Algorithms for finding minimum dominating set in a graph

Ernesto Parra Inza [1], Frank Angel Hernández Mira [2], Nodari Vakhania [1]* and José María Sigarreta Almira [2]

[1] Centro de Investigación en Ciencias UAEMor; Universidad Autónoma del Estado de Morelos, Av. Universidad 1001, Col.Chamilpa, post code 62209, Cuernavaca, Morelos, México.

[2] Facultad de Matemáticas UAGro; Universidad Autónoma de Guerrero, Carlos E. Adame No.54, Col.Garita, post code 39650, Acapulco, Guerrero, México.

∗ Corresponding author: nodari@uaem.mx

IOCA 2021

Abstract

In a simple connected graph $G = (V, E)$, a subset of vertices $S \subseteq V$ is a dominating set in graph G if any vertex $v \in V \setminus S$ is adjacent to some vertex x from this subset. It is known that this problem is NP-hard, and hence there exists no exact polynomial-time algorithm that finds an optimal solution to the problem. This work aims to present an exact enumeration and heuristic algorithm that can be used for large-scale real-life instances. Our exact enumeration algorithm begins with specially derived lower and upper bounds on the number of vertices in an optimal solution and carries out a binary search within the successively derived time intervals. The proposed heuristic accomplishes a kind of depth-first search combined with breadth-first search in a solution tree. The performance of the proposed algorithms is far better than that of the state-of-the-art ones. For example, our exact algorithm has solved optimally problem instances with order 300 in 165 seconds. This is a drastic breakthrough compared to the earlier known exact method that took 11036 seconds for the same problem instance. On average, over all the 100 tested problem instances, our enumeration algorithm is 168 times faster.

Keywords: graph; dominating set; enumeration algorithm; heuristic; time complexity.

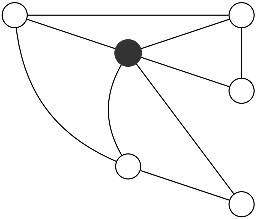
IOCA 2021

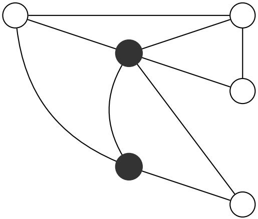
Outline

IOCA 2021

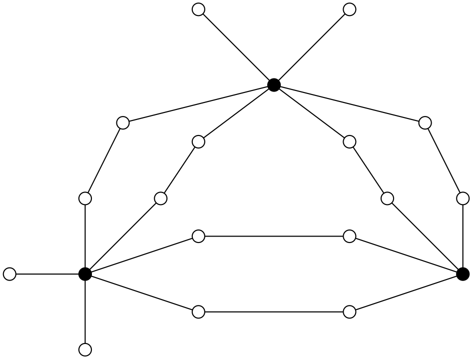
Introduction

In a simple connected graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges, a subset of vertices $S \subseteq V$ is a *dominating set* in graph G if any vertex $v \in V$ is *adjacent* to some vertex x from this subset (i.e., there is an edge $(v, x) \in E$) unless vertex v itself belongs to set S. Any subset S with this property will be referred to as a *feasible solution*, whereas any subset of vertices from set V will be referred as a *solution*. The number of vertices in a solution will be referred to as its *size (order)*. The **objective** is to find an *optimal solution*, a feasible solution with the minimum possible size $\gamma(G)$.





Since the problem is known to be *NP-hard*, there exists no exact algorithm that finds an optimal solution in polynomial time.



In this work, we aim to develop an exact implicit enumeration algorithm that can be used in real-life scenarios with graphs with a considerable number of vertices.

IOCA 2021

Implicit enumeration

Initially, we create a feasible solution using the approximation algorithm from [10]. This solution defines the initial upper bound U on the size of a feasible solution, whereas the initial lower bound L is obtained based on the following known results.

Theorem

[9] $\gamma(G) \geq \frac{n}{\Delta(G)+1}$.

Theorem

[9] $\gamma(G) \geq \frac{2r(G)}{3}$ *and* $\gamma(G) \geq \frac{d(G)+1}{3}$.

Theorem

[9] $|Supp(G)| \leq \gamma(G) \leq n - |Leaf(G)|$.

IOCA 2021

Implicit enumeration

The next corollary is an immediate consequence of the Theorems 1, 2 and 3.

Corollary

$L = \max\{\frac{n}{\Delta(G)+1}, \frac{2r(G)}{3}, \frac{d(G)+1}{3}, s\}$ *is a lower bound on the number of vertices in a minimum dominating set.*

IOCA 2021

Implicit enumeration

Procedure_Next(ν)

For each trial $\nu \in [L, U]$, the solutions of size at most ν are generated in a special priority order that is intended to help in a faster convergence to a feasible solution. Heuristic considerations are used to determine that order. *Procedure_Next*(ν) determines the (next) solution $\sigma_h(\nu)$ of size ν at iteration h. An auxiliary subroutine *Procedure_Priority_LIST*() generates a priority list of vertices which is used for the creation of solution $\sigma_h(\nu)$. While creating this list, the support and leaf vertices are ignored: by Theorem 3, for every iteration h, all vertices from set $Supp(G)$ can be included in solution $\sigma_h(\nu)$ and no vertex from set $l(G)$ is to be part of it.

Implicit enumeration

Algorithm 1 Algorithm_BDS

Input: A graph G.
Output: A $\gamma(G)$-*set* S.
$Supp(G)$:= Set of support vertex of graph G;
$l(G)$:= Set of leaf vertex of graph G;
$L := \max\{\frac{n}{\Delta(G)+1}, \frac{2r}{3}, \frac{d+1}{3}, |Supp|\}$;
S := Feasible solution
$U := |S|$;
$v := \lfloor (L+U)/2 \rfloor$;
$Procedure\_Priority\_LIST()$;
{ iterative step }
while $U - L > 1$ **do**
  if $Procedure\_Next(v)$ returns NIL **then**
    $L := v$;
    $v := \lfloor (L+U)/2 \rfloor$;
  else { A feasible solution was found $(\sigma_h(v))$}
    $U := v$;
    $v := \lfloor (L+U)/2 \rfloor$;
    $Procedure\_Priority\_LIST()$;
  end if
end while

IOCA 2021

Implicit enumeration

Procedure_Next(ν) verifies the feasibility of each solution $\sigma_h(\nu)$ generated.

Remark

The feasibility of every generated solution of a given size is verified in time $O(n)$.

Let $s = |Supp(G)|$ and $l = |I(G)|$. Now, with the above mentioned and Remark 1, we obtain the following lemma.

Lemma

The time complexity of Algorithm_BDS is

$$O\left(n\log(\frac{n}{2}-1)\binom{n}{n/4}\right).$$

A combined DFS and BFS search

Our second algorithm DBS (Depth Breadth Search), combines depth-first search with a breadth-first search in solution tree T, a binary tree of depth n, in which vertex v^i is associated with level i. The path from the root to a leaf uniquely defines a solution in that tree. With each solution, a binary number with n digits is naturally associated, with 0 entry in position i if vertex v^i does not pertain to that solution, and with the entry 1 otherwise. Every path in tree T from the root to a leaf represents a binary number of n digits and the corresponding solution. If the edge of this path at level i of the tree is marked as 0 then vertex v^i does not belong to that solution, and if that edge is marked as 1 then it belongs to the solution.

IOCA 2021

Given solution $\sigma = (v^1, v^2, \ldots, v^{U_0})$ obtained by the greedy algorithm from [10], we define an auxiliary parameter $\beta = \lfloor \alpha(U - s) \rfloor + s$, for $0 < \alpha < 1$, as the size of a base solution $\sigma(\beta)$ (U is the current upper bound, initially it is U_0, $s = |Supp(G)|$). A base solution is constructed by the procedure and serves as a basis for the construction of the following larger sized solutions sharing the β vertices with solution $\sigma(\beta)$. In case none of these extensions of solution $\sigma(\beta)$ turn out to be feasible, the current base solution is replaced by another base solution of size β and the search similarly continues.

The set of vertices in a base solution is determined according to one of the following alternative rules:

1. The first β vertices $(v^1, v^2, \ldots, v^\beta)$ from solution σ.
2. Randomly selected β vertices from solution σ.
3. Randomly selected β vertices from set $V \setminus I(G)$.

IOCA 2021

Each base solution is iteratively extended by one vertex per iteration and each of these extensions are checked for feasibility until either *(i)* one of them turns out to be feasible or *(ii)* an extension of size $U-1$ is created. In the latter case *(ii)* if the corresponding extension of size $U-1$ is not feasible, the next base solution with size β is constructed and the procedure is repeated for the newly created base solution. In the former case *(i)*, the current lower bound is updated; correspondingly, the parameter β is also updated, the first base solution of the new size β is created and the procedure is again repeated for this newly created base solution. Procedure DBS halts if the extensions of all the base solutions of the current size β were tested and none of them turned out to be feasible. Then $\gamma(G) = U$ and the procedure return the corresponding feasible solution of size U, which is minimal.

The definition of the upper bound U, lower bound L, and the fact that none of the solutions of size lower to $U - 1$ is feasible, immediately follows from the following remark.

Remark

The Procedure DBC returns a minimal dominating set.

Remark

If none extension of all base solutions of current size β is feasible and $\beta > L$, then $\beta < \gamma(G) \leq U$.

The Procedure DBS does not guarantee the optimal solution, but it allows to improve the solutions of [10]. Some computational experiments are discussed in Table 3.

Results and Discussion

We have implemented the algorithms in C++ using Visual Studio for Windows 10 (64 bits) on a personal computer with Intel Core i7-9750H (2.6 GHz) and 12 GB of RAM DDR4. The order and the size of an instance were generated randomly utilization function *random()*. To complete the set E(G), each new edge was added in between two yet non-adjacent vertices randomly until the corresponding size was attained. The results for the instances are shown in Table 1. We can observe a significant difference in the time of the algorithms tested. We have obtained that for 100% of the analyzed instances, with a density greater or equal to 0.5, $Time(BDS) \approx \frac{1}{168} Time(MSC)$. The $Time(A)$ function returns the time in seconds that it takes for algorithm A to give a response.

IOCA 2021

Table: Graphs with density ≈ 0.5.

No.	$\|V(G)\|$	$\|E(G)\|$	Time BDS (s)	Time MSC (s)	Lower Bounds $\frac{n}{\Delta(G)+1}$	$\frac{d+1}{3}$	$\frac{2r}{3}$	$\|Supp(G)\|$	$\gamma(G)$	Upper Bounds $\|S\|$	$n-\Delta(G)$
1	201	8081	31.0313	1136.67	2	1	1	2	5	5	103
2	207	8569	35.9424	1361.77	2	1	1	1	5	6	105
3	209	8735	37.3502	1454.44	1	2	1	1	5	5	104
4	215	9243	42.9108	1810.09	1	1	1	1	5	5	103
5	217	9415	42.881	1892.35	1	1	1	3	5	5	107
6	221	9765	49.91	2011.15	2	1	1	1	6	7	115
7	226	10211	51.2046	2278.89	2	1	1	1	5	5	114
8	230	10575	4.6399	2537.6	1	2	1	4	5	5	111
9	233	10852	57.8398	2793.59	2	1	1	1	5	6	118
10	238	11322	65.053	3093.64	2	1	1	1	5	6	123
11	242	11705	67.0996	3463.78	1	1	1	2	5	5	119
12	250	12491	83.1936	4065.44	1	2	1	1	5	5	125
13	257	13199	93.983	4919.85	2	1	1	1	5	6	131
14	264	13927	112.518	5925.73	2	1	1	2	5	5	135
15	269	14459	112.609	6102.11	1	1	1	1	5	6	134
16	275	15111	114.446	6887.98	2	1	1	1	5	6	144
17	283	16002	418.72	8040.4	1	1	1	1	5	6	141
18	290	16803	148.712	9215.59	2	1	1	1	5	5	152
19	296	17505	155.861	10269.4	2	1	1	1	5	6	152
20	300	17981	165.301	11035.8	2	1	1	1	5	5	151

IOCA 2021

When analyzing graphs with a density of approximately 0.2, the execution times of the analyzed algorithms behave differently from the cases analyzed previously. In low-density instances, the MSC algorithm is faster than the algorithm proposed in this paper. The results of the experiments, with this type instance, can be seen in Table 2.

IOCA 2021

Table: Graphs with density ≈ 0.2.

No.	$\|V(G)\|$	$\|E(G)\|$	Time BDS (s)	Time MSC (s)	Lower Bounds				$\gamma(G)$	Upper Bounds	
					$\frac{n}{\Delta(G)+1}$	$\frac{d+1}{3}$	$\frac{2r}{3}$	$\|Supp(G)\|$		$\|S\|$	$n-\Delta(G)$
1	50	286	1.89587	0.616285	2	1	1	2	6	6	33
2	60	357	3.13747	0.715912	2	1	1	1	6	7	40
3	70	524	7.04679	1.46234	2	1	1	1	6	6	45
4	80	678	12.5997	2.81906	2	1	1	1	6	7	53
5	90	842	390.903	5.18101	3	1	1	1	7	8	63
6	100	1031	803.741	8.31738	2	1	1	2	7	7	67
7	101	1051	804.208	8.69229	3	1	1	1	7	7	70
8	106	1154	1143.92	10.8275	2	1	1	1	7	8	71
9	113	1306	1594.92	16.2542	3	1	1	1	7	7	77
10	117	1398	3364.4	19.955	3	1	1	2	8	9	84
11	121	1493	2240.49	23.1156	3	1	1	1	7	7	87

IOCA 2021

The computational experiments with the Procedure DBS showed that in 98.62% of the analyzed instances the solution given by [10] was improved. Table 3 shows some of these results.

Table: Results Procedure DBS.

No.	$\|V(G)\|$	$\|E(G)\|$	$\|S\|$	Solution 1				Solution 2			
				β	$\sigma_i(\beta)$ generates	Time(s)	$\|DS\|$	β	$\sigma_i(\beta)$ generates	Time(s)	$\|DS\|$
1	600	84557	12	4	43	37.815	11				
2	610	87490	12	4	3225	2876.67	11				
3	620	90472	13	3	21	20.953	12	3	6	25.745	11
4	630	93505	13	4	107	90.532	12	3	35266	29750	11
5	640	96587	13	4	22	23.207	11				
6	650	102571	9	2	No solution found						
7	660	105798	10	3	4080	4635.46	8				
8	670	109076	9	2	18	24.966	8				
9	680	109417	12	4	65	86.754	11				
10	690	117488	10	3	812	1055.51	9				
11	700	116132	13	4	39	55.176	12				
12	710	120941	10	3	12	21.208	9				
13	720	127996	9	2	11299	15411.1	8				
14	730	131598	9	2	25142	37801.4	8				
15	740	130162	13	4	52	72.597	12	3	31056	40681.6	11
16	750	137096	10	3	4498	6453.14	9				
17	760	138953	10	3	9	18.662	9				
18	770	141210	13	4	13	24.707	12	3	9561	16496.5	11

Conclusions

We proposed an exact branch and bound and an approximation heuristic algorithms for the domination problem in general graphs which outperform the state-of-the-art exact and approximation, respectively, algorithms from Van Rooij et al. [13] and Hernández et al. [10], respectively. The first proposed exact Binary Domination Search algorithm combines upper and lower bounds with binary search. The initial lower bound is obtained directly from the earlier known properties and the initial upper bound is obtained by the earlier known best heuristic algorithm for the problem. The practical behavior of the algorithm was tested on a considerable number of the randomly generated problem instances with a size up to 300. On random instances with graphs with an average density of 0.5 algorithms from Van Rooij et al. [13] delayed 168 times more than the Binary Domination Search algorithm. The approximate Depth Breadth Search heuristic combine depth-first search with breadth-first search and was able to improve solutions delivered by the earlier known state-of-the-art algorithm (Hernández et al. [10]) in 98.62% of the tested instances.

IOCA 2021

Author Contributions: The authors contributed equally to this research. Investigation, F.A.H.M., E.P.I., J.M.S.A. and N.V.; writing—review and editing, F.A.H.M., E.P.I., J.M.S.A. and N.V. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by SEP PRODEP publication grant. The fourth author was supported by SEP PRODEP 511/6 grant and CONACyT 2020-000019-01NACV-00008 grant.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References I

Abel Cabrera Martínez, Juan Carlos Hernández-Gómez, E Parra Inza, and José M Sigarreta.
On the total outer k-independent domination number of graphs.
Mathematics, 8(2):194, 2020.

Alina Campan, Traian Marius Truta, and Matthew Beckerich.
Fast dominating set algorithms for social networks.
In *MAICS*, pages 55–62, 2015.

Vasek Chvatal.
A greedy heuristic for the set-covering problem.
Mathematics of operations research, 4(3):233–235, 1979.

Stephen Eubank, VS Anil Kumar, Madhav V Marathe, Aravind Srinivasan, and Nan Wang.
Structural and algorithmic aspects of massive social networks.
In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 718–727, 2004.

Klaus-Tycho Foerster.
Approximating fault-tolerant domination in general graphs.
In *2013 Proceedings of the Tenth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 25–32. SIAM, 2013.

Fedor V Fomin, Fabrizio Grandoni, and Dieter Kratsch.
A measure & conquer approach for the analysis of exact algorithms.
Journal of the ACM (JACM), 56(5):1–32, 2009.

Serge Gaspers, Dieter Kratsch, Mathieu Liedloff, and Ioan Todinca.
Exponential time algorithms for the minimum dominating set problem on some graph classes.
ACM Transactions on Algorithms (TALG), 6(1):1–21, 2009.

IOCA 2021

References II

Matt Gibson and Imran A Pirwani.
Approximation algorithms for dominating set in disk graphs.
arXiv preprint arXiv:1004.3320, 2010.

TeresaW Haynes.
Domination in Graphs: Volume 2: Advanced Topics.
Routledge, 2017.

Frank Angel. Hernández Mira, Ernesto Parra Inza, Jose María Sigarreta Almira, and N. Vakhania.
A polynomial-time approximation to a minimum dominating set in a graph.
Theoretical Computer Science. A submitted manuscript.

Harry B Hunt III, Madhav V Marathe, Venkatesh Radhakrishnan, Shankar S Ravi, Daniel J Rosenkrantz, and Richard E Stearns.
Nc-approximation schemes for np-and pspace-hard problems for geometric graphs.
Journal of algorithms, 26(2):238–274, 1998.

Anupriya Jha, Dinabandhu Pradhan, and S Banerjee.
The secure domination problem in cographs.
Information Processing Letters, 145:30–38, 2019.

Johan MM Van Rooij and Hans L Bodlaender.
Exact algorithms for dominating set.
Discrete Applied Mathematics, 159(17):2147–2164, 2011.

Kexiang Xu, Xia Li, and Sandi Klavžar.
On graphs with largest possible game domination number.
Discrete Mathematics, 341(6):1768–1777, 2018.

IOCA 2021