Laboratoire d'Analyses et Modélisation de Systèmes pour l'Aide à la Décision
UMR 7243

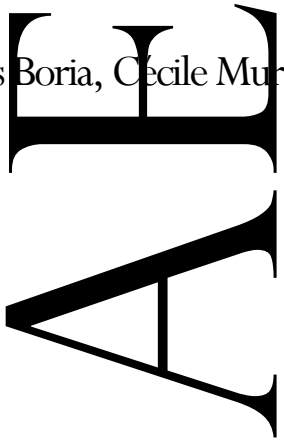
CAHIER DU LAMSADE

325

Juillet 2012

An emergency management model for a wireless sensor network problem

Nicolas Boria, Cécile Murat, Vangelis Th. Paschos



An emergency management model for a wireless sensor network problem*

Nicolas Boria[(a)] Cécile Murat Vangelis Th. Paschos[(b)]

Paris Sciences et Lettres Research University, Université Paris-Dauphine, LAMSADE
CNRS, UMR 7243
{boria,murat,paschos}@lamsade.dauphine.fr

July 13, 2012

Abstract

We present a natural wireless sensor network problem, which we model as a probabilistic version of the MIN DOMINATING SET problem. We show that this problem, being a generalization of the classical MIN DOMINATING SET, is **NP**-hard, even in bipartite graphs. We first study the complexity of PROBABILISTIC MIN DOMINATING SET in graphs where MIN DOMINATING SET is polynomial, mainly in trees and paths and then we give some approximation results for it.

1 Wireless sensor networks and probabilistic dominating set

Very frequently, in wireless sensor networks [31], one wishes to identify a subset of sensors, called "master" sensors, that will have a particular role in messages transmission, namely, to centralize and process messages sent by the rest of the sensors, called "slave" sensors, in the network. These latter sensors will be only nodes of intermediate messages transmission, while the former ones will be authorized to make several operations on messages received and will be, for this reason, better or fully equipped and preprogrammed.

So, the objective for designing such a network is to identify a subset of sensors (the master sensors) such that, every other sensor is linked to some sensor in this set. In other words, one wishes to find a dominating set in the graph of sensors. If we suppose that equipment of master sensors induces some additional cost with respect to that of the slave ones, if this cost is the same for all master sensors, we have a minimum cardinality dominating set problem (MIN DOMINATING SET), while if any master sensor has its own cost, we have a minimum weight dominating set problem.

Sensors can be broken down any time but, since the network must be remain operational, once a break down, one must be able to recompute a new set of master sensors very quickly (and in any case as quickly as solution from scratch is not allowed). This is the problem handled in this paper.

For simplicity, we deal with master sensors of uniform equipment cost (hopefully, it will be clear later that this assumption is not restrictive for the model) and we suppose that any sensor, can be broken down with some probability q_i (so, it remains operational, i.e., present in the

*Research supported by the French Agency for Research under the program TODO, ANR-09-EMER-010

[(a)]Author's current address: D.A.I., Politecnico di Torino, Torino, Italy

[(b)]Institut Universitaire de France

network, with probability $p_i = 1 - q_i$) depending on its construction, proper equipment, age, etc. Informally, the approach we propose, in order to maintain the network operational at any time, is the following:

- design an algorithm M that, given a set D of master sensors of the network, if some sensors of D fail, it adapts D to the surviving network (in other words, the new D becomes the new master set of sensors for the surviving part of the network); this algorithm must be as efficient as possible, in order that long idle periods for the network are avoided;
- given the network, its sensors' surviving probabilities p_i and M, compute a solution D^*, called *"a priori" solution* that, informally, has the basic property to be close, in a sense that will be defined later, to every possible solution obtained by a modification of D^* when applying M.

The problem of determining an optimal a priori master set D^* is the problem handled in this paper.

It is clear that given a network of sensors, identifying master set of them is equivalent to determining a dominating set in the associated graph where vertices are the sensors of the network and, for any linked pair of them, an edge links the corresponding vertices. The MIN DOMINATING SET problem is formally defined as follows. Let $G(V, E)$ be a connected graph defined on a set V of vertices with a set $E \subseteq V \times V$ of edges. A vertex-set D is said to be a dominating set of G if, for any $v \in V \setminus D$, v has at least one neighbor in D. In the MIN DOMINATING SET problem, the objective is to determine a minimum-size dominating set in G. The decision version of MIN DOMINATING SET problem is one of the first 21 **NP**-complete problems [21] and remains **NP**-complete even in bipartite graphs, while it is polynomial in trees.

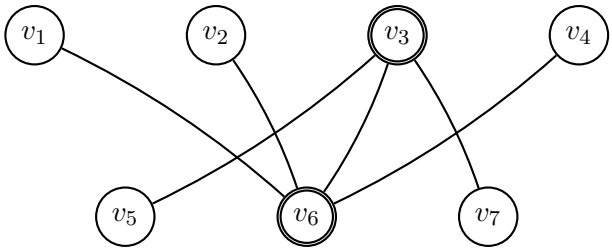


Figure 1: A graph together with a dominating set (bold-circled vertices).

In our sensor network problem handled, we associate a probability p_i to every vertex $v_i \in V$ (the probability that sensor i remains operational). We specify also a strategy M, called *modification strategy*, that when given a dominating set D of G and a subgraph $G' = G[V']$ induced by a set $V' \subseteq V$ (the surviving sensors), it transforms D into a set D' that is a dominating set of G'. Let us note that the simplest modification strategy, consisting of just returning $D' = D \cap V'$ is not feasible since it does not always produces feasible dominating sets for G'. For example, consider the graph of Figure 1 for which the set $D = \{v_3, v_6\}$ is dominating. If we consider $V' = \{v_1, v_2, v_4, v_6, v_7\}$, then $D' = D \cap V' = \{v_6\}$ is no longer a dominating set of G'. Hence, we will consider the following somewhat more complicated modification strategy M associated with our problem:

> given a graph $G(V, E)$, a dominating set D of G and subgraph G', set $D' := \emptyset$ and, for any $v_i \in V'$:

- if $v_i \in D$, then set $D' := D' \cup \{v_i\}$;
- otherwise, if $\Gamma(v_i) \cap (D \cap V') = \emptyset$, then set $D' := D' \cup \{v_i\}$, where, for any vertex v, $\Gamma(v)$ denotes the set of its neigbors.

In other words, M first takes $D \cap V'$ in D' and then completes it with all the non-dominated vertices of V'. It is easy to see that the complexity of M is polynomial, since it is bounded by the sum of the degrees of the vertices of $V' \setminus D$ that is at most $O(|E|)$.

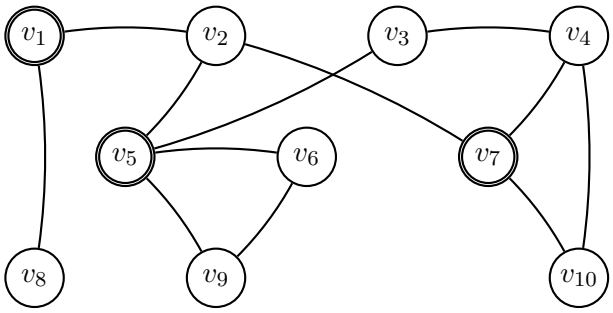


Figure 2: The graph of Example 1.

Example 1. Consider the graph of Figure 2 and a dominating set $D = \{v_1, v_5, v_7\}$ and its subgraph induced by the set $V' = \{v_1, v_2, v_3, v_6, v_7, v_9\}$ (Figure 3). For these vertices two cases are possible:

1. either a vertex is in D and it will be part of D' also (this is the case of v_1 and v_7);
2. or a vertex does not belong to D and in this case it will be added to D' if all its neighbors that were in D are not in V' (this is the case of v_3, v_6 and v_9).

The set D' is shown in Figure 3 by the bold-circled vertices ∎

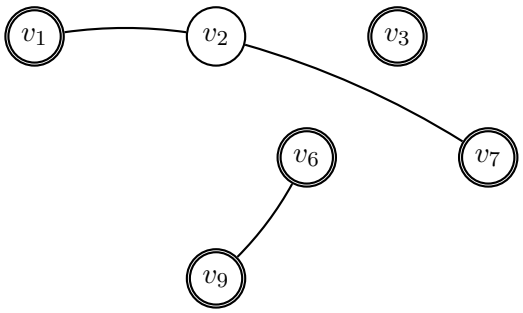


Figure 3: The subgraph $G[V']$ and the solution computed by M in Example 1.

Consider now a graph $G(V, E)$, a probability p_i for every vertex $v_i \in V$, a dominating set D of G and the modification strategy M just specified. The *functional* of M can be expressed by:

$$\mathbb{E}(G, D, \mathtt{M}) = \sum_{V' \subseteq V} \Pr\left[V'\right] \left|D'\right| \tag{1}$$

where $\Pr[V']$ is the probability of V' (i.e., the probability that only the vertices of V' will not be broken down) and D' is the dominating set returned by $\mathtt{M}$ for $G[V']$.

Following $\mathtt{M}$, if a vertex $v_i \in D$, then it will be always in D' if it belongs to V'; hence, its contribution to $\mathbb{E}(G, D, \mathtt{M})$ in (1) will be equal to p_i. On the other hand, if $v_i \notin D$ but $v_i \in V'$, it will be included in D' only if all its neigbors in G that belonged to D are not in V'; in this case, its contribution to $\mathbb{E}(G, D, \mathtt{M})$ in (1) will be equal to $p_i \prod_{v_j \in \Gamma(v_i) \cap D}(1 - p_j)$. Based upon these remarks, starting from (1) we get:

$$\begin{aligned}
\mathbb{E}(G, D, \mathtt{M}) &= \sum_{V' \subseteq V} \Pr\left[V'\right] \sum_{v_i \in V} 1_{v_i \in D'} = \sum_{v_i \in V} \sum_{V' \subseteq V} \Pr\left[V'\right] 1_{v_i \in D'} \\
&= \sum_{v_i \in D} \sum_{V' \subseteq V} \Pr\left[V'\right] 1_{v_i \in V'} + \sum_{v_i \notin D} \sum_{V' \subseteq V} \Pr\left[V'\right] 1_{(v_i \in V') \cap (\Gamma(v_i) \cap (D \cap V') = \emptyset)} \\
&= \sum_{v_i \in D} p_i + \sum_{v_i \notin D} \sum_{V' \subseteq V} \Pr\left[V'\right] 1_{(v_i \cap V' = v_i)} 1_{\Gamma(v_i) \cap (D \cap V') = \emptyset} \\
&= \sum_{v_i \in D} p_i + \sum_{v_i \notin D} p_i \prod_{v_j \in \Gamma(v_i) \cap D} (1 - p_j) \qquad (2)
\end{aligned}$$

It is easy to see that, from (2), $\mathbb{E}(G, D, \mathtt{M})$ can be computed in polynomial time. Also, setting $p_i = p$, i.e., considering that vertices of G have identical probabilities (this is quite natural if we assume identical sensors), one gets:

$$\mathbb{E}(G, D, \mathtt{M}) = p|D| + \sum_{v_i \notin D} p(1 - p)^{|\Gamma(v_i) \cap D|} \qquad (3)$$

We consider $\mathbb{E}(G, D, \mathtt{M})$ *as the objective function of the problem handled here which, by symmetry, we call* PROBABILISTIC MIN DOMINATING SET. *Its goal is to determine a dominating set* D^* *of* G*, called a priori dominating set, that minimizes* $\mathbb{E}$. Obviously, such a solution D^*, has the basic property that, in average, any solution obtained by D^* via $\mathtt{M}$ on any subgraph G' of G is optimal for G'.

In other words, for our wireless sensor network design problem, the a priori solution D^* has the property that, under the modification strategy $\mathtt{M}$ described above, the solution constructed on the surviving network minimizes in average the additional cost needed in order that this network remains operational.

The following results holds for PROBABILISTIC MIN DOMINATING SET.

Proposition 1. PROBABILISTIC MIN DOMINATING SET *is* ***NP****-hard, even in bipartite graphs and inapproximable in polynomial time within ratio* $O(\log n)$.

Proof. The inclusion of PROBABILISTIC MIN DOMINATING SET in **NP** is immediately deduced from (2) that is computable in polynomial time. Also, if in (3) we set $p = 1$, then $\mathbb{E}(G, D, \mathtt{M}) = |D|$, i.e., the a priori solution minimizing $\mathbb{E}$ is a minimum dominating set in G. So, PROBABILISTIC MIN DOMINATING SET, having as particular case MIN DOMINATING SET, inherits all its hardness results either in exact computation [15, 21], or in polynomial approximation [14]. Note the $O(\log n)$ lower bound of [14] mentioned in the statement of the proposition is originally addressed to MIN SET COVER problem. But, by a classical approximability preserving reduction (that preserves both constant ratios and ratios depending on the sizes of the instances [28]), this bound also applies to MIN DOMINATING SET. ∎

As we have already mentioned, MIN DOMINATING SET is polynomial in trees. In Section 3, we explore complexity of PROBABILISTIC MIN DOMINATING SET in those graphs, while in Section 4, we give non-trivial (positive) approximation results.

2 Some words about probabilistic combinatorial optimization

It is hopefully clear that the way PROBABILISTIC MIN DOMINATING SET has been mathematically defined in Section 1 remains valid for every combinatorial optimization (deterministic) problem Π. In this way the probabilistic version PROBABILISTIC Π of Π becomes another deterministic problem Π', the solutions of which have the same feasibility constraints as those of Π but Π' has a different objective function where vertex-probabilities intervene. In this sense, probabilistic combinatorial optimization is very close to what in the last years has been called "one stage optimisation under independent decision models", an area very popular in the stochastic optimization community. Also, as one can see from (2) and from the whole discussion deriving it, D' strongly depends on the modification strategy $\mathtt{M}$ used to adapt the a priori solution D to the present graph G'. So, both $\mathbb{E}(G, D, \mathtt{M})$ and the a priori solution D^* optimizing it, also strongly depend on $\mathtt{M}$. In other words, for a fixed instance I of a deterministic problem Π, *two distinct modification strategies induce two distinct probabilistic problems* having I and Π as common deterministic supports. These two distinct problems may have very different functionals and, consequently, different optimal a priori solutions that induce very different complexities for computing them techniques for handling them.

What are the main mathematical problems dealing with probabilistic consideration of a problem Π in the sense discussed above? One can identify at least three interesting mathematical and computational problems:

1. write the functional down in an analytical closed form and, if possible characterize the optimal a priori solution;

2. if such an expression of the functional is possible, prove that its value is polynomially computable (this amounts to proving that the decision version of the modified problem Π' belongs to **NP**); this is neither trivial nor senseless (see, for example, [13, 23, 33]) if one considers that the summation for the functional includes, in a graph of order n, 2^n terms, one for each subgraph of G; so, polynomiality of the computation of the functional is, in general, not immediate;

3. determine the complexity of the computation of the optimal *a priori* solution, i.e., of the solution optimizing the functional (in other words, determine the computational complexity of Π') and if Π' is **NP**-hard, study approximation issues, and/or complexity of Π' in special cases where Π is polynomial.

The framework of *probabilistic combinatorial optimization* that we adopt in this paper was introduced by [16, 3]. In [1, 3, 4, 5, 6, 16, 17, 18, 19], restricted versions of routing and network-design probabilistic minimization problems (in complete graphs) have been studied under the robustness model dealt here (called *a priori optimization*). In [2, 7, 8, 11, 12], the analysis of the probabilistic minimum travelling salesman problem, originally performed in [3, 16], has been revisited. Several other combinatorial problems have been also handled in the probabilistic combinatorial optimization framework, including minimum coloring ([27, 10]), maximum independent set and minimum vertex cover ([25, 26]), longest path ([24]), Steiner tree problems ([29, 30]), minimum spanning tree [5, 9].

Revisit PROBABILISTIC MIN DOMINATING SET. Unfortunately, even if this functional is written in an analytical closed form satisfying at half point 1 above, this form does not derive a compact combinatorial characterization for the optimal a priori solution of PROBABILISTIC MIN DOMINATING SET. In particular, the form of the functional does not imply solution, for instance, of some well-defined weighted version of the (deterministic) MIN DOMINATING SET. This is due to the second term of (2). There, the "costs" assigned to the edges depend on the structure of the anticipatory solution chosen and of the present subgraph of G.

3 Probabilistic dominating set in paths, cycles and trees

We handle in this section PROBABILISTIC DOMINATING SET in paths, cycles and trees. Let us recall that MIN DOMINATING SET in these graphs is polynomial. We prove that PROBABILISTIC DOMINATING SET in paths and cycles remains polynomial for any vertex-probability, while PROBABILISTIC MIN DOMINATING SET in trees is plynomial either when the maximum degree of the input tree is bounded, or when the vertex-probabilities are all equal.

Recall that the *contribution* of a node refers to the probability for this node to be present in D' for a given a priori solution D and for the modification strategy `M` adopted in Section 1. Let us recall that the contribution of a node v_i that belongs to D is p_i, and for a node that does not belong to D, its contribution amounts to $p_i \prod_{v_j \in \Gamma(v_i) \cap D}(1 - p_j)$. This notion will be extended to a set: the *contribution* $C(V')$ of a node-set V' is the expected number of nodes of V' in D'. In other words, $C(V')$ is the sum of the contributions of all the nodes in V'.

3.1 Paths and cycles

Given a path we consider its nodes labeled in the following way: the leftmost endpoint of the path is labelled by v_1 (which might be referred as the first, or left end node), while the rightmost endpoint will be labelled by v_n (last or right end node). Of course, all nodes in between will be labelled in increasing order from left to right.

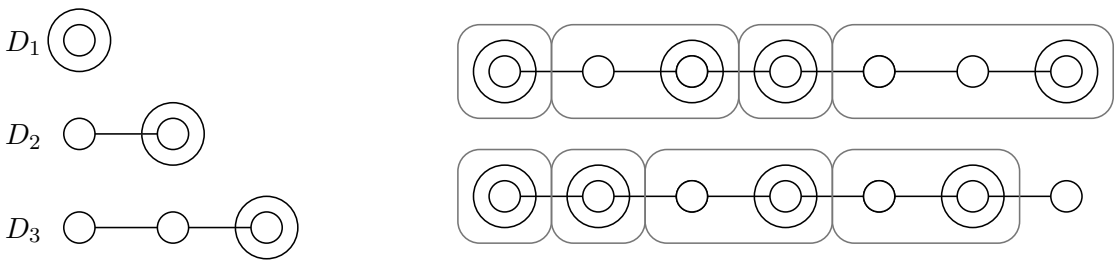
Proposition 2. PROBABILISTIC DOMINATING SET *in paths can be solved in polynomial time.*

Proof. We show how PROBABILISTIC DOMINATING SET in paths can be solved by dynamic programming. First, let us make some preliminary remarks that will help us to build the final algorithm.

Remark 1. For a given node v_i in a dominating set, the "next" dominating node (if any) will be v_{i+1}, v_{i+2} or v_{i+3}. Indeed, if none of them is in the dominating set, then v_{i+2} wouldn't be dominated. ∎

Remark 2. The last dominating node is either v_n or v_{n-1}. ∎

Considering Remarks 1 and 2, one can see that, except the very last node of the path, any dominating set D on a path can be partitioned in elementary patterns of shapes D_1, D_2 or D_3, which are illustrated in Figure 4.



(a) Elementary patterns

(b) Examples of partitions

Figure 4: Partitionning nodes with elementary patterns.

One also can see that any sequence of patterns where the first pattern is not D_3, forms a dominating set D *where the last node (i.e., v_n) is necessarily in* D.

At this point, it is interesting to notice that the contribution of a given pattern can be determined, regardless the rest of the sequence, but depending on its position in the path. To be more precise, denoting by $D_j(i)$ the pattern D_j with its dominating node being v_i:

$$\begin{aligned}
C\left(D_1(i)\right) &= p_i && \forall i && \\
C\left(D_2(i)\right) &= p_i + p_{i-1}\left(1-p_i\right) && i=2 && (4a)\\
C\left(D_2(i)\right) &= p_i + p_{i-1}\left(1-p_i\right)\left(1-p_{i-2}\right) && i \geqslant 3 && (4b)\\
C\left(D_3(i)\right) &= p_i + p_{i-1}\left(1-p_i\right) + p_{i-2}\left(1-p_{i-3}\right) && i \geqslant 4 && (4c)
\end{aligned}$$

Indeed, a pattern D_1 has clearly a contribution of p_i, namely, the probability of its only node, which is a dominating one. The contribution of $D_2(i)$ depends on its position: either $i=2$, then its last node is the second one of the path. Its first node v_1 (which is also the first node of the path) is only dominated by v_2; this amounts to (4a). For any other i, it is clear that v_{i-1} is dominated by both v_i and v_{i-2}, given the way a sequence is built: any sequence ends with a dominating node, so that v_{i-2} has to be a dominating node. This leads to (4b). For the same reason, in a sequence $D_3(i)$ (which can only appear with $i \geqslant 4$), v_{i-1} is only dominated by v_i, and v_{i-2} is only dominated by v_{i-3}, which leads to (4c).

This will be very useful for the method. Indeed, if a pattern has no impact in terms of contribution neither on the preceding sequence, nor on the following one, then it is quite easy to give a recursive definition of W_i which is the sequence up to node v_i of minimum contribution, where v_i is a dominating node:

$$C\left(W_i\right) = \begin{cases} 0 & i=0 \\ p_1 & i=1 \\ \min\limits_{j=1,2}\left(C\left(W_{i-j}\right) + C\left(D_j(i)\right)\right) & i=2,3 \\ \min\limits_{j=1,2,3}\left(C\left(W_{i-j}\right) + C\left(D_j(i)\right)\right) & 4 \leqslant i \leqslant n \end{cases} \quad (5)$$

Indeed, a sequence W_i always ends with a dominating node. So, leaving aside particular cases when $i \leqslant 3$, it ends necessarily with one of the three patterns defined earlier. If it ends with D_1 (resp., D_2 or D_3) then it should be completed with a minimal sequence up to node v_{i-1} (resp., v_{i-2} or v_{i-3}), namely W_{i-1} (resp., W_{i-2} or W_{i-3}). That among these three possibilities that returns the minimal contribution, is that chosen for W_i.

At this point, we only have a set of optimal sequences, but these are not necessarily optimal dominating sets. Indeed, a dominating set can end with a dominated (i.e., non-dominating) node, whereas a sequence cannot. To take this distinction into account, and allow our final solution to end with a dominated node, we define the contribution of an optimal dominating set D^* as follows:

$$C\left(D^*\right) = \min\left(C\left(W_n\right), C\left(W_{n-1}\right) + p_n\left(1-p_{n-1}\right)\right)$$

In all, $3n$ contributions $C(D_j(i))$ have to be computed and, in order to compute a W_i, one basically compares 3 possible values, which amounts to $6n$ operations to get an optimal anticipatory solution for PROBABILISTIC DOMINATING SET in paths. This concludes the proof. ∎

By adapting the method, one can extend the result of Proposition 2 to cycles. In what follows, we prove the following result.

Proposition 3. PROBABILISTIC DOMINATING SET *in cycles can be solved in polynomial time.*

Proof. To extend the result of Proposition 2, it suffices to choose an arbitrary node as starting point v_1 (whose neighbors will be v_2 and v_n), and to notice that there are three possible positions regarding the "first" (i.e. the lowest-labelled) dominating node, namely v_1 itself, v_2 or v_3. Depending on the position of the first dominating node, there are different possibilities for the position of the "last" (e.g. highest labelled) one:

1. the first is v_1 and the last v_n, or v_{n-1}, or v_{n-2};

2. the first is v_2, the last v_n or v_{n-1};

3. the first is v_3, the last is necessarily v_n (or else v_1 is not dominated).

So, we have a total of 6 cases for the position of the "first" and "last" dominating nodes. These 6 cases are illustrated in Figure 3.1, and will be compared by the method.

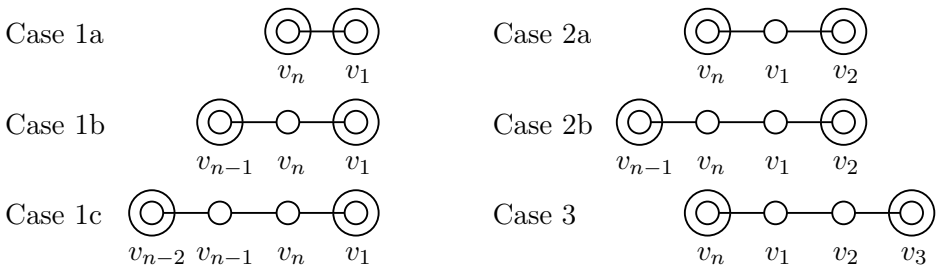


Figure 5: 6 cases regarding positions of the first and last dominating nodes.

In order to take all these cases into account, we will have to define three kinds of partial sequences W_i,W_i' and W_i'' which represent,respectively, the three cases defined earlier. More precisely, W_i represents the optimal sequence ending with node v_i and starting with pattern D_1, W_i' corresponds to ending with node v_i but starts with D_2 and W_i'' does the same but starts with D_3. The contributions of these three sequences are as follows:

$$C\left(W_i\right) = \begin{cases} p_1 & i = 1 \\ p_1 + p_2 & i = 2 \\ \min\limits_{j=1,2} \left(C\left(W_{i-j}\right) + C\left(D_j(i)\right)\right) & i = 3 \\ \min\limits_{j=1,2,3} \left(C\left(W_{i-j}\right) + C\left(D_j(i)\right)\right) & i \geqslant 4 \end{cases}$$

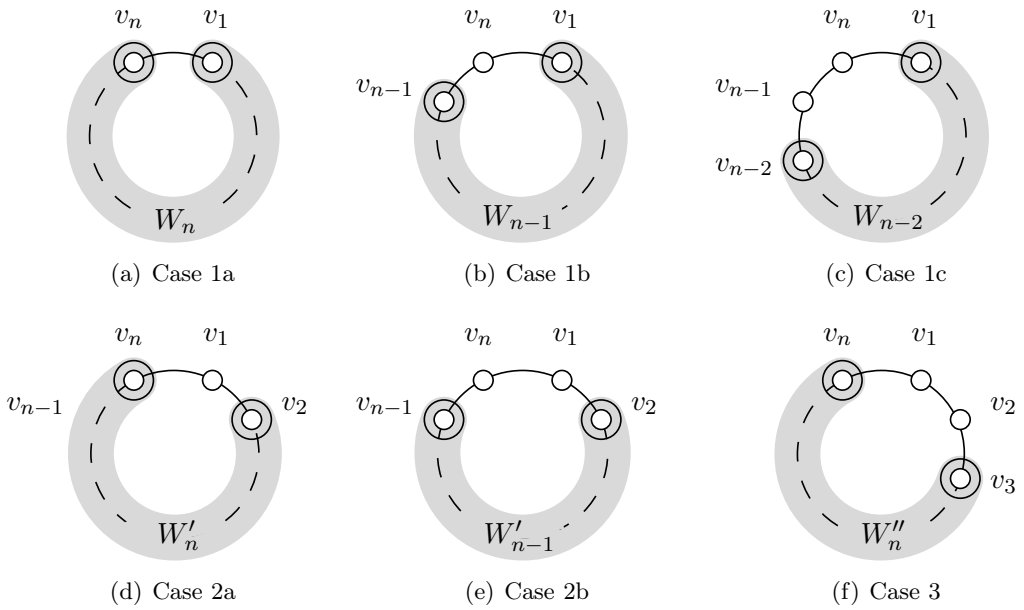
This definition differs from that defined in (5) only on case $i = 2$, to ensure that v_1 will remain a dominating node anyway. For the same reason, we define $C(W_i')$ and $C(W_i'')$ as follows:

$$C\left(W_i'\right) = \begin{cases} p_1\left(1 - p_2\right) + p_2 & i = 2 \\ p_1\left(1 - p_2\right) + p_2 + p_3 & i = 3 \\ \min\limits_{j=1,2} \left(C\left(W_{i-j}'\right) + C\left(D_j(i)\right)\right) & i = 4 \\ \min\limits_{j=1,2,3} \left(C\left(W_{i-j}'\right) + C\left(D_j(i)\right)\right) & i \geqslant 5 \end{cases} \tag{6}$$

$$C\left(W_i''\right) = \begin{cases} p_1\left(1 - p_n\right) + p_2\left(1 - p_3\right) + p_3 & i = 3 \\ p_1\left(1 - p_n\right) + p_2\left(1 - p_3\right) + p_3 + p_4 & i = 4 \\ \min\limits_{j=1,2} \left(C\left(W_{i-j}''\right) + C\left(D_j(i)\right)\right) & i = 5 \\ \min\limits_{j=1,2,3} \left(C\left(W_{i-j}''\right) + C\left(D_j(i)\right)\right) & i \geqslant 6 \end{cases} \tag{7}$$

In (7), we consider that v_1 has a contribution of $p_1(1 - p_n)$. Indeed, it is not dominated in any partial sequence W_i'', but will have to be dominated by v_n in our final solution, so that its contribution has to be $p_1(1 - p_n)$ anyway.

In (6), we consider that v_1 has a contribution of $p_1(1 - p_2)$. Actually, this represents its contribution in the partial sequence but, in the end, its contribution might be decreased in



(a) Case 1a (b) Case 1b (c) Case 1c

(d) Case 2a (e) Case 2b (f) Case 3

Figure 6: Overall structures on the 6 cases.

a dominating set that includes v_n (Case 2a). Precisely, the contribution would decrease from $p_1(1-p_2)$ downto $p_1(1-p_2)(1-p_n)$. Thus a decrease of $p_n \cdot p_1(1-p_2)$.

Now, using these sequences, in order to build an optimal dominating set, one defines the optimal solution D^* as the minimal dominating set over the 6 cases defined earlier, whose structure is shown in Figure 6 and whose contributions amount to:

$$C(D^*) = \min \left(\begin{array}{cc} W_n & \text{(Case 1a)} \\ W_{n-1} + p_n(1-p_{n-1})(1-p_1) & \text{(Case 1b)} \\ W_{n-2} + p_{n-1}(1-p_{n-2}) + p_n(1-p_1) & \text{(Case 1c)} \\ W'_n - p_n.p_1(1-p_2) & \text{(Case 2a)} \\ W'_{n-1} + p_n(1-p_{n-1}) & \text{(Case 2b)} \\ W''_n & \text{(Case 3)} \end{array} \right)$$

Here, since three types of sequences are built, the number of operations required to compute an optimal solution is $12n$ which concludes the proof. ∎

3.2 Trees

With a similar, yet more complex method, one can generalize the result of Proposition 2 from paths (trees of maximum degree bounded by 2) to trees with bounded maximum degrees. In what follows, v_1 will denote the root of the input tree T, T_i will denote the subtree rooted at node v_i (so, $T_1 = T$), F_i will denote the set of chilren of given node v_i, and v_{f_i} the father of v_i. The depth $D(T_i)$ of a subtree T_i will refer to the minimum number of edges on a path between the root and a leaf of the subtree.

Proposition 4. PROBABILISTIC MIN DOMINATING SET *in trees of maximum degree bounded by k can be solved in $O^*(2^k)$, where $O^*(\cdot)$ notation ignores the polynomial terms.*

Proof. Consider a tree T and a dominating set D on T. For any subtree T_i of T, there are only three possible configurations regarding its root v_i:

1. $v_i \in D$;
2. $v_i \notin D$, and $v_{f_i} \notin D$; in this case, the root has to be dominated in the subtree T_i (i.e., dominated by at least one of its children);
3. $v_i \notin D$ and $v_{fi} \in D$; in this case, the root might be non-dominated in the subtree T_i (i.e., not dominated by any of its children).

Considering Cases 1 to 3, for a given dominating set D in a tree T, it is always possible to partition nodes of T into three sets: D, S and N, where D is the dominating set itself, S is the set of nodes that are dominated, but not by their fathers, and N is the set of nodes dominated by their fathers (and also possibly by some of their children). Figure 3.2 gives an example of such a partition.

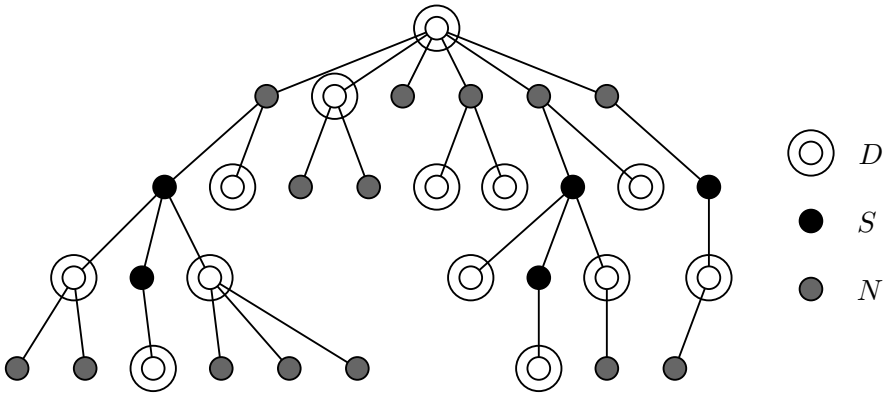


Figure 7: Partitioning nodes in three subsets.

Now, let us analyze what possible cases can occur for nodes of F_i with respect to the status of v_i. The following cases can occur:

- $v_i \in D$; then children of v_i (if any) might be in D, or in N;
- $v_i \in S$; then children of v_i might be in D or in S, and at least one of them should be in D;
- $v_i \in N$; in this case, children of v_i might be in D or in S.

It is interesting to notice that the situation of a given node impacts only its own contribution if the status of its children is known, so that, once more, it is possible to run a dynamic programming method. It will be based upon three partial solutions for each subtree T_i, namely, W_i, W_i' and W_i'', where W_i (resp., W_i', W_i'') is the partial solution of minimum contribution for subtree T_i rooted at v_i when $v_i \in D$ (resp., $v_i \in S$, $v_i \in N$).

Partial solution W_i can be computed on a tree of any depth. Children of v_i can belong either to D, or to N, so that there are $2^{|F_i|}$ possible combinations to evaluate (bounded by 2^k). The combination minimizing the overall contribution leads to the structure we are trying to define, namely W_i. Setting $D_i = D \cap F_i$ (the subset of children of v_i that are dominating nodes), W_i is defined by:

$$C(W_i) = \min_{D_i \subseteq F_i} \left(p_i + \sum_{v_j \in D_i} C(W_j) + \sum_{v_j \in F_i \setminus D_i} C\left(W_j''\right) \right) \tag{8}$$

This value is quite easy to initialize with leaves where $W_i = p_i$.

Now, let us tackle W_i'', which can also be initialized on leaves (unlike W_i'). Indeed, if a leaf is dominated, then its father has to be dominating, or else it would not be dominated. So that a leaf can only be in D or in N. As we said earlier, children of v_i in a partial solution W_i'' can belong either to D or to S. Thus, the following holds:

$$C\left(W_i''\right) = \min_{D_i \subseteq F_i} \left(p_i \left(1 - p_{f_i}\right) \prod_{v_j \in D_i} \left(1 - p_j\right) + \sum_{v_j \in D_i} C\left(W_j\right) + \sum_{v_j \in F_i \setminus D_i} C\left(W_j'\right) \right) \tag{9}$$

Note there are also at most 2^k combinations to examine in this case.

Finally, let us specify W_i'. In this case, children of v_i should be in S or in D and at least one of them should be in D. Of course, by definition, none of them can be in N. Once more, each combination of sons in S or in D (at most 2^k combinations) leads to a specific contribution for the subtree T_i. The partial solution W_i' is the one minimizing this contribution. Thus, the following holds:

$$C\left(W_i'\right) = \min_{D_i \subseteq F_i, |D_i| \geqslant 1} \left(p_i \prod_{v_j \in D_i} \left(1 - p_j\right) + \sum_{v_j \in D_i} C\left(W_j\right) + \sum_{v_j \in F_i \setminus D_i} C\left(W_j'\right) \right) \tag{10}$$

Note that such a value can be computed for any subtree T_i of depth at least 1 (not on leaves). This might be a problem when computing a value W_i' or W_i'' on a tree T_i of depth 1 since, according to (9) and (10), in order to compute W_i' and W_i'', one needs values W_j' for all the children v_j of v_i but these values do not exist for leaves. To keep all formulæ valid and still ensure that a leaf will never be in S, we will initialize W_j' to an arbitrarily large value M for any v_j that is a leaf. Thus, when applying (9) and (10), all leaf-children of v_i will be forced to be in D_i.

Note also, that definitions of $C(W_i')$ and $C(W_i'')$ differ only by a factor $(1 - p_{f_i})$ in the contribution of v_i.

The dynamic programming method runs in a "bottom up" way. It is initialized with leaves where values W_i, W_i' and W_i'' are equal to p_i , M and $p_i(1 - p_{f_i})$, respectively. Then, each subtree T_i is associated with structures W_i, W_i' and W_i'' (apart from leaves that are associated only with W_i and W_i'', and the overall tree which will be associated only with W_i and W_i' for obvious reasons), whose computation relies on the same structures on subtrees induced by children of the root v_i. In other words, in order to compute, for instance, a value W_i, one needs to have already computed values W_j and W_j'' for all children v_j of v_i. Since the method is easily initialized with leaves, all values can be computed for all subtrees, starting from leaves and ending with the whole tree.

Finally, the optimum D^* is given by $D^* = \text{argmin}_{W=W_i,W_i'}\left(C(W)\right)$. In all, $O(2^k n) = O^*(2^k)$ operations are necessary to compute D^*. ∎

Corollary 1. PROBABILISTIC MIN DOMINATING SET *in trees with maximum degree bounded by* $O(\log n)$ *can be solved in polynomial time.*

Corollary 2. PROBABILISTIC MIN DOMINATING SET *in trees is fixed parameter tractable with respect to parameter "maximum degree"*

As one can see from the proof of Proposition 4 the bound 2^k for evaluating $C(W_i)$ and its gang was somewhat greedy and includes the real complexity of this evaluation that depends on which of the children of v_i are in D, the quantity of them depending on their probabilities that

affect $C(v_i)$. So, the real complexity of the method presented is exponential to the number of distinct vertex probabilities of the children of v_i which can be as large as k. In what follows, we conclude this section by restricting ourselves to the natural case where all the vertices of the tree have the same probability.

Proposition 5 . PROBABILISTIC MIN DOMINATING SET *is polynomial in general trees with equiprobable nodes.*

Proof. We will use the same kind of recursion as in Proposition 4, but we will adapt slightly its definition to take advantage of the equiprobability property. What is interesting and useful with this property, is that the contribution of a given dominated node does not depend on *which* of its neighbors are dominating ones, but only on *how many* of them are dominating. Indeed, if $F_i \cup v_{f_i}$ represents the set of neighbors of a dominated node v_i ($i \neq 1$), then:

$$C\left(v_i\right) = p(1-p)^{\left|\left(F_i \cup \{v_{f_i}\}\right) \cap D\right|}$$

Now, let us specify the dynamic programming method. The idea is basically the same as in Proposition 4, but given the equiprobability assumption, (8), (9) and (10) can be computed in $O(k \log k)$ instead of $O(2^k)$, which enables us to apply the method to trees of any structure, and any degree.

If one considers that a given node has h children in D ($0 \leqslant h \leqslant k$), then one does not have to check all the possible configurations (at most $\binom{k}{h}$) for the h children in D. Indeed, the contribution of v_i does not depend on which children are in D, but only on how many of them, namely h. Thus the following holds:

$$C\left(v_i\right) = \begin{cases} p & v_i \in D \\ p(1-p)^h & v_i \in S, (h \geqslant 1) \\ p(1-p)^{h+1} & v_i \in N \end{cases} \tag{11}$$

Now, we will use this, to simplify computations of W_i, W_i' and W_i''. We will detail the simplification for W_i' only, and simply present the final results for W_i and W_i'', but the ideas and methods are exactly the same as in the proof of Proposition 4.

Let us denote by $W'_{i,h}$ the partial solution W_i' where the number $|D_i|$ of children of v_i in D is equal to h. Of course, if one computes $W'_{i,h}$ for each possible h between 1 and $|F_i|$, it is quite easy to compute W_i' since $C(W_i') = \min_{1 \leqslant h \leqslant |F_i|}(W'_{i,h})$. We will show that it is possible to generate $W'_{i,h}$ for each possible h in time $k \log k$.

Combining (10) and (11), one gets:

$$\begin{aligned} C\left(W'_{i,h}\right) &= \min_{D_i \subset F_i, |D_i| = h} \left(p(1-p)^h + \sum_{j \in D} C\left(W_j\right) + \sum_{j \in F_i \setminus D} C(W_j') \right) \\ &= p(1-p)^h + \min_{D_i \subset F_i, |D_i| = h} \left(\sum_{j \in D} C\left(W_j\right) + \sum_{j \in F_i \setminus D} C(W_j') \right) \\ &= p(1-p)^h + \sum_{j \in F_i} C\left(W_j'\right) + \min_{D_i \subset F_i, |D_i| = h} \left(\sum_{j \in D} C\left(W_j\right) - C\left(W_j'\right) \right) \\ &= p(1-p)^h + \sum_{j \in F_i} C\left(W_j'\right) + \min_{D_i \subset F_i, |D_i| = h} \left(\sum_{j \in D} \Delta_j' \right) \end{aligned} \tag{12}$$

where $\Delta_j' = C(W_j) - C(W_j')$ for all j. This value represents the "additional cost" for a given subtree T_j to have its root v_j as dominating node instead of dominated one. Indeed, if the root

of the subtree is dominating then it must have the structure W_j and weight $C(W_j)$, whereas if the root is dominated then it must have the structure W'_j and weight $C(W'_j)$. Obviously, this value can be negative if the partial solution W_j (with the node v_j considered as dominating) has a better expected value than the partial solution W'_j (with v_j considered dominated).

Understanding this, one also understands that the min operator can be computed in $O(k \log k)$ for every h. Indeed, it suffices to compute the values Δ'_j for all j, and to sort them in increasing order (which takes $O(k \log k)$ operations).

Then, in order to compute the min for a given h, simply pick as subset D the h nodes v_j corresponding to the h first values Δ'_j in the sorted list. Furthermore, one needs to sort Δ'_j's only once to get any $W'_{i,h}$ for all possible h. Indeed, once sorted, one can use the sorted list to compute all $W'_{i,h}$: just pick the lowest Δ'_j (thus the first element of the sorted list) to compute $W'_{i,1}$, the two lowest (first and second elements of this same sorted list) to compute $W'_{i,2}$, and so on, for each h.

This method returns the proper min as well as its associated structure. Indeed, if one wants to minimize the sum of h values in a set of k values, one only needs to pick the h lowest values. In this case, the h subtrees with lowest "additional costs" (linked to setting the root as dominating instead of as dominated) will be the h that actually have their respective roots as dominating nodes, the $|F_i| - h$ other subtrees having their roots set as dominated nodes. Obviously, given the recursive structure of the method, one knows the optimal partial solutions associated with all these subtrees.

By the same method, and setting $\Delta''_j = C(W_j) - C(W''_j)$, one gets:

$$C\left(W_{i,h}\right) = p + \sum_{j \in F_i} C\left(W''_j\right) + \min_{D_i \subset F_i, |D_i| = h} \left(\sum_{j \in D} \Delta''_j \right) \tag{13}$$

$$C\left(W''_{i,h}\right) = p(1-p)^{h+1} + \sum_{j \in F_i} C\left(W'_j\right) + \min_{D_i \subset F_i, |D_i| = h} \left(\sum_{j \in D} \Delta'_j \right) \tag{14}$$

Each of these values can be computed for all h in $O(k \log k)$ in exactly the same way as described previously. Using these values, one directly defines W_i and W''_i as follows:

$$C\left(W_i\right) = \min_{0 \leqslant h \leqslant |F_i|} \left(W_{i,h}\right)$$

$$C\left(W''_i\right) = \min_{0 \leqslant h \leqslant |F_i|} \left(W''_{i,h}\right)$$

The rest of the algorithm is the same as that described in Proposition 4. In all, each value W_i, W'_i or W''_i takes $O(k \log k)$ operations to be computed, and the method computes $3n$ of them. So, the overall running time of the method is $O(n^2 \log n)$, which concludes the proof. ∎

In order to make clearer how each W_i and gang are computed, in appendix we run the algorithm on a small instance, and detail some interesting steps.

4 Polynomial approximation of PROBABILISTIC MIN DOMINATING SET

4.1 Networks with identical sensors

We consider in this section that the probability that a sensor fails is the same for all of them.

As we have already mentioned, MIN DOMINATING SET is approximate equivalent to MIN SET COVER, in the sense that an approximation algorithm for one of them can be transformed in polynomial time into an approximation algorithm for the other one achieving the same approximation ratio. Recall also that the natural greedy algorithm for MIN SET COVER achieves approximation

ratio either $1+\ln|S_{\max}|$ where $|S_{\max}|$ is the cardinality of the largest set $S_{\max}$ in the set-system describing the instance of MIN SET COVER [20, 22], either $O(\log n)$ where n is the cardinality of the ground set describing this instance [32]. In the transformation of MIN DOMINATING SET into MIN SET COVER, any set S of the set-system becomes a vertex with degree $|S|+1$. So, in the derived instance of MIN DOMINATING SET, $\Delta = |S_{\max}|+1$, where Δ denotes the maximum degree of the derived graph. For facility and because of the form of the functional given in (3), we will use in what follows the former of the above ratios.

The following two easy lemmata that will be used later hold. We prove the first of them (Lemma 1). The proof of the second one (Lemma 2) is immediate.

Lemma 1. *For any instance of* MIN DOMINATING SET *of size n and maximum degree Δ, any minimal (for inclusion) solution (hence, the minimum-size one also) has size bounded below by $n/(\Delta+1)$.*

Proof. Consider an instance G of MIN DOMINATING SET of size n and maximum degree Δ, and a minimal (for inclusion) MIN DOMINATING SET-solution D in G. Set $\bar{D} = V \setminus D$ and, for every vertex $v \in D$, denote by $d_{\bar{D}}(v)$ the number of the neighbours of v in $\bar{D}$. Since, D dominates V, we have $\sum_{v\in D} d_{\bar{D}}(v) \geqslant \left|\bar{D}\right| = n - |D|$

On the other hand, obviously, for every $v \in D$, $d_{\bar{D}}(v) \leqslant \Delta$, so that the former sum becomes $\Delta|D| \geqslant n - |D| \Longrightarrow |D| \geqslant n/(\Delta+1)$ as claimed. ∎

Lemma 2. *Let D be a minimal dominating set in a graph $G(V,E)$. Then, $V \setminus D$ is also a dominating set and, moreover, the smallest of them is smaller than $n/2$.*

Revisit now (3) and observe that the following framing can be easily derived for $\mathbb{E}(G, D, \mathtt{M})$:

$$\mathbb{E}(G,D,\mathtt{M}) \leqslant p|D| + (n-|D|)p(1-p) = p^2|D| + pn(1-p) \quad (15)$$

$$\mathbb{E}(G,D,\mathtt{M}) \geqslant p|D| + p(n-|D|)(1-p)^{\Delta} = p\left(1-(1-p)^{\Delta}\right)|D| + pn(1-p)^{\Delta} \quad (16)$$

Denote by D^* and $\hat{D}$ an optimal solution for PROBABILISTIC MIN DOMINATING SET and for MIN DOMINATING SET in G, respectively. Remark that since D^* is a feasible solution for MIN DOMINATING SET, we have:

$$|D^*| \geqslant \left|\hat{D}\right| \quad (17)$$

Take D the smallest between the $(1+\ln\Delta)$-approximate solution computed by the greedy algorithm in G and the complement of it with respect to V as a priori solution for MIN DOMINATING SET. Remark that, so computed, the size of D guarantees the ratio $1+\ln\Delta$ and, simultaneously, according to Lemma 2, it is smaller than $n/2$. Applying the rightmost bound of (15) to D and the rightmost bound of (16) to D^*, we immediately get:

$$\mathbb{E}(G,D,\mathtt{M}) \leqslant p^2|D| + pn(1-p) \leqslant p^2(1+\ln\Delta)\left|\hat{D}\right| + p(1-p)n \quad (18)$$

$$\mathbb{E}\left(G,D^*,\mathtt{M}\right) \geqslant p\left(1-(1-p)^{\Delta}\right)|D^*| + pn(1-p)^{\Delta} \quad (19)$$

Combining (18) and (19), we have the following for the approximation ratio of D, first taking into account that it achieves approximation ratio $1+\ln\Delta$ (21), then taking into account (17) (in the first inequality of (22)), then that $n \geqslant |D^*|$ (second inequality of (22)), and finally using

Lemma 1 (in the first inequality of (23)):

$$\begin{aligned}
\frac{\mathbb{E}(G,D,\mathtt{M})}{\mathbb{E}\,(G,D^*,\mathtt{M})} &\leqslant \frac{p|D|+n(1-p)}{(1-(1-p)^{\Delta})\,|D^*|+n(1-p)^{\Delta}} && (20)\\
&\leqslant \frac{p(1+\ln\Delta)\left|\hat{D}\right|+n(1-p)}{(1-(1-p)^{\Delta})\,|D^*|+n(1-p)^{\Delta}} && (21)\\
&\leqslant \frac{p(1+\ln\Delta)\,|D^*|+n(1-p)}{(1-(1-p)^{\Delta})\,|D^*|+n(1-p)^{\Delta}} \leqslant \frac{p(1+\ln\Delta)\,|D^*|+n(1-p)}{|D^*|} && (22)\\
&\leqslant p(1+\ln\Delta)+(1-p)(\Delta+1) = \Delta+1-p(\Delta-\ln\Delta) && (23)
\end{aligned}$$

Revisit (20) and use Lemma 2 for D and Lemma 1 for D^*. Then, some very easy algebra gives:

$$\frac{\mathbb{E}(G,D,\mathtt{M})}{\mathbb{E}\,(G,D^*,\mathtt{M})} \leqslant \frac{(\Delta+1)\left(1-\frac{p}{2}\right)}{1+\Delta(1-p)^{\Delta}} \leqslant \frac{\Delta+1}{1+\Delta(1-p)^{\Delta}} \tag{24}$$

Ratio's expression in (23) is bounded above by $\Delta-\ln\Delta$ for $p \geqslant (\ln\Delta+1)/(\Delta+1-\ln\Delta) = \ln(e\Delta)/(\Delta+1-\ln\Delta)$. On the other hand, if $p \leqslant \ln(e\Delta)/(\Delta+1-\ln\Delta)$, expression $\Delta(1-p)^{\Delta}$ in the denominator of (24) is $\Delta(1-p)^{\Delta} \sim 1+e^{-1} = 1.37$ and the ratio given by (24) becomes $(\Delta+1)/1.37$.

Discussion above, leads so to the following approximation result for PROBABILISTIC MIN DOMINATING SET.

Proposition 6. *The set D selected to be the smallest between the $(1+\ln\Delta)$-approximate solution computed by the greedy algorithm in G and the complement of it with respect to V achieves for* PROBABILISTIC MIN DOMINATING SET *approximation ratio bounded above by $\Delta-\ln\Delta$, where Δ is the maximum degree of G.*

The mathematical analysis leading to the result of Proposition 6 seems to be quite tight. Indeed, a numerical study of the ratio in the first inequality of (24) with respect to p derives that function $f(p) = (\Delta+1)(1-(p/2))/(1+\Delta(1-p)^{\Delta})$ attains its maximum value for $p = 2\ln\Delta/\Delta$. In this case, after an easy algebra, the maximum is $\Delta-\ln\Delta$.

4.2 Networks with heterogeneous sensors

Let us now suppose that the sensors of the network are heterogeneous so that each of them has its own failure probability that can be different from that of another sensor in the network. We will show that, in this case, any PROBABILISTIC MIN DOMINATING SET-solution achieves approximation ratio bounded above by $O(\Delta^2/\log\Delta)$.

Revisit (2) and observe that any dominating set D satisfies the trivial inequality:

$$\mathbb{E}(G,D,\mathtt{M}) \leqslant \sum_{v_i\in V} p_i \tag{25}$$

Fix an optimal a priori dominating set D^* and, for a probability p' that will be fixed later, partition the vertices of G into four subsets:

D_1^*: the set of vertices of D^* whose probabilities are at least p'; furthermore, set $|D_1^*| = \kappa$;

D_2^*: the rest of vertices of D^*, i.e., $D_2^* = D^* \setminus D_1^*$;

$\bar{D}_1^*$: the set $\Gamma_{\bar{D}^*}(D_1^*)$ of neighbours of D_1^* in $\bar{D}^* = V \setminus D^*$, i.e., $\bar{D}_1^* = \Gamma(D_1^*) \cap \bar{D}^*$;

$\bar{D}_2^*$: the set $\Gamma_{\bar{D}^*}(D_2^*) \setminus \bar{D}_1^*$, i.e., the vertices of $\bar{D}^*$ that have no neighbours in D_1^*, i.e., $\bar{D}_2^* = \bar{D}^* \setminus \Gamma_{\bar{D}^*}(D_1^*)$.

Denote, for simplicity, by p the largest vertex-probability. Now, taking into account the partition above (2) can be rewritten as:

$$\begin{aligned}
\mathbb{E}\left(G, D^*, \mathtt{M}\right) &= \sum_{v_i \in D_1^*} p_i + \sum_{v_i \in D_2^*} p_i + \sum_{v_i \in \bar{D}_1^*} p_i \prod_{v_j \in \Gamma(v_i) \cap D^*} (1 - p_j) + \sum_{v_i \in \bar{D}_2^*} p_i \prod_{v_j \in \Gamma(v_i) \cap D_2^*} (1 - p_j) \\
&\geqslant \kappa p' + \sum_{v_i \in D_2^*} p_i + (1-p)^{\Delta} \sum_{v_i \in \bar{D}_1^*} p_i + \left(1 - p'\right)^{\Delta} \sum_{v_i \in \bar{D}_2^*} p_i \qquad (26) \\
&\geqslant \sum_{v_i \in D_2^*} p_i + \left(1 - p'\right)^{\Delta} \sum_{v_i \in \bar{D}_2^*} p_i \qquad (27)
\end{aligned}$$

Let us first suppose that $\sum_{v_i \in D_2^* \cup \bar{D}_2^*} p_i \geqslant \sum_{v_i \in V} p_i / x$ for some x to be also defined later. Then, using (25) and (27), it holds that:

$$\begin{aligned}
\frac{\mathbb{E}(G, D, \mathtt{M})}{\mathbb{E}\left(G, D^*, \mathtt{M}\right)} &\leqslant \frac{\sum_{v_i \in V} p_i}{\sum_{v_i \in D_2^*} p_i + (1-p')^{\Delta} \sum_{v_i \in \bar{D}_2^*} p_i} \leqslant \frac{\sum_{v_i \in V} p_i}{(1-p')^{\Delta} \sum_{v_i \in D_2^* \cup \bar{D}_2^*} p_i} \\
&\leqslant \frac{\sum_{v_i \in V} p_i}{(1-p')^{\Delta} \frac{\sum_{v_i \in V} p_i}{x}} \leqslant \frac{x}{(1-p')^{\Delta}} \qquad (28)
\end{aligned}$$

Suppose now that $\sum_{v_i \in D_2^* \cup \bar{D}_2^*} p_i \leqslant \sum_{v_i \in V} p_i / x$. Then, for $\sum_{v_i \in D_1^*} p_i + \sum_{v_i \in \bar{D}_1^*} p_i$, it holds that:

$$\sum_{v_i \in D_1^*} p_i + \sum_{v_i \in \bar{D}_1^*} p_i \geqslant \left(\frac{x-1}{x}\right) \sum_{v_i \in V} p_i \qquad (29)$$

Remark, furthermore, that, since $|D_1^*| = \kappa$, $|\bar{D}_1^*| \leqslant \Delta\kappa$ and, using (29), the following holds:

$$\kappa\Delta p + \kappa p \geqslant \sum_{v_i \in D_1^*} p_i + \sum_{v_i \in \bar{D}_1^*} p_i \geqslant \frac{x-1}{x} \sum_{v_i \in V} p_i \Longrightarrow \sum_{v_i \in V} p_i \leqslant \frac{x}{x-1} \kappa p(\Delta + 1) \qquad (30)$$

Remark finally that, from (25) and (26), one can immediately derive another easy expression for the approximation ratio, namely, $\mathbb{E}(G, D, \mathtt{M}) / \mathbb{E}(G, D^*, \mathtt{M}) \leqslant \sum_{v_i \in V} p_i / \kappa p'$, since $\mathbb{E}(G, D^*, \mathtt{M}) \geqslant \kappa p'$. This, using (30), becomes:

$$\frac{\mathbb{E}(G, D, \mathtt{M})}{\mathbb{E}\left(G, D^*, \mathtt{M}\right)} \leqslant \frac{xp(\Delta + 1)}{(x-1)p'} \leqslant \frac{x(\Delta + 1)}{(x-1)p'} \qquad (31)$$

If we fix $p' = \ln \Delta / \Delta$ and $x = \Delta / \ln \Delta$, then both ratios given in (28) and (31) become $\approx \Delta^2 / \ln \Delta$ as claimed in the beginning of the section.

5 Conclusion

It seems to us that probabilistic combinatorial optimization is an interesting and appropriate mathematical framework for handling numerous risk management natural problems where, faced to some disaster that has destroyed a part of the solution, whatever this solution represents, the objective of a decision maker is to restore it, or to efficiently compute one having several good properties, for example, having a good quality (under a predefined criterion) minimizing, or maximizing the reconstruction costs.

In this paper we have studied emergency management for a wireless sensor network problem modelled as a probabilistic version of MIN DOMINATING SET. This has led to a problem that is quite more difficult to handle than its original deterministic version, in particular when trying to approximately solve it. We have proposed algorithms for paths, cycles and trees (cases where MIN DOMINATING SET is polynomial), as well as we have tried a first study of approximability of PROBABILISTIC MIN DOMINATING SET is general graphs. What is missing with respect to the latter issue is to propose algorithms that are proper to the probabilistic nature of this problem. This seems to be a hard issue since no compact characterization of the optimal a priori solution can be derived from (2) but work is in progress.

References

[1] I. Averbakh, O. Berman, and D. Simchi-Levi. Probabilistic a priori routing-location problems. *Naval Res. Logistics*, 41:973–989, 1994.

[2] P. Balaprakash, M. Birattari, T. Stützle, and M. Dorigo. Estimation-based metaheuristics for the probabilistic traveling salesman problem. *Computers and Operations Research*, 37(11):1939–1951, 2010.

[3] D. J. Bertsimas. *Probabilistic combinatorial optimization problems*. Phd thesis, Operations Research Center, MIT, Cambridge Mass., USA, 1988.

[4] D. J. Bertsimas. On probabilistic traveling salesman facility location problems. *Transportation Sci.*, 3:184–191, 1989.

[5] D. J. Bertsimas. The probabilistic minimum spanning tree problem. *Networks*, 20:245–275, 1990.

[6] D. J. Bertsimas, P. Jaillet, and A. Odoni. A priori optimization. *Oper. Res.*, 38(6):1019–1033, 1990.

[7] L. Bianchi, J. Knowles, and N. Bowler. Local search for the probabilistic traveling salesman problem: correlation to the 2-p-opt and 1-shift algorithms. *European J. Oper. Res.*, 161(1):206–219, 2005.

[8] M. Birattari, P. Balaprakash, T. Stützle, and M. Dorigo. Estimation-based local search for stochastic combinatorial optimization using delta evaluations: a case study on the probabilistic traveling salesman problem. *INFORMS J. Computing*, 20(4):644–658, 2008.

[9] N. Boria, C. Murat, and V. Th. Paschos. On the PROBABILISTIC MIN SPANNING TREE problem. *J. Mathematical Modelling and Algorithms*. To appear.

[10] N. Bourgeois, F. Della Croce, B. Escoffier, C. Murat, and V. Th. Paschos. Probabilistic coloring of bipartite and split graphs. *J. Comb. Optimization*, 17(3):274–311, 2009.

[11] A. M. Campbell. Aggregation for the probabilistic traveling salesman problem. *Computers and Operations Research*, 33(9):2703–2724, 2006.

[12] A. M. Campbell and B. W. Thomas. Probabilistic traveling salesman problem with deadlines. *Transportation Sci.*, 42(1):1–21, 2008.

[13] A. M. Campbell and B. W. Thomas. Runtime reduction techniques for the probabilistic traveling salesman problem with deadlines. *Computers and Operations Research*, 36(4):1231–1248, 2008.

[14] U. Feige. A threshold of $\ln n$ for approximating set cover. *J. Assoc. Comput. Mach.*, 45:634–652, 1998.

[15] M. R. Garey and D. S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness*. W. H. Freeman, San Francisco, 1979.

[16] P. Jaillet. Probabilistic traveling salesman problem. Technical Report 185, Operations Research Center, MIT, Cambridge Mass., USA, 1985.

[17] P. Jaillet. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. *Oper. Res.*, 36(6):929–936, 1988.

[18] P. Jaillet. Shortest path problems with node failures. *Networks*, 22:589–605, 1992.

[19] P. Jaillet and A. Odoni. The probabilistic vehicle routing problem. In B. L. Golden and A. A. Assad, editors, *Vehicle routing: methods and studies*. North Holland, Amsterdam, 1988.

[20] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. System Sci.*, 9:256–278, 1974.

[21] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of computer computations*, pages 85–103. Plenum Press, New York, 1972.

[22] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete Math.*, 13:383–390, 1975.

[23] A.-R. Mahjoub, V. Markakis, I. Milis, and V. Th. Paschos, editors. *Combinatorial Optimization. Proceedings of the 2nd International Symposium on Combinatorial Optimization, ISCO'12*, volume 7422 of *Lecture Notes in Computer Science*. Springer-Verlag, 2012. To appear.

[24] C. Murat and V. Th. Paschos. The probabilistic longest path problem. *Networks*, 33:207–219, 1999.

[25] C. Murat and V. Th. Paschos. A priori optimization for the probabilistic maximum independent set problem. *Theoret. Comput. Sci.*, 270:561–590, 2002.

[26] C. Murat and V. Th. Paschos. The probabilistic minimum vertex-covering problem. *Int. Trans. Opl Res.*, 9(1):19–32, 2002.

[27] C. Murat and V. Th. Paschos. On the probabilistic minimum coloring and minimum k-coloring. *Discrete Appl. Math.*, 154:564–586, 2006.

[28] V. Th. Paschos. A survey about how optimal solutions to some covering and packing problems can be approximated. *ACM Comput. Surveys*, 29(2):171–209, 1997.

[29] V. Th. Paschos, O. A. Telelis, and V. Zissimopoulos. Steiner forests on stochastic metric graphs. In A. Dress, Y. Xu, and B. Zhu, editors, *Proc. Conference on Combinatorial Optimization and Applications, COCOA'07*, volume 4616 of *Lecture Notes in Computer Science*, pages 112–123. Springer-Verlag, 2007.

[30] V. Th. Paschos, O. A. Telelis, and V. Zissimopoulos. Probabilistic models for the STEINER TREE problem. *Networks*, 56(1):39–49, 2010.

[31] A. C. Sandos, F. Bendali, J. Mailfert, C. Duhamel, and K.-M. Hou. Heuristics for designing energy-efficient wireless sensor network topologies. *J. Networks*, 4(6):436–444, 2009.

[32] P. Slavík. A tight analysis of the greedy algorithm for set cover. In *Proc. STOC'96*, pages 435–441, 1996.

[33] D. Weyland, R. Montemanni, and L. M. Gambardella. Hardness results for the probabilistic traveling salesman problem with deadlines. In Mahjoub et al. [23]. To appear.

An example of the algorithm of Proposition 5

Suppose $p = 0.2$ and consider the tree of Figure 5. The algorithm is initialized with leaves:

$$
\begin{array}{l}
C(W_3) = C(W_5) = C(W_7) = C(W_8) = C(W_9) = 0.2 \\
C(W'_3) = C(W'_5) = C(W'_7) = C(W'_8) = C(W'_9) = M \\
C(W''_3) = C(W''_5) = C(W''_7) = C(W''_8) = C(W''_9) = 0.8 \cdot 0.2 = 0.16 \\
\Delta'_3 = \Delta'_5 = \Delta'_7 = \Delta'_8 = \Delta'_9 = 0.2 - M \\
\Delta''_3 = \Delta''_5 = \Delta''_7 = \Delta''_8 = \Delta''_9 = 0.2 - 0.16 = 0.04
\end{array}
$$

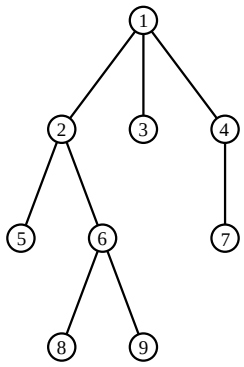


Figure 8: A tree T.

With all leaves initialized, it is possible to find substructures W_i, W'_i and W''_i on subtrees rooted at v_6 and v_4.

Let us start with v_6. Here it is senseless to sort children v_j of v_6 in increasing values of Δ'_j or Δ''_j since they all have the same values Δ'_j and Δ''_j. This means that if, in a given substructure, only one child v_j of v_6 must be dominating, then it is indifferent to choose v_8 or v_9 as such a node (in (32b) and (33a) below, we will choose arbitrarily v_8). Applying (13), we get the following:

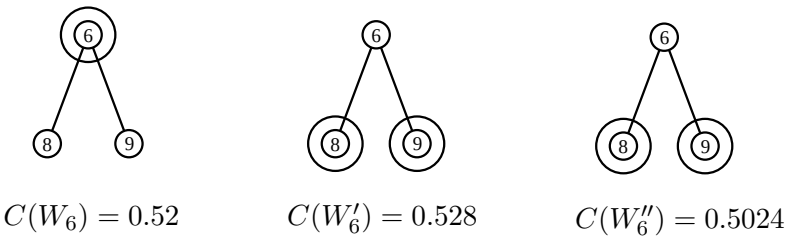
$$
\begin{aligned}
C(W_{6,0}) &= p + \sum_{i=8,9} C\left(W''_i\right) = 0.2 + 2 \times 0.16 = 0.52 \\
C(W_{6,1}) &= p + \sum_{i=8,9} C\left(W''_i\right) + \Delta''_8 = 0.2 + 2 \times 0.16 + 0.04 = 0.56 \qquad (32a) \\
C(W_{6,2}) &= p + \sum_{i=8,9} C\left(W''_i\right) + \sum_{i=8,9} \Delta''_i = 0.2 + 2 \times 0.16 + 2 \times 0.04 = 0.6 \qquad (32b)
\end{aligned}
$$

and thus, $C(W_6) = \min_{h=0,1,2} C(W_{6,h}) = C(W_{6,0}) = 0.52$. We now use (12) to get $C(W'_6)$:

$$\begin{aligned} C\left(W'_{6,1}\right) &= p(1-p) + \sum_{i=8,9} C\left(W'_i\right) + \Delta'_8 = 0.16 + 2M + 0.2 - M = 0.36 + M \quad (33a) \\ C\left(W'_{6,2}\right) &= p(1-p)^2 + \sum_{i=8,9} C\left(W'_i\right) + \sum_{i=8,9} \Delta'_i \\ &= 0.128 + 2M + 2(0.2 - M) = 0.528 \end{aligned}$$

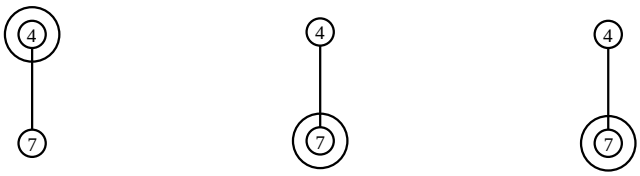
This example shows clearly the role of M in forcing all leaves to be either dominating nodes, or dominated by their fathers. In any structure with one leaf that is neither dominating, nor dominated by its father (as it is the case in $W'_{6,1}$), the overall weight will be in $O(M)$. This is the case in (33a). Thus, for an arbitrarily large M, $W'_{6,1}$ cannot be picked as optimal substructure, and $C(W'_6) = \min_{h=1,2} C(W'_{6,h}) = C(W_{6,2}) = 0.528$. Similarly, applying (14), one gets $C(W''_6) = \min_{h=1,2} C(W''_{6,h}) = C(W''_{6,2}) = 0.5024$.

All these results and corresponding substructures are illustrated in Figure 9. Substructures W_4, W'_4 and W''_4 are computed in the same way and are illustrated in Figure 10.



$C(W_6) = 0.52$ $C(W'_6) = 0.528$ $C(W''_6) = 0.5024$

Figure 9: Substructures W_6, W'_6 and W''_6.



$C(W_4) = 0.36$ $C(W'_4) = 0.36$ $C(W''_4) = 0.328$

Figure 10: Substructures W_4, W'_4 and W''_4.

Now, all substructures rooted at all children of v_2 have been computed; we continue by computing substructures rooted at v_2. First, let us sort children v_i of v_2 in increasing order of Δ'_i and Δ''_i. Recall that $\Delta'_5 = 0.2 - M$ and $\Delta''_5 = 0.04$ and notice that $\Delta'_6 = -0.08$ and $\Delta''_6 = 0.0176$. Thus, the two sorted lists are: (Δ'_5, Δ'_6) and (Δ''_6, Δ''_5).

From the first sorted list, we can assert that if v_2 is dominated and if it has only one dominating child, then this child must me v_5, since Δ'_5 appears first in the list. From the second sorted list, we can assert that if v_2 is dominating and if it has only one dominating child, then

this child must be v_6, since Δ''_6 appears first in the list. We now compute substructure for v_2:

$$\begin{aligned} C\left(W_{2,0}\right) &= p + \sum_{i=5,6} C\left(W''_i\right) = 0.8624 \\ C\left(W_{2,1}\right) &= p + \sum_{i=5,6} C\left(W''_i\right) + \Delta''_6 = 0.88 \\ C\left(W_{2,2}\right) &= p + \sum_{i=5,6} C\left(W''_i\right) + \sum_{i=5,6} \Delta''_i = 0.92 \end{aligned}$$

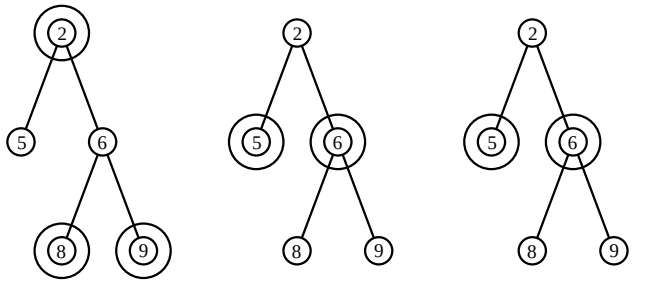
and $C(W_2) = C(W_{2,0}) = 0.8624$. Then:

$$\begin{aligned} C\left(W'_{2,1}\right) &= p(1-p) + \sum_{i=5,6} C\left(W'_i\right) + \Delta'_5 = 0.888 \\ C\left(W'_{2,2}\right) &= p(1-p)^2 + \sum_{i=5,6} C\left(W'_i\right) + \sum_{i=5,6} \Delta'_i = 0.776 \end{aligned}$$

and $C(W'_2) = C(W'_{2,2}) = 0.776$. Finally:

$$\begin{aligned} C\left(W''_{2,0}\right) &= p(1-p) + \sum_{i=5,6} C\left(W'_i\right) = 0.688 + M \\ C\left(W''_{2,1}\right) &= p(1-p)^2 + \sum_{i=5,6} C\left(W'_i\right) + \Delta'_5 = 0.856 \\ C\left(W''_{2,2}\right) &= p(1-p)^3 + \sum_{i=5,6} C\left(W'_i\right) + \sum_{i=5,6} \Delta'_i = 0.7504 \end{aligned}$$

and $C(W''_2) = C(W''_{2,2}) = 0.7504$. All results and structures are represented in Figure 11.



$C(W_2) = 0.8624 \quad C(W'_2) = 0.776 \quad C(W''_2) = 0.7504$

Figure 11: Substructures W_2, W'_2 and W''_2.

Finally, let us compute structures W_1 and W'_1. Note that we do not compute W''_1, since v_1 has no father, and a fortiori it cannot be dominated by such a node. As before, we first build sorted lists with Δ'_i's and Δ''_i's for all children v_i of v_1. These sorted lists are $(\Delta'_3, \Delta'_4, \Delta'_2)$ and $(\Delta''_4, \Delta''_3, \Delta''_2)$ since:

$$\begin{array}{lll} \Delta'_2 = 0.0864 & \Delta'_3 = 0.2 - M & \Delta'_4 = 0 \\ \Delta''_2 = 0.112 & \Delta''_3 = 0.04 & \Delta''_4 = 0.032 \end{array}$$

Suppose that v_1 is dominated. The first sorted list indicates us that:

- if v_1 has 1 dominating child, this must be v_3; this leads to (34b);

- if v_1 has 2 dominating children, these must be v_3 and v_4. this leads to (34c).

Similarly, suppose that v_1 is dominating. According to the second sorted list:

- if v_1 has 1 dominating child, this must be v_4, leading to (35a);
- if v_1 has 2 dominating children, these must be v_4 and v_3, leading to (35b).

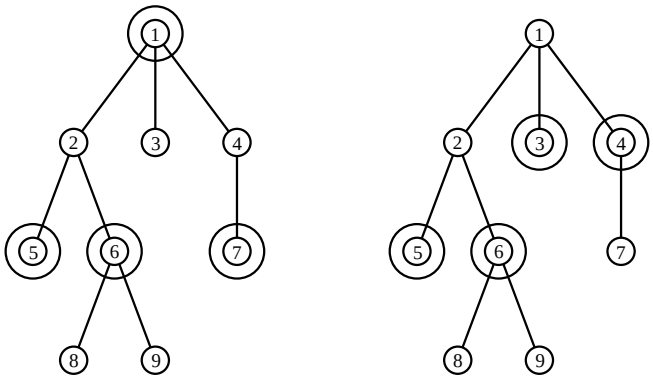
Thus:

$$\begin{aligned} C\left(W_{1,0}\right) &= p + \sum_{i=2,3,4} C\left(W_i''\right) = 1.4384 \\ C\left(W_{1,1}\right) &= p + \sum_{i=2,3,4} C\left(W_i''\right) + \Delta_4'' = 1.4704 \qquad (34a) \\ C\left(W_{1,2}\right) &= p + \sum_{i=2,3,4} C\left(W_i''\right) + \sum_{i=3,4} \Delta_i'' = 1.5104 \qquad (34b) \\ C\left(W_{1,3}\right) &= p + \sum_{i=2,3,4} C\left(W_i''\right) + \sum_{i=2,3,4} \Delta_i'' = 1.6224 \end{aligned}$$

and $C(W_1) = C(W_{1,0}) = 1.4384$. Also:

$$\begin{aligned} C\left(W_{1,1}'\right) &= p\left(1-p\right) + \sum_{i=2,3,4} C\left(W_i'\right) + \Delta_3'' = 1.496 \qquad (35a) \\ C\left(W_{1,2}'\right) &= p(1-p)^2 + \sum_{i=2,3,4} C\left(W_i'\right) + \sum_{i=3,4} \Delta_i'' = 1.464 \qquad (35b) \\ C\left(W_{1,3}'\right) &= p + (1-p)^3 \sum_{i=2,3,4} C\left(W_i'\right) + \sum_{i=2,3,4} \Delta_i'' = 1.5504 \end{aligned}$$

and $C(W_1') = C(W_{1,2}) = 1.464$. Corresponding structures are represented in Figure 12. The optimum on this instance is given by W_1, since $C(W_1) < C'(W_1)$.



$C(W_1) = 1.4384$ $\qquad$ $C(W_1') = 1.464$

Figure 12: Substructures W_1 and W_1'.