Solving the Minimum Independent Domination Set Problem in Graphs by Exact Algorithm and Greedy Heuristic: Experimental Approach

Christian Laforest and Raksmey Phan

LIMOS, CNRS UMR 6158 – Université Blaise Pascal
24 avenue des Landais
63173 Aubière Cedex, France
{laforet,phan}@isima.fr
http://limos.isima.fr/

1 Introduction

In this paper we consider an undirected, unweighted finite graph \( G = (V, E) \) where \( V \) is the set of its vertices and \( E \) is the set of its edges. If \( uv \) is an edge between vertices \( u \) and \( v \) then \( u \) and \( v \) are adjacent or neighbors. A dominating set in a graph \( G = (V, E) \) is a subset \( D \subseteq V \) such that every vertex \( v \in V \setminus D \) is neighbor of at least one vertex in \( D \). An independent set in \( G \) is a subset \( I \) of vertices such that no two of them are adjacent. The minimum Independent Dominating Set (mIDS) problem is to find an independent dominating set of minimum size in a given graph \( G \). It is known to be NP-hard and not possible, under usual complexity hypotheses, to approximate within better than \( n^{1-\epsilon} \) for any positive \( \epsilon \) (where \( n \) is the number of vertices).

The problem has been studied on various specific graph families (see [3]). For general graphs, [4] developed a smart \( O^*(\sqrt{3}^{|V|}) \) time algorithm to solve this problem. [1] devised a branching algorithm that finds the mIDS in \( O^*(2^{0.424|V|}) \) running time. All these works are theoretical since the complexity is evaluated in worst-case mode and the algorithms have not been implemented and experimented by their authors. In 2011, [5] has computed and compared the algorithm of [4] with their Intelligent Enumeration Algorithm.

In [3] we have presented our new approach to solve the mIDS problem and we have proved that our algorithm tends to have a complexity better than any other known algorithm regarding the problem when the graph is large and dense. Here, we give results of our experimentations and we test a greedy heuristic to compute an independent dominating solution on large graphs (up to 65536 vertices). Note that for future comparisons, our instances and our code are available online at: “http://todo.lamsade.dauphine.fr/spip.php?article42”.

* This work is supported by the French Agency for Research under the DEFIS program TODO, ANR-09-EMER-010.
2 Computational Evaluation: Exact Algorithm by Clique Partition For The mIDS

We present computational results of our exact method described in [3] on various graphs. Our software is implemented in C language. We have evaluated it on hundreds of different graphs (random graph, tree graph, hypercube graph, . . . ). Each instance is executed 10 times on an AMD Opteron Processor 2352 clocked at 2.1GHz, with different seeds to have average values.

We use the classical random graphs model. We have tested on graphs from 80 to 149 vertices with probabilities \( p = \{0.1, 0.2, \ldots, 0.9\} \). Then we have observed that our software can solve large graphs in reasonable running time (less than 300s) when \( p \) is larger than 0.3. However for sparse graphs (\( p = 0.1 \)) the running time rises very fast from 740s for 80 vertices to more than one day for 110 vertices. This is why we then tested it on random tree graphs. We are able to solve instances up to 75 vertices in less than 15 minutes. We also can note that the Greedy Heuristic is very efficient here: Its size gets very close to the optimal in quasi-null time. We have also evaluated our software on hypercube graphs: For any integer \( d \), a hypercube is a graph \( G = (V, E) \) with \( |V| = 2^d \) and \( |E| = 2^{d-1}d \). Our method can solve a large hypercube graphs (of 64 vertices) in reasonable running time (approximately 7 minutes).

In order to compare our method with the experimentations of [5], we have evaluated our method on several grid graphs from \( 5 \times 5 \) to \( 8 \times 8 \). Table 1 summarizes the comparison. We give their running time in seconds. Clearly our method is better on finding \( mIDS \) than the two others. For 49 vertices, our algorithm takes 9.90s running time while IEA takes more than one day.

<table>
<thead>
<tr>
<th>n</th>
<th>25</th>
<th>36</th>
<th>49</th>
<th>56</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our algorithm</td>
<td>0.00</td>
<td>0.00</td>
<td>9.90</td>
<td>630.00</td>
</tr>
<tr>
<td>IEA</td>
<td>1.00</td>
<td>254.00</td>
<td>141242.00</td>
<td>—</td>
</tr>
<tr>
<td>Liu and Song Algorithm</td>
<td>11.00</td>
<td>39225.00</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

3 Computational Evaluation: Greedy Approach For The mIDS

In order to try to treat graphs of large size we further analyse the computational results of Greedy Heuristic (GH). Because of the size of graphs it is not possible to have the optimal solution, so we compare our solution to a lower bound [3]. We have implemented our algorithms in the same protocol as previously. We have evaluated several large graphs from 8 vertices to 65536 vertices.
As it was said previously GH is very fast on all of our experimentations. On several 10000-vertices random graphs, the running times do not exceed 1s. As expected, size of solutions rises when probability is small. On random tree graphs (from 10000 to 40000 vertices), for each execution the running time is less than 20s and we notice that GH gives solutions close (up to a factor at most 2) to lower bound. On hypercube graphs, with [2] we can compare our methods with the optimal solutions. Tests show that GH finds 7 of 8 optimal solutions. The running times do not exceed 11s for the largest graph.

With these tests, we confirm that Greedy Heuristic is very fast: All our executions take less than 20s of computational time. Experiences show that solutions produced by Greedy Heuristic do not exceed 5 times the lower bound which means they are within a factor at most 5 of an optimal solution. After several hundred tests, Figure 1 summarizes our results. For each family of graphs we calculated the mean-ratio of solutions, the maximum ratio of solutions and the standard deviation. As it is shown Greedy Heuristic seems to be very efficient (ratio close to 1). Moreover the standard deviations are small thus the given solutions are close to eachother.

![Image](image.png)

Fig. 1. Experimental Approximation Ratio.

References