An IP-based swapping algorithm for the metric dimension and minimal doubly resolving set problems in hypercubes

Alain Hertz

Polytechnique Montréal and GERAD, Canada

alain.hertz@gerad.ca

March 13, 2017

Abstract

We consider the problems of determining the metric dimension and the minimum cardinality of doubly resolving sets in n-cubes. Most heuristics developed for these two NP-hard problems use a function that counts the number of pairs of vertices that are not (doubly) resolved by a given subset of vertices, which requires an exponential number of distance evaluations, with respect to n. We show that it is possible to determine whether a set of vertices (doubly) resolves the n-cube by solving an integer program with $O(n)$ variables and $O(n)$ constraints. We then demonstrate that small resolving and doubly resolving sets can easily be determined by solving a series of such integer programs within a swapping algorithm. Results are given for hypercubes having up to a quarter of a billion vertices, and new upper bounds are reported.

1 Introduction

Consider a connected undirected graph G, and let $d(u,v)$ be the distance between vertices u and v in G. A vertex x *resolves* two vertices u and v if $d(x,u) \neq d(x,w)$. A subset W of vertices *resolves* G if every two vertices in G are resolved by some vertex of W. The *metric dimension* of G, denoted $\beta(G)$, is the minimum cardinality of a resolving set for G. The problem of determining the metric dimension of a graph was introduced independently by Slater [12] and by Harary and Melter [5]. It arises in many areas, including robot navigation [6], telecommunication [1] and chemistry [4].

Cáceres et al. [2] have introduced the notion of doubly resolving sets : vertices x and y *doubly resolve* vertices u and v if $d(u,x) - d(u,y) \neq d(v,x) - d(v,y)$. A subset W of vertices *doubly resolves* G if every two vertices in G are doubly resolved by two vertices of W. The minimum cardinality of a doubly resolving set for G is denoted $\Psi(G)$. Clearly, every doubly resolving set is a resolving set, which implies $\beta(G) \leq \Psi(G)$ for all graphs G.

Determining $\beta(G)$ and $\Psi(G)$ are NP-hard problems, as proved in [6] and [8], respectively. In this paper, we focus on n-cubes for which the computation of $\beta(G)$ and $\Psi(G)$ is particularly challenging, due to the exponential growth of the number of vertices, with respect to n. Various heuristics and metaheuristics have been developed for computing $\beta(G)$ and $\Psi(G)$ in n-cubes, including greedy algorithms [11], genetic algorithms [7, 8], variable neighborhood search [9] and particle swarm optimization [10]. They all use an objective function $f(W)$ which counts the number of pairs of vertices that are not (doubly) resolved by a given subset W of vertices. Hence, W is a (doubly) resolving set if and only if $f(W) = 0$. Determining $f(W)$ requires $O(2^{2n})$ comparisons of distances, which is time consuming for large values of n. Instead of computing $f(W)$, we rather determine whether $f(W)$ in strictly positive by solving an integer programming problem (IP for short) with $O(n)$ variables and $O(n)$ constraints. We show that small (doubly) resolving sets can easily by generated by solving a series of such IPs.

The next section contains basic definitions and properties on $\beta(G)$ and $\Psi(G)$ in hypercubes. The IP model is described in Section 3, while a swapping heuristic based on repeated solutions of such IPs is proposed in Section 4. Computational experiments are reported in Section 5.

2 Definitions and properties

For a positive integer n, let Q_n be the n-dimensional hypercube, also called n-cube, with vertex set $\{0,1\}^n$. The *Hamming distance* $d(\boldsymbol{x}, \boldsymbol{y})$ between two vertices $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{y} = (y_1, \ldots, y_n)$ is the number of integers i such that $x_i \neq y_i$. A graph can be associated with Q_n by linking two vertices $\boldsymbol{x}$ and $\boldsymbol{y}$ with an edge if and only if $d(\boldsymbol{x}, \boldsymbol{y}) = 1$. In what follows, we use β_n (instead of $\beta(Q_n)$) and Ψ_n (instead of $\Psi(Q_n)$) for the minimum cardinality of a resolving and doubly resolving set in Q_n. For $\boldsymbol{x} = (x_1, \ldots, x_n)$ in Q_n, we denote $\bar{\boldsymbol{x}} = (1 - x_1, \ldots, 1 - x_n)$ its opposite (or complement). Also, for two vertices $\boldsymbol{x} = (x_1, \ldots, x_s) \in Q_s$ and $\boldsymbol{y} = (y_1, \ldots, y_t) \in Q_t$, we denote $\boldsymbol{xy}$ the vertex $(x_1, \ldots, x_s, y_1, \ldots, y_t)$ in Q_{s+t}. For example, for $\boldsymbol{x} \in Q_n$, $\boldsymbol{x}(0)$ is the vertex of Q_{n+1} obtained from $\boldsymbol{x}$ by adding 0 as nth component. The following interesting upper bound on β_n was proved in [2].

Property 2.1 $\quad$ *$\beta_n \leq \beta_{n-i} + \Psi_i - 1$ for all $i = 1, \ldots, n-1$, $n \geq 2$.*

The proof of this upper bound on β_n is that if S resolves Q_{n-i}, T doubly resolves Q_i, $\boldsymbol{s} \in S$, and $\boldsymbol{t} \in T$, then $\{\boldsymbol{sv} : \boldsymbol{v} \in T\} \cup \{\boldsymbol{at} : \boldsymbol{a} \in S\}$ resolves Q_n. Since $\beta_1 = 1$ and $\Psi_1 = 2$ (which is easy to check), we get

$$\beta_n \leq \min\{\Psi_{n-1}, \beta_{n-1} + 1\}. \tag{1}$$

In particular, if $W = \{\boldsymbol{x}^{\boldsymbol{1}}, \ldots, \boldsymbol{x}^{|\boldsymbol{W}|}\}$ is a known resolving set for Q_{n-1}, it is easy to construct a resolving set W' for Q_n with $|W'| = |W| + 1$. Indeed, let $\boldsymbol{y}^j = \boldsymbol{x}^j(0)$, $j = 1, \ldots, |W|$, and let $\boldsymbol{y}^{|W|+1} = \boldsymbol{x}^1(1)$. Then $W' = \{\boldsymbol{y}^1, \ldots, \boldsymbol{y}^{|W|+1}\}$ resolves Q_n.

For example, since $W = \{(0,0),(0,1)\}$ resolves Q_2, $W' = \{(0,0,0),(0,1,0),(0,0,1)\}$ resolves Q_3.

Similarly, knowing that $W = \{\boldsymbol{x^1}, \ldots, \boldsymbol{x^{|W|}}\}$ doubly resolves Q_{n-1} implies that $W' = \{\boldsymbol{x}^j(0) : 1 \leq j \leq |W|\}$ resolves Q_n. For example, since $W = \{(0),(1)\}$ doubly resolves Q_1, $W' = \{(0,0),(1,0)\}$ resolves Q_2.

As noticed in [11], if W resolves Q_n, then the set obtained from W by replacing one of its element $\boldsymbol{x}$ by its opposite $\bar{\boldsymbol{x}}$ also resolves Q_n. Also, it is easy to prove that if W resolves Q_n, then the set obtained by removing the kth component $(1 \leq k \leq n)$ to every vertex in W resolves Q_{n-1}, which proves that $\beta_n \geq \beta_{n-1}$. These observations lead to the following property [3].

Property 2.2 *If a resolving set W for Q_n contains two vertices $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $d(\boldsymbol{x},\boldsymbol{y}) = 1$ or $d(\boldsymbol{x},\boldsymbol{y}) = n-1$, then $\beta_{n-1} \leq |W| - 1$.*

Proof. Assume, W contains two vertices $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $d(\boldsymbol{x},\boldsymbol{y}) = 1$, and let k be such that $x_k \neq y_k$. The set W' obtained by removing the kth component to every vertex of W resolves Q_{n-1}. Since $(x_1,\ldots,x_{k-1},x_{k+1},\ldots,x_n) = (y_1,\ldots,y_{k-1},y_{k+1},\ldots,y_n)$, W' contains at most $|W|-1$ vertices, which proves that $\beta_{n-1} \leq |W|-1$.

If W contains two vertices $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $d(\boldsymbol{x},\boldsymbol{y}) = n-1$, then the set obtained by replacing $\boldsymbol{x}$ by $\bar{\boldsymbol{x}}$ resolves Q_n with $d(\bar{\boldsymbol{x}},\boldsymbol{y}) = 1$, and we have proved above that this implies $\beta_{n-1} \leq |W| - 1$. □

As shown in [8], the following property if helpful when trying to identify doubly resolving sets.

Property 2.3 *A set $W = \{\boldsymbol{x^1}, \ldots, \boldsymbol{x^{|W|}}\}$ doubly resolves Q_n if and only if for every pair $\boldsymbol{u}, \boldsymbol{v}$ of distinct verties in Q_n there exists an integer $j \in \{2, \ldots, |W|\}$ such that $d(\boldsymbol{u}, \boldsymbol{x^1}) - d(\boldsymbol{u}, \boldsymbol{x^j}) \neq d(\boldsymbol{v}, \boldsymbol{x^1}) - d(\boldsymbol{v}, \boldsymbol{x^j})$.*

In other words, among the pairs $\boldsymbol{x^j}, \boldsymbol{x^k}$ of vertices in W that doubly resolve $\boldsymbol{u}$ and $\boldsymbol{v}$, there is at least one such pair that contains $\boldsymbol{x^1}$.

Given two vertices $\boldsymbol{x} = (x_1,\ldots,x_n)$ and $\boldsymbol{y} = (y_1,\ldots,y_n)$ in Q_n, we consider two ways of computing $d(\boldsymbol{x},\boldsymbol{y})$. The first one is purely algebraic and is based on the fact that if $b \in \{-1,0,1\}$ then $|b| = b^2$. We therefore have :

$$d(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{n} |x_i - y_i| = \sum_{i=1}^{n} (x_i - y_i)^2 = \sum_{i=1}^{n} x_i(1-2y_i) + \sum_{i=1}^{n} y_i. \tag{2}$$

The second way is based on the solution of the following constrained maximization problem, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are given vectors, while the components of $\boldsymbol{q}$ are variables:

$$\max \quad \sum_{i=1}^{n} q_i \tag{3}$$

$$\text{s.t.} \quad q_i \leq x_i + y_i \leq 2 - q_i \qquad \forall i = 1,\ldots,n \tag{4}$$

$$q_i \in \{0,1\} \qquad \forall i = 1,\ldots,n \tag{5}$$

Clearly, $d(\boldsymbol{x},\boldsymbol{y})$ is the optimal value of the above IP. Indeed, constraints (4) impose $q_i = 0$ when $x_i = y_i$, while q_i can take value 0 or 1 when $x_i \neq y_i$. By maximizing $\sum_{i=1}^{n} q_i$, we therefore count the number of indices i such that $x_i \neq y_i$, which is exactly the distance between $\boldsymbol{x}$ and $\boldsymbol{y}$.

3 An IP model

Given a subset $W = \{\boldsymbol{x}^{\boldsymbol{1}}, \ldots, \boldsymbol{x}^{|\boldsymbol{W}|}\}$ of vertices of Q_n, we are interested in determining if W resolves Q_n. This can be done by solving the following constrained maximization problem:

$$\max \quad d(\boldsymbol{u}, \boldsymbol{v}) \tag{6}$$

$$\text{s.t.} \quad d(\boldsymbol{x}^{\boldsymbol{j}}, \boldsymbol{u}) = d(\boldsymbol{x}^{\boldsymbol{j}}, \boldsymbol{v}) \qquad \forall j = 1, \ldots, |W| \tag{7}$$

$$\boldsymbol{u}, \boldsymbol{v} \in \{0, 1\}^n \tag{8}$$

If W resolves Q_n, then $\boldsymbol{u}$ must be equal to $\boldsymbol{v}$, which implies $d(\boldsymbol{u}, \boldsymbol{v}) = 0$. Otherwise, constraints (7) are satisfied by at least two distinct vertices $\boldsymbol{u}$ and $\boldsymbol{v}$, which means that $d(\boldsymbol{u}, \boldsymbol{v}) > 0$. Hence the optimal value is strictly positive if and only if W does not resolve Q_n. Using equations (1), we can rewrite (7) as

$$\sum_{i=1}^{n} u_i(1 - 2x_i^j) + \sum_{i=1}^{n} x_i^j = \sum_{i=1}^{n} v_i(1 - 2x_i^j) + \sum_{i=1}^{n} x_i^j \qquad \forall j = 1, \ldots, |W|$$

$$\Leftrightarrow \quad \sum_{i=1}^{n} (1 - 2x_i^j)(u_i - v_i) = 0 \qquad \forall j = 1, \ldots, |W|$$

Also, $d(\boldsymbol{u}, \boldsymbol{v})$ can be determined with the model (3)-(5), which means that one can determine if $W = \{\boldsymbol{x}^{\boldsymbol{1}}, \ldots, \boldsymbol{x}^{|\boldsymbol{W}|}\}$ resolves Q_n by solving the following integer program:

$$\max \quad \sum_{i=1}^{n} q_i \tag{9}$$

$$\text{s.t.} \quad q_i \leq u_i + v_i \leq 2 - q_i \qquad \forall i = 1, \ldots, n \tag{10}$$

$$\sum_{i=1}^{n} (1 - 2x_i^j)(u_i - v_i) = 0 \qquad \forall j = 1, \ldots, |W| \tag{11}$$

$$q_i, u_i, v_i \in \{0, 1\} \qquad \forall i = 1, \ldots, n \tag{12}$$

The problem of determining if a set $W = \{\boldsymbol{x}^{\boldsymbol{1}}, \ldots, \boldsymbol{x}^{|\boldsymbol{W}|}\}$ of vertices doubly resolves Q_n is similar. It follows from Property 2.3 that the optimal value of the following constrained maximization problem is strictly positive if and only if W does not doubly resolve Q_n:

$$\max \quad d(\boldsymbol{u}, \boldsymbol{v}) \tag{13}$$

$$\text{s.t.} \quad d(\boldsymbol{u}, \boldsymbol{x}^{\boldsymbol{1}}) - d(\boldsymbol{u}, \boldsymbol{x}^{\boldsymbol{j}}) = d(\boldsymbol{v}, \boldsymbol{x}^{\boldsymbol{1}}) - d(\boldsymbol{u}, \boldsymbol{x}^{\boldsymbol{j}}) \qquad \forall j = 2, \ldots, |W| \tag{14}$$

$$\boldsymbol{u}, \boldsymbol{v} \in \{0, 1\}^n \tag{15}$$

Using equations (1), we can rewrite (14) as

$$\sum_{i=1}^{n} 2(x_i^j - x_i^1)(u_i - v_i) = 0 \qquad \forall j = 2, \ldots, |W| \tag{16}$$

Hence, one can determine if $W = \{\boldsymbol{x}^{\boldsymbol{1}}, \ldots, \boldsymbol{x}^{|\boldsymbol{W}|}\}$ doubly resolves Q_n by solving the integer program with objective (9) and constraints (10), (16) and (12).

4 An IP-based swapping algorithm

In order to determine small resolving sets for Q_n, we show in this section that it is possible to embed the integer program of the previous section in a swapping algorithm. More precisely, assume we know a resolving set V for Q_{n-1}. As shown in Section 2, it is easy to construct a resolving set of size $|V|+1$ for Q_n. We try to determine a resolving set W for Q_n of size $|W|=|V|$ by choosing an initial set W of $|V|$ vertices in Q_n, and by repeatedly replacing a vertex $\boldsymbol{x} \in W$ with a vertex $\boldsymbol{y} \notin W$ until W resolves Q_n, or a stopping criterion is met. In order to guide the search, vertex $\boldsymbol{y}$ is chosen so that it resolves as few pairs of vertices in W as possible. Vertex $\boldsymbol{y}$ is determined by solving the following constrained minimization problem:

$$\min \quad \sum_{j=1}^{|W|-1} \sum_{k=j+1}^{|W|} p_{jk} \tag{17}$$

$$\text{s.t.} \quad 1 \le d(\boldsymbol{x^j}, \boldsymbol{y}) \le n-1 \qquad \forall j = 1, \ldots, |W| \tag{18}$$

$$-np_{jk} \le d(\boldsymbol{x^j}, \boldsymbol{y}) - d(\boldsymbol{x^k}, \boldsymbol{y}) \le np_{jk} \qquad \forall 1 \le j < k \le |W| \tag{19}$$

$$p_{jk} \in \{0,1\} \qquad \forall 1 \le j < k \le |W| \tag{20}$$

$$\boldsymbol{y} \in \{0,1\}^n \tag{21}$$

Constraints (18) impose $\boldsymbol{y} \notin W$ and $\bar{\boldsymbol{y}} \notin W$. Constraints (19) and (20) imply $p_{jk} = 1$ if and only if $\boldsymbol{y}$ resolves the pair $\boldsymbol{x^j}, \boldsymbol{x^k}$ of vertices. Using equations (1), we can rewrite (18) and (19) as

$$1 - \sum_{i=1}^{n} x_i^j \le \sum_{i=1}^{n} (1 - 2x_i^j) y_i \le n - 1 - \sum_{i=1}^{n} x_i^j \qquad \forall j = 1, \ldots, |W| \tag{22}$$

$$-np_{jk} \le \sum_{i=1}^{n} (x_i^j - x_i^k)(1 - 2y_i) \le np_{jk} \qquad \forall 1 \le j < k \le |W| \tag{23}$$

One can then combine the integer program of the previous section with the above one to not only detect if $W = \{\boldsymbol{x^1}, \ldots, \boldsymbol{x^{|W|}}\}$ resolves Q_n, but also determine a vertex $\boldsymbol{y}$ to add to W. The resulting maximisation problem reads as follows, and will be called IP_1^r, with minimum value z_1^r:

$$\max \quad z_1^r = \sum_{i=1}^{n} n^2 q_i - \sum_{j=1}^{|W|-1} \sum_{k=j+1}^{|W|} p_{jk} \tag{24}$$

$$\text{s.t.} \quad q_i \le u_i + v_i \le 2 - q_i \qquad \forall i = 1, \ldots, n \tag{10}$$

$$\sum_{i=1}^{n} (1 - 2x_i^j)(u_i - v_i) = 0 \qquad \forall j = 1, \ldots, |W| \tag{11}$$

$$1 - \sum_{i=1}^{n} x_i^j \le \sum_{i=1}^{n} (1 - 2x_i^j) y_i \le n - 1 - \sum_{i=1}^{n} x_i^j \qquad \forall j = 1, \ldots, |W| \tag{22}$$

$$-np_{jk} \le \sum_{i=1}^{n} (x_i^j - x_i^k)(1 - 2y_i) \le np_{jk} \qquad \forall 1 \le j < k \le |W| \tag{23}$$

$$q_i, u_i, v_i, y_i \in \{0,1\} \qquad \forall i = 1, \ldots, n \tag{25}$$

$$p_{jk} \in \{0,1\} \qquad \forall 1 \le j < k \le |W| \tag{20}$$

Since $\beta_1 = 1$ and $\beta_n \leq \beta_{n-1} + 1$, we have $\beta_n \leq n$ for all $n \geq 1$, and resolving sets of size n for Q_n are easy to generate. We will therefore only consider sets W with strictly less than n vertices, which means that the value of objective (17) is always strictly smaller than $n(n-1)/2$. As a consequence, the new objective (24) is strictly positive if and only if at least one variable q_i is strictly positive, which is equivalent to say that W does not resolve Q_n.

Let $\bar{\beta}_n$ and $\bar{\Psi}_n$ denote the best known upper bounds on β_n and Ψ_n, respectively. Let V be a resolving set for Q_{n-1} with $|V| = \bar{\beta}_{n-1}$ vertices. As explained above, we try to determine a resolving set W of size $|W| = |V|$ for Q_n. It follows from Property 2.2 that if such a resolving set exists and $\bar{\beta}_{n-1} = \beta_{n-1}$, then $1 < d(\boldsymbol{x}, \boldsymbol{y}) < n-1$ for all pairs $\boldsymbol{x}, \boldsymbol{y}$ of vertices in W. We can therefore increase the left bound and decrease the right one of equations (22) to obtain:

$$2 - \sum_{i=1}^{n} x_i^j \leq \sum_{i=1}^{n} (1 - 2x_i^j) y_i \leq n - 2 - \sum_{i=1}^{n} x_i^j \qquad \forall j = 1, \ldots, |W| \qquad (22')$$

The resulting integer program, where (22') replaces (22) will be called IP_2^r, with minimum value z_2^r.

When looking for doubly resolving sets, we replace equations (11) by (16). Since every doubly resolving set is a resolving set, we also use equations (22') instead of (22) when trying to generate a doubly resolving set for Q_n with $\bar{\beta}_{n-1}$ vertices. The integer programs obtained by replacing (11) by (16) in IP_1^r and IP_2^r are called IP_1^d and IP_2^d, with minimum values z_1^d and z_2^d, respectively.

The vertex $\boldsymbol{x}$ that is removed from W and replaced by $\boldsymbol{y}$ in the swapping algorithm is chosen at random. The following algorithm determines resolving sets for Q_n with $n = n_{min}, \ldots, n_{max}$, assuming that a resolving set $W_{n_{min}-1}$ is known for $Q_{n_{min}-1}$. For example, for $n_{min} = 2$ we can set $W_1 = \{(0)\}$.

Algorithm that generates resolving sets

Data: A resolving set $W_{n_{min}-1}$ for $Q_{n_{min}-1}$;
Result: Resolving sets W_n for Q_n, $n = n_{min}, \ldots, n_{max}$;

1 **for** $n = n_{min}$ *to* n_{max} **do**
2   Set $W = \{\boldsymbol{x}(0) : \boldsymbol{x} \in W_{n-1}\}$;
3   Choose a vertex $\boldsymbol{x} \in W_{n-1}$ at random, and set $W_n = W \cup \{\boldsymbol{x}(1)\}$;
4   **while** $z_2^r > 0$ *and no stopping criterion is met* **do**
5   Choose a vertex $\boldsymbol{x} \in W$ at random, and replace it with $\boldsymbol{y}$;
6   **end**
7   **if** $z_2^r \leq 0$ **then** Set $W_n = W$;
8 **end**

Instructions 2-3 build a set W_n with $|W_{n-1}| + 1$ vertices by adding 0 as nth component to every vertex in W_{n-1}, and by adding 1 as nth component to one vertex $\boldsymbol{x} \in W_{n-1}$. As observed in Section 2, W_n resolves Q_n. The initial set W, with $|W_{n-1}|$ vertices, to which swaps are performed, contains all but the last vertex of W_n. Swapping (instructions 4-6) occurs until a stopping criterion is met, or W resolves Q_n. Vertex

$\boldsymbol{y}$ that replaces vertex $\boldsymbol{x}$ is determined by solving IP_2^r (i.e., the integer program with equations (22') instead of (22)) because we are trying to determine a resolving set for Q_n of size $|W_{n-1}|$. At the end of the while loop, instruction 7 sets W_n equal to W only if the optimal value z_2^r of the integer program is at most equal to 0, since this indicates that W resolves Q_n.

The algorithm that generates doubly resolving sets is similar to the previous one, except that we don't know how to build such a set having $\bar{\Psi}_{n-1}+1$ vertices. Given a doubly resolving set V for Q_{n-1}, let Δ be a positive integer such that we are confident to be able to generate a doubly resolving set of size $|V|+\Delta$ for Q_n. We first generate a set W with $|V|+\Delta$ vertices (instructions 2-5), and perform swaps until we get a doubly resolving set (instructions 9-11). We then try to find a doubly resolving set with one vertex less. This process is repeated until a doubly resolving set of size $|W_{n-1}|$ for Q_n is found, or a stopping criterion is met. As explained above, swaps are performed by solving IP_2^d if $|W| = \bar{\beta}_{n-1}$, and IP_1^d otherwise. The algorithm reads as follows.

Algorithm that generates doubly resolving sets

Data: A doubly resolving set $W_{n_{min}-1}$ for $Q_{n_{min}-1}$; a positive integer Δ;
Result: Doubly resolving sets W_n for Q_n, $n = n_{min}, \ldots, n_{max}$;
1 **for** $n = n_{min}$ *to* n_{max} **do**
2 | Set $W = \{\boldsymbol{x}(0) : \boldsymbol{x} \in W_{n-1}\}$;
3 | **for** $i = 1$ *to* Δ **do**
4 | | Randomly choose a vertex $\boldsymbol{x}$ of Q_n not in W and add it to W;
5 | **end**
6 | **repeat**
7 | | **if** $|W| = \bar{\beta}_{n-1}$ **then** s=2;
8 | | **else** s=1;
9 | | **while** $z_s^d > 0$ *and no stopping criterion is met* **do**
10 | | | Choose a vertex $\boldsymbol{x} \in W$ at random, and replace it with $\boldsymbol{y}$;
11 | | **end**
12 | | **if** $z_s^d \leq 0$ **then** Set $W_n = W$ and remove the last vertex of W;
13 | **until** $z_s^d > 0$ *or* $|W| = |W_{n-1}| - 1$;
14 **end**

5 Computational experiments

We have run our algorithms on a 3 GHz Intel Xeon X5675 machine with 8 GB of RAM, and all integer programs were solved using CPLEX (v12.2). The stopping criterion in both algorithms was fixed to one million swaps, and we have set $\Delta = 1$ for the generation of doubly resolving sets.

Experiments with a genetic algorithm and with a variable neighborhood search (VNS) are reported in [7] and [9] for the metric dimension problem, with $n = 8, \ldots, 17$. Table 1 compares these previous results with ours. Columns 'best' contain the cardinality of the resolving sets obtained by each algorithm, while columns 't' contain computing times in seconds. For our algorithm, we also report the number of swaps needed to generate the best resolving set. As mentioned in the previous section, the set

W of size $|W_{n-1}|+1$ built with instructions 2-3 resolves Q_n, and is obtained without any swap. Hence, if no resolving set for Q_n of size $|W_{n-1}|$ is found, we report no swap and no computing time. We observe that, while we get the same upper bounds on β_n as in [9], ours are obtained much faster.

	genetic [7]		VNS [9]		our algorithm		
n	best	t	best	t	best	t	swaps
8	6	17	6	1	6	<1	22
9	7	51	7	2	7	-	-
10	7	113	7	18	7	1	25
11	8	258	8	48	8	-	-
12	8	637	8	308	8	5	128
13	9	1378	8	1970	8	155	7954
14	9	2524	9	4841	9	-	-
15	10	5414	9	31262	9	4886	119670
16	11	15321	10	66831	10	-	-
17	11	34162	10	86400	10	895	5870

Table 1: Upper bounds on β_n for hypercubes of dimension $n = 8, \ldots, 17$.

Results for larger hypercubes of dimension $n \leq 22$, obtained with a greedy algorithm, are reported in [11], but without any computing time. Their algorithm failed for $n > 22$ because of memory space problems. Upper bounds for larger hypercubes are however derived from their best values. Table 2 compares these results with those produced by our algorithm for $n = 18, \ldots, 28$. As can be observed, we improve the best known upper bound for β_n by one unit for $n = 23, \ldots, 27$.

	greedy [11]	our algorithm		
n	best	best	t	swaps
18	11	11	-	-
19	11	11	316	917
20	12	12	-	-
21	12	12	5016	9949
22	13	13	-	-
23	14	**13**	5225	4660
24	15	**14**	-	-
25	15	**14**	4099	783
26	16	**15**	-	-
27	16	**15**	75757	2995
28	16	16	-	-

Table 2: Upper bounds on β_n for hypercubes of dimension $n = 18, \ldots, 28$.

The best upper bounds for Ψ_n are obtained with a genetic algorithm [8] and a variable neighborhood search [9]. Both algorithms have considered hypercubes Q_n with

n up to 17. Larger hypercubes could not be solved due to space and time limitations. In Table 3, we reports these results and compare them to ours for $n = 8, \ldots, 21$. As can be seen, while we reach the best known upper bounds on Ψ_n for $n \leq 17$, our algorithm can generate upper bounds for larger dimensions.

	genetic [8]		VNS [9]		our algorithm	
n	best	t	best	t	best	t
8	7	14	7	1	7	<1
9	7	33	7	7	7	1
10	8	78	8	20	8	<1
11	8	196	8	141	8	5
12	9	403	8	896	8	577
13	9	980	9	2019	9	1
14	10	1940	9	13511	9	31745
15	10	4752	10	26505	10	3
16	11	10873	10	86400	10	52677
17	12	24356	11	86400	11	7
18	-	-	-	-	**11**	3055
19	-	-	-	-	**12**	18
20	-	-	-	-	**12**	129080
21	-	-	-	-	**13**	152

Table 3: Upper bounds on Ψ_n for hypercubes of dimension $n = 8, \ldots, 21$.

6 Conclusion

We have shown that it is possible to determine if a given set of vertices (doubly) resolves the n-cube by solving an integer program with $O(n)$ variables and $O(n)$ constraints. By embedding such an integer program in a swapping algorithm, we have been able to improve the best known upper bounds on the metric dimension and on the minimum cardinality of a doubly resolving set in hypercubes having up to 268 million vertices. The swapping algorithm is only an example of the possible use of the integer programs of Section 3. Other more sophisticated techniques would possibly provide better results in shorter times.

Acknowledgments

The author would like to thank Issoufou Abdou Amadou and Gilles Éric Zagré for initiating the idea of solving an integer program to determine if a set of vertices (doubly) resolves the n-cube. Special thanks go to Serge Bisaillon for his invaluable help with the use of CPLEX.

References

[1] Z. Beerliova, F. Eberhard, T. Erlebach, A. Hall, M. Hoffmann, M. Mihalak, and L. Ram. Network discovery and verification. *IEEE Journal on Selected Areas in Communications*, 24(12):2168–2181, 2006.

[2] J. Cáceres, C. Hernando, M. Mora, I. M. Pelayo, M. L. Puertas, C. Seara, and D. R. Wood. On the metric dimension of cartesian products of graphs.

[3] M. Cangalovič. Private communication.

[4] G. Chartrand, L. Eroha, M. Johnson, and O. Oellermann. Resolvability in graphs and the metric dimension of a graph. *Discrete Applied Mathematics*, 105:99–113, 2000.

[5] F. Harary and R. Melter. On the metric dimension of a graph. *Ars Combinatoria*, 2:191–195, 1976.

[6] S. Khuller, B. Raghavachari, and A. Rosenfeld. Landmarks in graphs. *Discrete Applied Mathematics*, 70(3):217–229, 1996.

[7] J. Kratica, V. Kovačevič-Vujčič, and M. Čangalovič. Computing the metric dimension of graphs by genetic algorithms. *Computational Optimization and Applications*, 44:343–361, 2009.

[8] J. Kratica, M. Čangalovič, and V. Kovačevič-Vujčič. Computing minimal doubly resolving sets of graphs. *Computers & Operations Research*, 36:2149–2159, 2009.

[9] N. Mladenovič, J. Kratica, V. Kovačevič-Vujčič, and M. Čangalovič. Variable neighborhood search for metric dimension and minimal doubly resolving set problems. *European Journal of Operational Research*, 220:328–337, 2012.

[10] D. Murdiansyah and Adiwijaya. Computing the metric dimension of hypercube graphs by particle swarm optimization algorithms. In T. Herawan, R. Ghazali, N. Nawi, and M. Deris, editors, *Recent Advances on Soft Computing and Data Mining: Proceedings of the 2nd International Conference on Soft Computing and Data Mining, Bandung, Indonesia*, pages 171–178. Springer International Publishing, 2017.

[11] N. Nikolič, M. Čangalovič, and I. Grujičič. Symmetry properties of resolving sets and metric bases in hypercubes. *Optimization Letters*, 2014. doi:10.1007/s11590-014-0790-2.

[12] P. Slater. Leaves of trees. *Congressus Numerantium*, 14:549–559, 1975.