Some new recursive algorithms and heuristics for the maximum clique problem

Dragan Stevanović
FILOZOFSKI FAKULTET, NIŠ, Čamojevića 10, e-mail: dragance@ban.unis.ni.ac.yu

1. INTRODUCTION

Throughout this paper $G = (V, E)$ is an arbitrary undirected graph without loops and multiple edges. The set $V$ is the vertex set of $G$, and $E \subseteq V \times V$ is the edge set of $G$. For $S \subseteq V$ the graph $G(S) = (S, E \cap S \times S)$ is called the subgraph induced by $S$. By $G[S]$ we denote the subgraph induced by $V[S]$. For the induced subgraph $G'[V', E']$, the set $N_G(S) = \{v \in V' : S | (3s \in S (v, s) \in E')\}$ is called the neighborhood of $S$ in $G'$. If $S = \{s\}$, we will write $N_G(s)$ instead of $N_G(\{s\})$.

The subset $C \subseteq V$ such that all the vertices from $C$ are pairwise adjacent, i.e. $\forall i, j \in C, (i, j) \in E$ is called the clique of $G$. The subset $C \subseteq V$ such that $\forall i, j \in C, (i, j) \in E$ is called the stable set of vertices of $G$. The clique (stable set) is maximal if it is not proper subset of another clique (stable set), and it is maximum if it has the greatest cardinality among all cliques (stable sets) of $G$. Cardinality of the maximum clique is called the clique number of $G$, denoted by $\omega(G)$.

The maximum clique problem asks for a clique of maximum cardinality. This problem is NP-complete, as is shown in [3]. More over, there is no polynomial time algorithm that can approximate the maximum clique within a factor of $n^{\epsilon}$ ($\epsilon > 0$), unless $P = NP$ (see [1]).

Despite these negative results, the maximum clique problem is important and has many practical applications in different areas, including coding theory, computer vision, aligning DNA and protein sequences, economics and information retrieval for example.

For a very good survey on the algorithms and heuristics proposed so far for the maximum clique problem, see [4].

2. ALGORITHM

We shall base our algorithm on the following obvious

Theorem 1. For any subset $S \subseteq V$ of vertices of graph $G$ the following relation holds

\[ \omega(G) = \max(1 + \omega(G(N_G(s))))_{s \in S}, \omega(G - S)). \]

The branch-and-bound algorithm selects a set of vertices $S$, check for the maximum clique recursively in the neighborhood of every vertex from $S$, and then recursively in the graph $G - S$. For the set $S$ we take a
maximal stable set of G, since for every s ∈ S holds |N(s)| ≤ |V |− |S|, and the vertex s is no further considered in any other subproblem.

The actual algorithm, written in pseudo-code, is given in Figure 1.

1. procedure hreCLIQUE (K, V )
2. begin
3. if V = ∅ then Best ← K
4. else
5. begin
6. find a maximal stable set S in G(V )
7. for s in S do
8. if |K| + 1 + |N_r(s)| > |Best| then
9. hreCLIQUE (K ∪ {s}, N_r(s))
10. if |K| + 1 + |V |− |S| > |Best| then
11. hreCLIQUE (K, V |− S)
12. end
13. end
14. begin
15. Best ← ∅
16. hreCLIQUE (∅, V )
17. end

Figure 1. The maximum clique algorithm

Here, in a call to procedure hreCLIQUE the set K denotes the clique that has been already "accumulated", and the set V' is the set of possible candidates for the inclusion in it. The set Best denotes the maximum clique found so far. The stable set S is lexicographically smallest in the graph G(V'). The upper bound for the size of the maximum clique in the graph induced by the set of candidate vertices is rather poor. Better one can be obtained by using some fast and relatively good coloring heuristic. For this purpose we have used well-known heuristic DSATUR (degree of saturation largest first) proposed by Brelaz [2].

Computational results of this algorithm (w/out use of DSATUR) on some classes of graphs are reported in Section 4.

3. HEURISTIC

The most time-consuming task in the procedure hreCLIQUE is the recursive call in the line 11 of Figure 1. Besides that, empirical results show that maximum cliques are found almost at the beginning of the execution of the algorithm. This is the motivation for considering heuristic shown in Figure 2 obtained by the exclusion of the lines 10 and 11 from the algorithm in Figure 1.

We shall now give the sufficient (but not necessary) condition for this heuristic to find the maximum clique of G.

A set S ⊆ V is called clique-intersecting in G if there exists a maximum clique M of G for which S ∩ M ≠ ∅. It is easy to show that the above heuristic will find the maximum clique of G if in each call of the procedure hreCLIQUE the maximal stable set S found in line 6 (see Figure 2) is clique-intersecting in G(V ).

1. procedure heuCLIQUE (K, V )
2. begin
3. if V = ∅ then Best ← K
4. else
5. begin
6. find a maximal stable set S in G(V )
7. for s in S do
8. if |K| + 1 + |N_r(s)| > |Best| then
9. heuCLIQUE (K ∪ {s}, N_r(s))
10. end
11. end
12. begin
13. Best ← ∅
14. heuCLIQUE (∅, V )
15. end

Figure 2. The maximum clique heuristic

The main result of this section is the following

Theorem 2. The graph G has the property that in each induced subgraph G' of G each maximal stable set is clique-intersecting if and only if no induced subgraph of G is isomorphic to either of the graphs G_1, G_2 shown in the Figure 3.

Figure 3. The forbidden subgraphs

Proof: If G contains either G_1 and G_2, then the maximal stable set formed by "white" vertices in these graphs provides a counterexample.

Therefore, we may suppose that there exists induced subgraph G'=(V', E') of G and its maximal stable set S such that for each maximum clique M of G' holds
Let $M$ be the particular maximum clique of $G'$, and let $T \subseteq S$ be the minimal set of vertices for which $M \subseteq N_G(T)$ (such set exists since $V \subseteq N_G(S)$, because of the maximality of $S$). For any vertex $a \in S$ it must hold $|M \cap N_G(a)| \geq 2$, otherwise $\{a\} \cup (M \cap N_G(a))$ is the maximum clique having non-empty intersection with $S$. Therefore, it must be $|T| \geq 2$. For a vertex $b \in T$ there must exist a vertex $b' \in M$ such that $b' \in N_G(b) \cap N_G(T\{b\})$, otherwise $M \subseteq N_G(T\{b\})$ and the set $T$ is not minimal, contradicting its choice.

If $T = \{a, b\}$ then there must exist vertices $a_1$ and $a_2$ in $M$ such that $a_1, a_2 \in N_G(a)$, and then $a_1, a_2 \in N_G(b)$. Similarly, there exist vertices $b_1$ and $b_2$ in $M$ such that $b_1, b_2 \in N_G(b)$ and $b_1, b_2 \in N_G(a)$. The subgraph induced by the vertices $a, a_1, a_2, b, b_1, b_2$ is isomorphic to $G_1$.

If $|T| \geq 3$, we shall first note that for any vertex $v \in T$ there exists a vertex $v' \in M$ such that $v' \in N_G(v)$ and $v' \in N_G(T\{v\})$, otherwise $M \subseteq N_G(T\{v\})$ giving contradiction. Then for any three vertices $a, b, c$ of $T$ there exist vertices $a', b', c' \in M$ such that $a'$ is adjacent to $a$ and not to $b$ and $c$, and similar for the vertices $b'$ and $c'$. Then the subgraph induced by the vertices $a, a', b, b', c, c'$ is isomorphic to $G_2$.

4. COMPUTATIONAL RESULTS

The computations, shown in Table 1 below, were done on the instances of the following graph classes:

- classic uniform random graphs RND $(n, p)$, where $n$ is the number of nodes, and $p$ is the probability of the existence of an edge between two arbitrary vertices of the graph.
- P.hat $(n, a, b)$ is a generalization of the above. Here $n$ is the number of nodes, the parameters $a, b$ satisfy $0 \leq a \leq b \leq 1$, and an edge between two vertices of the graph exists with probability between $a$ and $b$. For the details, see [5].
- Hamming graph Ham $(n, d)$ arises from the coding theory. It has a node for each binary vector of length $n$. Two nodes are adjacent if and only if the corresponding bit vectors are hamming distance at least $d$ apart. A clique then represents a feasible set of vectors for a code.

- The Johnson graph $Joh(n, w, d)$ is similar to Ham$(n, d)$ except that it has a node for every binary vector of length $n$ with exactly $w$ 1's.

If the entry is not shown in the Table 1, then it required at least 600 seconds, at which point the run was typically killed.

From the Table 1 we see that for random graphs (and also P.hat) using hreCLIQUE (heuCLIQUE) with poor upper bounds gives smaller execution times than using DSATUR despite the smaller number of subproblems that need to be examined, since the bounding procedure has to be called many times during the execution. On the other hand, use of hreCLIQUE with DSATUR is faster for Hamming and Johnson graphs. The interesting observation is that for Hamming and Johnson graphs heuCLIQUE finds a maximum clique, while it is not known at present whether these graphs satisfy the conditions of Theorem 2.

ACKNOWLEDGEMENT

The computer used for computations was Silicon Graphics Indy, generously provided by Ei SiGraf.

REFERENCES


<table>
<thead>
<tr>
<th></th>
<th>Clique Size</th>
<th>heuCLIQUE</th>
<th></th>
<th>Clique Size</th>
<th>heuCLIQUE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
<td>Time</td>
<td></td>
<td>Time</td>
<td>Time</td>
<td></td>
</tr>
<tr>
<td></td>
<td>without</td>
<td>with</td>
<td></td>
<td>without</td>
<td>with</td>
<td></td>
</tr>
<tr>
<td></td>
<td>DSATUR</td>
<td>DSATUR</td>
<td></td>
<td>DSATUR</td>
<td>DSATUR</td>
<td></td>
</tr>
<tr>
<td>RND (100, 0.1)</td>
<td>4</td>
<td>0.01</td>
<td>0.02</td>
<td>4</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>RND (100, 0.2)</td>
<td>5</td>
<td>0.01</td>
<td>0.04</td>
<td>5</td>
<td>0.01</td>
<td>0.01</td>
</tr>
<tr>
<td>RND (100, 0.3)</td>
<td>6</td>
<td>0.02</td>
<td>0.10</td>
<td>6</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td>RND (100, 0.4)</td>
<td>7</td>
<td>0.04</td>
<td>0.24</td>
<td>7</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td>RND (100, 0.5)</td>
<td>9</td>
<td>0.17</td>
<td>0.76</td>
<td>9</td>
<td>0.01</td>
<td>0.06</td>
</tr>
<tr>
<td>RND (100, 0.6)</td>
<td>12</td>
<td>0.57</td>
<td>1.89</td>
<td>12</td>
<td>0.01</td>
<td>0.12</td>
</tr>
<tr>
<td>RND (100, 0.7)</td>
<td>14</td>
<td>4.23</td>
<td>8.99</td>
<td>14</td>
<td>0.03</td>
<td>0.28</td>
</tr>
<tr>
<td>RND (100, 0.8)</td>
<td>20</td>
<td>46.71</td>
<td>35.92</td>
<td>18</td>
<td>0.08</td>
<td>0.58</td>
</tr>
<tr>
<td>RND (100, 0.9)</td>
<td>29</td>
<td>1.72</td>
<td>2.60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RND (200, 0.1)</td>
<td>4</td>
<td>0.02</td>
<td>0.11</td>
<td>4</td>
<td>0.01</td>
<td>0.02</td>
</tr>
<tr>
<td>RND (200, 0.2)</td>
<td>6</td>
<td>0.05</td>
<td>0.29</td>
<td>6</td>
<td>0.01</td>
<td>0.04</td>
</tr>
<tr>
<td>RND (200, 0.3)</td>
<td>7</td>
<td>0.16</td>
<td>1.02</td>
<td>7</td>
<td>0.01</td>
<td>0.07</td>
</tr>
<tr>
<td>RND (200, 0.4)</td>
<td>9</td>
<td>0.72</td>
<td>3.58</td>
<td>9</td>
<td>0.02</td>
<td>0.21</td>
</tr>
<tr>
<td>RND (200, 0.5)</td>
<td>11</td>
<td>3.92</td>
<td>15.01</td>
<td>10</td>
<td>0.04</td>
<td>0.44</td>
</tr>
<tr>
<td>RND (200, 0.6)</td>
<td>14</td>
<td>32.12</td>
<td>75.26</td>
<td>14</td>
<td>0.09</td>
<td>0.82</td>
</tr>
<tr>
<td>RND (200, 0.7)</td>
<td>16</td>
<td>0.64</td>
<td>5.44</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RND (200, 0.8)</td>
<td>24</td>
<td>4.37</td>
<td>24.77</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P_hat (300, 0.0, 0.5)</td>
<td>8</td>
<td>0.46</td>
<td>2.71</td>
<td>7</td>
<td>0.02</td>
<td>0.18</td>
</tr>
<tr>
<td>P_hat (300, 0.25, 0.75)</td>
<td>25</td>
<td>188.48</td>
<td>19</td>
<td>0.22</td>
<td>1.39</td>
<td></td>
</tr>
<tr>
<td>P_hat (500, 0.0, 0.5)</td>
<td>9</td>
<td>4.77</td>
<td>25.40</td>
<td>9</td>
<td>0.07</td>
<td>0.68</td>
</tr>
<tr>
<td>P_hat (500, 0.25, 0.75)</td>
<td>29</td>
<td>3.44</td>
<td>13.84</td>
<td>9</td>
<td>0.15</td>
<td>1.47</td>
</tr>
<tr>
<td>P_hat (700, 0.0, 0.5)</td>
<td>11</td>
<td>19.83</td>
<td>95.35</td>
<td>9</td>
<td>0.40</td>
<td>4.13</td>
</tr>
<tr>
<td>P_hat (1000, 0.0, 0.5)</td>
<td>10</td>
<td>106.12</td>
<td>545.71</td>
<td>9</td>
<td>0.40</td>
<td>4.13</td>
</tr>
<tr>
<td>Ham (6, 2)</td>
<td>28</td>
<td>5.37</td>
<td>0.13</td>
<td>28</td>
<td>0.02</td>
<td>0.08</td>
</tr>
<tr>
<td>Ham (8, 2)</td>
<td>112</td>
<td>10.02</td>
<td>112</td>
<td>6.41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ham (8, 4)</td>
<td>16</td>
<td>425.83</td>
<td>140.43</td>
<td>16</td>
<td>3.22</td>
<td>9.42</td>
</tr>
<tr>
<td>Joh (8, 4, 4)</td>
<td>14</td>
<td>1.12</td>
<td>0.61</td>
<td>14</td>
<td>0.03</td>
<td>0.18</td>
</tr>
<tr>
<td>Joh (16, 2, 4)</td>
<td>8</td>
<td>76.00</td>
<td>444.33</td>
<td>8</td>
<td>0.02</td>
<td>0.30</td>
</tr>
</tbody>
</table>

Table 1. The execution times for the problem instances