only if $\tilde{D}$ is feasible and in both solutions i and j contribute 0 to the feasibility constraint of i. On the other hand, let $\tilde{D} = D \cup \{i\} \setminus \{j\}$ with $j \in D$ and $i \notin D$. Then D is a feasible solution if and only if $\tilde{D}$ is feasible and i and j contribute 1 to the feasibility constraint of i. Therefore, the two-vertex inclusion/exclusion property of Table 1 holds. The first and the last case of Table 1 are equivalent to Case 1a. For the remaining two cases we define $c'(i) = c(j)$, $\pi'(i) = \pi(i)$, $V'^o = V^o \setminus \{i, j\}$, $V'^c = V^c \cup \{i\}$ and $f : Y \to \mathcal{S}_{I'}$ with

$$f(D) = \begin{cases} D \cup \{i\} \setminus \{j\} & \text{if } j \in D \\ D \setminus \{i\} & \text{if } j \notin D \end{cases}$$

Observe that $c(D) = c(f(D))$ if $c(i) > c(j)$ and $c(D) = c(f(D)) + c(i)$ if $c(i) \leq c(j)$. □

In the previous subsections we described how to reduce the original instance until the graph contains a single vertex. Let G be the final graph with $V(G) = \{v\}$ then an optimal solution D_1^* is of the following form:

$v \in V^c$	$\pi(v) = 1$	$D_1^* = \{v\}$
	$\pi(v) = 0$	$D_1^* = \emptyset$
$v \in V^o$	$\pi(v) = 1$	infeasible
	$\pi(v) = 0$	if $c(v) < 0$ then $D_1^* = \{v\}$ otherwise $D_1^* = \emptyset$

Let f_j be the bijection applied in that iteration where vertex j is deleted. Note that in each iteration exactly one vertex is deleted, i. e., there are $n-1$ iterations. Now it is possible to determine an optimal solution

$$D^* = f_n^{-1}(f_{n-1}^{-1}(\dots f_2^{-1}(D_1^*)))$$

of the original problem. Since iteration j takes at most $\mathcal{O}(\deg(j))$ time we can state the main result of this section:

Theorem 5.8. *The parity domination problem can be solved in $\mathcal{O}(m)$ time on distance-hereditary graphs with m edges.*

6 Conclusion

In this paper a generalized parity domination problem is investigated where we deal with open and closed as well as with even and odd neighbourhoods. Furthermore, a vertex cost function is considered. For this problem linear time algorithms on graphs with bounded treewidth and on distance-hereditary graphs are developed. Observe that the classical domination problem can be solved efficiently on these graph classes. Due to the fact that the parity domination problem is strongly related to the classical dominating set problem it may be an interesting approach to consider other graph classes where the dominating set problem is solvable in polynomial time. Moreover, investigations of the problem where the congruence relation in the feasibility constraints are modulo k for an arbitrary $k \in \mathbb{N}$ are of interest.

References

[1] A. Amin, P. Slater, Neighborhood Domination with Parity Restriction in Graphs. *Congressus Numerantium* 91, 1992, 19–30.

[2] H.-J. Bandelt, H.M. Mulder, Distance-hereditary graphs. *Journal of Combinatorial Theory, Series B* 41, 1986, 182–208.

[3] H. Bodlaender, A tourist guide through treewidth. *Acta Cybernetica* 11, 1993, 1–21.

[4] H. Bodlaender, A linear time algorithm for finding tree-decomposition of small treewidth. *SIAM Journal of Computing* 25, 1996, 1305–1317.

[5] A. Brandstädt and F.F. Dragan, A Linear-Time algorithm for connected r-Domination and Steiner Tree on Distance-Hereditary Graphs. *Networks* 31, 1998, 177–182.

[6] A. Brandstädt, V.B. Le, and J.P. Spinrad, Graph Classes — A survey. *SIAM Monographs on Discrete Mathematics and Applications*, 1999.

[7] H.J. Broersma, E. Dahlhaus, and T. Kloks, A linear time algorithm for minimum fill-in and treewidth for distance hereditary graphs. *Discrete Applied Mathematics* 99, 2000, 367–400.

[8] Y. Caro, Simple Proofs to Three Parity Theorems. *Ars Combinatoria* 42, 1996, 175–180.

[9] Y. Caro, M. Jacobson, On non-zero (mod k) dominating sets. *Discussiones Mathematicae Graph Theory* 23, 2003, 189–199.

[10] M.S. Chang, S.C. Wu, G.J. Chang, and H.G. Yeh, Domination in distance-hereditary graphs. *Discrete Applied Mathematics* 116, 2002, 103–113.

[11] O. Cogis and E. Thierry, Computing maximum stable sets for distance-hereditary graphs. *Discrete Optimization* 2, 2005, 185–188.

[12] A. D'Atri and M. Moscarini, Distance-hereditary graphs, Steiner trees, and connected domination. *SIAM Journal of Computing* 17, 1988, 521–538.

[13] R. Dawes, Minimum Odd Neighbourhood Covers for Trees. *Lecture Notes in Computer Science* 507, 1989, 161 –169.

[14] R. Diestel, Graph theory, Second edition. *Springer*, New York, 2000.

[15] G. Damiand, M. Habib, and C. Paul, A simple paradigm for graph recognition: application to cographs and distance hereditary graphs. *Theoretical Computer Science* 263, 2001, 99–111.

[16] P.L. Hammer and F. Maffray, Completely separable graphs. *Discrete Applied Mathematics* 27, 1990, 85–99.

[17] T. Haynes, S. Hedetniemi, and P. Slater, Fundamental of Domination in Graphs *Marcel Dekker*, 1998.

[18] E. Howorka, A characterization of distance-hereditary graphs. *Quart. J. Math. Oxford*, Ser. 2, 28, 1977, 417-420.

[19] S.Y. Hsieh, C.W. Ho, T.S. Hsu, and M.T. Ko, The hamiltonian problem on distance-hereditary graphs. *Discrete Applied Mathematics* 154, 2006, 508–524.

[20] R.W. Hung and M.S. Chang, Linear-time algorithms for the Hamiltonian problems on distance-hereditary graphs. *Theoretical Computer Science* 341, 2005, 411–440.

[21] T. Kloks, Treewidth — Computations and approximations. *Lecture Notes in Computer Science* 842, 1994.

[22] H. Moser. Exact algorithms for generalizations of vertex cover. *Master thesis*, University of Jena, (2005).

[23] N. Robertson, P.D. Seymour, Graph Minors II. Algorithmic aspects of tree-width. *Journal of Algorithms* 7, 1986, 309–322.

[24] K. Sutner, Linear Cellular automata and the Garden of Eden. *The Mathematical Intelligence* 11, 1989, 49–53

[25] H.G. Yeh, G.J. Chang, Weighted connected domination and Steiner trees in distance-hereditary graphs. *Discrete Applied Mathematics* 87, 1998, 245–253.

[26] H.G. Yeh and G.J. Chang, Weighted connected k-domination and weighted k-dominating clique in distance hereditary graphs. *Theoretical Computer Science* 263, 2001, 3–8.

[27] H.G. Yeh and G.J. Chang, Centers and medians of distance-hereditary graphs. *Discrete Mathematics* 265, 2003, 297–310.