An $O^*(1.1939^n)$ Time Algorithm for Minimum Weighted Dominating Induced Matching

Min Chih Lin[1,⋆], Michel J. Mizrahi[1,⋆], and Jayme L. Szwarcfiter[2,⋆⋆]

[1] CONICET, Instituto de Cálculo and Departamento de Computación Universidad de Buenos Aires, Buenos Aires, Argentina

[2] I. Mat., COPPE and NCE, Universidade Federal do Rio de Janeiro, and Instituto Nacional de Metrologia, Qualidade e Tecnologia, Rio de Janeiro, Brazil

oscarlin@dc.uba.ar, michel.mizrahi@gmail.com, jayme@nce.ufrj.br

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 finding a minimum weighted dominating induced matching, if any, and counting the number of dominating induced matchings of a graph with weighted edges. We describe an exact algorithm for general graphs that runs in $O^*(1.1939^n)$ time and polynomial (linear) space, for solving these problems. This improves over the existing exact algorithms for the problems in consideration.

Keywords: exact algorithms, dominating induced matchings, branch & reduce.

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 lead to a growing interest in the area of design and analysis of exact exponential time algorithms for NP-Hard problems [14,25]. 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 [1,2].

⋆ Partially supported by UBACyT Grants 20020100100754 and 20020120100058, PICT ANPCyT Grant 1970 and PIP CONICET Grant 11220100100310.

⋆⋆ Partially supported by CNPq, CAPES and FAPERJ, research agencies.

In this paper we give an exact algorithm 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 [3,4,5,7,8,17,9,20,21]. Further notes about this problem and some applications related to encoding theory, network routing and resource allocation can be found in [15,19].

The unweighted version of the DIM problem is known to be NP-complete [15], even for planar bipartite graphs of maximum degree 3 [3] or regular graphs [9]. There are polynomial time algorithms for some classes, such as chordal graphs [20], generalized series-parallel graphs [20] (both for the weighted problem), claw-free graphs [7], graphs with bounded clique-width [7], hole-free graphs [3], convex graphs [17], dually-chordal graphs [4], P_7-free graphs [5], bipartite permutation graphs [21], AT-free graphs [4], interval-filament graphs [4], weakly chordal graphs [4]. See also [6].

If $P \neq NP$ it is not possible to solve this problem in polynomial time, hence it becomes important to improve the exponential algorithm in order to identify instances that can be solved within reasonable time.

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

The paper [18] describes an algorithm for solving the weighted DIM problem in $O^*(1.7818^n)$, time while requiring $O(n + m)$ time, if the graph contains a vertex dominating set of fixed size. In the same work another approach based on enumerating maximal independent sets allows to solve both DIM problems (minimum weighted problem and counting problem) in $O^*(1.4423^n)$ time. There is also an $O^*(1.5849^n)$ algorithm from [16].

The minimum weighted DIM 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 [4,22] for the unweighted DIM problem. The MWIS problem can be solved in $O^*(1.2377^n)$ time [24] (how one obtains an algorithm for MWIS from an algorithm for weighted 2-Sat is described in [10]). On the other hand, the minimum weighted dominating set problem can be solved in time $O^*(1.5535^n)$ [12], and the special case where the weights are polynomially bounded in time $O^*(1.5048^n)$ [23]. Hence the minimum weighted DIM problem for a graph G can be solved by using the $L^2(G)$ algorithm in $O^*(1.2377^m)$ time using the MWIS algorithm and in $O^*(1.5048^m)$ time using the minimum weighted dominating set algorithm.

For the counting problem, there exist algorithms such as [11] 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 propose an algorithm for solving the problems of finding the minimum weighted DIM and that of counting the DIM's in $O(m \cdot 1.1939^n) \in O^*(1.1939^n)$ time and $O(m)$ space in general graphs, which improves over the existing algorithms. We employ techniques described in [14] for the analysis of our algorithm, and as such we use their terminology. The proposed algorithm was designed using the *branch & reduce paradigm*. More information about this design technique as well as the running time analysis for this kind of algorithms can be found in [14].

2 Preliminaries

By $G(V, E)$ we denote a *simple undirected graph* with vertex set V and edge set E, $n = |V|$ and $m = |E|$. We consider G as a *weighted* graph, that is, one in which there is a non-negative real value, denoted $weight(vw)$ assigned to each edge vw of G. If $v \in V$ and $V' \subseteq V$, then denote by $N(v)$, the set of vertices adjacent (neighbors) to v, denote $d(v) = |N(v)|$ the degree of the vertex, denote by $G[V']$ the subgraph of G induced by V', and write $N_{V'}(v) = N(v) \cap V'$. Some special graphs or vertices are of interest for our purposes. A graph formed by two triangles having a common edge is called a *diamond*. By removing an edge incident to a vertex of degree 2 of a diamond, we obtain a *paw*. Finally, a vertex of degree 1 is called *pendant*.

Given an edge $e \in E$, say that *e dominates* itself and every edge sharing a vertex with e. 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 DIM problem is to find a DIM such that the sum of the weights of its edges is minimum among all DIM's, if any. The counting version of the problem consists on counting the number of DIM's of the graph. We assume the weights to be non-negative. However, the methods can be easily extended to the case of negative weights, without increasing the complexity of the algorithms.

It is not hard to see that every DIM is a maximum induced matching, and hence the number of edges of every DIM in G is the same. Therefore it is straightforward to modify the graph in order to solve the problem with non-negative weights and then transform it back to the original graph.

We assume the graph G to be connected, otherwise, the DIM of G is the union of the DIM's of its connected components, and so we can restrict to the connected case.

We will use an alternative definition [8] of 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. 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 an 1-regular graph.

A straightforward brute-force algorithm for finding the DIM of a graph G consists in finding all bipartitions of $V(G)$, color one of the parts as *white*, the other as *black*, and checking if the result is a valid DIM. The complexity of this algorithm is $O(2^n \cdot m) \in O^*(2^n)$.

3 Extensions of Colorings

Assigning one of the two possible colors, white or black, 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 black vertex is *single* if it has no black neighbor, and is paired if it has exactly one black neighbor. Each coloring, partial or total, can be *valid* or *invalid*.

Next, we describe the natural conditions for determining if a coloring is valid or invalid.

Definition 1. *RULES FOR VALIDATING COLORINGS:*

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 $C = C' \cup BLACK(V') \cup WHITE(V'')$.

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'. 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*. 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).*

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

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. 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, 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 4. *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 $C'_2 := C \cup WHITE(\{v\})$*

(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* $C'_2 := C \cup BLACK(\{v'\})$

Each rule is applied after the previous rule, that is, if the condition of the previous case is not verified in the entire graph. Note that this applies to subitems of case (B3).

4 The Algorithm

The lemmas described in the last section 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. 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 $COLORINGS$ 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 4 satisfying C', compute the extensions C' and C'', insert them in $COLORINGS$, select a new coloring from $COLORINGS$ and repeat the process.

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'. Finally, $BIFURCATE(C, C'_1, C'_2)$ computes the extensions C'_1 and C'_2 of C.

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')
```

5 Correctness and Complexity

It is easy to see that our algorithm fits the *branch & reduce* paradigm [14]. The *propagation rules* can be mapped into *reduction rules.*

Theorem 1. *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 4 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. *[14] 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 $T(n) \leq T(n-t_1) + T(n-t_2) + \ldots, T(n-t_r)$.

Theorem 2. *[14] 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*

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

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 [14]

Theorem 3. *The algorithm above described requires $O^*(1.1939^n)$ time and $O(n+m)$ space for completion.*

6 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.

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 $TVC(C) = TVC(C'_1) + 2 \cdot TVC(C'_2)$.

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

References

1. Bjorklund, A.: Determinant sums for undirected hamiltonicity. In: Annual Symposium on Foundations of Computer Science, FOCS 2010, pp. 173–182 (2010)
2. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: Fourier meets möbius: fast subset convolution. In: Annual Symposium on Foundations of Computer Science, STOC 2007, pp. 67–74 (2007)
3. Brandstädt, A., Hundt, C., Nevries, R.: Efficient edge domination on hole-free graphs in polynomial time. In: López-Ortiz, A. (ed.) LATIN 2010. LNCS, vol. 6034, pp. 650–661. Springer, Heidelberg (2010)
4. Brandstädt, A., Leitert, A., Rautenbach, D.: Efficient dominating and edge dominating sets for graphs and hypergraphs. In: Chao, K.-M., Hsu, T.-S., Lee, D.-T. (eds.) ISAAC 2012. LNCS, vol. 7676, pp. 267–277. Springer, Heidelberg (2012)
5. 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 2011. LNCS, vol. 7074, pp. 100–109. Springer, Heidelberg (2011)
6. Brandstädt, A., Lozin, V.V.: On the linear structure and clique-width of bipartite permutation graphs. Ars Combinatoria 67, 273–281 (2003)
7. Cardoso, D.M., Korpelainen, N., Lozin, V.V.: On the complexity of the dominating induced matching problem in hereditary classes of graphs. Discrete Applied Mathematics 159, 21–531 (2011)
8. 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. LNCS, vol. 5420, pp. 77–86. Springer, Heidelberg (2009)
9. Cardoso, D.M., Cerdeira, J.O., Delorme, C., Silva, P.C.: Efficient edge domination in regular graphs. Discrete Applied Mathematics 156, 3060–3065 (2008)
10. Dahllöf, V., Jonsson, P., Wahlström, M.: Counting models for 2sat and 3sat formulae. Theoretical Computer Science 332, 265–291 (2005)
11. Dahllöf, V., Jonsson, P.: An algorithm for counting maximum weighted independent sets and its applications. In: ACM-SIAM Symposium on Discrete Algorithms, SODA 2002, pp. 292–298 (2002)
12. Fomin, F.V., Gaspers, S., Saurabh, S., Stepanov, A.A.: On two techniques of combining branching and treewidth. Algorithmica 54, 181–207 (2009)
13. Fomin, F.V., Grandoni, F., Kratsch, D.: Measure and conquer: a simple $O^*(1.220^n)$ independent set algorithm. In: SODA 2006 ACM-SIAM Symposium on Discrete Algorithms, pp. 18–25 (2006)
14. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. EATCS Series in Theoretical Computer Science. Springer, Berlin (2010)
15. Grinstead, D.L., Slater, P.J., Sherwani, N.A., Holmes, N.D.: Efficient edge domination problems in graphs. Information Processing Letters 48, 221–228 (1993)
16. Gupta, S., Raman, V., Saurabh, S.: Maximum r-regular induced subgraph problem: Fast exponential algorithms and combinatorial bounds. SIAM Journal on Discrete Mathematics 26, 1758–1780 (2012)
17. Korpelainen, N.: A polynomial-time algorithm for the dominating induced matching problem in the class of convex graphs. Electronic Notes in Discrete Mathematics 32, 133–140 (2009)
18. Lin, M.C., Mizrahi, M.J., Szwarcfiter, J.L.: Exact algorithms for dominating induced matchings. CoRR, abs/1301.7602 (2013)

19. Livingston, M., Stout, Q.F.: Distributing resources in hypercube computers. In: C3P Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications: Architecture, Software, Computer Systems, and General Issues, vol. 1, pp. 222–231. ACM (1988)
20. Lu, C.L., Ko, M.-T., Tang, C.Y.: Perfect edge domination and efficient edge domination in graphs. Discrete Applied Mathematics 119, 227–250 (2002)
21. Lu, C.L., Tang, C.Y.: Solving the weighted efficient edge domination problem on bipartite permutation graphs. Discrete Applied Mathematics 87, 203–211 (1998)
22. Milanic, M.: Hereditary efficiently dominatable graphs. Journal of Graph Theory 73, 400–424 (2013)
23. van Rooij, J.M.M., Nederlof, J., van Dijk, T.C.: Inclusion/exclusion meets measure and conquer. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 554–565. Springer, Heidelberg (2009)
24. Wahlström, M.: A tighter bound for counting max-weight solutions to 2SAT instances. In: Grohe, M., Niedermeier, R. (eds.) IWPEC 2008. LNCS, vol. 5018, pp. 202–213. Springer, Heidelberg (2008)
25. Woeginger, G.J.: Exact algorithms for NP-hard problems: a survey. In: Combinatorial Optimization - Eureka, you Shrink!, pp. 185–207. Springer-Verlag New York, Inc., New York (2003)