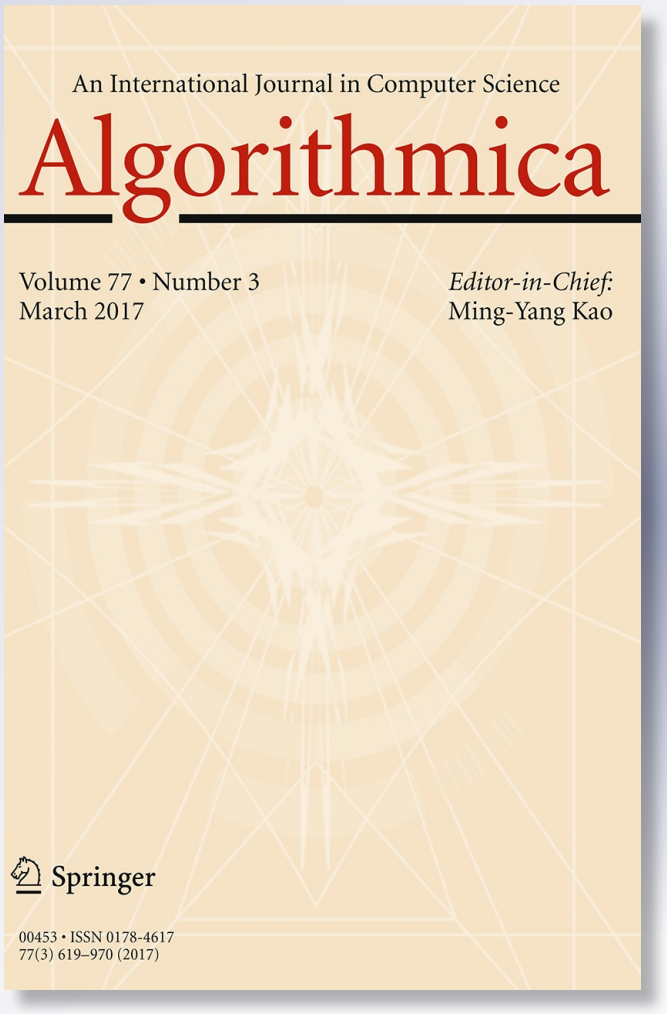
Exact Algorithms for Minimum Weighted Dominating Induced Matching

Min Chih Lin, Michel J. Mizrahi & Jayme L. Szwarcfiter

Algorithmica

ISSN 0178-4617
Volume 77
Number 3

Algorithmica (2017) 77:642-660
DOI 10.1007/s00453-015-0095-6



Springer

Algorithmica (2017) 77:642–660
DOI 10.1007/s00453-015-0095-6

Exact Algorithms for Minimum Weighted Dominating Induced Matching

Min Chih Lin[1,2] · **Michel J. Mizrahi**[1,2] · **Jayme L. Szwarcfiter**[3,4]

Received: 23 February 2014 / Accepted: 9 November 2015 / Published online: 19 November 2015

Abstract Say that an edge of a graph G dominates itself and every other edge sharing a vertex of it. An edge dominating set of a graph $G = (V, E)$ is a subset of edges $E' \subseteq E$ which dominates all edges of G. In particular, if every edge of G is dominated by exactly one edge of E' then E' is a dominating induced matching. It is known that not every graph admits a dominating induced matching, while the problem to decide if it does admit it is NP-complete. In this paper we consider the problems of counting the number of dominating induced matchings and finding a minimum weighted dominating induced matching, if any, of a graph with weighted edges. We describe three exact algorithms for general graphs. The first runs in linear time for a given vertex dominating set of fixed size of the graph. The second runs in polynomial time if the graph admits a polynomial number of maximal independent sets. The third

M.C. Lin was partially supported by UBACyT Grant 20020120100058, and PICT ANPCyT Grants 2010-1970 and 2013-2205. M.J. Mizrahi was partially supported by PICT ANPCyT Grants 2010-1970 and 2013-2205. J.L. Szwarcfiter was partially supported by CNPq, CAPES and FAPERJ, research agencies.

✉ Michel J. Mizrahi
michel.mizrahi@gmail.com

Min Chih Lin
oscarlin@dc.uba.ar

Jayme L. Szwarcfiter
jayme@nce.ufrj.br

1 CONICET, Instituto de Cálculo, Buenos Aires, Argentina

2 Departamento de Computación, Universidad de Buenos Aires, Buenos Aires, Argentina

3 I. Mat., COPPE and NCE, Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brazil

4 Instituto Nacional de Metrologia, Qualidade e Tecnologia, Rio de Janeiro, Brazil

one is an $O^*(1.1939^n)$ time and polynomial (linear) space, which improves over the existing algorithms for exactly solving this problem in general graphs.

Keywords Exact algorithms · Graph theory · Dominating induced matchings

1 Introduction

Under the widely accepted assumption that $P \neq NP$ there are several problems with important applications for which no polynomial algorithm exists. The need to get an exact solution for many of those problems has led to a growing interest in the area of design and analysis of exact exponential time algorithms for NP-Hard problems [16,32]. Even a slight improvement of the base of the exponential running time may increase the size of the instances being tractable. There has been many new and promising advances in recent years towards this direction [1–3,12].

In this paper we give exact algorithms for the weighted and counting versions of the problem dominating induced matching (also known as DIM or efficient edge domination) which has been extensively studied [5,6,8–11,20,24,25]. Further notes about this problem and some applications related to encoding theory, network routing and resource allocation can be found in [17,23]. We represent the minimum weight DIM problem for a graph G as $\text{DIM}_\Omega(G)$ and the counting of DIM's in G as $\text{DIM}_C(G)$.

The unweighted version of the DIM problem is known to be NP-complete [17], even for planar bipartite graphs of maximum degree 3 [5] or regular graphs [10]. There are polynomial time algorithms for some graph classes, such as chordal graphs [24], generalized series-parallel graphs [24] (both for the weighted problem), claw-free graphs [9], graphs with bounded clique-width [9], hole-free graphs [5], convex graphs [20], dually-chordal graphs [6], P_7-free graphs [8], bipartite permutation graphs [25], AT-free graphs [6], interval-filament graphs [6], weakly chordal graphs [6]. See also [7].

A straightforward brute-force algorithm using an alternative definition of the problem explained later to solve the weighted DIM has running time $O^*(2^n)$ time and polynomial space.

The $\text{DIM}_\Omega(G)$ problem can be expressed as an instance of the maximum weighted independent set (MWIS) problem on the square of the line graph $L(G)$ of G, and also as an instance of the minimum weighted dominating set problem on $L(G)$, by slightly adjusting the models [6,26] for the unweighted DIM problem. The MWIS problem can be solved in $O^*(1.2377^n)$ time [31] (how one obtains an algorithm for MWIS from an algorithm for weighted 2-Sat is described in [14]). On the other hand, the minimum weighted dominating set problem can be solved in time $O^*(1.5535^n)$ [15], and the special case where the weights are polynomially bounded in time $O^*(1.5048^n)$ [30]. The unweighted minimum dominating set can be solved in $O(1.4864^n)$ [19]. Hence $\text{DIM}_\Omega(G)$ can be solved by using the $L^2(G)$ algorithm in $O^*(1.2377^m)$ time applying the MWIS algorithm and in $O^*(1.5048^m)$ time using the minimum weighted dominating set algorithm. The unweighted DIM problem can also be solved in $O(1.5849^n)$ time [18] (Theorem 10) by listing all maximal induced matchings of the graph G, and can be easily adapted to solve $\text{DIM}_\Omega(G)$ and $\text{DIM}_C(G)$ within the same complexity.

For the counting problem, there exist algorithms such as [13] which can be used to count the number of MWIS's in $O^*(1.3247^n)$ time, leading an $O^*(1.3247^m)$ time algorithm to count the numbers of DIM's.

In this paper, we describe three exact (exponential time) algorithms for the weighted problem. The first runs in linear time for a given vertex dominating set of fixed size of the graph, for instance, the non-Helly Circular-arc graph class among others. The second runs in polynomial time if the graph admits a polynomial number of maximal independent sets, for instance, pK_2-free graphs, for any fixed positive integer p. The third one solves the problems of finding the minimum weighted DIM and that of counting the DIM's in $O(m^2 + m \cdot 1.1939^n) \in O^*(1.1939^n)$ time and $O(m)$ space in general graphs, which improves over the existing algorithms. We employ techniques described in [16] for the analysis of our algorithms, and as such we use their terminology.

This last proposed algorithm was designed using the *branch and reduce paradigm*. More information about this design technique as well as the running time analysis for this kind of algorithms can be found in [16].

An extended abstract with some partial results described in the present article has been presented in *24th International Symposium on Algorithms and Computation (ISAAC' 2013), Lecture Notes in Computer Science 8283 (2013), 558–567.* [22]

After the submission of this work, Xiao and Nagamochi [33] described an $O^*(1.1467^n)$ time algorithm for the dominating induced matching problem. The algorithm [33] is also based on the application of the branching rules, as presented in [22] and [21]. However, by describing structural properties of dominating induced matchings and making an explicit use of partioning the graph into an independent set and an induced matching, these properties allowed to design an exact algorithm having the above complexity [33].

2 Preliminaries

Given an edge $e \in E$, say that e *dominates* itself and every edge sharing a vertex with e. A subset $E' \subseteq E$ is an *induced matching* of G if each edge of G is dominated by at most one edge in E'. A *dominating induced matching (DIM)* of G is a subset of edges which is both dominating and an induced matching. Not every graph admits a DIM, and the problem of determining whether a graph admits it is also known in the literature as *efficient edge domination problem*. The weighted version of the DIM problem is to find a DIM such that the sum of weights of its edges is minimum among all DIMs, if any.

We will use an alternative definition [11] for the problem of finding a dominating induced matching. It asks to determine if the vertex set of a graph G admits a partition into two subsets such that the vertices of the first subset are called *white* and induce an independent set of the graph, while those of the second subset are named *black* and induce a 1-regular graph.

We consider only graphs without isolated vertices, because isolated vertices must be white in any black-white partition. So, we can delete them and solve the problem in the residual graph.

Assigning one of the two possible colors to vertices of G is called a coloring of G. A coloring is *partial* if only part of the vertices of G have been assigned colors, otherwise it is *total*. A partial coloring is *valid* if no two white vertices are adjacent and no black vertex has more than one black neighbor. A black vertex is *single* if it has no black neighbors, otherwise, it is *paired*. A total coloring is valid if no two white vertices are adjacent and every black vertex is paired. Clearly, G admits a DIM if and only if it admits a total valid coloring. In fact, a total valid coloring defines exactly one DIM, given by the set of *black* vertices.

3 Colorings and Extensions

Each coloring C of a graph G, partial or total, can be *valid* or *invalid*. We describe the natural conditions for determining if a coloring is valid or invalid.

Definition 1 Validation rules:

A partial coloring is valid whenever:

(V1) No two white vertices are adjacent, and
(V2) Each black vertex is either single or paired. Each single vertex has some uncolored neighbor.

A total coloring is valid whenever:

(V3) No two white vertices are adjacent, and
(V4) Each black vertex is paired.

Lemma 1 *There is a one-to-one correspondence between total valid colorings and dominating induced matchings of a graph.*

Proof It follows from the definitions. □

For a coloring C of the vertices of G, denote by $C^{-1}(white)$ and $C^{-1}(black)$, the subsets of vertices colored white and black. A coloring C' is an *extension* of a C if $C^{-1}(black) \subseteq (C')^{-1}(black)$ and $C^{-1}(white) \subseteq (C')^{-1}(white)$. For $V', V'' \subset V(G)$ if C' is obtained from C by adding to it the vertices of V' with the color black and those of V'' with the color white then write $C = C' \cup BLACK(V') \cup WHITE(V'')$. Denote with U the set of still uncolored vertices, and $N_U(v)$, the uncolored vertices from the neighborhood of vertex v.

Given a partial coloring C, the basic idea of the algorithm is to iteratively find extensions C' of C, until eventually a total valid coloring is reached. It follows from the validation rules that if C is invalid, so is C', since the colored vertices are maintained in C' and thus, the cause of the break of the validation rules. Therefore, the algorithm keeps checking for validation, and would discard an extension whenever it becomes invalid.

Basically, there are two different ways of possibly extending a coloring, using propagation rules and branching rules. At the beginning, there are partial colorings C which force the colors of some of the so far uncolored vertices, leading to an extension C' of C. In this case, say that C' has been obtained from C by *propagation*. Therefore,

at each step, every propagation rule that can be applied is executed, until no one can be used. If the coloring is still partial and valid, then one branching rule is used to obtain new colorings into which the algorithm tries again the propagation rules, and so on.

The following is a convenient set of rules, whose application may extend C, in the above described way.

Lemma 2 *Rules for propagating colors:*
The following are forced colorings for the extensions of a partial coloring of G.

(P1) The degree-3 vertices of a diamond must be black and the remaining ones must be white
(P2) The neighbor of a pendant vertex must be black
(P3) Each neighbor of a white vertex must be black
(P4) Except for its pair, the neighbors of a paired (black) vertex must be white
(P5) Each vertex with two black neighbors must be white
(P6) If a single black vertex has exactly one uncolored neighbor then this neighbor must be black
(P7) In an induced paw, the two odd-degree vertices must have different colors
(P8) In an induced C_4, adjacent vertices must have different colors
(P9) If $\forall v \in N_U(s)$, $N[v] \subseteq N[s]$ where s is a single (black vertex) then an uncolored neighbor v of s minimizing weight(sv) must be black. Break ties arbitrarily. We require rules (P1) and (P8) to be applied before (P9).

Proof The rules (P1), (P7), (P8) follows from [5](using observations 1 and 3). while rules (P3), (P4), (P5), (P6) follows from [11]. The rule (P2) follows from the coloring definition since each black vertex must be paired in order for the coloring to be *valid*. Finally, for (P9), let s be a single vertex. Suppose the neighborhood of all uncolored neighbors of s lies within the neighborhood of s. Then the choice of the vertex to become the pair of s is independent of the choices for the remaining single vertices of the graph. Therefore, to obtain a minimum weighted dominating induced matching of G, the neighbor v of s minimizing $weight(sv)$ must be black. □

Observe that rules (P1) and (P2) should be applied once at the beginning of the algorithm, and note that as long as rule (P8) is executed before rule (P9), the order of the application does not matter.

Say that a coloring C is *empty* if all vertices are uncolored. Let C be a valid coloring and C' an extension of it, obtained by the application of the propagation rules. If $C = C'$ then C is called *stable*. On the other hand, if $C \neq C'$ then C' is not necessarily valid. Therefore, after applying iteratively the propagation rules, we reach an extension which is either invalid or stable.

Lemma 3 [5] *If G contains a K_4 then G has no DIM.*

3.1 An Algorithm Based on Vertex Domination

Next, we will propose an exact algorithm for solving the weighted dominating induced matching problem, for general graphs, based on vertex dominations. The only rules to be used will be some of the propagation ones.

Let C be a *partial* valid coloring of $G = (V, E)$. Such as in [11], this coloring can be further propagated according to the previous defined propagation rules.

Let C be a partial valid coloring of G, and C' be a stable coloring obtained from C, by the applications of rules (P3)–(P6), above. Denote by D and D', respectively the subsets of vertices of G which are colored in C and C'. Clearly, $D' \supseteq D$. For our purposes, assume that the initial set D of colored vertices is a vertex dominating set of G.

Lemma 4 *Let C be a stable coloring and D a dominating set with its vertices colored. Then*

1. *If there are no single (black) vertices then C' is a total coloring,*
2. *Any uncolored vertex has exactly one black neighbor, and such a neighbor must be single.*

Proof Recall that D is the initial colored vertices and is a dominating set of the graph G. C' is not a total coloring if and only if there is some uncolored vertex v. Clearly, $v \notin D$ and $N(v) \cap D \neq \emptyset$. Let w be some neighbor of v in D. If the color of w is white then v must be colored black by rule (P3) which is a contradiction. Hence w is a black vertex. Again, if w is a paired black vertex or v has another black neighbor $w' \neq w$, v must get color white by rules (P4) and (P5) and this is a contradiction. Consequently, v has exactly one black neighbor and which is single black vertex. Therefore, if there are no single black vertices then there are not uncolored vertices and C' is a total coloring. □

Let D' be the colored vertices of the stable coloring C', let $S = \{s_1, \ldots, s_p\}$ be the set of single vertices, and recall that U represents the set of still uncolored vertices of G, that is, $U = V \backslash D'$. The above lemma implies that U admits a partition into (disjoint) parts:

$$U = (N(s_1) \cap U) \cup \cdots \cup \left(N(s_p) \cap U\right)$$

Theorem 1 *Let C be a coloring of the vertices of G, C' a stable extension of it, and $D = C^{-1}(black) \cup C^{-1}(white)$ a dominating set of G. Then (i) $S \subseteq C^{-1}(black)$; and (ii) if C extends to a valid total coloring C'' then C'' is an extension of C'.*

Proof Suppose that $S \nsubseteq C^{-1}(black)$ which means that exists a vertex $s_i \in S$ and $s_i \notin C^{-1}(black)$. By definition of S, s_i is a single black vertex. If $s_i \in D$ then $s_i \in C^{-1}(black)$, contradiction. Therefore $s_i \notin D$.

Since D is a dominating set, then $\exists v \in D$ such that $s_i \in N(v)$. If v is black then s_i is not a single black vertex, again a contradiction. Hence v must be white.

- If s_i has no uncolored neighbors then C is not extensible to a total valid coloring because s_i can not become a paired vertex, contrary to the hypothesis.
- Otherwise, let y be an uncolored neighbor of s_i. Clearly, $y \notin D$. Since y is uncolored then it has exactly one neighbor in D'. That is, s_i is the unique neighbor of y in D'. Since $D \subseteq D'$ and $s_i \in D' \backslash D$, it follows that D is not a dominating set, contradiction.

On the other hand, C' and C'' are extensions of C. Then the vertices of D have the same color in these colorings. Any colored vertex $v \notin D$ of C' was obtained by some propagation rule base on previous colored vertices. The rules are correct and deterministic. Hence, v must have the same color in C'' and C'' is an extension of C'. □

Clearly, given a partial valid coloring C, we can compute efficiently a stable extension C' of it. In addition, if D is a dominating set then we can try to obtain a total valid coloring from the stable coloring C' by appropiately choosing exactly one vertex from each subset $N_U(s_i)$, to be black, that is, to be the pair of the so far single vertex s_i.

Lemma 5 *Let U and S, respectively be the sets of uncolored and single vertices, relative to some stable coloring C' of graph G. If C' extends to a total valid coloring then, for each $s_i \in S$, $G[N_U(s_i)]$ is a union of a star and an independent set, any of them possibly empty. Moreover, the pair of s_i must be a maximum degree vertex in $G[N_U(s_i)]$.*

Proof Suppose by contrary that $G[N_U(s_i)]$ is not a union of a star and an independent set. Then $G[N_U(s_i)]$ contains either two non-adjacent edges, or a K_3.

- Let $\{(u_1, u_1), (v_1, v_2)\}$ be two disjoint edges in $G[N_U(s_i)]$. Since no white vertices can be adjacent, let u' be the black vertex from $\{u_1, u_2\}$ and v the black vertex from $\{v_1, v_2\}$. Then $\{u, s_i, v\}$ is a black P_3 or K_3 and therefore can not be extended to a valid coloring.
- Let $\{(u_1, u_2, u_3)\}$ be a K_3 in $G[N_U(s_i)]$. Therefore $\{s_i, u_1, u_2, u_3\}$ is a K_4 and therefore G has no valid coloring.

Consequently, $G[N_U(s_i)]$ must be a union of a star and an independent set. Now, suppose by contrary that the pair of s_i is a vertex $v \in N_U(s_i)$ and v has not maximum degree in $G[N_U(s_i)]$. Clearly, the rest of vertices in $N_U(s_i)$ are white vertices. In particular, a maximum degree vertex u in $G[N_U(s_i)]$ is white. But, there is some neighbor $z \neq v$ of u in $N_U(s_i)$ and z is not adjacent to v. Hence, z and u are white adjacent vertices, which is a contradiction. □

We can repeatedly execute the procedure below described for choosing the vertices to be paired to the single vertices s_i of the partial colorings. The procedure is repeated until all parts of the partition $U = N_U(s_1) \cup \cdots \cup N_U(s_p)$ have selected their paired black vertices or the coloring becomes invalid.
Let $s_i \in S$ be a single vertex. *Case 1*: $N_U(s_i) = \emptyset$: then stop, it will not lead to a valid one. *Case 2*: There is exactly one maximum degree vertex in $G[N_U(s_i)]$: then clearly, the only alternative is to choose this vertex. *Case 3*: There is no edge vw, where $v \in N_U(s_i)$ and $w \in N_U(s_j)$, for any $j \neq i$: then the choice of the neighbor of s_i to become black is independent on the choices of the others parts of the partition. Choose the vertex w of maximum degree in $G[N_U(s_i)]$ that minimizes the weight of the edge ws_i. *Case 4*: There is an edge vw, where $v \in N_U(s_i)$ and $w \in N_U(s_j)$, for some $j \neq i$: then v may become white if and only if w may become black. Each of these two choices may led to valid or invalid total colorings. So, we proceed with both alternatives (denoted by P_4-alternatives).

After applying any of the above Cases 2, 3 or 4, perform the propagation rules again and validate the coloring so far obtained. Proceed so until eventually the coloring becomes invalid, or a valid solution is obtained. At the end, choose the minimum weight solution obtained throughout the process.

As for the complexity, it is clear that it depends on the cardinality of the dominating set D and on the number of P_4-alternatives iterations, considered sequentially. Next, we describe bounds for these parameters.

Lemma 6 *There are at most 2^q P_4-alternative computations where $q \leq p = |S| \leq |D|$, and $q \leq \frac{n}{3}$.*

Proof By Theorem 1, it follows that $p \leq |D|$. On the other hand, we can apply the above Cases 1–4, in such an ordering that we keep applying Cases 1 and 2, with propagation until all remaining single vertices s_i satisfy $|N(s_i) \cap U| \geq 2$. Let $S' \subset S$ denote the set of remaining single vertices, and $q = |S'|$. Consequently, $q \leq \frac{n}{3}$.

Next, examine the P_4-alternative computations. They are generated by Case 4. Let vw be an edge of G, where $v \in N(s_i) \cap U$ and $w \in N(s_j) \cap U$, $i \neq j$. In one of the instances, v is black, meaning that s_i becomes paired, while in the other one w is black, implying that s_j becomes paired. This means that the size of the set S' of single vertices always decreases by at least one unit in all computations. Hence there are at most 2^q P_4-alternative computations. □

Considering that the remaining operations involved in each parallel thread of the algorithm can be performed in linear time, it is not hard to conclude that there there is an $O(2^q m)$ time algorithm to obtain a minimum DIM, if any, extensible from a partial valid coloring C of a weighted graph $G = (V, E)$ such that $D = C^{-1}(black) \cup C^{-1}(white)$ is a dominating set of G.

The complexity of the algorithm depends on the size of the dominating set D employed. We remark that if $G = (V, E)$ has no isolated vertices then we can easily find in linear time a dominating set with at most half the vertices. Just determine a maximal independent set I. Clearly, I and $V \setminus I$ are both dominating sets of G and one of them has at most $\frac{n}{2}$ vertices.

Finally, in order to obtain the minimum weighted DIM of the graph G, we have to apply the described algorithm for all possible bi-colorings of D. There are exactly $2^{|D|}$ such colorings.

Theorem 2 *There is an algorithm of complexity $O(min\{2^{2|D|}, 2^{\frac{5n}{6}}\} \cdot m) \approx O^*(min\{4^{|D|}, 1.7818^n\})$ to compute a minimum weighted DIM of a weighted graph, if existing.*

Proof The complexity is $O(2^{|D|} \cdot 2^q \cdot m) = O(2^{|D|} \cdot 2^{min\{|D|, \frac{n}{3}\}} \cdot m) = O(2^{|D|} \cdot min\{2^{|D|}, 2^{\frac{n}{3}}\} \cdot m) = O(min\{2^{2|D|}, 2^{|D|+\frac{n}{3}}\} \cdot m) = O(min\{2^{2|D|}, 2^{\frac{n}{2}+\frac{n}{3}}\} \cdot m) = O(min\{2^{2|D|}, 2^{\frac{5n}{6}}\} \cdot m)$. □

The upper bound depends heavily on the size of the dominating set. There are many sharp bounds over the size of a dominating set regarding the minimum degree of the graph. Ore proved that given minimum degree $\delta(G) \geq 1$, then $\gamma(G) \leq \frac{n}{2}$ [27]. Blank

[4] proved that $\gamma(G) \leq \frac{2n}{5}$ for every n-vertex graph with $\delta(G) \geq 2$ if $n \geq 8$, and Reed [28] prove that $\gamma(G) \leq \frac{3n}{8}$ if $\delta(G) \geq 3$.

Corollary 1 *If the dominating set size is $c \lg n$, where c is a constant, the above algorithm solves the minimum weighted DIM problem in polynomial time.*

Proof Given $|D| = c \lg n$, the running time is $O(2^{2(c \lg n)} \cdot m) = O(2^{\lg n \cdot 2c}) = O(n^{2c})$, where c is a constant, hence $O(n^{2c})$ is polynomial. □

Corollary 2 *The above algorithm solves the minimum weighted DIM problem in $O(m)$ time given a dominating set of fixed size.*

3.2 An Algorithm Based on Maximal Independent Sets

In this section, we describe an exact algorithm for finding a minimal weighted DIM of a graph, based on enumerating maximal independent sets. We consider a weighted graph $G = (V, E)$.

Any maximal independent set $I \subseteq V$ induces a partial bi-coloring in G as follows:

- color as black all vertices of $V \setminus I$
- color as white the vertices of I except those having exactly one single neighbor.

Observation If all vertices of G have degree $\neq 1$ then the above partial coloring is total.

The algorithm is then based on the following lemma.

Lemma 7 *Let G be a graph, I a maximal independent set of it and C the partial bi-coloring induced by I. Then C is extensible to a DIM if and only if C is a valid coloring and each single vertex, if existing, has at least one uncolored neighbor in C.*

Proof $\Rightarrow$) It is easy to see that if C is not a valid coloring, then it is not extensible to a *DIM*. Besides, if C has a single vertex v with no uncolored neighbors then all neighbors of v are white in C and in any extension of C. Also, C is not extensible to a *DIM* because v can not ever get its pair.
$\Leftarrow$) Let C be a valid coloring where each single black vertex has at least one uncolored neighbor. Then for each single black vertex v, choose any uncolored neighbor w to be its pair (w has exactly one single neighbor) and the remaining of uncolored vertices get color white. In this total coloring, the black vertices induce an 1-regular subgraph and the white vertex set induce an independent set because it is part of I. Hence, the total coloring is valid and hence a DIM. □

The algorithm can then be formulated as follows. Generate the maximal independent sets I of G. For each I, find its induced coloring C. If C is invalid or some single vertex has no uncolored neighbor then do nothing. Otherwise, for each single vertex v in C, if any, choose the minimum weight vw, among the uncolored neighbors of v; then color w as black and the remaining neighbors of v as white. The set of black vertices then forms a DIM of G. At the end select the minimum weight among all DIMs obtained in the process, if any.

Clearly, this algorithm determines the minimum weight DIM of a weighted graph $G = (V, E)$ because given any DIM $E' \subseteq E$ of G, the vertex set formed by those vertices not incident to any of the edges of E' is an independent set and as such, is clearly a subset of some maximal independent set of G. So, any DIM E' is considered in the algorithm.

All the operations performed by the algorithm relative to a fixed maximal independent set can be performed in linear time $O(m)$. If G has μ maximal independent sets, we can generate them all in time $O(nm\mu)$ time [29]. Therefore the complexity of the entire algorithm is $O(nm^2\mu)$. On the other hand, $\mu \leq O(3^{\frac{n}{3}})$, leading to a worst case of $O(3^{\frac{n}{3}}nm^2) \approx O^*(1.44225^n)$ time. In particular, if G is a bipartite graph then $\mu \leq 2^{\frac{n}{2}}$ and the complexity reduces to $O^*(1.41421^n)$. In any case, if G has a polynomial number of maximal independent sets then the algorithm terminates within polynomial time.

Finally, we observe the following additional relation between maximal independent sets and DIM's.

Lemma 8 *Let $G(V, E)$ be a graph with no isolated edges, $E' \subseteq E$ a DIM of G, and $I \subseteq V$ the independent set formed by those vertices not incident to any of the edges of E'. Then I is contained in exactly one maximal independent set of G.*

Proof If I is a maximal independent set there is nothing to prove. Otherwise, suppose the lemma is false and let I_1, I_2 be two distinct maximal independent sets properly containing I. Let $V_1 = I_1 \backslash I$, and $V_2 = I_2 \backslash I$. Choose any $v_2 \in V_2$. Clearly, $\{v_2\} \cup I$ is an indepedent set, and we know that $I_1 = V_1 \cup I$ is a maximal one. Consequently, there must be some vertex $v_1 \in V_1$ adjacent to v_2. However, both v_1 and v_2 are vertices incident to edges of the DIM E'. Consequently, $v_1v_2 \in E'$. In this case, v_1v_2 must form an isolated edge of G, a contradiction. Therefore the lemma holds. □

Theorem 3 *The minimum weighted DIM along with the counting version of the problem can be solved in $O^*(1.44225^n)$*

Proof Based on the above lemma and the fact that every isolated edge must be part of any DIM, it is simple to extend the exact algorithm proposed in this section, so as to count the number of distinct DIM's (unweighted or minimum weighted) of G, in the same complexity as deciding whether G admits a DIM. □

Observe that G may contain an exponential number of DIM's.

3.3 An $O^*(1.1939^n)$ Algorithm for $\mathrm{DIM}_{\Omega}(G)$ and $\mathrm{DIM}_{C}(G)$

In order to possibly extend a stable coloring C, we apply *branching rules*. Any coloring directly obtained by these rules is not forced. Instead, in each of the these rules, there are two possibly conflicting alternatives leading to distinct extensions C'_1, C'_2 of C. Each of C'_1 or C'_2 may be independently valid or invalid. The next lemma describes the branching rules. We remark that there exist simpler branching rules. However, using the rules below we obtain a sufficient number of vertices that get forced colorings,

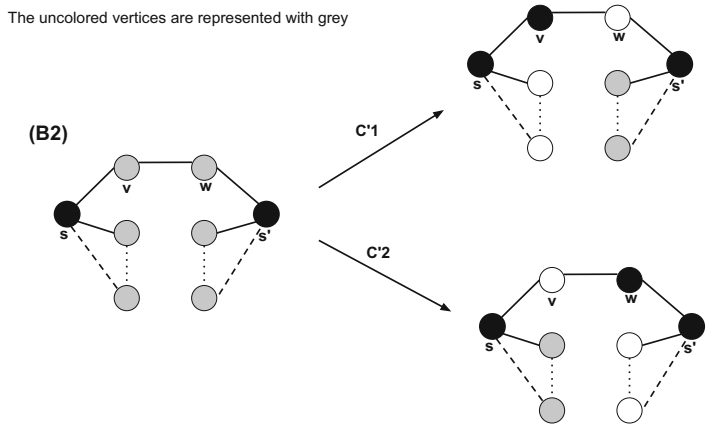


Fig. 1 The *grey* (*uncolored*) vertices are painted with *white* or *black* leading to C'_1 and C'_2

through the propagation which follow the application of any branching rule, so as to guarantee a decrease of the overall complexity of the algorithm. The complexity obtained relies heavily on this fact.

In general, we adopt the following notation. If C is a stable coloring then S denotes the set of single vertices of it, U is the set of uncolored vertices and $T = U \setminus \cup_{s \in S} N_U(s)$.

Lemma 9 *Branching rules*
Let C be a partial (valid) stable coloring of a graph G. At least one of the following alternatives can be applied to define extensions C'_1, C'_2 of C.

(B1) *If C is an empty coloring: choose an arbitrary vertex v then $C'_1 := C \cup BLACK(\{v\})$ and $C'_2 := C \cup WHITE(\{v\})$*

(B2) *If $\exists$ edge vw s.t. $v \in N_U(s)$ and $w \in N_U(s')$, for some $s, s' \in S, s \neq s'$ then $C'_1 := C \cup BLACK(\{v\})$ and $C'_2 := C \cup WHITE(\{v\})$. See* Fig. 1

(B3) *For some $s \in S$, if $\exists v \in N_U(s)$ s.t. $\exists w \in N_T(v)$:*

B3(a) *If $|N_U(s)| \neq 3 \vee d(w) \neq 3 \vee |N_T(v)| \geq 2$ then $C'_1 := C \cup BLACK(\{v\})$ and $C'_2 := C \cup WHITE(\{v\})$.*

B3(b) *If $|N_U(s)| = 3 \wedge d(w) = 3 \wedge N_T(v) = \{w\}$, let $N_U(s) = \{v, v', v''\}$.*

B3(b).i *If $N_U(v') = N_U(v'') = \emptyset$ then $C'_1 := C \cup BLACK(\{v\})$ and $C'_2 := C \cup WHITE(\{v\})$*

B3(b).ii *If $N_U(v') \neq \emptyset$, let $w' \in N_T(v')$, with $w' \neq w$. If $|N(w) \cup N(w')| > 5$ or $ww' \notin E(G)$ then $C'_1 := C \cup BLACK(\{v\})$ and $C'_2 := C \cup WHITE(\{v\})$*

B3(b).iii *If $N_U(v') \neq \emptyset$, let $w' \in N_T(v')$, with $w' \neq w$. If $ww' \in E(G)$ and $z \in N(w) \cap N(w')$ then $C'_1 := C \cup BLACK(\{v''\})$, while if $weight(sv) + weight(w'z) \leq weight(sv') + weight(wz)$ then $C'_2 := C \cup BLACK(\{v\})$, otherwise $C'_2 := C \cup BLACK(\{v'\})$ See* Fig. 2.

Lemma 10 *The branching rules are correct*

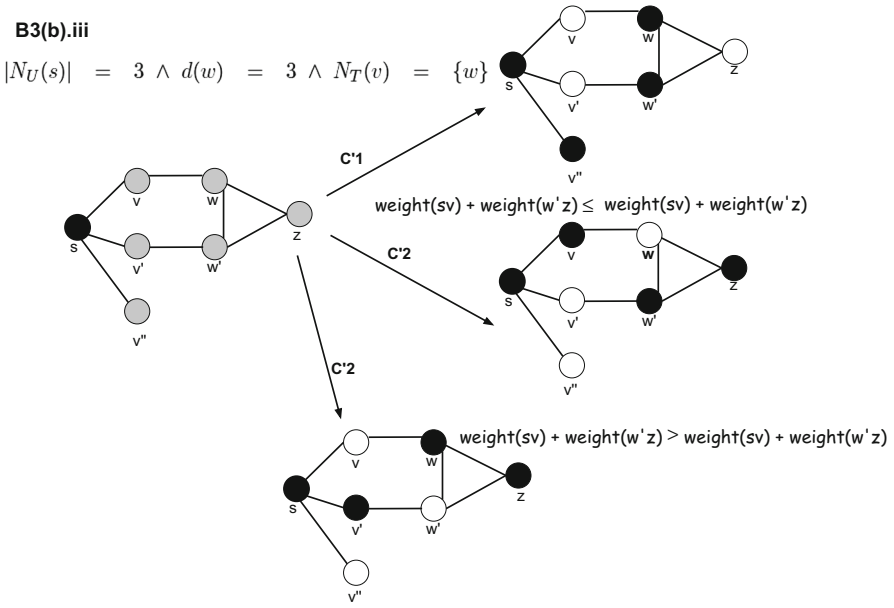


Fig. 2 In this case there are two colorings C'_2

Proof We assume that no propagation rule can be applied, hence branching rules are executed. If C is an *empty* coloring then rule (B1) applies. Otherwise, if C is not an *empty* coloring and C is not a *total* coloring then $S \neq \emptyset$. Since C is not *total* and the graph is connected, there is at least one edge sv where v is uncolored. If s is white then v must be black (P3), else if s is a *paired* vertex then v must be white (P4). Therefore s must be a single black vertex, hence $S \neq \emptyset$. Let $s \in S$. Since C is valid then $N_U(s) \neq \emptyset$ by validation rule (V2) and since is *stable* $|N_U(s)| \neq 1$ by (P6). Therefore $|N_U(s)| \geq 2$. Moreover rule (P9) can not be applied, therefore $\exists v \in N_U(s)$ s.t. $|N_U(v) \backslash N(s)| > 0$, let $w \in N_U(v) \backslash N(s)$. If $\exists s' \in S, s \neq s'$ s.t. $w \in N_U(s')$ then rule (B2) is applied. Otherwise, suppose rule (P2) can not be applied. Then $w \in N_T(v) (|N_T(v)| \geq 1)$. Clearly, $d(w) \neq 1$, otherwise, rule (P2) must be applied and v must get color black.

In case $|N_U(s)| \neq 3$ or $d(w) \neq 3$ or $|N_T(v)| \geq 2$ we apply rule B3(a). Otherwise: $|N_U(s)| = 3, d(w) = 3, |N_T(v)| = 1$. Note that in B3(b), $v'w'$ behaves symmetrically in respect to vw since otherwise $v'w'$ were found in step B3(a) replacing vw.
The first subcase of B3(a) corresponds to $N_U(v') = N_U(v'') = \emptyset$, while in the second and third subcases, v' or v'' has uncolored neighbors.
Suppose w.l.o.g. $N_U(v') \neq \emptyset$ where $w' \in N_T(v')$. It is easy to see that $w \neq w'$ since otherwise $svwv'$ is a C_4 and therefore w cannot be uncolored by rule (P8).
Now there are three cases which lead to two possible outcomes from the algorithm: In case $ww' \in E(G)$ or $|N_U(v) \cup N_U(w)| > 5$ then the result of the algorithm is given by the second subcase (ii), else it is given by the third subcase (iii). Note that $\{v, v', v''\} \in N_U(s)$ while $\{w, w'\} \in T$, hence these vertices are different since

they belong to disjoint sets. Also note that $\exists\, z \in N(w) \cap N(w')$ since otherwise the connected component has 7 vertices and can be trivially solved. □

Note that branching rules are applied in the order given: (B1), (B2), B3(a), B3(b).i, B3(b).ii, B3(b).iii.

We propose an algorithm for solving the problems of finding the minimum weighted DIM and that of counting the DIM's in $O(m^2 + m \cdot 1.1939^n) \in O^*(1.1939^n)$ time and $O(m)$ space in general graphs, which improves over the existing algorithms. We employ techniques described in [16] for the analysis of our algorithm, and as such we use their terminology. The proposed algorithm was designed using the *branch and reduce paradigm*. More information about this design technique as well as the running time analysis for this kind of algorithms can be found in [16].

The lemmas described in the coloring Sect. 3 lead to an exact algorithm for finding a minimum weight DIM of a graph G, if any.

In the initial step of the algorithm, we find the set $K4$ containing the K_4's of G in $O(m^2)$ time. If $K4 \neq \emptyset$, by Lemma 3, G does not have DIM's, and terminate the algorithm. Otherwise, define the set *COLORINGS* to contain through the process the candidates colorings to be examined and eventually extended.

Next, include in COLORINGS an *empty* coloring. In the general step, we choose any coloring C from i and remove it from this set. Then iteratively propagate the coloring by Lemma 2 into an extension C' of it, and validate the extension by Definition 1.

The iterations are repeated until one of the following situations is reached: C' is invalid, C' is a total valid coloring, or a partial stable (valid) coloring. In the first alternative, C' is discarded and a new coloring from *COLORINGS* is chosen. If C' is a a total valid coloring, find its weight and if smaller than the least weight so far obtained, it becomes the current candidate for the minimum weight of a DIM of G. Finally, when C' is stable we extended it by branching rules: choose the first rule of Lemma 9 satisfying C', compute the extensions C' and C'', insert them in *COLORINGS*, select a new coloring from *COLORINGS* and repeat the process.

Note that only once, at the beginning, the rules (P1) and (P2) are executed. Regarding (P7) and (P8), it is also convenient to collect at the beginning of the algorithm the information of the induced paw's and induced C_4's of the graph which is useful for efficient propagation of rules (P7) and (P8). For each induced paw, let v and w be the vertices of odd degree, then the vertex v is stored in the list $black\_force\_white(w)$ and w is stored in the list $black\_force\_white(v)$. Those lists represents vertices that should be colored with white color in case the representing vertex of the list is colored with black color. A similar approach is used for the induced C_4's of the graph. For each induced $C_4 = \{v_0, v_1, v_2, v_3\}$, the list $black\_force\_white(v_i)$ should append the vertices $v_{(i-1) \bmod 4}$, $v_{(i+1) \bmod 4}$. This is done for each $i \in \{0, 1, 2, 3\}$.

The formulation below describes the details. The propagation and validation of a coloring C are performed by the procedure $PROPAGATE - VALIDATE(C, RESULT)$. At the end, the returned coloring corresponds to the extension C' of C, after iteratively applying propagation. The variable RESULT indicates the outcome of the validation analysis. If C' is invalid then $RESULT$ is 'invalid'; if C' is a valid total coloring then it contains 'total', and otherwise $RESULT$ equals 'partial'.

Algorithm Minimum Weighted DIM / Counting DIM

1. Find the subset $K4$
   if $K4 \neq \emptyset$ **then** terminate the algorithm: G has no DIM
   $SOLUTION := NODIM$
2. $COLORINGS := \{C\}$, where C is an *empty* coloring
3. **while** $COLORINGS \neq \emptyset$ **do**
   a. choose $C \in COLORINGS$ and remove it from $COLORINGS$
   b. $PROPAGATE - VALIDATE(C, RESULT)$
   c. **if** $RESULT$ = 'total' **and** $weight(C) < SOLUTION$ **then**
   $SOLUTION := weight(C)$
   else if $RESULT$ = 'partial' **then**
   Set C'_1 and C'_2 according to branching RULES on C
   $COLORINGS := COLORINGS \cup \{C'_1, C'_2\}$
   end if
4. **Output** $SOLUTION$

procedure $PROPAGATE - VALIDATE(C, RESULT)$

Comment Phase 1: Propagation
1. $C' := C$
2. **repeat**
   $C := C'$
   $C' :=$ extension of C obtained by the PROPAGATION RULES
   until $C = C'$
Comment Phase 2: Validation
3. Using the VALIDATION RULES **1** do as follows:
   if C is an invalid coloring **then return** (C, 'invalid')
   else if C is a partial coloring **then return** (C,'partial')
   else return (C, 'total')

3.3.1 Correctness and Complexity

It is easy to see that our algorithm fits the *branch and reduce* paradigm [16]. The *propagation rules* can be mapped into *reduction rules*.

Theorem 4 *The algorithm described in the previous section correctly computes the minimum weight of a dominating induced matching of a graph G.*

Proof The correctness of the algorithm follows from the fact that the algorithm considers all colorings that represent a DIM that can have minimum weight. Lemmas 2 and 9 are applied to extend partial colorings. Invalid colorings are discarded, while valid colorings are further extended, except if some other valid coloring representing a better DIM (with less weight) appeared before.

For proving the worst-case running time upperbound for the algorithm we will use the following useful definition and theorem. □

Definition 2 [16] Let b a branching rule and n the size of the instance. Suppose rule b branches the current instance into $r \geq 2$ instances of sizes respectively at most $n-t_1, n-t_2, \ldots, n-t_r$, for all instances of size $n \geq max\{t_i : i = 1, 2, \ldots, r\}$. Then we call $b = (t_1, t_2, \ldots, t_r)$ the *branching vector* of branching rule b.

The branching vector $b = (t_1, t_2, \ldots, t_r)$ implies the linear recurrence $T(n) \leq T(n-t_1) + T(n-t_2) + \cdots, T(n-t_r)$.

Theorem 5 [16] *Let b be a branching rule corresponding to the branching vector $(t_1, t_2, \ldots, t_r)$. Then the running time of the branching algorithm using only branching rule b is $O^*(\alpha^n)$, where α is the unique positive real root of*

$$x^n - x^{n-t_1} - x^{n-t_2} - \cdots - x^{n-t_r} = 0$$

The unique positive real root α is the *branching factor* of the branching vector b. We denote the branching factor of $(t_1, t_2, \ldots, t_r)$ by $\tau(t_1, t_2, \ldots, t_r)$.

Therefore for analyzing the running time of a branching algorithm we can compute the factor α_i for every branch rule b_i, and an upper bound of the running time of the branching algorithm is obtained by taking $\alpha = max_i \alpha_i$ and the result is an upper bound for the running time of $O^*(\alpha^n)$.

The upper bound is obtained by counting the leaves of the search tree given by the algorithm, using the fact that each leaf can be executed in polynomial time. The complexity of the algorithm without hiding the polynomial factor depends on the time for the execution of each branch in the search tree.

Further notes on this topic can be found in [16].

Theorem 6 *The algorithm above described requires $O^*(1.1939^n)$ time and $O(n+m)$ space for completion.*

Proof Using Definition 3.3.1 and Theorem 5 the computation of the upper bound time is reduced to calculating the *branching vector* for each branching rule (i.e. branching rules in our algorithm) and obtaining the associated *branching factor* for each case. Then the bound is given by the maximum *branching factor*. Note that it is required that the *reduction rules* (i.e. propagation rules in our algorithm) can be computed in polynomial time and leads to at most one valid extension of the considered coloring. So, the propagation rules do not affect the exponential factor of the algorithm. Moreover, each branch of the algorithm has cost $O(n+m)$ in time and space. This is easy to note since from the *empty* coloring up to any *total* coloring each vertex v is colored once and the cost for coloring each vertex is given by the updating of the color of the vertex and its neighbors, hence $O(|N(v)|)$ time. Therefore, the total cost for each branch is $O(n+m)$.

We consider as the size of an instance (a coloring) its number of uncolored vertices after application of propagation (reduction) rules. Let's analyze each branching rule to obtain the maximum *branching factor*:

1. If C is an *empty* coloring: choose an arbitrary vertex v then $C'_1 := C \cup BLACK(\{v\})$ and $C'_2 := C \cup WHITE(\{v\})$: It is easy to see that this rule is executed once, after that, the coloring is never empty again. Since each application of a branching rule

originates two branches, we can bound the time of the algorithm by twice the complexity of for computing an instance of size $n-1$. Therefore the asymptotic behavior of the algorithm is not affected.

2. *If* $\exists$ *edge* vw *s.t.* $v \in N_U(s)$ *and* $w \in N_U(s')$, *for some* $s, s' \in S, s \neq s'$ *then* $C'_1 := C \cup BLACK(\{v\})$ *and* $C'_2 := C \cup WHITE(\{v\})$.
   Here we extend the original coloring C' to C'_1 and C'_2 by coloring the vertex v with black and white respectively. Recall that there exists an edge vw such that $v \in N_U(s)$, $w \in N_U(s')$. If v is black then all the vertices of $N_U(s)\backslash v$ are white, while w is white. On the other hand, if v is white then w is black and $N_U(s')\backslash w$ are white. Therefore the size of uncolored vertices is reduced for each branch (i.e. for each new coloring). The associated branching vector is $(1+|N_U(s)|, 1+|N_U(s')|)$. By rule (P2) $|N_U(s)| \geq 2$ and $|N_U(s')| \geq 2$. The following observation turns out to be useful:
   Let $s_i \in S$ If $|N_U(s_i)| = 2$ then $N_U(s_i)$ can be totally colored, whether v is black or white. Therefore the branching vector with largest branching factor is (3, 5) ($\tau(3,5) \approx 1.1939$) and occurs whenever one of $\{s, s'\}$ has two uncolored neighbors and the other one has three uncolored neighbors.
3. For some $s \in S$, if $\exists v \in N_U(s)$ s.t. $\exists w \in N_T(v)$:
   Note that if $\nexists w \in N_T(v)$ for any $v \in N_U(s)$ then either the propagating rule (P9) or (P5) can be applied to get an extension of the coloring.
   (a) If $|N_U(s)| \neq 3 \vee d(w) \neq 3 \vee |N_T(v)| \geq 2$ then $C'_1 := C \cup BLACK(\{v\})$ and $C'_2 := C \cup WHITE(\{v\})$.
   Since v is uncolored then w is not a *pendant* vertex, $d(w) > 1$. Since w is uncolored then it has neither a white nor a paired black neighbor. Moreover, if w has a single black neighbor then this is the case analyzed above. Therefore w has uncolored neighbors and let x be one of them.
      (a.1) $|N_T(v)| \geq 2$: Let $v' \in N_T(v)$. Using the same reasoning, we claim: $\exists x' \in N_U(v')$. In C'_1, $\{v, x\}$ will be black, while $\{x, v', w\}$ will be white In C'_2, $\{v\}$ will be white while $\{v', w\}$ will be black. This leads to the branching vector (3, 5).
      (a.2) $d(w) \neq 3$. If $d(w) = 2$ then in C'_1 the vertices $N_U(s) \cup \{w, x\}$ will be colored and in C'_2 the vertices $\{v, x\}$ will be black, while $\{w\}$ will be white. Therefore the branching vector with largest branching factor is (3, 5).
      Else if $d(w) > 3$ then in C'_1 the vertices $N_U(s) \cup N_U[w]$ will be colored and in C'_2 the vertices $\{v, w\}$ will be colored. In case $|N_U(s)| = 2$ then v_1 will be colored too. Therefore the branching vectors are either (2, 7) ($\tau(2,7) = 1.1908$) or (3,6) ($\tau(3,6) = 1.1739$).
      (a.3) $|N_U(s)| = 2$: Let $N_U(s) = \{v, v_1\}$ and $N(w) = \{v, x, x'\}$. In C'_1 after applying propagation rules the vertices $\{v, x, x'\}$ will be black, while $\{v_1, w\}$ will be white. In C'_2 after applying propagation rules the vertices $\{v_1, w\}$ will be black, while $\{v\}$ will be white. The result is the branching vector (3, 5).
      (a.4) $|N_U(s)| > 3$: Let $\{v_1, v_2, v_3\} \subseteq N_U(s)$ and $N(w) = \{v, x, x'\}$. In C'_1 after applying propagation rules the vertices $\{v, x, x'\}$ will be black, while $\{v_1, v_2, v_3, w\}$ will be white. In C'_2 after applying propagation rules the

vertices $\{w\}$ will be black, while $\{v\}$ will be white. The result is the branching vector (2, 7).

(b) *If* $|N_U(s)| = 3 \wedge d(w) = 3$ *where* $N_U(w) = \{v, x, x'\}$, $N_U(s) = \{v, v', v''\}$
Note that $\{x, x'\} \cap \{v, v', v''\} = \emptyset$ since otherwise at least one of them must be colored by rule (P8).

(b.1) *If* $N_U(v') = N_U(v'') = \emptyset$ *then*
$C'_1 := C \cup BLACK(\{v\})$ *and* $C'_2 := C \cup WHITE(\{v\})$:
Suppose w.l.o.g. $weight(sv') \leq weight(sv'')$, then:
In C'_1, after applying the propagation rules the vertices $\{v, x, x'\}$ will be black, while, $\{v', v'', w\}$ will be white.
In C'_2, after applying propagation rules the vertices $\{v', w\}$ will be black, while $\{v, v''\}$ will be white. The result is the branching vector (4,6) ($\tau(4, 6) = 1.1510$).

(b.2) *If* $N_U(v') \neq \emptyset$, *let* $w' \in N_T(v')$, *with* $w' \neq w$:
If $|N_T[w] \cup N_T[w']| > 5$ then
$C'_1 := C \cup BLACK(\{v\})$ and $C'_2 := C \cup WHITE(\{v\})$
Note that if $d(w') \neq 3$ then $v'w'$ satisfies the properties of an already analized case, hence $C'_1 := C \cup BLACK(\{v'\})$ and $C'_2 := C \cup WHITE(\{v'\})$.
Since $d(w) = d(w') = 3$ and $|N_T[w] \cup N_T[w']| > 5$, then $\exists x, y$ s.t. $x \in N_T(w), x \notin N_T(w')$ and $y \in N_T(w'), y \notin N_T(w)$. In C'_1 after applying propagation rules the vertices $\{v, x, x', w'\}$ will be black, while $\{v', v'', w\}$ will be white. If $x' = w'$ then y must be black by rule (P6). In C'_2 the vertex $\{w\}$ will be black, while the vertex $\{v\}$ will be white. The result is the branching vector (2, 7).

(b.3) *If* $N_U(v') \neq \emptyset$, *let* $w' \in N_T(v')$, $w' \neq w$
If $|N_T[w] \cup N_T[w']| \leq 3$ *and* $z \in N(w) \cap N(w')$ then
$C'_1 := C \cup BLACK(\{v''\})$,
if $weight(sv) + weight(w'z) \leq weight(sv') + weight(wz)$ then
$C'_2 := C \cup BLACK(\{v\})$
otherwise $C'_2 := C \cup BLACK(\{v'\})$
Since $d(w) = d(w') = 3$ then $ww' \in E(G)$ and $\exists z \in N_T(v) \cap N_T(w)$, otherwise the case is one of the above.
In both colorings, C'_1 and C'_2 the vertices $\{v, v', v'', w, w', z\}$ will be colored. The branching vector is (6,6) ($\tau(6, 6) = 1.1225$).

The worst branching factor is $\tau(3, 5) \approx 1.1939$. In consequence, the time complexity of this algorithm is $O^*(1.1939^n)$. To achieve linear space complexity, we use a stack to store the coloring sequence of the current branch. □

3.3.2 Counting the Number of DIM's

The previous algorithm can be easily adapted to count the number of DIM's. Given a coloring C we define $TVC(C)$ the number of *total valid* colorings that can be extended from C. *If we apply any propagation rule to coloring* C *we obtain a coloring* C'. *Clearly* $TVC(C) = TVC(C')$, *except for rule (P9). In the latter case* $TVC(C) = TVC(C') \cdot |N_U(s)|$ *where* s *is the single vertex chosen to apply the rule.*

Note that by swapping s for any vertex $v \in N_U(s)$ we get another valid DIM, since it satisfy Definition 1 from valid coloring. Thus, each vertex $v \in N_U(s)$ defines a different DIM.

If we apply any branching rule to coloring C we obtain two extended colorings C'_1 and C'_2. Clearly $TVC(C) = TVC(C'_1) + TVC(C'_2)$, except for rule B3(b).iii. In the latter case $TVC(C) = TVC(C'_1) + 2 \cdot TVC(C'_2)$.

Note that since C'_1 and C'_2 have a different color for at least one vertex, therefore the colorings from $TVC(C'_1)$ are disjoint from the colorings of $TVC(C'_2)$ and we must count each one separately. On the other hand, the colorings from $TVC(C'_2)$ can have either v black or v' black, each one leading to a different DIM. Thus, each coloring of those vertices gives a different coloring to be counted. Therefore there are $2 \cdot TVC(C'_2)$ different DIM's.

Using the above facts it is trivial to modify the algorithm to solve the counting problem.

References

1. Björklund, A.: Determinant sums for undirected hamiltonicity. In: Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, pp. 173–182 (Washington, DC, USA), FOCS '10, IEEE Computer Society (2010)
2. Björklund, A., Husfeldt, T.: Exact graph coloring using inclusion-exclusion. In: Kao, M.-Y. (ed.) Encyclopedia of Algorithms. Springer, Berlin (2008)
3. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets mobius: fast subset convolution. In: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '07, ACM, pp. 67–74 (2007)
4. Blank, M.: An estimate of the external stability of a graph without pendant vertices. Prikl. Math. Programmirovanie **10**, 3–11 (1973)
5. Brandstädt, A., Hundt, C., Nevries, R.: Efficient edge domination on hole-free graphs in polynomial time. In: Proceedings of the 9th Latin American Conference on Theoretical Informatics, pp. 650–661 (Berlin, Heidelberg), LATIN'10, Springer (2010)
6. Brandstädt, A., Leitert, A., Rautenbach, D.: Efficient dominating and edge dominating sets for graphs and hypergraphs. In: (Chao, K.-M., Hsu, T., Lee, D.-T. (eds.) ISAAC, Lecture Notes in Computer Science, vol. 7676, pp. 267–277. Springer (2012)
7. Brandstädt, A., Lozin, V.V.: On the linear structure and clique-width of bipartite permutation graphs. Ars Comb. **67**, 273–281 (2003)
8. Brandstädt, A., Mosca, R.: Dominating induced matchings for P_7-free graphs in linear time. In: Asano, T., Nakano, S.-I., Okamoto, Y., Watanabe, O. (eds.) ISAAC, Lecture Notes in Computer Science, vol. 7074, pp. 100–109. Springer (2011)
9. Cardoso, D.M., Korpelainen, N., Lozin, V.V.: On the complexity of the dominating induced matching problem in hereditary classes of graphs. Discrete Appl. Math. **159**(7), 521–531 (2011)
10. Cardoso, D.M., Cerdeira, J.O., Delorme, C., Silva, P.C.: Efficient edge domination in regular graphs. Discrete Appl. Math. **156**(15), 3060–3065 (2008)
11. Cardoso, D.M., Lozin, V.V.: Dominating induced matchings. In: Lipshteyn, M., Levit, V.E., McConnell, R.M. (eds.) Graph Theory, Computational Intelligence and Thought , Lecture Notes in Computer Science, vol. 5420, pp. 77–86. Springer (2009)
12. Cygan, M., Pilipczuk, M.: Even faster exact bandwidth. ACM Trans. Algorithms **8**(1), 8 (2012)
13. Dahllöf, V., Jonsson, P.: An algorithm for counting maximum weighted independent sets and its applications. In: Eppstein, D. (ed.) SODA, pp. 292–298. ACM/SIAM, Philadelphia (2002)
14. Dahllöf, V., Jonsson, P., Wahlström, M.: Counting models for 2SAT and 3SAT formulae. Theor. Comput. Sci. **332**(1–3), 265–291 (2005)
15. Fomin, F.V., Gaspers, S., Saurabh, S., Stepanov, A.A.: On two techniques of combining branching and treewidth. Algorithmica **54**(2), 181–207 (2009)

16. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. Texts in Theoretical Computer Science. Springer, Berlin (2010)
17. Grinstead, D.L., Slater, P.J., Sherwani, N.A., Holmes, N.D.: Efficient edge domination problems in graphs. Inf. Process. Lett. **48**(5), 221–228 (1993)
18. Gupta, S., Raman, V., Saurabh, S.: Maximum r-regular induced subgraph problem: fast exponential algorithms and combinatorial bounds. SIAM J. Discrete Math. **26**(4), 1758–1780 (2012)
19. Iwata, Y.: A faster algorithm for dominating set analyzed by the potential method. In: Marx, D., Rossmanith, P. (eds.) Parameterized and Exact Computation—6th International Symposium, IPEC 2011, Saarbrücken, Germany, September 6–8, 2011. Revised Selected Papers, Lecture Notes in Computer Science, vol. 7112, pp. 41–54. Springer (2011)
20. Korpelainen, N.: A polynomial-time algorithm for the dominating induced matching problem in the class of convex graphs. Electron. Notes Discrete Math. **32**, 133–140 (2009)
21. Lin, M.C., Mizrahi, M.J., Szwarcfiter, J.L.: Exact algorithms for dominating induced matchings. CoRR abs/1301.7602 (2013)
22. Lin, M.C., Mizrahi, M.J., Szwarcfiter, J.L.: An O*(1.1939 n) time algorithm for minimum weighted dominating induced matching. In: Cai, L., Cheng, S.-W., Lam, T.W. (eds.) ISAAC, Lecture Notes in Computer Science, vol. 8283, pp. 558–567. Springer (2013)
23. Livingston, M., Stout, Q.F.: Distributing resources in hypercube computers. In: Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications: Architecture, Software, Computer Systems, and General Issues, vol. 1, pp. 222–231 (New York, NY, USA), C3P, ACM (1988)
24. Lu, C.L., Ko, M.-T., Tang, C.Y.: Perfect edge domination and efficient edge domination in graphs. Discrete Appl. Math. **119**(3), 227–250 (2002)
25. Lu, C.L., Tang, C.Y.: Solving the weighted efficient edge domination problem on bipartite permutation graphs. Discrete Appl. Math. **87**(1–3), 203–211 (1998)
26. Milanic, M.: Hereditary efficiently dominatable graphs. J. Graph Theory **73**(4), 400–424 (2013)
27. Ore, O.: Theory of graphs. Am. Math. Soc. Colloq. Publ. **38** (1962)
28. Reed, B.A.: Paths, stars and the number three. Comb. Probab. Comput. **5**, 277–295 (1996)
29. Tsukiyama, S., Ide, M., Ariyoshi, H., Shirakawa, I.: A new algorithm for generating all the maximal independent sets. SIAM J. Comput. **6**(3), 505–517 (1977)
30. van Rooij, J.M.M., Nederlof, J., van Dijk, T.C.: Inclusion/exclusion meets measure and conquer. In: Fiat, A., Sanders, P. (eds.) ESA, Lecture Notes in Computer Science, vol. 5757, pp. 554–565. Springer (2009)
31. Wahlström, M.: A tighter bound for counting max-weight solutions to 2sat instances. In: Proceedings of the 3rd International Conference on Parameterized and Exact Computation, pp. 202–213 (Berlin, Heidelberg). IWPEC'08, Springer (2008)
32. Gerhard, J.: Woeginger, Combinatorial optimization–Eureka, You Shrink!. Springer-Verlag New York Inc., New York (2003)
33. Xiao, M., Nagamochi, H.: Exact algorithms for dominating induced matching based on graph partition. Discrete Appl. Math. **190–191**, 147–162 (2015)