UTILITAS MATHEMATICA 123 (2025) 61–86

Utilitas Mathematica
www.combinatorialpress.com/um

Combinatorial Press

Weighted domination models and randomized heuristics

Lukas Dijkstra[1,✉], Andrei Gagarin[1], Vadim Zverovich[2]

[1] *School of Mathematics, Cardiff University, Cardiff, UK*

[2] *Mathematics and Statistics Research Group, University of the West of England, Bristol, UK*

ABSTRACT

We consider the minimum weight and smallest weight minimum-size dominating set problems in vertex-weighted graphs and networks. The latter problem is a two-objective optimization problem, which is different from the classic minimum weight dominating set problem that requires finding a dominating set of the smallest weight in a graph without trying to optimize its cardinality. In other words, the objective of minimizing the size of the dominating set in the two-objective problem can be considered as a constraint, i.e. a particular case of finding Pareto-optimal solutions. First, we show how to reduce the two-objective optimization problem to the minimum weight dominating set problem by using Integer Linear Programming formulations. Then, under different assumptions, the probabilistic method is applied to obtain upper bounds on the minimum weight dominating sets in graphs. The corresponding randomized algorithms for finding small-weight dominating sets in graphs are described as well. Computational experiments are used to illustrate the results for two different types of random graphs.

Keywords: domination in weighted graphs, probabilistic method, upper bounds, heuristics, integer linear programming (ILP)

2020 Mathematics Subject Classification: 05C69 .

✉ Corresponding author.
E-mail address: dijkstral@cardiff.ac.uk (L. Dijkstra).

Received 09 October 2024; Accepted 15 June 2025; Published 26 June 2025.

DOI: 10.61091/um123-05

1. Introduction

1.1. Basic notions, notation, and motivation

We consider undirected simple vertex-weighted graphs

$$G = (V, E, w\colon V \to \mathbb{R}),$$

where $V = V(G) = \{v_1, v_2, ..., v_n\}$ is the set of vertices, $E = E(G) = \{e_1, e_2, ..., e_m\}$ is the set of edges of G, and $w\colon V \to \mathbb{R}$ is a weight/cost function that assigns a certain weight $w_i = w(v_i)$ to each vertex v_i of G, $i = 1, ..., n$. The neighbourhood of a vertex v in G is denoted by $N(v)$, i.e. $N(v) = \{u \mid vu \in E,\ u \neq v\}$. Any vertex in $N(v)$ is a neighbour of v. The closed neighbourhood of v is denoted by $N[v] = N(v) \cup \{v\}$. For a set of vertices $A \subseteq V$,

$$N(A) = \bigcup_{v \in A} N(v) \quad \text{and} \quad N[A] = N(A) \cup A.$$

The degree of a vertex v_i is the number of its neighbours and is denoted by $d_i = d(v_i)$, $i = 1, ..., n$. A sequence of vertex degrees of G is denoted by $\bar{d} = (d_1, d_2, ..., d_n)$. The minimum and maximum vertex degrees of G are denoted by $\delta = \delta(G)$ and $\Delta = \Delta(G)$, respectively.

A subset $X \subseteq V(G)$ is called a *dominating set* of G if every vertex not in X is adjacent to at least one vertex in X. The minimum cardinality of a dominating set of G is called the *domination number* of G and denoted by $\gamma(G)$. We denote by $\gamma_w(G)$ the smallest weight of a dominating set in a vertex-weighted graph G, and by $\gamma^*_w(G)$ the smallest weight of a minimum-cardinality dominating set D in G. Clearly, $\gamma_w(G) \leq \gamma^*_w(G)$. As an illustration, Figure 1 shows a vertex-weighted graph $K_{3,3}$, where the vertices are labelled A, B, C, D, E, F, and the corresponding weights are shown inside of each vertex. This graph has two (minimal by inclusion) dominating sets of cardinality 3, $\{A, B, C\}$ and $\{E, D, F\}$, and nine dominating sets of cardinality 2, $\{A, D\}$, $\{A, E\}$, $\{A, F\}$, $\{B, D\}$, $\{B, E\}$, $\{B, F\}$, $\{C, D\}$, $\{C, E\}$, and $\{C, F\}$. It is easy to see that $\gamma(K_{3,3}) = 2$, $\gamma_w(K_{3,3}) = 3$ (given by $\{A, B, C\}$), and $\gamma^*_w(K_{3,3}) = 4$ (given by $\{C, D\}$, $\{B, D\}$, and $\{A, D\}$).

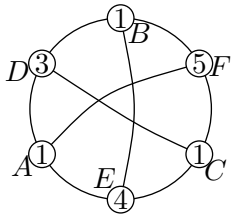

Fig. 1. Vertex-weighted graph $K_{3,3}$

The total weight of a vertex-weighted graph G is

$$w_G = \sum_{v_i \in V} w_i.$$

Also,

$$w_{\max} = \max_{1 \le i \le n} w_i, \quad w_{\min} = \min_{1 \le i \le n} w_i, \quad w_{\text{ave}} = \frac{w_G}{n},$$

so that

$$w_{\max} \ge w_{\text{ave}} \ge w_{\min}.$$

Weighted domination in graphs and networks can be used, for example, for modelling a problem of the placement of a small number of transmitters in a communication network such that every site in the network either has a transmitter or is connected by a direct communication link to a site that has such a transmitter. In addition, there are some 'costs' associated with placing a transmitter in each particular location of the network (i.e. a vertex of the corresponding graph). The minimum weight dominating set problem usually does not place any restrictions on the size of the dominating set, i.e. the number of transmitters in this case – it only requires us to find a smallest weight/cost dominating set in a vertex-weighted graph. However, the total emitted radiation in the environment would be smaller with fewer transmitters installed. Therefore, if the weight associated with each vertex of the graph is considered as a cost function or some nuisance measurement parameter (e.g. a level of noise at the site of an installed transmitter), the corresponding problem becomes a multi-criteria minimization problem, where the objective is to find a minimum cost smallest-cardinality dominating set in a weighted graph.

The weights of vertices in the graph can also indicate a local impact or a level of influence of placing a facility in each particular location; for example, see positive influence dominating sets in social networks [17, 24]. In this case, we have a multi-criteria optimization problem of finding a smallest-cardinality dominating set X in a graph G because of the limited availability of the resource. On the other hand, the set X needs to provide the maximum positive level of impact on the whole network represented by G. In other words, it is necessary to find a smallest-cardinality dominating set X in G such that its total weight $w(X) = \sum_{v_i \in X} w_i$ is maximized.

Assuming that all weights w_i are non-negative and $w_{\max} > 0$, the last problem can be reduced to the corresponding minimization problem as follows. We can scale the weights at vertices of G to the interval $[0, 1]$, for example by dividing each of them by a positive constant $w_{\max}$, so that $\alpha w_i \in [0, 1]$, where $\alpha = 1/w_{\max}$. Then, replace all original weights w_i by the values $\psi_i = 1 - \alpha w_i$. Clearly, $\psi_i \in [0, 1]$ for $i = 1, ..., n$. Now, the problem of finding a minimum-size dominating set X of the largest possible weight $w(X) = \sum_{v_i \in X} w_i$ in G is equivalent to maximizing $\alpha \sum_{v_i \in X} w_i$ on the set of all minimum-size dominating sets $X \subseteq V(G)$. Assuming that $\gamma(G)$ is known, this is equivalent to minimizing the sum $\sum_{v_i \in X} \psi_i$ with an additional constraint on the cardinality of the dominating set, i.e. $|X| = \gamma(G)$, and we minimize

$$\sum_{v_i \in X} \psi_i = |X| - \alpha \sum_{v_i \in X} w_i = \gamma(G) - \alpha \sum_{v_i \in X} w_i,$$

where $\gamma(G)$ and α are constants. Notice that

$$w(X) = w_{\max} \cdot \left(\gamma(G) - \sum_{v_i \in X} \psi_i \right).$$

Therefore, we can focus on the minimization version of the general two-objective optimization problem. Notice that, in the classic single-objective minimum weight dominating set problem, vertices of weight zero and their neighbours can be removed from the graph.

For some other examples of applications of dominating sets, multiple dominating sets in a so-called reachability graph are used in [12, 8] to optimize refuelling facility locations in a road network. Notice that each facility location in a road network can be associated with a certain cost, and this can be modelled by vertex-weighted reachability graphs. An application of dominating sets to model and optimize backbone sets in wireless sensor networks is described in [27].

1.2. Related results and complexity issues

Some recent research progress and methods for solving different types of dominating set problems can be found in [1, 3, 8, 12, 21, 27]. The problem of finding the exact value of $\gamma(G)$ in a graph G is one of the classic NP-hard problems [15]. Moreover, the problem is known to be APX-hard (e.g. see [23]) and not fixed parameter tractable [10]. Hence, the problem of finding a minimum or maximum weight smallest-cardinality dominating set in a graph G is also NP-hard and not fixed parameter tractable in general. Therefore, it is necessary to have efficient and effective heuristic algorithms and methods for finding small-size and light- or heavy-weight dominating sets in graphs. Also, to estimate the quality of a given dominating set, it is important to have good bounds for the domination number $\gamma(G)$ and for the smallest or largest weight $w(D)$ of dominating sets $D \subseteq V(G)$.

The following upper bounds for the domination number $\gamma(G)$, which can be obtained using the probabilistic method, are well-known:

Theorem 1.1 ([2, 20]). *For any graph G,*

$$\gamma(G) \leq \frac{\ln(\delta+1)+1}{\delta+1} n. \tag{1}$$

Theorem 1.2 ([6, 20, 19, 22]). *For any graph G with $\delta \geq 1$,*

$$\gamma(G) \leq \left(1 - \frac{\delta}{(1+\delta)^{1+1/\delta}}\right) n. \tag{2}$$

These upper bounds are known to be asymptotically sharp by using Alon's construction (e.g. see [26]). Also, by applying the probabilistic method, these upper bounds have been generalized for directed graphs [22] and for several other domination parameters in simple unweighted graphs [13, 14, 27].

Chen et al. [7] showed that the single-objective minimum weight dominating set problem is APX-hard and provided a simple greedy heuristic achieving an $O(\log n)$ approximation ratio, which is asymptotically the best possible in that case. The authors of [7] also gave a randomized rounding heuristic for a linear relaxation of an Integer Linear Programming (ILP) formulation of the problem. The authors in [17] used a randomized rounding heuristic for a linear relaxation of an ILP formulation for a variation of this problem. Notice that the minimum size dominating set problem can be considered as a

particular case of the minimum weight dominating set problem by assigning unit weights to all vertices of the graph. It is easy to see that the (two-objective) smallest or largest weight minimum-size dominating set problem in vertex-weighted graphs is APX-hard as a generalization of the minimum-size dominating set problem in simple graphs.

Also, notice that the domination number $\gamma(G)$ of a graph G exhibits the asymptotic property of having two points of concentration in random graphs [25] (and digraphs [22]). However, the asymptotic property of a "typical" graph stated in this kind of theorems cannot be used as a bound for the domination number $\gamma(G)$ of a given graph G, it does not help to determine the domination number exactly, and it does not provide any ideas how to find the corresponding dominating sets of size $\gamma(G)$ in G.

In this paper, we focus on the (two-objective) smallest or largest weight minimum-size dominating set problem and direct applications of the probabilistic method to tackle the related (single-objective) minimum weight dominating set problem in vertex-weighted graphs. This allows us to analyze deterministic and several heuristic solution methods for these two problems by considering corresponding computational results. First, in Section 2, we consider an ILP formulation of the two-objective optimization problem and show its connection and reduction to the single-objective minimum weight dominating set problem. Then, probabilistic constructions, the corresponding randomized algorithms, and upper bounds are described for the single-objective minimum weight dominating set problem in Section 3. The new upper bounds presented in Section 3 can be considered as a generalization of the classic upper bounds (2) and (1). To illustrate the concepts and better analyze the results, some computational experiments with random graphs are described in Section 4. Section 5 provides a summary of our findings and conclusions.

2. ILP formulation and reduction to the minimum weight problem

For simplicity, we focus on the minimization version of the problem with all positive vertex weights $w_i > 0$ to find a minimum-size dominating set X in a vertex-weighted graph G such that $w(X)$ is the smallest possible. In this section, we describe how to formulate the two-objective minimum weight smallest-cardinality dominating set problem as an ILP problem to solve it deterministically. We also show connections with the single-objective minimum weight dominating set problem and provide a reduction to the latter problem.

Given a graph G, the problems of finding the exact values of $\gamma_w(G)$ and $\gamma_w^*(G)$ and corresponding optimal dominating sets $X \subseteq V(G)$ can be formulated as ILP problems as follows. A binary decision variable $x_i \in \{0, 1\}$ is associated with each vertex $v_i \in G$ to indicate whether the vertex is in the solution set X or not, i.e. $x_i = 1$ if and only if v_i is in the optimal dominating set X of G, otherwise $x_i = 0$, $i = 1, ..., n$. Then the ILP formulation for finding $\gamma_w(G)$ and a corresponding optimal dominating set is

$$\begin{cases} \text{minimize} & z(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} w_i x_i \\ \text{subject to:} & \sum_{v_i \in N[v_j]} x_i \geq 1, & j = 1, \ldots, n, \\ & x_i \in \{0, 1\}, & i = 1, \ldots, n. \end{cases} \tag{3}$$

To find $\gamma_w^*(G)$ and a corresponding optimal dominating set, we can use weights $w_i + w_G$ at the vertices of G instead of w_i, $i = 1, 2, \ldots, n$, and minimize the following objective function:

$$z(X) = z(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} (w_i + w_G) x_i = w(X) + |X| w_G. \tag{4}$$

Notice that, if X' is any dominating set such that $|X'| < |X|$, then $z(X') < z(X)$ in (4) because $w(X') \leq w_G$, and w_G is a constant. Thus, minimizing (4) with the constraints of (3) gives a dominating set of minimum cardinality and, moreover, of minimum weight among such sets. Furthermore, we can normalize the weights $w_i + w_G$ in (4) by dividing them by w_G, $i = 1, 2, \ldots, n$. This gives the following ILP formulation to find $\gamma_w^*(G)$ and a corresponding optimal dominating set:

$$\begin{cases} \text{minimize} & z(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} \frac{w_i}{w_G} x_i \\ \text{subject to:} & \sum_{v_i \in N[v_j]} x_i \geq 1, & j = 1, \ldots, n, \\ & x_i \in \{0, 1\}, & i = 1, \ldots, n. \end{cases} \tag{5}$$

Assuming G is non-trivial and non-empty, we have $0 < \frac{w_i}{w_G} < 1$, $i = 1, ..., n$, and $0 < \sum_{i=1}^{n} \frac{w_i}{w_G} x_i < 1$ in the objective function for any non-trivial feasible solution of problem (5). Therefore, at optimum, $z^* = z(x_1^*, x_2^*, ..., x_n^*)$, we have $\gamma(G) = \sum_{i=1}^{n} x_i^*$ and $\gamma_w^*(G) = w_G \cdot \sum_{i=1}^{n} \frac{w_i}{w_G} x_i^* = \sum_{i=1}^{n} w_i x_i^*$, i.e. $\gamma_w^*(G) = w_G \cdot (z^* - \gamma(G))$. Reassigning the graph vertices weights $w_i' = 1 + \frac{w_i}{w_G}$, $i = 1, ..., n$, the ILP formulation (5) becomes

$$\begin{cases} \text{minimize} & z(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} \left(1 + \frac{w_i}{w_G}\right) x_i = \sum_{i=1}^{n} w_i' x_i \\ \text{subject to:} & \sum_{v_i \in N[v_j]} x_i \geq 1, & j = 1, \ldots, n, \\ & x_i \in \{0, 1\}, & i = 1, \ldots, n. \end{cases} \tag{6}$$

This is the single-objective optimization problem of finding $\gamma_{w'}(G)$ and the corresponding minimum weight dominating set in G with respect to the vertex weights w_i', $i = 1, ..., n$. Clearly, $\gamma_{w'}(G) = z^* = z(x_1^*, x_2^*, ..., x_n^*)$ at optimum in (5) and (6), and

$\gamma^*_w(G) = w_G \cdot (\gamma_{w'}(G) - \gamma(G))$, as above. In other words, this provides a reduction from the two-objective problem of finding $\gamma^*_w(G)$ in G to the single-objective problem of finding $\gamma_{w'}(G)$ in G. Notice that $\gamma(G) = \lfloor \gamma_{w'}(G) \rfloor$.

After solving the ILP problem (5) or (6), to find other Pareto-optimal solutions to the two-objective optimisation problem, it is possible to add one constraint

$$\sum_{i=1}^{n} x_i \geq \gamma(G) + k, \tag{7}$$

to the formulations (5) and (6), $k = 1, 2, \ldots, n - \gamma(G) - 1$, and solve the corresponding ILP problems, one for each value of k. Checking weights of the obtained dominating sets would provide a set of Pareto efficient solutions to the two-objective problem for larger dominating set sizes. These general Pareto-optimal solutions can be important in applications, when size of the dominating set becomes less important and some priority is supposed to be given to the set weight.

Notice that here problem (6) is a special case of the general minimum weight dominating set problem because the new weights are restricted to be $1 < w'_i < 2$, $i = 1, ..., n$. This may allow us to better understand the initial problem (5) and help to find more efficient solution methods. On the other hand, for the general minimum weight dominating set problem, we can always assume that the vertex weights belong to the interval $(0, 1)$; for example, we can remove zero-weight vertices and their neighbours from the graph and divide the remaining weights by $w_{max} + 1$. Also, notice that for the linear relaxation of problems (5) and (6) with $x_i \in [0, 1]$, $i = 1, ..., n$, the objective function value $z' = z(x'_1, x'_2, ..., x'_n)$ at the linear relaxation optimum provides lower bounds on $\gamma(G)$ and $\gamma_{w'}(G)$, i.e. $\gamma(G) \geq \lfloor z' \rfloor$ and $\gamma_{w'}(G) \geq z'$. For $\gamma^*_w(G)$, more complicated lower bounds can be deduced. For instance, if $\gamma(G) \leq U$, where U is an upper bound, we get $\gamma^*_w(G) \geq w_G(z' - U)$, where we can trivially take $U = \sum_{i=1}^{n} \lceil x'_i \rceil$. On the other hand, any feasible solution to ILP problems (5) and (6), e.g. randomized rounding of their linear relaxation, allows us to recover the dominating set size and its weight. Therefore, to tackle the problem of finding $\gamma^*_w(G)$ formulated in (5), one can use general tools to solve the minimum weight dominating set problem in a graph.

3. Probabilistic constructions, upper bounds, and randomized heuristics

In this section, we focus on the minimum weight dominating set problem, i.e. finding upper bounds for $\gamma_w(G)$ in a graph G, analytically and algorithmically. Notice that, in view of the problem reduction from Section 2, the problems of finding $\gamma(G)$ and $\gamma^*_w(G)$ in G can be considered as particular cases of this more general problem.

3.1. Using the probabilistic method for upper bounds

In the case of optimization of dominating sets in vertex-weighted graphs, it is reasonable to consider some vertex degrees, e.g. the minimum, mean, or median vertex degree of the graph [27, 12], and vertex weights as key parameters that influence the likelihood of

inclusion of a vertex into an optimal dominating set with respect to its size and weight. For example, suppose we deal with the maximization version of the problem searching for a reasonably small-size dominating set X of the maximum total weight $w(X)$ in a graph G. Then, it is plausible that a vertex of high degree and heavy weight is more likely to be included into X than a vertex of lower degree and light weight. We will use this and similar assumptions in our applications of the probabilistic method and the corresponding randomized algorithms and techniques.

As above, we assume positive weights $w_i > 0$ for all vertices in a given vertex-weighted graph G, $i = 1, 2, ..., n$, and try to find a dominating set X in G such that $w(X)$ is the smallest possible. First, we generalize the probabilistic approach that was used to obtain the classic upper bounds of Theorems 1.1 and 1.2 for the domination number. As mentioned above, for this kind of optimization problem, the probability p_i of a vertex v_i to be in a "close-to-optimal" dominating set (small/large weight and small size) should depend on the vertex weight w_i and take into consideration vertex degrees in the graph. Suppose this probability is represented as a function $p_i = f(\bar{d}, w_i)$. Since we focus on the total weight minimization problem of a small-size dominating set X in G, we may assume that the probability of a vertex v_i to be in X depends on vertex degrees $\bar{d}$ and is reciprocally proportional to the vertex weight w_i. In other words, p_i can be computed by an expression of the form

$$p_i = p \cdot \frac{x}{w_i}, \tag{8}$$

where p is a coefficient depending on vertex degrees in G and x is a coefficient depending on vertex weights in G such that

$$0 \le p \cdot \frac{x}{w_i} \le 1 \quad \text{for all} \quad i = 1, \ldots, n.$$

In some situations, we may estimate the probabilities p_i to be the same and not dependent on weights, $i = 1, \ldots, n$. For example, we can assume that the weights do not vary too much among the vertices, that is, the ratio $w_{\max}/w_{\min}$ is reasonably close to 1. Then, the probability p_i for each vertex $v_i \in G$ to be in the dominating set X does not depend on weights. Indeed, substituting $x = w_{\text{ave}}$ in the expression (8) above, we obtain $p_i = pw_{\text{ave}}/w_i$. Now, $w_i \approx w_{\text{ave}}$ implies $p_i \approx p$ for every $i = 1, ..., n$. In the proof of the following upper bound, which is reminiscent of the classic bound for $\gamma(G)$ in (1.2), we assume equal probabilities $p_i = p$ when applying the probabilistic method.

Theorem 3.1. *For any graph G with $\delta \ge 1$,*

$$\gamma_w(G) \le \left(1 - \frac{\delta}{(1+\delta)^{1+1/\delta}}\right) w_G.$$

Proof. Let A be a set formed by an independent choice of vertices of G, where each vertex is selected with probability p. Denote by B the set of vertices that are not in A and do not have a neighbour in A: $B = V(G) - N[A]$. Consider the set $D = A \cup B$.

Clearly, D is a dominating set in G. The expected weight of such a set D is

$$\begin{aligned}
\mathbb{E}[w(D)] &= \mathbb{E}[w(A)] + \mathbb{E}[w(B)] \\
&= \sum_{i=1}^{n} w_i \cdot \mathbb{P}[v_i \in A] + \sum_{i=1}^{n} w_i \cdot \mathbb{P}[v_i \in B] \\
&= \sum_{i=1}^{n} w_i \cdot p + \sum_{i=1}^{n} w_i \cdot (1-p)^{d_i+1} \\
&= \sum_{i=1}^{n} w_i \cdot \left(p + (1-p)^{d_i+1}\right) \\
&\leq \sum_{i=1}^{n} w_i \cdot \left(p + (1-p)^{\delta+1}\right) \\
&= w_G \cdot \left(p + (1-p)^{\delta+1}\right).
\end{aligned}$$

Minimizing the function $\psi(p) = w_G \cdot \left(p + (1-p)^{\delta+1}\right)$, we obtain

$$p = 1 - \frac{1}{(\delta+1)^{1/\delta}}.$$

Therefore,

$$\mathbb{E}[w(D)] \leq \left(1 - \frac{\delta}{(\delta+1)^{1+1/\delta}}\right) w_G.$$

Since the expectation is an average value, there exists a particular dominating set satisfying the bound, as required. □

An analogue of Theorem 1.1 for $\gamma_w(G)$ easily follows from this proof:

Corollary 3.2. *For any graph G,*

$$\gamma_w(G) \leq \frac{\ln(\delta+1)+1}{.\delta+1} w_G.$$

Notice that the probability $p = 1 - \frac{1}{(\delta+1)^{1/\delta}}$ used in the probabilistic construction of Theorem 3.1 is the same as in the probabilistic construction used in the proof of Theorem 1.2 (e.g., see [14]). Therefore, in this particular case, the corresponding randomized heuristic described below in Algorithm 1 tends to obtain small-size dominating sets satisfying the bound of Theorem 1.2 at the same time.

Now suppose that $x = w_{\max}$ in the expression (8), i.e.

$$p_i = p \cdot \frac{w_{\max}}{w_i} \quad \text{for} \quad i = 1, \ldots, n.$$

Because $0 \leq p \cdot \frac{w_{\max}}{w_i} \leq 1$, we obtain

$$p \leq \frac{w_i}{w_{\max}} \quad \text{for} \quad i = 1, \ldots, n.$$

Hence, $p \leq w_{\min}/w_{\max}$, which is the third assumption in the following theorem. The second assumption in the statement of Theorem 3.3, $w_{\max}/w_{\text{ave}} \leq \delta + 1$, is needed to guarantee that $p_i \geq 0$.

Theorem 3.3. *Let G be a graph such that $\delta \geq 1$, $k = w_{\max}/w_{\text{ave}} \leq \delta + 1$, and $p = 1 - \left(\frac{k}{\delta+1}\right)^{1/\delta} \leq w_{\min}/w_{\max}$.*
Then

$$\gamma_w(G) \leq npw_{\max} + \sum_{i=1}^{n} w_i (1-p)^{d_i+1} \leq \left(1 - \frac{\delta k^{1/\delta}}{(\delta+1)^{1+1/\delta}}\right) kw_G.$$

Proof. Let A be a set formed by an independent choice of vertices of G, where each vertex v_i is selected with some probability

$$p_i = p \cdot \frac{w_{\max}}{w_i}, \quad 0 \leq p_i \leq 1, \quad i = 1, 2, \ldots, n.$$

Similar to the proof of Theorem 3.1, denote by B the set of vertices that are not in A and do not have a neighbour in A, i.e. $B = V(G) - N[A]$. Consider the set $D = A \cup B$. Clearly, D is a dominating set in G. The expected weight of D is

$$\begin{aligned}
\mathbb{E}[w(D)] &= \mathbb{E}[w(A)] + \mathbb{E}[w(B)] \\
&= \sum_{i=1}^{n} w_i \cdot \mathbb{P}[v_i \in A] + \sum_{i=1}^{n} w_i \cdot \mathbb{P}[v_i \in B] \\
&= \sum_{i=1}^{n} w_i \left(p \cdot \frac{w_{\max}}{w_i}\right) + \sum_{i=1}^{n} w_i \prod_{v_j \in N[v_i]} \left(1 - p \cdot \frac{w_{\max}}{w_j}\right) \\
&\leq \sum_{i=1}^{n} p\, w_{\max} + \sum_{i=1}^{n} w_i \prod_{v_j \in N[v_i]} \left(1 - p \cdot \frac{w_{\max}}{w_{\max}}\right) \\
&= npw_{\max} + \sum_{i=1}^{n} w_i (1-p)^{d_i+1} \\
&\leq (nw_{\max})p + \sum_{i=1}^{n} w_i (1-p)^{\delta+1} \\
&= (nw_{\max})p + (1-p)^{\delta+1} w_G. \qquad (9)
\end{aligned}$$

Now, the value of p is obtained by minimizing the function $\xi(p) = (nw_{\max})p + (1-p)^{\delta+1} w_G$ with respect to p:

$$p = 1 - \left(\frac{nw_{\max}}{(\delta+1)w_G}\right)^{1/\delta} = 1 - \left(\frac{nw_{\max}}{(\delta+1)nw_{\text{ave}}}\right)^{1/\delta} = 1 - \left(\frac{k}{\delta+1}\right)^{1/\delta}.$$

Notice that the assumption $k = \frac{w_{\max}}{w_{\text{ave}}} \leq \delta + 1$ guarantees that $0 \leq p \leq 1$, but we additionally need to assume $p \leq \frac{w_{\min}}{w_{\max}}$ to guarantee $0 \leq p_i \leq 1$ for any $i = 1, ..., n$.

We already know that

$$\gamma_w(G) \leq \mathbb{E}[w(D)] \leq npw_{\max} + \sum_{i=1}^{n} w_i (1-p)^{d_i+1}.$$

Also,

$$\begin{aligned}
\mathbb{E}[w(D)] &\leq pnw_{\max} + w_G(1-p)^{\delta+1} \\
&= pnw_{\text{ave}} \cdot \frac{w_{\max}}{w_{\text{ave}}} + w_G(1-p)^{\delta+1} \\
&= pw_G k + w_G \left(\frac{k}{\delta+1}\right)^{1+1/\delta} \\
&= \left(p + \frac{k^{1/\delta}}{(\delta+1)^{1+1/\delta}}\right) kw_G \\
&= \left(1 - \left(\frac{k}{\delta+1}\right)^{1/\delta} + \frac{k^{1/\delta}}{(\delta+1)^{1+1/\delta}}\right) kw_G \\
&= \left(1 - \frac{\delta k^{1/\delta}}{(\delta+1)^{1+1/\delta}}\right) kw_G,
\end{aligned}$$

as required. □

Note that the bounds of Theorem 3.3 are not necessarily better than the bound of Theorem 3.1. However, the heuristic based on the proof technique of Theorem 3.3 usually produces better results. This is because of the assumptions for Theorem 3.3. In the proof of Theorem 3.1, the differences in the vertex weights are ignored when trying to select vertices for the initial set A. In contrast, in the proof of Theorem 3.3, the probability of a vertex to be in the initial set A is assumed to be reciprocally proportional to its weight, which better reflects the situation with distribution of vertex weights. This and other points are illustrated later in Section 4.

Now, assume that the probability of a vertex v_i to be in the initial set A inversely depends on its weight w_i, $i = 1, 2, ..., n$. More precisely, the probability is computed by an expression of the form

$$p_i = p \cdot \left(1 - \frac{w_i}{\alpha}\right), \tag{10}$$

where α is a coefficient dependent on some weights in G, and

$$0 \leq p\left(1 - \frac{w_i}{\alpha}\right) \leq 1 \quad \text{for all} \quad i = 1, \ldots, n.$$

If we set $\alpha = w_{\min} + w_{\max}$, the next statement follows.

Theorem 3.4. *Let G be a graph such that $\delta \geq 1$, $z = w_{\max}/w_{\min} \leq \delta + 1$, and $q = 1 - \left(\frac{z}{\delta+1}\right)^{1/\delta}$. Then*

$$\gamma_w(G) \leq qzw_G + \sum_{i=1}^{n} w_i (1-q)^{d_i+1} \leq \left(1 - \frac{\delta z^{1/\delta}}{(\delta+1)^{1+1/\delta}}\right) zw_G.$$

Proof. Let us denote by A a set formed by an independent choice of vertices of G, where each vertex v_i is selected with some probability

$$p_i = p\left(1 - \frac{w_i}{\alpha}\right), \quad 0 \leq p_i \leq 1, \quad i = 1, 2, \ldots, n.$$

Similar to the proof of Theorem 3.1, we construct the set $B = V(G) - N[A]$ and the dominating set $D = A \cup B$. Taking into account that

$$\frac{w_{\min}}{\alpha} \leq 1 - \frac{w_i}{\alpha} \leq \frac{w_{\max}}{\alpha},$$

we obtain for the expected weight of D:

$$\begin{aligned}
\mathbb{E}[w(D)] &= \mathbb{E}[w(A)] + \mathbb{E}[w(B)] \\
&= \sum_{i=1}^{n} w_i \cdot \mathbb{P}[v_i \in A] + \sum_{i=1}^{n} w_i \cdot \mathbb{P}[v_i \in B] \\
&= \sum_{i=1}^{n} w_i p \left(1 - \frac{w_i}{\alpha}\right) + \sum_{i=1}^{n} w_i \prod_{v_j \in N[v_i]} \left(1 - p\left(1 - \frac{w_j}{\alpha}\right)\right) \\
&\leq \sum_{i=1}^{n} w_i p \cdot \frac{w_{\max}}{\alpha} + \sum_{i=1}^{n} w_i \prod_{v_j \in N[v_i]} \left(1 - p \cdot \frac{w_{\min}}{\alpha}\right) \\
&= \sum_{i=1}^{n} w_i p \cdot \frac{w_{\max}}{\alpha} + \sum_{i=1}^{n} w_i \left(1 - p \cdot \frac{w_{\min}}{\alpha}\right)^{d_i+1} \\
&= \frac{pzw_G}{z+1} + \sum_{i=1}^{n} w_i \left(1 - \frac{p}{z+1}\right)^{d_i+1}.
\end{aligned}$$

If we denote $q = p/(z+1)$, then

$$\begin{aligned}
\mathbb{E}[w(D)] &\leq qzw_G + \sum_{i=1}^{n} w_i (1-q)^{d_i+1} \\
&\leq qzw_G + \sum_{i=1}^{n} w_i (1-q)^{\delta+1} \\
&= \left(qz + (1-q)^{\delta+1}\right) w_G.
\end{aligned}$$

It is easy to see that the last expression is minimized at

$$q = 1 - \left(\frac{z}{\delta+1}\right)^{1/\delta}.$$

Notice that to guarantee that $0 \leq p_i \leq 1$ for all $i = 1, 2, ..., n$, we need to require $q = 1 - \left(\frac{z}{\delta+1}\right)^{1/\delta} \leq w_{\min}/w_{\max} = \frac{1}{z}$ ($q \geq 0$ is guaranteed by $z \leq \delta + 1$). However, it is possible to see that the inequality $1 - \left(\frac{z}{\delta+1}\right)^{1/\delta} \leq \frac{1}{z}$ is equivalent to

$$\left(1 - \frac{1}{z}\right)^{\delta} \leq \frac{z}{\delta+1},$$

which holds for any z, $1 \leq z \leq \delta+1$, $\delta \geq 1$. This is because the function $f(z) = \frac{z^{1+1/\delta}}{(\delta+1)^{1/\delta}} - z + 1$ is non-negative for all $z \in [1, \delta+1]$ (it is non-negative at each endpoint of the interval and at the critical point on the interval). Then, since naturally $z = w_{\max}/w_{\min} \geq 1$, the condition $z \leq \delta + 1$ is enough to have $0 \leq p_i \leq 1$ for all $i = 1, ..., n$. Thus, we have

$$\mathbb{E}[w(D)] \leq \left(1 - \frac{\delta z^{1/\delta}}{(\delta+1)^{1+1/\delta}}\right) zw_G,$$

as required. □

The second and third conditions for vertex weights in Theorem 3.3 can be rewritten as

$$\left(1-\frac{w_{\min}}{w_{\max}}\right)^{\delta}\cdot(\delta+1)\le k=\frac{w_{\max}}{w_{\text{ave}}}\le\delta+1,$$

and the corresponding vertex weights conditions in Theorem 3.4 are

$$1\le z=\frac{w_{\max}}{w_{\min}}\le\delta+1.$$

Clearly, there are problem instances where the conditions of Theorem 3.3 are satisfied, but the conditions of Theorem 3.4 are not satisfied, e.g. when $k=\frac{w_{\max}}{w_{\text{ave}}}$ is close to $\delta+1$, but $z=\frac{w_{\max}}{w_{\min}}>\delta+1$. On the other hand, some problem instances satisfy the conditions of Theorem 3.4, but not the conditions of Theorem 3.3, e.g. when $z=\frac{w_{\max}}{w_{\min}}$ and $k=\frac{w_{\max}}{w_{\text{ave}}}$ are close enough to 1, but $k=\frac{w_{\max}}{w_{\text{ave}}}<\left(1-\frac{w_{\min}}{w_{\max}}\right)^{\delta}\cdot(\delta+1)$. More precisely, a graph with $\delta=9$, a sufficiently large number of vertices, and vertex weights distributed uniformly from $w_{\min}=1$ to $w_{\max}=10$ will have $w_{\text{ave}}=5.5$ and will satisfy the conditions of Theorem 3.4, but not those of Theorem 3.3. Thus, both theorems and the two corresponding probabilistic constructions are meaningful.

3.2. Randomized heuristics

Theorems 3.1, 3.3, and 3.4 provide respectively optimized probabilities

$$\begin{aligned}
p_i =& 1-\frac{1}{(\delta+1)^{1/\delta}}\ ,\\
p_i =& \left(1-\left(\frac{w_{\max}}{(\delta+1)w_{\text{ave}}}\right)^{1/\delta}\right)\cdot\frac{w_{\max}}{w_i}\ , \quad \text{and}\\
p_i =& \left(1-\left(\frac{w_{\max}}{(\delta+1)w_{\min}}\right)^{1/\delta}\right)\cdot\left(1+\frac{w_{\max}-w_i}{w_{\min}}\right), \quad i=1,2,\ldots,n,
\end{aligned}$$

for corresponding randomized heuristics to find reasonably good solutions (by weight) to the weight-based problems quickly. The heuristics are described in Algorithm 1 below. Note that the probability p used in the proof of Theorem 3.1 is the same as in the probabilistic construction for the proof of Theorem 1.2 and the corresponding randomized algorithm. Therefore, this probability provides certain optimality in finding small-size dominating sets at the same time.

Algorithm 1: Randomized small-weight dominating set

Input: A vertex-weighted graph G.

Output: A small-weight dominating set D' of G.

`/* Form an initial subset` $A \subseteq V(G)$ `*/` Compute the probability p_i for each vertex $v_i \in V$, $i = 1, \ldots, n$;

Initialize set $A = \emptyset$;

foreach $v_i \in V(G)$ **do**

  With probability p_i, decide whether $v_i \in A$;

`/* Greedy heuristic to extend` A `to a dominating set` D `*/` **foreach** $v \in V(G) \setminus A$ **do**

  Compute $pc(v) = |N[v] \setminus \bigcup_{x \in A} N[x]|$;

while $\bigcup_{x \in A} N[x] \neq V(G)$ **do**

  Select a vertex $u \in U = \arg\max_{v \in V(G) \setminus A} pc(v)$;

  Put $A = A \cup \{u\}$;

  Update the values of $pc(v)$;

Put $D = A$;

`/*` D `is a dominating set of` G `*/`

`/* Finding a minimal-by-inclusion dominating set` $D' \subseteq D$ `*/` Order the vertices $v_i \in D$, $i = 1, \ldots, |D|$, so that $i \leq j \iff |N(v_i) \setminus D| \leq |N(v_j) \setminus D|$;

Put $D' := D$;

foreach *vertex* $v_i \in D$ **do**

  if $D' \setminus \{v_i\}$ *is a dominating set of* G **then**

    Put $D' := D' \setminus \{v_i\}$;

return D'

After the initial set A of vertices is randomly generated in Algorithm 1, a greedy heuristic is used to extend the initial set A recursively and to obtain a dominating set D of G. This is done by computing and updating the potential coverage parameter $pc(v) = |N[v] \setminus \bigcup_{x \in A} N[x]|$ for each vertex $v \in V(G) \setminus A$ and by choosing a vertex adjacent to the largest number of vertices which are currently not dominated. Then, a minimal by inclusion dominating set D' is found in Algorithm 1 by using another greedy heuristic. Notice that vertex weights are not used in the greedy strategies of the recursive vertex addition and removal. Naturally, running Algorithm 1 several times and selecting the best outcome usually provides better results, i.e. usage of more CPU time can be considered as a natural heuristic improvement for Algorithm 1, if necessary.

Computational experiments to illustrate and evaluate different solution methods are described in the next section. The deterministic method, which uses the ILP formulations from Section 2 and a generic ILP solver, allows us to solve to the optimum problem instances only for graphs of a few hundred vertices. Therefore, simple and quick heuristic solution methods provided by the probabilistic constructions in this section become important tools to tackle the problems for larger graphs.

4. Experimental evaluation

As an illustration for the theoretical results of Sections 2 and 3, we have implemented and tested the deterministic and heuristic solution methods for both problems, attempting to find $\gamma_w(G)$ and $\gamma_w^*(G)$ for random graph instances of two types. The first type of random graphs is obtained by using the classic Erdős–Rényi random graph model [16, 11]. The other type corresponds to the random graph model used to prove asymptotic sharpness of the upper bounds of Theorems 1.1 and 1.2 (e.g., see [26]).

All the solution methods were implemented using computer programming language C/C++, and the experiments were conducted on a Stone desktop PC with a 3.00 GHz Intel Core i5 processor and 16 GB of RAM, running Windows 10 Education OS, version 21H2. We used Gurobi Optimizer [18] to solve deterministically and heuristically both problems by considering their ILP formulations described in Section 2.

The ILP reduction (6) to the single-objective optimization problem of finding $\gamma_{w'}(G)$ from Section 2 was not used to search for heuristic randomized solutions for the two-objective optimization problem of finding $\gamma_w^*(G)$. This is because, in the reduced problem of finding $\gamma_{w'}(G)$, the average vertex weight is $w'_{\text{ave}} = 1 + 1/n$, where n is the order of the graph (in general, $w'_i = 1 + \frac{w_i}{w_G}$, where $w_G = \sum_{i=1}^{n} w_i$, so that $1 < w'_i < 2$, $i = 1, ..., n$). Assuming the initial weights w_i are distributed uniformly, this makes all the weights w'_i in the reduced single-objective optimization problem of finding $\gamma_{w'}(G)$ close to 1 and not varying much among the vertices, $i = 1, ..., n$. Thus, the assumptions described before Theorem 3.1 – the weights do not vary too much among the vertices, and the probabilities do not depend on weights – are satisfied in this case. Therefore, the corresponding randomized algorithm can be applied to the graph with the initial vertex weights directly. On the other hand, it is possible to see that the bounds and probabilities in Theorems 3.3 and 3.4 are close to those in Theorem 3.1 in the case of reduction (6) and when searching for $\gamma_{w'}(G)$.

In Subsections 4.1.1 and 4.2.1, we show that the randomized heuristics arising from the proofs of Theorems 3.3 and 3.4 are more sensitive to vertex weights in the graphs and provide better results in comparison to the randomized heuristic given by Theorem 3.1. In Subsections 4.1.2 and 4.2.2, we use the generic ILP solver Gurobi [18] to obtain some deterministic results for small size graphs and heuristic results for medium size graphs. These experiments show that the new randomized heuristics can find good quality initial solutions to the problems quickly. Also, these experiments show the quality of quickly found randomized heuristic solutions is reasonably close to the Gurobi heuristic solutions, while Gurobi is run on the test instances for a long time (30 minutes of CPU time). Finally, in Subsections 4.1.3 and 4.2.3, we consider large-scale graphs and show that the new randomized heuristics clearly provide better results than Gurobi, while using less memory and CPU time resources.

4.1. Erdős–Rényi random graph model

For the Erdős–Rényi model, denoted by $ER(n, p)$, a graph G of order $n \in \mathbb{N}$ is generated in such a way that, for each (unordered) pair of distinct vertices u and v of G, the

corresponding edge has the same probability $p \in [0,1]$ to be included in the graph. Such a model generates a graph whose degree distribution is a Binomial distribution with $n-1$ trials and probability p. In other words, given a graph order $n \in \mathbb{N}$, $n \geq 2$, and an edge-inclusion probability $p \in [0,1]$, one starts with the empty graph on n vertices. Then an edge uv is added to the graph with probability p (or, alternatively, not added with probability $1-p$) independently for each pair of distinct vertices $\{u,v\}$. This results in a graph $G \in ER(n,p)$.

As a part of the experiments, Erdős–Rényi random (unweighted) graphs on $100k$ vertices were generated for $k = 1,...,10$, using the edge-inclusion probability $p = 1/3$, which was manually determined to be the most illustrative in an ad-hoc way. (By the most illustrative here we mean graphs with larger minimum vertex degree to have better upper bounds, but, at the same time, to have large enough minimum cardinality dominating sets.) Ten graphs were generated for each k, $k = 1,...,10$, one hundred graphs in total, to form the set $ER(n = 100k, p = 1/3)$ of 100 test graphs. Then, for each graph $G \in ER(n = 100k, p = 1/3)$, $k = 1,...,10$, weights were assigned to the graph vertices as integer numbers in the range $\{101,102,...,200\}$, uniformly at random. The choice of vertex weights from the range $\{101,102,...,200\}$ was motivated by the assumption that opening a facility must have a certain minimum basic cost (100 in this case) plus a certain percentage of potential additional costs. On the other hand, this range was chosen to increase the likelihood that G satisfies the assumptions of Theorem 3.3 in a simple way.

4.1.1. Randomized heuristics and Erdős–Rényi graphs. The randomized heuristics based on Algorithm 1, arising from the probabilistic constructions of Theorems 3.1, 3.3, and 3.4, were run on each of the above 100 graph instances to quickly find a reasonably good solution for the minimum weight dominating set problems in G. Each of the randomized algorithms was run twenty times on each graph instance, and the best found solution (out of twenty) by the dominating set size and also by the dominating set weight were recorded. The aggregated averages for ten graph instances for each order $n = 100k$, $k = 1,...,10$, are presented in Table 1, together with the maximum average CPU run-times (out of three; the average run-times are for the corresponding twenty runs of each of the three heuristics). The corresponding average solution parameters for the best found set size are shown in the upper subrows, and the averages for the best found solutions by the set weight are shown in the lower subrows, for each $k = 1,...,10$.

Each of the randomized algorithms of Theorems 3.1, 3.3, and 3.4 finds better solutions by the set size for 73%, 68%, and 77% of all instances, respectively (corresponding dominating sets may have the same cardinality). In other words, here the three randomized algorithms show a similar performance by the dominating set size. However, the randomized algorithm arising from the proof of Theorem 3.1 is much less successful in finding the best heuristic solution by the set weight (only 20% of all instances). The randomized algorithms arising from the proofs of Theorems 3.3 and 3.4 find better solutions by the set weight for 45% and 36% of all instances, respectively, i.e. perform clearly better for this parameter. The upper bounds of Theorems 1.2 and 3.1 were satisfied for all the problem instances, with the results getting closer to the upper bounds for graphs of larger order.

Table 1. Aggregated results of running the randomized heuristics on the Erdős–Rényi graphs

$\lvert G\rvert$, $n = 100k$	Theorem 3.1		Theorem 3.3		Theorem 3.4		Max avg. time(s)
	Size	Wt	Size	Wt	Size	Wt	
$k = 1$	6	897	6.1	825.5	6	835.6	0.027
	6.3	885.2	6.2	824.4	6.2	833.1	
$k = 2$	7.5	1066.9	7.4	1041	7.5	1049.8	0.062
	7.8	1047.6	7.6	1011.8	7.7	1027.8	
$k = 3$	8.6	1225.5	8.5	1140.4	8.4	1217.7	0.1
	8.6	1225.5	8.7	1126.6	8.8	1185.1	
$k = 4$	9	1293.6	9.3	1306	9.1	1308.4	0.16
	9.2	1276.4	9.5	1290.3	9.5	1283.1	
$k = 5$	9.8	1387.5	10	1346.9	9.7	1341.8	0.21
	9.9	1381.4	10.1	1345.8	9.8	1340.5	
$k = 6$	10	1433.5	10.1	1385.7	10	1403.1	0.29
	10.2	1431.3	10.2	1384.1	10.5	1392.6	
$k = 7$	10.5	1463.9	10.4	1430.7	10.7	1462.8	0.37
	10.5	1463.9	10.4	1430.7	10.7	1462.8	
$k = 8$	11	1599.7	11	1532.7	10.8	1538.9	0.45
	11.1	1599.1	11.3	1524.2	11.1	1507.5	
$k = 9$	11.1	1626.3	11.1	1591.6	11	1557	0.56
	11.5	1605.1	11.6	1556.2	11.2	1531.2	
$k = 10$	11.3	1622.6	11.4	1542	11.3	1590.2	0.64
	11.4	1622	11.6	1534.6	11.6	1573.8	

It is possible to see from Table 1 that, in most of the cases, the heuristic methods derived from Theorems 3.3 and 3.4 provide better results for the two-objective optimization problem by weight and by size than that of Theorem 3.1, although both methods of Theorems 3.3 and 3.4 are designed for optimization by weight only.

4.1.2. Using a generic ILP solver on Erdős–Rényi graphs. For each weighted graph $G \in ER(n = 100k, p = 1/3)$, $k = 1, ..., 10$, considered above, we made an attempt to find exact deterministic solutions to the problems of computing $\gamma_w^*(G)$ and $\gamma_w(G)$ by using the ILP formulations described in Section 2 and Gurobi. This was successful in a reasonable amount of CPU time at most 30 min (1,800 sec) only for $k = 1, 2$, when computing $\gamma_w^*(G)$, and for $k = 1, 2, 3$, when computing $\gamma_w(G)$. For the larger (medium) size graphs, i.e. for $k \geq 3$, when computing $\gamma_w^*(G)$, and for $k \geq 4$, when computing $\gamma_w(G)$, Gurobi was run for 30 min (1,800 sec CPU time) as a heuristic solver in the hope the results would be close to the optimum. The best possible solution (BPS) of the ILP formulation in 1,800 sec CPU time was recorded. Also, the Gurobi's initial heuristic solutions (IHS) were recorded to compare to the quick best randomized heuristic solutions from Section 4.1.1. The aggregated results are presented in Table 2 and 3 (the exact optimal solution results are shaded).

Table 2. Aggregated results of running the generic ILP solver on the Erdős–Rényi graphs for $\gamma_w^*(G)$

$\lvert G\rvert$, $n = 100k$	Best randomized for $\gamma_w^*(G)$			ILP for $\gamma_w^*(G)$				
	size	wt	CPU time (s)	IHS size	IHS CPU time (s)	BPS size	BPS wt	BPS CPU time (s)
$k = 1$	6	835.6	0.027	7.9	0	5	620.4	1.33
$k = 2$	7.4	1,041	0.061	9.6	0.025	6	699.3	494.34
$k = 3$	8.4	1,217.7	0.1	10.5	0.084	6.8	800.4	1,800
$k = 4$	9	1,293.6	0.16	10.7	0.19	7	887.6	1,800
$k = 5$	9.7	1,341.8	0.21	11.1	0.35	7.8	933.9	1,800
$k = 6$	10	1,403.1	0.28	11.8	0.74	8	957.4	1,800
$k = 7$	10.4	1,430.7	0.36	12	1.14	8	1,017.4	1,800
$k = 8$	10.8	1,538.9	0.45	12.6	1.63	8.8	1,053.1	1,800
$k = 9$	11	1,557	0.54	12.6	2.26	9	1,076.6	1,800
$k = 10$	11.3	1,590.2	0.64	13	3.004	9	1,127.4	1,800

Table 3. Aggregated results of running the generic ILP solver on the Erdős–Rényi graphs for $\gamma_w(G)$

$\lvert G\rvert$, $n = 100k$	Best randomized for $\gamma_w(G)$			ILP for $\gamma_w(G)$				
	wt	size	CPU time (s)	IHS wt	IHS CPU time (s)	BPS wt	BPS size	BPS CPU time (s)
$k = 1$	824.4	6.2	0.027	1,205.5	0	616	5.1	0.4
$k = 2$	1,011.8	7.6	0.061	1,432.5	0.022	699.3	6	30.42
$k = 3$	1,126.6	8.7	0.1	1,600.6	0.081	762.1	7	646.74
$k = 4$	1,276.4	9.2	0.16	1,550.3	0.2	818.1	7.4	1,800
$k = 5$	1,340.5	9.8	0.21	1,651	0.39	859.7	8	1,800
$k = 6$	1,384.1	10.2	0.28	1,771.2	0.91	910.6	8.2	1,800
$k = 7$	1,430.7	10.4	0.36	1,827.6	1.37	931.6	8.9	1,800
$k = 8$	1,507.5	11.1	0.45	1,990.9	1.55	955.3	9	1,800
$k = 9$	1,531.2	11.2	0.54	1,851.4	2.04	972.6	9	1,800
$k = 10$	1,534.6	11.6	0.64	1,954.7	2.92	1,003.5	9.4	1,800

In the deterministic computational experiments (optimal solutions), only one graph instance was found to have a dominating set of a non-minimum size with weight lower than $\gamma_w^*(G)$, i.e. with $\gamma_w(G) < \gamma_w^*(G)$ (see shaded cells in Tables 2 and 3). It can be seen from these two tables that, for the medium size graphs ($k \geq 4$), the new randomized heuristics clearly provide better results and use much less CPU time than the IHS's of Gurobi. It can also be seen that, on average, the best solutions found by the randomized algorithms for $k = 1, 2, 3$ are about $34 - 48\%$ worse by weight and, for $k = 1, 2$, about $20 - 23\%$ worse by size than the optimal (deterministic) solutions. In comparison to the heuristic results for medium size graphs ($k \geq 4$) obtained by Gurobi in 1,800 sec, the very quick (less than a second) heuristic solutions by the new randomized algorithms are about $52 - 58\%$ worse by weight and about $22 - 30\%$ worse by size. However, this gap can be significantly reduced and even better results can be obtained by running the randomized algorithms for more iterations (e.g., for 1,800 sec CPU time). This is clearly shown for large size graphs in Section 4.1.3. The average CPU times to compute exact values of $\gamma_w^*(G)$ and $\gamma_w(G)$ (for $k = 1, 2, 3$) using the ILP formulations and Gurobi (see

shaded cells in Tables 2 and 3) clearly show the considerable growth of computational time requirements with respect to the graph order, as well as differences in computational complexity for finding $\gamma_w^*(G)$ and $\gamma_w(G)$.

4.1.3. Large-scale Erdős–Rényi graphs. Finally, we ran the three randomized algorithms and Gurobi on two large size Erdős–Rényi random graphs, generated as described at the beginning of Section 4.1. In these computational experiments, each of the four solvers was given 30 minutes (1,800 sec) of CPU time to find a solution to the problems corresponding to finding $\gamma_w^*(G)$ and $\gamma_w(G)$. One graph has $n = 20{,}000$ vertices, and the other has $n = 40{,}000$ vertices. Both graphs were generated with the edge probability $p = 0.1$.

For the Erdős–Rényi graph on $n = 20{,}000$ vertices, the randomized algorithms arising from Theorems 3.1, 3.3, and 3.4 respectively used 566, 558, and 556 iterations. For this graph, Gurobi produced lower bounds on the solutions by size, i.e. established that $\gamma(G) \geq 10$ (in 1,519 sec CPU time), and by weight, i.e. established that $\gamma_w(G) \geq 1{,}159$ (in 1,452 sec CPU time). The performance profiles of the four tested solvers on this large graph are presented in Figure 2 and Table 4.

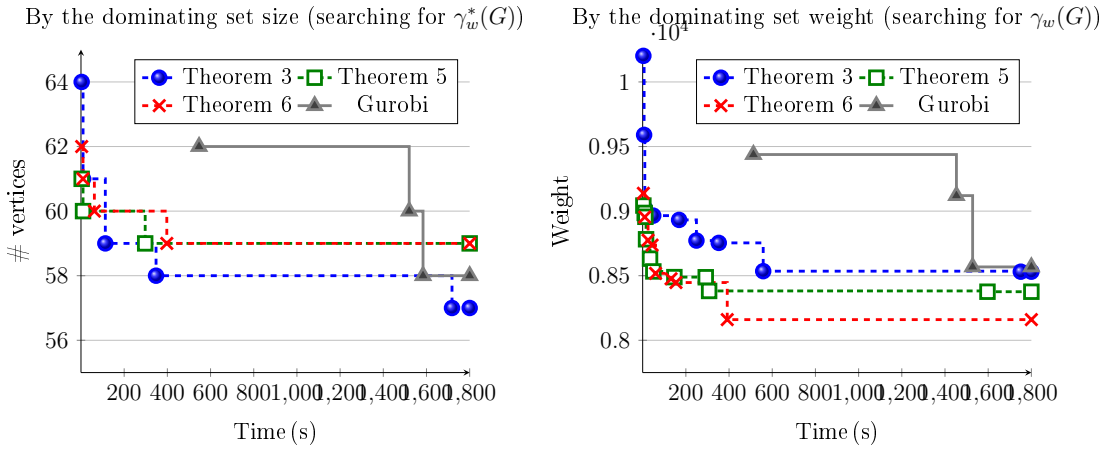

Fig. 2. Performance profiles of the solvers for the Erdős–Rényi graph on 20,000 vertices

Table 4. Best found results for the Erdős–Rényi graph on 20,000 veritces

Problem	Theorem 3.1			Theorem 3.3			Theorem 3.4			Gurobi		
	Size	Wt	Time (sec)	Size	Wt	Time (sec)	Size	Wt	Time (sec)	Size	Wt	Time (sec)
$\gamma_w^*(G)$	57	8,531	1,718	59	8,382	297	59	8,160	392	58	8,183	1,584
$\gamma_w(G)$	57	8,531	1,718	60	8,376	1,597	59	8,160	392	58	8,568	1,528

In general, the randomized algorithms find better quality solutions and much quicker than Gurobi. When trying to find $\gamma_w^*(G)$, the randomized algorithm arising from Theorem 3.1 provides the best solution by the dominating set size. This can be explained by the same optimal probability used in Theorems 3.1 and 1.2. When searching for $\gamma_w(G)$, all three randomized algorithms find better solutions and quicker than Gurobi. In this case, the randomized algorithm arising from Theorem 3.4 provides the best solution, which can

be explained by its sensitivity to the vertex weights in the graph. However, notice that Gurobi finds a better solution by weight when searching for $\gamma_w^*(G)$, which is still not the best (out of four).

For the Erdős–Rényi graph on $n = 40{,}000$ veritces, the randomized algorithms arising from Theorems 3.1, 3.3, and 3.4 respectively used 129, 132, and 132 iterations. For this graph, Gurobi ran out of memory and was not able to produce any solution. The results of the three randomized algorithms on this large size graph are presented in Table 5. The randomized algorithms found the same quality solutions by the dominating set size when trying to find $\gamma_w^*(G)$, with the randomized algorithm of Theorem 3.4 finding a set of this size quicker than the other two. However, the randomized algorithm of Theorem 3.1 finds the best (out of three) dominating set by weight of the same size. When searching for $\gamma_w(G)$, the best (out of three) solutions is found by the randomized algorithm of Theorem 3.4. However, this dominating set is two vertices larger than the dominating sets found by the randomized algorithms of Theorem 3.1 and Theorem 3.3.

Table 5. Best found results for the Erdős–Rényi graph on 40,000 veritces

Problem	Theorem 3.1			Theorem 3.3			Theorem 3.4		
	Size	Wt	Time (sec)	Size	Wt	Time (sec)	Size	Wt	Time (sec)
$\gamma_w^*(G)$	65	9,284	1,730	65	9,418	1,616	65	9,654	1,118
$\gamma_w(G)$	65	9,284	1,730	65	9,418	1,616	67	9,207	1,459

4.2. Sun graphs

Given a minimum vertex degree $\delta \in \mathbb{N}$, $\delta \geq 3$, we define the random *sun graph* model, denoted by $SG(\delta)$, following the description in [26]. Assuming we are given a complete graph $K_{\lfloor \delta \ln \delta \rfloor}$ with the vertex set $V(K_{\lfloor \delta \ln \delta \rfloor})$, we add an independent set of δ other vertices $O_\delta = \{v_1, v_2, ..., v_\delta\}$ and make each vertex v_i, $i = 1, 2, ..., \delta$, adjacent to exactly δ vertices of the set $V(K_{\lfloor \delta \ln \delta \rfloor})$, chosen uniformly at random. The resulting graph $G \in SG(\delta)$ has the vertex set $V(G) = V(K_{\lfloor \delta \ln \delta \rfloor}) \cup O_\delta$ and $m = \binom{\lfloor \delta \ln \delta \rfloor}{2} + \delta^2$ edges. We call this graph G a *sun graph*, which is clearly a split graph on $n = \lfloor \delta \ln \delta \rfloor + \delta$ vertices with $\delta(G) = \delta$.

Random (unweighted) sun graphs were generated for $\delta = 50k$, $k = 1, 2, ..., 7$, ten graphs for each k, seventy graphs in total, to form the set $SG(\delta = 50k)$. Then, for each $G \in SG(\delta = 50k)$, $k = 1, 2, ..., 7$, weights were assigned to the vertices in the clique $V(K_{\lfloor \delta \ln \delta \rfloor})$ as integers in the range $\{101, 102, ..., 200\}$ and, to the vertices in the independent set O_δ, as integers in the range $\{1, 2, ..., 100\}$, uniformly at random.

4.2.1. Randomized heuristics and sun graphs. Three randomized heuristics, corresponding to Algorithm 1 and using the optimized probabilities of Theorems 3.1, 3.3, and 3.4, were run on each of these seventy sun graph instances. Each of the randomized algorithms was run twenty times on each sun graph instance, and the best found solutions (out of twenty) by the dominating set size and also by the dominating set weight were recorded. See the aggregated averages for ten sun graph instances for each $\delta = 50k$, $k = 1, ..., 7$, respectively in the upper and lower subrows of Table 6, together with the maximum average

CPU run-times.

Table 6. Aggregated results of running the randomized heuristics on the sun graphs

Min degree, $\delta = 50k$	Theorem 3.1		Theorem 3.3		Theorem 3.4		Max avg. time(s)
	Size	Wt	Size	Wt	Size	Wt	
$k = 1$	6.1	876	5.8	783.9	6	822.5	0.12
	6.3	864	5.8	783.9	6.1	812.8	
$k = 2$	9	1,279.7	8.7	1,221.6	8.5	1,164.5	0.44
	9	1,279.7	9.1	1,210.2	8.6	1,155.3	
$k = 3$	11	1,669	10.9	1,503.6	10.8	1,457	1.04
	11.3	1,645.8	11.2	1,493.6	10.9	1,456.2	
$k = 4$	12.5	1,818.3	12.3	1,671.5	12.2	1,734.6	2.14
	12.8	1,802	12.5	1,671.2	12.5	1,681.3	
$k = 5$	14	2,070.4	13.7	1,905.8	13.8	1,959	3.11
	14.3	2,048.3	14	1,872.2	14.2	1,927.4	
$k = 6$	15.1	2,211.6	15.1	2,079.3	14.9	2,143.5	4.95
	15.3	2,194.6	15.3	2,061.3	15.4	2,088.7	
$k = 7$	16	2,375	15.5	2,147.5	16	2,245.2	6.62
	16.1	2,371.7	15.6	2,145.5	16.2	2,213.1	

Table 6 shows that the randomized algorithm corresponding to the probabilistic construction of Theorem 3.1 performs worse on sun graphs than those corresponding to Theorems 3.3 and 3.4. The randomized algorithms corresponding to Theorems 3.1, 3.3, and 3.4 find better dominating sets by size for $52.9\%, 71.4\%$, and 70% of all instances, respectively, and better dominating sets by weight for $7.1\%, 42.9\%$, and 50% of all instances, respectively. The upper bounds of Theorems 1.2 and 3.1 were satisfied for all the problem instances. Table 6 also shows that the heuristic methods derived from Theorems 3.3 and 3.4 provide better results for the two-objective optimization problem than that of Theorem 3.1 in all the cases, although both methods of Theorems 3.3 and 3.4 are designed for optimization by weight only.

4.2.2. Using a generic ILP solver on sun graphs. For each weighted sun graph $G \in SG(\delta = 50k)$, $k = 1, 2, ..., 7$, we attempted to find exact deterministic solutions to the problems of finding $\gamma_w^*(G)$ and $\gamma_w(G)$ by using the ILP formulation (5) from Section 2 and Gurobi. This was successful in a reasonable amount of CPU time of at most 30 min (1,800 sec) only for $k = 1, 2$ (245 and 560 vertices, respectively) when computing $\gamma_w^*(G)$, and for $k = 1, 2, 3$ (245, 560, and 901 vertices, respectively) when computing γ_w. In these deterministic computational experiments, almost half of the graphs (nine out of twenty) were found to have a dominating set of a non-minimum size with weight lower than $\gamma_w^*(G)$, i.e. $\gamma_w(G) < \gamma_w^*(G)$ (see Tables 7 and 8). For the larger (medium) size sun graphs, i.e. for $k \geq 3$, when computing $\gamma_w^*(G)$, and for $k \geq 4$, when computing $\gamma_w(G)$, Gurobi was run for 30 min (1,800 sec CPU time) as a heuristic solver. The aggregated results are presented in Table 7 and 8 (the deterministic solution results are shaded).

Table 7. Aggregated results of running the ILP generic solver on the sun graphs for $\gamma_w^*(G)$

Min degree, $\delta = 50k$	Best randomized for $\gamma_w^*(G)$			ILP for $\gamma_w^*(G)$				
	size	wt	CPU time (s)	IHS size	IHS CPU time (s)	BPS size	BPS wt	BPS CPU time (s)
$k = 1$	5.8	783.9	0.012	50	0.026	4.5	583.1	2.78
$k = 2$	8.5	1,164.5	0.43	100	0.18	6	758.8	183.68
$k = 3$	10.8	1,457	1.03	150	0.47	7.8	947.7	1,800
$k = 4$	12.2	1,734.6	2.14	200	0.92	8.8	1,120	1,800
$k = 5$	13.7	1,905.8	3.11	250	1.57	9.9	1,169.1	1,800
$k = 6$	14.9	2,143.5	4.76	300	2.51	10.8	1,341.3	1,800
$k = 7$	15.5	2,147.5	6.6	350	3.76	11.4	1,489.2	1,800

Table 8. Aggregated results of running the ILP generic solver on the sun graphs for $\gamma_w(G)$

Min degree, $\delta = 50k$	Best randomized for $\gamma_w(G)$			ILP for $\gamma_w(G)$				
	wt	size	CPU time (s)	IHS wt	IHS CPU time (s)	BPS wt	BPS size	BPS CPU time (s)
$k = 1$	783.9	5.8	0.12	7,623.5	0.031	564.9	4.9	0.9
$k = 2$	1,155.3	8.6	0.43	15,175.6	0.19	747.1	6.5	31.44
$k = 3$	1,456.2	10.9	1.03	22,540.9	0.46	859.8	8	1,381.21
$k = 4$	1,671.2	12.5	2.14	30,105.4	0.91	1,005.9	9.2	1,800
$k = 5$	1,872.2	14	3.11	37,734	1.56	1,122.5	10.2	1,800
$k = 6$	2,061.3	15.3	4.8	45,061.2	2.48	1,218.3	11.2	1,800
$k = 7$	2,145.5	15.6	6.6	52,535	3.71	1,283.3	11.9	1,800

It can be seen from Tables 7 and 8 that, for the small and medium size sun graphs, the new randomized heuristics provide solutions of about one order of magnitude better than the IHS's of Gurobi, but Gurobi finds its (apparently trivial) IHS's faster. Also, on average, the best solutions found by the randomized algorithms for $k = 1, 2, 3$ are about $39 - 69\%$ worse by weight and, for $k = 1, 2$, about $29 - 42\%$ worse by size than the optimal (deterministic) solutions. The computing time of the deterministic solution methods (ILP) becomes prohibitively high for graphs of 1,259 and more vertices in these experiments. In comparison to the heuristic results for medium size sun graphs ($k \geq 4$) obtained by Gurobi in 1,800 sec, the very quick (less than ten seconds) heuristic solutions by the new randomized algorithms are about $66-69\%$ worse by weight and about $36-39\%$ worse by size. Again, this gap can be significantly reduced and even better results can be obtained by running the randomized algorithms for more iterations in the same amount of time (1,800 sec). In Section 4.2.3, we show that the randomized algorithms provide clearly better results for large size sun graphs when run for the same amount of time as Gurobi. The average CPU run-times to compute exact values of $\gamma_w^*(G)$ and $\gamma_w(G)$ (for $k = 1, 2, 3$) using the ILP formulations and Gurobi (see BPS columns in Tables 7 and 8) clearly show the considerable growth of computational time requirements with respect to the graph order, as well as differences in computational complexity for finding $\gamma_w^*(G)$ and $\gamma_w(G)$ in sun graphs. It can be seen from Tables 2, 3, 7, and 8 that running Gurobi on the ILP formulations of the problems is much more efficient for the sun graphs than for the Erdős–Rényi graphs. This may be explained by the structure of the sun graphs, which

are split graphs. However, both types of random graphs exhibit exponential growth in the ILP solution time with respect to the graph order.

4.2.3. Large-scale sun graphs. Finally, we ran the three randomized algorithms and Gurobi on two large size sun graphs, generated as described at the beginning of Section 4.2, except that all weights were assigned to the graph vertices as integers in the range $\{101, 102, ..., 200\}$, uniformly at random. One graph has the minimum vertex degree δ = 1,250 (k = 25) and n =10,163 vertices, the other has δ = 2,500 (k = 50) and n = 22,060 vertices. Each of the four solvers was given 30 min (1,800 sec) of CPU time to find a solution to the problems corresponding to searching for $\gamma_w^*(G)$ and $\gamma_w(G)$.

For the sun graph on 10,163 vertices (δ = 1,250), the randomized algorithms arising from Theorems 3.1, 3.3, and 3.4 respectively used 125, 101, and 87 iterations. For this graph, Gurobi was able to find only the initial trivial solution of 1,250 vertices (in 128 sec) and weight 189,407 (in 116 sec). No other solution was found by Gurobi in 1,800 sec. The results of the randomized algorithms are presented in Table 9 below. When trying to find $\gamma_w^*(G)$, the randomized algorithm arising from Theorem 3.1 provides the best solution by the dominating set size. Again, this can be explained by the same optimal probability used in Theorems 3.1 and 1.2. When searching for $\gamma_w(G)$, the randomized algorithm arising from Theorem 3.4 provides the best solution, which, similarly to the large Erdős–Rényi graphs, can be explained by its sensitivity to the vertex weights in the graph.

Table 9. Best found results for the sun graph on 10,163 vertices

Problem	Theorem 3.1			Theorem 3.3			Theorem 3.4		
	Size	Wt	Time (sec)	Size	Wt	Time (sec)	Size	Wt	Time (sec)
$\gamma_w^*(G)$	24	3,557	1,246	25	3,726	1,259	25	3,461	251
$\gamma_w(G)$	24	3,557	1,246	27	3,558	756	25	3,461	251

For the sun graph on 22,060 vertices (δ = 2,500), the randomized algorithms arising from Theorems 3.1, 3.3, and 3.4 respectively used 11, 9, and 8 iterations. For this graph, Gurobi ran out of memory and was not able to produce any solution. The results of the randomized algorithms are described in Table 10. For this large sun graph, the best solutions are found by the randomized algorithm arising from Theorem 3.4. While this seems to be normal for the search by weight, the result by size illustrates the randomized nature of the three solvers.

Table 10. Best found results for the sun graph on 22,060 vertices

Problem	Theorem 3.1			Theorem 3.3			Theorem 3.4		
	Size	Wt	Time (sec)	Size	Wt	Time (sec)	Size	Wt	Time (sec)
$\gamma_w^*(G)$	34	5,253	393	34	5,054	559	33	4,720	557
$\gamma_w(G)$	35	5,207	182	35	4,842	223	33	4,720	557

5. Conclusions

We have considered the problems of finding a minimum-size dominating set of the smallest weight and of finding a minimum weight dominating set in a vertex-weighted simple graph. The two-objective problem is different from the classic single-objective problems of finding a smallest cardinality dominating set or a smallest weight dominating set in a graph, but is shown to be a particular case of one of them. Using the probabilistic method, we have shown three generalizations of the classic upper bounds for the domination number. The probabilistic constructions used to prove the new bounds provide randomized heuristics to find quick good quality solutions for the weighted domination problems. We have shown a connection between the problems in graphs via ILP formulations of the problems and provided a reduction from the two-objective problem to the minimum weight dominating set problem.

The simple and efficient randomized heuristics presented in this paper can be used, for example, to quickly find a better initial solution for the local search and other heuristics presented in [1, 5], or to speed up deterministic algorithms that use binary search in an interval, like the algorithm in [21]. The proposed randomized heuristics are very efficient in usage of computer memory and CPU time. Since the probability used in the proof of Theorem 3.1 is the same as in the proof of Theorem 1.2, the corresponding randomized algorithm from the framework of Algorithm 1 provides a certain optimality in finding a good dominating set not only by weight, but also by size. On the other hand, the computational experiments show that the randomized algorithms arising from Theorems 3.3 and 3.4 are more sensitive to the vertex weight parameters and usually provide better results than the randomized algorithm arising from Theorems 3.1, in particular, in the case of sun graphs of Section 4.2. Clearly, some heuristic enhancements can be used to make Algorithm 1 more effective and efficient, in particular, when forming the initial dominating set D in G and finding the minimal (by inclusion) dominating set D'. For future experiments and research in this direction, the random vertex-weighted graph instances used for the (reproducible) computational experiments in this paper are made available online [9].

The upper bound and probabilistic construction of Theorem 3.1 are reminiscent and similar to those used in Theorem 1.2 for unweighted graphs, and the assumptions of Theorem 3.1 require the graph weights not to vary too much among the vertices. On the other hand, ILP reduction (6) from Section 2 guarantees that the reduced problem has all the vertex weights close to one and not varying too much. In this case, the bounds and probabilities of Theorems 3.3 and 3.4 turn out to be close to those of Theorem 3.1, which makes all three of the theorems useful in the context of reduction.

As a direction for future work, it would be interesting to consider a relaxation of the problems to find dominating sets whose size and/or weight are within a certain limit from $\gamma(G)$ and/or $\gamma_w(G)$ in a graph, respectively, and a connection between $\gamma(G)$ and vertex weight parameters. Also, it would be interesting to devise other heuristic and deterministic solution methods for the problems. In particular, it would be useful to develop some deterministic solution algorithms, like the state-of-the-art deterministic algorithms for

the minimum-size dominating sets [3, 4, 21]. Eventually, the upper bounds and outcomes of the randomized algorithms presented in this paper can be used to reduce the search space in smart exhaustive search algorithms, like backtracking and branch-and-bound, for the minimum-weight dominating sets. Using different initial solutions in the deterministic or heuristic solvers would be a good direction of future research to extend these results.

Acknowledgement

Lukas Dijkstra acknowledges funding from the Maths DTP 2020, EPSRC grant EP/V520159/1.

References

[1] M. Albuquerque and T. Vidal. An efficient matheuristic for the minimum-weight dominating set problem. *Applied Soft Computing*, 72:527–538, 2018. https://doi.org/10.1016/j.asoc.2018.06.052.

[2] N. Alon and J. H. Spencer. *The Probabilistic Method*. John Wiley & Sons, 2016.

[3] N. Assadian. Dominating sets of the Cartesian products of cycles. Technical report, Department of Computer Science, University of Victoria, Canada, 2019.

[4] W. H. Bird. Computational Methods for Domination Problems. Technical report, Department of Computer Science, University of Victoria, Canada, 2017.

[5] S. Bouamama and C. Blum. A hybrid algorithmic model for the minimum weight dominating set problem. *Simulation Modelling Practice and Theory*, 64:57–68, 2016. https://doi.org/10.1016/j.simpat.2015.11.001.

[6] Y. Caro and Y. Roditty. Improved bounds for the product of the domination and chromatic numbers of a graph.(english summary). *Ars Combinatoria*, 56:189–191, 2000.

[7] N. Chen, J. Meng, J. Rong, and H. Zhu. Approximation for dominating set problem with measure functions. *Computing and Informatics*, 23(1):37–49, 2004.

[8] P. Corcoran and A. Gagarin. Heuristics for k-domination models of facility location problems in street networks. *Computers & Operations Research*, 133:105368, 2021. https://doi.org/10.1016/j.cor.2021.105368.

[9] L. Dijkstra, A. Gagarin, and V. Zverovich. Experimental data set for dominating sets in random vertex-weighted graphs. *Figshare*, 2025. https://doi.org/10.6084/m9.figshare.29257469.v1. Figshare. Retrieved: June 12, 2025.

[10] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer Science & Business Media, 2012.

[11] P. Erdos and A. Rényi. On the evolution of random graphs. *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, 5(1):17–60, 1960.

[12] A. Gagarin and P. Corcoran. Multiple domination models for placement of electric vehicle charging stations in road networks. *Computers & Operations Research*, 96:69–79, 2018. https://doi.org/10.1016/j.cor.2018.03.014.

[13] A. Gagarin, A. Poghosyan, and V. Zverovich. Upper bounds for α-domination parameters. *Graphs and Combinatorics*, 25:513–520, 2009. https://doi.org/10.1007/s00373-009-0864-6.

[14] A. Gagarin, A. Poghosyan, and V. Zverovich. Randomized algorithms and upper bounds for multiple domination in graphs and networks. *Discrete Applied Mathematics*, 161(4-5):604–611, 2013. https://doi.org/10.1016/j.dam.2011.07.004.

[15] M. R. Gary and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. WH Freman and Co, 1979.

[16] E. N. Gilbert. Random graphs. *The Annals of Mathematical Statistics*, 30(4):1141–1144, 1959.

[17] D. V. Greetham, A. Poghosyan, and N. Charlton. Weighted alpha-rate dominating sets in social networks. In *2014 Tenth International Conference on Signal-Image Technology and Internet-Based Systems*, pages 369–375. IEEE, 2014. https://doi.org/10.1109/SITIS.2014.51.

[18] Gurobi Optimization, LLC. Gurobi optimizer. https://www.gurobi.com/, 2025. Accessed: June 21, 2025.

[19] J. Harant, A. Pruchnewski, and M. Voigt. On dominating sets and independent sets of graphs. *Combinatorics, Probability and Computing*, 8(6):547–553, 1999. https://doi.org/10.1017/S0963548399004034.

[20] T. W. Haynes, S. Hedetniemi, and P. Slater. *Fundamentals of Domination in Graphs*. Marcel Dekker, New York, 1998.

[21] E. P. Inza, N. Vakhania, J. M. S. Almira, and F. A. H. Mira. Exact and heuristic algorithms for the domination problem. *European Journal of Operational Research*, 313(3):926–936, 2024. https://doi.org/10.1016/j.ejor.2023.08.033.

[22] C.-W. Lee. Domination in digraphs. *Journal of the Korean Mathematical Society*, 35(4):843–853, 1998.

[23] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcp characterization of np. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 475–484, 1997. https://doi.org/10.1145/258533.258641.

[24] F. Wang, H. Du, E. Camacho, K. Xu, W. Lee, Y. Shi, and S. Shan. On positive influence dominating sets in social networks. *Theoretical Computer Science*, 412(3):265–269, 2011.

[25] K. Weber. Domination number for almost every graph. *Rostocker Mathematisches Kolloquium*, 16:31–43, Jan. 1981.

[26] V. Zverovich and A. Poghosyan. On roman, global and restrained domination in graphs. *Graphs and Combinatorics*, 27:755–768, 2011. https://doi.org/10.1007/s00373-010-0992-z.

[27] V. Zverovich. *Modern Applications of Graph Theory*. Oxford University Press, 2021.