scientific reports

Check for updates

OPEN

Theoretical modelling of efficient fire safety water networks by certified domination

Joanna Raczek[1,3]✉ & Mateusz Miotk[2,3]

This paper explores a new way of designing water supply networks for fire safety using ideas from graph theory, focusing on a method called certified domination. Ensuring a good water supply is crucial for fire safety in communities, this study looks at the rules and problems in Poland for how much water is needed to fight fires in different areas and how this can be achieved at a lowest possible cost. We present a way to plan water supply networks for fire protection as a graph, where each point (node) is a place that needs water, and the lines (links) show where water can go between these points. The main idea is to find the best places to put pumping stations and wells in the network to save money and still meet all the fire safety requirements. Our approach assumes that it costs more to build a pumping station than a well. We use some examples to show how this method can find cost-effective solutions for water supply networks, while ensuring that they meet fire safety requirements and are not too expensive to build. This approach is a new and efficient way to improve the design of water supply networks for fire safety. Key challenges that are solved in this paper are a linear time algorithm finding an optimal solution for networks without cycles and a BLP (Binary Linear Programming) algorithm solving the problem in arbitrary networks.

Keywords Fire protection, Water supply networks, Graph theory, Certified domination

Providing water to the fire protection water supply network is a crucial aspect of the overall fire protection and life safety strategy of an entire community. To meet the expectations of fire safety when a new building is developed or an existing building is renovated, necessary calculations are performed so that the building is complied with fire safety regulations. Before everything it is important to make sure that the proper amount of water is available to the responding fire department for both suppression of the fire in the building, and protection of any exposed buildings. All water-based fire protection systems need water. Without access to an adequate water supply these systems will not function properly. When determining a water supply one needs to make sure it is reliable, and has sufficient volume and pressure to meet the system demand. Because of these reasons, the fire safety regulations require a minimum amount of water be provided, as well as appropriate water pressure, network construction, and supply diameters for external hydrants or automatic sprinkler systems.

In Poland the provision of fire water supply for external fire extinguishing is required for all settlement units with a population exceeding 100 people, not constituting colony buildings, and within their boundaries: public utility buildings and collective residences, as well as production and storage buildings[1]. The demand of the amount of water may be decreased in rural areas with no buildings with increased fire safety, while, for example, in a production and warehouse building the required amount of water for fire-fighting purposes for external fire extinguishing should be higher.

The fire protection water supply network should be constructed as a circumferential network. It is allowed to build a branched fire-fighting water supply network outside urban areas Moreover, it is allowed to build branches from the peripheral network to supply external hydrants.

In Poland the fire protection network can be a part of the main supply network, and in general they are not separated. If in a settlement unit the water resources intended for the population supplied via a water pipeline do not provide the amounts required for fire protection purposes, at a distance of no more than 250 m from the outermost buildings of the settlement unit or the protected building, at least one of the following supplementary water sources with sufficient parameters is provided:

[1]Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland. [2]Faculty of Mathematics, Physics and Informatics, University of Gdańsk, Gdańsk, Poland. [3]These authors contributed equally: Joanna Raczek and Mateusz Miotk. ✉email: joanna.raczek@pg.edu.pl

- a well;
- a water intake point at a natural or artificial water reservoir;
- fire-fighting water tank meeting the requirements of the Polish Standards[2].

The supplementary water sources are used in places with no water supplied via a water pipeline or when the supplied pipeline water has insufficient parameters (capacity, pressure, efficiency). This solution is applied, if necessary, for example, to hotels or sport facilities in rural areas, usually where the fire-fighting water supply network is branched[1].

The supply system in the water supply network includes the elements such as water intakes, first stage pumping stations (pumping stations from the intake to the treatment station), water treatment plants (if needed), clean treated water tanks, second stage pumping stations (pumping stations from the reservoir to the network) and network expansion tanks[3].

Act on collective water supply and collective sewage disposal provides that collective water supply and collective sewage disposal is the commune's own responsibility, but also municipalities are not obliged to build water supply and sewage facilities directly to every object. Hence, while the local community office should provide fire water supply with the required parameters in the neighbourhood of each newly built (or rebuilt) object, in practice it may take a few years (due to lack of funds) and in such situations investors usually prefer to build wells, water tanks or use water from a nearby lake (if there is any) to fulfill the fire safety regulations, however by their own expense. Our model is meant to help develop planning techniques which can be applied in fulfilling the fire safety regulations at a lowest possible cost at the same time.

In this paper we introduce a theoretical model of a water supply network given in the language of graph theory. The model focuses on placing the water supply issues and hence other, less important parameters are omitted. The we apply the notion of minimum certified domination sets to solve placing water supplies: wells and pumping stations in such a way their number for the entire network is minimized. Other types of domination numbers do not fulfill all the requirements of our model, specifically do not take into account placing two different facilities, so they are not suitable in this model.

Results

This section introduces definitions and properties of certified domination in graphs. Afterwards we study computational results concerning finding a minimum certified dominating sets. In particular, we give some basic results obtained in previous works to help understand the background for our study. Then we present a linear time algorithm finding a minimum certified dominating sets in trees. A tree is a connected graph in which any two node are connected by exactly one path. Next we present a binary integer programming (BLP) formulation that finds a minimum certified dominating sets in arbitrary graphs. Even though the BLP works for any graph, its computational complexity is greater. Performances of both algorithms are compared in next chapters.

Graph theory definitions and network model

Let $G = (V, E)$ be a model of a water supply network where V is a set of nodes, where each node represents a place that needs water, and E is the set of links between nodes. Two places u and v are connected if and only if placing a supply system in u gives water in the required quantity and parameters at v. This may be influenced by the landforms, natural resources (e.g. nature reserves, forests), type of developments in the area and other factors. We assume that if u is connected to v then v is also connected to u, and thus G is an undirected graph (see Fig. 4). It is worth to note that the network is more dense in urban areas, as in this case town Liniewo. In rural areas nodes have lower degree, and nodes of higher degree are more distant from each other.

In what follows we assume, without loss of generality, that the only possible supplementary water source (apart from pumping station) is a well. . Our task is to place in G pumping stations and individual wells in such a way that the total number of water supply devices is as small as possible providing the following conditions are met:

1. **Connection to Pumping Stations** Each node without a pumping station and without a well must be connected to a node with a pumping station. This ensures that every location receives adequate water supply from a nearby pumping station.
2. **Pressure Maintenance for Wells** If a node has a well, every node connected to it should be equipped either with a well or a pumping station. This prevents a drop in water pressure and ensures that the parameters at the node with the well remain within acceptable limits.
3. **Financial Viability of Pumping Stations** The construction of a pumping station is financially viable only in locations adjacent to at least two other locations without wells and without pumping stations. This criterion ensures that pumping stations are built in locations where they provide maximum benefit and cost-effectiveness.

Our aim is to minimize the total number of wells and pumping stations.

Figure 1. Graphs P_1, P_2 and P_3.

Figure 2. Graph P_4.

For example, (see Fig. 1) if the water supply network is just one node, P_1, then it is enough to build there a well. If the network are two adjacent nodes, namely P_2, then the optimal solution is to build wells in both nodes. For P_3 the optimal solution is to build a pumping station in the middle node.

However the most interesting case among small graphs is P_4, see Fig. 2.

Note that placing a pumping station in x_1 or in x_4 is not possible by condition 3. Similarly, placing one pumping station (for example at node x_2) and one well (at x_4) is impossible by condition 2. Hence, the optimal solution for P_4 is to build a well at each node.

For the graph G_1 (see Fig. 3, left), the optimal solution is one pumping station and two wells, while for the graph G_2 (see Fig. 3, right), which has no support vertices, and all nodes have a minimum degree of 2, the optimal solution is to place two pumping stations.

We generally follow the notation and terminology of a book on domination in graphs[4]. For a graph G, the set of nodes is denoted by V and the links set by E. For a vertex $v \in V$, the *open neighbourhood* $N(v)$ of v is the set of all nodes adjacent to v, and $N[v] = N(v) \cup \{v\}$ is the *closed neighbourhood* of v. The *degree* of a node v in G is $d_G(v) = |N(v)|$. The number $\min\{d_G(v) : v \in V\}$ is the minimum degree of G and is denoted by $\delta(G)$. A node of degree 0 is called an *isolated vertex*, while a node of degree one in G is called a *leaf* of G. If v is a leaf, then its only neighbour is called a *support vertex* of v. If a support vertex is adjacent to exactly one leaf, then we call it a *weak support vertex* and a *strong support vertex*, otherwise.

Given a graph G, we say that a subset $D \subseteq V$ is a *dominating set* of G if every node belonging to $V - D$ is adjacent to at least one node in D. The *domination number* of a graph G, denoted by $\gamma(G)$, is the cardinality of a smallest (that is minimum) dominating set of G. A dominating set of G of minimum cardinality is called a γ-set of G.

A subset $D \subseteq V$ is called a *certified dominating set* of G if D is a dominating set of G and every node belonging to D has either zero or at least two neighbours in $V - D$. The cardinality of a smallest (that is minimum) certified dominating set of G is called the *certified domination number* of G and is denoted by $\gamma_{cer}(G)$. A certified dominating set of G of minimum cardinality is called a γ_{cer}-set of G. The certified domination was introduced by Dettlaff et al.[5] and continued in[6] in order to describe some possible relations in social networks.

Let $G = (V, E)$ be a model of a water supply network and let D be a set of all nodes in which a pumping station or a well is placed in such a way all requirements 1.–3. are met. Then D is a dominating set of G, because of condition 1. Moreover, if the total number of pumping stations and wells in G is as low as possible, then D is a minimum certified dominating set by conditions 2. and 3.

On the other hand, let $D \subseteq V$ be a minimum certified dominating set of G. If $v \in D$ and v is adjacent to at least 2 nodes not in D, then in the optimal solution we should place a pumping station in v. If $v \in D$ is not adjacent to any nodes not in D, then we should place a well in v.

Figure 4 shows a case study of placing wells and pumping stations at a lowest possible number while meeting conditions 1.–3. Wells are denoted in blue, while pumping station – in red. Every place, denoted black, has a well or a pumping station, or is adjacent to a place with a pumping station, so condition 1. is met. Additionally, if a place has a well, each neighbour also has a well or a pumping station (condition 2.) Also, every place with a pumping station is adjacent to at least two places with no water source. Since the network is more dense in urban areas, the optimal solution suggests building wells only in the suburbs and rural areas.

It is observed[5] that every support vertex in a graph G is part of every certified dominating set of G. For connected graphs of order at least three, there is an interesting equivalence between the standard domination number and the certified domination number. This equivalence exists when the graph has a γ-set where each vertex has at least two neighbours outside the set. Furthermore, for any graph with a minimum degree of at least two, it is found that there's always a γ-set where every vertex satisfies the neighbourhood condition, leading to equal domination numbers. The results also show that graphs with a unique γ-set naturally have equal domination and certified domination numbers. For connected P_4-free graphs, except K_2, these domination numbers are also equal.

Background and motivation

In graph theory, the concept of a certified domination set has received considerable attention. Many scholarly articles explore different aspects of this subject. The idea of certified domination has been explained using different practical examples[5]. In this paper the researches have calculated exact values for the certified domination number in elementary graph types and set basic upper limits for this number in any type of graph. They also

Figure 3. Graph G_1 (left) and graph G_2 (right). PS denotes a pumping station, while W denotes a well.

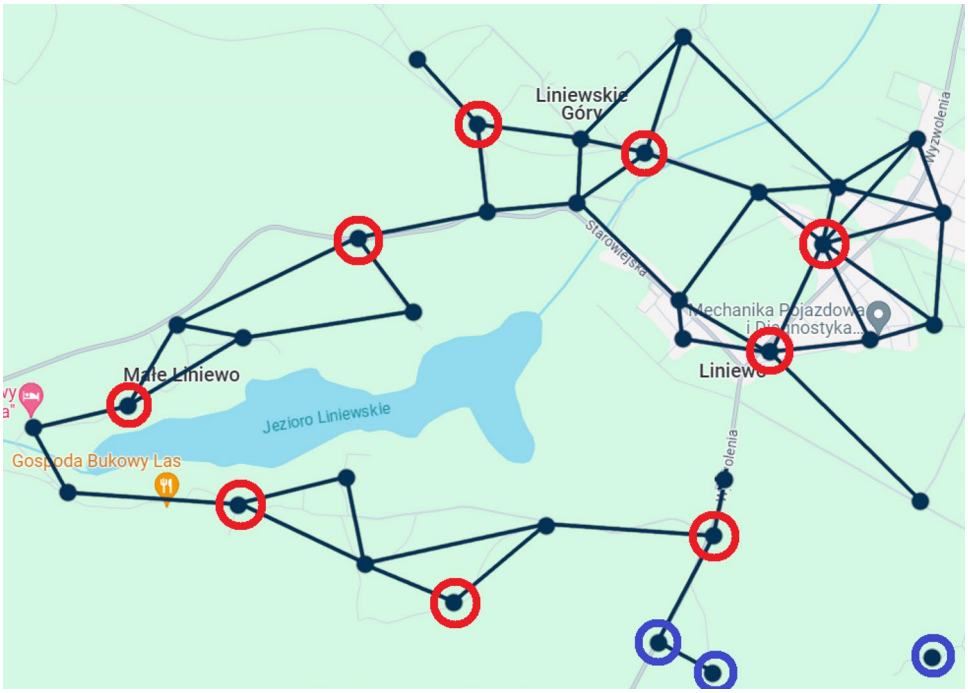

Figure 4. Map with an optimal solution for placing wells (blue circles) and pumping stations (red circles). Source of the map: Google Maps.

noticed that the certified domination number is equal to the well studied ordinary domination number in graph without weak support vertices. On the other hand, the authors in[7] have proven that the question if the certified domination number is equal to the domination number is NP-complete even for graphs with just one weak support vertex. For this reason using known algorithms for finding the domination number does not solve the problem of the minimum certified dominating set. At the same time studying the certified domination number imposes new challenges requiring new techniques, tools and algorithms.

The motivation behind this paper also comes from the complexity of the certified dominating set problem, which is known to be NP-hard even for split graphs, star convex bipartite graphs, comb convex bipartite graphs, and planar graphs[8]. Therefore an important open problem in this area is ascertaining the computational complexity of identifying the certified domination number across various graph classes. Despite so many studies, the challenge of finding the certified domination number for non-elementary graph classes remains unresolved. Our work aims to address this gap by providing a solution to the open problem presented in[6] and[9], specifically targeting the certified domination number for trees (connected acyclic graphs). Despite the relative simplicity of finding dominating sets in trees[10], the task of finding certified dominating sets in trees is significantly more difficult due to the certification requirement. This additional complexity renders the problem both challenging and intriguing for theoretical research and practical use in graph theory. The linear time algorithm presented in this paper finds a minimum certified dominating set in trees, while the ILP algorithm finds a minimum certified dominating set for any graph, however in exponential time. Other studies have compared traditional and certified domination number in graphs [6], and have analyzed certified perfect dominating sets[11]. Research on isolate-free graphs has explored the structural properties of graphs related to certified domination[12]. The concept of certified critical vertices and vertex certified domination critical graphs were also examined[9].

Mathematical models and tools has been used for solving various problems in water distribution networks since many years, see a review article[13]. Depending on the problem to solve, the mathematical methods include for example global gradient method, perturbation or delta expansion, Newton-Raphson methods, linear methods[14], extended linear graph theory[15,16], genetic algorithms[17], and many others.

Linear algorithm for trees

The tree-order is the partial ordering on the vertices of a tree with $u < v$ if and only if the unique path from the root to v passes through u, see a tree in Fig. 5. Let $C(v)$ be the set of all children of a vertex v, and $F(v)$ the father of v (if v is the root, then $F(v) = null$). Denote by $S(T)$ the set of all support nodes and by $SW(T)$ the set of all weak support nodes of T.

The algorithm described in this section proceeds the nodes of T along its tree ordering, from the last to the root. It assumes that the number of nodes of tree, denoted $n(T)$, is greater than 2, and the root is not a leaf. The algorithm consists of three phases, Phase 0, Phase 1 and Phase 2.

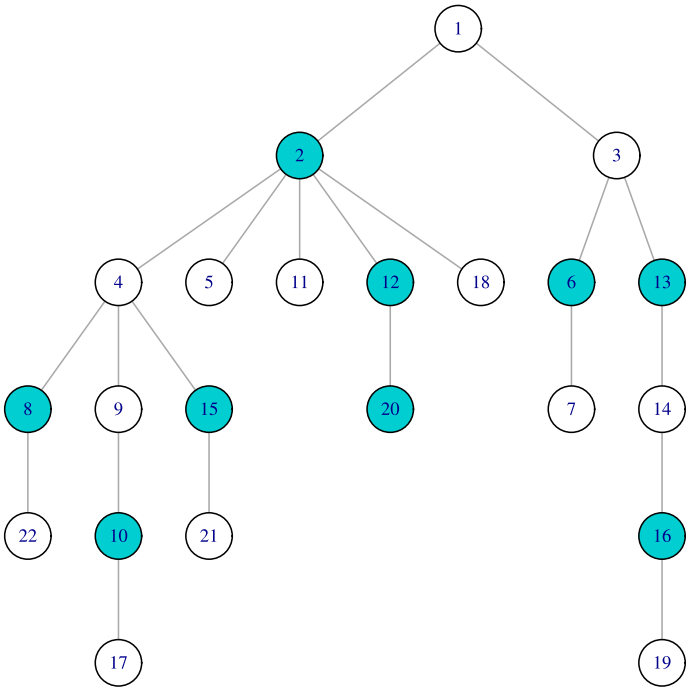

Figure 5. Rooted tree T_1. Blue nodes form the minimum certified dominating set.

Data: A tree T with tree-order
Result: T with identified L, SW and S nodes
Function `Phase0 of MinimuCertifiedDominatingSet(`T`)`**:**
  for $v \in V(T)$ *(from the last to the root)* **do**
    if *v is of degree 1* **then**
      $v.sta \leftarrow L$;
      $F(v).leaf \leftarrow F(v).leaf + 1$;
      $F(v).add \leftarrow v$;
    else
      if $v.leaf > 0$ **then**
        if $v.leaf = 1$ **then** $v.sta \leftarrow SW$ **else** $v.sta \leftarrow S$;
  return T

Algorithm 1. `Phase 0 of MinimumCertifiedDominatingSet`

In Phase 0 of MinimumCertifiedDominatingSet (Algorithm 1), or MCDS for short, determines in T: leaves (assigns to such nodes status L), weak support vertices (assigns to such nodes status SW) and strong support vertices (status S). For this reason, each node stores local variables, *sta*, *leaf* and *add* and their values are updated while performing the algorithm. Before the algorithm starts, we assume each variable of every node is equal to 0. To determine support nodes, the algorithm analyzes values of the local variables *leaf*. Moreover, every weak support should remember his unique leaf (variable *add*, it is used further by SW nodes only). The Phase 0 Algorithm works in linear time.

```
Data: A tree T with tree-order and identified L, SW and S nodes
Result: T with determined statuses
1  Function Phase1 of MinimuCertifiedDominatingSet(T):
2      for v ∈ V(T) − L(T) (from the last to the root) do
3          if v.leaf > 0 then
4              if v.n0 + v.n1 + v.n22 = 0 and v.leaf = 1 then F(v).sw ← F(v).sw + 1;
5              else v.sta ← S;
6              F(v).n2 ← F(v).n2 + 1;
7          else
8              if v.n0 = 0 then
9                  if v.n2 > 0 then
10                     v.sta ← 1;
11                     F(v).n1 ← F(v).n1 + 1;
12                 else
13                     v.sta ← 0;
14                     F(v).n0 ← F(v).n0 + 1;
15             else
16                 if v.n0 = 1 and v.n1 + v.sw = 0 then
17                     v.sta ← 22;
18                     F(v).n2 ← F(v).n2 + 1;
19                     F(v).n22 ← F(v).n22 + 1;
20                 else
21                     if F(v).sta = SW then f = 1 else f = 0;
22                     if v.n0 > v.sw + f then
23                         v.sta ← 2;
24                         F(v).n2 ← F(v).n2 + 1;
25                     else
26                         v.sta ← 11;
27                         F(v).n1 ← F(v).n1 + 1;
28     if root.n2 = 0 then root.sta ← 2;
29     return T
```

Algorithm 2. `Phase 1 of MinimumCertifiedDominatingSet`

At this point we assume that all additional local variables (namely $n_0, n_1, n_2, n_{22}, sw$) are equal 0 at the beginning of the performance of the Algorithm 2 Phase 1, while *sta*, *leaf*, *add* have values as determined in Algorithm 1 Phase 0. At Phase 1 *sta* may also be equal 0, 1, 11, 2 or 22. In general, when v is dominated by its child, $v.sta = 1$, while $v.sta \in \{2, 22\}$ means that v should be a member of a minimum certified dominating set of T, together with support vertices (statuses SW or S). Statuses 11 and 22 have special meaning, explained later. In some cases there is no need to change the status of a node from 0 to 1, so its status remains unchanged. It means that at the end of the performance of the algorithm nodes with $sta = 0$ or $sta = 1$ are certified dominated by the nodes with $sta \in \{2, 22, S, SW\}$. After performing the main loop (lines 2–27) on node v, the local variable n_0 shows how many children of v have status 0, n_1 is equal to the number of children of v with $sta \in \{1, 11\}$, n_2 is equal to the number of children of v with $sta \in \{2, 22, S, SW\}$ and sw is equal to the number of children of v with status sw.

Let $L(T)$ be the set of all nodes of status L, that is leaves. Algorithm 2 Phase 1 proceeds only non-leaf nodes (line 2). The statements differ depending whether node v is a support node (then lines 4–6 are performed) or not (lines 8–27 are performed). Line 28 is performed additionally for the root node only.

If v is a support vertex, then the condition (line 4) checks whether v has exactly one leaf child and no other child with status 0 or status 1 or status 22. If that is the case, the father of v gets to know that there is one more node with these properties. If v is a weak support vertex but has at least one non-leaf child with status 0 or status 1 or status 22, then there is no risk at the end of the Phase 2 that v has exactly one neighbour not in the minimum certified dominating set of T. Hence, v may be treated as a strong support vertex and therefore its status is changed to S. In line 6 we increase the local variable n_2 of the father of v by one.

Lines 8–27 are performed for nodes, which are neither leaves nor support vertices. If v does not have children that should be dominated by v (with status 0) and has a child with $sta \in \{2, 22, S, SW\}$, then it simply gets status 1 (lines 8–11). If v neither has children with status 0 nor status 2, then it gets status 0.

Lines 16–27 are performed for nodes with at least one child with status 0. Statuses 11 and 22 indicate that additional actions may be taken on those nodes and its children or parents during Phase 2.

Since there is only one `for` loop, the algorithm `MinimumCertifiedDominatingSet` in Phase 1 works in linear time.

Data: A tree T with determined statuses
Result: T with determined minimum certified dominating set
1 **Function** `Phase2` (T) **:**
2 **for** $v \in V(T) - L(T)$ *(from last to the root)* **do**
3 **if** v *is not the root* **then**
4 **if** $v.sta = SW$ *and* $F(v).sta \in \{2, SW, S\}$ **then** $(v.add).sta \leftarrow 2$;
5 **if** $v.sta = 0$ *and* $F(v).sta = 11$ **then** $v.sta \leftarrow 2$;
6 **if** $v.sta = 0$ *and* $F(v).sta = 22$ *and* $F(F(v)).sta \in \{2, 22, SW, S\}$ **then**
7 $v.sta \leftarrow 2$;
8 $F(v).sta \leftarrow 1$;
9 **if** $F(F(v)).sta = SW$ **then** $F(F(v)).sta = S$;
10 **if** $root.sta = SW$ *and* $root.n_0 + root.n_1 = 0$ **then** $(root.add).sta \leftarrow 2$;
11 $D \leftarrow \{v \in V(G) : v.sta \in \{2, 22, S, SW\}\}$;
12 **return** D

Algorithm 3. `Phase 2 of MinimumCertifiedDominatingSet`

The algorithm `MinimumCertifiedDominatingSet` in Phase 2 deals with possible situations where a node may dominate exactly one node outside the minimum certified dominating set D. This may happen when v is a weak support vertex and its father is supposed to belong to D (lines 4–5). Then the child of v is added to D. For similar reasons line 6 is performed for particular situation, as well as lines 7–10.

At the end, the minimum certified dominating set D consisting of nodes with status 2, 22, S or SW is returned.

Since there is only one `for` loop, the algorithm `MinimumCertifiedDominatingSet` in Phase 2 works in linear time and therefore, the complete algorithm works in linear lime.

Figure 5 shows results of performing the `MCDS` algorithm on trees. Nodes in blue (set D) show places to build a pumping station or a well. Specifically, nodes 2, 6, 8, 10, 13, 15 and 16 should be placed pumping stations, while nodes 12 and 20 have no neighbours in $V - D$, so in those places wells should be built.

Correctness proof of the linear algorithm

Denote by T_v the tree induced by $v \in V(T)$ and all its descendants. For example, T_9 for the tree T in Fig. 5 is the path (9, 10, 17).

The height of a tree (also known as depth), denoted $h(T)$, is the maximum distance between the root node of the tree and the leaf node of the tree.

Theorem 1 *The* `MinimumCertifiedDominatingSet (MCDS)` *performed on a rooted tree* T, $n(T) > 2$, *where the root is not a leaf, finds a minimum certified dominating set of* T.

Proof If $h(T) = 1$, then T is a star with at least two leaves (because the root is not a leaf). In Phase 0 the root gets status S and every other node status L. In Phase 1 only one thing happens: the root changes status to 2. In Phase 2 the set D consists of 1 element, namely the central node of the star, which is the minimum certified dominating set of this tree.

In what follows we proceed by the induction on the number of nodes in a tree T. Assume that for every tree T' with $n(T') < n(T)$ the `MCDS` algorithm finds a minimum certified dominating set of T' and assume $h(T) > 1$.

Let $v_0, v_1, \ldots, r$ be the longest path from a leaf to the root r. Then v_0 is a leaf and v_1 is a support vertex. If additionally v_2 is a support vertex, then at the end of the algorithm `MinimumCertifiedDominatingSet (MCDS)` performed on T $v_1, v_2 \in D$ and, if v_1 is a weak support vertex, then $v_0 \in D$. Define $T' = T - T_{v_1}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$ if v_1 is a strong support vertex or $\gamma_{cer}(T) = \gamma_{cer}(T') + 2$ if v_1 is a weak support vertex. By the hypothesis, the algorithm `MCDS` finds the minimum certified dominating set of T'. Hence `MCDS` also finds the minimum certified dominating set of T.

Assume now that v_2 is not a support vertex and v_2 has at least two children that are support vertices. Then at the end of the algorithm `MCDS` performed on T, $v_2 \notin D$ (in Phase 1 lines 8– 11 gets status 1), every child of v_2 belongs to D and each such child has at least two neighbours outside D. Define $T' = T - T_{v_1}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$, regardless whether v_1 is a strong support vertex or a weak support vertex. The rest of the proof follows as in previous case.

In what follows we assume that v_2 has exactly one child, namely v_1. Then, since the root is not a leaf, $h(T) > 2$. Moreover, if x_0 is a leaf such that the distance from r to x_0 is the same as the distance from r to v_0, then $(x_0, x_1, x_2, \ldots, r)$ is a path from x_0 to r, then by applying a similar reasoning as in previous paragraph, we obtain that $deg(x_1) = deg(x_2) = 2$, so this we assume for the remaining part of the proof.

If v_3 is a support vertex, then at the end of the algorithm `MCDS` performed on T, $v_0, v_2 \notin D$ and $v_1, v_3 \in D$ and v_3 has at least two neighbours outside D (at least one leaf and v_2). Define $T' = T - T_{v_1}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$. The rest of the proof follows as in previous cases.

Now assume that v_3 is not a support vertex, but has at least one child which is a support vertex. Denote such a child by z. Then by our assumptions, every child of z is a leaf. In this situation at the end of performing the algorithm `MCDS` on T, $v_0, v_2, v_3 \notin D$ and $v_1, z \in D$ and z has at least two neighbours outside D (at least one leaf and v_3). Define $T' = T - T_{v_2}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$. If v_3 is not a leaf in T' then the rest of the proof follows

as in previous cases. If v_3 is a leaf in T', then v_3 is a root, T' is a star. It is easy to verify that in this situation the algorithm returns D consisting of the two support vertices of P_6, namely $D = \{v_1, z\}$.

Assume that every child of v_3 is a father of exactly one support vertex. Then by applying the algorithm to T, every leaf in T_{v_3} gets status 0, every support vertex gets status SW and every child of v_3 gets status 1. If $v_3 = r$, then at the end of Phase 1 v_3 gets status 2 and it is easy to see that D is a minimum certified dominating set of T. If $v_3 \neq r$, then by applying the algorithm to T, v_3 gets status 0 in Phase 1. Depending on the statuses of other children of v_4, if v_4 gets status 11 in Phase 2, or if v_4 gets status 22 and $sta(v_5) \in \{2, 22, SW, S\}$, then at the end of the algorithm $v_3 \in D$. In both cases $v_4 \notin D$ (see lines 6–10 of Phase 3). Define $T' = T - T_{v_1}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$ and the rest of the proof is similar to the previous cases. Therefore assume $v_3 \notin D$. Define $T' = T - T_{v_2}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$ and the rest of the proof follows as in previous cases regardless whether v_3 is a leaf in T' or not.

In what follows we assume that v_3 has exactly one child, namely v_2. Then, since the root is not a leaf, $h(T) > 3$. Assume also that if $(x_0, x_1, \ldots, r)$ is another path from a leaf x_0 to r of length $h(T)$, then $deg(x_1) = deg(x_2) = deg(x_3) = 2$.

If v_4 is a support vertex, then at the end of the algorithm `MCDS` performed on T, $v_0, v_2, v_3 \notin D$ and $v_1, v_4 \in D$ and v_4 has at least two neighbours outside D (at least one leaf and v_3). Define $T' = T - T_{v_2}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$. The rest of the proof follows as in previous cases.

Now assume that v_4 is not a support vertex, but has at least one child which is a weak support vertex. Denote such children by $z_1, z_2, \ldots, z_k$. Then during Phase 1 lines 21–27 are performed for v_4. If v_4 has more children of status 0 (like v_3) than nodes with status SW in its neighbourhood (which in our case means that v_4 has at least two children of status 0), then v_4 gets status 2 and at the end of performing the algorithm `MCDS` on T, $v_4, z_1, \ldots, z_k \in D$ together with the leaf children of $z_1, \ldots, z_k$ (the children are added to D during Phase 2 line 4). Define $T' = T - T_{v_2}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$ and v_4 has at least two neighbours outside D. Now the rest of the proof follows as in previous cases.

If v_4 does not have more children of status 0 than nodes with status SW in its neighbourhood, then v_4 gets status 11 (lines 25–27) and during Phase 2 v_3 together with every child of v_4 of status 0 get status 2. Hence $v_1, v_3 \in D$ and $v_2, v_4 \notin D$ and each $z_1, \ldots, z_k$ has at least two neighbours outside D (one leaf and v_4). Define $T' = T - T_{v_1}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$, since v_3 has at least two neighbours outside D: v_2 and v_4. Now the rest of the proof follows as in previous cases (note that T' has one more child with status SW compared to T, so during performing the algorithm on T' node v_4 also gets status 11).

Now assume every child of v_4, except for v_3 is a strong support vertex. Then in Phase 1 of the algorithm applied for T, v_4 gets status 22 (lines 16–19). Then, if $v_4 \neq r$ and $sta(v_5) \in \{2, 22, SW, S\}$, v_3 gets status 2 during Phase 2 and define $T' = T - T_{v_1}$. If not, v_3 gets status 0 and define $T' = T - T_{v_2}$. In both cases $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$ and the rest of the proof follows as in previous cases. In what follows assume that v_4 is not a support vertex nor has a child which is a weak support vertex nor each child of v_4 is a strong support vertex. Hence, if $deg(v_4) > 2$, then v_4 is a father of a strong support vertex (but not exclusively) or is a grandfather of a support vertex or is a father of at least two children which are grandfathers of support vertices (like v_3). In this case during performing Phase 1 on node v_4 the condition in line 16 is false and condition in line 22 is true, so v_4 gets status 2. Then $v_4 \in D$ and v_4 has at least two neighbours outside D. Define $T' = T - T_{v_2}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$, since v_4 is a support vertex in T'. Now the rest of the proof follows as in previous cases.

In what follows we assume that v_4 has exactly one child, namely v_3. Then, since the root is not a leaf, $h(T) > 4$. Then during performing the Phase 2 on v_4, the condition in line 16 of Phase 1 is true, so v_4 gets status 22. Depending on the status of v_5 either v_3 gets status 2 at the end of Phase 2 or not. If the father of v_4 is of status in $\{2, 22, SW, S\}$ (Phase 2 lines 6–9), then $v_1, v_3 \in D$ and $v_0, v_2, v_4 \notin D$. Define $T' = T - T_{v_1}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$, since v_3 is a support vertex in T'. Now the rest of the proof follows as in previous cases. If the father of v_4 is not of status in $\{2, 22, SW, S\}$, then $v_1, v_4 \in D$ and $v_0, v_2, v_3 \notin D$. Define $T' = T - T_{v_2}$. Then $\gamma_{cer}(T) = \gamma_{cer}(T') + 1$, since v_4 is a support vertex in T'. Now the rest of the proof follows as in previous cases.

This completes the proof. □

Integer linear programming formulation for minimum certified dominating sets

In this section, we present an integer linear programming (ILP) formulation aimed at solving the problem of identifying a minimum certified dominating set D_{cer} in a given graph $G = (V, E)$, where $V = \{i : i = 1, 2, \ldots, n\}$ and $|V| \geq 2$. Linear programming is a mathematical optimization technique that involves the minimization or maximization of a linear objective function subject to a set of linear equality or inequality constraints. The objective function in our formulation seeks to minimize the cardinality of a certified dominating set D, while the constraints encapsulate the specific conditions that define a certified dominating set. We provide a theorem followed by a proof of correctness to validate the efficacy of this BLP formulation. It should be noted there is an alternative solution approach discussed in[8]. However, must be emphasized that the formula presented in that article is incorrect, demonstrated by the counterexample of a path P_4: an optimal feasible solution for this graph is not a certified dominating set. This underlines the importance of the careful proof of correctness presented in this paper. This ILP-based solution provides a mathematical *and computationally efficient* way to solve this complex, combinatorial problem.

We define the following sets of binary decision variables:

$x_i = 1$ if and only if $i \in D$,

a_i is an auxiliary variable.

The above decision variables determine the vertex set $D = \{i \in V : x_i = 1\}$ of cardinality $|D| = \sum_{i \in V} x_i$.

The linear program to find a minimum certified dominating set D_{cer} in a graph $G = (V, E)$ is formulated as follows:

$$\text{Minimize} \quad Z = \sum_{i=1}^{n} x_i$$

subject to

$$\forall i \in V, \quad x_i + \sum_{j \in N(i)} x_j \geq 1 \tag{C1}$$

$$\forall i \in V, \quad \sum_{j \in N(i)} (1 - x_j) + |V| \cdot a_i \geq 2x_i \tag{C2}$$

$$\forall i \in V, \quad \sum_{j \in N(i)} (1 - x_j) + |V|(a_i - 1) + 2 \leq 2x_i \tag{C3}$$

$$\forall i \in V, \quad x_i \in \{0, 1\},\ a_i \in \{0, 1\} \tag{C4}$$

We present a theorem to establish the validity and correctness of the linear program formulated to find the minimum certified dominating set in a given graph $G = (V_G, E_G)$. The theorem aims to rigorously prove that the linear program not only satisfies to the constraints defining a certified dominating set but also optimally minimizes its cardinality.

Let Ω denote the domain determined by all feasible solutions to Z, i.e.

$$\Omega = \{(x, a) : x \in \{0, 1\}^n,\ a \in \{0, 1\}^n \text{ satisfy (C1), (C2) and (C3)}\}.$$

Proposition 2 *Any feasible solution $(x, a) \in \Omega$ determines a certified dominating set.*

Proof Let $(x, a) \in \Omega$. Consider $D = \{i \in V : x_i = 1\}$.

- Since (x, a) is feasible, constraint (C1) ensures that every vertex or one of its neighbours is in D, thus satisfying the dominating set condition.
- Constraints (C2) and (C3) work together to satisfy the certified condition: zero neighbours or at least two neighbours in $V - D$ for each vertex in D. Assume first that $x_i = 0$. Then by (C1) $\sum_{j \in N(i)} x_j \geq 1$, and so

$$0 \leq \sum_{j \in N(i)} (1 - x_j) \leq |V| - 2. \tag{1}$$

In this case (C2) simplifies to $\sum_{j \in N(i)} (1 - x_j) + |V| \cdot a_i \geq 0$ which is always true, regardless of a_i. At the same time (C3) simplifies to

$$\sum_{j \in N(i)} (1 - x_j) + |V|(a_i - 1) + 2 \leq 0. \tag{2}$$

By (1) and by substituting $a_i = 0$ in (C3) we obtain $\sum_{j \in N(i)} (1 - x_j) - |V|(a_i - 1) + 2 \leq |V| - 2 - |V| + 2 \leq 0$, so (2) is true. Therefore if (x, a) is a feasible solution and if $x_i = 0$, then substituting $a_i = 0$ the constraints (C2) and (C3) are always true. Assume now $x_i = 1$. Then (C2) simplifies to

$$\sum_{j \in N(i)} (1 - x_j) + |V| \cdot a_i \geq 2, \tag{3}$$

while (C3) simplifies to

$$\sum_{j \in N(i)} (1 - x_j) + |V| \cdot a_i \leq |V|. \tag{4}$$

If $\sum_{j \in N(i)} (1 - x_j) = 0$, then substituting $a_i = 1$ makes (3) and (4) both true. Similarly, if $\sum_{j \in N(i)} (1 - x_j) \geq 2$, then substituting $a_i = 0$ makes (3) and (4) both true. However if $\sum_{j \in N(i)} (1 - x_j) = 1$, then neither $a_i = 1$ nor $a_i = 0$ gives a feasible solution to both (C2) and (C3). Therefore, if (x, a) is a feasible solution and if $x_i = 1$, then (C2) and (C3) are true only when $\sum_{j \in N(i)} (1 - x_j) \neq 1$, which means i does not have exactly one neighbour in $V - D$. Hence if (x, a) is a feasible solution, then D is a certified set.

The conclusion of the above analysis is that any feasible solution $(x, a) \in \Omega$ determines a certified dominating set $D = \{i \in V : x_i = 1\}$. □

Below we see that the reverse of the above result also holds. In particular

Proposition 3 *Any certified dominating set D can be associated with a solution $(x, a) \in \Omega$.*

Proof Let D be a certified dominating set of G. Consider the following solution (x, a):

- $x_i \in \{0,1\}$ and $a_i \in \{0,1\}$ for every $i \in \{1,2,\ldots,n\}$, so (C4) is true.
- $x_i = 1$ if and only if $i \in D$.
- if $x_i = 0$, then $a_i = 0$.
- if $x_i = 1$ and i has no neighbours in $V - D$, then $a_i = 1$.
- if $x_i = 1$ and i has at least two neighbours in $V - D$, then $a_i = 0$.

Let us see that $(x,a) \in \Omega$, because since D is dominating, (C1) is satisfied. Next, if $i \notin D$, then $x_i = a_i = 0$ and $\sum_{j\in N(i)}(1-x_j) \leq |V| - 2$. In this situation it is easy to check that (C2) and (C3) are satisfied. If $x_i = a_i = 1$, then $\sum_{j\in N(i)}(1-x_j) = 0$ and hence both (C2) and (C3) are true. While if $x_i = 1$ and $a_i = 0$, then $\sum_{j\in N(i)}(1-x_j) \geq 2$ and again both (C2) and (C3) satisfied.

Note that if $i \in D$ and if i has exactly one neighbour not in D, then $\sum_{j\in N_G(v_i)}(1-x_j) = 1$ and $x_i = 1$. Therefore (C2) can be rewritten and can be simplified to $|V| \cdot a_i \geq 1$, while (C3) is equivalent to the formula $|V|(a_i - 1) + 1 \leq 0$. Then (C2) implies that $a_i = 1$, but this causes (C3) simplify to $1 \leq 0$, a contradiction. □

Methods

In our recent study, we conducted a comprehensive evaluation of the two algorithms presented in this paper, which we renamed for need of testings and presentation. The algorithm `MinimumCertifiedDominatingSet` is named `linear_Trees` for short, while the algorithm that uses integer linear programming has name `certified_ILP`, the latter of which is based on linear programming techniques. The algorithms were tested on random trees, and their results and timing were compared using a server computer. The obtained results are given in tables and graphs.

Server computer tests

We performed this assessment on a high-performance server equipped with an Intel(R) Xeon E321xx 3.20 GHz processor, 32.0 GB of RAM, and running the Ubuntu Desktop 22.04.3 LTS system. The algorithms were implemented using SageMath version 9.8 due to its advanced computational features and compatibility with complex algorithmic processes.

The experimental setup generated random trees with varying numbers of vertices, specifically 100, 200, 300, and 400, to fully understand the algorithms' behavior across different scales. Each algorithm was applied to the tree structures, allowing for an analysis of their performance metrics and efficiency levels.

The `certified_ILP` algorithm, which focuses on linear programming, used the `MixedIntegerLinearProgram` package and the GLPK solver as its default computational engine. This integration was crucial for ensuring the accuracy and reliability of the algorithm's linear programming components.

Tables 1 and 2 systematically compile the outcomes of rigorous testing and present a detailed breakdown of performance metrics. The tables offer insights into the efficiency and effectiveness of algorithms for different graph sizes. Results for graphs with 100, 200, 300, and 400 nodes are included, providing a nuanced understanding of the scalability and robustness of the `linear_Trees` and `certified_ILP`.

It can be derived that for the tree on 100 nodes the linear time algorithm was over 40 times faster than the integer linear programming algorithm, while for the 200 nodes – nearly 300 times faster.

.

The information obtained from these tables and the graph of timings are anticipated to make a substantial contribution to the wider discussion of computational algorithm research, especially in the application of linear and linear programming techniques to intricate tree structures in graph theory.

Enhancing fire safety in urban planning through graph theory

The research on fire protection systems in this study is significant for several reasons. A new model for fire hydrant placement is introduced using graph theory to minimize distances between them. This is crucial because in urban areas, the distance and availability of fire hydrants can significantly affect the effectiveness of fire department response. We use certified dominating sets to enhance fire safety in urban areas by optimizing the placement of hydrants.

	Tests for 100 nodes		**Tests for 200 nodes**	
	`linear_Trees` **[ms]**	`certified_ILP` **[ms]**	`linear_Trees` **[ms]**	`certified_ILP` **[ms]**
Min	0.59	26.10	1.19	337.87
25%	0.76	33.92	1.55	439.22
Mean	1.17	52.19	2.38	675.73
Median	0.88	39.14	1.79	506.80
75%	1.23	54.80	2.50	709.52
Max	1.76	78.29	3.57	1013.60

Table 1. Basic time statistics for trees on 100 nodes and 200 nodes for a server computer.

	Tests for 300 nodes		Tests for 400 nodes	
	`linear_Trees` [ms]	`certified_ILP` [ms]	`linear_Trees` [ms]	`certified_ILP` [ms]
Min	1.79	3998.46	2.48	977270.79
25%	2.33	5198.00	3.22	1270452.02
Mean	2.38	7996.91	4.96	1954541.57
Median	2.69	5997.68	3.72	1465906.18
75%	3.76	8396.76	5.21	2052268.65
Max	5.37	11995.37	7.44	2931812.36

Table 2. Basic time statistics for trees on 300 nodes and 400 nodes for a server computer.

Furthermore, the distribution system is analysed through the pump stations and wells network to ensure a continuous water supply in the event of a natural fire. This approach not only enhances the ability to extinguish fires, but also reduces certain types of fire damage. Mathematical models are also employed to determine the most cost-effective and efficient location for building supporting infrastructure. This ensures that fire safety devices that are too expensive, redundant, or useless are not purchased by the administrator.

Using mathematical models to determine the optimal placement of hydrant slots and associated infrastructure can eliminate defunct fire safety equipment, resulting in cost savings. This research is essential as it provides a low-cost method to improve our country's fire safety performance.

In addition, it has a significant impact on urban planning and development through the integration of fire safety considerations into the early stages of urban planning, thus promoting a more comprehensive approach to urban planning. The aim is to ensure that new developments have effective fire safety measures from the outset, thus reducing the need for costly upgrades.

Conclusion

This study presents a comprehensive approach for enhancing the efficacy of fire hydrant systems through the application of graph theory. The impact on urban fire safety, economic efficiency, and city planning is significant and provides a plan for more effective and sustainable fire safety strategies in urban environments. This study contributes to the theoretical understanding of graph theory and its practical applications in fire safety in cities.

In this paper we have solved a problem of constructing a linear time algorithm that finds a minimum certified dominating sets in trees. Also, we have identified and corrected an error that appears in the research literature concerning the ILP model for finding a minimum certified dominating set. We have tested and analyzed both algorithms on a server computer.

Data availability

The datasets generated and analysed during the current study are available in the GitHub repository, https://github.com/mmiotk/CertifiedDominatingSets_Algorithms/tree/main/datasets

Code availability

https://github.com/mmiotk/CertifiedDominatingSets_Algorithms/tree/main.

Received: 2 January 2024; Accepted: 5 September 2024
Published online: 16 September 2024

References

1. Control and Reconnaissance Section of District Headquarters, State Fire Service in Pruszcz Gdański, Poland, personal communication (2023).
2. Journal of Laws 2009.124.1030—Regulation of the Minister of Internal Affairs and Administration of July 24, 2009 on fire-fighting water supply and fire roads.
3. Sanitary engineering technician, materials for classes. Municipal networks, construction and maintenance. https://instsani.pl/technik-inzynierii-sanitarnej/materialy-do-zajec/sieci-komunalne-budowa-i-konserwacja/sieci-wodociagowe-2/.
4. Haynes, T. W., Hedetniemi, S. T. & Henning, M. A. *Fundamentals of Domination, 27–47* (Springer International Publishing, Cham, 2023).
5. Dettlaff, M., Lemańska, M., Topp, J., Ziemann, R. & Żyliński, P. Certified domination. *AKCE Int. J. Graphs Combin.* **17**, 86–97. https://doi.org/10.1016/j.akcej.2018.09.004 (2020).
6. Dettlaff, M. *et al.* Graphs with equal domination and certified domination numbers. *Opus. Math.* **39**, 815–827. https://doi.org/10.7494/OpMath.2019.39.6.815 (2019).
7. Raczek, J. & Miotk, M. Manuscript. *Draft* (2024).
8. Jakkepalli, P. K., Arumugam, S., Khandelwal, H. & P., V. S. R. Algorithmic aspects of certified domination in graphs. *Commun. Combin. Optim.* **7**, 247–255. https://doi.org/10.22049/cco.2021.27302.1226 (2022). http://comb-opt.azaruniv.ac.ir/article_14269_0dabbd07d38631d59ef285a2fe950400.pdf.
9. Miotk, M. Certified domination critical graphs upon vertex removal. submitted.
10. Cockayne, E. J., Goodman, S. E. & Hedetniemi, S. T. A linear algorithm for the domination number of a tree. *Inf. Process. Lett.* **4**, 41–44 (1975).
11. Hamja, J. Certified perfect domination in graphs. *Eur. J. Pure Appl. Math.* **16**, 2763–2774. https://doi.org/10.29020/nybg.ejpam.v16i4.4894 (2023).

12. Lone, A. I. & Goswami, V. S. Connected certified domination edge critical and stable graphs. *Acta Universitatis Sapientiae, Informatica* **15**, 25–37 (2023).
13. Bello, O. et al. Solving management problems in water distribution networks: A survey of approaches and mathematical models. *Water* **11**. https://doi.org/10.3390/w11030562 (2019).
14. Jakobus E. Van Zyl, D. A. S. & Walters, G. A. Extended-period modeling of water pipe networks-a new approach. *J. Hydraulic Res.* **43**, 678–688. https://doi.org/10.1080/00221680509500387 (2005).
15. Gupta, R. & Prasad, T. D. Extended use of linear graph theory for analysis of pipe networks. *J. Hydraul. Eng.* **126**, 56–62. https://doi.org/10.1061/(ASCE)0733-9429(2000)126:1(56) (2000).
16. Luigi Berardi, O. G. & Todini, E. Accounting for uniformly distributed pipe demand in wdn analysis: enhanced gga. *Urban Water J.* **7**, 243–255. https://doi.org/10.1080/1573062X.2010.491550 (2010).
17. Savic, D. A. & Walters, G. A. Genetic algorithms for least-cost design of water distribution networks. *J. Water Resour. Plann. Manag.* **123**, 67–77. https://doi.org/10.1061/(ASCE)0733-9496(1997)123:2(67) (1997).

Author contributions
Conceptualization M.M., visualization J.R. All authors did the investigation, conducted the experiments, analysed the results, prepared the original draft and reviewed the manuscript.

Funding
This research was supported by the Ministry Subsidy for Research for the Gdańsk University of Technology.

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to J.R.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.